Length Controllable Image Captioning

Pavlo Vasylenko (1880258), Onur Çopur (1891194), Anil Keshwani (1919705)
Lev Telyatnikov (1902392)

April 20, 2022

We aimed to create an architecture which allowed for length control of image captions. Such an architecture could allow captions to be less "lossy" in their reporting of high level image features (e.g. recognized objects) and can be considered to have applications in accessibility, for example by providing more or less detailed linguistic information to individuals with visual impairments insofar as caption length proxies detail.

Literature and Previous Work

We started by surveying relevant literature. Specifically, we used work by Deng et al. (2020) Length-Controllable Image Captioning published in ECCV, which we also refer to as LaBERT (length-aware BERT), as a basis for our initial approach. In the paper, the authors use the usual encoder-decoder image captioning architecture with input images represented using the late layer (fc6) features of 100 object proposals acquired from a Faster R-CNN pre-trained on the Visual Genome dataset (the encoder).

They elaborate on the BERT architecture by learning length information for the captions provided in MS COCO. Their central contribution is the inclusion of a this length-level embedding via an additive term when representing the tokens used in the BERT decoder. The embeddings they use consist of this *length embedding* summed with the usual word (or WordPiece) embedding and the requisite positional embedding. The length embeddings are learnt by only training corresponding rows of the embedding matrix with caption data falling into that caption length category.

We consulted various works cited in Deng et al. (2020) including Kikuchi et al. (2016) Controlling Output Length in Neural Encoder-Decoders EMNLP, which informed our alternative approach. In effect, this approach was very similar, involving a simpler LSTM decoder which is again given learnt length-embedding information at each time step in the autoregressive decoding process. The authors also attempt use of a modified beam search procedure wherein the EOS end-of-sequence token is suppressed (by masking with $-\infty$ inside the softmax) at inference time. We report our modified version of the first approach from Kikuchi et al. (2016) but omit the beam search approach which yielded poor performance.

Approaches

LaBERT

Training

Deng et al. proposed 4 caption length levels to create the *length-aware embeddings* by training on COCO: 7-9, 10-14, 15-19 and 20-25 and we used the same categories for our model. The codebase accompanying the paper did not make the authors' results replicable and storage and processing limitations using Google Colab made use of a pre-processed image features dataset provided inviable[1]. As such, we simplified the original idea from LaBERT by replacing the encoder with ResNet output, in particular, the last convolutional layer output. By applying 6×6 adaptive average pooling, we obtained image representations of $6 \times 6 \times 2048$ pixels which we flattened and fed to a 2-layer fully-connected network using ReLU activations and dropout with probability 0.1. The last layer is of dimensionality 258. The image representation vector returned from this

[1]The dataset accompanying the paper is provided following Zhou et al. (2020) Unified Vision-Language Pre-Training for Image Captioning and VQA but totals over 160GB in compressed form.

component is then concatenated with length-aware embeddings (also 258, hence overall is 516) and fed to a pre-trained BERT (BERT Base consisting of 110M parameters[2], as opposed to BERT Large). It's worth noting that we tried both fine-tuning the ResNet encoder during training or freezing the pre-trained model. In our experiments, fine-tuning simply slowed training with little to no performance improvements. As such, the ResNet encoder was ultimately not fine-tuned during the training procedure.

For training, we used the usual Masked Language Model (*cloze*) task wherein a specified number of input tokens are replaced by a [MASK] token and the model is trained by minimizing cross-entropy loss over all masked positions. In our work, we tried masking 30% and 50% of tokens. The different proportions of masking during training yielded indistinguishable performances across experiments so the latter 50% was chosen. The authors used the so-called Label Smoothing Loss (with a smoothing parameter of 0.1) in their implementation to avoid model overconfidence and we retained this for our training.

Inference

At prediction time, we follow the process of *iterative refinement*[3] where for each length level, we feed the trained model a sentence (sequence) consisting of all [MASK] tokens where the sentence length is the upper bound of the desired length level, for example in the case of length level 4, the initialization sentence would be of 25 [MASK] tokens long. After obtaining predicted tokens, we compute a score for each token in the generated caption which is either (1) the maximum probability (corresponding to the most likely token for that position) if a token is masked at the current iteration, or else (2) it is the simple mean of the previous score and the maximum probability of a prediction. Overall, this process provides a means of updating tokens since a proportion of the tokens can be selected for re-masking in the next iteration on the basis of these produced confidence scores.

For length levels 2 and 3, we artificially decrease the probability of outputting an EOS end-of-sequence token (in our case, the [SEP] token) at the beginning of a sentence by multiplying the probability by a constant (between 0 and 1) raised of a power and then return it to its true probability (as produced by the model) by decreasing the power to which this constant is raised until it reaches 0, resulting in no suprression of the EOS token. This is done to make our captions of desired length.

Implemention

As mentioned, Deng and colleagues supply a codebase on GitHub, but this was not possible to run in its given state as the encoder relies on a very large dataset which itself is frequently unavailable via the server. Therefore, to make everything work smoothly, we had to combine code from another repository[4]. In particular, we co-opted the data loading and encoder components, and rewrote other useful functions, for example in order to encode captions with BertTokenizer.

Performance

We report below the BLEU scores according to caption length levels.

	BLEU@1	**BLEU@2**	**BLEU@3**	**BLEU@4**
Length Level 1	0.51747251	0.24986884	0.09854183	0.03552699
Length Level 2	0.46756844	0.22495891	0.09488268	0.03662744
Length Level 3	0.39600186	0.18785209	0.07572873	0.02907017
Length Level 4	0.32329879	0.14842255	0.05583477	0.01976678

[2]Devlin et al. (2019) BERT: Pre-training of Deep Bidirectional Transformers forLanguage Understanding
[3]Ghazvininejad et al. (2019) Mask-predict: Parallel decoding of conditional masked language models EMNLP
[4]https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Image-Captioning

InceptionV3 + GRU

Model

The implementation of our second approach is based on the *TensorFlow Image captioning with visual attention* tutorial. In this approach, first the input images are passed to InceptionV3 (which is pre-trained on Imagenet) to extract features from the last convolutional layer. This feature extraction step produces $8 \times 8 \times 2048$ tensors for each image in the dataset. Next, these features are squashed to 2 dimensions with shape 64×2048 and passed to the CNN encoder, which is a fully-connected network with number of output neurons equal to the embedding dimension; it uses the ReLU non-linearity. The resulting vectors with the initial hidden state from the RNN decoder are then passed to the Bahdanau attention unit to compute the context vector and attention weights. The RNN decoder which predicts the next word in the captions is a Gated Recurrent Unit (GRU). In the tutorial, the GRU takes the context vector and word embedding as input at time t and returns a prediction for the word and hidden state at time $t + 1$. To generate captions with variable lengths, we created an additional embedding - the *length embedding* - and pass the desired length information to the GRU by summing the word and length embeddings. This vector is then concatenated with the context vector and forms the input for the decoder. In figure 6, you can see the architecture of the model we constructed.

Training

To train the model we mentioned above, we set the decoder parameters as: embedding dimension and hidden size to 256, vocabulary size to 5000. Encoder parameters: number of features to $(64, 2048)$ and number of attention features to 64. For the optimizer we used Adam with learning rate 0.001 and sparse categorical cross entropy loss. To avoid overfitting, a dropout probability of 0.5 is added to the decoder. With this parameters the model is trained for 25 epochs with a batch size of 64. The model was trained on the sampled COCO dataset with size equal to 30000. The balance of the dataset with respect to caption length level categories can be seen on figure 5. During the training we used Teacher Forcing, i.e. the use of the correct ground truth label in the autoregressive output generation procedure at training time.

Performance

Our second architecture yielded the following BLEU scores.

	BLEU@1	**BLEU@2**	**BLEU@3**	**BLEU@4**
Length Level 1	0.79235876	0.57364857	0.36816953	0.27723183
Length Level 2	0.77084645	0.57162517	0.37395903	0.28099373
Length Level 3	0.68439230	0.52432671	0.34302701	0.25107386
Length Level 4	0.63188866	0.49659187	0.32324233	0.23039007

Problems

Balancing COCO Caption Lengths

We have faced a lot of different types of problems. Here we are going to highlight the most interesting. One of the main problem which we had to cope with is balancing the dataset. To train the InceptionV3 + GRU approach we decided to use COCO dataset. Overall there are 414112 captions for 82783 images. The length levels used were the same as those for the LaBERT approach. The distribution of lengths of the captions can be seen on figure 1 and figure 2.

First of all, we discarded all captions which lengths greater than 25, then we got the proportions of each level in the dataset and sampled with weight equal to the inverse of the the level proportion in the dataset. The distribution of sampled datasets (for the first 10, 20 and 30 thousand samples can be found in the appendix figures 3, 4, 5.

LaBERT

Our simplified model of LaBERT was hard to train because, strangely, training and validation loss were always decreasing but this did not correspond necessarily to the BLEU evaluation metric we used. We could spot improvements visually just by looking at actual predictions for captions after increased training time or when loss was lower. This disconnect (at least non-monotonicity) between our training loss metric and the NLP metrics we used for validation is hard to reconcile.

We faced an interesting situation when we found that we wrongly masked input by omitting the `[CLS]` and [SEP] tokens. Such a bug results in artefacts of output when the beginning of a sentence was always incorrect and might contain words or tokens which cannot be in the first place of sentence by default.

Finally, as already mentioned, fine-tuning of the encoder only increases training time substantially whilst bringing limited performance gains.

Conclusions

There are many takeaways from this piece of work. One of the main aspects of the prediction-time behaviour of both the LaBERT-based and InceptionV3 + GRU + Attention architectures is that we are only specifying the length-embedding information learnt at training time in the form of an additive change to the embedding vectors fed to the decoder network. The model is not constructed so as to produce captions of length within the desired range in a deterministic manner. This can be seen from the example outputs in Figures 8 and 9. Whilst it might be reasonable to ask if the simple addition of the embedding vector might be either lossy or noisy, it is worth noting that this approach to constructing embeddings is prevalent, especially when using transformers (e.g. Vaswani and colleagues specify positional embedding to their original Transformer via additive terms, or the additive relationship of latent concepts in Mikolov's word2vec word embeddings[5]). We experimented with different embedding dimensions but this at best did not improve performance.

A second key point is that the BLEU evaluation metric, which constitutes a modified form of precision wherein words in the predicted text may only be "rewarded" up to the maximum number of times they appear in the reference texts, misses several desirable aspects of "good" captions. For example, grammatical correctness of the captions can only be judged by examining BLEU computed using increasingly large n-grams (e.g. BLEU4) but this necessarily forces a trade-off wherein captions are penalized for not matching reference texts exactly. Whilst use of other metrics (such as the ROUGE-S metric calculated on skip-bigrams) may overcome this limitation, the best solution would certainly be to use a range of metrics. We report BLEU score for its simplicity, ubiquity and for the fact that it was also used in the literature we consulted.

In the future, further work might look at experimenting with different weighting of the length embeddings when adding these to the token embeddings or exploitation of different (potentially "richer") input embeddings to the Transformer decoder (e.g. a model pre-trained on the Visual Genome dataset).

Erratum

Following the first version of this report, BLEU scores (see subsection on *Performance* under *InceptionV3 + GRU*) for the second architecture have been corrected.

[5]Mikilov et al. (2013) Distributed Representations of Words and Phrases and their Compositionality NIPS

Appendix

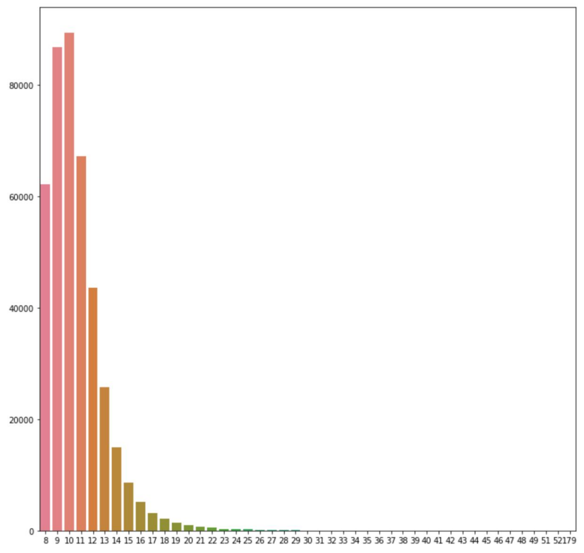


Figure 1: Initial distribution of captions lenghts

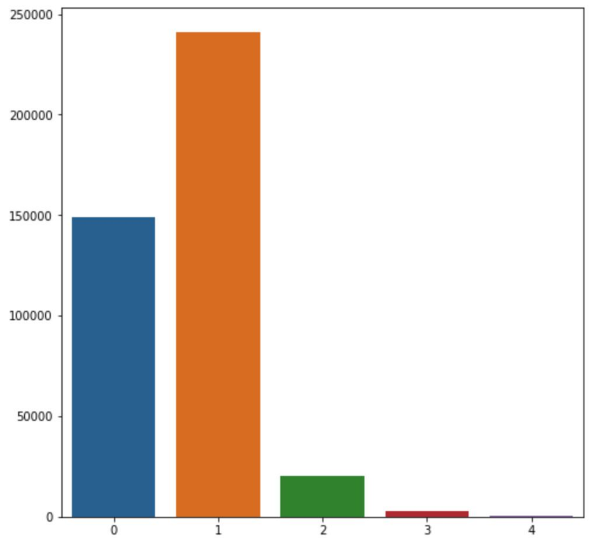


Figure 2: Initial distribution of captions lenghts with respect to the levels. The levels 0,1,2,3 corresponds to the 7-9, 10-14, 15-19, 20-25 captions lengths. Level 4 corresponds to all captions which lengths greater than 25

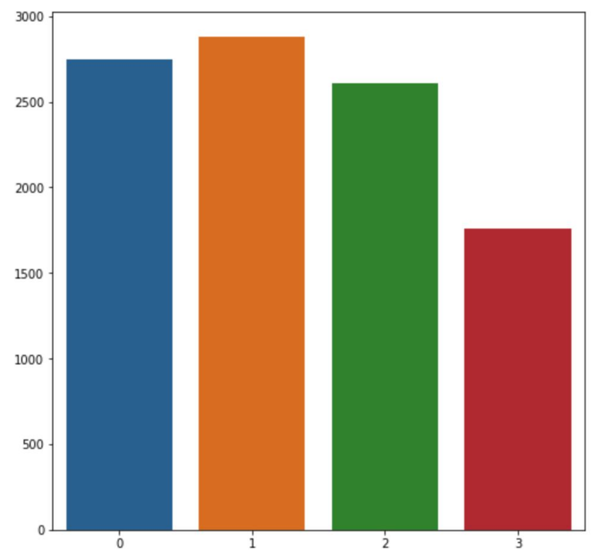


Figure 3: 10 thousand samples

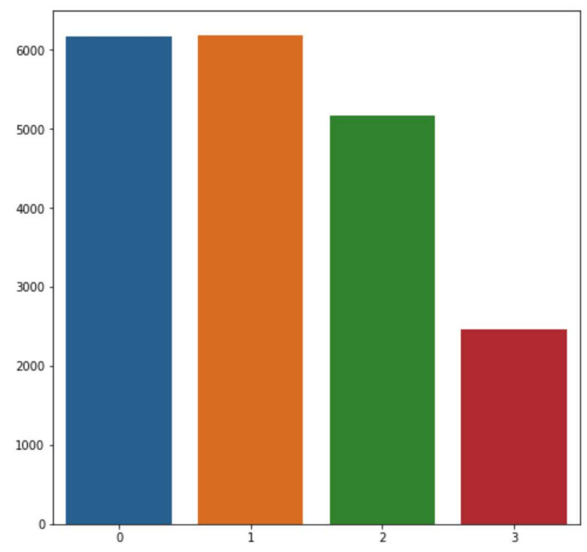


Figure 4: 20 thousand samples

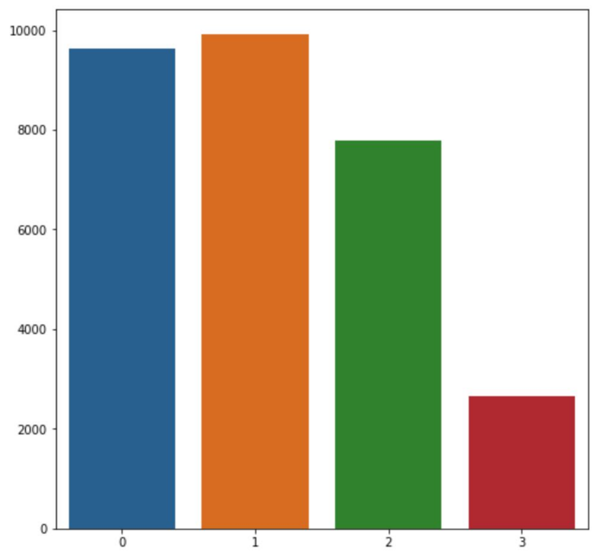
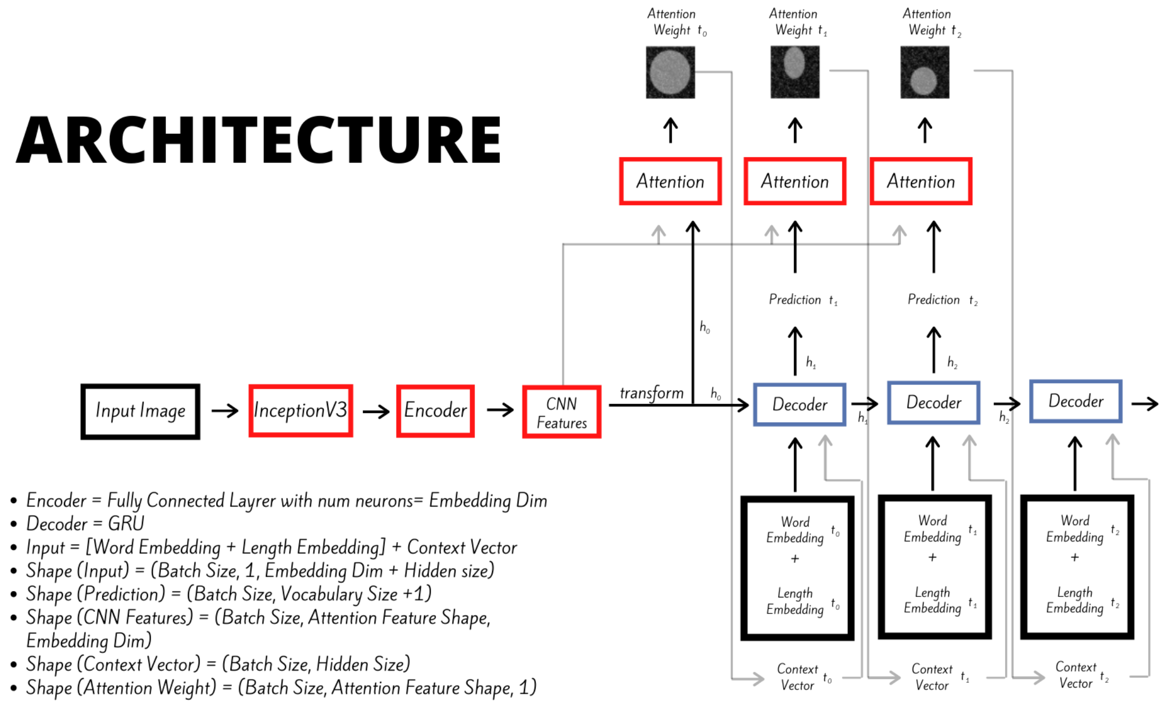
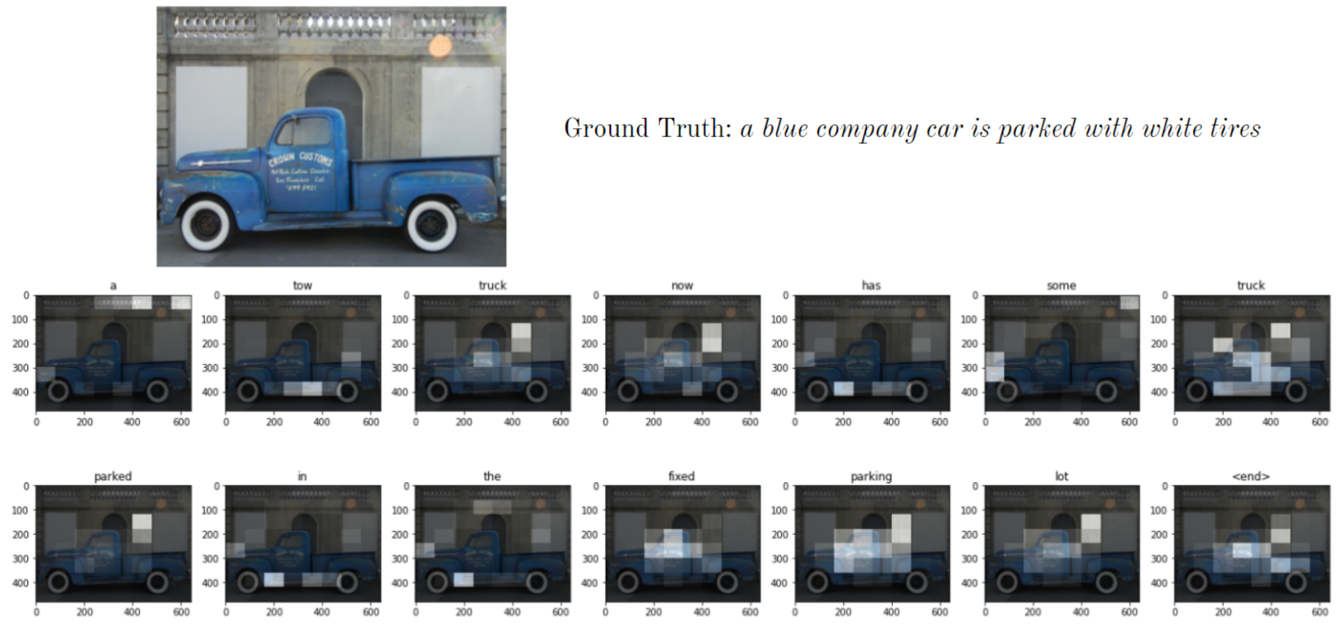


Figure 5: 30 thousand samples



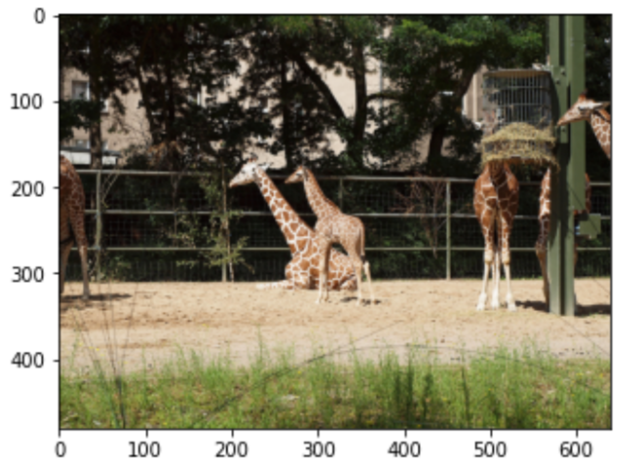
- Encoder = Fully Connected Layrer with num neurons= Embedding Dim
- Decoder = GRU
- Input = [Word Embedding + Length Embedding] + Context Vector
- Shape (Input) = (Batch Size, 1, Embedding Dim + Hidden size)
- Shape (Prediction) = (Batch Size, Vocabulary Size +1)
- Shape (CNN Features) = (Batch Size, Attention Feature Shape, Embedding Dim)
- Shape (Context Vector) = (Batch Size, Hidden Size)
- Shape (Attention Weight) = (Batch Size, Attention Feature Shape, 1)

Figure 6: Model architecture of InceptionV3 + GRU



Ground Truth: *a blue company car is parked with white tires*

Figure 7: An example of a caption produced by the InceptionV3 + GRU architecture



Ground truth: *some giraffes are laying down in a pin.* **(8 words)**

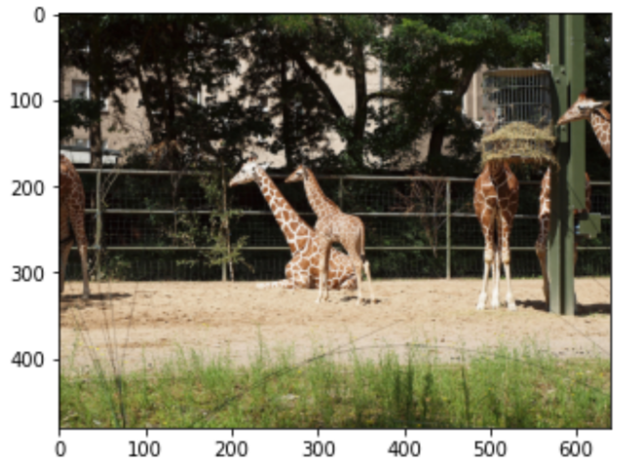
Level 1 Caption: *a giraffe are walking around on the fence.* **(8 words)**

Level 2 Caption: *a giraffe standing on the fence. (6 words)*

Level 3 Caption: *this giraffe along side of another giraffe as they stand on the ground with their heads at a wooden fence.* **(20 words)**

Level 4 Caption: *this giraffe <unk> on licking area reaching out here and reaching out side of focus of grass a blackboard on one horse in front of the camera.* **(26 words)**

Figure 8: Determining Caption Length at Prediction Time - Caption lengths respond to the length-embedding information specified at prediction time



Ground truth: *some giraffes are laying down in a pin.* **(8 words)**

Level 1 Caption: *a giraffe are walking around on the fence.* **(8 words)**

Level 2 Caption: *a giraffe standing on the fence. (6 words)*

Level 3 Caption: *this giraffe along side of another giraffe as they stand on the ground with their heads at a wooden fence.* **(20 words)**

Level 4 Caption: *this giraffe <unk> on licking area reaching out here and reaching out side of focus of grass a blackboard on one horse in front of the camera.* **(26 words)**

Figure 9: Determining Caption Length at Prediction Time - Caption lengths are not guaranteed to fall without the length-level bounds specified at training time, as this information is only provided via the length-embeddings and so is not deterministic.