Final Report: Image Captioning of Earth Observation Imagery

Dora Qian, Fanli Zhou, James Huang and Mike Chen

MDS Mentor: Varada Kolhatkar

MDA Partners: Andrew Westwell-Roper, Shun Chi

Executive Summary

In our MDS-MDA joint Capstone project, we aim to caption overhead Earth observation images captured by satellites using deep neural networks. We've developed a pipeline that processes raw images paired with captions and trains a model using this data. We've also built an interactive visualization tool capable of uploading and captioning new images using our model, adding/modifying human captions to the image-caption database, as well as inspecting the quality of model-generated captions. In this pipeline, a CNN model utilizing transfer learning was used to encode images, paired with an LSTM model to decode descriptive sentences. Model performance was evaluated using a combination of different metrics that included semantic similarity metrics and n-gram based similarity metrics. Our current model was trained on UCM_captions and RSICD. The overall performance of the model is fair on the testing portion of UCM_captions and RSICD datasets. We also examined the generalizability of the model using the Sydney dataset that was unseen during training.

Introduction

MDA is a Canadian aerospace company, specializing in manufacturing equipment for space applications, in particular space surveillance, space robotics, and satellite systems. MDA has access to a vast database of images captured by satellites. Our ultimate aim of this project is to extract quality, descriptive captions from these images. Extracting a caption from an image makes it much more accessible. These captions can be used to tag and sort images based on their content, return a search query, and evaluate similarity between images, for example. Captioning images is not a binary problem; with a wide range of quality for captions, we expect this to be a complex problem with a complex solution. Our goals in this project are to develop a pipeline that processes raw images paired with captions and trains a model using this data – an end-to-end image captioning system – as well as provide a visualization tool capable of uploading and captioning new images, as well as display results from previously trained images.

Given that MDA's images are uncaptioned, we have used several public datasets containing captioned images of overhead satellite images to train our model. This data is processed first by standardizing all the images to be of the same resolution and format, and then pairing them with captions extracted from the corresponding .json files for each dataset. Our task can be broken down into several steps. First is creating the data storage structure: we need to transform the data into a format that we can work with, that is also easily reproducible, for when our pipeline is going to be used with a different dataset; straightforward file organization and a simple, well-defined JSON structure to store the captions in. The next step is developing the model itself, which can be broken down into at least two sub-steps: extracting machine-interpretable information from an image, and generating a sentence from this information. Candidate models would be evaluated with algorithms that measure the similarity of sentences; semantic similarity based and n-gram based algorithms are viable options. The visualization tool should be able to generate captions for any user image, upload the results to the database, view previously generated captions with evaluation scores if applicable, and overall should be designed as a user-friendly way to interact with the model.

? data description?

Data Science Methods

Model

Image captioning has been a popular problem in recent years and several state-of-the-art deep learning models have been proposed in the literature for this problem. We focused on the encoder-decoder model as it is the most common method for image captioning. Here are the three model architectures we tried:

1. Our baseline architecture combines CNN and LSTM. At each step during generation, we combine the LSTM output with the image feature vector and pass the result through a dense layer and an output layer to generate the next word, which is fed back as input to the LSTM layer in the next step.

This model architecture is relatively simple and easy to optimize. But the image features used in this model is a high-level image summary and may not carry enough information for a good caption. Based on literature, adding attention layers can improve image captioning. So, we tried two model architectures with attention layers.

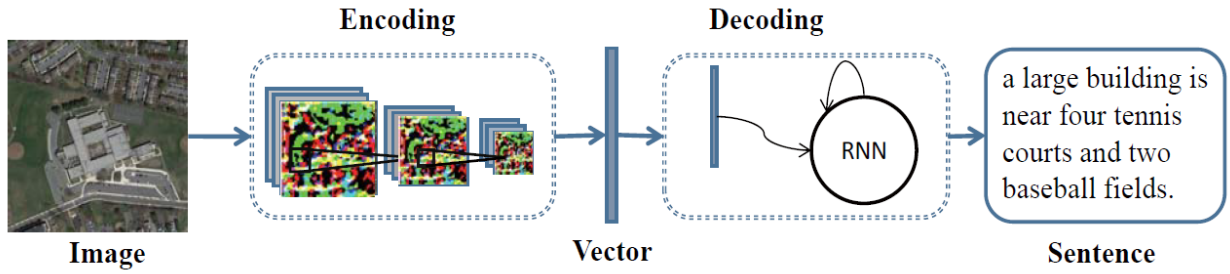

Figure 3. The baseline model architecture (adapted from (Lu et al. 2018)).

2. Our second model architecture has an attention layer on top of the baseline model (Figure 4). Attention is an interface between the CNN and LSTM that provides the LSTM with weighted image features from the CNN convolutional layer. Overall, the model can selectively focus on useful parts of the input image and align image features with words (Xu et al. 2015; Zhang 2019).

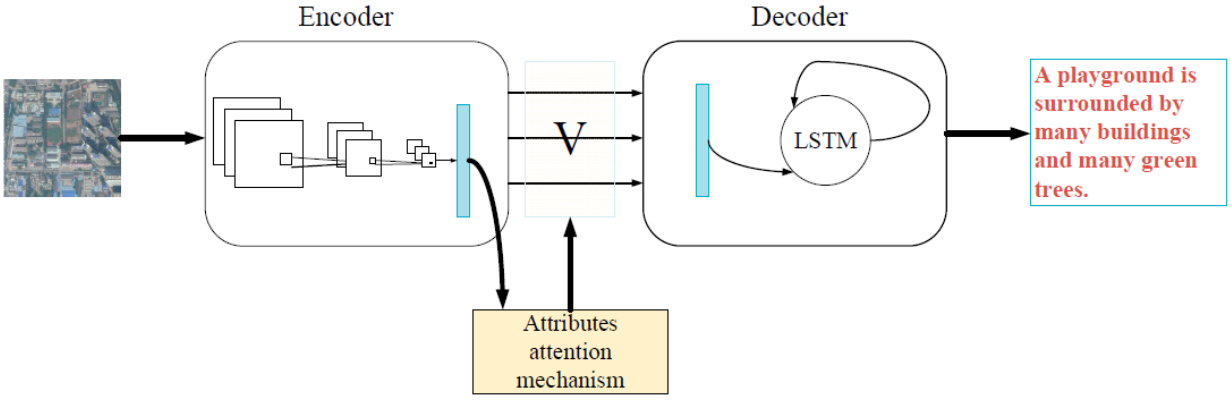

Figure 4. The second model architecture (adapted from (Zhang 2019)).

3. As an extension of the second model, the third model architecture contains three attention structures on top of the baseline model (Figure 5). This multi-level attention model better mimics human attention mechanisms and act as moving the focus between the image and the word context to help generate better captions (Li 2020).

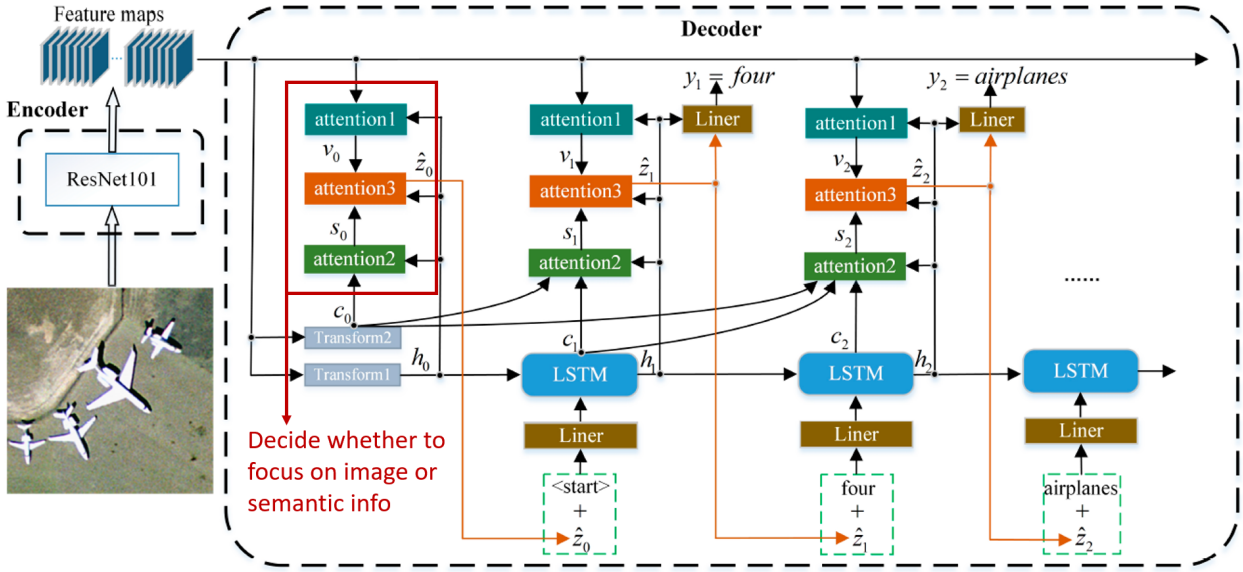

Figure 5. The third model architecture (adapted from (Li 2020)).

Transfer Learning

For each model structure, we use heavy transfer learning. Given an image, we extracted a feature vector from the pre-trained `InceptionV3` or `vgg16` model, a CNN trained on `ImageNet`. For LSTM, we used an embedding layer and initialized embedding weights with pre-trained `GloVe` (`glove.6B.200d`) or `Wikipedia2Vec` (`enwiki_20180420_500d`) embeddings. Pre-trained models or embedding weights were trained on a large dataset and achieved good performance. Incorporating pre-trained models or embedding weights is simple and can reduce training time. The caveat is that the performance depends on task similarity.

Evaluation Metrics

Evaluating the quality of machine-generated captions is challenging as there exist many possible ways to describe an image and there is no one correct way.

To examine our model performance, we have used 2 types of evaluation metrics: N-gram based and semantic-based metrics. In total, 9 different evaluation metrics are used in the evaluation stage.

N-gram based metrics includes Bleu 1-4 [REF], Rouge_L [REF], Meteor [REF] and CIDEr [REF]. These metrics are commonly used in the natural language processing community and related research papers. Bleu score counts the occurrence of n-grams of generated captions in the reference captions and is precision-based. Similar to the Bleu score, Rouge_L is recall-based and calculated as an F-measure using the longest common subsequences. Meteor is generated by using alignments between reference and generated captions. CIDEr is the newest one which is proven to have a more human consensus as it incorporates TF-IDF weights in the calculation. We used the defence version of CIDEr in our evaluation script. The main problem with the n-gram based metrics is that they are sensitive to word overlapping. However, for two captions to have the same meaning, word overlapping is not necessary. Moreover, MDA is more interested in the semantic meaning of the caption, therefore we have includes 2 more metrics.

Semantic-based metrics include Universal Sentence Encoder similarity (i.e. USC_Simialrity)[REF] and SPICE[REF]. USC_Similarty first encodes any caption into a matrix using their pre-trained multi-language model and then computes the inner product of any two captions. While SPICE parses a caption into a semantic scene graph that lists all the objects, attributes and relations in the sentences. By using the dependency graph, the captions with similar semantic meanings will have a much more reasonable score compared with using n-gram based metrics.

By incorporating both n-gram and semantic-based metrics, we have a more comprehensive view of model performances.

Results

To assess those models, we used `USC Similarity` and `SPICE` to evaluate semantic similarity and some other metrics to evaluate syntactic similarity based on `n-gram` comparison, including `BLEU 1`, `BLEU 2`, `BLEU 3`, `BLEU 4`, `Meteor`, `ROUGE_L`, and `CIDEr` (Li 2020). All scores range from zero to one, except the `CIDEr` score, which ranges from zero to ten.

As shown in Table 1, when testing on a dataset like the training data, the baseline model achieves better scores than other models. Those scores are comparable to scores in literature (Li 2020). But models with attention layers did not improve the performance. It could be that we didn't spend enough time fine tuning those models. But MDA is more interested in building a working end-to-end pipeline than getting the state-of-the-art results. So instead of further optimizing the models, we decided to spend more time on the pipeline and just used this baseline model in our final data product.

Table 1: Table 1. Evaluation scores from the best model of each structure on the test dataset (Combined RSICD and UCM datasets).

	BLEU 1	BLEU 2	BLEU 3	BLEU 4	Meteor	ROUGE L	CIDEr	SPICE	USC Similarity
Baseline	0.648	0.523	0.440	0.381	0.300	0.553	2.125	0.400	0.612
Attention	0.572	0.435	0.351	0.294	0.256	0.473	1.540	0.324	0.550
Multi-Attention	0.593	0.463	0.380	0.321	0.271	0.498	1.738	0.345	0.583

To test the model generalization capability, we tested our models on the `Sydney` dataset that is different from the training data. As shown in Table 2, the baseline model has the best scores, but all scores are lower than scores when testing on a similar dataset in Table 1. It indicates that the models have poor generalization capabilities.

Table 2: Table 2. Evaluation scores from the best model of each structure on the Sydney dataset.

	BLEU 1	BLEU 2	BLEU 3	BLEU 4	Meteor	ROUGE L	CIDEr	SPICE	USC Similarity
Baseline	0.453	0.220	0.117	0.072	0.145	0.29	0.210	0.119	0.458
Attention	0.431	0.209	0.108	0.069	0.140	0.28	0.146	0.114	0.449
Multi-Attention	0.431	0.194	0.078	0.034	0.133	0.27	0.144	0.097	0.450

Other Considerations

Besides, we tried to train our own CNN classifiers based on labeled satellite images but the performance could not beat the pre-trained CNN models. We also tried to train embeddings using our training captions and then tested the embeddings by predicting cosine similarity between words. Again, we found that pre-

trained embeddings perform better than embeddings learned from scratch. So we decided to use pre-trained CNN models and embeddings.

Future Improvements

Models with attention layers have great potential. We could further improve the performance of those models. If we have time, we could try optimizing hyperparameters, fine tuning the pre-trained CNN, extracting features from different convolutional layers, and improving attention structures.

Data Product

The final data product is a complete image captioning pipeline , consisting of 3 independent modules: a database, a deep learning model and a visualization tool. When designing our product pipeline, we have separated the visualization tool from the other two because it can be run without GPU. The flowchart describing the whole workflow can be found here. For the main pipeline, we use Make file to create the whole workflow. The process starts with loading raw data and preprocessing them, to model generating and evaluating. All the steps can be executed by using `make all` command in the terminal. We also allow users to call any specific part of the workflow. For example, `make data` to prepare the data for training and testing. The visualization tool workflow is implemented using Django. it can interact with our database and model in 3 different ways.

Database

The first module is a database. AWS S3 bucket is chosen to be used as our database mainly because it can integrate well with AWS GPU instance that we used for training the model. Other advantage includes great scalability and ease of use.

In order to use this database, we will provide a private link on google drive for users to download raw data and upload them to their own S3 bucket as the starting files. After running the whole pipeline, the users should have the database structure as shown below. They will have 8 folders containing raw, preprocessed image and JSON files, as well as model results and scores.

Deep learning model

The second module is a deep learning model. The final model we used in data product is the baseline model with VGG 16 as pre-trained CNN and Glove embedding as the pre-trained word embedding. The model is written in Pytorch and AWS GPU instance is required to train the model. In this module, We allow users to train the model, generate caption and evaluate the results. After running the pipeline, all the trained model, model results and scores will be saved back to the S3 database.

Visualization tool

For the visualization tool, our client had a few required functionalities. Such as the ability to upload their own image, and have a machine learning model produce captions for the uploaded image. Another functionality is the ability to view the training captions, the generated captions and the evaluation scores for those captions on different images in the test set. The visualization tool should also be able to allow the user to upload multiple images and their JSON caption file to a bucket in S3. Overall it should be a user-friendly way to interact with the model.

The frontend of the visualization tool is made using HTML, CSS and Javascript. The backend is made using Django, which is a python based framework for web development. And we're using AWS S3 as our database to save all the images and caption files.

We have three tabs for the three required features. ("User Image" tab for generating captions, "Demo Example" to view results, and "Database Upload" for multiple images uploads)The user would click on the relevant tab for the task they want to do. The pros of making a web app instead of a dash app or shiny app is the fact that the technology used to make web apps are more ubiquitous and common in the industry. Which means if it's easy for other developers to add upon the existing app. In addition, web apps utilizing native tech such as HTML, CSS and etc are faster compared to using a Dash framework.

There are numerous improvements that can be implemented on the visualization tool. Security features should be implemented to ensure that the user uploads the correct files, or file types. For example, uploading multiple images a good security check would be to check if the image content is legitimate. For instance, the user didn't accidentally upload an image of themselves with their satellite images.

Conclusion and recommendations

We have delivered a product complete with all the features that were proposed. The performance on the test portion is fair. However, the generalizability of the model may be poor as shown by the performance on the unseen dataset. We hope that MDA is able to iterate and improve upon our work.

Due to the short time of this project we have had a few limitations. The datasets we worked with are fairly small compared to other available datasets, which inhibited our ability to train a CNN model from scratch. When this was attempted, the model trained from scratch performed poorly compared to utilizing transfer learning from a model trained on ImageNet type images. Our attention layer didn't perform as expected. In the later stages of the project we had to prioritize the pipeline structure and making sure we had a well documented and iterable product over refining the model.

Our project has much room to be improved and expanded upon. Utilizing larger datasets, a CNN model tailored to satellite images can be trained that potentially outperforms transfer learning. We believe an attention model has great potential and could be further experimented on. The model can also be refined by adding or tuning model layers. Cross dataset performance should be more rigorously tested, since we ultimately are interested in performance on the MDA dataset, while the model will have to be trained on other datasets.

References

Li, S.; Jiao, Y.; Fang. 2020. "A Multi-Level Attention Model for Remote Sensing Image Captions." *Remote Sens.* 12 (6): 939. https://doi.org/10.3390/rs12060939.

Lu, Xiaoqiang, Binqiang Wang, Xiangtao Zheng, and Xuelong Li. 2018. "Exploring Models and Data for Remote Sensing Image Caption Generation." *IEEE Transactions on Geoscience and Remote Sensing* 56 (4): 2183–95. https://doi.org/10.1109/tgrs.2017.2776321.

Xu, Kelvin, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention." http://arxiv.org/abs/1502.03044.

Zhang, X.; Tang, X.; Wang. 2019. "Description Generation for Remote Sensing Images Using Attribute Attention Mechanism." *Remote Sens.* 11 (6): 612. https://doi.org/10.3390/rs11060612.