Image Captioning
Show, Attend, Tell

استاد : دکتر مهدی افتخاری*

مباحث ویژه در هوش مصنوعی (یادگیری عمیق)

سپهر بزمی

بخش مهندسی کامپیوتر، دانشگاه شهید باهنر کرمان، کرمان، ایران

Introduction

در این پروژه یک مدل Image Captioning طراحی خواهیم کرد که با اتصال یک DecoderRNN و EncoderCNN با دریافت تصاویر آن هارا شرح میدهد . این مدل برگرفته از مدل های Image captioning ای که در مقاله های show and tell و show attend and tell شرح داده شده است. ابتدا با هر دو مدل ذکر شده به طور خلاصه آشنا خواهیم شد، سپس با بررسی دیتاست flickr8k و پردازش آن مدل خود را آموزش داده و مراحل و دقت آن را بررسی خواهیم کرد.

Show and Tell

در سال 2015 مقاله show and tell یک مدل عصبی end-to-end که Computer vision را با Natural language processing ادقام میکرد معرفی کرد که با استفاده از آن تصاویر را توصیف میکرد. این روش از machine translation الهام گرفته بود که ورودی های آن تصاویری که encode شده بودند بود. در ادامه با ساختار آن بیشتر آشنا خواهیم شد.

CNN ها ثابت کرده بودند که میتوانند encoder ها خوبی برای embed کردن تصاویر به بردار هایی با سایز ثابت باشند. با در نظر گرفتن همین ویژگی خروجی classifier آن ها به ورودی یک decoder تبدیل شد و Neural Image Caption یا به اختصار NIC طراحی شد. ConvNet اولیه میتواند یک ResNet یا Inception model باشد. و برای قسمت

decoder از یک LSTM یا Long Short-Term Memory Network استفاده شده است. این LSTM یاد میگیرد که با استفاده از embedding تصویر و کلمات پیشین یک caption برای تصویر بسازد. این مدل در هنگام یاد گیری تلاش میکند که negative log-likelihood را حداقل کند. همچنین در هنگام یادگیری از تکنیک teacher forcing استفاده میشود که به نوعی در هنگام آموزش کلمات درست قبلی را به مدل تغذیه میکند.

- **چرا LSTM ؟**

انتخاب LSTM به عنوان decoder دلایل متعددی دارد، مثلا این مدل ها توانایی generate کردن داده های sequential را دارند. همچنین حافظه کوتاه مدت مناسبی برای جملات محدود دارند که در ساخت متون پیوسته و stable مفید خواهد بود. خصوصیت کارآمد دیگر LSTM ها تولید خروجی با طول متفاوت است که در همچنین مسئله ای برای ما ضروری است. در نهایت استفاده از این مدل ها به جای یک RNN ساده از فراموشی و محو شدگی گرادیان نیز جلوگیری میکند.

- **LSTM چگونه در اینجا عمل میکند ؟**

درک LSTM ها به صورت ریاضی کار هرکسی نیست (مخصوصا من) به همین دلیل برای درک فرایند captioning سعی میکنیم به ساده ترین شکل ممکن این کار را انجام دهیم.

فرض میکنیم که یک یخچال(CNN) داریم که اطلاعات تمام مواد داخلش رو به صورت یک لیست آماده دراختیار آشپز(LSTM) میدهد. حالا این آشپز برای جلوگیری از فراموشی از یک سری ابزار استفاده میکنه :

1- دفترچه یادداشت (Ct – **memory cell**) : اطلاعات مهم را در آن مینویسد مثلا اینکه میخواهیم سوپ درست کنیم نه سالاد. (میخواهیم چه چیزی در تصویر را توصیف کنیم.)
2- فیلتر فراموشی (**Forget gate**) : تصمیم میگیرد چه چیزی را دور بیندازد و استفاده نکند. مثلا اگر از هویج استفاده کرد دیگر نمیخواهد از آن استفاده کند.
3- **Input gate :** تصمیم میگیرد که میخواهد چه اطلاعات جدیدی را ذخیره کند، مثلا این که میخواهد در مرحله بعدی از مرغ استفاده کند.
4- **Output gate :** که با استفاده از آن تصمیم میگیرد حرف بعدی ش چیست، مثلا میگوید الان که مرغ هم اضافه کردم باید برنج اضافه کنم.

با توجه با مطالب بالا، فرایند آشپزی او (تولید caption ما) به صورت زیر پیش میرود.

0- ابتدا آشپز لیست مواد داخل یخچال را مشاهده میکند
1- میگوید که میخواهد از هویج استفاده کند و سپس هویج را فراموش میکند.
2- میگوید که میخواهد از مرغ استفاده کند (یادش می آید که داریم سوپ درست میکنیم.)
3- میگوید که برنج را اضافه کنید.
4- در نهایت جمله نهایی او به شکل "با هویج شروع میکنیم، مرغ اضافه میکنیم و برنج را در نهایت اضافه میکنیم" خواهد بود.

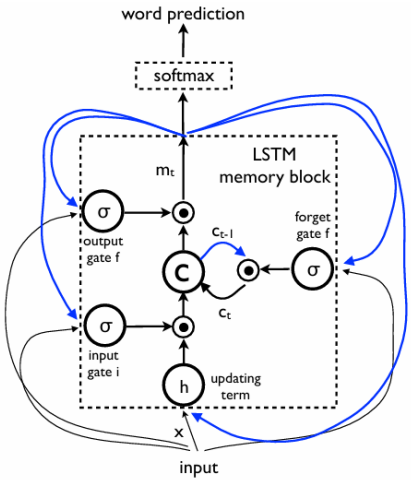
با درک این مراحل، فرایند در مدل ما به شکل زیر خواهد بود:

Sees dog → Says "A dog" (saves "dog" in memory).

Sees park → Says "plays in the park" (remembers "dog" is the subject).

Caption: **"A dog plays in the park".**

Component	Formula	Purpose
Forget Gate	$f_t = \sigma(W^f \cdot [h_{t-1}, x_t] + b^f)$	Decides what old info to discard (0=forget, 1=keep)
Input Gate	$i_t = \sigma(W^i \cdot [h_{t-1}, x_t] + b^i)$	Decides what new info to store
New Memory	$\tilde{C}_t = \tanh(W^c \cdot [h_{t-1}, x_t] + b^c)$	Creates candidate memory (what could be added)
Memory Cell	$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$	Combines old and new memory (actual update)
Output Gate	$o_t = \sigma(W^o \cdot [h_{t-1}, x_t] + b^o)$	Decides what parts of memory to output
Hidden State	$h_t = o_t * \tanh(C_t)$	Final output (used for prediction)



Step	Input (x_t)	Forget Gate (f_t)	Input Gate (i_t)	Memory (C_t)	Output (h_t)	Word Predicted
0	Image features	Keep all (1.0)	Save "dog" (1.0)	["dog"]	Initialize	-
1	"A"	Keep "dog" (0.9)	Save "in" (0.8)	["dog", "in"]	"dog"	"dog"
2	"dog"	Forget "A" (0.1)	Save "park" (0.9)	["in", "park"]	"in"	"in"
3	"in"	Keep all (1.0)	-	["in", "park"]	"the park"	"the park"
4	"the park"	-	-	-		(stop)

- **مدل چگونه یاد میگیرد؟**

این مدل با استفاده از یک loss function مناسب از اشتباهات خود در هنگام تولید caption یاد میگیرد. این مدل سعی میکند که تفاوت بین کلمات پیش بینی شده با کلمات درست استفاده شده در داده های آموزشی حداقل کند.

$$L(I,S) = -\sum_{t=1}^{N} \log p_t(S_t) \; . \quad \mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \log(p(y_t^{(i)} | y_{1:t-1}^{(i)}, I^{(i)}))$$

به صورت کلی یادگیری این مدل ها و evaluate کردن آن ها میتواند کار دشواری باشد، چه جوابی را خوب و چه جوابی را نا مناسب میشناسیم؟ برای سنجش خوب یا بد بودن جواب ها جلوتر metric های متفاوتی را معرفی میکنیم، اما فعلا فرایند یادگیری را بیشتر بررسی میکنیم.

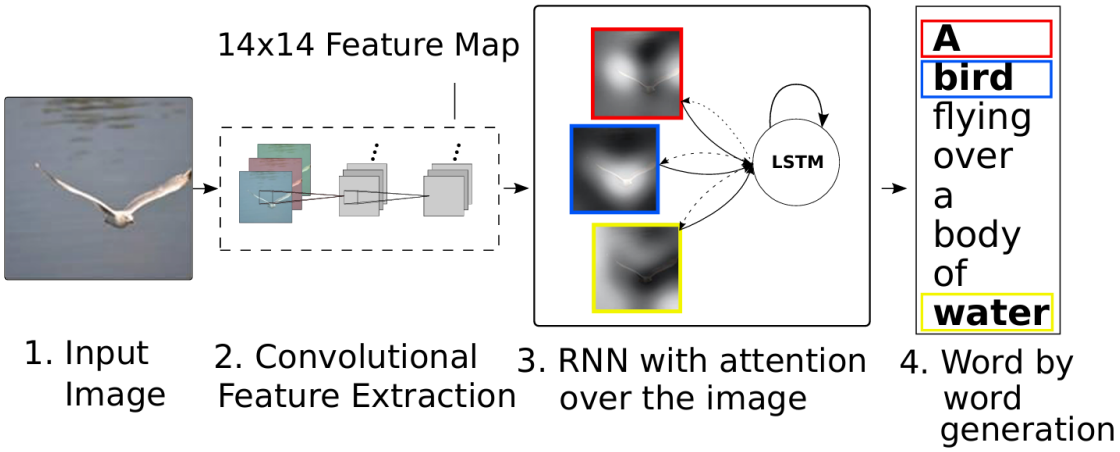
Step (t)	Correct Word	Predicted Probabilities (Example)	Loss Contribution (-log(p))	Notes
1	"A"	`{"A": 0.7, "The": 0.2, "An": 0.1}`	0.36	Confident correct prediction
2	"dog"	`{"dog": 0.6, "cat": 0.3, "bird": 0.1}`	0.51	Decent prediction
3	"plays"	`{"plays": 0.4, "runs": 0.5, "jumps": 0.1}`	0.92	Penalized for uncertainty
4	"in"	`{"in": 0.9, "at": 0.1}`	0.11	Very confident
5	"park"	`{"park": 0.8, "garden": 0.2}`	0.22	Good prediction
Total			**2.12**	Sum of all word-level losses

Show <Attend> and Tell

درادامه برای بهبود مدل قبل، یک مدل جدید ارائه شد که تا حد زیادی از مدل قبلی پیشی میگرفت ، این مقاله در سال 2016 منتشر شد و خیلی سر و صدا کرد. تغییراتی که این مدل با مدل قبلی دارد را مورد بررسی قرار میدهیم.

Feature	"Show and Tell" (2015)	"Show, Attend and Tell" (2016)
Image Representation	Single CNN feature vector (global embedding)	**Grid of CNN features** (spatial map, e.g., `14x14x512`)
Decoding	LSTM processes entire image at once	LSTM **dynamically attends** to relevant image regions per word
Focus	Treats image as one summary	Aligns words with specific regions (e.g., "dog" → dog pixels)

همینطور که مشاهده میشود تغییرات هم در Encoding و هم در Decoding اتفاق افتاده ست، به گونه ای که ورودی LSTM یک feature vector نخواهد بود بلکه یک spatial map خواهد بود(این مدل با قابلیت spatial attention تصویر را به grid های 14x14 میشکاند و وزن هر ناحیه را در هر بازه زمانی محاسبه میکند.) این عملیات مشکل bottleneck تبدیل تصاویر به بردار هارا حل میکند.همچنین LSTM به صورت پویا و با توجه local عکس را مورد بررسی قرار میدهد، این خصوصیت به مدل قابلیت توجه را میدهد.



- **LSTM مورد استفاده در این مدل**

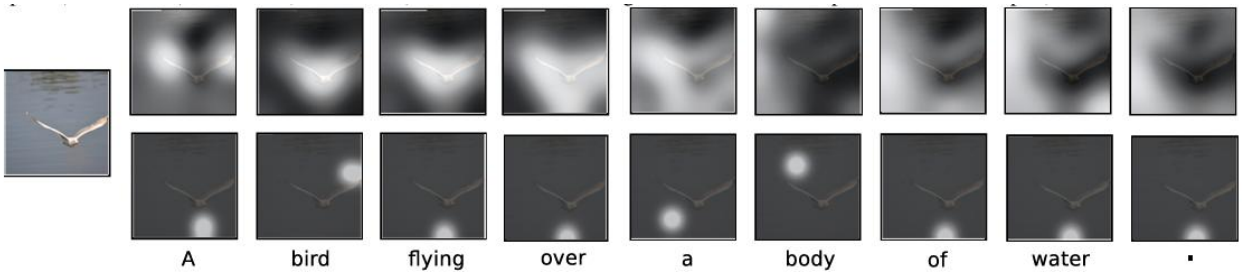
شبکه LSTM مورد استفاده در این مدل به دو بخش تقسیم میشود،

Attention-LSTM : با وظیفه محاسبه وزن های attention روی قسمت های تصویر.

Language-LSTM : تولید کلمات با استفاده از feature هایی که با آن ها توجه شده.

در هر iteration زمانی (t) قسمت اول attention score تمام قسمت های اول را تولید میکند، سپس context vector با استفاده از weighted sum فیچر های تصویر (وابسته به a_t) ساخته میشود. در نهایت language-LSTM با استفاده از آن context vector کلمه بعد را پیش بینی میکند.

توجه میتواند به صورت "soft" و "hard" اتفاق بیفتاد که در تصویر پایین تفاوت آن ها مشهود است.



Model Comparison

در نهایت، show and tell در شرایطی استفاده میشود، که نیاز به یک مدل سبک برای تصاویری که پیچیده نیستند مثلا thumbnail ها و avatar ها و... استفاده میشود، show attend and tell در شرایطی استفاده میشود که نیاز به یک مدل دقیق داریم و تصاویرمون خصوصیات و جزییات spatial دارند، مثل تصاویر پزشکی. در کل مدل اول سرعت بیشتر ولی مدل دوم دقت بیشتری به ما میدهد و این یک trade off است که باید نسبت به نیاز خود بسنجیم.

Feature/Circumstance	"Show and Tell" (No Attention)	"Show, Attend and Tell" (With Attention)	Best Choice
Image Complexity	Simple images (single object)	Complex scenes (multiple objects)	**Attention** for cluttered scenes
Speed	Faster (single vector processing)	Slower (dynamic attention per word)	**No Attention** for real-time apps
Caption Precision	Generic captions ("A dog in a park")	Detailed captions ("A black dog on grass")	**Attention** for fine-grained tasks
Training Data	Works with smaller datasets	Needs larger datasets to learn attention	**No Attention** if data is limited
Interpretability	Black-box (no region focus)	Explains decisions via attention maps	**Attention** for debug/analysis
Hardware Requirements	Lower (shorter sequence steps)	Higher (attention computations)	**No Attention** for edge devices
Use Cases	Thumbnail tagging,	Medical imaging,	
	Social media auto-captions,	Autonomous driving,	
	Low-resource apps	Detailed product descriptions	

Project Structure

ابتدا با ساختار پروژه برای درک بهتر فرایند ها آشنا میشویم :

```
image_captioning_assignment/
├── data/
│   └── download_flickr.py
├── models/
│   ├── encoder.py
│   ├── decoder.py
│   └── caption_model.py
├── utils/
│   ├── dataset.py
│   ├── vocabulary.py
│   ├── trainer.py
│   └── metrics.py
├── notebooks/
│   ├── 1_Data_Exploration.ipynb
│   ├── 2_Feature_Extraction.ipynb
│   ├── 3_Model_Training.ipynb
└   └── 4_Evaluation_Visualization.ipynb
```

این ساختار بر روی google drive و github نیز قابل دسترس و مشاهده خواهد بود.

Dataset and Data collection

برای این مدل از دیتاست flickr8k که مجموعه ای از 8092 تصویر و 40460 caption است که هرکدام از تصاویر 5 caption متفاوت در ساختار اما مشابه در معنی دارند.

ابتدا داده‌های تصویری و متنی مجموعه از منابع آنلاین بارگیری می‌شوند، سپس تصاویر و توضیحات متنی از فایل‌های ZIP استخراج شده و توضیحات به‌صورت ساختاریافته در قالب فایل CSV ذخیره می‌شوند. در ادامه، تصاویر به پوشه‌ای مشخص منتقل شده و مجموعه‌داده به سه بخش train، validation و test تقسیم می‌شود.

با بررسی token های به دست آمده از دیتاست میتوانیم insight های مفید تری به اطلاعات خود داشته باشیم. توزیع نسبتا خوبی در caption ها داریم که تقریبا بین 8 تا 13 توکن است که میتواند تعداد مناسبی برای یک caption model باشد. طبق این اطلاعات max tokens برای تولید Caption های مدل را 12 در نظر گرفتیم که در بازه distribution مشخص شده قرار بگیرد.

Statistic	Value
Count	40,460
Mean length	10.79
Standard Deviation	3.76
Minimum length	1
25th percentile	8
Median (50%)	10
75th percentile	13
Maximum length	36

Raw Caption	Tokenized Caption
A child in a pink dress is climbing up a set of stairs.	`[a, child, in, a, pink, dress, is, climbing, up, a, set, of, stairs]`
A girl going into a wooden building.	`[a, girl, going, into, a, wooden, building]`
A little girl climbing into a wooden playhouse.	`[a, little, girl, climbing, into, a, wooden, playhouse]`
A little girl climbing the stairs to her playhouse.	`[a, little, girl, climbing, the, stairs, to, her, playhouse]`
A little girl in a pink dress going into a wooden playhouse.	`[a, little, girl, in, a, pink, dress, going, into, a, wooden, playhouse]`

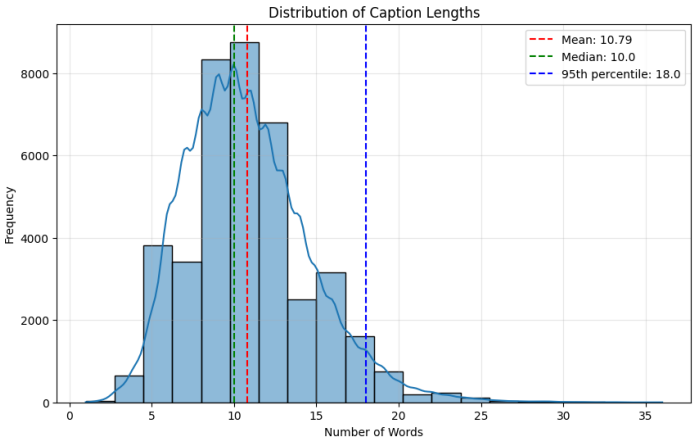
1. A man is cycling on the road .
2. A man rides a bicycle down a road with concrete barriers .
3. A man rides his bicycle on a wet road .
4. A person biking on a road near a fence .
5. Man cycles through diry bike path wearing white helmet and gloves

1. A boy jumping in a fountain .
2. a boy plays in the fountains .
3. A child plays in a fountain .
4. A little boy playing in the water .
5. A young boy is jumping through water being sprayed up from the ground .

1. A boy plays in the ocean with a boogie board .
2. An excited boy playing in the surf with a body board .
3. A smiling young boy wearing a bathing suit is holding a surfboard and laying down in the ocean water .
4. A young boy holds his boogie board in shallow ocean water .
5. A young boy with a blue body board .

1. A lady holding one dog while another dog is playing in the yard .
2. A woman holding a white dog points at a brown dog in the grass .
3. A woman holds a dog while another dog stands nearby in a field .
4. A woman holds her little white dog and points to a big brown dog at the bottom of the hill .
5. A woman is standing in a green field holding a white dog and pointing at a brown dog .

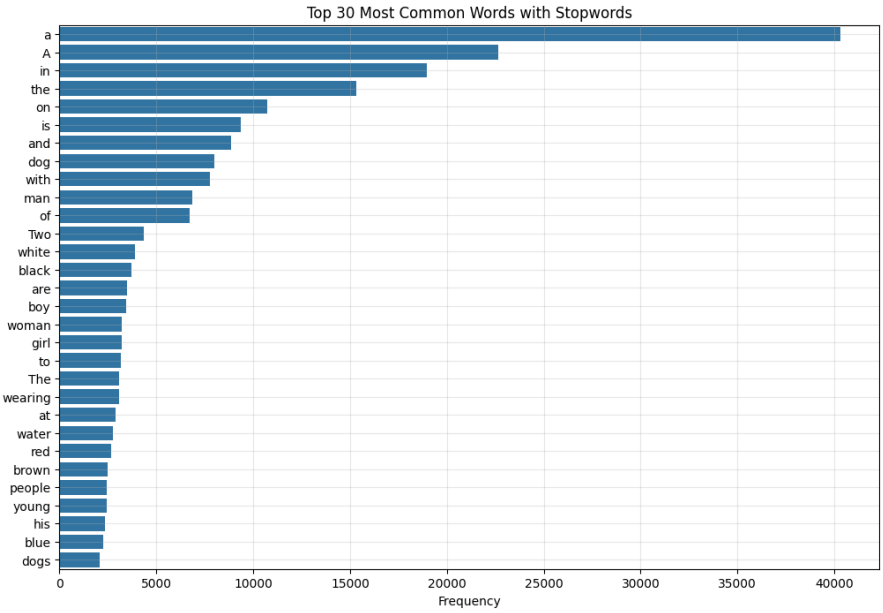
بررسی دیتاست جمع آوری شده از تصاویر اطلاعات مفیدی درباره داده هایی که با آن ها سر و کار داریم در اختیار ما قرار میدهد.

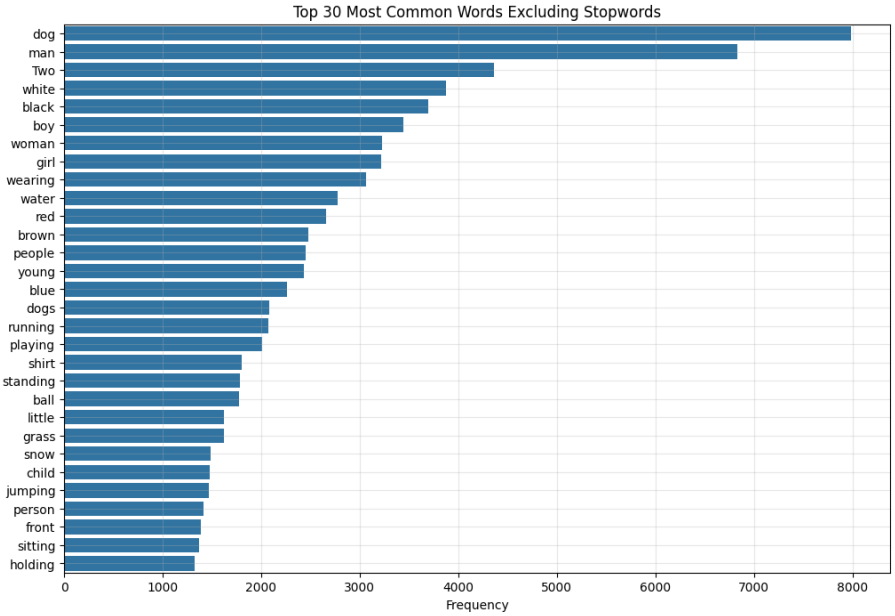


Min length	1
Max length	36
Mean length	10.79
Median length	10.0
90th percentile length	16.0
95th percentile length	18.0

بررسی ها نشان میدهد که vocabulary کلمات استفاده شده در caption های این دیتاست، مجموعا 9542 کلمه است که 4055 تا از آن ها فقط یک بار استفاده شده اند.

در نمودار زیر ، بیشترین کلمات استفاده شده با احتساب و بدون احتساب stop word هارا مشاهده میکنیم .

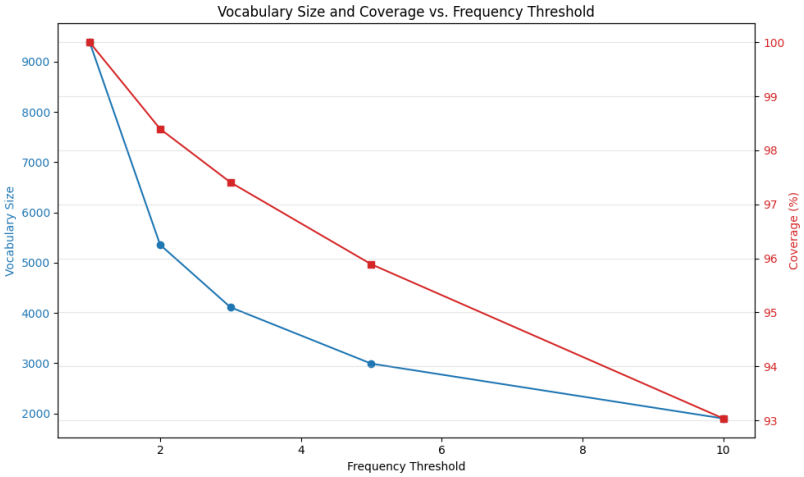




با مشاهده frequency کلمات پر تکرار در این دیتاست، و نسبت چندین برابری کلماتی مثل dog و من و two به سایر لغات یک سوگیری شدید به سمت این کلمات اتفاق می افتد و overfitting به high frequency patterns اتفاق می افتد. و به دلیل نبودن context قوی برای image embedding ها و یا محدودیت های decoding به صورت explicit، generation به سمت پترن هایی که به صورت آماری common تر هستند. که این باعث میشه caption هایی که به صورت semantically پوچ و بی معنی یا ناکامل مثل "two the" یا "two man" یا ... باشند.

مشاهده شدن two در تعداد زیادی از تصاویر به معنی بودن یک جفت از هرچیزی در آن تصویر است ولی مدل درکی از این قضیه ندارد و به اندازه کافی discriminative نیست و به صورت یک reflex کلمه two را برای تصاویر در نظر میگیرد بدون توجه به محتوایی که واقعا داره اتفاق میفاتد.

برای ساخت dictionary با سایز مناسب ، با انتخاب threshold های مناسب آزمایش میکنیم که کلمات با چه تعداد occurrence ای را در دیکشنری بیاوریم و در نهایت چه سایزی برای دیکشنری ما مناسب است که کمترین information loss را داشته باشد.



Threshold	Vocabulary Size	Coverage (%)
1	9383	100.00
2	5357	98.40
3	4113	97.41
5	2992	95.89
7	2400	94.62
10	1901	93.04
100	374	74.31
1000	45	38.38

پس از بررسی اطلاعات بالا، ترشهولد وکب مدل ما 5 در نظر گرفته شد که پوشش 95 درصدی دارد اما تهداد کلمات را از 9383 به 2992 که بسیار کوچک تر است میرساند.

Image-Captioner Model

برای پیاده سازی این مدل، طبق صحبت های قبل، نیاز به یک Encoder CNN و یک Decoder RNN و اتصال این دو به یک دیگر داریم، برای مدل Encoder گزینه هایی مانند resnet18 و mobilenet_v2 و inception_v3 داریم که از دو تاشون برای مقایسه بهتر استفاده کردیم، و برای Decoder میتوانیم از LSTM و GRU استفاده کنیم، با اینکه GRU سرعت بیشتری در آموزش برای ما خواهد داشت اما به دلیل اهمیت زیاد LSTM در فرایند های image captioning تصمیم به استفاده از آن در هر دو مدل آزمایشی شد.

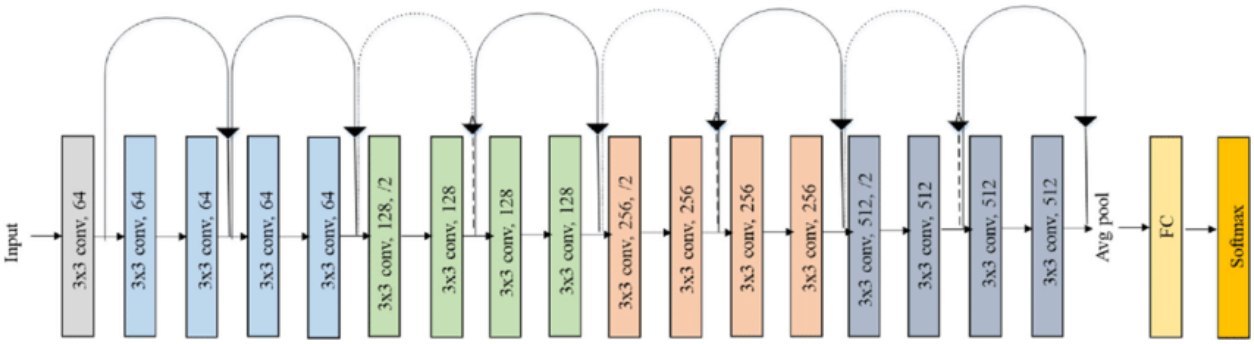
- **Encoder CNN**

در این ماژول، لایه‌های classification نهایی حذف شده و ویژگی‌های خروجی از طریق یک projection head به فضای embedding با ابعاد تعبیه شده‌ی دلخواه فشرده می‌شوند. پارامتر embed_size مشخص‌کننده‌ی ابعاد embedding space است که ویژگی‌های تصویری استخراج‌شده توسط شبکه‌ی CNN به آن map می‌شوند. این فضا نمایی فشرده و قابل استفاده از تصویر ارائه می‌دهد که توسط مدل decoder برای تولید caption به کار می‌رود.

در دو آزمایش انجام شده از دو مدل resnet18 و mobilenet_v2 استفاده شده. (به دلیل محدودیت های کولب از این دو مدل سبک تر استفاده شده است.)

مدل اصلی در این آزمایش resnet18 بود که از shortcut connection ها یا residual connection ها استفاده میکند که عمق شبکه را به نوعی کنترل و از محوشدگی گرادیان در انتشار از آن جلوگیری کند. این اتصالات به شبکه اجازه یادگیری identity mapping را میدهد. شبکه از 18 لایه متشکل از convolutional layers، batchnorm، Relu و pooling layers که باهم در residual block ها قرار گرفته اند تشکیل شده. هر بلاک

شامل دو تا conv layer و یک skip connection است که آن را برای تولید embedding بسیار مناسب میک



Projection Head نقش بسیار مهمی در پل زدن بین ویژگی‌های خام استخراج‌شده توسط ResNet18 و فضای معنایی مورد نیاز برای تولید کپشن ایفا می‌کند. به طور خاص، پس از استخراج بردارهای ویژگی با بعد بالا از خروجی شبکه یResNet18 ، هد پروجکشن که شامل یک لایه Linear، BatchNorm، ReLU و droput، این ویژگی‌ها را به یک embedding space با بعد پایین‌تر (مثلاً ۲۵۶) نگاشت می‌کند. اگر این هد پروجکشن وجود نداشته باشد، مدل ویژگی‌های بسیار پیچیده و گاهی پر از نویز را به دیکودر منتقل می‌کند، که یادگیری ساختارهای معنادار زبانی را برای مدل سخت‌تر می‌کند. در واقع، هد پروجکشن کمک می‌کند مدل روی اطلاعات مهم‌تر در فضای ویژگی‌های تصویری تمرکز کند و کپشن‌هایی با پایه بصری قوی‌تر تولید نماید. اما با projection head نمایش تصویر با مدل زبانی دیکودر هم راستا می‌شوند.

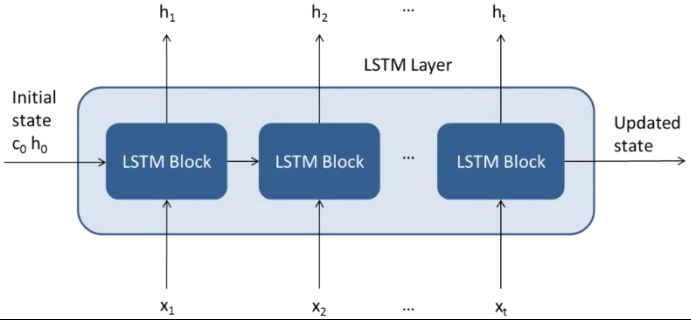
- **Decoder RNN**

Decoder مسئولیت transform کردن تصاویر encode شده به caption های با معنی را دارد کلمه کلمه پشت سر هم (تولید یک sequence با استفاده از کلمات قبل). این مدل متشکل است از embedding layer که word indices را به vector های با ابعدا embed size تبدیل میکند. این vector ها باعث میشوند که semantic similarity ها را حفظ کنیم.

یک مدل RNN که در اینجا LSTM استفاده شده است و یک Projection Head که hidden state های RNN را تبدیل به logits هایی در vocab مان میکند. که این آخرین مرحله قبل از انتخاب token بعدی میباشد.

فرایند forward pass این مدل به صورت زیر کار میکند :

Image feature ها بعد از تغییر شکل و ابعاد (reshape) به embed caption sequence اضافه میشن (در اصل دارن prepend میشند) و sequence جدید وارد RNN میشود. خروجی ها به vocab ای که داریم project میشوند. و output اولیه هم discard میشه و فقط کپشنی که مرتبط با خروجی هست خارج میشود.



Step	Description	Input Shape	Output Shape
1	Embed caption tokens using the embedding layer	`[batch_size, seq_len]`	`[batch_size, seq_len, embed_size]`
2	Apply dropout to the embeddings	`[batch_size, seq_len, embed_size]`	`[batch_size, seq_len, embed_size]`
3	Unsqueeze image features to add sequence dimension	`[batch_size, embed_size]`	`[batch_size, 1, embed_size]`
4	Concatenate image features with caption embeddings	`[batch_size, 1 + seq_len, embed_size]`	`[batch_size, seq_len + 1, embed_size]`
5	Pass the sequence through the RNN (LSTM or GRU)	`[batch_size, seq_len + 1, embed_size]`	`[batch_size, seq_len + 1, hidden_size]`
6	Apply dropout to RNN outputs	`[batch_size, seq_len + 1, hidden_size]`	`[batch_size, seq_len + 1, hidden_size]`
7	Project RNN outputs to vocabulary space using a linear layer	`[batch_size, seq_len + 1, hidden_size]`	`[batch_size, seq_len + 1, vocab_size]`
8	Discard the output corresponding to the image feature step	`[batch_size, seq_len + 1, vocab_size]`	`[batch_size, seq_len, vocab_size]`

نمونه‌گیری حریصانه – (Greedy Sampling)

- تولید جمله با توکن <start> آغاز می‌شود.
- در هر مرحله، محتمل‌ترین کلمه‌ی بعدی انتخاب می‌شود.
- زمانی متوقف می‌شود که یا توکن <end> تولید شود یا طول جمله به حداکثر تعیین‌شده برسد.
- این روش سریع است، اما ممکن است کپشن‌هایی تکراری یا غیر بهینه تولید کند.

جستجوی پرتوی – (Beam Search)

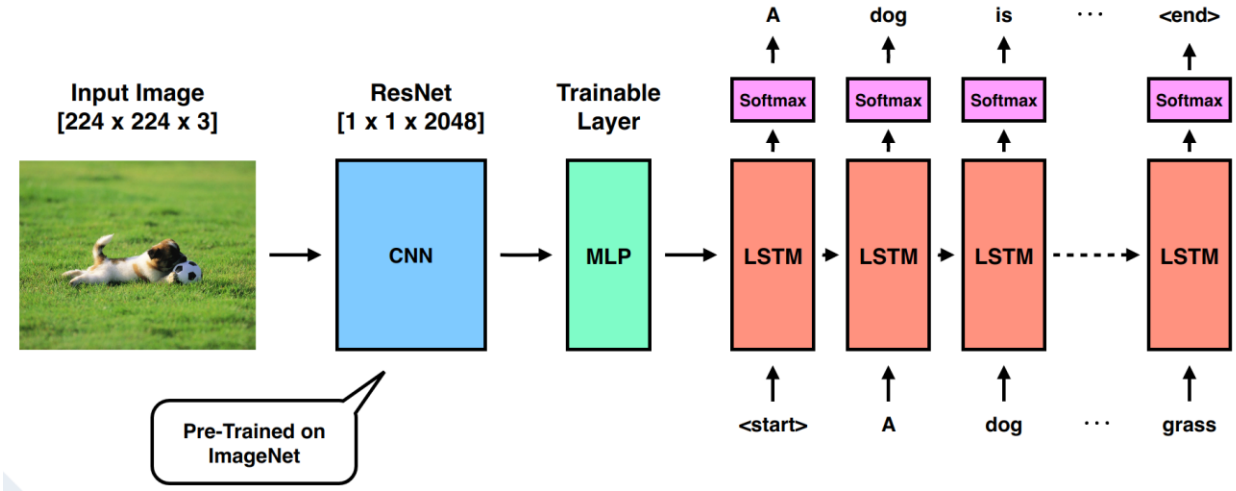
- در هر مرحله چندین دنباله‌ی ممکن نگه داشته می‌شود.
- مسیرهایی که بیشترین احتمال تجمعی دارند، انتخاب می‌شوند.
- این روش کندتر است، اما کپشن‌هایی سازگارتر و با کیفیت زبانی بالاتر تولید می‌کند.

درکل این decoder تبدیل به یک پلی بین مودالیتی تصویر و متن تبدیل میشود. از word embedding ها برای فهم زبان ، از LSTM برای sequence modeling و از linear projection برای پیشبینی از vocab استفاده میکند.

پس از درک Decoder با اتصال دو مدل encoder و decoder با هم میتوانیم Caption Model را بسازیم

• Caption model

با اتصال دو مدل قبلی به یک دیگر یک pipeline جدید میسازیم که بتوانیم از تصاویر caption های معنی دار بسازیم.
ساختار کلی مدل به صورت جدول زیر میباشد.



Component	Description
EncoderCNN	Uses a pre-trained CNN (e.g., ResNet18, InceptionV3) with the final classification head removed. It outputs a compact embedding vector that represents the image's content.
DecoderRNN	An LSTM/GRU-based decoder that conditions on the image feature vector and generates a caption one word at a time using teacher forcing during training, or sampling during inference.
Integration	The encoder outputs a fixed-size feature vector that is passed as the first input to the decoder, initiating the caption generation process.

Constructor های این مدل به صورت خلاصه در جدول زیر قابل مشاهده میباشند.

Argument	Type	Description
`embed_size`	`int`	Size of the shared embedding space between image features and word embeddings
`hidden_size`	`int`	Hidden state size for the RNN decoder
`vocab_size`	`int`	Number of unique tokens in the vocabulary
`num_layers`	`int`	Number of stacked RNN layers
`encoder_model`	`str`	Backbone model for the encoder (`'resnet18'` , `'resnet50'` , `'mobilenet_v2'` , `'inception_v3'`)
`decoder_type`	`str`	Type of RNN cell: `'lstm'` or `'gru'`
`dropout`	`float`	Dropout rate used in the decoder
`train_encoder`	`bool`	Whether to allow fine-tuning of the encoder during training

Feature Extraction

فرایند feature extraction توسط سه مدل مختلف resnet18 و resnet50 و inception_v3 انجام شد و نتایج مورد مقایسه قرار گرفت. توجه داشته باشید که تصاویر قبل از انتشار در شبکه ها normalized شده اند.

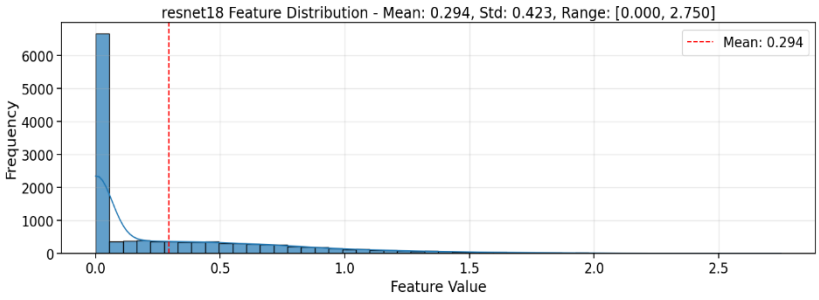
Model	Feature Dim	Parameters (M)	Extraction Time (8091 images)	Description
`resnet18`	512	11.7	759.97 seconds	Lightweight model with good performance/speed trade-off
`resnet50`	2048	25.6	62.05 seconds	Deeper model with higher-dimensional features
`mobilenet_v2`	1280	3.5	61.29 seconds	Efficient model designed for mobile and edge devices

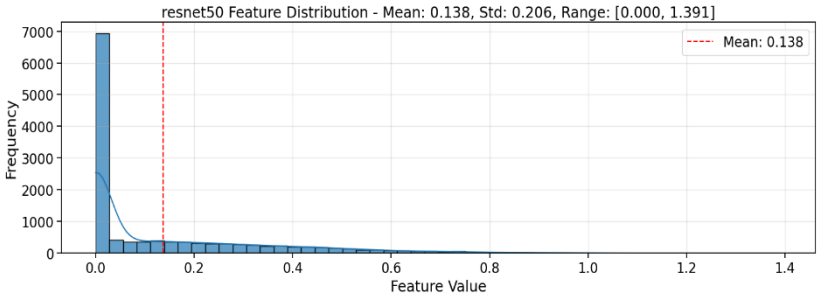
توزیع و distribution داده های feature های استخراج شده توسط مدل های مختف به صورت جدول زیر میباشد.

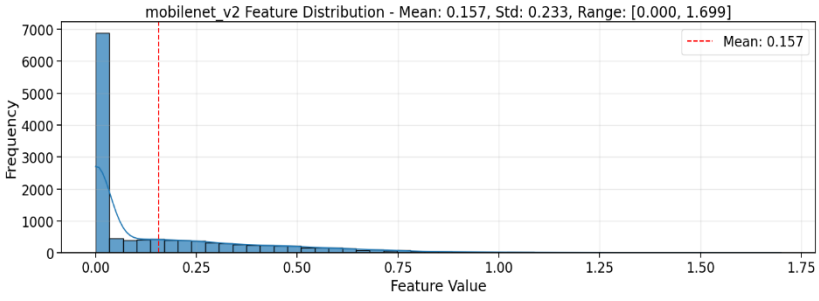
ResNet18ویژگی‌هایی با پراکندگی بیشتر و میانگین و انحراف معیار بالاتری تولید می‌کند، که ممکن است بر همگرایی مدل یا کالیبراسیون خروجی‌ها تأثیر بگذارد.

هر دو مدل ResNet50و MobileNetV2ویژگی‌هایی با توزیع فشرده‌تر و پراکندگی کمتر ایجاد می‌کنند؛ این موضوع از میانه و چارک پایین که برابر با صفر هستند، قابل مشاهده است.

این تفاوت‌ها نشان می‌دهند که ویژگی‌های تولیدشده توسط ResNet18 فعال‌تر و متراکم‌تر هستند، در حالی که ویژگی‌های دو مدل دیگر می‌توانند از پراکندگی کمتر و مقیاس‌پذیری بهتر برای نرمال‌سازی بهره‌مند شوند.

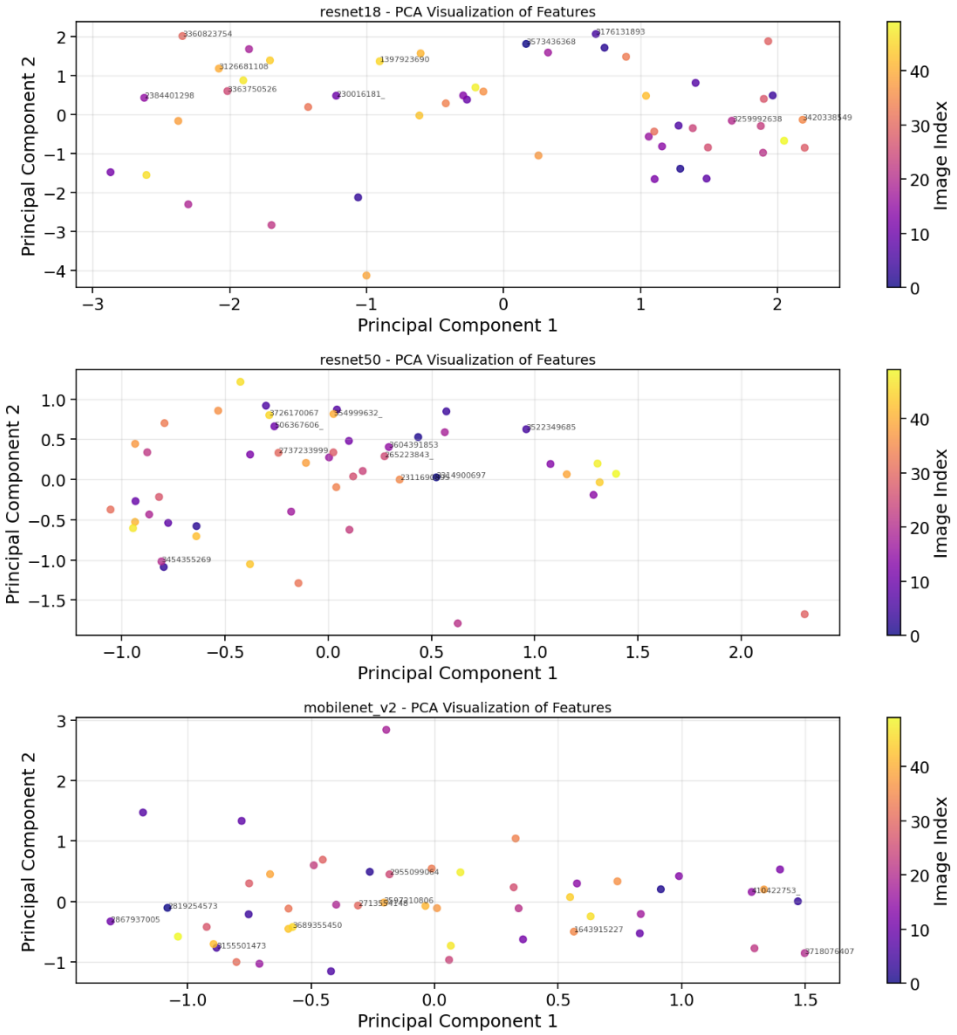






Statistic	ResNet18	ResNet50	MobileNetV2
Mean	0.2937	0.1376	0.1569
Std	0.4234	0.2057	0.2334
Min	0.0000	0.0000	0.0000
Max	2.7503	1.3914	1.6991
25th Percentile	0.0000	0.0000	0.0000
Median	0.0206	0.0000	0.0000
75th Percentile	0.4971	0.2308	0.2584

نمایش دو بعدی feature ها با استفاده از الگوریتم PCA :

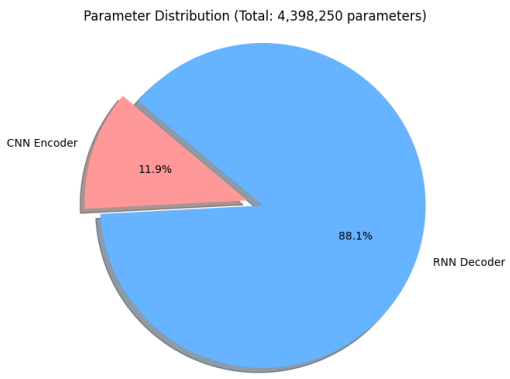


ResNet18ممکن است سیگنال‌هایی متنوع‌تر و تفکیک‌پذیرتر برای دیکودر فراهم کند، اما در عین حال می‌تواند دچار ویژگی‌های نویزی یا تکراری شود.

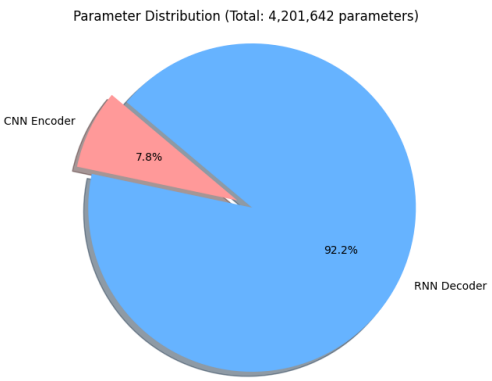
ResNet50ویژگی‌هایی نرمال‌سازی‌شده و یکنواخت‌تر تولید می‌کند که احتمالاً منجر به تعمیم بهتر و همگرایی سریع‌تر در فرآیند آموزش می‌شود.

MobileNetV2با وجود کارایی بالا، ممکن است به دلیل ظرفیت نمایش محدود، نیاز به تنظیمات بیشتر یا به‌کارگیری مکانیزم‌های توجه (Attention) داشته باشد تا کیفیت نمایش ویژگی‌های آن افزایش یابد.

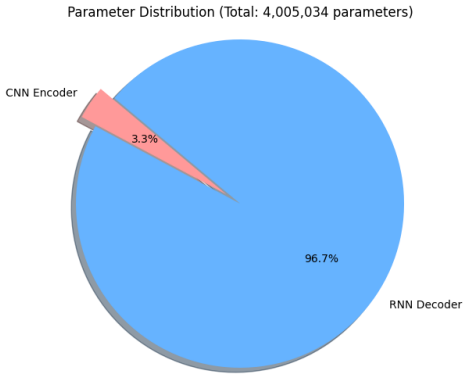
در تصاویر زیر نسبت پارامتر های مدل های متفاوت در صورت استفاده شدن به عنوان backbone را مشاهده خواهید کرد.



Inception



Mobilenet



Resnet18

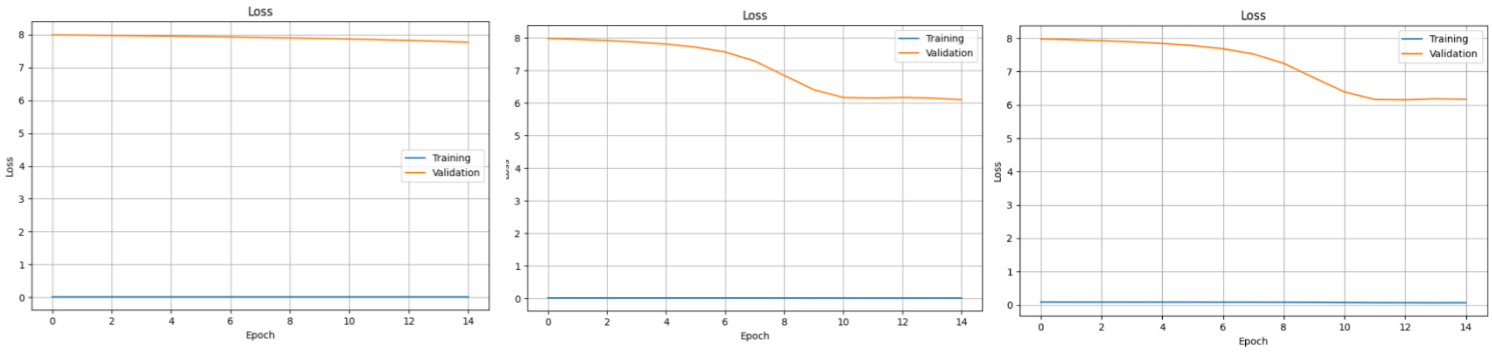
در نهایت برای مدل خود تصمیم به استفاده از resnet18 برای یک experience نسبتا بالانس صورت گرفت که در مرحله آموزش شرایط آن را بررسی خواهیم کرد.

Model Training

فرایند آموزش مدل با استفاده از config زیر صورت گرفت.

Parameter	Value	Description
`batch size`	64	reasonable batch size
`num of workers`	4	predetermined
`encoder_model`	`resnet18`	as mentioned in previous part
`embed_size`	256	Size of the shared embedding space
`hidden_size`	512	Size of the RNN hidden state
`num_layers`	1	Number of RNN layers
`dropout`	0.5	Dropout rate for regularization
`decoder_type`	`lstm`	Options: `'lstm'`, `'gru'`
`vocab_size`	2996	Set based on vocabulary analysis in notebook
`learning_rate`	1e-3	Learning rate (10^{-3})
`num_epochs`	15	Number of training epochs
`early_stopping_patience`	5	Stop training if no improvement after this many epochs

فرایند آموزش این مدل همانطور که عقب تر گفتیم و توقع داشتیم اصلا خوب و قابل قبول نبود! نتایج سه بار اجرای کامل آموزش با تغییرات کوچکی که اندکی به بهتر شدن آن کمک کردند را مشاهده و بررسی میکنیم، ابتدا آموزش اول را مشاهده میکنیم که در آن به دلیل biased شدن شدید کپشن ها توسط کلمات با frequency بالا که قبلا به آن اشاره کردیم. مدل از یک training loss بسیار کم و غیر منطقی شروع کرد و نتوانست validation loss را به هیچ مقدار قابل توجه کاهش دهد، همچنین BLEU score کپشن های جنریت شده هیچ مقدار بالا نمیرفت!

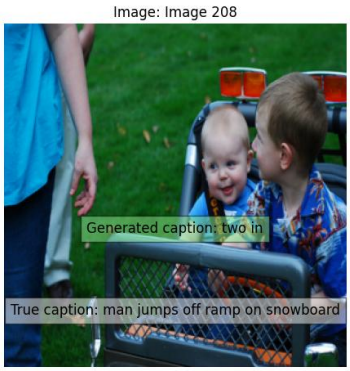


برای اندکی بهبود این مدل به سراغ روش های استفاده شده در خود مقاله show and tell رفتیم، تفاوت هایی که آن مقاله با مدل ما داشت، استفاده از یک دیتاست بهتر، و همچنین آموزش تعداد بسیار زیادی از لایه های شبکه Encoder به جای یک لایه ای که ما آموزش میدهیم، استفاده از label smoothing و یک learning rate بالا تر بود، به این دلیل که دو مورد اول برای ما دست نیافتنی بود، دو رش دوم را اعمال کردیم و مدل عملکرد بهتری داشت (خیلی کم...) . و در نهایت در آموزش سوم که تلاش کردیم تصاویر آموزشی تکراری کمتری نشان مدل دهیم باز هم به اندازه بسیار کمی مدل بهتر اما هنوز غیر قابل فهم و یادگیری ماند.

چرا مدل مقاله NIC کار میکند ولی مدل ما نه؟

NIC Paper (2015)	Your Project
Fine-tunes CNN after RNN stabilizes	CNN is frozen throughout
Uses **LSTM with tuned depth and memory**	Single-layer LSTM, fixed hidden size
Uses **label smoothing**	Just added (but late)
Implements **scheduled sampling**	Not yet implemented
Uses **Beam Search** (k=3 optimal)	Not Supported
Trains with **large datasets** (MSCOCO)	Using Flickr8k (much smaller)
Applies **dropout + ensembling**	Not used yet
Caption generation is **log-likelihood driven**, but inference-time mismatch fixed with **scheduled sampling**	Your training only uses ground-truth inputs (teacher forcing)

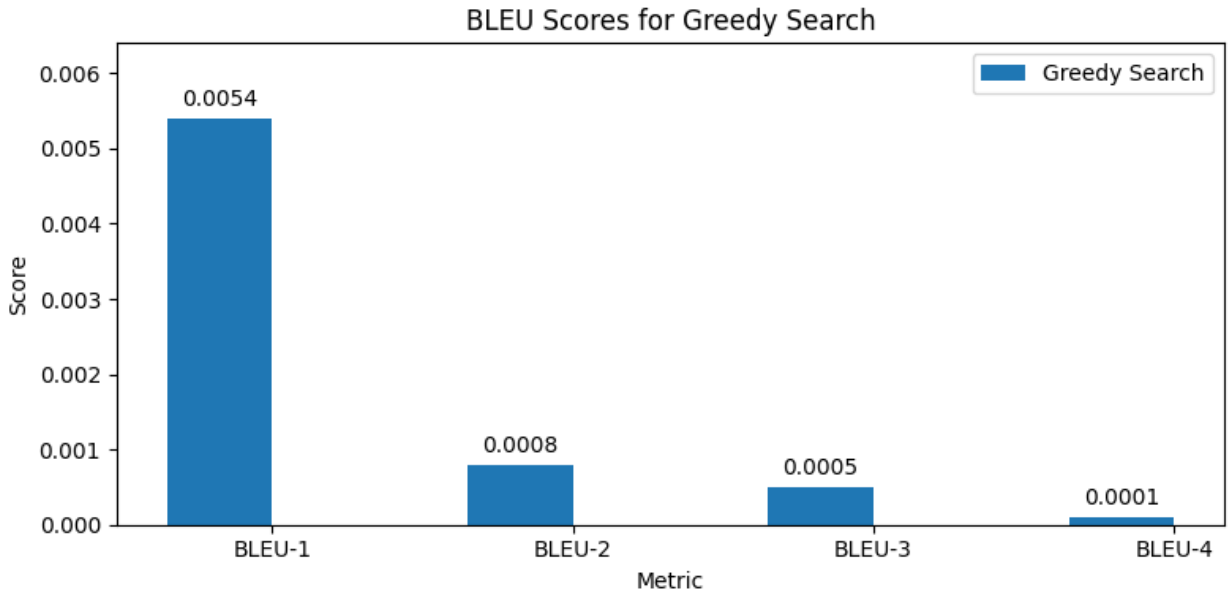
همینطور که مشاهده میشود مدل ما یک مدل بسیار تا بسیار ساده شده از مدل NIC میباشد، با وجود این سادگی باز هم در صورت استفاده از یک دیتاست بالانس تر احتمال دریافت خروجی های بهتر ممکن بود، حالا بیاید خروجی های فاجعه مدل رو باهم مشاهده کنیم.



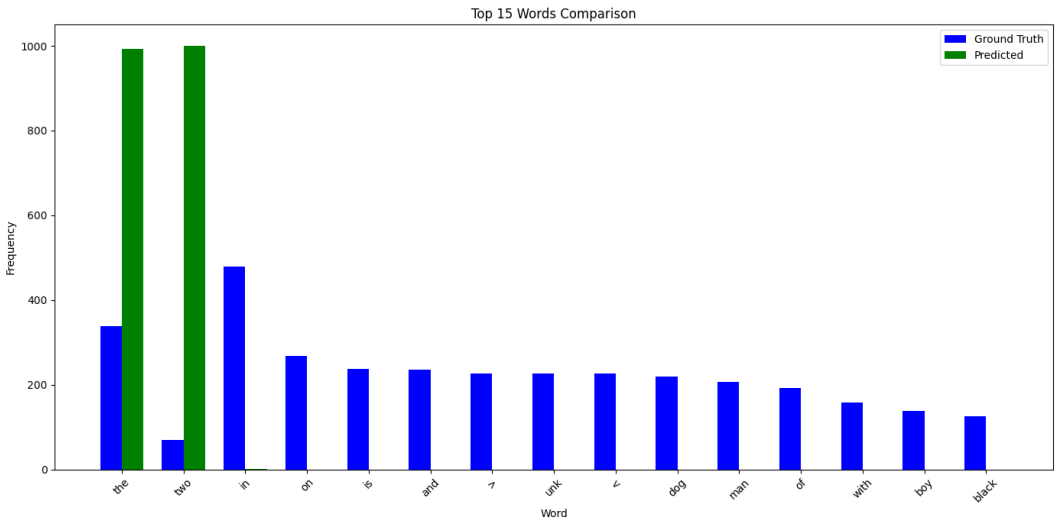
BLEU : Bilingual Evaluation Understudy

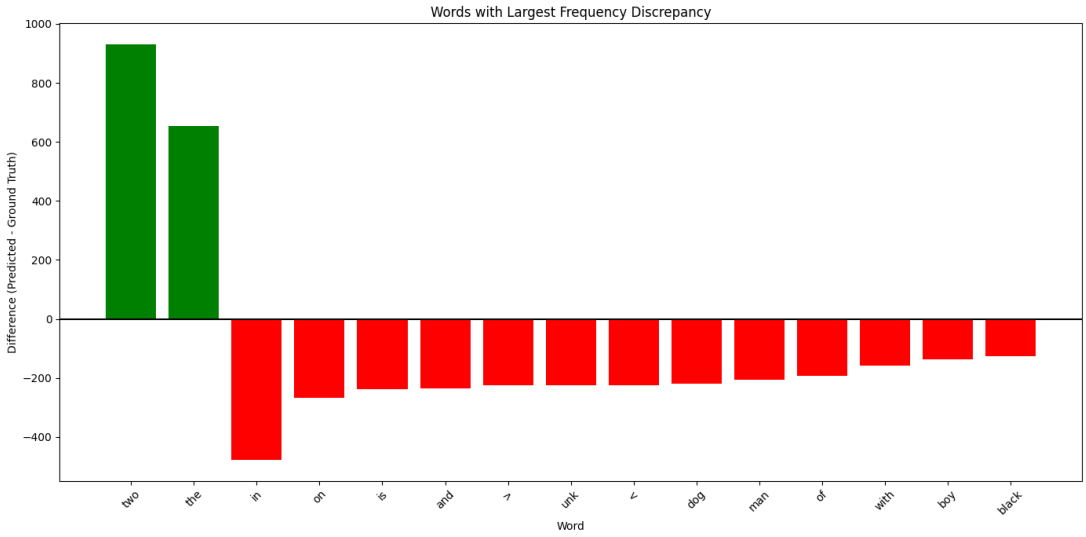
از معیارها برای ارزیابی کیفیت ترجمه ماشینی یا تولید متن مثل کپشن‌سازی تصویر است .نمره‌ای عددی بین ۰ تا ۱ (یا ۰ تا ۱۰۰) است که نشان می‌دهد چقدر متن تولیدشده توسط مدل (مثلاً کپشن تصویر) به متون مرجع انسانی شباهت دارد.

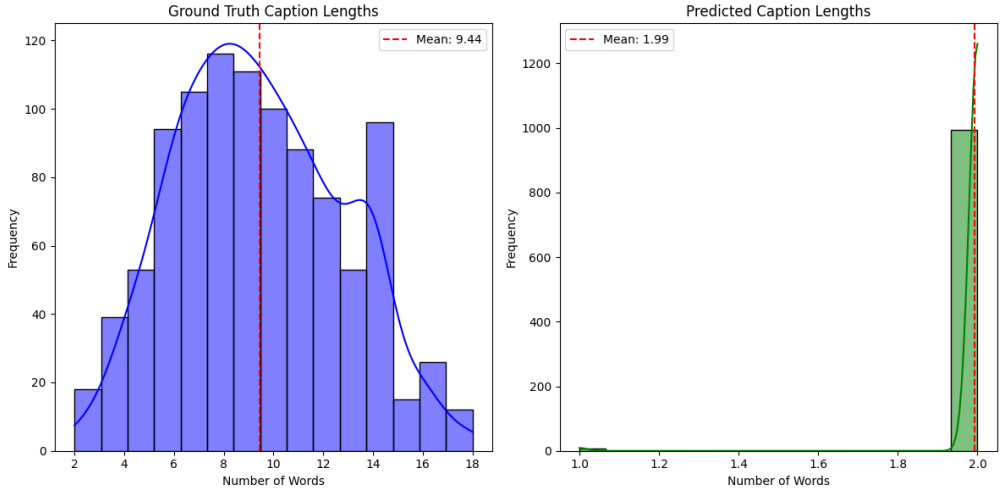
BLEUبا مقایسهn-gram ها (توالی‌های یک، دو، سه یا چند کلمه‌ای) بین کپشن تولیدشده و مرجع انسانی، میزان شباهت آنها را می‌سنجد. این معیار به‌جای recall، بر precision تمرکز دارد؛ یعنی بررسی می‌کند چه درصدی ازn-gram های تولیدی در متن مرجع هم وجود دارند. علاوه بر این، اگر کپشن بیش از حد کوتاه باشد، BLEUبا اعمال BrevityPenalty نمره را کاهش می‌دهد تا جلوی تولید جملات کوتاه ولی بی‌محتوا گرفته شود. همینطوری که از مدل ما توقع میره کپشن های generate شده توسطش BLEU score بسیار پایینی دارند.



در نمودار زیر توزیع ground truth را با کلمات پیشبینی شده مشاهده میکنیم و میبینیم که چقدر زیاد مدل ما biased روی تعداد خاصی از کلمات میباشد و تقریبا طول تمام جمله های generate شده 2 تا میباشد.







Conclusion & Improvements

تمام صحبت هایی که درباره بهبود و دلایل خوب نبودن مدلمان باید گفته میشد گفته شد، اما در نهایت برای اتمام این گزارش موارد دوباره به صورت منسجم ذکر خواهند شد، مورد اول استفاده از یک دیتاست با تصاویری متفاوت تر و بزرگ تر برای جلوگیری از bias شدن مدل به دیتا ها، زیرا دیتاست flickr8k تصاویر مشابه زیاد و با caption هایی که کلماتی با frequency زیاد در آن هاست که حتی با حذف stop word ها کمکی به بهتر شدن آن نکرد، دوم اینکه آموزش لایه انتهایی encoder کافی نیست و باید لایه های بیشتری را در آموزش دخیل کنیم. همچنین LSTM نیز باید به صورت through and through آموزش یابد. استفاده از روش های دیگر Normalization شاید با وجود یک دیتای مناسب عملکرد بهتری به مدل ما بدهد. استفاده از beam search به جای greedy search مثلا با سایز 3 مثل مقاله NIC قطعا برای ما مفید خواهد بود. استفاده از scheduled sampling در کنار drop out های متعدد و ensemble learning.