به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

یادگیری عمیق با کاربردها

تمرین شماره سه

نام و نام خانوادگی	امید نائیج نژاد
شماره دانشجویی	610301189
تاریخ ارسال گزارش	1404/3/8

فهرست مطالب

فهرست شکل ها

فهرست جدول ها

مدل شرح تصویر پایه

سوال اول

مجموعه داده Flickr8k شامل 8092 تصویر منتخب است .هر تصویر با پنج توضیح متنی (caption) متفاوت همراه است. این توضیحات توسط انسان‌ها و به دقت نوشته شده‌اند و جزئیات مربوط به اشیاء و رویدادهای برجسته در تصویر را بیان می‌کنند. این تنوع در توضیحات متنی بسیار ارزشمند است، زیرا نشان می‌دهد یک تصویر می‌تواند به روش‌های مختلفی توصیف شود. در شکل زیر 5 تصویر از دیتاست به همراه یکی از کپشن هایشان بصورت تصادفی انتخاب شده است و قابل رویت است.

شکل 1 - 5 تصویر تصادفی از دیتاست به همراه شرح آن ها

سوال دوم

توکنایزر (Tokenizer) نقش حیاتی در تسک توصیف تصویر، برای بخش "تولید متن (decoder) "مدل ایفا می‌کند و دلایل و ضرورت‌های آن به شرح زیر است:

1. تجزیه متن به واحدهای قابل مدیریت**(Tokens)** و با معنا:

- ماشین‌ها و مدل‌های یادگیری عمیق نمی‌توانند مستقیماً با رشته‌های متنی (مانند یک جمله کامل) کار کنند. آن‌ها به ورودی‌های عددی نیاز دارند.
- توکنایزر مسئول شکستن یک جمله یا پاراگراف به واحدهای کوچکتر و معنی‌دار به نام "توکن (Token) "است. این توکن‌ها معمولاً کلمات، اما می‌توانند کاراکترها یا زیرکلمات (subwords) باشند.
- مثال :جمله "A dog is playing in the park." می‌تواند به توکن‌های ["A", "dog", "is", "playing", "in", "the", "park", "."]تبدیل شود.

2. ایجاد واژه‌نامه **(Vocabulary)** و نگاشت به اعداد:

- پس از توکن‌سازی، توکنایزر یک واژه‌نامه از تمام توکن‌های منحصر به فرد موجود در مجموعه داده متنی (در اینجا، تمام کپشن‌های موجود در Flickr8k) ایجاد می‌کند. و به هر توکن منحصر به فرد یک (ID) اختصاص می‌دهد. این نگاشت از متن به عدد ضروری است تا مدل بتواند داده‌های متنی را پردازش کند.

3. مدیریت کلمات خارج از واژه‌نامه :

- معمولاً واژه‌نامه به کلمات پرکاربرد محدود می‌شود تا اندازه مدل کنترل شود. توکنایزر می‌تواند کلمات ناشناخته (که در واژه‌نامه نیستند) را با یک توکن خاص (مانند <unk>) جایگزین کند. این کار به مدل کمک می‌کند تا با کلمات جدیدی که در زمان آموزش ندیده است، به شکلی منطقی برخورد کند.

4. افزودن توکن‌های خاص:

- برای مدل‌های توالی‌به‌توالی، توکنایزر توکن‌های خاصی مانند :
  - **<start>** : برای نشان دادن شروع یک کپشن. این توکن به مدل می‌گوید که زمان شروع تولید متن است.
  - **<end>** : برای نشان دادن پایان یک کپشن. وقتی مدل این توکن را تولید می‌کند، به معنای اتمام تولید کپشن است.
  - **<pad>** : برای یکسان‌سازی طول دنباله‌ها. از آنجایی که طول کپشن‌ها متفاوت است، توکنایزر با افزودن توکن‌های پد به انتهای دنباله‌های کوتاه‌تر، طول همه دنباله‌ها را به یک مقدار ثابت (معمولاً طول بلندترین کپشن) می‌رساند. این برای پردازش دسته‌ای (Batch Processing) توسط مدل ضروری است.

سوال سوم و چهارم

در توضیحات بالا، به این موضوعات پرداخته شد.

سوال پنجم

5 تصویر قبلی بعد از اعمال توکنایزر روی توصیفشان قابل مشاهده است:

شکل 2 - 5 تصویر تصادفی از دیتاست به همراه توکنایزشده شرح آن ها

سوال ششم

روند رسیدن به **DataLoader** :

1. دانلود و استخراج دیتاست: ابتدا تصاویر و فایل کپشن (متنی) را دانلود و استخراج می‌کنیم.
2. پیش‌پردازش کپشن‌ها و ساخت واژگان (Vocabulary)
   - پاک‌سازی متن‌ها (تبدیل به حروف کوچک، حذف علائم نگارشی، توکنایز کردن).
   - افزودن توکن‌های خاص:
     <START>, <END>, <PAD>, <UNK>
   - تعیین حداکثر طول کپشن (20 کلمه)
   - ساخت mapping بین کلمات و اندیس‌ها
3. تعریف Dataset سفارشی: یک کلاس از torch.utils.data.Dataset می‌سازیم که:
   - یک تصویر را از دیسک لود و normalize می‌کند.
   - کپشن مربوطه را به لیست توکن‌ها تبدیل کرده و آن را به یک دنباله از اندیس‌ها تبدیل می‌کند.

4. تعریف collate برای batch کردن

چون طول کپشن‌ها متفاوت است، باید هنگامbatch سازی آنها را **pad**کنیم تا همه‌ی کپشن‌ها طول یکسانی داشته باشند . این کار توسط پاس دادن MyCollate به torch.utils.dataloader انجام میشود.

5. در نهایت DataLoader ساخته میشود.

توسعه معماری پایه

سوال هفتم

لایه تعبیه (Embedding Layer) در شبکه‌های عصبی بازگشتی (Recurrent Neural Networks - RNNs) نقش حیاتی دارد چون با استفاده از آن، دنباله ورودی داده شده به نمایش برداری پیوسته و معنی‌دار تبدیل میشوند. تا بتوانیم اطلاعات معنایی و نحوی را به مدل منتقل کنیم.

راه حل ساده تر برای پردازش این داده ها، استفاده از One Hot Coding هست بدین صورت که یک بردار به طول کلمات موجود در واژگان در نظر میگیریم که هر ایندکس آن متعلق به یک کلمه خاص هست و زمانی مساوی یک خواهد بود که بخواهیم آن کلمه را نشان بدهیم، اما بدلیل مصرف زیاد حافظه و مهمتر از آن عدم توجه به شباهت یا تفاوت معنایی کلمات (فاصله هر دو کلمه از هم برابر است!) اصلا روش مناسبی به شمار نمی رود و بدین منظور از لایه تعبیه که قابل آموزش هست استفاده میشود.

سوال هشتم

بنظر من در تسک توصیف تصاویر، آموزش لایه تعبیه در کنار تمام معماری بهتر است؛ چون در این روش، وزن‌های لایه تعبیه به صورت تصادفی مقداردهی اولیه می‌شوند و در طول فرآیند آموزش مدل، بهینه‌سازی می‌شوند.

بدیهی است که در کنار آموزش بهتر لایه تعبیه، هزینه آموزش مدل بالاتر خواهد رفت و نیاز به داده بیشتری برای عملکرد بهتر خواهیم داشت.

در ضمن، بدلیل اینکه در جدول 1 در فایل توضیح تمرین گفته شده که باید سایز لایه تعبیه 512 باشد پس اساسا امکان استفاده از روش GloVe وجود ندارد.

سوال نهم

روش GloVe مخفف (Global Vectors) یک روش تعبیه کلمات بر اساس شمارش (count-based) است که مزایای فاکتورگیری ماتریس سراسری مانند (LSA) را با رویکرد محلی مبتنی بر پنجره‌ی متنی ترکیب می‌کند. در این روش ابتدا یک ماتریس پراکنده‌ی هم‌رخدادی کلمات (word-word co-occurrence matrix) X ساخته می‌شود که در آن هر درایه‌ی X_{ij} تعداد دفعاتی را می‌شمارد که واژه‌ی j در کانتکست واژه‌ی i در کل بدنه متنی (corpus) ظاهر شده است. شهود اصلی این است که نسبت‌های این احتمال‌های هم‌رخدادی تفاوت‌های معنایی مهمی را منتقل می‌کنند و باید در هندسه‌ی فضای تعبیه‌شده لحاظ شوند. با طراحی تابع هزینه ای که ضرب داخلی دو بردار واژه (به‌علاوه‌ی بایاس‌ها) را وادار می‌کند با لگاریتم تعداد هم‌رخدادی‌شان برابر شود، GloVe تضمین می‌کند که هم آمار رخدادن کلمات و هم ساختار معنایی محلی را میتواند در بردارهای نهایی منعکس کند.

به‌طور مشخص، GloVe بردارهای واژه w_i و بردارهای بافت $\widehat{w}_i$ را با کمینه‌سازی تابع هدف وزن‌دار زیر یاد می‌گیرد:

$$J = \sum_{i,j} f(X_{ij})\left(w_i^T \widehat{w}_i + b_i + \widehat{b}_j - log\, X_{ij}\right)^2$$

که در آن تابع f(x) یک تابع وزن‌دهی است (مثلاً $f(x) = \left({}^{x}/_{x_{max}}\right)^{a}$ برای $x < x_{max}$ ، در غیر این صورت برابر 1) که هم‌رخدادی‌های بسیار نادر یا بسیار رایج را کم‌اثر می‌کند. آموزش مدل (از طریق گرادیان کاهشی تصادفی یا AdaGrad) تنها روی درایه‌های غیرصفر X_{ij} انجام می‌شود، که باعث می‌شود حتی با واژگان بسیار بزرگ نیز کارایی حفظ شود. پس از پایان آموزش، بردار نهایی هر واژه با ترکیب (معمولاً به‌صورت جمع یا میانگین) بردارهای "واژه" و "بافت" آن به دست می‌آید، و فضایی با ابعاد پایین حاصل می‌شود که در آن روابط معنایی (مثلاً قیاس‌هایی مانند «ملکه ≈ پادشاه − مرد + زن ») به‌طور طبیعی ظاهر می‌شوند.

سوال دهم

در شبکه‌های عصبی بازگشتی ، به‌ویژه در مسائل مربوط به مدل‌سازی زبان (language modeling) یا تولید توالی (sequence generation)، محاسبه احتمال خروجی به صورت مرحله‌به‌مرحله انجام می‌شود. به این صورت که در هر مرحله یک بردار خروجی تولید می‌کند. این خروجی معمولاً از طریق یک لایه softmax عبور داده می‌شود تا احتمال توزیع‌شده‌ای روی واژگان (vocabulary)تولید کند.

در این روش، فرض های زیر در نظر گرفته میشود:

1. هر پیش‌بینی فقط به اطلاعاتی که تا لحظه فعلی در حالت پنهان h_t خلاصه شده، وابسته است. یعنی شبکه فرض می‌کند h_t می‌تواند اطلاعات لازم از کل گذشته را حفظ کند.
2. مدل فرض می‌کند که احتمال تولید کلمه در هر مرحله فقط به کلمات قبلی بستگی دارد، نه آینده. به همین دلیل از قانون زنجیره‌ای استفاده می‌شود.

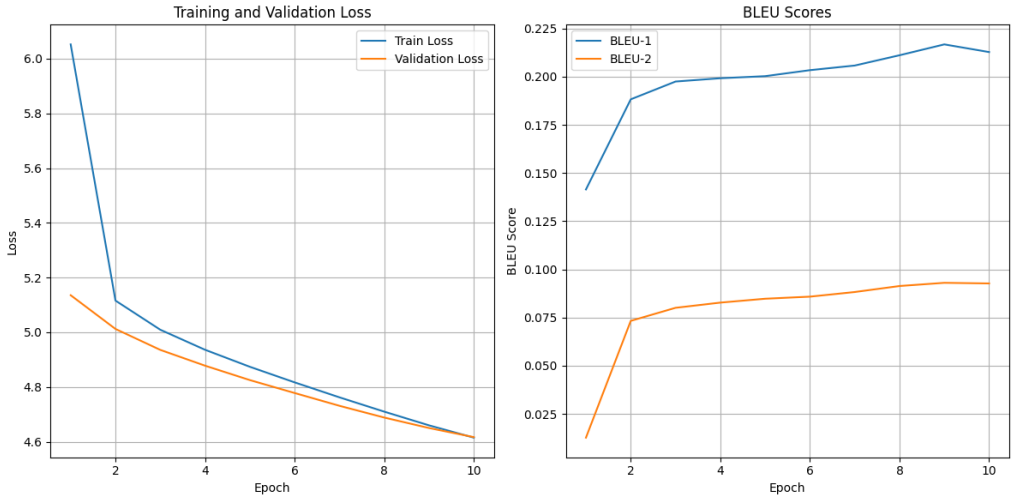
مشکلات این روش محاسبه:

1. شبکه های بازگشتی ساده در حفظ وابستگی‌های بلندمدت بین واژگان ضعیف هستند، زیرا اطلاعات در طول زمان به مرور تضعیف می‌شود (پدیده gradient vanishing).
2. مدل فقط از کلمات قبل برای پیش‌بینی استفاده می‌کند و نمی‌تواند اطلاعات آینده را در نظر بگیرد (برخلاف مدل‌های دوطرفه یا ترنسفورمرها).
3. به دلیل وابستگی زنجیره‌ای و به‌روزرسانی‌های پی‌درپی، آموزش RNN به زمان زیادی نیاز دارد و ممکن است به راحتی overfit یا underfit کند.
4. عدم قابلیت موازی سازی برای استفاده بهینه از ظرفیت های محاسباتی GPU

سوال یازدهم

برای آموزش مدل از روش توقف زودهنگام بر اساس مقدار تابع هزینه روی مجموعه داده های ارزیابی با حداکثر تعداد ایپاک برابر 10 و patience = 3 استفاده شد که نتایج به صورت زیر است:

(معیار BLEU-1 & BLEU-2 با استفاده از کتابخانه NLTK برای داده های ارزیابی بدست آمده است.)



شکل 3 - نمودار خطا و بلو حین فرایند آموزش برای مدل پایه

تعداد وزن های مدل به صورت زیر است:

Encoder (ResNet101) Parameters:

Total: 42,500,160

Trainable: 0

Frozen: 42,500,160

Decoder (LSTM) Parameters:

Total: 8,367,916

Trainable: 8,367,916

Frozen: 0

بنابراین تقریبا 16.45% از وزن های مدل رمزنگار-رمزگشا قابل آموزش است.

در شکل زیر چند نمونه از خروجی مدل برای داده های آزمایشی را میبینید:

شکل 4 - چند نمونه تصادفی از خروجی مدل پایه برای داده های آزمایشی

همانطور که مشاهده میکنید، مدل تنها میتواند کلیات تصویر را بفهمد مثل وجود مرد یا وجود سگ. اما خروجی مدل اصلا شباهتی به یک جمله برای توصیف تصویر ندارد!

بهبودهای مدل پایه

سوال دوازدهم

معیار BLEU (Bilingual Evaluation Understudy) یک معیار خودکار برای ارزیابی کیفیت ترجمه ماشینی یا تولید متن است که شباهت آماری بین جملهٔ تولیدشده (candidate) و جمله مرجع (reference) را می‌سنجد.

بصورت دقیق تر این معیار، میانگین هندسی دقت n-gram ها را اندازه‌گیری می‌کند (مثلاً unigram یا bigram) و سپس آن را با یک پنالتی طول (brevity penalty) ترکیب می‌کند تا از ترجمه‌های خیلی کوتاه جلوگیری شود.

$$BLEU = exp\left(\sum_n \omega_n \log p_n\right) \cdot BP$$

دقت n-gramها : P_n . معمولا $\omega_n = \frac{1}{N}$

BP = 1 اگر c > r

$BP = e^{(r/c-1)}$ اگر c ≤ r

که در آن:

- c: طول خروجی تولیدشده
- r: طول مرجع

توجه شود:

1. مقدار BLEU بین ۰ تا ۱ است و مقدار بالاتر نشان‌دهنده شباهت بیشتر با مرجع است.
2. BLEU فقط بر پایه تطابق سطح کلمه عمل می‌کند، و درک معنایی ندارد.

3. بیشتر برای مقایسه مدل‌ها مفید است، نه قضاوت نهایی درباره کیفیت زبانی.

سوال سیزدهم

در روش رمزگشایی حریصانه(Greedy Decoding) ، در هر مرحله فقط پرمحتمل‌ترین کلمه (یعنی کلمه با بیشترین احتمال خروجی) انتخاب می‌شود. اما این روش ممکن است به دنباله‌های ضعیف‌تری منتهی شود، چون در هر مرحله فقط یک گزینه را در نظر می‌گیرد و از مسیرهای دیگر صرف‌نظر می‌کند.

Beam Search این مشکل را با دنبال کردن چند مسیر ممکن به طور همزمان حل می‌کند. روش کار:

1. در ابتدا (در زمان تولید اولین کلمه)، به جای انتخاب فقط یک کلمه، kتا از بهترین (بالاترین احتمال) کلمات را انتخاب می‌کند که .k = beam size
2. برای هر یک از این k کلمه، در مرحله بعد دوباره k کلمه‌ی بعدی محتمل را گسترش می‌دهد. در مجموع k×k حالت ایجاد می‌شود.
3. سپس فقط k تا از بهترین دنباله‌ها را (بر اساس مجموع لگاریتم احتمال یا نمره کل دنباله) نگه می‌دارد.
4. این فرآیند تا رسیدن هر دنباله به توکن پایانی <end> یا رسیدن به حداکثر طول مجاز برای پاسخ ادامه می‌یابد.

مزایا نسبت به روش حریصانه:

- احتمال تولید دنباله‌های با کیفیت و روانتر بیشتر میشود.
- بررسی چند مسیر به جای فقط یک مسیر باعث می‌شود از گیر کردن در مسیرهای بد تا حدی جلوگیری شود.

اما باید توجه داشت که این روش هم مشکلاتی دارد:

1. با افزایش beam size هزینه محاسباتی به‌شدت بالا می‌رود.
2. Beam Search یک الگوریتم تقریبی است و تضمین نمی‌کند که بهترین دنباله ممکن (global optimum) را پیدا کند.
3. افزایش زیاد beam size ممکن است باعث شود تمام مسیرها مشابه هم شوند و تنوع کاهش یابد.
4. بایاس به دنباله‌های کوتاه‌تر: در Beam Search دنباله‌های کوتاه‌تر ترجیح داده میشود چون احتمال کلی آن‌ها بیشتر است چون تعداد کمتری عدد بین صفر و یک در هم ضرب میشوند. راه‌حل: استفاده از length normalization.

سوال چهاردهم

در مدل‌هایی که باید دنباله‌ای از خروجی‌ها را تولید کنند (مثلاً ترجمه، خلاصه‌سازی، توضیح تصویر و ...)، مدل در هر مرحله باید کلمه بعدی را بر اساس کلمه قبلی پیش‌بینی کند.

در روش Teacher Forcing، در هنگام آموزش:

- به جای آنکه مدل خروجی خودش در گام قبلی را به‌عنوان ورودی گام بعد استفاده کند،
- کلمهٔ واقعی (ground truth) از داده مرجع را به‌عنوان ورودی گام بعد به مدل می‌دهیم.

و بدین شکل سرعت آموزش مدل و دقت آن بهتر میشود.

سوال پانزدهم

مکانیزم توجه (Attention) در کاربرد توصیف تصویر (Image Captioning) نقش کلیدی در انتخاب قسمت‌های مهم تصویر در هنگام تولید هر کلمه از کپشن دارد.

محلی‌سازی مکانیزم توجه برای توصیف تصویر:

- ورودی :تصویر پس از عبور از = ماتریسی از ویژگی‌ها به شکل (B, N, D)
  - B = سایز batch
  - N = تعداد پیکسل‌های ویژگی (14*14=196)
  - D = بعد ویژگی (مثلاً 2048)
- خروجی :در هر لحظه، باید یک کلمه تولید کنیم.
  برای این کار، با توجه به وضعیت فعلی RNN (h_t) ، مشخص می‌کنیم که باید روی کدام قسمت‌های تصویر تمرکز شود.

روابط ریاضی توجه نرم(Bahdanau-style Attention) که در پیاده سازی از آن استفاده شده است:

1. پیش‌پردازش ویژگی‌ها و وضعیت مخفی:

$$e_i = v^T * \tanh(W_e * h_i + W_s * s_t)$$

در کد پایتورچ نوشته شده:

```
att1 = self.encoder_att(encoder_out)      # W_e * h_i
att2 = self.decoder_att(decoder_hidden)   # W_s * s_t
att = self.full_att(self.relu(att1 + att2.unsqueeze(1))).squeeze(2)
```

2. محاسبه وزن‌های توجه (Softmax) :

$$\alpha_i = softmax(e_i)$$

3. بردار توجه وزن‌دار(context vector) :

$$z_t = \Sigma \alpha_i * h_i$$

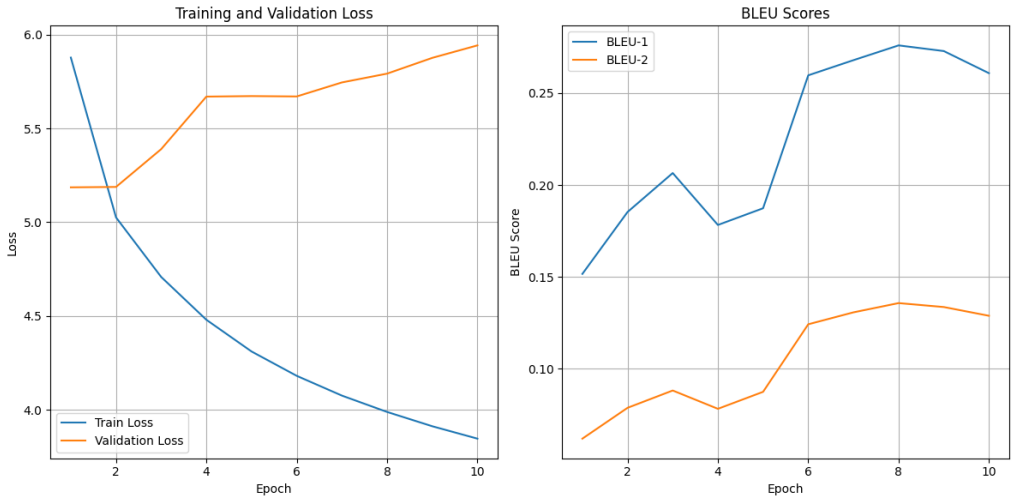
کد پایتورچ معادل:

```
attention_weighted_encoding = (encoder_out * alpha.unsqueeze(2)).sum(dim=1)
```

سوال شانزدهم

برای آموزش مدل از روش توقف زودهنگام بر اساس مقدار بلو روی مجموعه داده های ارزیابی با حداکثر تعداد ایپاک برابر 10 و patience = 3 استفاده شد که نتایج به صورت زیر است:

(معیار BLEU-1 & BLEU-2 با استفاده از کتابخانه NLTK برای داده های ارزیابی بدست آمده است.)



شکل 5 - نمودار خطا و بلو حین فرایند آموزش برای مدل بهبود یافته

تعداد وزن های مدل به صورت زیر است:

Encoder (ResNet101) Parameters:

Total: 42,500,160

Trainable: 0

Frozen: 42,500,160

Decoder (LSTM) Parameters:

Total: 12,172,589

Trainable: 12,172,589

Frozen: 0

بنابراین تقریبا 22.16% از وزن های مدل رمزنگار-رمزگشا قابل آموزش است.

در شکل زیر چند نمونه از خروجی مدل برای داده های آزمایشی را میبینید:

شکل 6 - چند نمونه تصادفی از خروجی مدل بهبود یافته برای داده های آزمایشی

همانطور که مشاهده میکنید، اضافه کردن مکانیزم توجه، روش Teacher Forcing و استفاده از Beam Search برای دیکودینگ باعث شد تا خروجی مدل بسیار بهتر از حالت قبل شود و بتواند بازیکن بسکتبال را به خوبی تشخیص بدهد. اما همچنان علامت هایی از شرطی شدن مدل (علی رغم اضافه کردن دراپ اوت با نرخ 0.15) وجود دارد بدین شکل که حتما رنگ لباس فرد (یا رنگ پوست حیوان) را اعلام کند و حتی رنگ اعلام شده هم ممکن است درست نباشد.

سوال هفدهم

مقدار بلو خروجی مدل پایه برای داده های آزمایشی:

Evaluating on test set using greedy decoding...

Test Set BLEU-1: 0.2111

Test Set BLEU-2: 0.0920

مقدار بلو خروجی مدل بهبود یافته برای داده های آزمایشی:

Evaluating on test set using beam decoding...

Test Set BLEU-1: 0.2611

Test Set BLEU-2: 0.1293

همانطور که میبینید معیار بلو بعد از اعمال کردن بهبود های گفته شده پیشرفت نسبتا خوبی داشت.

سوال هجدهم

در تصویر زیر، تفاوت چشمگیری در مقدار توجه برای پیکسل های تصویر رویت نمیشود. اگر Multi-Head Attention پیاده سازی شود احتمالا خروجی بهتری خواهیم گرفت و بدیهتا تصویر زیر یکنواخت نخواهد بود. اما بدلیل سربار محاسباتی و نیاز به حافظه بیشتر برای آموزش قابل اجرا کردن روی Google Colab نبود!

شکل 7 - مقدارتوجه برای یک ورودی

سوالات امتیازی

سوال نوزدهم

برای ستون فقرات های بهتر میتوانیم ازSwin Transformer که یکی از مدل‌های پیشرفته در حوزه بینایی ماشین است استفاده کنیم.

ساختار بخش انکودر در**Swin Transformer**

Swin Transformerبا هدف بهبود کارایی و دقت در پردازش تصاویر با وضوح بالا طراحی شده است .برخلاف مدل‌های ViT که توجه (attention) را به‌صورت سراسری اعمال می‌کنند، Swin Transformerاز مکانیزم "پنجره‌های جابه‌جا شده" برای محاسبه توجه محلی استفاده می‌کند .این رویکرد باعث کاهش پیچیدگی محاسباتی از مرتبه مربعی به مرتبه خطی نسبت به اندازه تصویر می‌شود.

ساختار انکودر Swin Transformer به‌صورت سلسله‌مراتبی (hierarchical) است و شامل مراحل زیر می‌باشد:

1. تقسیم تصویر به پچ‌های غیرهمپوشان :تصویر ورودی به پچ‌های ۴×۴ تقسیم می‌شود.
2. تعبیه خطی پچ‌ها :هر پچ با استفاده از یک لایه خطی به یک بردار ویژگی با ابعاد مشخص تبدیل می‌شود.
3. محاسبه توجه در پنجره‌های محلی :توجه چندسری (Multi-Head Self-Attention) در پنجره‌های محلی بدون همپوشانی اعمال می‌شود.
4. جابه‌جایی پنجره‌ها :برای ایجاد ارتباط بین پنجره‌های مجاور، در لایه‌های بعدی پنجره‌ها به‌صورت افقی و عمودی جابه‌جا می‌شوند.
5. ادغام پچ‌ها :در مراحل بعدی، پچ‌های مجاور با هم ادغام شده و ابعاد ویژگی افزایش می‌یابد، در حالی که وضوح مکانی کاهش می‌یابد.

این ساختار به مدل اجازه می‌دهد تا ویژگی‌ها را در سطوح مختلف و با در نظر گرفتن زمینه محلی و جهانی استخراج کند، که برای وظایفی مانند تولید توضیح تصویر بسیار مفید است.

سوال بیستم

برای جایگزینی شبکه‌های عصبی بازگشتی در بخش انکودر استفاده از معماری‌های مبتنی بر ترنسفورمر پیشنهاد می‌شود .این معماری‌ها توانسته‌اند عملکرد بهتری را ارائه دهند.

1. استفاده از مکانیزم توجه **(Attention)** پیشرفته‌تر

- استفاده از **Multi-Head Attention**به‌جای attention تک‌سری.
- استفاده از **Self-Attention** بین کلمات تولیدشده قبلی.
- توجه سلسله‌مراتبی یا **Hierarchical Attention** برای لحاظ کردن ساختار معنایی.

مثال) Transformer-based decoders **:**مانند مدل‌های ViT-GPT2 یا.GRIT

2. جایگزینی LSTM با ساختارهای مدرن‌تر

- مشکل :LSTM حافظه محدود، یادگیری سخت وابستگی‌های دور.
- بهبود:
  - استفاده از Transformer Decoder
  - استفاده از GRUیا Bi-LSTM برای ترکیب دو جهت.

3. استفاده از Pretrained Language Models

- اتصال خروجی‌های تصویری به یک مدل از پیش آموزش‌دیده تا با استفاده از *transfer learning*، کیفیت توصیف‌ها بالا ببرد. مانند:
  - GPT-2, BERT (decoder-style), T5,
  - OFA ،GIT ،BLIP

4. استفاده از ساختار دو-مرحله‌ای (Two-Stage Decoding)

- مرحله اول :تولید توصیف خام با یک دیکودر پایه.
- مرحله دوم :بهبود یا بازنویسی توصیف با یک مدل بازنویسگر (refiner) مثلاً BART یا .T5

مراجع

1. J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.
2. https://arxiv.org/abs/2103.14030
3. https://medium.com/aiguys/swin-transformer-hierarchical-vision-transformer-using-shifted-window-part-i-5dc3fe7ae774
4. https://www.ecva.net/papers/eccv_2022/papers_ECCV/papers/136960165.pdf
5. https://www.ecva.net/papers/eccv_2022/papers_ECCV/papers/136960165.pdf
6.