تمرین دوم

نام درس: یادگیری عمیق

استاد درس: دکتر محمدرضا محمدی

نام: محمد حقیقت

شماره دانشجویی: 403722042

گرایش: هوش مصنوعی

دانشکده: مهندسی کامپیوتر

نیم سال دوم 1403-1404

سوال اول

مقاله‌ی CAM-RNN یه روش خیلی جالب و پیشرفته برای توصیف ویدیوها به زبان طبیعی (Video Captioning) ارائه کرده که توی دسته مسائل Sequence-to-Sequence و از نوع Many-to-Many قرار می‌گیره. حالا بیایم ببینیم این مدل چطور کار می‌کنه و آیا می‌تونیم ازش برای توصیف تصاویر (Image Captioning) هم استفاده کنیم یا نه.

مدل CAM-RNN ترکیبی از شبکه‌های کانولوشنی (CNN) و شبکه‌های بازگشتی(RNN) به‌خصوص LSTM هست که با یه مکانیزم به اسم Co-Attention اطلاعات بصری و متنی رو باهم مدیریت می‌کنه. این مدل چهار بخش اصلی داره:

1. **ماژول توجه بصری (Visual Attention) :**

این بخش دو مرحله داره: اول روی نواحی مهم هر فریم تمرکز می‌کنه (مثلاً اگه تو فریم یه سگ در حال دویدن باشه، اون ناحیه رو شناسایی می‌کنه). بعد، فریم‌هایی که بیشترین ارتباط رو با متن توصیف دارن انتخاب می‌شن.

2. **ماژول توجه متنی (Text Attention) :**

برخلاف مدل‌های معمولی که فقط به آخرین کلمه تولیدشده نگاه می‌کنن، این بخش به عبارات چندکلمه‌ای قبلی مثل bigram یا trigram توجه می‌کنه. این کار باعث می‌شه مدل بهتر بتونه ارتباط بین کلمات رو درک کنه و کلمه بعدی رو دقیق‌تر پیش‌بینی کنه.

3. **دروازه تعادل (Balancing Gate) :**

یه جور تنظیم‌کننده‌ست که مشخص می‌کنه تو هر مرحله چقدر باید به اطلاعات بصری (تصویر) و چقدر به متن تکیه کنیم. مثلا برای کلماتی مثل «the» یا «is» که خیلی به تصویر ربطی ندارن، مدل بیشتر به متن توجه می‌کنه.

4. **تولیدکننده توصیف (Caption Generator - LSTM) :**

با استفاده از اطلاعاتی که از ماژول‌های توجه بصری و متنی می‌گیره، به علاوه وضعیت قبلی LSTM کلمه بعدی رو پیش‌بینی می‌کنه.

آیا می‌تونیم برای توصیف تصویر استفاده کنیم؟

خیلی از ایده‌های CAM-RNN برای توصیف تصاویر هم کاربرد دارن، ولی چندتا تفاوت کلیدی هم وجود داره.

شباهت‌ها:

- **توجه بصری:** همون‌طور که تو ویدیو روی نواحی مهم فریم‌ها تمرکز می‌کنه، تو تصویر هم می‌تونه روی قسمت‌های مهم تصویر (مثل یه شی خاص) زوم کنه.
- **توجه متنی:** چون ترتیب کلمات تو توصیف تصویر هم مهمه، این ماژول مستقیما قابل استفاده‌ست.
- **دروازه تعادل(Balancing Gate):** این بخش برای تنظیم میزان وابستگی به تصویر یا متن موقع تولید کلمات غیربصری (مثل «یک» یا «است») همچنان مفیده.

تفاوت‌ها:

- تو ویدیو، مدل باید ویژگی‌های زمانی رو از فریم‌های پشت سر هم استخراج کنه، ولی تو تصویر فقط با یه فریم سروکار داریم. پس توجه در سطح فریم (Frame-Level Attention) یا لازم نیست یا باید با چیزی مثل توجه مکانی (Spatial Attention) جایگزین بشه.
- تو ویدیو، یکنواختی زمانی (Temporal Consistency) خیلی مهمه، ولی تو تصویر اصلاً چنین چیزی مطرح نیست.

ایده‌ی CAM-RNN برای توصیف تصاویر هم خیلی خوب می‌تونه کار کنه، مخصوصا بخش‌های مربوط به توجه بصری، متنی و دروازه تعادل. فقط کافیه قسمت‌هایی که به توالی زمانی مربوطن (مثل توجه در سطح فریم) حذف بشن یا برای تصاویر بازطراحی بشن. با این تغییرات، این مدل می‌تونه توصیف تصاویر رو هم دقیق‌تر و طبیعی‌تر کنه.

سوال دوم

ابعاد ورودی 128 بعدی و ابعاد بردار نهان 64 بعدی

1. لایه simple RNN:

فرمول:

$h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$

تعداد پارامترها:

پارامترها = (اندازه ورودی × اندازه نهان) + (اندازه نهان × اندازه نهان) + بایاس

W_{xh}: ماتریس وزن برای ورودی به حالت نهان با ابعاد (64, 128)

W_{hh}: ماتریس وزن برای حالت نهان قبلی h_{t-1} به حالت نهان جدید h_t با ابعاد (64, 64)

b_h: بردار بایاس برای هر واحد نهان با ابعاد (64,)

Parameters = (128 × 64) + (64 × 64) + 64 = 12352

2. لایه GRU

فرمول:

دروازه به‌روزرسانی:

$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)$

دروازه بازنشانی:

$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)$

حالت پیشنهادی (candidate state):

$h\sim_t = \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h)$

⊙: ضرب عنصر به عنصر(element-wise)

حالت نهایی hidden state:

$h_t = (1-z_t) \odot h\sim_t + z_t \odot h\sim_t h_{t-1}$

درGRU سه گیت Update، Reset و Candidate وجود دارد که هر کدام به ماتریس‌های وزن و بایاس مخصوص به خود نیاز دارند.
برای هر گیت:
۱. ابعاد وزن‌های مرتبط با ورودی(W_{x*}) :

hidden_dim × input_dim = 64 × 128

۲. ابعاد وزن‌های مرتبط با حالت پنهان(W_{h*}) :

hidden_dim × hidden_dim = 64 × 64

۳. ابعاد بایاس(b_*) :

hidden_dim = 64

از آنجا که هر سه گیت به صورت مستقل محاسبه می‌شوند، تعداد کل پارامترها برابر است با:

$3 \times (W_{x*} + W_{h*} + b_*) = 3 \times 12{,}352 = 37{,}056$

3. لایه LSTM :

معادلات یک لایه LSTM به صورت زیر هستند:

$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$ (Forget Gate)

$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$ (Input Gate)

$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$ (Output Gate)

$\tilde{C}_t = \tanh(W_{xC}x_t + W_{hC}h_{t-1} + b_C)$ (Candidate Cell State)

$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$ (New Cell State)

$h_t = o_t \odot \tanh(C_t)$ (New Hidden State)

C_t: حالت حافظه (Cell State) با ابعاد ۶۴

W_{x*}, W_{h*}: ماتریس‌های وزن برای هر گیت

b_*: بایاس‌ها

در LSTM، ۴ گیت اصلی وجود دارد (Forget, Input, Output, Candidate Cell State)

هر گیت شامل:

۱ .ابعاد وزن‌های ورودی (W_{x*}) :

hidden_dim × input_dim = 64 × 128

۲ .ابعاد وزن‌های بازگشتی (W_{h*}) :

hidden_dim × hidden_dim = 64 × 64

۳ .ابعاد بایاس (b_*) :

hidden_dim = 64

از آنجا که هر ۴ گیت به صورت مستقل محاسبه می‌شوند، تعداد کل پارامترها برابر است با:

$4 \times (W_{x*} + W_{h*} + b_*) = 3 \times 12{,}352 = 49{,}408$

4. لایه BiLSTM

یک لایه BiLSTM شامل دو LSTM مستقل است:

LSTM پیشرو (Forward): پردازش داده از ابتدا به انتها.

LSTM پسرو (Backward): پردازش داده از انتها به ابتدا.

از آنجایی که پارامتر های لایه LSTM را حساب کردیم تعداد پارامترها دو برابر یک LSTM خواهد بود:

۴۹,۴۰۸ × ۲ = ۹۸,۸۱۶

سوال 3

1. Text Summarization

برای مسئله Text Summarization (خلاصه‌سازی متن)، معماری مناسب Many-to-Many از نوع Sequence-to-Sequence (Seq2Seq) که معمولا از یک RNN با حافظه بلندمدت (مانند LSTM یا GRU) استفاده می‌کند.

دلایل انتخاب معماری Many-to-Many (Seq2Seq):

ورودی و خروجی هر دو دنباله‌ای از کلمات هستند:

در خلاصه‌سازی متن، ورودی یک متن طولانی (دنباله‌ای از کلمات/توکن‌ها) است و خروجی نیز یک خلاصه کوتاه‌تر (دنباله‌ای دیگر از کلمات) است.

این دقیقا مطابق با تعریف معماری Many-to-Many است که در آن مدل یک دنباله را دریافت و یک دنباله دیگر تولید می‌کند.

2. Machine Translation

برای مسئله Machine Translation (ترجمه ماشینی)، معماری مناسب Many-to-Many با حالت Sequence-to-Sequence (Seq2Seq) است که معمولا از RNNهای پیشرفته (مانند LSTM یا GRU) استفاده می‌کند.

در ترجمه ماشینی، ورودی ما یک جمله (توالی کلمات) به یک زبان (مثلا انگلیسی) و خروجی هم همان جمله به زبان دیگر (مثلا فارسی) است. بنابراین:

ورودی: توالی‌ای از کلمات با طول متغیر (جمله زبان مبدا)

خروجی: توالی‌ای از کلمات با طول متغیر (جمله زبان مقصد)

به همین دلیل، نیاز به معماری داریم که توانایی تبدیل یک sequence به sequence دیگری را داشته باشد.

3. Video Captioning

برای مسئله Video Captioning (تولید توضیحات متنی برای ویدیو)، معماری مناسب Many-to-Many با حالت Sequence-to-Sequence (Seq2Seq) است که اغلب از ترکیب شبکه‌های کانولوشنی (CNN) و RNN یا ترانسفورمر استفاده می‌کند.

دلایل انتخاب معماری Many-to-Many (Seq2Seq):

ورودی دنباله‌ای از فریم‌ها و خروجی دنباله‌ای از کلمات است:

ورودی یک ویدیو (دنباله‌ای از فریم‌های تصویری) است که باید به صورت پردازش ترتیبی تحلیل شود.

خروجی یک جمله یا عبارت توصیفی (دنباله‌ای از کلمات) است که به صورت توکن‌به‌توکن تولید می‌شود.

این تطابق کامل با تعریف معماری Many-to-Many دارد.

4. Sentiment Analysis

برای مسئله Sentiment Analysis (تحلیل احساسات)، معماری مناسب Many-to-One است.

دلیل انتخاب:

ورودی: توالی کلمات

متن ورودی به صورت یک دنبالهٔ کلمات (یا توکن‌ها) با طول متغیر است.

خروجی: یک برچسب یا مقدار عددی

در تحلیل احساسات معمولا یک برچسب (مثلاً مثبت/منفی/خنثی) یا یک نمرهٔ پیوسته (مثلا از ۰ تا ۱) تولید می‌کنیم و این یعنی صرفا یک خروجی نهایی (One) برای کل توالی نیاز داریم.

5. Automatic Speech Recognition

برای مسئله Automatic Speech Recognition (ASR) (تشخیص خودکار گفتار)، معماری مناسب Many-to-Many با حالت دنباله‌به‌دنباله (Seq2Seq) است که اغلب از RNN های پیشرفته مانند LSTM/GRU یا ترانسفورمرها همراه با مکانیزم‌های ویژه برای مدیریت ناهماهنگی طول دنباله‌های ورودی و خروجی استفاده می‌کند.

دلایل انتخاب معماری Many-to-Many (Seq2Seq):

ورودی و خروجی هر دو دنباله‌ای از داده‌ها هستند:

ورودی: یک سیگنال صوتی که به صورت فریم‌های زمانی (مانند MFCCها یا Spectrogram frames) نمایش داده می‌شود.

خروجی: یک دنباله متنی (کاراکترها یا کلمات).

این تطابق کامل با تعریف Many-to-Many دارد، زیرا مدل باید یک دنباله بلند (صدا) را به یک دنباله کوتاه‌تر (متن) تبدیل کند.

6. Question Answering

برای مسئله پاسخگویی به سوالات، بهترین معماری از بین انواع معماری‌های RNN، معماری Many-to-Many است.

ورودی و خروجی به صورت توالی

در این مسئله، ورودی یک سوال است که به صورت توالی کلمات (مثلا چرا آسمان آبی است؟) وارد می‌شود.

خروجی نیز معمولا یک پاسخ است که می‌تواند به صورت توالی کلمات (مثلا به دلیل پراکندگی نور خورشید) باشد.

معماری Many-to-Many به طور خاص برای کارهایی طراحی شده که هم ورودی و هم خروجی آن‌ها به صورت توالی هستند.

7. Text-to-Speech

برای مسئله Text-to-Speech (TTS) (تبدیل متن به گفتار)، معماری مناسب Many-to-Many با حالت دنباله‌به‌دنباله (Seq2Seq) است که معمولا از ترکیب مدل‌های مبتنی بر توجه (Attention) و شبکه‌های عصبی بازگشتی (RNN) یا ترانسفورمرها استفاده می‌کند.

دلایل انتخاب معماری Many-to-Many (Seq2Seq):

ورودی و خروجی هر دو دنباله‌ای با طول متغیر هستند:

ورودی: یک دنباله متنی (کاراکترها، کلمات یا توکن‌های زبانی).

خروجی: یک دنباله صوتی (مثلا طیف‌نگاره‌ها یا سیگنال‌های موجی با نرخ نمونه‌برداری بالا).

طول خروجی (مثلا چند هزار فریم صوتی) بسیار بلندتر از ورودی (مثلا چند کلمه) است، اما مدل باید این ناهماهنگی را مدیریت کند.

8. Paraphrase Generation

برای مسئله Paraphrase Generation (تولید بازنویسی جمله)، معماری مناسب Many-to-Many با حالت دنباله‌به‌دنباله (Seq2Seq) است.

دلایل انتخاب معماری Many-to-Many (Seq2Seq):

ورودی و خروجی هر دو دنباله‌ای از کلمات هستند:

ورودی یک جمله (مانند "هوا امروز بسیار گرم است") و خروجی یک جمله معادل با ساختار متفاوت (مانند "امروز دمای هوا به شدت بالا رفته") است.

این مسئله نیازمند تبدیل یک دنباله به دنباله دیگر با حفظ معناست که دقیقا با تعریف Many-to-Many مطابقت دارد.

9. Code Translation

برای مسئله Code Translation (ترجمه کد از یک زبان برنامه‌نویسی به زبان دیگر)، معماری مناسب Many-to-Many با حالت دنباله‌به‌دنباله (Seq2Seq) است.

دلایل انتخاب معماری Many-to-Many (Seq2Seq):

ورودی و خروجی هر دو دنباله‌ای از توکن‌ها هستند:

ورودی: کد منبع در یک زبان برنامه‌نویسی (مانند Python) به صورت دنباله‌ای از توکن‌ها (کلیدواژه‌ها، عملگرها، متغیرها و ...).

خروجی: کد معادل در زبان مقصد (مانند JavaScript) که آن نیز یک دنباله ساختاریافته از توکن‌ها است.

سوال 4

الف)

از قانون زنجیره‌ای مشتق‌گیری (Chain Rule) در محاسبه گرادیان‌های شبکه‌های بازگشتی استفاده می‌کنیم.

ابتدا رابطه بازگشتی State ها را می نویسیم:

$$h_t = \sigma(Wh_{t-1} + Ux_t)$$

ورودی به سیگموید:

$$z_{t+1} = Wh_t + Ux_{t+1}$$

سپس مشتق فرمول سیگموید را می گیریم:

$$\sigma'(z) = \sigma(z)(1 - \sigma(z))$$

یعنی:

$$\frac{d\sigma(z)}{dz} = \sigma(z)(1 - \sigma(z))$$

در اینجا داریم:

$$\frac{\partial h_{t+1}}{\partial z_{t+1}} = h_{t+1}(1 - h_{t+1})$$

مشتق z_{t+1} نسبت به h_t:

$$z_{t+1} = Wh_t + Ux_{t+1}$$

پس:

$$\frac{\partial z_{t+1}}{\partial h_t} = W$$

با استفاده از قانون زنجیره ای داریم:

$$\frac{\partial h_{t+1}}{\partial h_t} = \frac{\partial h_{t+1}}{\partial z_{t+1}} \times \frac{\partial z_{t+1}}{\partial h_t}$$

یعنی:

$$\frac{\partial h_{t+1}}{\partial h_t} = (h_{t+1}(1 - h_{t+1})) \times W$$

نوشتن گرادیان $\frac{\partial L}{\partial h_t}$

طبق قانون زنجیره‌ای برای تابع ضرر داریم:

$$\frac{\partial L}{\partial h_t} = \frac{\partial L}{\partial h_{t+1}} \times \frac{\partial h_{t+1}}{\partial h_t}$$

که حالا مقدار $\frac{\partial h_{t+1}}{\partial h_t}$ رو جایگذاری می کنیم:

$$\frac{\partial L}{\partial h_t} = \frac{\partial L}{\partial h_{t+1}} \times W \times h_{t+1}(1 - h_{t+1})$$

ب)

می‌خواهیم گرادیان $\frac{\partial L}{\partial h_0}$ را بر حسب گرادیان $\frac{\partial L}{\partial h_t}$ بنویسیم

در قسمت قبل گفتیم که بین دو زمان پشت سر هم داریم:

$$\frac{\partial L}{\partial h_t} = \frac{\partial L}{\partial h_{t+1}} \times W \times h_{t+1}(1 - h_{t+1})$$

یعنی گرادیان در زمان t بستگی به گرادیان در زمان t+1 دارد.

حالا برای رفتن از h_t به h_0 باید بارها این فرمول را پشت سر هم اعمال کنیم.

$$\frac{\partial L}{\partial h_{t-1}} = \frac{\partial L}{\partial h_t} \times W \times h_t(1 - h_t)$$

$$\frac{\partial L}{\partial h_{t-2}} = \frac{\partial L}{\partial h_{t-1}} \times W \times h_{t-1}(1 - h_{t-1})$$

و همینطور ادامه بده تا برسیم به h_0.

نوشتن رابطه نهایی برای $\frac{\partial L}{\partial h_0}$

ترکیب این روابط پشت سر هم باعث می‌شود به فرمول زیر برسیم:

$$\frac{\partial L}{\partial h_0} = \frac{\partial L}{\partial h_t} \times \prod_{k=0}^{t-1} (W \times h_{k+1}(1 - h_{k+1}))$$

ج)

برش گرادیان (Gradient Clipping) تکنیکی است که برای جلوگیری از "منفجر شدن گرادیان‌ها " (exploding gradients) در هنگام آموزش شبکه‌های عصبی استفاده می‌شود.
دو روش اصلی برای این کار داریم: برش بر اساس مقدار و برش بر اساس اندازه.

1. برش گرادیان بر اساس مقدار (Clipping by Value)

در این روش، هر مولفه‌ی گرادیان (هر عددی از ماتریس گرادیان) جداگانه بررسی می‌شود.
اگر یک مولفه از حد خاصی بیشتر شود (مثلا از عدد ۵ بزرگ‌تر شود)، ما آن را به همون حد مجاز محدود می‌کنیم.

مثال:
فرض کن یک مولفه گرادیان مقدارش ۷ شده و حد ما ۵ است. در این حالت، آن مولفه را ۵ قرار می‌دهیم.
برعکس، اگر مقدارش خیلی منفی (مثلا -۸) شده باشد، آن را به -۵ محدود می‌کنیم.

2. برش گرادیان بر اساس اندازه (Clipping by Norm)

در این روش، ما به همه‌ی گرادیان به صورت یک بردار کامل نگاه می‌کنیم و "اندازه یا نُرم" کل بردار مثلا نُرم درجه ۲ یا همان Euclidean Norm را محاسبه می‌کنیم.
اگر اندازه‌ی کل بردار از یک مقدار تعیین‌شده (threshold) بیشتر شد، کل بردار را به صورت یکنواخت کوچک می‌کنیم (یعنی همه‌ی مولفه‌ها را با یک نسبت ثابت کوچک‌تر می‌کنیم) تا اندازه‌ی کل بردار برابر با حد مجاز شود.

مثال:
فرض کن نُرم گرادیان برابر ۱۰ شده ولی ما حداکثر اندازه را ۵ تعیین کرده‌ایم. در این حالت همه‌ی مقادیر گرادیان را نصف می‌کنیم تا نُرم کلش بشود ۵.

حالا برتری برش بر اساس اندازه نسبت به برش بر اساس مقدار چیه؟

هماهنگی بهتر: برش بر اساس اندازه، همه‌ی مولفه‌های گرادیان را با هم مقیاس می‌کند و جهت بردار گرادیان را حفظ می‌کند. این خیلی مهم است چون جهت گرادیان تعیین می‌کند شبکه به چه سمتی آموزش ببیند.

رفتار طبیعی‌تر: در برش بر اساس مقدار، ممکن است فقط بعضی مولفه‌ها قطع شوند و بعضی دیگر نه، که این باعث تغییر غیرطبیعی در جهت کلی گرادیان می‌شود. این تغییر جهت می‌تواند یادگیری مدل را مختل کند.

پیش‌بینی‌پذیری بهتر: چون کل اندازه‌ی بردار کنترل می‌شود، راحت‌تر می‌شود رفتار مدل را تحلیل و مدیریت کرد.

برای رفع برخی ایرادات و ابهامات از AI استفاده شده است.