

ادیب رضائی - امیرمحمد خسروی	نام و نام خانوادگی
810198386 - 810198401	شماره دانشجویی
۱۴۰۱.۱۲.۲۲	تاریخ ارسال گزارش

به نام خدا
 دانشگاه تهران
 دانشکده مهندسی
 برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق
 تمرین سوم

فهرست

1	پاسخ ۱. توصیف عکس
1	پیش پردازش داده
5	مدل Freeze شده
8	مدل Unfreeze شده
12	پاسخ ۲ - تشخیص اندیشه

شكل‌ها

.1.1

جدول‌ها

پاسخ 1. توصیف عکس

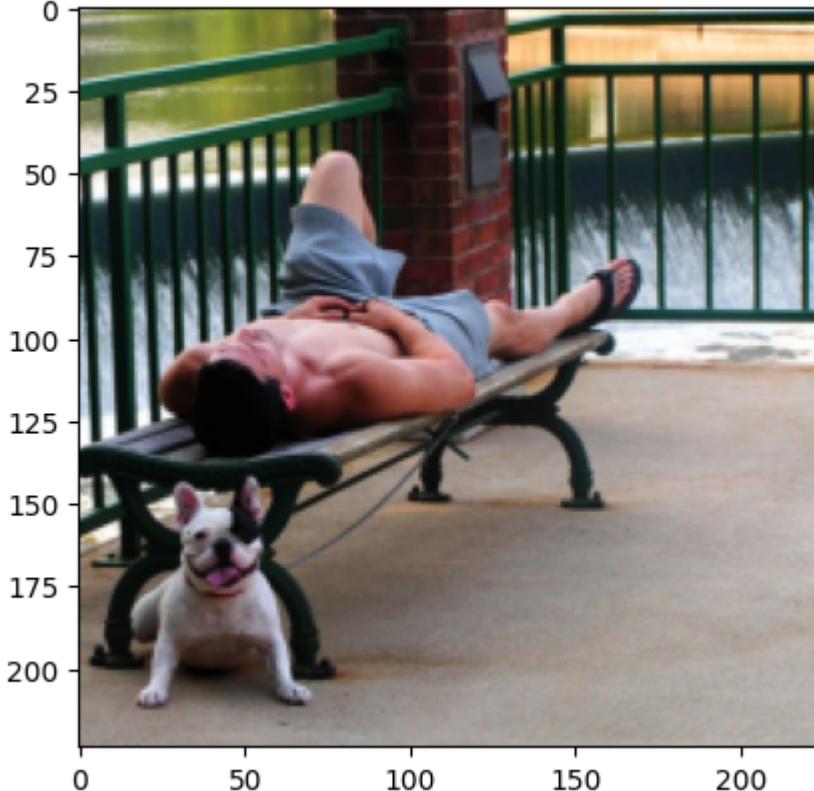
در این سوال به کپشن زدن روی عکس ها میپردازیم.

پیش پردازش داده

در این مرحله باید کپشن های عکس هارا tokenize کنیم. بدین منظور یک کلاس دیتاست به نام CustomDataset ایجاد میکنیم و در آن از کلاس Dictionary استفاده میکنیم. وظیفه کلاس Dictionary گرفتن جملات، حذف علامت های نگارشی خاص، و tokenize کردن آن هاست. همچنین به هر توکن ایندکسی تعلق میگیرد تا در vectorize کردن آن به کار گرفته شود. هر آیتم این کلاس شامل عکس مورد نظر و کپشن شده عددی است.

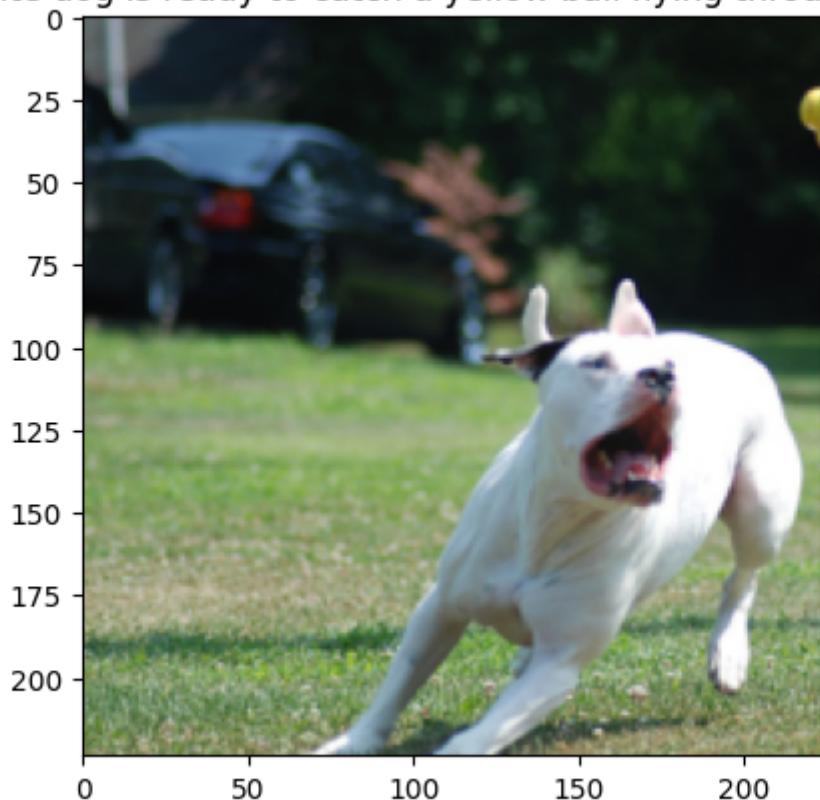
برای مشاهده کلی دیتاست سه عکس رندوم از آن را نمایش میدهیم.

a shirtless man lies on a park bench with his dog



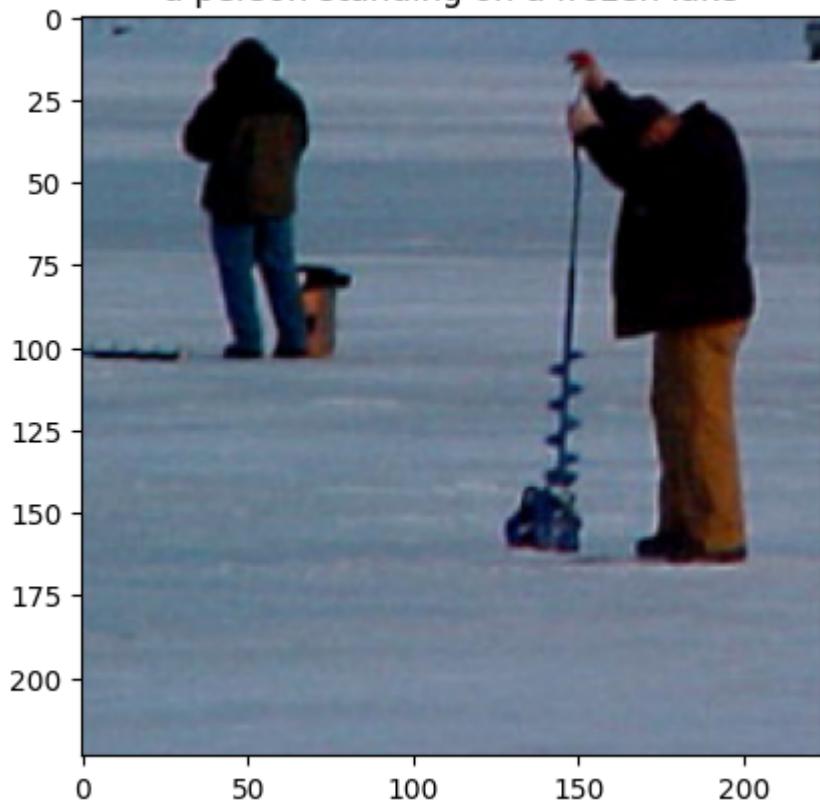
شکل ۱. مردی در حال خواب

a white dog is ready to catch a yellow ball flying through the air



شكل ٢. سگ سفید

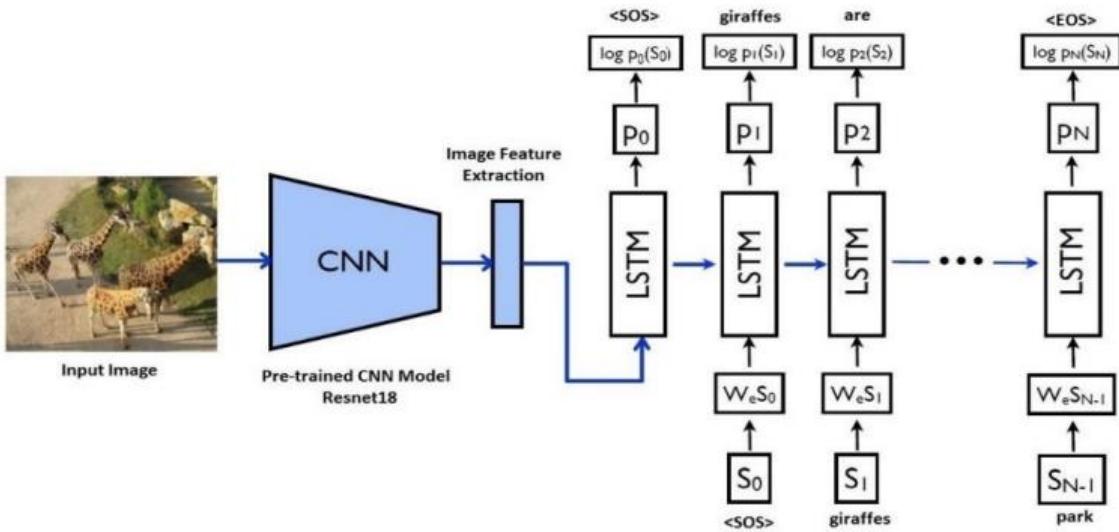
a person standing on a frozen lake



شکل ۳. مردی ایستاده روی دریاچه یخی

طراحی شبکه

مدل کلی شبکه به صورت زیر است:



شکل ۴. مدل شبکه در حال پیاده سازی

در اینجا تصاویر به CNN (مدل Resnet18) داده میشود تا فیچر مپ های آن استخراج شود. بدین منظور لازم است از مدل از پیش آموزش داده شده Resnet18 لایه آخر آن که مربوط به Fully Connected ها است جدا شود تا فیچر مپ عکس را در خروجی خود بدهد.

همچنین در تصویر شبکه LSTM دیده میشود که در ورودی خود فیچر مپ به دست آمده از شبکه CNN را دریافت میکند و با کپشن های آن آموزش میدهد. این شبکه دارای لایه مخفی با سایز ۲۵۶ و embedding با سایز ۳۰۰ برخوردار است. هدف این است که مدل بتواند با ورودی گرفتن تصاویر کپشن جدیدی با استفاده از دیکشنری تولید کند.

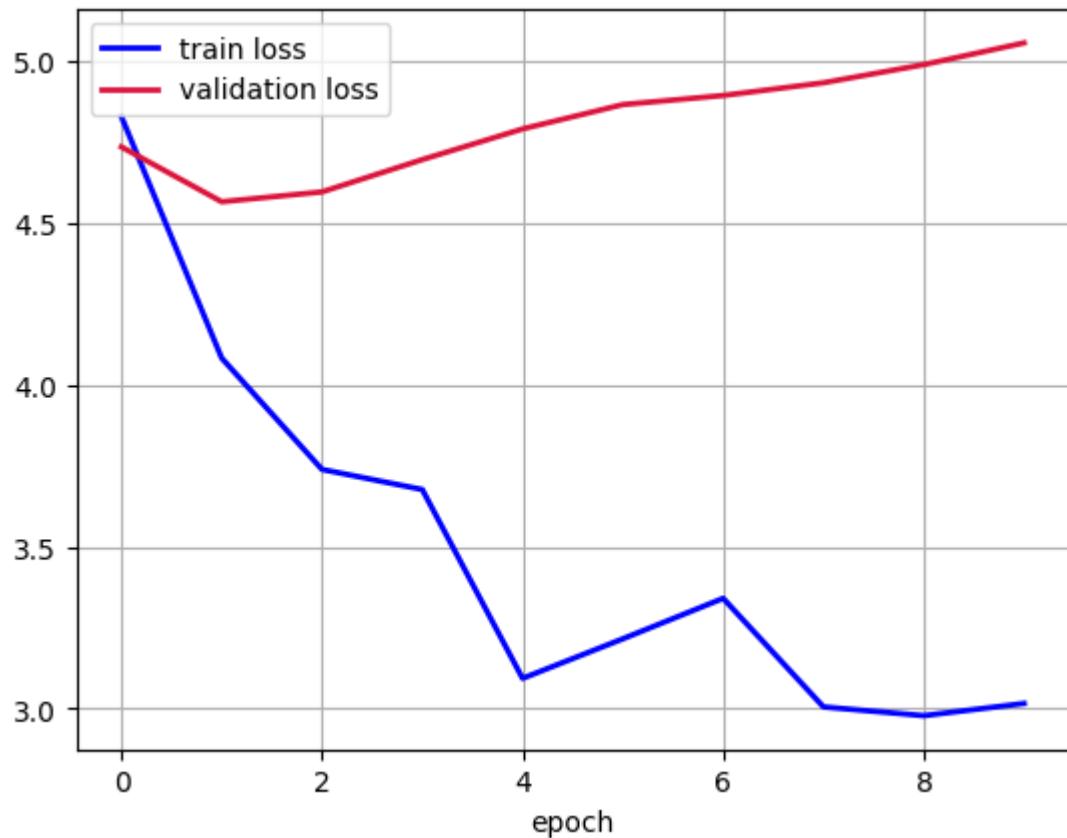
ماژول های CNN و LSTM را با پای تورج تولید میکنیم و در کلاس جدیدی آنها را با هم ادغام میکنیم. دو بار در این فرآیند آن ها را train میکنیم. در بار اول وزن های CNN را freeze کرده و در بار دوم آن هارا unfreeze میکنیم. دلیل این است که میخواهیم یک بار شبکه CNN روی عکس های train نیز آموزش داده شود امید به اینکه دقیق شبكه بالاتر رود.

مدل Freeze شده

در این مدل تمامی پارامتر های شبکه CNN را فریز میکنیم و شبکه را آموزش میدهیم. تعداد ایپاک های طی شده در این شبکه 10 تاست، همچنین Batch size دیتا لودر ۲۵۶ و از learning_rate=0.00004 استفاده شده است.

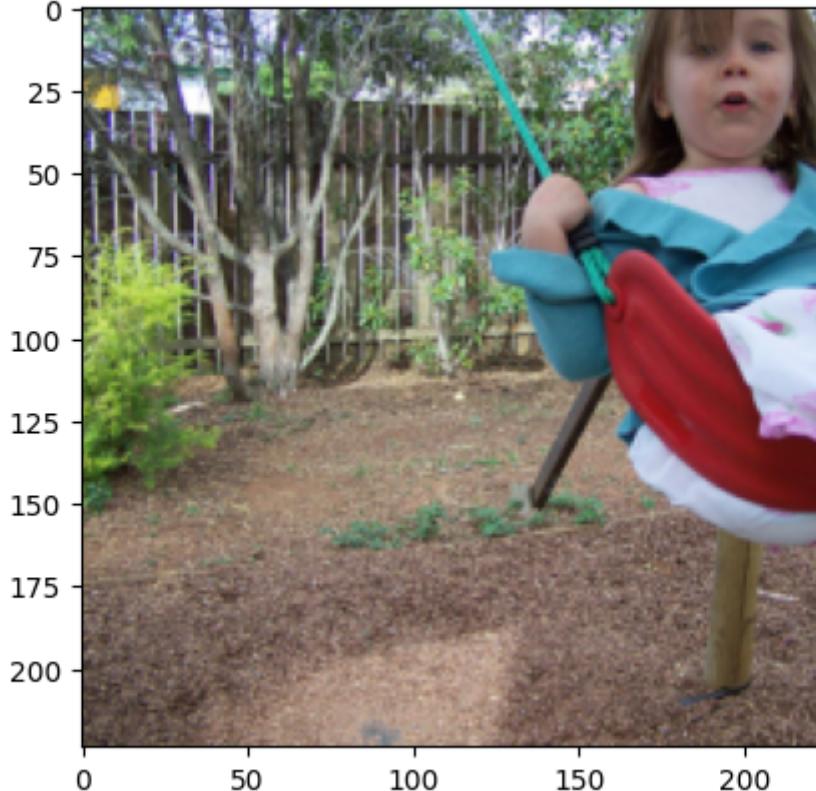
```
Epoch 7/10 -- Step 142/142 -- loss: 3.3410, val_loss: 4.8940 -- Time: 313.093 s
Epoch 8/10 -- Step 142/142 -- loss: 3.0056, val_loss: 4.9332 -- Time: 315.665 s
Epoch 9/10 -- Step 142/142 -- loss: 2.9786, val_loss: 4.9892 -- Time: 314.570 s
Epoch 10/10 -- Step 142/142 -- loss: 3.0163, val_loss: 5.0564 -- Time: 316.294 s
```

شکل ۵. نتایج ایپاک آخر

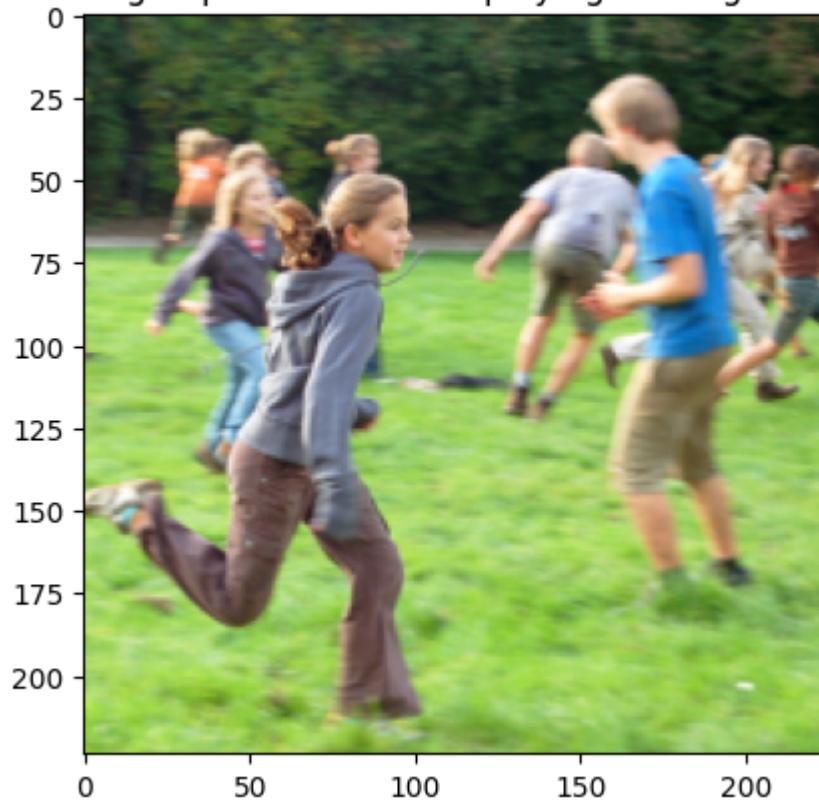


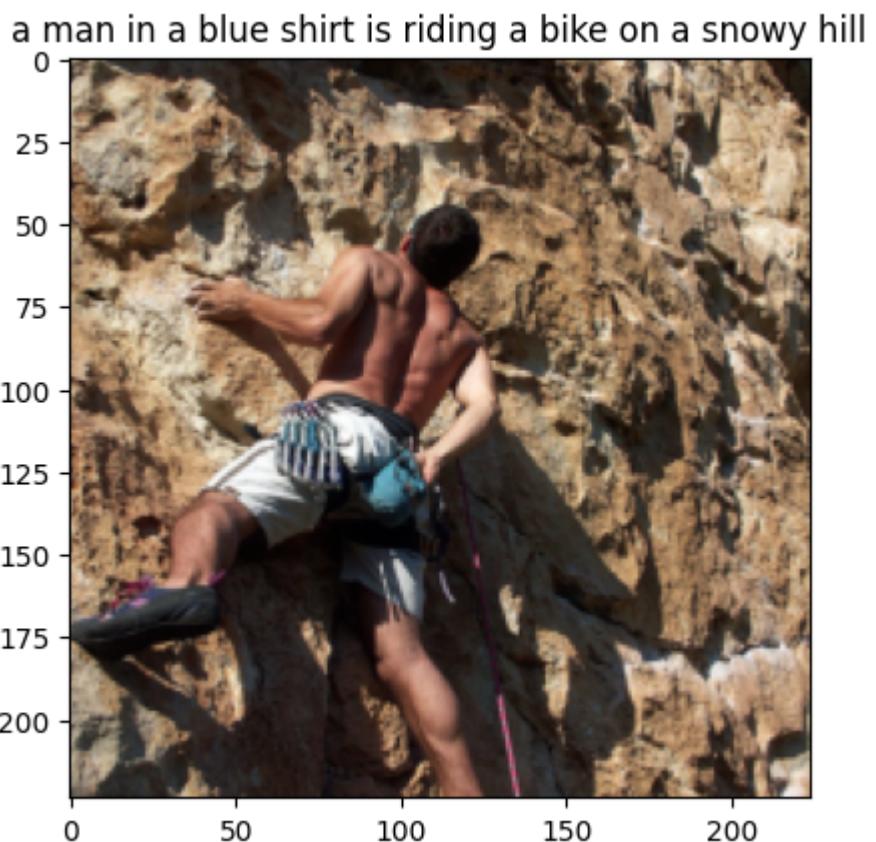
شکل ۶. نتایج خطای شبکه

a young girl in a red shirt and a blue shirt is standing on a swing



a group of children are playing in the grass





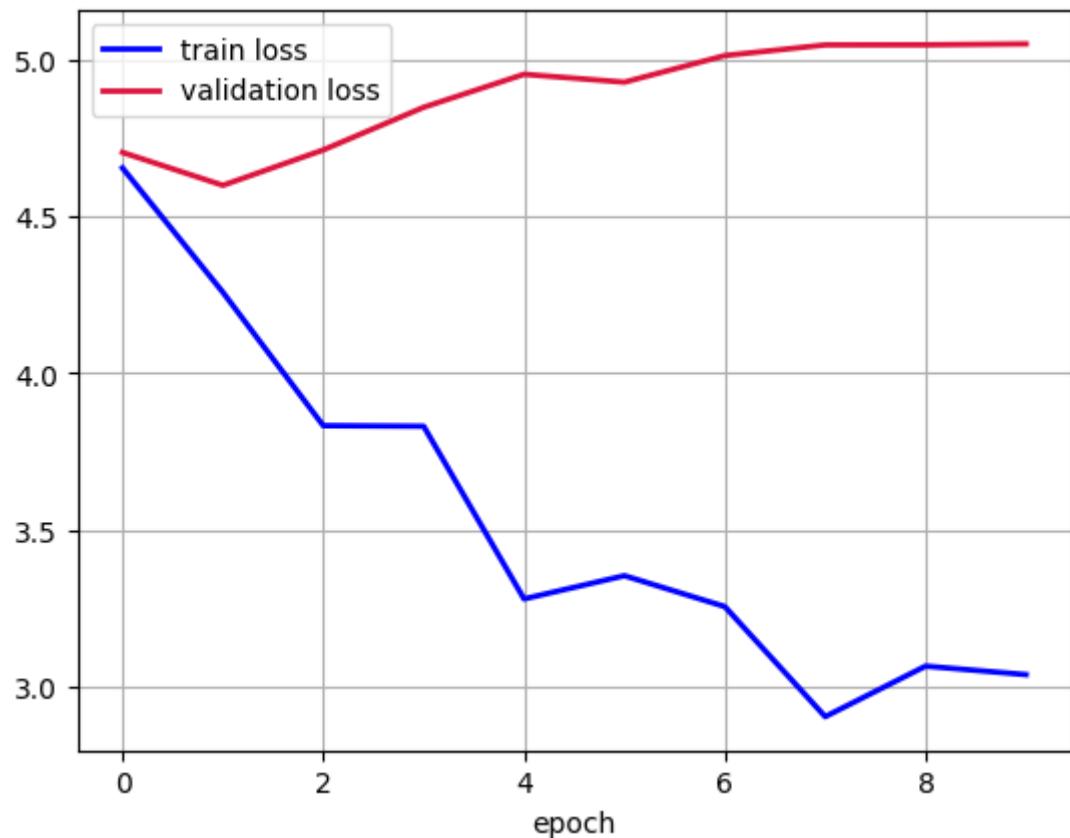
شکل ۷. خروجی شبکه روی ۳ عکس رندوم.

تصویر ۲ کاملا دقیق حدس زده شده و تصاویر او ۳ به طور تقریبی درست اند.

مدل Unfreeze شده

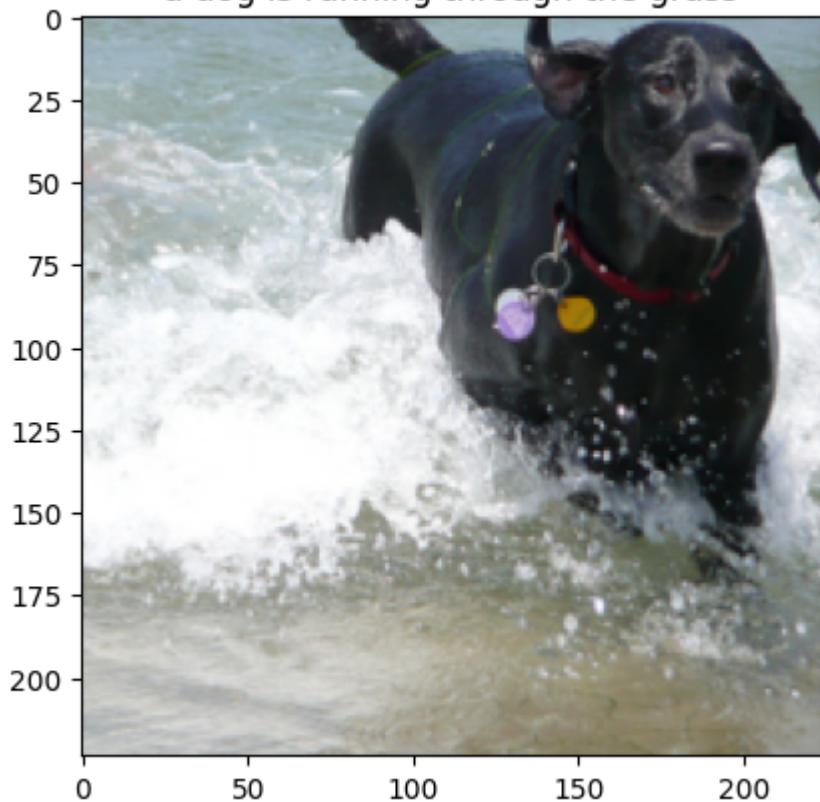
```
Epoch 7/10 -- Step 142/142 -- loss: 3.2557, val_loss: 5.0128 -- Time: 329.510 s
Epoch 8/10 -- Step 142/142 -- loss: 2.9047, val_loss: 5.0468 -- Time: 333.161 s
Epoch 9/10 -- Step 142/142 -- loss: 3.0659, val_loss: 5.0471 -- Time: 331.343 s
Epoch 10/10 -- Step 142/142 -- loss: 3.0388, val_loss: 5.0500 -- Time: 338.069 s
```

شکل ۸. نتایج چهار ایپاک آخر

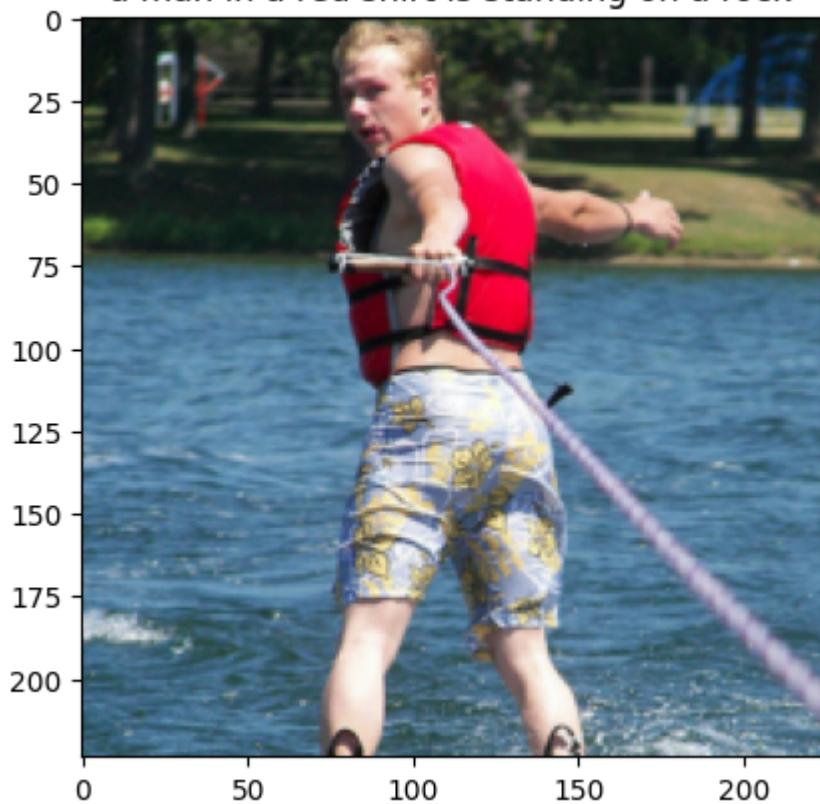


شکل ۸. نمودار loss در ۱۰ ایپاک

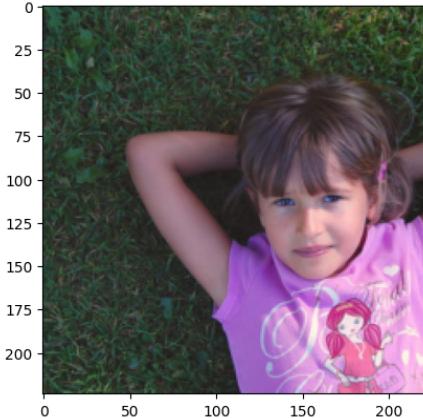
a dog is running through the grass



a man in a red shirt is standing on a rock



a young girl in a red shirt and a red shirt and a woman in a red shirt and a red shirt and a woman in a red shirt and a red shirt and a



شکل ۹. سه تصویر خروجی با کپشن های تولید شده

تصاویر او ۲ دقت قابل قبولی دارند اما در تصویر سوم یک جمله پشت هم تکرار شده است.

با توجه به نمودار های کشیده شده متوجه میشویم نمودار loss در حالتی که تمامی وزن ها آپدیت شدند به طور smooth تری کاهش یافت اما در نهایت هر دوی آنها از دقت برابری برخوردار شدند. هر کدام از این اجراء ها حدود یک ساعت به طول انجامید و پس از چند بار اجرا روی learning rate های متفاوت بهترین نتیجه چنین حاصل شد. با توجه به تعداد ایپاک های کم گذرانده شده تغییر محسوسی در نتایج ملاحظه نمیکنیم. زمان اجرا در مدل دوم افزایش یافت و این بخاطر آموزش شبکه CNN بود. همچنین loss به طور ملایم تری کاهش یافت که میتوان حدس زد بخاطر یادگیری ویژگی های خاص هر عکس در این مجموعه تصاویر است.

همچنین بالا رفتن validation loss نشان دهنده overfit شدن روی داده های train مان هست. برای رهایی از این مشکل میتوان از dropout و موارد مشابه استفاده کرد تا overfit رخ ندهد. همچنین earlyStopping میتواند کمک کننده باشد. چون در مقاله چنین مواردی ذکر نشده از آن در مدل نهایی خود استفاده نکردیم.

پاسخ ۲ - تشخیص اندیشه

۱- معماری LSTM و embedding

معماری LSTM در مقایسه با معماری RNN دارای چندین مزیت است:

1. حفظ وابستگی‌های طولانی‌مدت: در RNN‌های سنتی، مشکل معروف "مشکل ناپدید شدن گرادیان" (vanishing gradient problem) وجود دارد که باعث می‌شود اطلاعات برهمنکنندهای طولانی‌مدت در دنباله‌های طولانی به طور کامل از بین بروند. LSTM با استفاده از ساختارهای خاصی به نام "بازگردنده‌های درونی" (internal gates) می‌تواند این مشکل را حل کند و وابستگی‌های طولانی‌مدت را حفظ کند.
2. کنترل بیشتر بر فراموشی: LSTM با داشتن ساختار "بازگردنده‌های درونی" قادر است اطلاعات مهم را در حافظه خود نگه‌داری کند و در عین حال اطلاعات غیرضروری را فراموش کند. این قابلیت باعث می‌شود LSTM در کاربردهایی که برای حل مسئله نیاز به حفظ اطلاعات مهم در طول زمان دارند (مانند ترجمه ماشینی و تشخیص گفتار) عملکرد بهتری داشته باشد.

استفاده از Embedding Word نیز یک روش برای نمایش کلمات در مدل‌های پردازش زبان طبیعی است. Embedding Word معنای یک کلمه را به یک بردار عددی چندبعدی نگاشت می‌دهد. عمدهاً به دو دلیل از Embedding Word استفاده می‌شود:

1. نمایش فضایی: Embedding Word می‌تواند کلمات را در یک فضای چندبعدی قرار دهد که در آن فاصله بین کلمات نشان‌دهنده شباهت معنایی آنها است. با این روش، می‌توان به طور معناداری به محاسبه شباهت بین کلمات و استفاده از این اطلاعات در مدل‌ها و وظایف مختلف بپردازیم.

2. کاهش ابعاد: Embedding Word ابعاد ویژگی‌های کلمات را کاهش می‌دهد. در متن‌های بزرگ، تعداد کلمات مختلف ممکن است بسیار زیاد باشد. با استفاده از Embedding Word، هر کلمه را با یک بردار اندازه‌گیری کمتر نمایش می‌دهیم که موجب کاهش ابعاد داده و کاهش پیچیدگی محاسباتی می‌شود.

برای کلماتی که چندین معنی متفاوت دارند، می‌توان از GloVe استفاده کرد. GloVe یک روش Embedding Word است که بر اساس تجمع آماری اطلاعات در یک مجموعه متنی بزرگ ساخته شده است. این روش قادر است به طور معقول با چالش معانی چندگانه در کلمات برخورد کند. به عنوان مثال، هر کلمه با چند بردار GloVe مختلف نشان داده می‌شود، به طوری که هر بردار نمایانگر یکی از معانی کلمه است. در نتیجه، می‌توان با استفاده از بردارهای GloVe مناسب، به طور خوبی با کلمات چندمعنی مواجهه کرد و آن‌ها را در مدل‌های پردازش زبان مورد استفاده قرار داد.

2- پیش پردازش دادگان

در این مرحله stop words ها را حذف و تمامی حروف را lowercase می‌کنیم. نتایج پیش پردازش و همچنین بررسی داده‌ها در فایل به همراه کد موجود است.

3- پیاده‌سازی طبقه‌بندی نیت

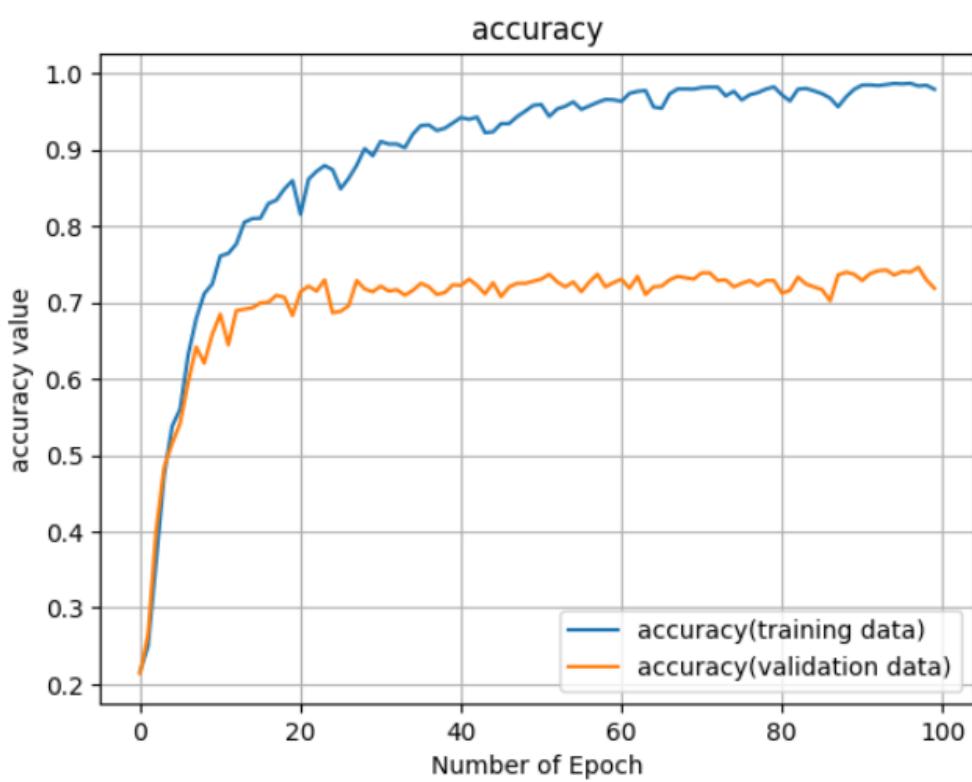
ابتدا GloVe را load می‌کنیم و embedding matrix را می‌سازیم. سپس مدل شرح داده شده در مقاله را پیاده‌سازی می‌کنیم. برای این کار از یک لایه با عنوان Embedding، یک لایه LSTM و در نهایت یک لایه با تابع فعال‌ساز softmax ایجاد می‌کنیم.

برای loss از categorical_crossentropy و همچنین Adam optimizer استفاده می‌کنیم.

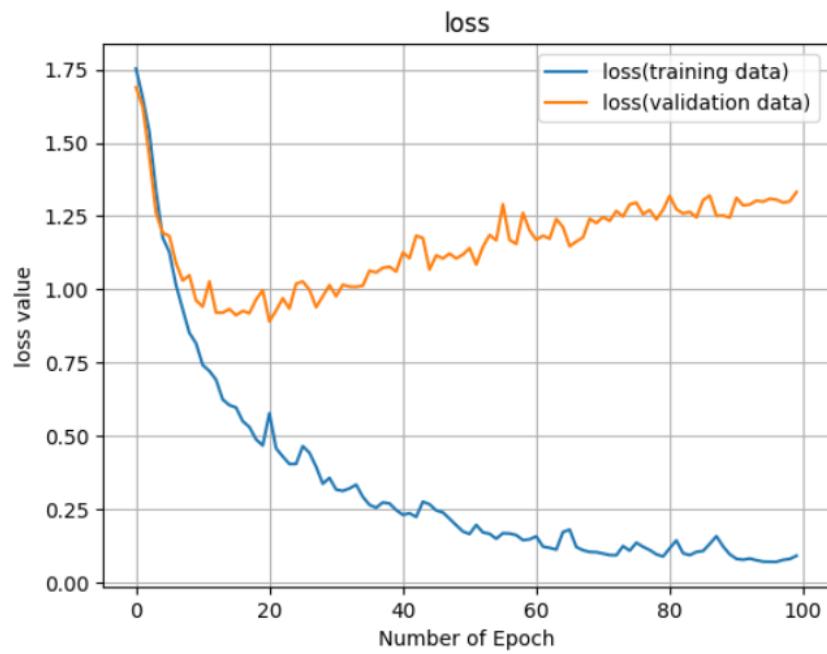
سپس در دو مرحله، یک بار با hidden_state = 25 و بار دیگر با 100 مدل را build می‌کنیم.

برای train کردن مدل، تعداد epoch 100 و batch_size 128 در نظر می‌گیریم و داده‌ها را به نسبت 80، 20 به بخش validation و train تقسیم می‌کنیم.

نتایج برای حالتی که hidden-state = 25 است به صورت زیر می‌باشد.



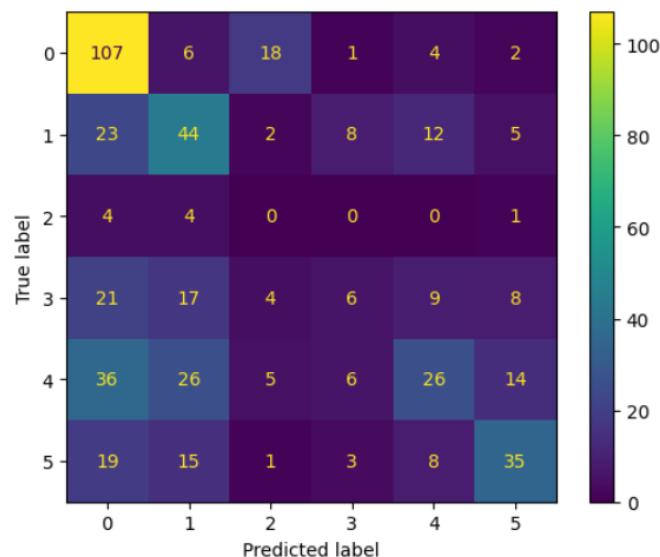
شکل 1.2. نمودار accuracy برای $hidden_state = 25$



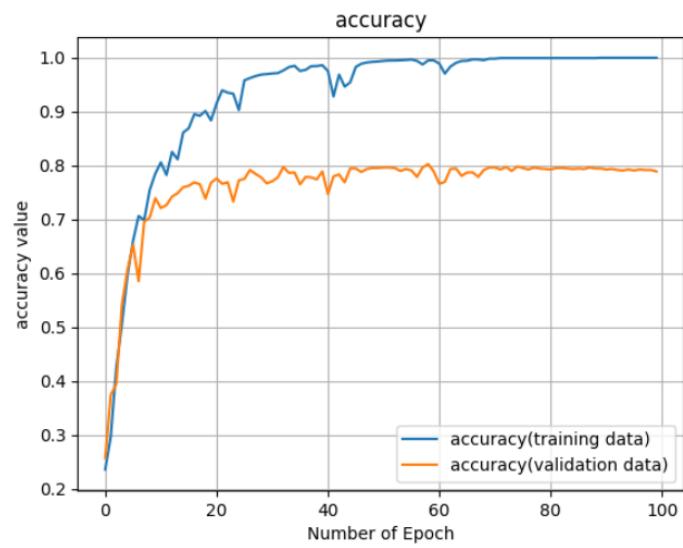
شکل 2.2. نمودار loss برای $hidden_state = 25$

	precision	recall	f1-score	support
0	0.51	0.78	0.61	138
1	0.39	0.47	0.43	94
2	0.00	0.00	0.00	9
3	0.25	0.09	0.13	65
4	0.44	0.23	0.30	113
5	0.54	0.43	0.48	81
accuracy			0.44	500
macro avg	0.36	0.33	0.33	500
weighted avg	0.43	0.44	0.41	500

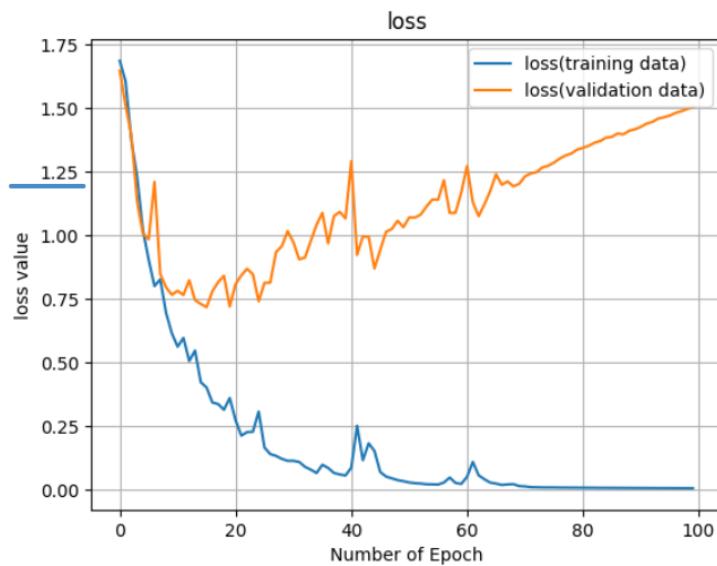
شكل 3.2. مقادير Score، Recall، Precision، Accuracy



شكل 4.2 confusion-matrix



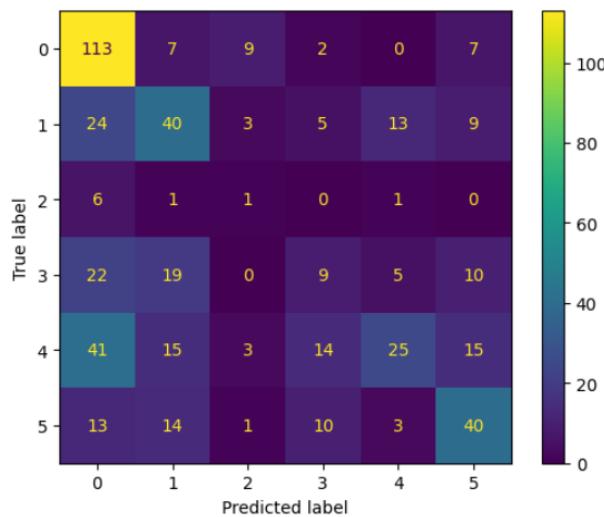
شکل 5.2. نمودار accuracy برای hidden_state = 100



شکل 6.2. نمودار loss برای hidden_state = 100

	precision	recall	f1-score	support
0	0.52	0.82	0.63	138
1	0.42	0.43	0.42	94
2	0.06	0.11	0.08	9
3	0.23	0.14	0.17	65
4	0.53	0.22	0.31	113
5	0.49	0.49	0.49	81
accuracy			0.46	500
macro avg	0.37	0.37	0.35	500
weighted avg	0.45	0.46	0.43	500

شکل 7.2. مقادیر Score, Recall, Precision, Accuracy



شکل 8.2 confusion-matrix

با توجه به نتایج گزارش‌های حالت hidden_state=100 و hidden_state=25، می‌توانیم نتایج را تحلیل کنیم.

در حالت `hidden_state=100`:
 - Precision: میانگین دقت برای هر دسته برابر با 37٪ است. این نشان می‌دهد که مدل در تشخیص دسته‌ها دقت کمی دارد.

- Recall : میانگین بازیابی (تشخیص) برای هر دسته برابر با ۳۷٪ است. مدل نمی‌تواند به طور کامل دسته‌ها را تشخیص دهد.

- F1-Score : میانگین F1-Score برابر با ۳۵٪ است. این نشان می‌دهد که مدل در تعادل بین دقت و بازیابی دسته‌ها ضعیف عمل می‌کند.

- Accuracy : دقت کلی مدل برابر با ۴۶٪ است، که نشان می‌دهد مدل در تشخیص درست دسته‌ها عملکرد بهتری دارد.

در حالت `hidden_state=25`

- Precision : میانگین دقت برای هر دسته برابر با ۳۶٪ است. مدل در تشخیص دسته‌ها دقت کمی دارد.

- Recall : میانگین بازیابی (تشخیص) برای هر دسته برابر با ۳۳٪ است. مدل نمی‌تواند به طور کامل دسته‌ها را تشخیص دهد.

- F1-Score : میانگین F1-Score برابر با ۳۳٪ است. این نشان می‌دهد که مدل در تعادل بین دقت و بازیابی دسته‌ها ضعیف عمل می‌کند.

- Accuracy : دقت کلی مدل برابر با ۴۴٪ است، که نشان می‌دهد مدل در تشخیص درست دسته‌ها عملکرد بهتری دارد.

با توجه به نتایج، هر دو مدل در تشخیص دسته‌ها دقت کمی دارند و نتایج آنها ایده‌آل نیست. احتمالاً با افزایش تعداد hidden state و بهبود سایر پارامترها می‌توان عملکرد مدل را بهبود بخشید. همچنانی، ب

بررسی دقیق‌تری از داده‌ها و فهم بهتری از ماهیت مسئله ممکن است منجر به بهبود نتایج شود.