دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده مهندسی برق و کامپیوتر

تمرین شماره: ۲

یادگیری عمیق با کاربرد در تصویر و صوت

نام و نام‌خانوادگی: مرتضی ملکی‌نژاد شوشتری

شماره دانشجویی: ۸۱۰۱۰۴۲۵۶

نیم‌سال دوم سال تحصیلی ۴۰۵-۴۰۴

فهرست مطالب

۱ برچسب‌گذاری دنباله ۴
۱.۱ بررسی پایگاه داده ۴
۲.۱ بررسی collate fn ۵
۳.۱ آموزش شبکه ۶
۱.۳.۱ مقایسه مدل‌ها ۸

۲ VIT ۹
۱.۲ آموزش شبکه ۹
۲.۲ بررسی حالت‌های دیگر ۱۰

فهرست تصاویر

۱.۱ نمودارهای خطا و دقت مدل RNN همراه با گزارش دسته‌بندی ۶
۲.۱ نمودارهای خطا و دقت مدل LSTM همراه با گزارش دسته‌بندی ۷
۳.۱ نمودارهای خطا و دقت مدل GRU همراه با گزارش دسته‌بندی ۷
۴.۱ نمودارهای خطا و دقت مدل‌ها درکنار هم ۸

۱.۲ نمودار خطا و دقت مدل ۱۰
۲.۲ نمودار خطا و دقت مدل با بُعد embedding برابر ۶۴ . . . . . ۱۱
۳.۲ نمودار خطا و دقت مدل با بُعد embedding برابر ۲۵۶ . . . . . ۱۱
۴.۲ نمودار خطا و دقت مدل با ۲ لایه‌ی Transformer ۱۱
۵.۲ نمودار خطا و دقت مدل با ۸ لایه‌ی Transformer ۱۲
۶.۲ نمودار خطا و دقت مدل با اندازه‌ی patch برابر ۲×۲ . . . . . ۱۲
۷.۲ نمودار خطا و دقت مدل با اندازه‌ی patch برابر ۸×۸ . . . . . ۱۲
۸.۲ نمودار خطا و دقت مدل با اندازه‌ی تصویر ۶۴×۶۴ . . . . . ۱۳
۹.۲ نقشه توجه برای شکل اول ۱۴
۱۰.۲ نقشه توجه برای شکل دوم ۱۵
۱۱.۲ نقشه توجه برای شکل اول با مدل vitb16 . . . . . ۱۶
۱۲.۲ نقشه توجه برای شکل دوم با مدل vitb16 . . . . . ۱۷

فهرست جداول

۱.۱ سطر اول مجموعه داده آموزش ۵

۱.۲ خطا و دقت مدل روی مجموعه‌داده‌های مختلف ۱۰

۲.۲ خطا، دقت و میزان زمان آموزش مدل‌های مختلف ۱۳

سوال ۱

برچسب‌گذاری دنباله

۱.۱ بررسی پایگاه داده

آ) یکی از سطرهای مجموعه داده [۱] در جدول ۱.۱ آمده است. می‌توان دید علاوه بر NER Tag مجموعه داده دو ستون دیگر به اسم‌های pos tag و chunk tag نیز دارد.

ب) pos tag عملا نقش گرامری هر توکن را مشخص می‌کند. برای مثال NNP یک اسم مشخصه [۲] است. اهمیت این برچسب در NER از این رو است که ما به دنبال تشخیص کلاس اسامی (و به طور خاص اسامی مشخصه) هستیم. پس می‌توان افعال را درنظر نگرفت و اگر یک کلمه فعل بود از آن صرفنظر کرد.

chunk tag عملا تعمیم pos tag است. یعنی سعی می‌کند جمله را بجای در سطح توکن در سطح عبارت [۳] بسنجد. و توکن‌ها را باهم دسته می‌کند.

[1] Dataset
[2] Proper noun
[3] phrase

جدول ۱.۱: سطر اول مجموعه داده آموزش

Token	EU	rejects	German	call	to	boycott	British	lamb	.
Pos Tag	NNP	VBZ	JJ	NN	TO	VB	JJ	NN	.
Chunk Tag	B-NP	B-VP	B-NP	I-NP	B-VP	I-VP	B-NP	I-NP	O
NER Tag	B-ORG	O	B-MISC	O	O	O	B-MISC	O	O

ج) کد مربوط به vocab پیاده‌سازی شد. این کد به ترتیب توکن‌ها را بررسی می‌کند و تعداد تکرار هرکدام که به مقدار cutoff رسید، آن را به لیست vocab اضافه می‌کند.

د)

```
vocab = Vocab(ds_train)
print(f"{len(vocab) = }")
print(f"Token 0 is '{vocab.id_to_token(0)}'")
print("'quick' token id is ", vocab.token_to_id("quick"))
```

نمونه کد ۱: موارد خواسته‌شده بخش «د»

```
len(vocab) = 4913
Token 0 is 'to'
'quick' token id is 4813
```

۲.۱ بررسی collate fn

آ) هرچقدر بیشتر یک جمله پد شود، باعث می‌شود احتمال توجه مدل به توکن‌های معنی‌دار آن کم‌تر شود. زیرا نسبت توکن‌های معنی‌دار به طول جمله کم می‌شود. ولی پد کردن برای محاسبه اجتناب‌ناپذیر است. حال اگر در یک دسته [۴] این کار انجام شود، هم محاسبه امکان‌پذیر است و هم پد کردن به میزان کم‌تری اتفاق می‌افتد.

ب)

[4] batch

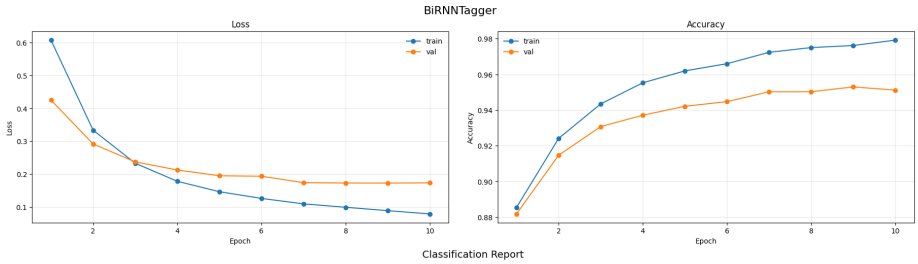
```python
dataloader = get_dataloader(ds_train)
first_batch = next(iter(dataloader))
tokens = [item[0] for item in first_batch]
tokens = np.array(tokens)
print(tokens.shape)
```

نمونه کد ۲: شکل بچ اول توکن‌ها

```
(32, 47)
```

۳.۱ آموزش شبکه

آ) مدل RNN:

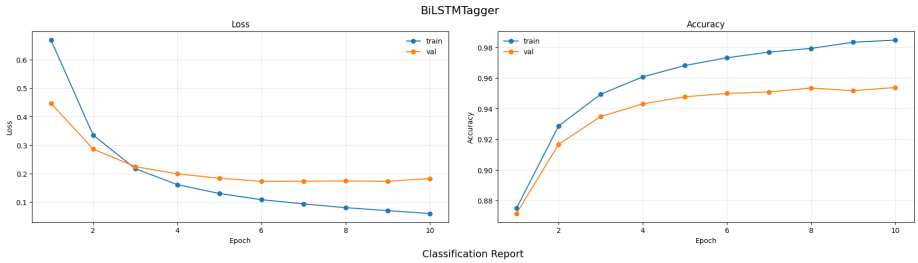


Classification Report

Class	Precision	Recall	F1	Support
B-ORG	0.8172	0.6868	0.7464	1341
O	0.9753	0.9844	0.9798	42759
B-MISC	0.8577	0.7451	0.7974	922
B-PER	0.7941	0.8268	0.8101	1842
I-PER	0.7995	0.9028	0.8480	1307
B-LOC	0.8797	0.8356	0.8571	1837
I-ORG	0.8441	0.6778	0.7518	751
I-MISC	0.7839	0.6185	0.6914	346
I-LOC	0.7471	0.7588	0.7529	257
accuracy			0.9512	51362
macro avg	0.8332	0.7818	0.8039	51362
weighted avg	0.9503	0.9512	0.9502	51362

شکل ۱.۱: نمودارهای خطا و دقت مدل RNN همراه با گزارش دسته‌بندی

ب) مدل LSTM:

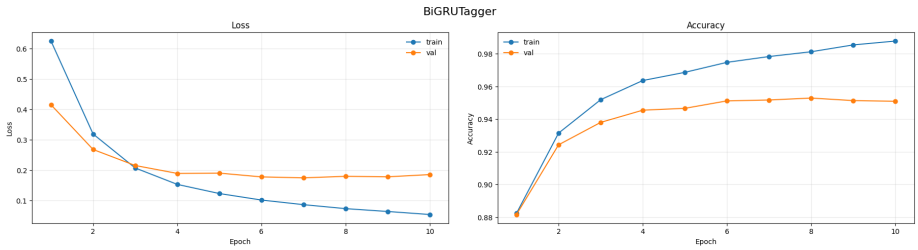


Classification Report

Class	Precision	Recall	F1	Support
B-ORG	0.8534	0.7122	0.7764	1341
O	0.9659	0.9937	0.9796	42759
B-MISC	0.8614	0.7213	0.7851	922
B-PER	0.8920	0.7758	0.8298	1842
I-PER	0.9008	0.8133	0.8548	1307
B-LOC	0.9127	0.8024	0.8540	1837
I-ORG	0.8506	0.6977	0.7666	751
I-MISC	0.7673	0.6098	0.6795	346
I-LOC	0.9026	0.6848	0.7788	257
accuracy			0.9537	51362
macro avg	0.8785	0.7568	0.8116	51362
weighted avg	0.9515	0.9537	0.9516	51362

شکل ۲.۱: نمودارهای خطا و دقت مدل LSTM همراه با گزارش دسته‌بندی

ج) مدل GRU:



Classification Report

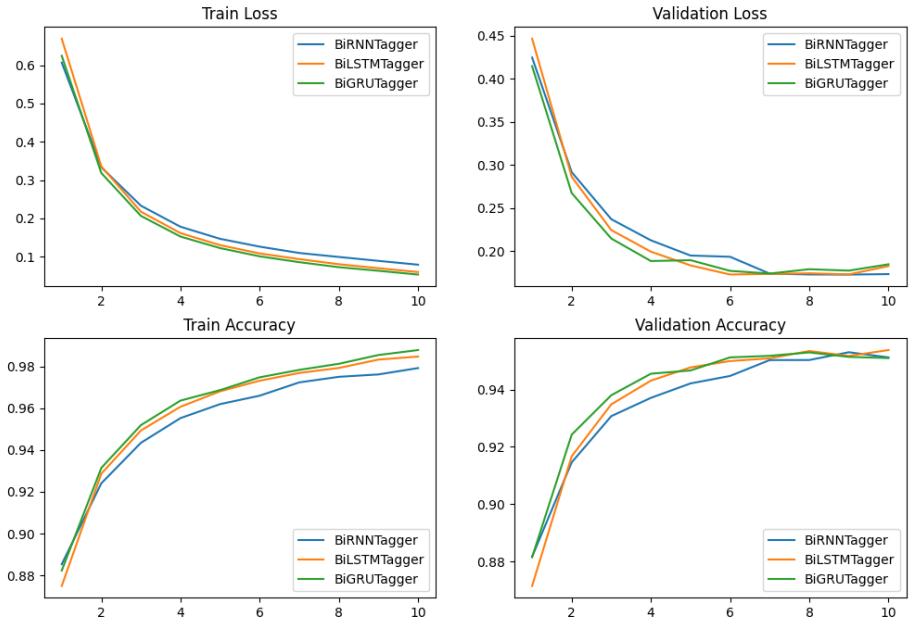
Class	Precision	Recall	F1	Support
B-ORG	0.7005	0.7621	0.7300	1341
O	0.9736	0.9860	0.9798	42759
B-MISC	0.8656	0.7332	0.7939	922
B-PER	0.8456	0.7997	0.8220	1842
I-PER	0.8803	0.8271	0.8529	1307
B-LOC	0.9152	0.8046	0.8563	1837
I-ORG	0.7332	0.7430	0.7381	751
I-MISC	0.8077	0.6069	0.6931	346
I-LOC	0.8962	0.7393	0.8102	257
accuracy			0.9511	51362
macro avg	0.8464	0.7780	0.8085	51362
weighted avg	0.9505	0.9511	0.9503	51362

شکل ۳.۱: نمودارهای خطا و دقت مدل GRU همراه با گزارش دسته‌بندی

۱.۳.۱ مقایسه مدل‌ها

در تصویر ۴.۱ می‌توان دقت و خطای مدل‌ها را در کنار هم دید. روی داده آموزش، دقت مدل RNN از دو مورد دیگر بدتر است و این به دلیل این است که در مدل RNN روابط بلندمدت کلمات حفظ نمی‌شوند. اما دقت مدل‌های GRU و LSTM به‌هم نزدیک است که این مطابق انتظار است. مهم‌ترین آورده GRU نسبت به LSTM در سادگی محاسبات است نه در دقت.

روی داده صحت‌سنجی، می‌توان دید در دورهای اول نیز، مدل RNN کندتر همگرا می‌شود. اما دورهای آخر، دیگر دقت افزایش پیدا نمی‌کند و هر سه مدل به نوعی اشباع شده‌اند.



شکل ۴.۱: نمودارهای خطا و دقت مدل‌ها درکنار هم

سوال ۲

VIT

۱.۲ آموزش شبکه

مطابق خواسته سوال، شبکه vit پیاده‌سازی شد. شبکه چند بلوک دارد. ابتدا یک بلوک برای patching embedding پیاده‌سازی شد. وظیفه این بلوک این است که تصویر را به patch تقسیم کند (هر patch مشابه یک کلمه در جمله است.) و بعد این patch ها را از یک لایه خطی عبور دهد. یک لایه کانولوشن [۱] که در آن سایز با گام stride برابر است، عملا همین کار را می‌کند.

ماژول بعدی مربوط به مکانیزم Attention است. برای هر توکن (Patch) باید مقدار توجه آن‌را با استفاده از ماتریس‌های k ، q و v درآورد. این ماتریس‌ها در ابتدا جدا حساب می‌شدند، ولی پس از کمک گرفتن از هوش مصنوعی، برای بهبود سرعت، هر سه این ماتریس‌ها باهم در یک لایه خطی بزرگ حساب می‌شوند و سپس جدا می‌شوند. خروجی این بلوک، حاصل ضرب مقدار توجه [۲] در مقدار [۳] است که از یک لایه خطی نیز گذر داده می‌شود.

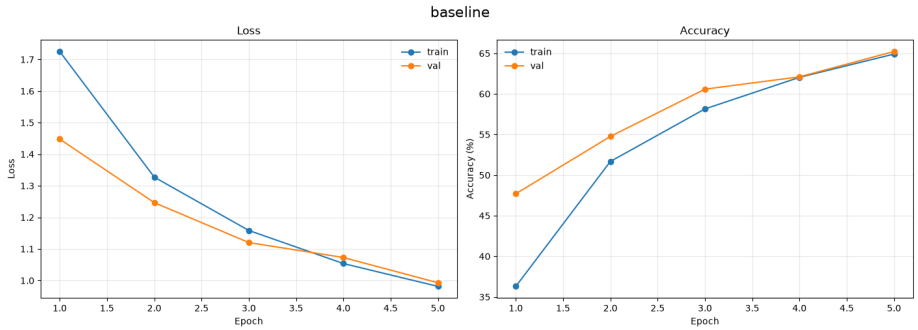
سپس یک بلوک MLP پیاده‌سازی شده که توضیح خاصی نیاز ندارد. تنها نکته آن این است که در مقاله ViT استفاده از تابع فعال‌ساز GELU پیشنهاد شده و تنها بین دو لایه خطی، فعالساز وجود دارد و پس از

[1] Convolution
[2] Attention
[3] value

لایه خطی دوم، تابع فعال‌سازی وجود ندارد.

درنهایت بلوک‌های قبل در یک ماژول Transformer سرهم می‌شوند. این بلوک Transformer فقط برای رمزگذار [۴] است.

شبکه ViT در عمل، پس از انجام Patch Embedding و اضافه کردن CLS embedding و Pos embedding خروجی را به لایه‌های transformer می‌دهد و درنهایت روی بخش CLS یک لایه خطی اجرا می‌کند تا نتیجه طبقه‌بندی را خروجی دهد.



شکل ۱.۲: نمودار خطا و دقت مدل

جدول ۱.۲: خطا و دقت مدل روی مجموعه‌داده‌های مختلف

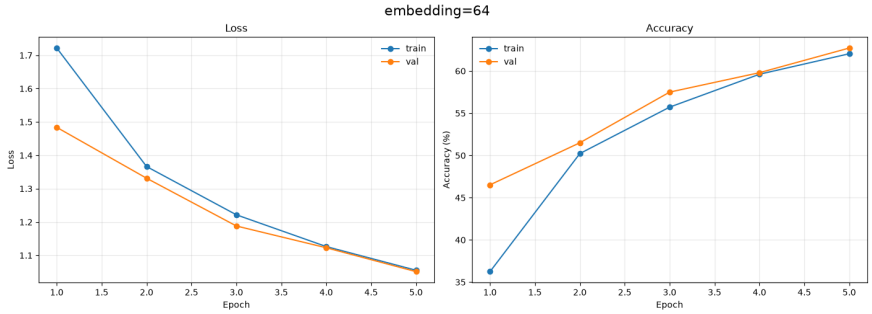
Dataset	Acc	Loss
Train	64.90%	0.9815
Val	65.22%	0.9931
Test	62.53%	1.0353

۲.۲ بررسی حالت‌های دیگر

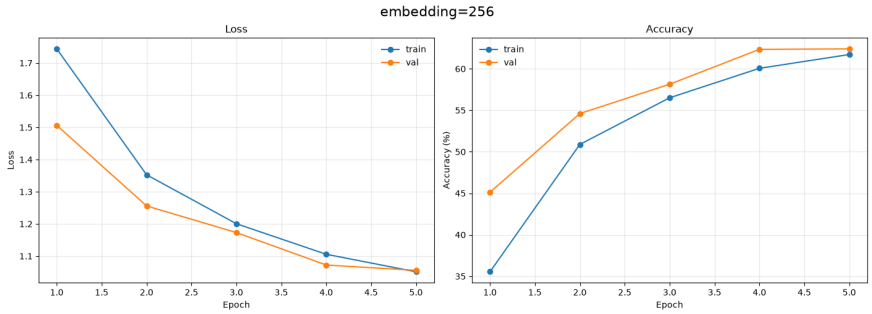
آ) خواسته‌های سوال در نوتبوک model_eval پیاده‌سازی شده‌اند.

در گزارش، نمودارهای مدل‌های مختلف آمده است که به دلیل اینکه سوال مقایسه آن‌ها را نخواسته، و دقت نهایی در جدول ۲.۲ آمده است، صرفا نتایج جدول مقایسه می‌شوند.

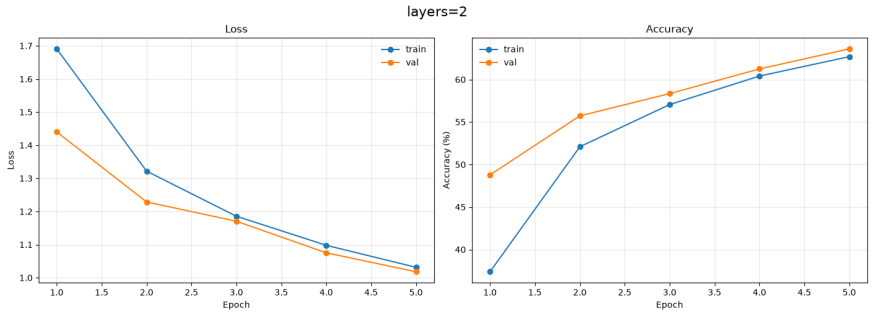
[4] Encoder



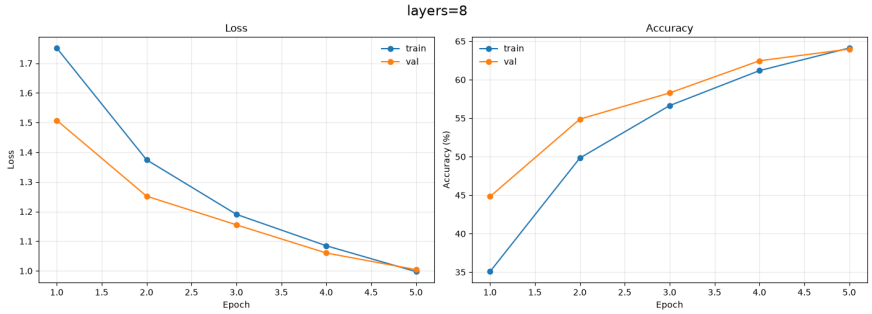
شکل ۲.۲: نمودار خطا و دقت مدل با بُعد embedding برابر ۶۴



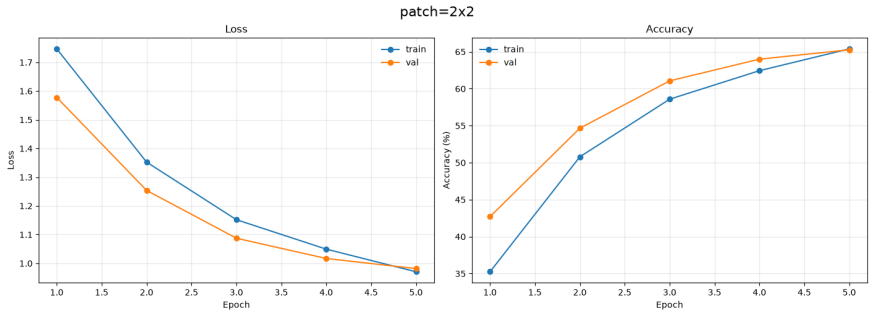
شکل ۳.۲: نمودار خطا و دقت مدل با بُعد embedding برابر ۲۵۶



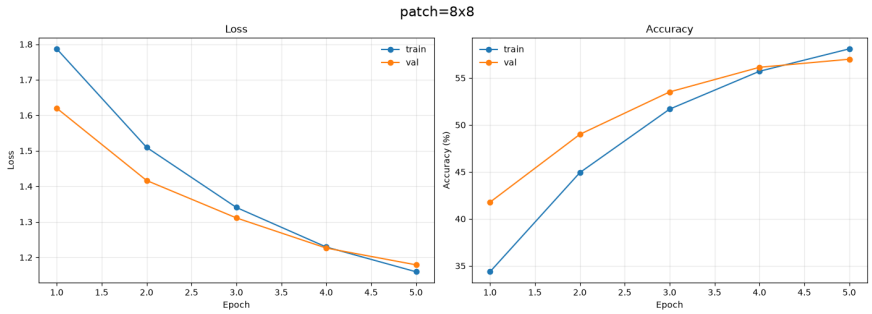
شکل ۴.۲: نمودار خطا و دقت مدل با ۲ لایه‌ی Transformer



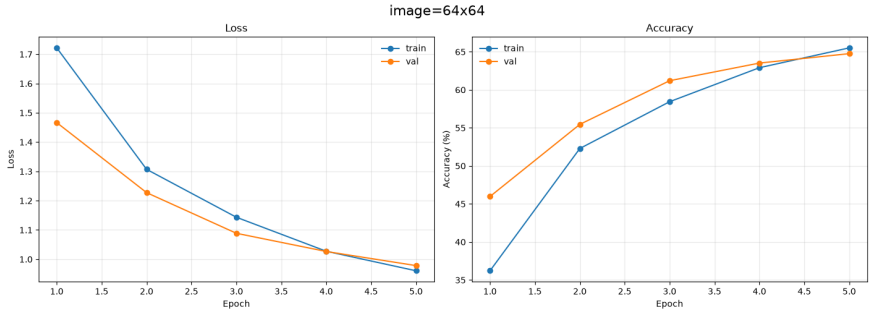
شکل ۵.۲: نمودار خطا و دقت مدل با ۸ لایه‌ی Transformer



شکل ۶.۲: نمودار خطا و دقت مدل با اندازه‌ی patch برابر ۲×۲



شکل ۷.۲: نمودار خطا و دقت مدل با اندازه‌ی patch برابر ۸×۸



شکل ۸.۲: نمودار خطا و دقت مدل با اندازه‌ی تصویر ۶۴x۶۴

همانطور که در جدول ۲.۲ می‌توان دید، بیش‌برازش زیادی در هیچ کدام از مدل‌ها رخ نداده است. مدلی که با تصاویر بزرگ‌شده و یا patch کوچک‌تر آموزش دیده شده، دقت بهتری دارد ولی زمان آموزش آن‌ها نیز به شدت بیشتر است. بطوری که اگر مدل‌های دیگر را با دورهای بیشتر آموزش داد احتمالا بتوان با همان زمان به دقت بهتری رسید. البته باید توجه داشت که احتمالا دورهای بیشتر منجر به بیش‌برازش شود و باید برای این مسئله راه‌حلی پیدا کرد.

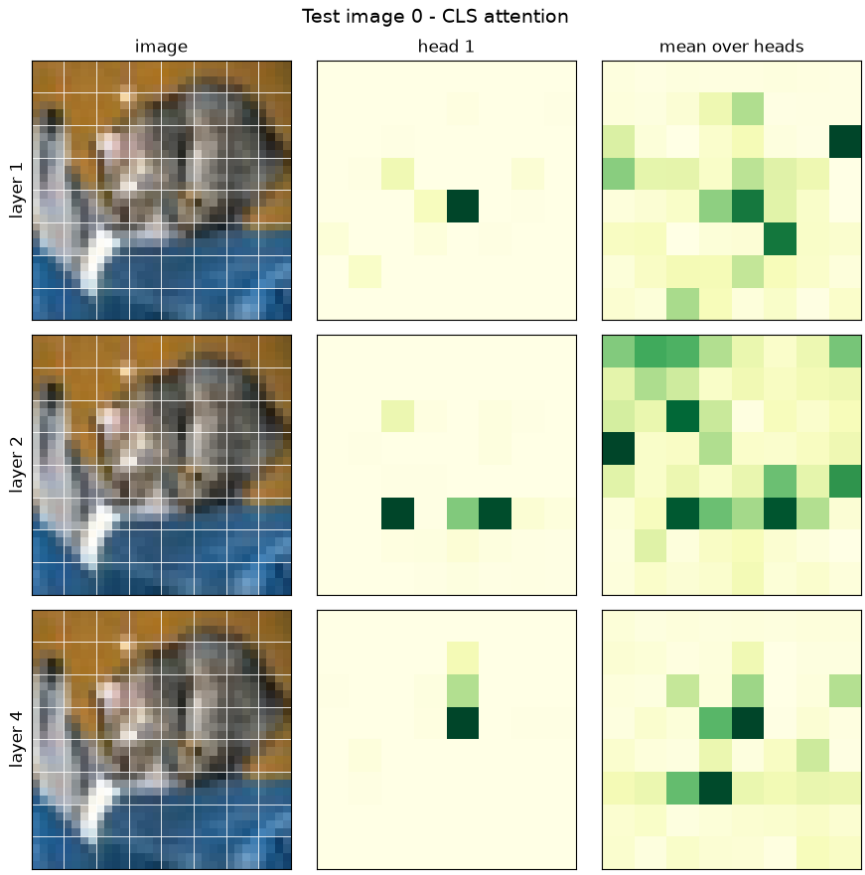
جدول ۲.۲: خطا، دقت و میزان زمان آموزش مدل‌های مختلف

Experiment	Train Acc	Train Loss	Val Acc	Val Loss	Test Acc	Test Loss	Duration (s)
baseline	64.90%	0.9815	65.22%	0.9931	62.53%	1.0353	85.8
embedding=64	62.04%	1.0554	62.72%	1.0513	60.84%	1.0794	59.6
embedding=256	61.72%	1.0513	62.40%	1.0557	62.10%	1.0545	147.8
layers=2	62.68%	1.0315	63.60%	1.0180	62.37%	1.0428	43.2
layers=8	64.12%	0.9974	63.98%	1.0041	62.78%	1.0407	156.6
patch=2x2	65.39%	0.9706	**65.26%**	0.9817	**64.20%**	0.9915	860.4
patch=8x8	58.10%	1.1596	57.00%	1.1791	56.48%	1.2031	20.5
image=64x64	**65.52%**	**0.9606**	64.76%	**0.9783**	63.99%	**0.9892**	702.3

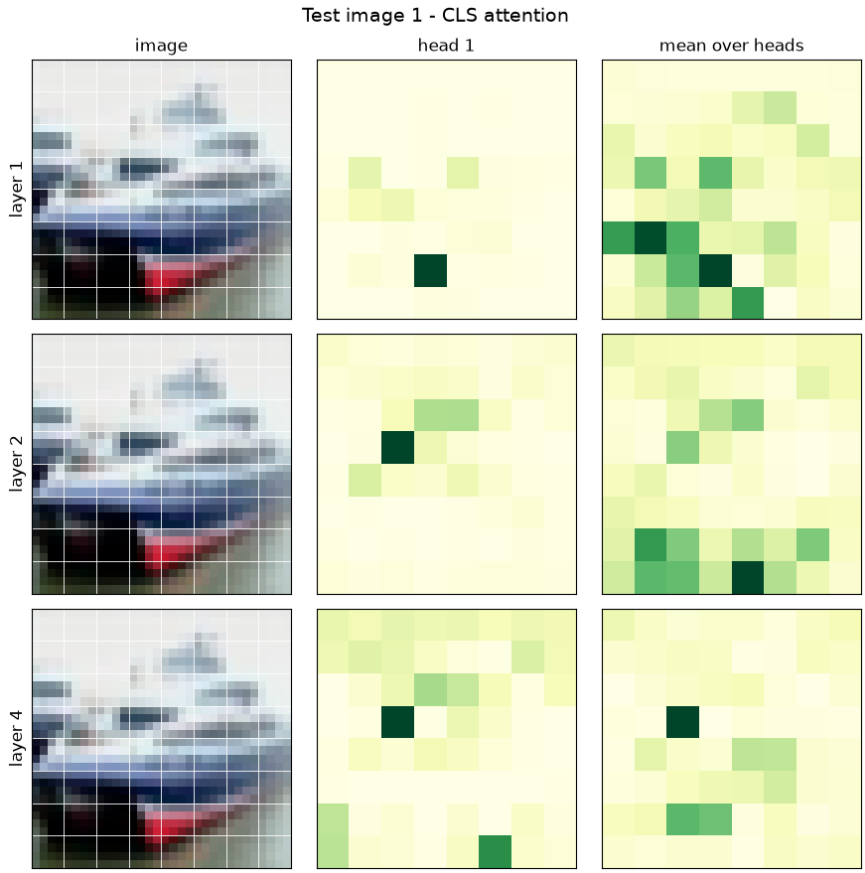
ب) خواسته‌های سوال در نوتبوک plot_attention پیاده‌سازی شده‌اند.

با دیدن تصاویر ۹.۲ و ۱۰.۲ می‌توان دید که head اول فقط روی یک بخش خاص از تصویر (مثلا یک لبه یا یک بخش کشتی) تمرکز کرده است. ولی با میانگین گرفتن از همه head ها، می‌توان دید مدل به بخش‌های مختلف شکل توجه کرده است. البته همچنان در جاهایی بجای شکل روی پس‌زمینه تمرکز کرده است که این مورد خوب نیست.

ج) نقشه توجه مدل vitb16 در تصاویر ۱۱.۲ و ۱۲.۲ آمده است. اگر این تصاویر را با تصاویر ۹.۲ و ۱۰.۲ مقایسه کنیم، می‌بینیم در لایه‌های اول، نقشه توجه به نسبت پراکنده است و مدل روی بخش‌های



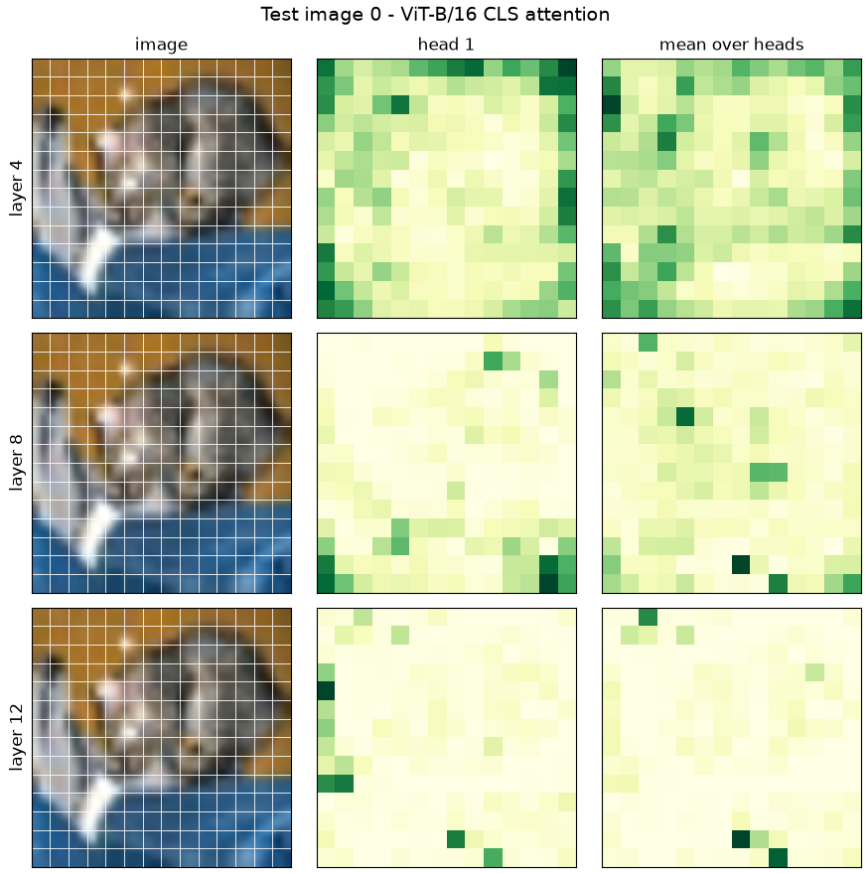
شکل ۹.۲: نقشه توجه برای شکل اول



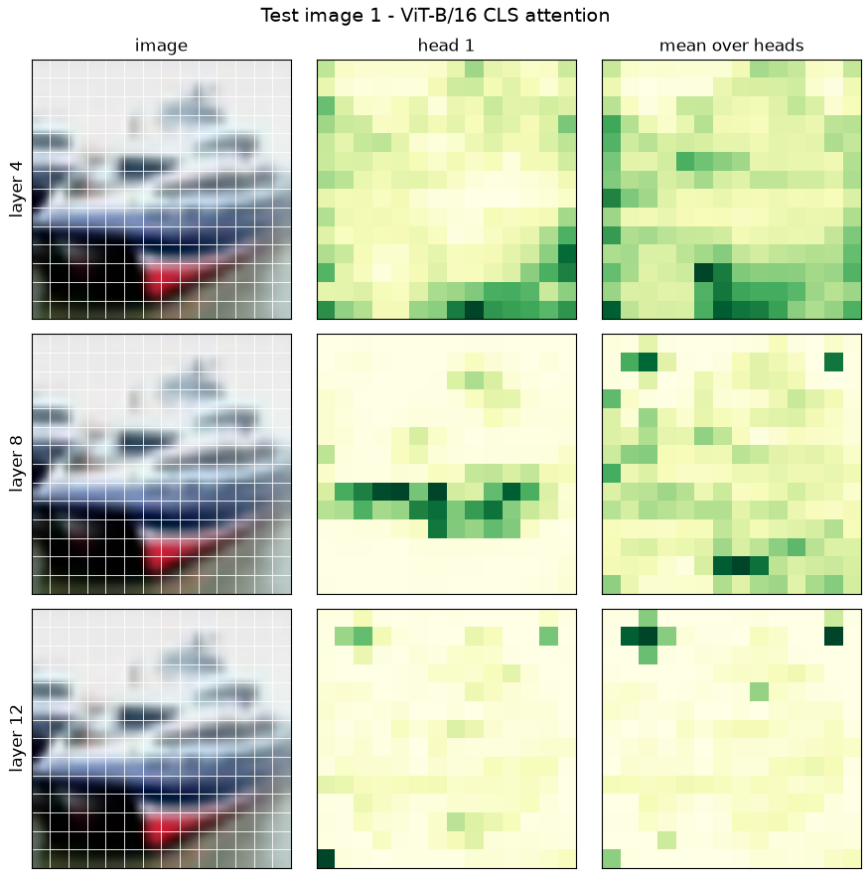
شکل ۱۰.۲: نقشه توجه برای شکل دوم

مختلفی از تصویر تمرکز کرده است. در مقابل مدل ما از اول روی بخش‌های اصلی تصویر تمرکز کرده است. این موضوع می‌تواند به این علت باشد که مدل vitb16 را روی داده cifar آموزش ندیده است و finetune نشده است و در مراحل اول سعی در بررسی بخش‌های مختلف تصویر دارد.

با عمیق‌تر شدن لایه‌ها می‌توان دید که مدل به مرور، اثر بخش‌های مختلف را خنثی می‌کند و روی نقاط محدودی از تصویر تمرکز می‌کند. اگر مجدد لایه‌های آخر را باهم مقایسه کنیم، می‌توان دید که مدل finetune شده، روی بخش‌های مختلفی از تصویر اصلی دقت می‌کند. پس احتمالا مدل مقاوم‌تری است.



شکل ۱۱.۲: نقشه توجه برای شکل اول با مدل vitb16



شکل ۱۲.۲: نقشه توجه برای شکل دوم با مدل vitb16