
Analysis of Multi-Modal Feature Representations by CLIP for Image Captioning

Saad Saleem

Department of Computer Science
University of Toronto
saad@cs.toronto.edu

Shamitra Rohan

Department of Computer Science
University of Toronto
shamitra@cs.toronto.edu

Malikeh Ehghaghi

Department of Computer Science
University of Toronto
malikeh.ehghaghi@mail.utoronto.ca

Abstract

Image captioning requires visual understanding and linguistic processing to learn relationships between images and text. Transformer-based models have made significant improvements in this aspect as they can represent multi-modal spaces into a combined semantic space. We analyze the performance on image captioning of the transformer-based CLIP pre-training model on the VizWiz-Captions dataset. We establish baseline metrics by implementing a Convolutional Neural Network - Recurrent Neural Network model on the same dataset. Our main contribution is to use CLIP’s image encoder to improve the performance of the baseline model on image captioning, a task on which CLIP’s performance has not been studied extensively. A link our code is in the Appendix section 7.1.

1 Introduction

Vision-Language tasks in deep learning such as image captioning have gained considerable popularity in the recent past. Image captioning combines both Computer Vision (CV) and Natural Language Processing (NLP), by generating captions for images through object and scene recognition, along with learning relationships among objects. Since image captioning relies on the quality of image-text embeddings, large-scale pre-training to learn multi-modal image-text representations is fundamental to achieve state-of-the-art (SoTA) performance.

Image captioning approaches often involve an encoder-decoder architecture in which the encoder extracts semantic information from the image and the decoder uses natural language to generate a caption from the image features. These frameworks are comprised of a Convolutional Neural Network - Recurrent Neural Network (CNN-RNN), with the RNN using a Long Short-Term Memory (LSTM) architecture to maximize the likelihood of the generated captions given the image features (1). Although LSTM models use a memory cell to remember long-term information, they are sequential in nature, i.e. caption generation is linearly dependent on caption length. To combat this, transformer architectures use deep bi-directional attention mechanisms enabling faster parallel computations.

In this paper, we study the performance of an encoder-decoder model for image captioning on the VizWiz dataset (2). We establish baseline metrics by implementing a traditional CNN-LSTM model, and improve this by using CLIP (3), a pre-trained SoTA encoder with the LSTM decoder. Finally, we experiment by using a CLIP encoder with an attention-based LSTM decoder.

2 Related Work

Early methods for image captioning mostly relied on retrieval (4)(5)(6) and template-based (7)(8)(9) approaches with hard-coded rules and hard-engineered features, resulting in models with limited expressiveness. With advancements in deep learning, several methods have emerged that enhance image captioning performance. Mao et al. (10) proposed a multi-modal model, in which a CNN extracts image features and a multi-modal RNN models word distributions conditioned on the image features and context words. Extending this to encoder-decoder frameworks, image captioning can be formulated as a modification of Neural Machine Translation (NMT), with the input being an image and the output being the caption. Vinyals et al. (11) proposed the Neural Image Caption (NIC) model that makes use of this architecture. Xu et al. (12) extended this framework by adding an attention mechanism, that is able to attend to salient image regions during caption generation.

The introduction of transformers in deep learning has led to the emergence of large scale pre-training methods for vision-language tasks. These methods are typically trained on a corpus of image-text pairs and can be classified as single-stream (13)(14)(15) (using one transformer to learn multi-modal representations) or two-stream (16) (using individual transformers and a cross-modal unifying transformer). However, representations of objects in the visual space are often oversampled and noisy, and without explicit information about image-text region alignment, the task becomes more complicated. OSCAR (17) is a SoTA pre-training method that combats this by using image tags as anchor points to learning better image-text representations.

3 Methods

The model design is based on the encoder-decoder framework used in NMT (Figure 1, Appendix). Our model uses the encoder to learn image representations and the decoder to produce a caption for it. In particular, we use three models: a CNN-LSTM, CLIP-LSTM, and a CLIP-LSTM (attention).

3.1 Encoder

The baseline model in Figure 1 (Appendix) consists of a CNN encoder which is a pre-trained ResNet-50 model trained on the ImageNet dataset (19)(20). The encoder extracts features from a batch of pre-processed images. The pre-processing step involves downsampling and cropping images to match the input requirements (224, 224, 3) of the ResNet-50 model. The final fully-connected layer of the Resnet-50 model is removed and the output is flattened before being passed through a linear layer to match the embedding size of the decoder. During training, the parameters of the Resnet-50 model are frozen and only the weights in the final linear layer are updated.

We also incorporated a CLIP encoder in place of the baseline encoder, resulting in a CLIP-LSTM model. Trained on 400 million image-text pairs from the internet, CLIP is a SoTA transformer-based neural network that uses contrastive image-text pre-training to classify images (3). Specifically, we use the image encoder of CLIP, since it is known to perform well on many zero-shot image classification tasks/datasets. Furthermore, it is not tested for image captioning in particular. To implement the image encoder, we applied the ViT-B/32 pre-trained model (Figure 2) which was reported to have the best performance among the image encoder models used in the CLIP paper (21).

3.2 Decoder

The baseline decoder used is a unidirectional LSTM with 2 layers. The encoded image is fed into the LSTM's first unit while the ground truth caption is fed into subsequent units, a concept known as Teacher Forcing (22). The LSTM generates the output tokens one by one, where each output depends on the current hidden state and previous output. All parameters of the LSTM decoder are updated during training. We minimize the cross-entropy loss between ground truth and generated tokens in order to train the network.

Expanding on the CLIP-LSTM model, we applied an attention-based LSTM decoder shown in Figure 3. With attention, the decoder looks at different parts of the image at different points in the sequence. The attention network computes weights which we use to average across all pixels, with the weights of the important pixels being larger. This weighted image representation can be concatenated with the previously generated token at each step to generate the next one. We used soft attention, where

the weights of the pixels add up to 1. By treating the attention locations as intermediate latent variables, we can assign a multinoulli distribution parametrized by α_i and view \hat{z}_t as a random variable. Learning stochastic attention requires sampling the attention location s_t each time, instead, we can take the expectation of the context vector \hat{z}_t directly.

$$E_{p(s_t|a)}[\hat{z}_t] = \sum_{i=1}^L \alpha_{t,i} a_i = 1 \quad (1)$$

We can formulate a deterministic attention model by computing a soft attention weighted annotation $\phi(a_i, \alpha_i) = \sum_i^L \alpha_i a_i$, while a_i is the i^{th} image feature (12). Note, the model is trained end-to-end by minimizing the following penalized negative log-likelihood using doubly stochastic regularization to encourage the model to attend equally to every part of the image over the course of generation (2).

$$L_d = -\log(P(y|x)) + \lambda \sum_i^L (1 - \sum_t^C \alpha_{t,i})^2 \quad (2)$$

4 Experiments and Analysis

4.1 Dataset and Metrics

The dataset used in this paper is the open-source VizWiz-Captions (2), which contains 39,181 images in total that are each paired with 5 captions of varying length. This dataset represents images taken by blind people and is intended to be used to create technology capable of describing images accurately to them. Captions are pre-processed by replacing words that occur fewer than 6 times with the '<unk>' token and adding '<start>' and '<end>' tokens around the sentence. The dataset is split into train (23,431 images, 117,155 captions), validation (7,750 images, 38,750 captions), and test (8,000 images) sets. Although train and validation sets have image-caption pairs, the test set has no captions provided. Hence, our metrics are reported on the validation set and we do a qualitative analysis on the test set. We measure our models against SoTA metrics that are often used in language tasks, i.e BLEU (23), ROUGE (24), METEOR (25), CIDEr (26), and SPICE (27) scores. While BLEU and ROUGE weakly correlate with human judgement, METEOR and CIDEr are comparatively better. CIDEr measures sentence similarity, grammar, importance, and accuracy (precision and recall) (26).

4.2 Evaluation Procedure

We performed several experiments to evaluate the performance of various encoder-decoder architectures. Specifically, we experimented with the batch size, learning rate, and depth of the LSTM network. All experiments were performed using the Adam optimizer (29) and trained for 1 epoch. Model performance on the validation set was determined by the best CIDEr score, as it is widely used and correlates well with human judgement. The best parameter values for the baseline CNN-LSTM model were established and were then extrapolated to the other two models, i.e. CLIP-LSTM and CLIP-LSTM (attention), to allow for better comparisons. The final three models were trained for 5 epochs and the best results have been summarized in Table 1.

Batch sizes [32, 64, 128, 256, 512] were varied in the model. Our results from Table 2 indicated that batch size 32 produced the best results over most metrics including CIDEr. Smaller batch size can allow for better generalization due to the noise in small-batch training. It prevents the network from overfitting on the training set by introducing small perturbations in the gradient descent steps, allowing for better learning of the network parameters.

We also experimented with learning rates [0.001, 0.005, 0.01, 0.05] on model performance. The best batch size 32 and 1 LSTM layer were used in this experiment. Results from Table 2 indicate 0.001 as the best learning rate. A lower learning rate allows for more accurate steps during gradient descent.

Lastly, we experimented with different layers [1, 2, 3, 4, 5] for the LSTM network. For this experiment, the best batch size 32 and learning rate 0.001 were used. Generally, adding more layers (or parameters) to the LSTM adds more levels of abstraction of the inputs over time and results in an increase in caption quality. Our results from Table 2 indicated that 2 LSTM layers performed the best

on CIDEr score, although 2 and 3 layers had roughly similar performance for other metrics. Adding more than 3 layers resulted in diminishing results due to overfitting on the training data.

4.3 Quantitative evaluation

Table 1: BLEU-1,2,3,4/METEOR metrics compared with other models

Model	Dataset	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR
CNN+LSTM	VizWiz	0.4	0.213	0.128	0.072	0.095
CLIP+LSTM	VizWiz	0.533	0.349	0.22	0.133	0.143
CLIP+LSTM (Attention)	VizWiz	0.241	0.152	0.092	0.053	-
IBM Research AI (30)	VizWiz	0.727	0.541	0.389	0.274	0.222
NIC (11)	Flickr30k	0.663	0.423	0.277	0.183	-
Soft-Attention (12)	Flickr30k	0.667	0.434	0.288	0.191	0.1849
Hard-Attention (12)	Flickr30k	0.669	0.439	0.296	0.199	0.1846

Table 1 summarizes the results for various models. We observe a considerable improvement in the performance of the CLIP+LSTM model over the baseline CNN+LSTM. However, the CLIP+LSTM attention-based decoder did not perform well. We suspect this is due to the difference in the way the decoders generate captions. The non-attention decoder generates captions continuously until it outputs an '<end>' token, whereas the attention-based decoder requires a preset length for the caption. There could also be model incompatibilities between the CLIP encoder and the attention-based decoder. Pixel dependencies and assigning attention weights to the outputs of the CLIP encoder may not have the same result as using raw pixel features as the input. Another observation was the discrepancy between the validation cross-entropy loss and other scores over epochs. For example, for the CNN+LSTM from epoch 2 to 3 during validation, CIDEr score improves from 0.1 to 0.125, whereas the loss increases from 3.9818 to 4.0161. This indicates that cross-entropy loss may not be the best metric to evaluate performance. We suspect our model scores to be lower due to the dataset itself, as they contain many images that are shaky, not object-focused, and are not centered.

Despite this, our findings are encouraging. We demonstrate CLIP encoder’s performance for image captioning which has not been reported to the best of our knowledge. We successfully enhance the performance of a baseline model by incorporating CLIP, which is trained on 300 times more data than the original encoder.

4.4 Qualitative Evaluation

Table 4 displays example test images and their generated captions from the CNN-LSTM and CLIP-LSTM models. The CLIP-LSTM model is able to distinguish important details in the test images better than the baseline model. In addition, we generated 2-D t-SNE plots (Figure 7) to pictorially understand model learnings for image and word embeddings.

5 Summary

In this paper, we implemented a CNN-LSTM, CLIP-LSTM, and a CLIP-LSTM (attention) model for image captioning. The baseline CNN encoder is a pre-trained Resnet-50 on the ImageNet dataset. We performed a search through various hyperparameter values and our analysis shows that replacing only the CNN encoder in the baseline model with CLIP improves caption quality and performance on all metrics, however, replacing the baseline decoder with an attention-based LSTM results in lower performance due to it requiring preset caption lengths and compatibility issues between CLIP encoding and decoder input. Our main contribution is to study CLIP’s performance on image captioning which is unreported to our knowledge. Future improvements would involve robust pre-processing and data augmentation, possibly through RotationNet (33) to learn rotation schemes for images. Optical Character Recognition could be used on images that have text with useful information, thereby enriching features. Transformer-based decoders could also be used along with ensemble techniques that have shown success.

6 Attributions

This project was worked on by Saad Saleem, Shamitra Rohan, and Malikeh Ehghaghi with equal contributions from each member.

More specifically, the dataset was studied by all members but setup and pre-processing were done primarily by Saad Saleem. The baseline CNN-RNN model was primarily set up by Saad Saleem but also worked on by Shamitra Rohan, while the CLIP-based model was worked on by Malikeh Ehghaghi. Overall, a collaborative team effort was made in obtaining findings, graphs, plots, and their subsequent analysis.

In terms of approaches that were unsuccessful, Shamitra Rohan was primarily responsible for understanding and setting up OSCAR (17), along with a contribution by Saad Saleem. Although they managed to successfully run the testing loop, the training would often stop due to GPU resource limitations. Moreover, the code was found to have many dependencies and out-of-date documentation on its repository. Experimenting on a new dataset also required the setup of a second repository (based on bottom-up attention) with limited documentation available on how to adapt its resulting features to match OSCAR's input requirements.

References

- [1] Liu, Shuang et al. "Image Captioning Based on Deep Neural Networks". MATEC Web of Conferences 232. (2018): 01052.
- [2] Danna Gurari, et al. "Captioning Images Taken by People Who Are Blind." (2020).
- [3] Alec Radford, et al. "Learning Transferable Visual Models From Natural Language Supervision." (2021).
- [4] Farhadi A. et al. (2010) Every Picture Tells a Story: Generating Sentences from Images. European Conference on Computer Vision.
- [5] Ordonez, Vicente et al. "Im2Text: Describing Images Using 1 Million Captioned Photographs." Advances in Neural Information Processing Systems.
- [6] Gupta, Ankush et al. "Choosing Linguistics over Vision to Describe Images." AAAI Conference on Artificial Intelligence.
- [7] Yang, Yezhou et al. "Corpus-Guided Sentence Generation of Natural Images." Proceedings of the Conference on Empirical Methods in Natural Language Processing.
- [8] G. Kulkarni, et al. "BabyTalk: Understanding and Generating Simple Image Descriptions". IEEE Transactions on Pattern Analysis and Machine Intelligence 35. 12(2013): 2891-2903.
- [9] Li, Siming et al. "Composing Simple Image Descriptions using Web-scale N-grams". CoNLL 2011 - Fifteenth Conference on Computational Natural Language Learning, Proceedings of the Conference. (2011).
- [10] Junhua Mao, et al. "Explain Images with Multimodal Recurrent Neural Networks." (2014).
- [11] Oriol Vinyals, et al. "Show and Tell: A Neural Image Caption Generator." (2015).
- [12] Kelvin Xu, et al. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention." (2016).
- [13] Yen-Chun Chen, et al. "UNITER: UNiversal Image-TExt Representation Learning." (2020).
- [14] Liunian Harold Li, et al. "VisualBERT: A Simple and Performant Baseline for Vision and Language." (2019).
- [15] Gen Li, et al. "Unicoder-VL: A Universal Encoder for Vision and Language by Cross-modal Pre-training." (2019).
- [16] Jiasen Lu, et al. "ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks." (2019).
- [17] XiuJun Li, et al. "Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks." (2020).

- [18] Garg, Deepesh. "Automatic Image Captioning With PyTorch." Medium (20 Aug. 2020). <https://medium.com/@deepeshrishu09/automatic-image-captioning-with-pytorch-cf576c98d319>
- [19] Kaiming He, et al. "Deep Residual Learning for Image Recognition." (2015).
- [20] J. Deng, et al. "ImageNet: A large-scale hierarchical image database." 2009 IEEE Conference on Computer Vision and Pattern Recognition.
- [21] Alexey Dosovitskiy, et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." (2020).
- [22] Lamb, Alex M et al. "Professor Forcing: A New Algorithm for Training Recurrent Networks." Advances in Neural Information Processing Systems.
- [23] Papineni, Kishore et al. "BLEU: a Method for Automatic Evaluation of Machine Translation". (2002).
- [24] Lin, Chin-Yew. "ROUGE: A Package for Automatic Evaluation of summaries."
- [25] Lavie, Alon et al. "METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments". (2007)
- [26] Vedantam, Ramakrishna et al. "CIDEr: Consensus-based image description evaluation."
- [27] Anderson, Peter et al. "SPICE: Semantic Propositional Image Caption Evaluation."
- [28] Tsung-Yi Lin, et al. "Microsoft COCO: Common Objects in Context." (2015).
- [29] Diederik P. Kingma, et al. "Adam: A Method for Stochastic Optimization." (2017).
- [30] "VizWiz-Captions Challenge 2020." EvalAI: Evaluating State of the Art in AI. <https://eval.ai/web/challenges/challenge-page/525/leaderboard/1467>
- [31] Hodosh, Micah et al. "Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics". Journal of Artificial Intelligence Research 47. (2013): 853-899.
- [32] Young, Peter et al. "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions". TACL 2. (2014): 67-78.
- [33] Asako Kanezaki, et al. "RotationNet: Joint Object Categorization and Pose Estimation Using Multiviews from Unsupervised Viewpoints." (2018).

7 Appendix

7.1 Code

All code used to train and test our models can be found here <https://github.com/Malikeh97/csc2516-final-project>

7.2 Models

The baseline CNN-LSTM model is described in Figure 1. It uses a pre-trained Resnet-50 model over which a linear layer is appended. The output representation of the CNN encoder is as follows:

$$CNN(Img) = W.Resnet(Img) + b \quad (3)$$

A review of the Vision Transformer Model applied in the CLIP image encoder is shown in Figure 2. The architecture of the attention-based decoder model utilized over the CLIP image encoder is illustrated in Figure 3.

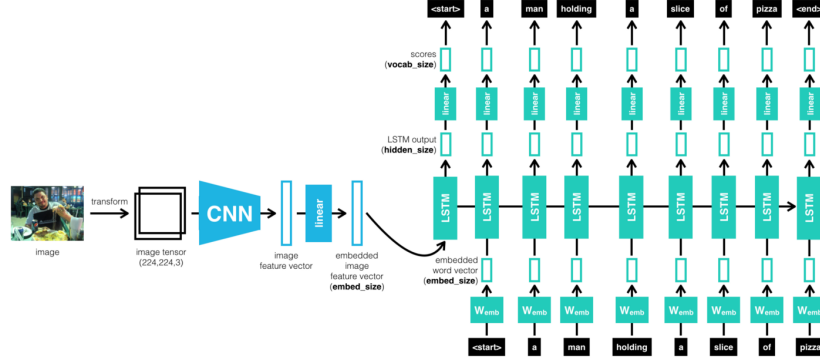


Figure 1: Baseline CNN-LSTM, based on the NIC model (11)(18). The CNN learns a vector representation of the image which is passed into the first unit of a 2-layered LSTM. The LSTM also takes the ground truth caption as input and generates the hidden states and output caption.

Model Architecture

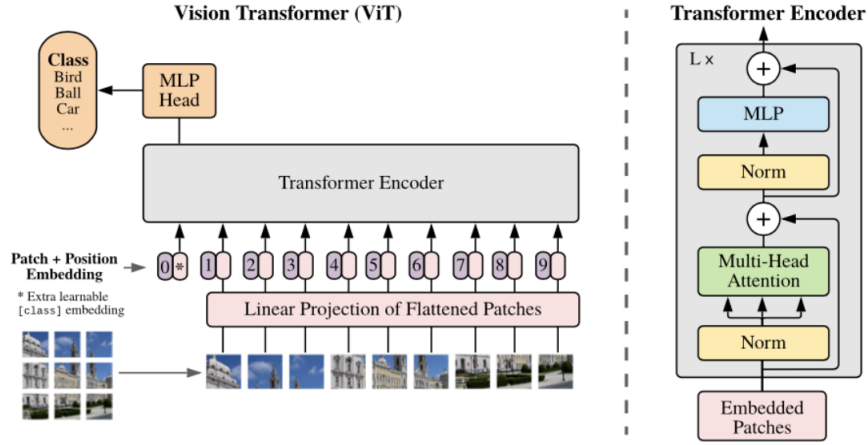


Figure 2: Vision Transformer(ViT) Model Review (21). The ViT model splits an image into fixed-size patches, linearly embeds each of them, adds position embeddings, and feed the resulting sequence of vectors to a standard transformer encoder.

7.3 Validation

Additional validation scores are included in the tables. Table (2) indicates different scores calculated under various configuration settings to experiment how batch size, learning rate, and number of LSTM layers can affect the final performance of the baseline model over the validation dataset. Also, Table 3 shows the validation scores of the best models per epoch. Additionally, the training loss versus validation loss plots are illustrated in Figures (4), (5), and (6).

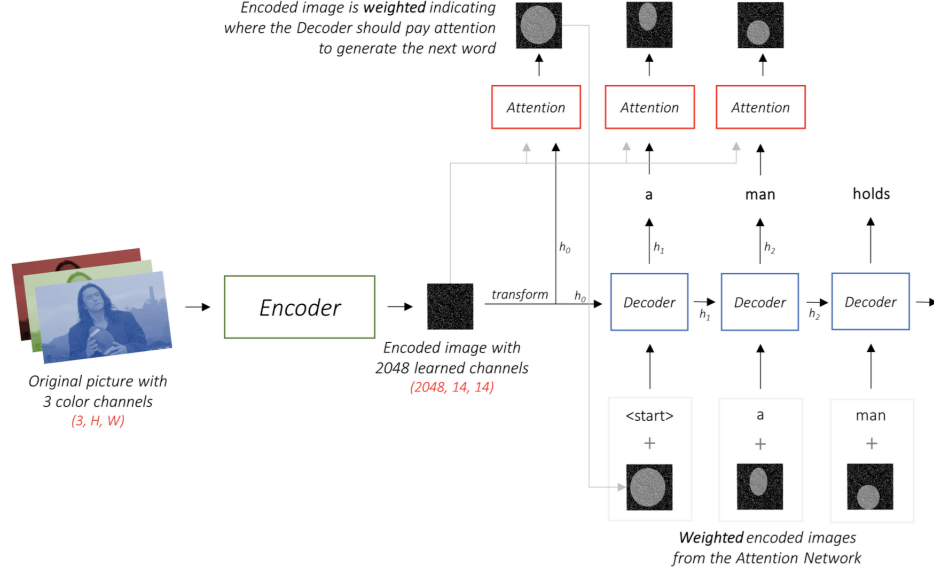


Figure 3: Attention-based Decoder Model Review (12).

Table 2: Validation scores for baseline CNN-RNN model hyperparameter tuning

	Baseline CNN-RNN							
	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	CIDEr	SPICE.	METEOR
Batch Size								
32	0.376	0.183	0.094	0.044	0.274	0.067	0.025	0.089
64	0.329	0.167	0.091	0.045	0.262	0.051	0.026	0.074
128	0.321	0.119	0.063	0.032	0.246	0.018	0.017	0.062
256	0.279	0.115	0.033	0.006	0.213	0.006	0.008	0.053
Learning Rate								
0.001	0.375	0.182	0.094	0.044	0.273	0.066	0.025	0.089
0.005	0.108	0.042	0.011	0.004	0.133	0	0.008	0.029
0.01	0.338	0.131	0.068	0.039	0.271	0.067	0.027	0.082
0.05	0.063	0.032	0.009	0.001	0.192	0.013	0.005	0.05
#LSTM Layers								
1	0.375	0.182	0.094	0.044	0.273	0.066	0.025	0.089
2	0.423	0.224	0.126	0.066	0.305	0.095	0.04	0.095
3	0.449	0.23	0.122	0.07	0.308	0.092	0.032	0.1
4	0.363	0.173	0.013	0	0.279	0.022	0.013	0.063
5	0.365	0.181	0.071	0.022	0.292	0.035	0.008	0.063

7.4 Testing and Visualization

The TSNE plot in Figure 7 shows the 2D feature representation of images vs captions produced by CLIP+LSTM model.

Table 3: Validation scores for final models

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	CIDEr	SPICE.	METEOR
CNN+LSTM epochs								
1	0.423	0.224	0.126	0.066	0.305	0.095	0.04	0.095
2	0.4	0.213	0.128	0.072	0.292	0.1	0.043	0.095
3	0.445	0.259	0.154	0.09	0.315	0.125	0.049	0.108
4	0.418	0.24	0.142	0.081	0.303	0.1	0.048	0.1
5	0.441	0.252	0.149	0.087	0.313	0.127	0.051	0.107
CLIP+LSTM epochs								
1	0.46	0.285	0.171	0.093	0.34	0.17	0.064	0.12
2	0.55	0.355	0.219	0.13	0.382	0.206	0.077	0.144
3	0.533	0.349	0.22	0.133	0.367	0.231	0.079	0.143
4	0.514	0.329	0.205	0.124	0.37	0.227	-	-
5	0.241	0.152	0.092	0.053	0.382	0.23	-	-
CLIP+Attention RNN								
1	0.253	0.131	0.065	0.033	0.233	0.004	0.07	0.092

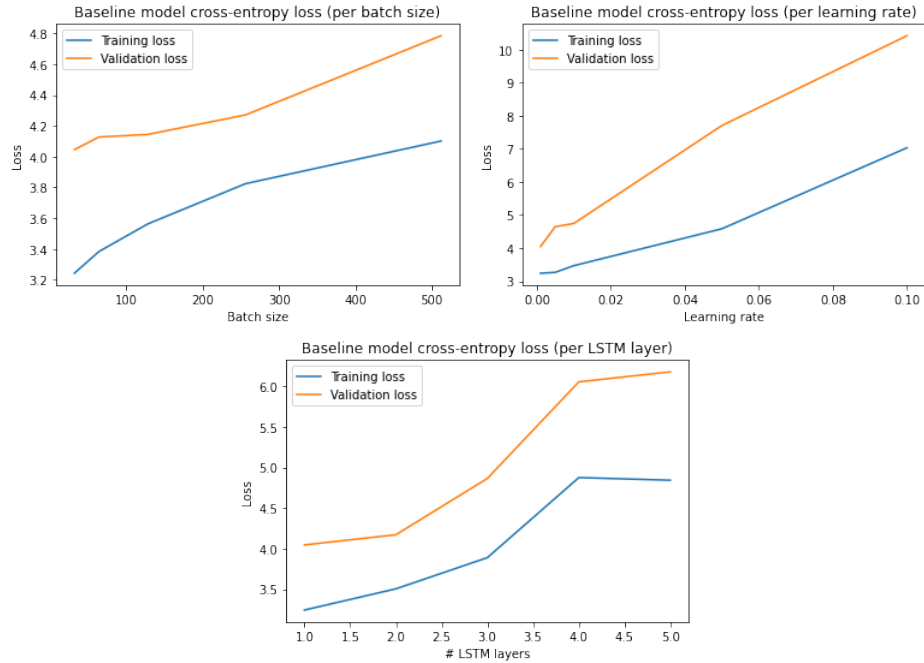


Figure 4: Cross-entropy loss for Baseline CNN-RNN model hyperparameter tuning for batch size, learning rate, and #LSTM layers. Best values are: batch size = 32, learning rate = 0.001, #LSTM layers = 1

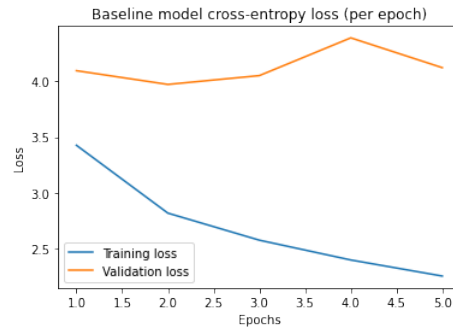


Figure 5: Baseline CNN-RNN model training over 5 epochs with best hyperparameter values

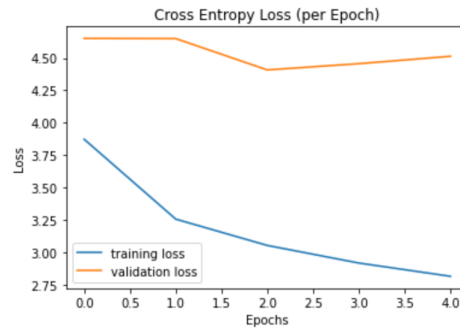


Figure 6: Attention-based CLIP+Attention RNN model training over 5 epochs

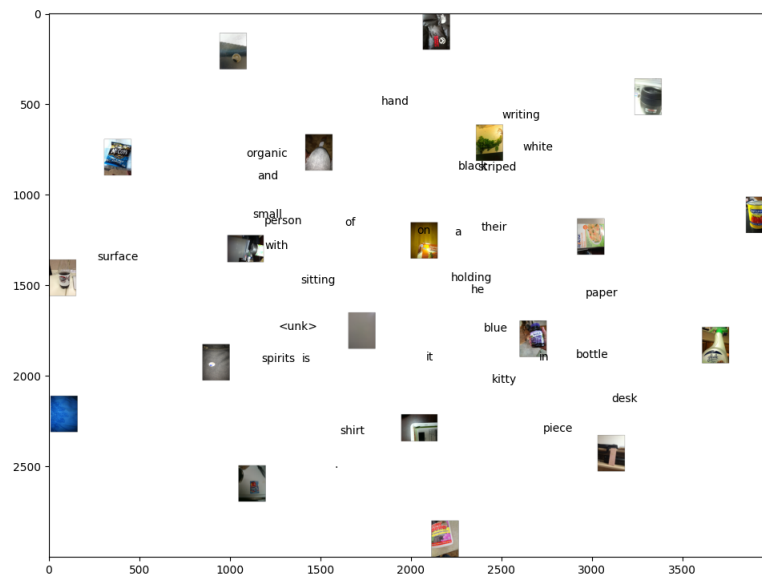



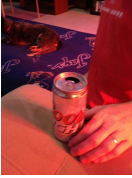



Figure 7: CLIP+LSTM TSNE plot of image and caption embeddings

Table 4: Captioned Test Images. <unk> is a token for unknown words

Images	CNN+LSTM	CLIP+LSTM
	"A computer screen with a blue background and a blue screen"	"A white equipment with a aluminum display and a aluminum clock ."
	"A computer screen with a blue background and a blue screen"	"A computer keyboard with a white background and black lettering ."
	"A bottle of <unk> <unk> wine from <unk> <unk>"	"A white container of somebody is on top of a table ."
	"A person is holding a bottle of deodorant ."	"A person holding a can of soda in front of a white wall ."
	"A box of <unk> <unk> <unk> <unk> <unk>"	"A servings with a picture of a stating on it ."