دانشگاه صنعتی شریف

دانشکده مهندسی مکانیک

شبکه‌های عصبی بازگشتی

گزارش تمرین ۳_تشریحی

رباتیک اجتماعی و شناختی

مریم کریمی جعفری، ۹۹۱۰۶۶۱۷

استاد

دکتر طاهری

بهار ۱۴۰۳

سوال ۱:

الف)

مشکل vanishing gradient یعنی مقدار گرادیان در مرحله backpropagation آنقدر کوچک شود که بعد از آپدیت پارامترها مقدار آن‌ها عملا تفاوتی نکند. حال برسی می‌کنیم که LSTM چگونه این مشکل را حل کرده است.

Backpropagation through time in LSTMs

گرادیان در شبکه‌ها عصبی بازگشتی به صورت زیر محاسبه می‌شود:

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W}$$

اگر بخواهیم ناپدیدشدن گرادیان رخ دهد باید مقدار بالا به ازای تمامی Tها صفر شود زیرا یک سری است که باید به صفر همگرا شود.

$$(S_1, S_2, S_3, \ldots)$$

$$S_n = \sum_{t=1}^{n} \frac{\partial E_t}{\partial W}$$

مقدار گرادیان برای time step k به صورت زیر است:

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \cdots \frac{\partial c_2}{\partial c_1} \frac{\partial c_1}{\partial W}$$

$$= \frac{\partial E_k}{\partial h_k} \frac{\partial h_k}{\partial c_k} \left(\prod_{t=2}^{k} \frac{\partial c_t}{\partial c_{t-1}} \right) \frac{\partial c_1}{\partial W}$$

این مقدار عبارت زیر است که در RNNها برای مقادیر بالای k به دلیل وجود تابع tanh، به صفر می‌کرد:

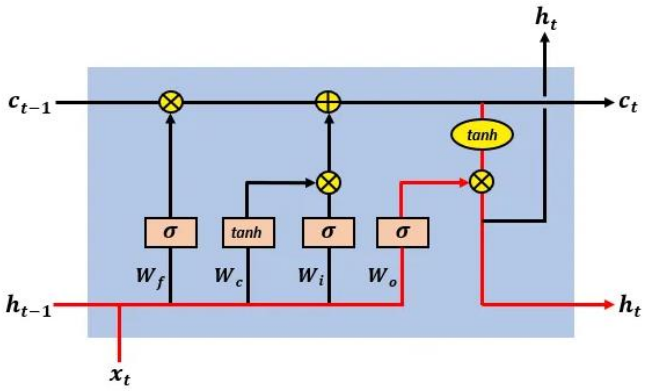
$$\prod_{t=2}^{k} \frac{\partial c_t}{\partial c_{t-1}}$$

مقدار c_t برابر است با:

$$c_t = c_{t-1} \otimes \sigma(W_f \cdot [h_{t-1}, x_t]) \oplus$$

$$tanh\,(W_c \cdot [h_{t-1}, x_t]) \otimes \sigma(W_i \cdot [h_{t-1}, x_t])$$

$$c_t = c_{t-1} \otimes f_t \oplus \tilde{c}_t \otimes i_t$$



این c_t تابع مقادیری است که باید برای محاسبه مشتقش، از قانون مشتق زنجیره‌ای استفاده کرد:

$$\frac{\partial c_t}{\partial c_{t-1}} = \frac{\partial}{\partial c_{t-1}}[c_{t-1} \otimes f_t \oplus \tilde{c}_t \otimes i_t]$$

$$= \frac{\partial}{\partial c_{t-1}}[c_{t-1} \otimes f_t] + \frac{\partial}{\partial c_{t-1}}[\tilde{c}_t \otimes i_t]$$

$$= \frac{\partial f_t}{\partial c_{t-1}} \cdot c_{t-1} + \frac{\partial c_{t-1}}{\partial c_{t-1}} \cdot f_t + \frac{\partial i_t}{\partial c_{t-1}} \cdot \tilde{c}_t + \frac{\partial \tilde{c}_t}{\partial c_{t-1}} \cdot i_t$$

در ادامه با محاسبه مشتق‌ها داریم:

$$\frac{\partial c_t}{\partial c_{t-1}} = \sigma'\big(W_f \cdot [h_{t-1}, x_t]\big) \cdot W_f \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot c_{t-1}$$
$$+ f_t$$
$$+ \sigma'(W_i \cdot [h_{t-1}, x_t]) \cdot W_i \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot \tilde{c}_t$$
$$+ \sigma'(W_c \cdot [h_{t-1}, x_t]) \cdot W_c \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot i_t$$

اگر در نظر بگیریم:

$$A_t = \sigma'\big(W_f \cdot [h_{t-1}, x_t]\big) \cdot W_f \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot c_{t-1}$$
$$B_t = f_t$$
$$C_t = \sigma'(W_i \cdot [h_{t-1}, x_t]) \cdot W_i \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot \tilde{c}_t$$
$$D_t = \sigma'(W_c \cdot [h_{t-1}, x_t]) \cdot W_c \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot i_t$$

با جایگذاری در فرمول مشتق c_t خواهیم داشت:

$$\frac{\partial c_t}{\partial c_{t-1}} = A_t + B_t + C_t + D_t$$

در نهایت فرمول گرادیان به صورت زیر خواهد شد:

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k}\frac{\partial h_k}{\partial c_k}\left(\prod_{t=2}^{k}[A_t + B_t + C_t + D_t]\right)\frac{\partial c_1}{\partial W}$$

مشکل ناپدیدشدن گرادیان به دلیل مشتق تابع tanh بود. در هر یک از عبارات A_t، C_t و D_t عبارت مشتق tanh وجود دارد. اما در عبارت B_t که همان forget gate است، این عبارت وجود ندارد. پس دروازه فراموشی در حل مشکل vanishing gradient نقش دارد.

منابع:

- **Arbel, Nir (2018). How LSTM networks solve the problem of vanishing gradients. https://medium.datadriveninvestor.com/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577**

ب)

گیت فراموشی: با توجه به ورودی جدید و h لایه قبل و با کمک تابع فعالسازی سیگموید، مشخص می‌کند که چه اطلاعاتی اط قبل که داخل cell state بود مفید هستند و می‌توانند باقی بمانند. در این مرحله اطلاعاتی که دیگر در این زمان نیازی به آن‌ها نیست حذف می‌شوند. این مرحله از اشباع شدن حافظه با اطلاعات غیر مرتبط جلوگیری می‌کند.

گیت ورودی: با توجه به ورودی جدید و h لایه قبل و با کمک تابع فعالسازی سیگموید، مشخص می‌کند که چه مقدار از این اطلاعات جدید برای ذخیره شدن در cell state مفید هستند. اطلاعات جدید از تابع فعالسازی tanh عبور کرده‌اند. این اطلاعات به اطلاعات گذرنده از گیت فراموشی اضافه می‌شوند.

گیت خروجی: با توجه به ورودی و h لایه قبل مشخص می‌کند که چه مقدار از اطلاعات داخل cell state بعد از عبور از دو دروازه قبلی، برای خروجی hidden state یا همان h همین لایه مناسب هستند.

به طور خلاصه موارد زیر توسط این گیت‌ها اتفاق می‌افتد:

فیلتر کردن اطلاعات: گیت‌های فراموشی و ورودی به LSTM اجازه می‌دهند تا اطلاعات نامربوط را فیلتر کرده و بر روی اطلاعات مرتبط تمرکز کنند.

جمع‌آوری اطلاعات: گیت ورودی اطلاعات جدید را به طور گزینشی به حافظه اضافه می‌کند و به LSTM کمک می‌کند تا الگوهای long-range را در طول زمان ایجاد کند.

تمرکز بر الگوهای long-range: گیت خروجی بخش‌های مرتبط از اطلاعات ذخیره‌شده در حافظه را فیلتر می‌کند و به LSTM اجازه می‌دهد تا بر ارتباط‌های long-range تمرکز کند.

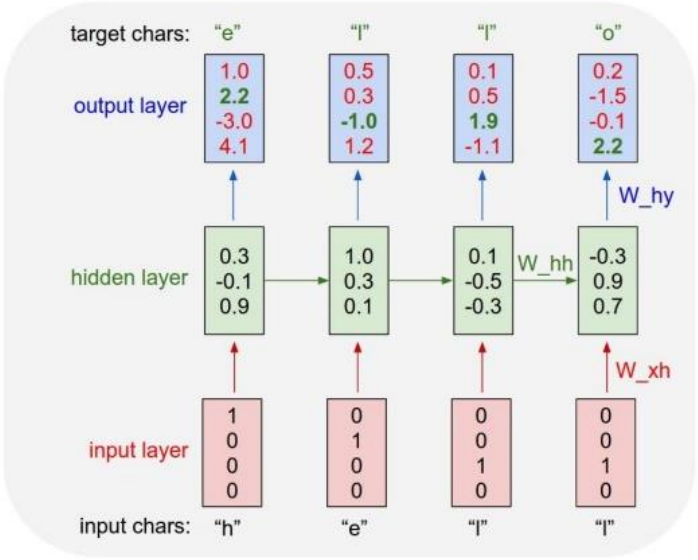
پ)

اولین راهی که به ذهن می‌رسد این است که داده‌های ناقص به کلی حذف شوند که البته این کار ممکن است باعث کاهش حجم اطلاعات و یا از بین رفتن اطلاعات شود. یا اینکه باید داده‌های حذف شده و یا ناقص را با مقادیری جایگزین کرد (imputation). یکی از این روش‌ها این است که با میان‌یابی (interpolation) جاهای خالی را پر کرد. خود میان‌یابی می‌تواند خطی و یا غیرخطی باشد. در روشی دیگر داده‌های ناقص را با میانگین feature مربوطه پر می‌کنند. یا اینکه می‌توان از شبکه‌های عصبی برای تکمیل کردن داده‌ها استفاده کرد. یعنی

شبکه عصبی مثل حتی یک MLP ساده train کنیم که بتواند به بهترین نحو داده‌های ناقص را کامل کند. autoencoderها و یا شبکه‌های generative می‌توانند در این مواقع به کار روند.

ت)

Teacher forcing مبحثی نیست که از قبل با آن آشنا نباشیم. در کی از اسلایدهای درس مثال زیر را داریم:



این مثال برای پیش بینی کلمه ‘hello’ به کار برده شده است. ورودی را به صورت برداری شامل ۴ عدد در نظر بگیرید که هر عدد به ترتیب متعلق به حروف h، e، l و o است. اولین ورودی h است و خروجی توالی(sequence) اول باید e باشد. اما با توجه به اعداد خروجی، مدل o را پیش بینی کرده است زیرا مقدار آن یعنی ۴.۱، بیشترین مقدار است. teacher forcing بیان می‌کند که در مرحله آموزش، به جای اینکه خروجی هر مرحله را (که در اینجا در مرحله اول o است) به عنوان ورودی مرحله بعد بدهیم، مقدار صحیح آن یعنی همان e را (در این مرحله از مثال) بدهیم که در تصویر بالا مشخص است.

مزایا:

Faster Convergence

با این روش فرایند آموزش مدل سریع‌تر اتفاق می‌افتد. یعنی مدل زودتر به دقت مطلوب می‌رسد.

Reduced Error Propagation

این روش **انتشار خطا** در طول فرایند را کاهش می‌دهد. یعنی از بزرگ و انباشته‌شدن خطایی که در مراحل اولیه آموزش وجود دارد، جلوگیری می‌کند. زیرا دیگر خطای زیادی در مراحل اولیه وجود ندارد. برای درک مفهوم انتشار خطا گلوله برفی را در نظر بگیرید که از بالای یک کوه برفی به پایین سر بخورد. این گلوله برفی در راه خود به دلیل تماس با برف‌های بیشتر، بزرگ و بزرگ‌تر می‌شود.

Stable Training

این روش با حفظ یک رابطه‌ی ورودی-خروجی شفاف و تعریف‌شده، فرآیند آموزشی باثبات و ثابتی را فراهم می‌کند.

Explicit Supervision

با این روش وقتی مدل به جواب صحیح دسترسی داشته باشد، بهتر می‌تواند در مواقعی که خروجی از الگوهای خاص پیروی می‌کند، عمل کند. برای مثال در تسک‌هایی مثل ترجمه ممکن است مدل با دانستن خروجی واقعی بتواند مواردی مثل گرامر یا لغات زبان مقصد را بیاموزد.

Controlled Exploration

مدل باید بتواند برای یافتن خروجی مطلوب، explorationهای درست انجام دهد. این روش کاوش‌خای مدل را کنترل و هدایت شده می‌کند؛ به طوری که قدرت generalization مدل برای مثال‌های unseen بیشتر شده و خروجی‌های معنادارتری تولید می‌کند.

Easier Evaluation

در نهایت این روش می‌تواند باعث بهتر و آسان‌تر انجام شدن ارزیابی‌ها شود زیرا که خروجی که با کمک آن می‌توان error را محاسبه کرد مشخص است.

معایب:

Exposure Bias

در این پدیده، مدل ممکن است بیش از حد به سیگنال‌های ارائه شده توسط معلم **وابسته** شود و در طول استنتاج (یعنی بدون اجبار معلم) با مشکل مواجه شود. استنتاج (inference)، فرآیندی است که در آن مدل از دانش آموخته‌ی خود برای تولید خروجی‌های جدید بر اساس ورودی‌های جدید استفاده می‌کند.

Mismatch Between Training and Inference

اگر مدل در طول آموزش به شدت به Teacher Forcing تکیه کند، ممکن است در طول فرایند inference عملکرد خوبی نداشته باشد. در استنتاج، مدل باید به طور مرحله به مرحله توالی را تولید کند، بدون اینکه به خروجی واقعی دسترسی داشته باشد. به عبارت دیگر، ناسازگاری بین نحوه آموزش مدل و نحوه عملکرد آن در دنیای واقعی وجود دارد. در طول آموزش، مدل با استفاده از teacher forcing، ورودی "ایده‌آل" را در هر مرحله دریافت می‌کند. اما در استنتاج، مدل باید بر اساس ورودی که دریافت می‌کند، خروجی بعدی را به طور مستقل پیش‌بینی کند. این عدم تطابق می‌تواند منجر به کاهش عملکرد مدل در هنگام استفادهٔ واقعی شود.

Lack of Real-World Noise

در واقعیت داده‌ها ممکن است نویزی باشند. مدلی که با teacher forcing آموزش دیده است ممکن است نتواند به خوبی داده‌های نویزی را handle کند.

Limited Exploration

از جمله مزایا این بود که explorationهای مدل جهت‌دار می‌شود. اما باید توجه کرد که در عین حال این کاوش‌ها محدود می‌شوند و مدل نمی‌تواند حتی کمی آزادی و خلاقیت فراتر از چیزی که به آن آموزش داده شده است، داشته باشد.

Incomplete Training Data

وقتی که قرار باشد مدل خیلی به ورودی درست تکیه کند، ممکن است در مواقعی که دیتا ناقص باشد نتواند خروجی مطلوبی تولید کند. این مشکل در تسک‌های ترجمه اتفاق می‌افتد.

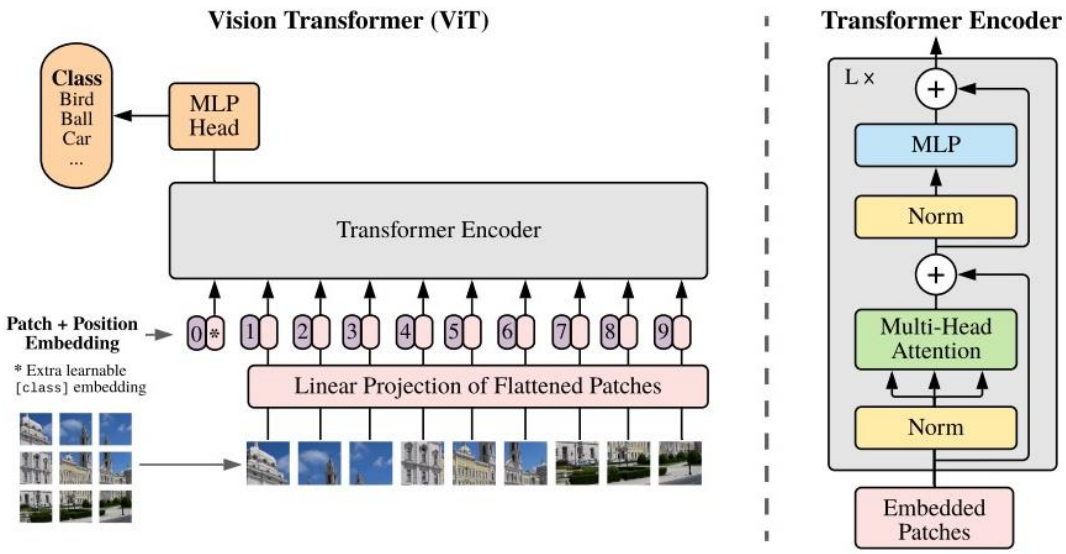
Resource-Intensive

در نهایت استفاده از teacher forcing نیازمند به داشتن دیتای غنی است که جمع آوری این میزان داده نیز زمان زیادی می‌طلبد.

منابع:

- Wong, Wanshun (2019). What is Teacher Forcing?. https://towardsdatascience.com/what-is-teacher-forcing-3da6217fed1c?gi=21e413ecd815
- Van Otten, Neri (2023). Teacher Forcing In Recurrent Neural Networks (RNNs): An Advanced Concept Made Simple. https://spotintelligence.com/2023/10/12/teacher-forcing-in-recurrent-neural-networks-rnns-an-advanced-concept-made-simple/

ث)



فرض می‌کنیم ابعاد تصویر H*W با C کانال است. vision transformer تقریبا از همان ترنسفورمرهای عادی تشکیل شده است اما تصاویر را باید به فرم یک بردار برای ورودی آن‌ها تبدیل کرد. هر تصویر به دسته‌هایی تبدیل می‌شوند. این دسته‌ها (patches) هر کدام یک تصویر با ابعاد پایین‌تری هستند. برای مثال تصویر H*W*C ما به N تصویر با ابعاد P*P*C تقسیم می‌شود. در تصور بالا این مرحله مشخص است.

در ادامه هر تصویر flatten می‌شود تا به یک بردار یک‌بعدی تبدیل شود. اما از آنجایی که همچنان سایز تصویر بزرگ است، با فرض اینکه ورودی transformer برداری با ابعاد 1*D است، تصویر توسط linear projection مپ می‌شود که به آن **patch embedding** می‌گویند. حال این بردار می‌تواند به عنوان ورودی به ترنسفورمر داده شود. اما در کنار بردار embedشدهٔ تصویر، برداری تحت عنوان **position embedding** با همان ابعاد نیز با آن جمع شده و سپس به ترنسفورمر وارد می‌شوند. این بردار نمادی از محل قرارگیری تصویر patch از تصویر اصلی است.

حال ما N ورودی داریم که در تصویر بالا این تعداد است. باید به این نکته توجه کرد که در ویژن ترنسفرها در کنار تصاویر، یک توکن اضافه‌تر در ابتدا به عنوان توکن صفر نیز به شبکه داده می‌شود. این توکن همان [class] مشابع توکن [cls] در BERT است. این توکن در پایان شبکه نمادی از دسته‌ای است که تصویر به آن تعلق دارد؛ یعنی از این توکن در آخر برای classification تصویر استفاده می‌شود. مقدار این توکن یکی از learnable parameterها است که باید train شود. در لایه اول با نام $z_0^0 = x_{class}$ و در لایه آخر با نام z_L^0 شناخته می‌شود. مقدار در آخر به یک شبکه MLP با تابع فعالسازی GELU داده می‌شود تا فرایند

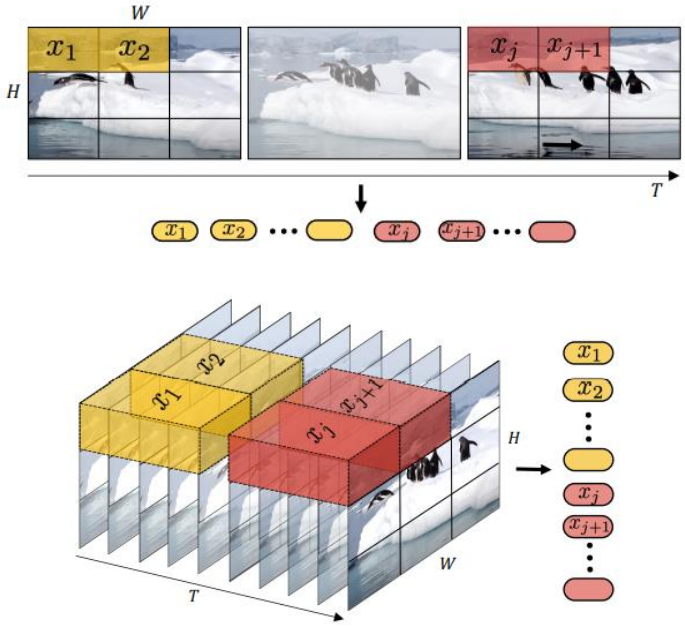
classification انجام گیرد. یعنی فرایند classification تنها به توکن ابتدایی که جزو تصویر نبود نیاز دارد. اما باید توجه کرد که این توکن خود به سایر توکن‌ها وابسته است.

در نهایت یکی از تفاوت‌های ساختار خود ترنسفورمر به ترنسفورمر عادی مقاله attention is all you need وجود فرایندهای نرمالیزیشن در شبکه است (Layernorm). نرمالیزیشن قبل از هر بلوکی انجام می‌شود که در تصویر صفحه قبل به خوبی مشخص است.

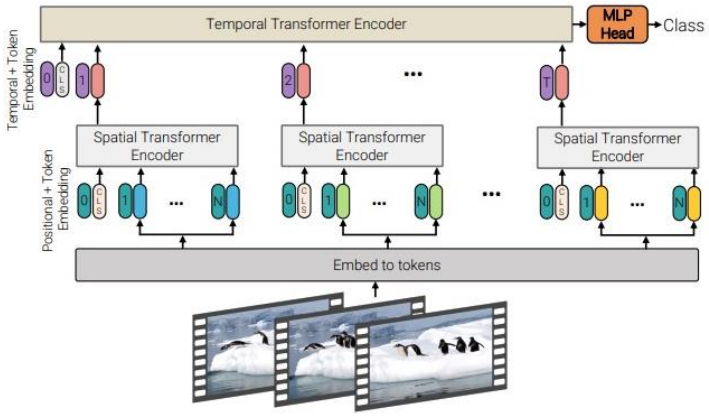
$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D},\ \mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D}$$
$$\mathbf{z}'_\ell = \text{MSA}(\text{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \qquad \ell = 1 \ldots L$$
$$\mathbf{z}_\ell = \text{MLP}(\text{LN}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell, \qquad \ell = 1 \ldots L$$
$$\mathbf{y} = \text{LN}(\mathbf{z}_L^0)$$

چگونه از *vision transformer* برای ویدیو استفاده کنیم؟

در قدم اول باید توجه کرد که در ویدیوها علاوه بر مکان، زمان نیز باید مدنظر قرار گیرد. ما باید به دنبال راه حلی باشیم که بتوان هر فیلم را به توکن‌هایی برای ورودی یک *transformer* از هر نوعی، تبدیل کرد. یک از راه‌ها یان است که هر فریم به عنوان یک تصویر به *patch*هایی تقسیم شود. برای مثال اگر n_t فریم داشته باشیم که هر کدام $n_h * n_w$ تعداد *patch* تولید کند، تعداد کل *patch*ها $n_t * n_h * n_w$ خواهد بود. حال هر *patch* تحت *linear projection* به توکن‌های ورودی *vision transformer* تبدیل شود. راه‌حل دوم این است که *patch*ها را در عمق ایجاد کنیم.



یک دیگر از راه‌هایی که وجود دارد این است که در ابتدا هر فریم از ویدیو را به ViT بدهیم سپس خروجی آن‌ها را به یک شبکه مثل *RNN*، *LSTM* و یا حتی یک *transformer* دیگر بدهیم تا بتواند توالی زمانی را بررسی کند.



منابع:

- *Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." arXiv preprint arXiv:2010.11929 (2020).*
- *Arnab, Anurag, et al. "Vivit: A video vision transformer." Proceedings of the IEEE/CVF international conference on computer vision. 2021.*

سوال ۲:

سوال 2 :

الف) به‌طور کلی برای هر t داریم :

$$h_t = (x_t + h_{t-1})(w)$$

ب) روابط انتشار به جلو این شبکه به صورت زیر است :

$$p = x_0 + h_0 \qquad r = x_1 + q \qquad t = x_2 + s$$
$$q = wp \qquad s = wr \qquad \hat{y} = wt$$

$$\Rightarrow \hat{y} = w\left[x_2 + w\left(x_1 + w(x_0 + h_0)\right)\right]$$

پ و ت)

$$\frac{\partial Loss}{\partial w} = \frac{\partial Loss}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w}$$
$$\frac{\partial Loss}{\partial \hat{y}} = -2(y - \hat{y}) = 2(\hat{y} - y)$$

حال به محاسبه $\frac{\partial \hat{y}}{\partial w}$ می‌پردازیم :

$$\frac{\partial \hat{y}}{\partial w} = \frac{\partial w}{\partial w} t + w \frac{\partial t}{\partial w} = t + w \frac{\partial t}{\partial w}$$
$$\frac{\partial t}{\partial w} = \frac{\partial s}{\partial w} \Rightarrow \frac{\partial \hat{y}}{\partial w} = t + w \frac{\partial s}{\partial w} = t + w\left[r + w \frac{\partial r}{\partial w}\right]$$
$$\frac{\partial r}{\partial w} = \frac{\partial q}{\partial w} = p \Rightarrow \frac{\partial \hat{y}}{\partial w} = t + w\left[r + wp\right]$$

$$\Rightarrow \frac{\partial \hat{y}}{\partial w} = x_2 + w\left(x_1 + w(x_0 + h_0)\right) + w\left[\left(x_1 + w(x_0 + h_0)\right) + w(x_0 + h_0)\right] =$$
$$x_2 + 2w x_1 + 3w^2(x_0 + h_0)$$

$$\Rightarrow \frac{\partial Loss}{\partial w} = 2(\hat{y} - y)\left[x_2 + 2w x_1 + 3w^2(x_0 + h_0)\right] \quad (\star)$$

مشتق جزئی روی یال اول:

$$\left(\frac{\partial loss}{\partial w}\right)_1 = 2(\hat{y}-y)(t) = 2(\hat{y}-y)\left[x_3 + w(x_4 + w(x_0+h_0))\right]$$

مشتق جزئی روی یال دوم:

$$\left(\frac{\partial loss}{\partial w}\right)_2 = 2(\hat{y}-y)(wr) = 2(\hat{y}-y)(w)\left(x_4 + w(x_0+h_0)\right)$$

مشتق جزئی روی یال سوم:

$$\left(\frac{\partial loss}{\partial w}\right)_3 = 2(\hat{y}-y)(w^2 p) = 2(\hat{y}-y)(w^2)(x_0+h_0)$$

همان طور که در رابطه (۴) دیده می‌شود، وزن‌ها با توان 2 نیز ظاهر شده‌اند. اگر این وزن‌ها خیلی کوچک باشند، مقدار $\frac{\partial loss}{\partial w}$ خیلی ناچیز تغییر خواهد کرد و حتی ممکن است به صفر میل کرده و به شکل ناپدید شدن گرادیان vanishing gradient را به وجود آورد. از طرفی اگر این وزن‌ها خیلی بزرگ باشند، از آنجایی که به توان 2 نیز می‌رسند، مقدار گرادیان ممکن است به بی‌نهایت میل کند و به شکل انفجار گرادیان exploding gradient را به وجود آورند.

سوال ۳:

سوال 3:
تمام حالت هایی که دو بیت می توانند داشته باشند به همراه حالت هایی که رقم carry می تواند داشته باشد، در جدول زیر آمده اند؛ رقم carry نشان دهنده این است که جمع دو بیت 2 شده است یا خیر:

x_1	x_2	carry	y
0	0	0	0
0	1	0	1
1	0	0	0
1	1	0	0 + 1 carry
0	0	1	1
0	1	1	0 + 1 carry
1	0	1	0 + 1 carry
1	1	1	1 + 1 carry

h های جدیدی که اضافه می‌شوند ← ← همان h لایه قبل

با توجه به جدول بالا ما در کل دو حالت برای carry داریم:

C : carry ⇒ 1
NC : No carry ⇒ 0

اما چهار جفت حالت برای x ها داریم: $\begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}$

همچنین خروجی y هم کلا دو حالت 0 یا 1 را نشان می‌دهد. اما هر یک از این دو می‌تواند NC یا C باشد.

پس در کل چهار حالت داریم: (NC,0), (C,0), (NC,1), (C,1)

برای مثال (NC,0) یعنی هیچ رقم carry از state قبل نداشته باشیم و 0 هم در خروجی ظاهر شود. هر بلوک RNN می‌تواند یکی از 4 حالت بالا باشد و همچنین می‌تواند از یکی از 4 حالت بالا به بلوک RNN فعلی برویم، یعنی بلوک‌های RNN قبل هم می‌توانند حالت های

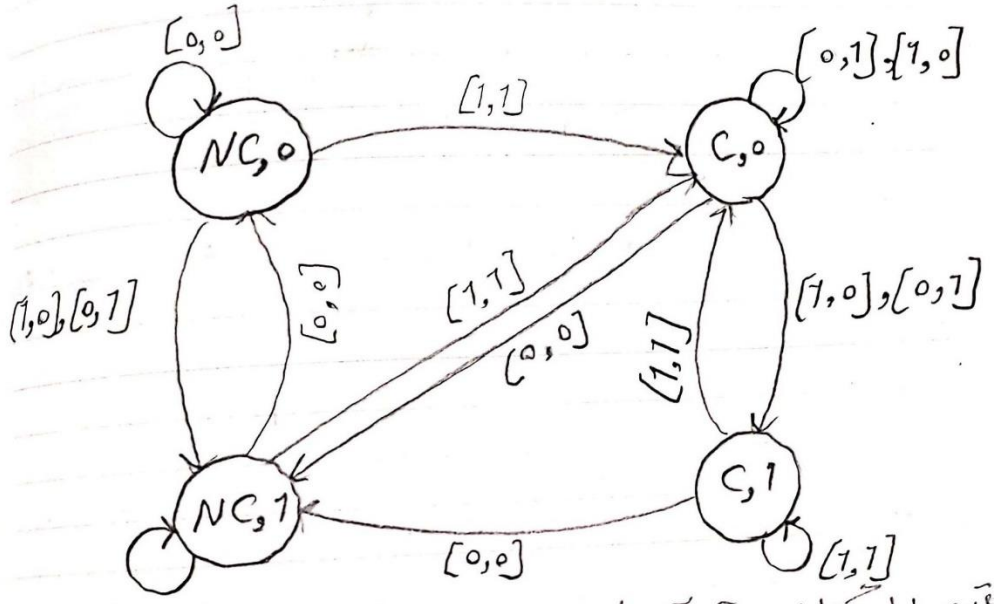
داشته باشند که به نتیجه‌ی نهایی ختم نشود. حالت‌های زیر در کل به وجود می‌آیند:

from \ to	(NC,0)	(NC,1)	(C,0)	(C,1)
(NC,0)	[0,0]	[1,0] و [0,1]	[1,1]	—
(NC,1)	[0,0]	[1,0] و [0,1]	[1,1]	—
(C,0)	—	[0,0]	[1,0] و [0,1]	[1,1]
(C,1)	—	[0,0]	[1,0] و [0,1]	[1,1]

این جدول بیان می‌کند که برای مثال ما در state قبلی در حالت (NC,0) بودیم، یعنی بلوک RNN قبل ما $y_t=0$ را پرینت کرده بود و رقم carry=0 را ایجاد کرده و به بلوک فعلی می‌فرستد. حال برای اینکه بلوک فعلی نیز $y_t=0$ را پرینت و carry=0 را ایجاد کند برای بلوک بعدی ما باید بیت‌های ورودی یا همان $[x_1, x_2]$ ما [0,0] باشند. با توجه به ردیف اول، هیچگاه از حالتی که رقم carry نداریم، نمی‌توان به حالتی که رقم carry داشته باشیم و همچنین $y_t=1$ شود، برویم.

یا برای مثال اگر بخواهم ردیف سوم را نیز توضیح دهم. بلوک RNN قبلی $y_t=0$ را در خروجی پرینت کرده و رقم carry=1 را ایجاد کرده تا به بلوک فعلی بیاید. برای اینکه برای مثال در خروجی $y_t=1$ پرینت شود و همچنان یک رقم $carry_{t-1}=1$ داشته باشیم، باید بیت‌های ورودی هر دو 1 باشند. جمع دو عدد 1، خودش $y_t=0$ و $carry_t=1$ ایجاد می‌کند، از طرفی ما $carry_{t-1}=1$ از بلوک قبلی داشتیم که با y_t جمع شده و $y_t=1$ می‌شود و $carry_t=1$ خواهد شد.

جدول صفحه قبل را به شکل دیاگرام زیر نیز می‌توان نشان داد:



تصویر بهتری از این گراف در آمده است.

گفتیم تعداد نرون‌های لایه مخفی ۳ تاست. خروجی لایه مخفی یا همان بردار hها مشخص می‌کند در کدام یک از حالت‌های $(C,1), \dots, (NC,0)$ هستیم. پس هر یک از این ۴ حالت را با یک بردار مشخص می‌کنیم که درایه‌های آن صفر یا یک هستند. چنین حالت مختلف داریم که من حالت‌های زیر را انتخاب کرده‌ام:

$(NC,0) \rightarrow [0,0,0]$

$(NC,1) \rightarrow [0,1,0]$

$(C,0) \rightarrow [1,1,1]$

$(C,1) \rightarrow [0,1,1]$

h_1 و h_3 به h_2

حال با توجه به Notation‌های اینجا، مدلی برای inputها، outputها و ... این مسئله می‌نویسیم تا نرون‌ها را بیابیم.

جدول در صفحه بعد آمده است.

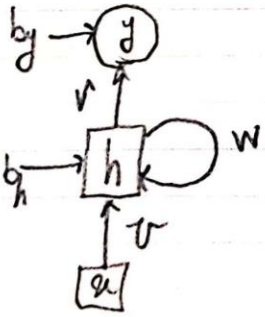
t	$h_1(t-1)$	$h_2(t-1)$	$h_3(t-1)$	$x_1(t)$	$x_2(t)$	$y(t)$	$h_1(t)$	$h_2(t)$	$h_3(t)$
1	0	1	0	1	0	1	0	1	0
2	0	1	0	1	1	0	1	1	1
3	1	1	1	1	0	0	1	1	1
4	1	1	1	0	0	1	0	1	0
5	0	1	0	0	1	1	0	1	0
6	0	1	0	1	1	0	1	1	1
7	1	1	1	0	0	1	0	1	0

به طور خلاصه:

$t-1$ | t

$t_1 \Rightarrow [NC,1] \rightsquigarrow [NC,1]$
$t_2 \Rightarrow [NC,1] \rightsquigarrow [C,0]$
$t_3 \Rightarrow [C,0] \rightsquigarrow [C,0]$
$t_4 \Rightarrow [C,0] \rightsquigarrow [NC,1]$
$t_5 \Rightarrow [NC,1] \rightsquigarrow [NC,1]$
$t_6 \Rightarrow [NC,1] \rightsquigarrow [C,0]$
$t_7 \Rightarrow [C,0] \rightsquigarrow [NC,1]$

شبکه بازگشتی ما به صورت زیر است که برای آن داریم:



$$h_t = f(Wh_{t-1} + Ux_t + b_h)$$

$$y_t = f(Wh_t + b_y)$$

$$\begin{cases} h_t \to 3\times 1 \\ x_t \to 2\times 1 \end{cases}$$

$$\begin{cases} W \to 3\times 3 \\ U \to 3\times 2 \\ b_h \to 3\times 1 \\ V \to 1\times 3 \\ b_y \to 1\times 1 \end{cases}$$

حال سعی می‌کنیم رابطه بین y_t و h_t را بیابیم؛ اگر به صورت y_t و h_t نگاه کنیم، متوجه می‌شویم که $h_2^{(t)}$ همواره ۲ است، اما هرگاه h_1 و h_3 صفر باشند، y_t ۲ شده است و برعکس.

پس می‌توانیم رابطه‌ای به شکل زیر بنویسیم که بایاس را نیز بیابیم و فراموش نکنیم:

$$a_t = \left(1-h_1^{(t)}\right) + h_2^{(t)} + \left(1-h_3^{(t)}\right) + b \quad , \quad f(a_t) = y_t$$

حال به ازای تمامی t ها داریم:

$$y_t = \begin{cases} 1 & \text{if } a_t > 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{(step)}$$

t	a_t	y_t
1	$3+b$	1
2	$1+b$	0
3	$1+b$	0
4	$3+b$	1
5	$3+b$	1
6	$1+b$	0
7	$3+b$	1

در تمامی حالاتی که $y_t = 1$ شده است $a_t = 3+b$ بوده است:

$$3+b > 0 \Rightarrow b > -3 \Rightarrow \boxed{b = -2,8}$$

$$\Rightarrow a_t = 1-h_1+h_2+1-h_3-2,8$$

$$\Rightarrow a_t = -h_1+h_2-h_3-0,2 \Rightarrow y_t = f(-h_1+h_2-h_3-0,2)$$

$$\Rightarrow y_t = f\left([-1,1,-1]\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} - 0,2\right) \Rightarrow \boxed{\begin{aligned} V &= [-1 \quad 1 \quad -1] \\ b_y &= -0,2 \end{aligned}}$$

در ادامه به سراغ محاسبه U و W ی رویم، مشابه حالت قبل، تمامی h_2 ها ۱ هستند، همچنین $h_1 = h_3$ است. پس برای راحتی در پیدا کردن نگاشت فقط h_3 و a_2 را به عنوان y_t در نظر می‌گیریم.

اگر $x_1(t)$ ، $x_2(t)$ و $h_3(t-1)$ را با هم جمع کنیم، مشاهده می‌شود که جمع این سه عدد ارتباطی با $h_1(t)$ دارد. به طوری که جمع این سه بیت مشخص می‌کند که چه چیزی print شود و چه رقم carry برای state بعد خواهیم داشت. یا به زبانی دیگر الرجمع آنها ۱ بشود، $y_t=1$ و $h_1(t)=0$ خواهد شد، اگر جمع آن‌ها ۲ بشود، $y_t=0$ و $h_1(t)=1$ خواهد شد. پس خواهیم داشت:

$$h_1(t) = f(d_t), \quad d_t = h_3(t-1) + x_1(t) + x_2(t) + b$$

برای هر t داریم:

t	d_t	$h_1(t)$
1	$1+b$	0
2	$0+b$	1
3	$2+b$	1
4	$1+b$	0
5	$1+b$	0
6	$2+b$	1
7	$1+b$	0

$$h_1(t) \begin{cases} 1 & \text{if } d_t > 0 \\ 0 & \text{otherwise} \end{cases}$$

به طوری که d_t برای هر t نیز داریم:

$$2+b>0 \Rightarrow b > -2 \Rightarrow \boxed{b > -1.8}$$

$$\Rightarrow h_1(t) = f\left(x_1(t) + x_2(t) + h_3(t-1) - 1.8\right)$$

$$\Rightarrow h_3(t) = h_1(t)$$

$$\Rightarrow h_1(t) = f\left(\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} h_1(t-1) \\ h_2(t-1) \\ h_3(t-1) \end{bmatrix} + \begin{bmatrix} 1 & 1 \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} - 1.8\right) = h_3(t)$$

مشابه h_1، برای $h_2(t)$ خواهیم داشت:

$$h_2(t) = f(c_t), \quad c_t = h_2(t-1) + x_1(t) + x_2(t) + b$$

t	c_t	$h_2(t)$
1	$2+b$	1
2	$3+b$	1
3	$2+b$	1
4	$1+b$	1
5	$2+b$	1
6	$3+b$	1
7	$1+b$	1

برای همه t داریم:

$3+b>0$

$2+b>0$

$1+b>0 \Rightarrow b>-1 \Rightarrow b<-0.8$

$\Rightarrow c_t > h_2(t-1)+a_1(t)+a_2(t)-0.5$

$$\Rightarrow h_2(t)=f\left(\begin{bmatrix}0 & 1 & 0\end{bmatrix}\begin{bmatrix}h_1(t-1)\\ h_2(t-1)\\ h_3(t-1)\end{bmatrix}+\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}a_1(t)\\ a_2(t)\end{bmatrix}-0.8\right)$$

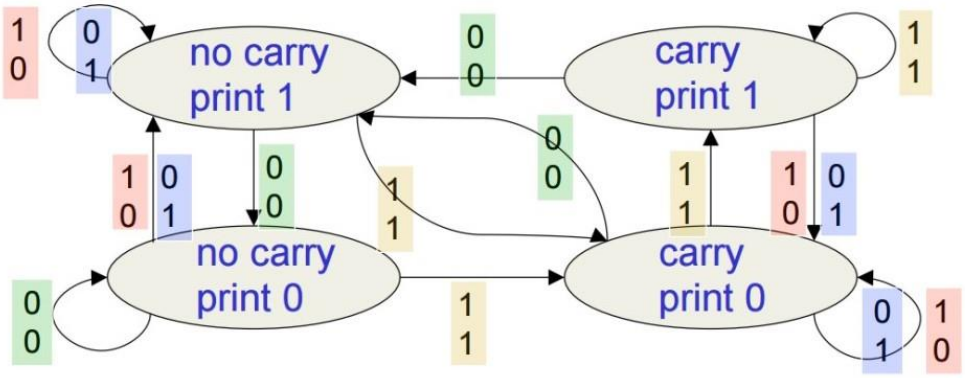
درنهایت داریم:

$$h(t)=f\left(\begin{bmatrix}0 & 0 & 1\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}h_1(t-1)\\ h_2(t-1)\\ h_3(t-1)\end{bmatrix}+\begin{bmatrix}1 & 1\\ 1 & 1\\ 1 & 1\end{bmatrix}\begin{bmatrix}a_1(t)\\ a_2(t)\end{bmatrix}+\begin{bmatrix}-1.8\\ -0.8\\ -1.8\end{bmatrix}\right)$$

$$\Rightarrow W=\begin{bmatrix}0 & 0 & 1\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$U=\begin{bmatrix}1 & 1\\ 1 & 1\\ 1 & 1\end{bmatrix}$$

$$b_h=\begin{bmatrix}-1.8\\ -0.8\\ -1.8\end{bmatrix}$$



سوال ۴:

الف)

بگذارید با یک مثال از شبکه $GPT3$ جلو برویم. ورودی‌ها یا همان xهای این مدل بردارهایی با اندازه‌ای تقریبا برابر ۵۰هزار هستند که از یک $word\ embedding$ عبور کرده و به فضای ۱۲هزار بعدی می‌روند. D در این مدل همان ۱۰هزار است. یعنی d اندازه هر x ورودی است. از طرفی w بیان می‌کند که ما چندمین ورودی هستیم. یعنی اگر برای مثال مدل GPT ما جمله در ورودی بگیرد، به طوری که هر x نشانگر یک کلمه باشد، w بیان می‌کند که x چندمین کلمه در جمله است. k هم می‌گوید که المان چندم P هستیم. برای هر x یک P دارم.

ب)

سوال 4 :

(ب)

$$M.\begin{bmatrix}\sin\omega_k t\\ \cos\omega_k t\end{bmatrix}=\begin{bmatrix}\sin(\omega_k(t+\phi))\\ \cos(\omega_k(t+\phi))\end{bmatrix}$$

$$\begin{bmatrix}a & b\\ c & d\end{bmatrix}\begin{bmatrix}\sin\omega_k t\\ \cos\omega_k t\end{bmatrix}=\begin{bmatrix}\sin(\omega_k(t+\phi))\\ \cos(\omega_k(t+\phi))\end{bmatrix}$$

$$\Rightarrow\begin{cases}a\sin(\omega_k t)+b\cos(\omega_k t)=\sin(\omega_k(t+\phi)) & \times\cos(\omega_k t)\\ c\sin(\omega_k t)+d\cos(\omega_k t)=\cos(\omega_k(t+\phi)) & \times\sin(\omega_k t)\end{cases}$$

$$\Rightarrow\begin{cases}(a\sin\omega_k t)(\cos\omega_k t)+b\cos^2(\omega_k t)=\sin(\omega_k(t+\phi))\cos(\omega_k t)\\ c\sin^2(\omega_k t)+(d\cos\omega_k t)(\sin\omega_k t)=\cos(\omega_k(t+\phi))\sin(\omega_k t)\end{cases}$$

$$\Rightarrow\begin{cases}\frac{1}{2}a\sin(2\omega_k t)+\frac{1}{2}b(1+\cos(2\omega_k t))=\frac{1}{2}[\sin(2\omega_k t+\phi)+\sin(\omega_k\phi)]\\ \frac{1}{2}c(1-\cos(2\omega_k t))+\frac{1}{2}d\sin(2\omega_k t)=\frac{1}{2}[\sin(2\omega_k t+\phi)-\sin(\omega_k\phi)]\end{cases}$$

$$\begin{cases}a\sin(2\omega_k t)+b\cos(2\omega_k t)+b=\sin(2\omega_k t+\phi)+\sin(\omega_k\phi)\\ -c\cos(2\omega_k t)+d\sin(2\omega_k t)+c=\sin(2\omega_k t+\phi)-\sin(\omega_k\phi)\end{cases}$$

$$\Rightarrow\begin{cases}b=\sin\omega_k\phi\\ -c=-\sin\omega_k\phi\end{cases}\Rightarrow b=-c$$

$$\Rightarrow\begin{cases}a\sin(2\omega_k t)+\sin(\omega_k\phi)\cos(2\omega_k t)=\sin(2\omega_k t+\phi)\\ \sin(\omega_k\phi)\cos(2\omega_k t)+d\sin(2\omega_k t)=\sin(2\omega_k t+\phi)\end{cases}\Rightarrow a=d=\cos(\omega_k\phi)$$

$$\Rightarrow M = \begin{bmatrix} \cos(\omega_k \phi) & \sin(\omega_k \phi) \\ -\sin(\omega_k \phi) & \cos(\omega_k \phi) \end{bmatrix}$$

M یک ماتریس متعامد است:

$$|M| = \cos^2 + \sin^2 = 1$$

$$M M^T = \begin{bmatrix} \cos & \sin \\ -\sin & \cos \end{bmatrix} \begin{bmatrix} \cos & -\sin \\ \sin & \cos \end{bmatrix} = 1$$

$$[\cos \quad \sin] \underset{\text{dot}}{\cdot} [-\sin \quad \cos] = -\sin\cos + \sin\cos = 0$$

$$P_{t+k} = T^{(k)} P_t$$

$T^{(k)}$ یک ماتریس مربعی خواهد بود که $d \times d$ است:

$$\begin{bmatrix} \cos(\omega_1 \phi) & \sin(\omega_1 \phi) & 0 & 0 & \cdots & 0 & 0 \\ -\sin(\omega_1 \phi) & \cos(\omega_1 \phi) & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos(\omega_2 \phi) & \sin(\omega_2 \phi) & \cdots & 0 & 0 \\ 0 & 0 & -\sin(\omega_2 \phi) & \cos(\omega_2 \phi) & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos(\omega_{\frac{d}{2}} \phi) & \sin(\omega_{\frac{d}{2}} \phi) \\ 0 & 0 & 0 & 0 & \cdots & -\sin(\omega_{\frac{d}{2}} \phi) & \cos(\omega_{\frac{d}{2}} \phi) \end{bmatrix}$$