Machine Learning Engineer Nanodegree

Capstone Project

Raiymbek Akshulakov

June 3rd, 2020

I. Definition

Project Overview

A human would probably look at this photo, and instantly come up with tens different thoughts on what is depicted in there. A few examples would include: two friends walk under the trees, two guys walk in the park, a person with a guitar walks

near another person. A quick glance for a second is enough to make such captions for humans. However, what is easy for our brain to do is sometimes very hard to do for machines. Automated text description generation is the task called image captioning. The application of such technology is endless. For example, an image descriptor for the visually impaired because this way the application can not only detect the objects in front of a person but also understand the relations among the objects.

The task of image captioning is the process of ongoing research. The novel approaches to image captioning include various RNN models. The use of those RNN models was inspired by the success in the task such as machine translation which is basically a quite similar task to image captioning. Examples of those models would be a paper *"Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models"* (https://arxiv.org/pdf/1411.2539.pdf) that used encoder-decoder RNN model and also *"Show, Attend and Tell: Neural Image Caption Generation with Visual Attention"* (https://arxiv.org/pdf/1502.03044.pdf) that added an attention mechanism to encoder-decoder RNN model for image captioning. For Udacity Capstone project I will build a simple LSTM seq-to-seq model to be able to generate image captions to a given image

Problem Statement

The goal is to build a model that takes an image as input and generates the text description of it.

Step:

- Find the appropriate dataset containing images and captions to them
- Build a model the model for image captioning
- Train the model in google Colab using free GPU
- Evaluate the model

The final model is expected to be able to take image from internet and generate the captions given the url

Metrics

Just like image captioning task gets a lot of inspirations from machine translation task, the image captioning evaluation is also very close to the evaluation of machine translation

The main metric I will be using is widely used for comparing generated sentences to a reference sentence - Bilingual Evaluation Understudy Score, or BLEU for short. It can be implemented with nltk python library.

The main advantages of BLEU are:

- Computationally efficient.
- A lot like human evaluation.
- Language independent.
- Widely adopted.

II. Analysis

Data Exploration

For the project I used for the Coco dataset which contains a huge amount of annotated images used for object detection, segmentation and image captioning

Downloading the dataset takes several hours and 20 GB. Data links:

- train images - http://msvocds.blob.core.windows.net/coco2014/train2014.zip
- validation images - http://msvocds.blob.core.windows.net/coco2014/val2014.zip
- captions for both train and validation - http://msvocds.blob.core.windows.net/annotations-1-0-3/captions_train-val2014.zip

Each image in the dataset contains 5 captions. It contains a training set for training and a validation set for evaluations.

Training and Validation annotations are stored in two big json files. The file can be seen as a huge python dictionary - let' say **data.** Then **data['images']** is a list of all image information such as the **name, id, height, width, and the link for image** and **data['annotations']** is a list of all annotations. Each annotation element has the id of the image and a caption.

Those are the images of different sizes but I resized all of them to (299,299) so that I will be able to put those images to the InceptionV3 model. Since the amount of data was so huge and my google drive was not able to handle so much data at the same time, I divided the whole dataset into smaller parts of 10000 images each.

The **dataset folder** contained those parts of images stored in h5 format, and alo names of the images stored in csv format. The file Embedding_retrieval.ipynb was used to turn all the images into image feature vectors from the InceptionV3 model. Those feature vectors weigh much less than the original images so those feature vectors are stored in **embeddings_dataset** folder and previous image files are deleted from the google drive.

In terms of the dataset split I left training set as it is, but I also split validation set into test and validations sets:

- Train set size - `82783 images`
- Validation set size - `35504 images`
- Test set size - `5000 images`

Exploratory Visualization

The COCO dataset consist of images like this:

Captions to the image above:
'This wire metal rack holds several pairs of shoes and sandals',
'A dog sleeping on a show rack in the shoes.',
'Various slides and other footwear rest in a metal basket outdoors.',
'A small dog is curled up on top of the shoes',
'a shoe rack with some shoes and a dog sleeping on them

Captions to the image above:

'a dog walking down the middle of a street next to a store lined sidewalk',

'cows people and vehicles on a city street',

'a cow walking on a city street with trees in the foreground',

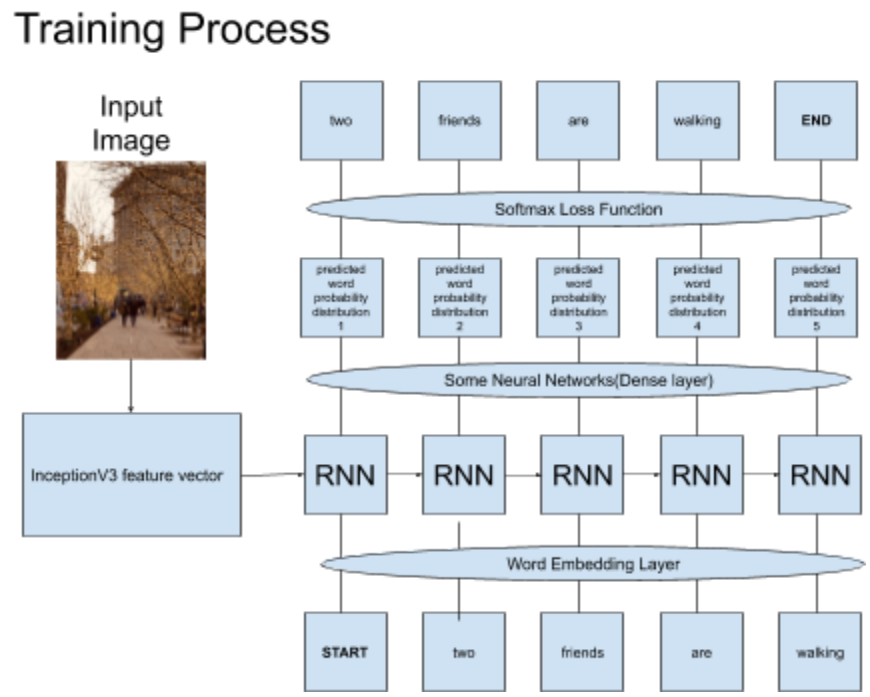
'several cows wander down a city street with buildings and cars in the background',

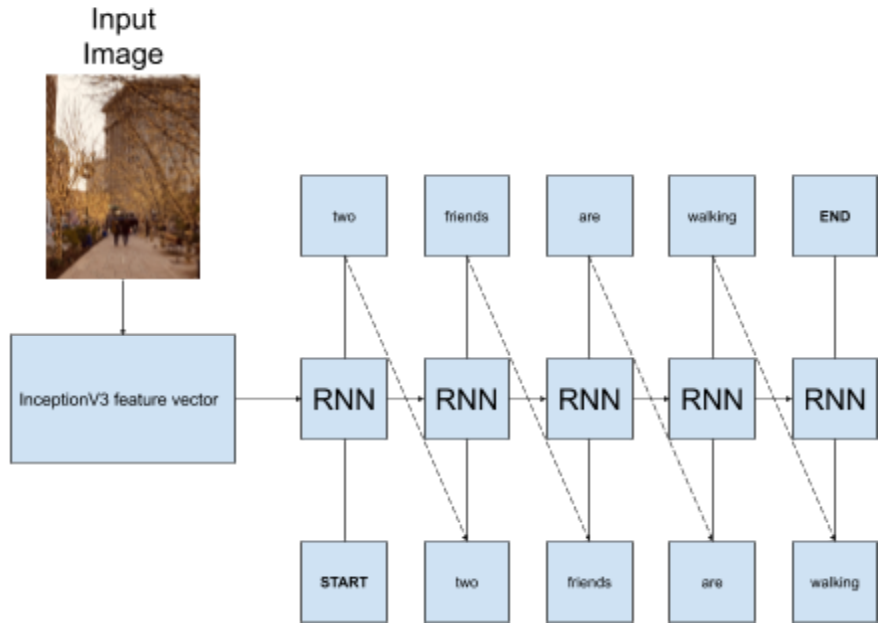
'a small herd of cows walking down a street'

Algorithms and Techniques

I will be using the LSTM seq-to-seq model. The model can be divided in two stages - encoder and decoder. Encoder takes the data that we want to generate a caption from,(in our case an image) and gets a feature vector from it. For image data it is the best to use a CNN for feature extraction. In fact, instead of training my own CNN I will use an already pre trained and established ***InceptionV3*** model. I will take the last layer before the last Dense layer in the model. Next, decoder is an RNN model that takes the encoder's feature vector as the initial hidden state. The decoder model tries to predict the next word given the current word. We start with a START token and recursively repeat the process until we get an END token or until we reach some length. Here the captions serve both as input and ground truth of the model - each word tries to predict

the next one. During generation model, the model will recursively plug the predicted word back into the model.





Benchmark

The following table was taken from research paper *"Show, Attend and Tell: Neural Image Caption Generation with Visual Attention"* (https://arxiv.org/pdf/1502.03044.pdf). It compared various research papers.

		BLEU				
Dataset	Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR
Flickr8k	Google NIC(Vinyals et al., 2014)$^{\dagger\Sigma}$	63	41	27	—	—
	Log Bilinear (Kiros et al., 2014a)$^{\circ}$	65.6	42.4	27.7	17.7	17.31
	Soft-Attention	**67**	44.8	29.9	19.5	18.93
	Hard-Attention	**67**	**45.7**	**31.4**	**21.3**	**20.30**
Flickr30k	Google NIC$^{\dagger\circ\Sigma}$	66.3	42.3	27.7	18.3	—
	Log Bilinear	60.0	38	25.4	17.1	16.88
	Soft-Attention	66.7	43.4	28.8	19.1	**18.49**
	Hard-Attention	**66.9**	**43.9**	**29.6**	**19.9**	18.46
COCO	CMU/MS Research (Chen & Zitnick, 2014)^{a}	—	—	—	—	20.41
	MS Research (Fang et al., 2014)$^{\dagger a}$	—	—	—	—	20.71
	BRNN (Karpathy & Li, 2014)$^{\circ}$	64.2	45.1	30.4	20.3	—
	Google NIC$^{\dagger\circ\Sigma}$	66.6	46.1	32.9	24.6	—
	Log Bilinear$^{\circ}$	70.8	48.9	34.4	24.3	20.03
	Soft-Attention	70.7	49.2	34.4	24.3	**23.90**
	Hard-Attention	**71.8**	**50.4**	**35.7**	**25.0**	23.04

Those are one of the best research papers at that time(April 2016), and I am probably not going to get anywhere close, but I expect to have BLEU-1 score around 0.6, BLEU-2 score around 0.4, BLEU-3 score around 0.2, BLEU-4 score around 0.1

III. Methodology

Data Preprocessing

- The very first step is to prepare the training images for training. It will be hard to upload the dataset to google colab every time because it is a very large dataset(20 GB). My plan is to divide the dataset into several smaller parts, convert each part to h5 format and upload to google drive from the local machine. In the Google Drive, using colab get each part and run it through the encoder and store it in the google drive(also in parts). Once it is done the initial data can be deleted, and now the encoder feature vector's weight will be a lot smaller than previous 20GB

- Once we get the encoder results, before training the decoder I will need to preprocess the caption data. In this case there is no need for stemming,

lemmatization, and removing stop words because we need almost all the variations of all words. First step would be to tokenize all the captions. To do that , I will get rid of very rare words and change them with UNK tokens(less than 5 or 10 occurrences). I will put a START token at the beginning of each captions and END token and the end. For batching purposes I will also need to use padding for all the captions to one length with PAD token.

Implementation

- It is an ImageCaptioning class containing all the architecture:
    - It will be able to train the model using the *train_model* method.
    - Save and restore the model once trained
    - Generate a caption for given image
    - Evaluate the model on the test set
- The model consists of 5 major layers:
    - **word_to_embedding** - embedding layer for words before plugging in the lstm
    - **img_to_prelstm** - one layer before the initializing the lstm layer
    - **prelstm_to_lstm** - layer that initializes the lstm
    - **lstm** - LSTM layer itself
    - **lstm_to_prelogits** - layer before the final logits that takes the hidden states as input
    - **prelogits_to_logits** - layer that outputs the final logits

- Model hyper parameters:
    - **word_embedding_size** - size of **word_to_embedding layer**
    - **prelstm_size** - size of the **prelstm_to_lstm layer**
    - **lstm_size - size of lstm layer**
    - **prelogit_size** - size of **lstm_to_prelogits layer**
- The generation process has two modes:
    - One is where at each time stamp we take the most probable word based on the softmax probability distribution
    - Second is when the model makes a random choice based on that softmax probability distribution.
- The whole architecture is implemented in tensorflow. The github main files are the **Image_Captioning_tf2.ipynb, Results.ipynb, and Embeddings_retrieval.ipynb, imgCap.py**
    - **Embeddings_retrieval.ipynb** - the file to generate encoding's image embeddings because the actual dataset is way too big
    - **Image_Captioning_tf2.ipynb** - the file that trains the model and saves the best model
    - **Results.ipynb** - the file to show the results of the trained model. People can use it to get caption to any image url
    - **imgCAP.py** - the file that just contains the model class to be used in the **Results.ipynb**

Refinement

- Initial model was the standard seq to seq model - without the **prelstm** and **prelogit** layers(look the implementation section)
    - Best model train loss value - `2.3630406856536865`
    - Best model validation loss value - `2.5217838287353516`

- Then I added the other two layers for more depth and played with hyper parameters. Hyper parameters were - (lstm_size = 200, word_embedding_size = 120, prelogit_size = 300,prelstm_size = 100)
    - Best model train loss value - `2.2627155780792236`
    - Best model validation loss value - `2.461115837097168`

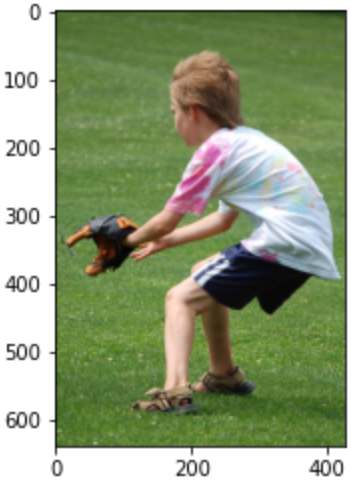
- The final result was the set with hyper parameters - (lstm_size = 400, word_embedding_size = 300, prelogit_size = 500,prelstm_size = 500). I thought it would be actually hard to overfit the problem since it nearly impossible to have train_loss function to be any close to 0, so I tried to increase all the parameters)
    - Best model train loss value - `2.2173445224761963,`
    - Best model validation loss value - `2.4496567249298096`

IV. Results

Model Evaluation and Validation

The final BLEU-1, BLUE-2, BLUE-3, BLUE-4 scores on the test dataset are 0.65, 0.45, 0.28, 0.13). Trying the model on some of the photos of the validation set and test set the results look very promising since a lot of times the model was able to capture what is on the picture, although with some minor mistakes.

For example, in the mode of taking a randomized next word based on the softmax probability distribution, sometimes the model was mistaken in very similar subjects like baseball, soccer and frisbee:

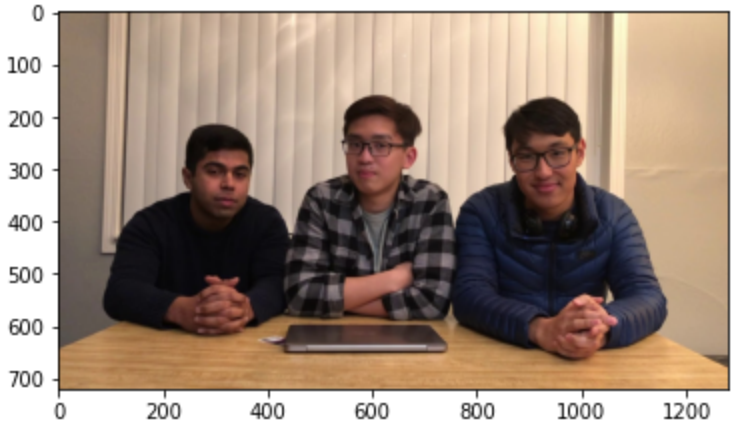


`Predicted: a young boy is playing with a frisbee`

Another example where model is mistaken for similar objects, on the photo with pizza, the model was able to identify pizza but sometimes would give out wrong toppings:

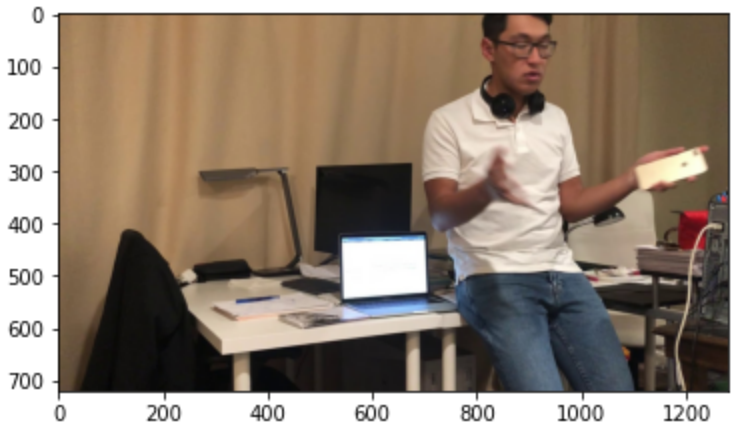

```
Predicted: a pizza with pepperoni and olives on a plate
```

Also when the model was processing the images with people, it had a lot of mistakes with the gender:



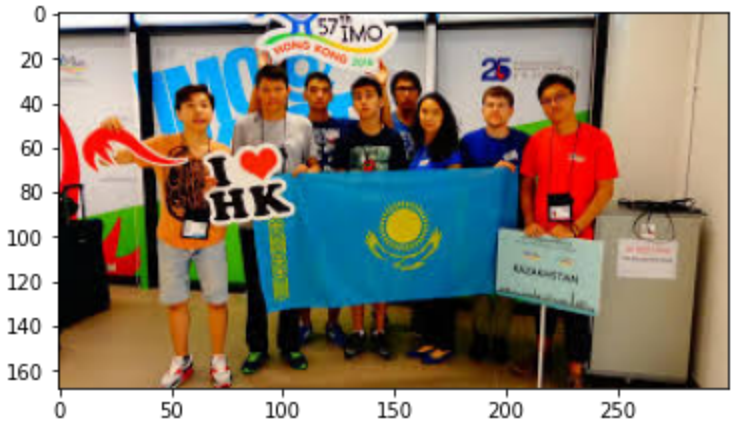

```
Predicted: a man and woman sitting at a table with a laptop
```

Sometimes the model did mistake the relationship in between the objects. When I submitted an image of myself near the laptop, the model outputted a man working on the laptop:

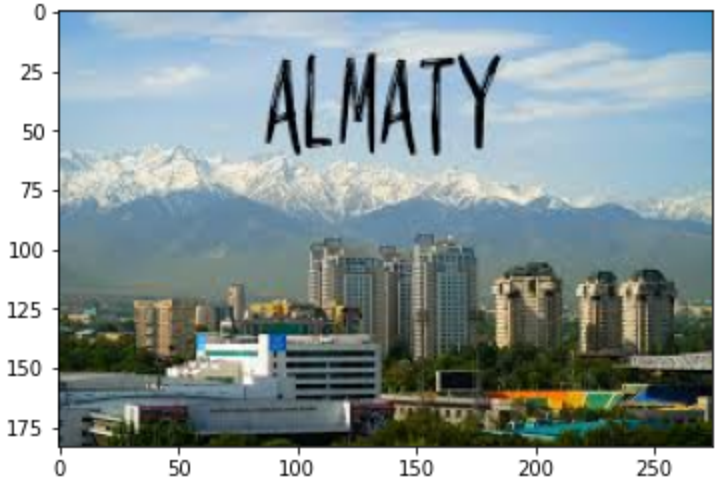



```
Predicted: a man is typing on a laptop computer
```

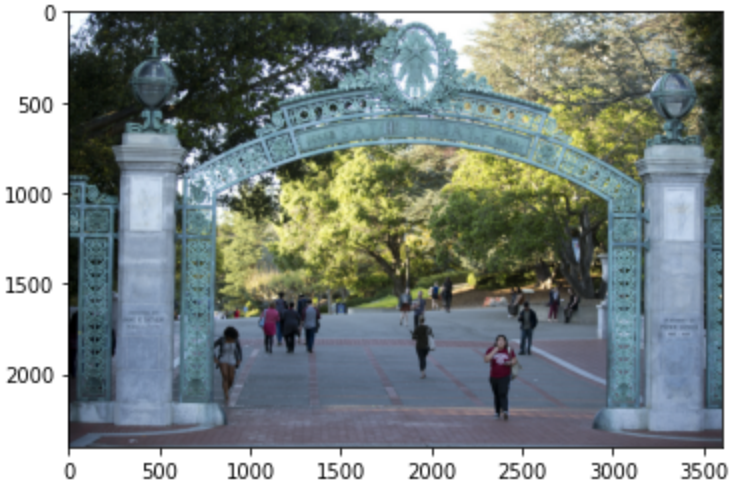
I think the model i still very dependent on the dataset still because sometimes it outputted a lot of completely wrong comments to images if the image was taken from the internet and not from COCO dataset:




```
Predicted: a man and a dog are playing a game of tennis
```




```
Predicted: a large body of water with a clock tower in the background
```




```
Predicted: a horse and carriage on a city street
```

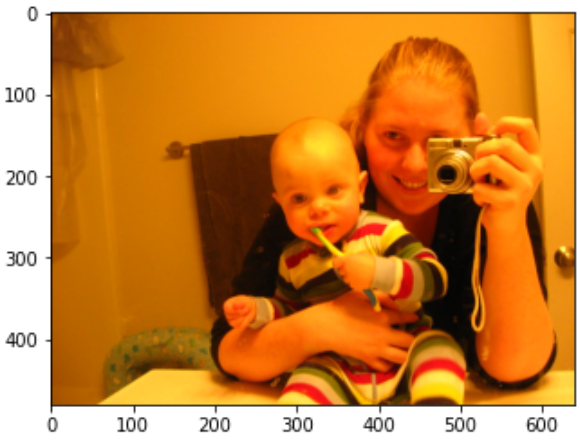
Justification

Comparing the BLEU scores it seems like the BLEU-1 score is 0.66 which is around 0.6 as I described in the benchmark section. In fact it is not even the smallest score in the COCO dataset section of the table. Regarding BLEU-2 score - 0.45, my model was not able to beat any model in the COCO section but it beat most of the models in the flickr dataset. When it comes to BLEU-3 - 0.28 my model is on the level with Google NIC on the flickr dataset. The BLEU-4 - 0.13 score is a lot smaller than any model in the table. Finally, compared to the expectations in the benchmark section, I think the model satisfies those expectations.

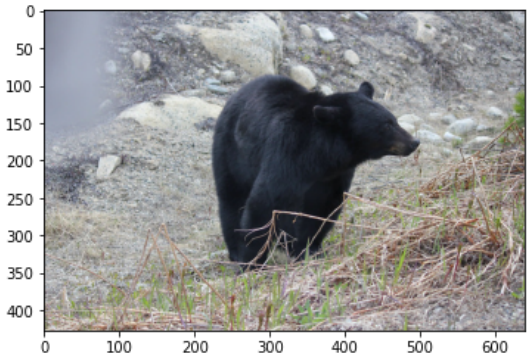
V. Conclusion

Free-Form Visualization

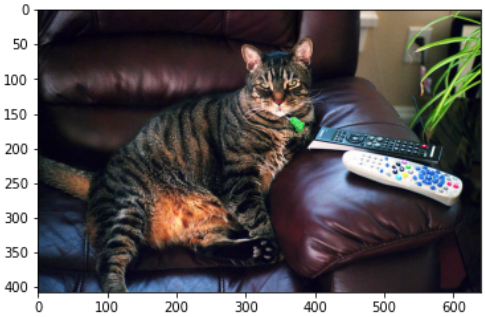
Some of examples of good predictions:



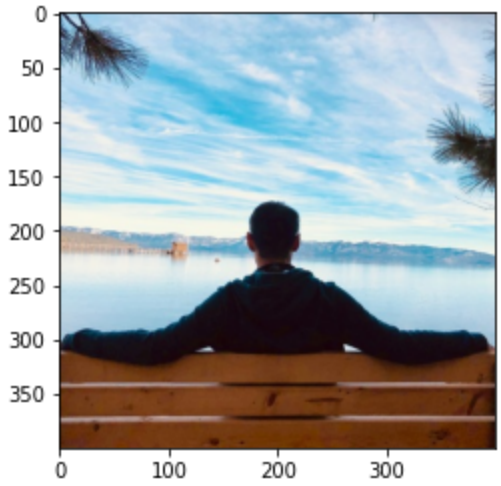
Prediction: a little girl brushing her teeth in a bathroom



Prediction: a black bear walking across a rocky field



Prediction: a cat laying on a couch with a remote control




```
Predicted: a man sitting on a bench in the water
```

Reflection

- The first step was data preparation. It was interesting to operate with such a big dataset, so that I had to divide the whole dataset into several pieces in order to to combine it back later
- Text preprocessing did not involved a lot of thinking or action since I needed most of the words in all possible forms, and the stop words especially
- Decoder building was an interesting part because instead of building a model in keras, I put everything in my own class because this way it'll be easier for me to implement something like attention and will give me more freedom to experiment with various variations models
- I actually was amazed by the final results as it was giving some very accurate captions to the images - even from some of the images I took in internet. It was very interesting to see which results the model gives to various images from internet and the test set

Improvement

- One of the biggest improvements I can make is to add an attention mechanism to the model. This approach showed state of the art results in image captioning as well as machine translation. The benchmark section also shows how attention mechanism can improve BLEU scores of the model

- Another improvement would also be to add a language model to the image captioning to make sure the output caption would give an output that makes sense