DL Group Report

Image Captioning with GUI

Andri Setiawan Susanto 1002849
Bhavy Mital 1002945
Nigel Chan Terng Tseng 1002027
Zwe Wint Naing 1002791

Table of Contents

Background Information

We have decided to work on the first choice out of the two - to implement an image-captioning with a GUI. Image Captioning is the “process of generating a textual description of an image. It uses both Natural Language Processing and Computer Vision to generate [said] captions.”[1] The dataset consists of images that are to be processed, and the output consists of a sentence to describe the image, i.e. a caption.

As described and instructed from the project description, the codes we used were pulled from:
https://github.com/yunjey/pytorch-tutorial/tree/master/tutorials/03-advanced/image_captioning,
The dataset used is MSCOCO 2014/2015. Along the way, we took a look into the structure of the code and understood it as much as we could. We then proceeded to make changes of our own to improve the performance of the base code.

Instructions on Running

Clone git repository

Run:

```
git clone https://github.com/AndriYang/image_captioning_DL.git
```

Followed by:

```
cd image_captioining_DL
```

Install the requirements

Next (and make sure to be in a new virtual env), get the necessary requirements by running:

```
pip install -r requirements.txt
```

Download pretrained GloVe word embeddings

As we are using pre trained GloVe word embeddings, run the following lines to download the GloVe files:

```
chmod +x download_glove6B.sh
```

[1] "Image Captioning in Deep Learning - Towards Data Science." https://towardsdatascience.com/image-captioning-in-deep-learning-9cd23fb4d8d2. Geopend op 26 apr.. 2020.

followed by:

```
./download_glove6B.sh
```

Build GloVe dictionary for training use

For the glove6B files, there are 4 different files of different dimensions - 50d, 100d, 200d, 300d. Running the following line would use the default dimension of 50.

```
python build_glove.py
```

If you want to test out with the other 3 dimensions, you will have to specify what embedding dimensions you prefer by using the "--embed_size" flag when running the 'build_glove.py' script.

Example to run with 100 dimensions:

```
python build_glove --embed_size 100
```

If you change the embedding dimension size here, you will have to include the same flag and dimension size for 'train.py' and 'app.py' scripts when running them.

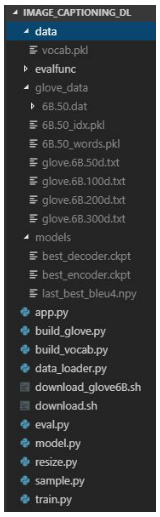
Using Pre-trained Weights

If you wish to see the captions without training the model, please follow the following steps:
Go to the following link:
https://drive.google.com/open?id=1WHztNi5WrrlBLbWzFEJy2rWUaiENAGGW

You'll see 4 different folders. Each has a different set of weights from different runs.
You can find our final set of weights in folder called **GloVe_GRU_10_epochs**

From within that folder, download the folders "models" and "data" as a zip file and unzip them in the project folder root.

Before running the GUI, make sure you have the following files:



```
IMAGE_CAPTIONING_DL
  data
    vocab.pkl
    evalfunc
    glove_data
      6B.50.dat
      6B.50_idx.pkl
      6B.50_words.pkl
      glove.6B.50d.txt
      glove.6B.100d.txt
      glove.6B.200d.txt
      glove.6B.300d.txt
    models
      best_decoder.ckpt
      best_encoder.ckpt
      last_best_bleu4.npy
  app.py
  build_glove.py
  build_vocab.py
  data_loader.py
  download_glove6B.sh
  download.sh
  eval.py
  model.py
  resize.py
  sample.py
  train.py
```

Pre-processing

Run:

```
python build_vocab.py
```

is run to build the vocabulary list. The default threshold is kept at 4 (min word count threshold). This can be changed to other integer values if desired by using the “--threshold” flag.

Only if you are planning to train the model, run:

```
python resize.py
```

to resize the images so that they will be suitable for the model to train on. By default, the images are resized to image_size = 256, that is, 256 x 256.

There is a “--image_size” flag that allows you to specify a different size if you wish to use another size.

Training

To train the code with the default 50 dimensions for GloVe embeddings, run:

```
python train.py
```

As mentioned earlier, if you decided to change the embedding dimensions from the default 50, please include the --embed_size flag

Example to run with 100 dimensions:

```
python train.py --embed_size 100
```

Train has other other arguments that can be changed if needed and the following table are some of the flags you can include.

Red denotes the paths and arguments that do not need a lot of changes.
Purple denotes the model parameters, i.e. ones that make significant changes to the results once modified.

Parameter	Description	Default Value
--model_path	Path for saving trained models.	“models/”
--crop_size	Size for randomly cropping images. Both H x W applies.	224, i.e. 224 x 224
--vocab_path	Path for the vocabulary wrapper.	data/vocab.pkl
--image_dir	Directory for resized images.	data/resized2014
--val_image_dir	Directory for validation resized images.	data/val_resized2014
--caption_path	Path for train annotation json file.	‘/home/jovyan/datasets/coco2014/trainval_coco2014_captions/captions_train2014.json'
--val_caption_path	Path for validation annotation json file.	'/home/jovyan/datasets/coco2014/trainval_coco2014_captions/captions_val2014.json'
--log_step	Step size for printing log info.	10
--reset_training	Whether or not to reset the model with random initialisation.	False, i.e. it will continue the training from last best-saved weights.

--embed_size	Dimension of word embedding vectors.	50
--hidden_size	Dimension of GRU hidden states.	512
--num_layers	Number of layers in GRU.	1
--num_epochs	Number of epochs.	5
--batch_size	Batch size.	128
--num_workers	Number of workers.	2
--encoder_learning_rate	Learning rate for encoders.	0.0001 / 1e-4
--decoder_learning_rate	Learning rate for decoder.	0.0004 / 4e-4

Table 1. List of training flags

Methodology

GloVe pre-trained word embeddings

To improve the existing code, we decided to introduce GloVe pre-trained word embeddings where they would be used with respect to the built vocabulary, to create a weights matrix where each word in the vocabulary had its respective GloVe embedding. We then initialized the decoder embedding layer with this weights matrix for training.

Alternate training of encoder and decoder

Additionally, we decided to train the encoder and decoder a little differently. Instead of training both the encoder and decoder models at the same time, we trained them in an alternate fashion, where each of them has their own optimizer and learning rate attached to them. It works such that only one of these models will be trained during 1 epoch. By the end of the epoch, if the BLEU score sees an improvement, we will continue to train that model. However if there is no visible improvement in the score, we will then switch over to the other model to train in the following epoch.

This method was inspired by this image captioning code.

Different Scores Used in Measuring the Accuracy of the Model

Quick overview: BLEU, or the Bilingual Evaluation Understudy, is a score for comparing a candidate translation of text to one or more reference translations.[2] Instead of evaluating word-by-word of what is written by the model, BLEU would ensure that the caption generator we have will not only generate the correct set of words, but also will generate the correct sequence of words.

The scores range from 0 to 1, when compared to a reference sentence. A perfect match results in a score of 1.0, whereas a perfect mismatch results in a score of 0.0.

The evaluation metrics that we have used in our model are: BLEU 1-4, Cider, and Rogue.

BLEU 1-4 is much more general in nature, used for Machine Translation evaluation. Meanwhile, Cider is used for image description generation evaluation, and Rouge being used for document summarization evaluation.[3]

[2] "A Gentle Introduction to Calculating the BLEU Score for Text in ...." 20 nov.. 2017, https://machinelearningmastery.com/calculate-bleu-score-for-text-python/. Geopend op 26 apr.. 2020.
[3] "Re-evaluating Automatic Metrics for Image Captioning." 22 dec.. 2016, https://arxiv.org/pdf/1612.07600. Geopend op 27 apr.. 2020.

As for the underlying ideas, BLEU uses n-gram precision, Cider uses tf-idf weighted n-gram similarity, and Rouge uses n-gram recall.

BLEU is to be used to measure similarity between two sentences. In other words, for our project, the metrics are going to be used such that the sentences generated will be similar and on par with the reference sentences. However, BLEU has a limitation in the sense that it only looks at the sentences by how similar they are to the

Cider is a relatively recent metric that is designed and specialised for image captioning evaluation. However, it only works linguistically (only via words and sentences), and thus only extends existing metrics with tf-idf weighting over n-grams. This means that details insignificant to the pictures are also weighted more, resulting in an inaccurate picture evaluation.

Lastly, Rouge is used for summary evaluations, which means that the metric relies on recalling sentences, rather than looking at precision. It looks at how many n-grams in the reference translation show up in the output, rather than the reverse. In other words, it measures how much the words (n-grams) in the human reference summaries appeared in the machine-generated summaries.

GUI Display

The Image Captioning GUI is a very simple one - it consists of a few components:

- A large space in order to display the picture.
- A caption line below it, starting with “Caption:”. This is where the generated captions will be shown.
- Predict: To initialise the model to start analysing the photo and to make predictions based on what it was trained on.
- Next Image: To go to another image for prediction.

Steps to use the GUI

1. Run “python app.py”. “Choose directory” screen will appear. Select the folder where the images that you want to predict the caption for are located in (make sure to double click the folder).

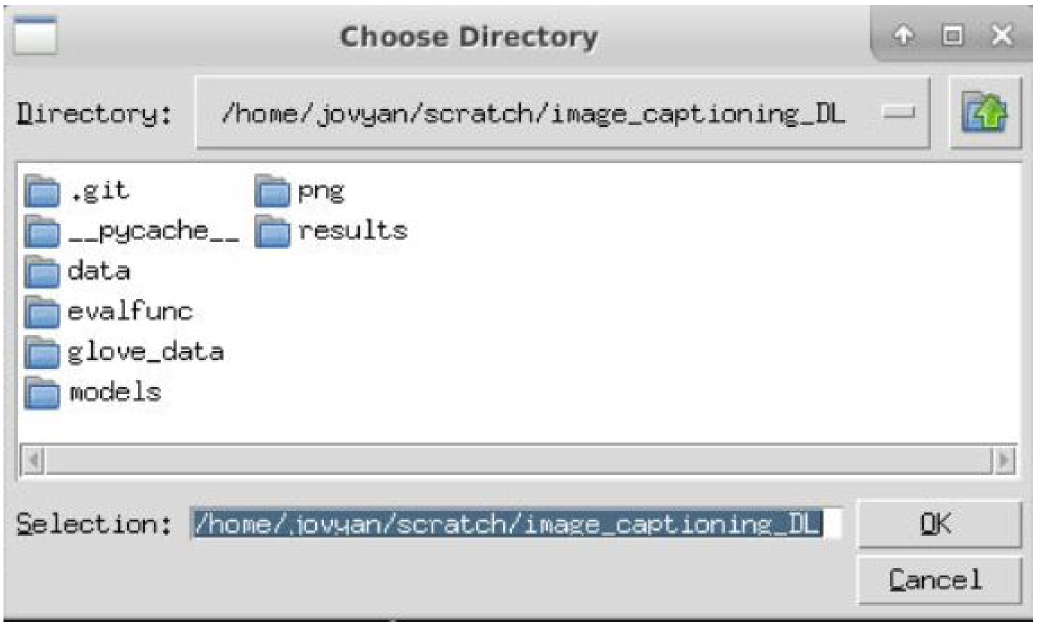

Choose Directory
Directory: /home/jovyan/scratch/image_captioning_DL
.git
png
__pycache__
results
data
evalfunc
glove_data
models
Selection: /home/jovyan/scratch/image_captioning_DL
OK
Cancel

Figure 1. GUI - Choose Folder

2. The GUI will be displayed with an image selected from the folder that you have selected. In order to predict the caption of the image, click the “Predict button”. In order to select another random image in the folder, click “Next Image”.

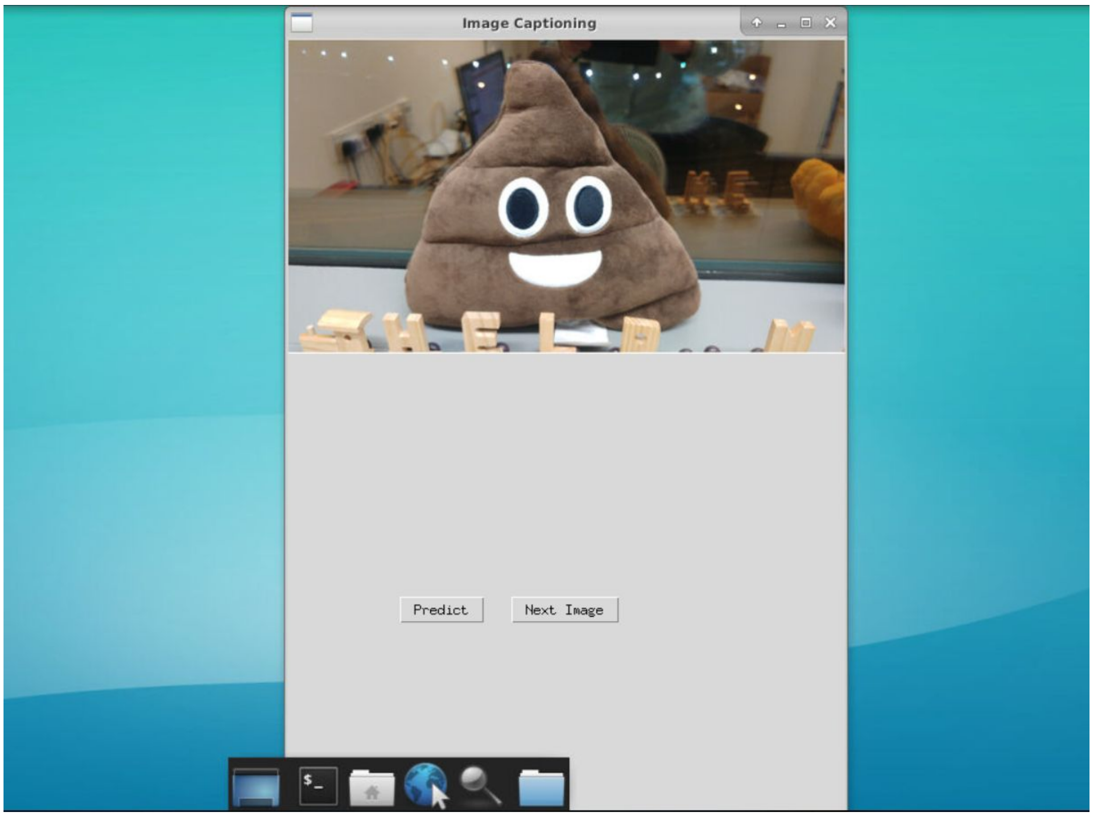


Figure 2. GUI Window

Results

Training with GloVe pre-trained word embeddings

We implemented GloVe pre-trained word embeddings to see if we could get better results as compared to the vanilla base code that we pulled from github. This was done by firstly creating a GloVe dictionary (was done with build_glove.py), followed by the process of comparing each word within the GloVe dictionary and the vocabulary that was generated with build_vocab.py. This resulted in a weight matrix built where the words that existed in both GloVe and vocab were added into it alongside their pre-trained GloVe word embeddings.

This weight matrix was then used to initialize the embedding layer of the decoder for further training.

We first trained, without GloVe, two models where one had GRU and the other had LSTM, for 5 epochs, to see the differences between using each of these layers.

Next, we trained another model for 5 and 10 epochs, where it used GloVe (50 dimensions) and GRU layer.

The following figures are the results obtained from the training for comparison:

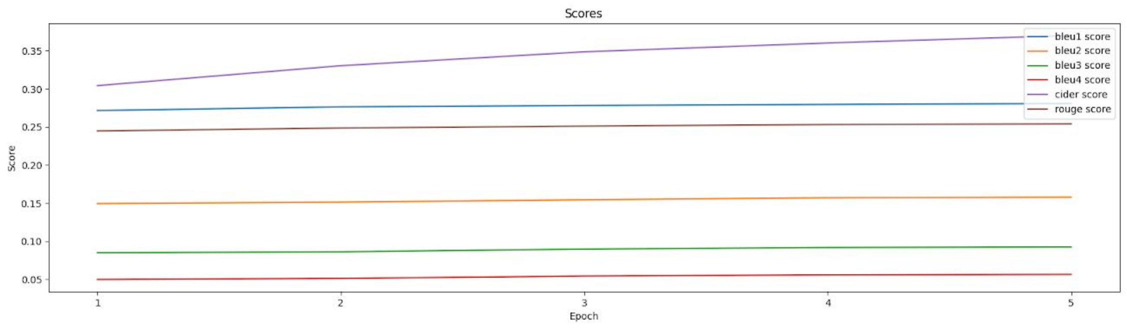


Figure 3: Different scores graph without GloVe, GRU

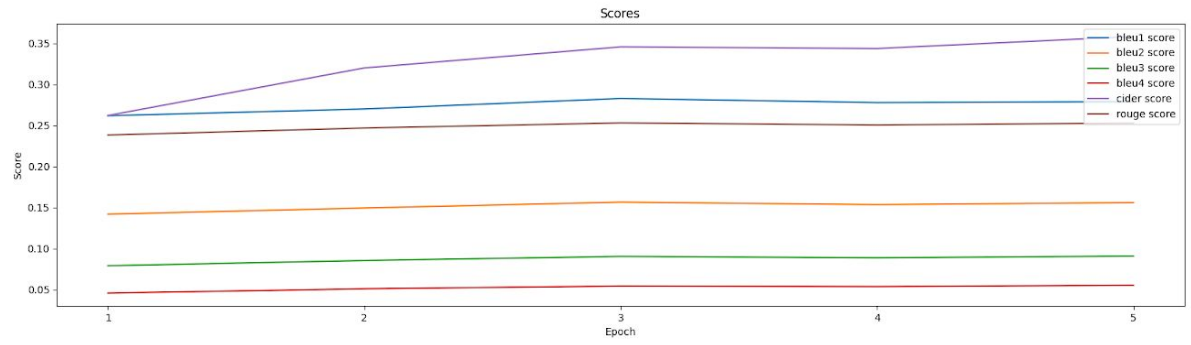


Figure 4: Different scores graph without GloVe, LSTM

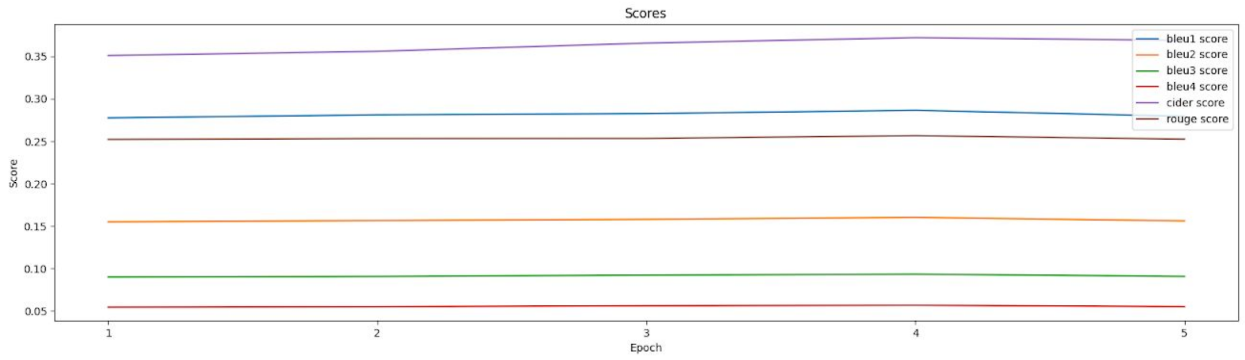


Figure 5: Different scores graph with GloVe, GRU (5 epochs)

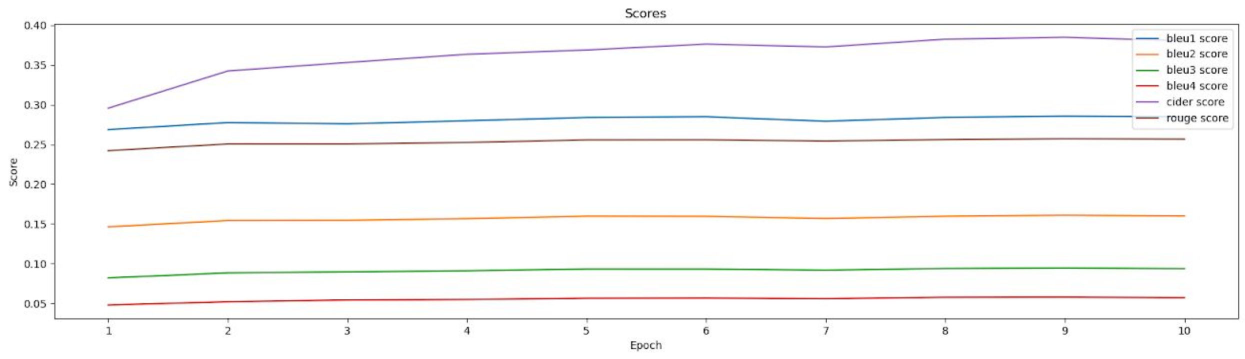


Figure 6: Different scores graph with GloVe, GRU (10 epochs)

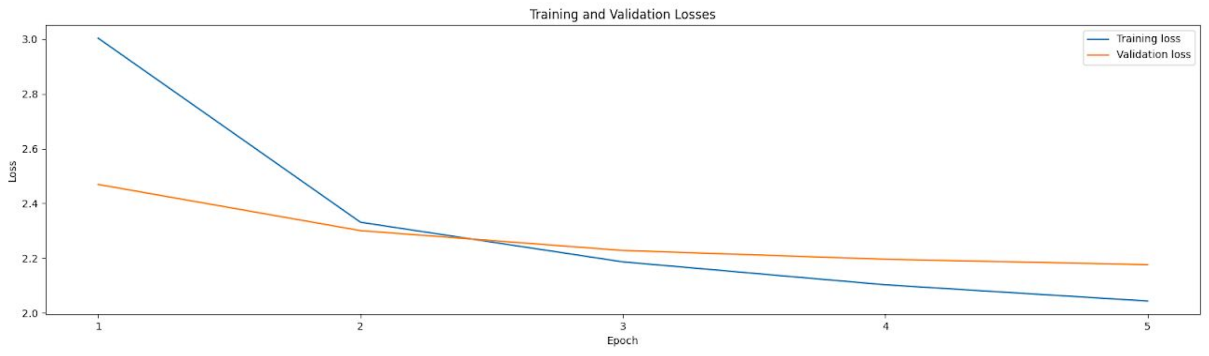


Figure 7: Train/Val losses without GloVe, GRU

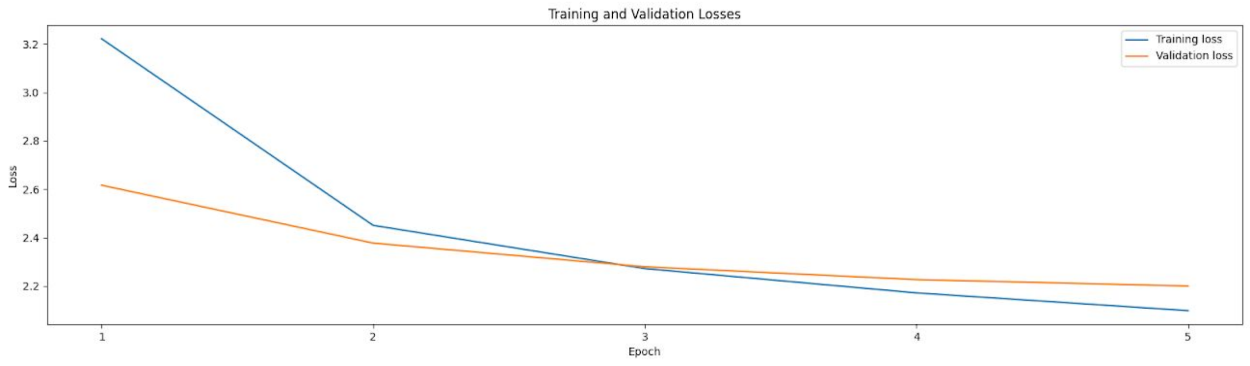


Figure 8: Train/Val losses without GloVe, LSTM

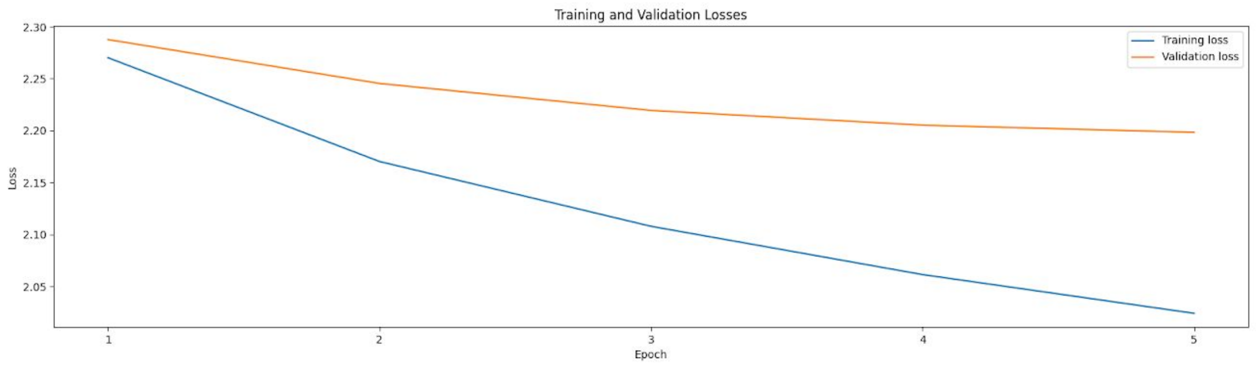


Figure 9: Train/Val losses graph with GloVe, GRU (5 epochs)

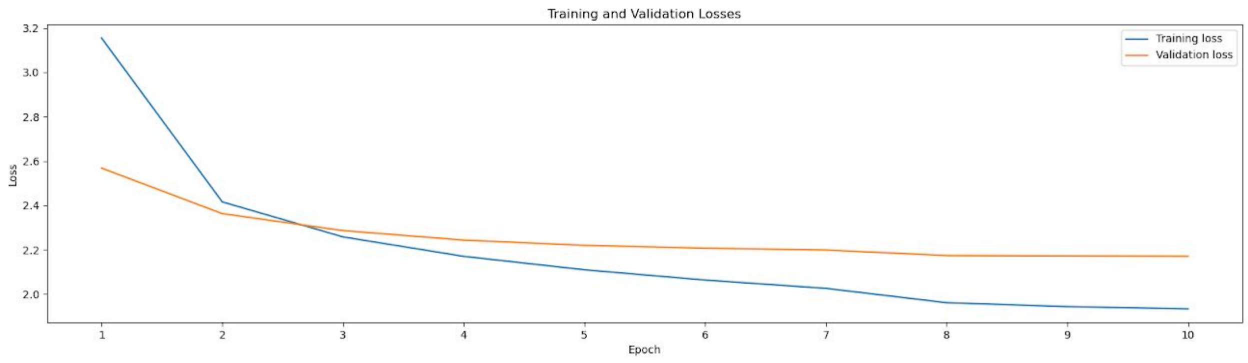
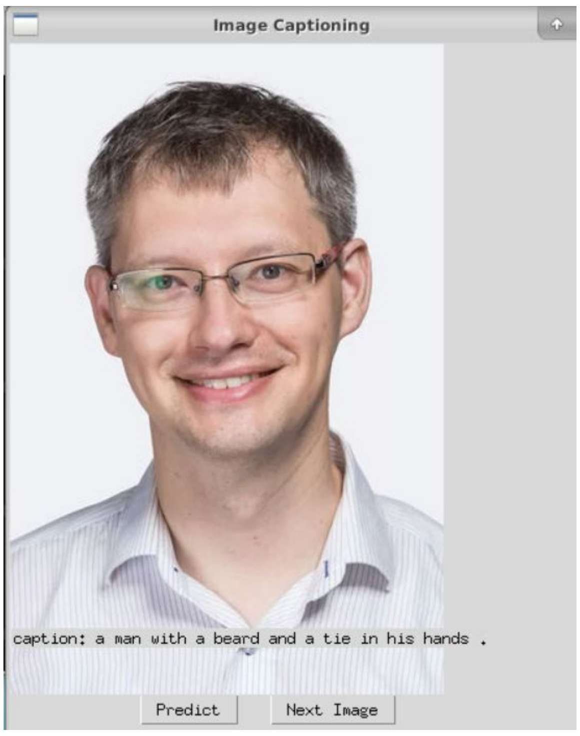
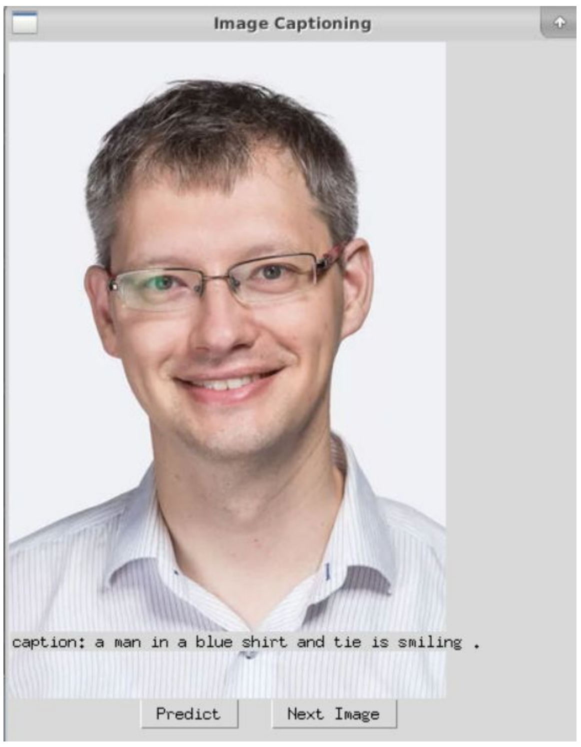
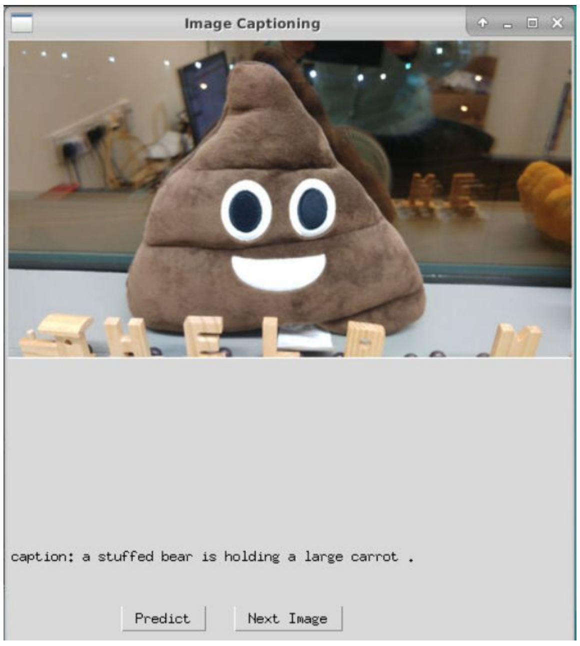


Figure 10: Train/Val losses graph with GloVe, GRU (10 epochs)

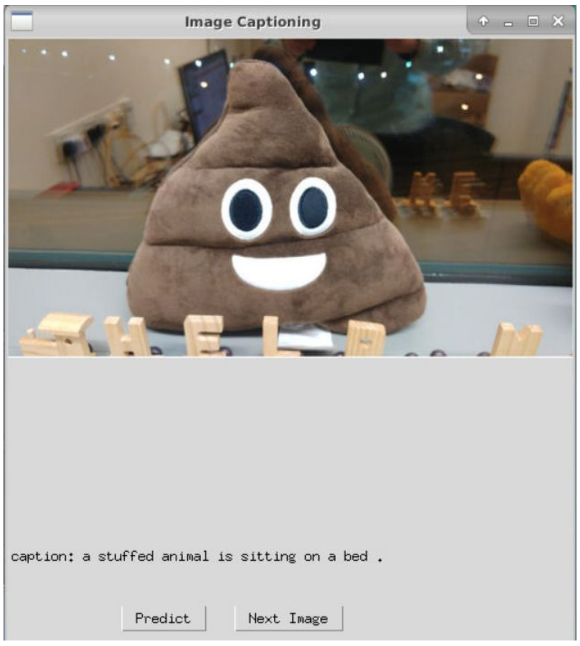
Results obtained of LSTM vs GRU without GloVe (5 epochs each with 3230 iterations)

GRU	LSTM
 caption: a man with a beard and a tie in his hands . Caption: A man with a beard and a tie in his hands.	 caption: a man in a blue shirt and tie is smiling . Caption: A man in a blue shirt and tie is smiling.



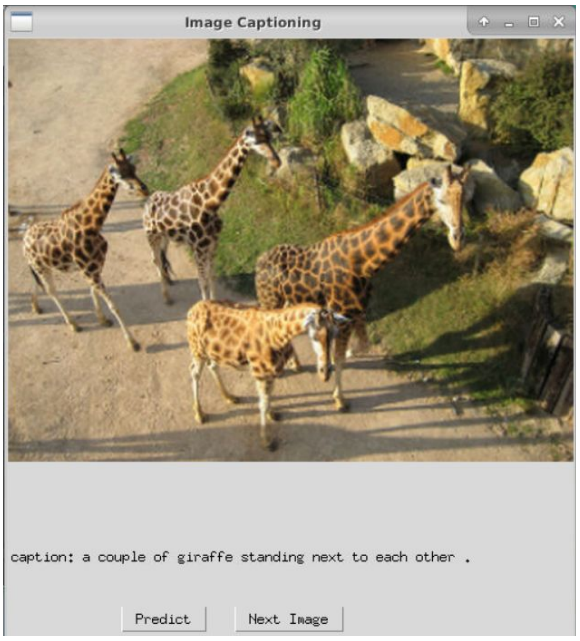
```
caption: a stuffed bear is holding a large carrot .
```

Caption: A stuffed bear is holding a large carrot.



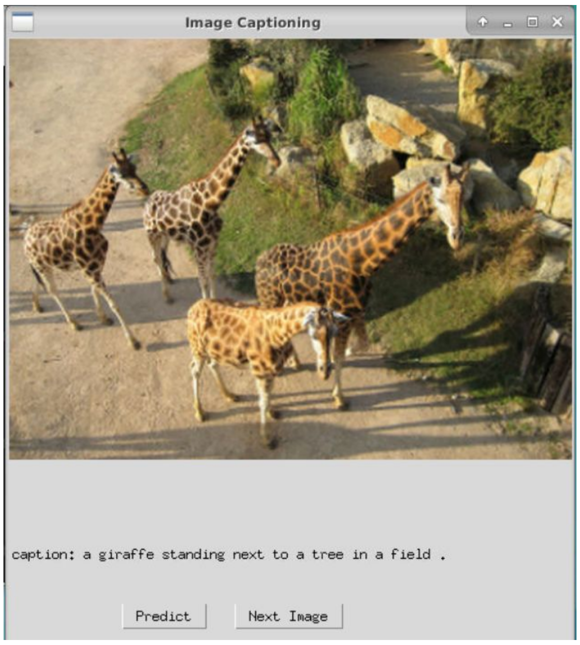
```
caption: a stuffed animal is sitting on a bed .
```

Caption: A stuffed animal is sitting on a bed.



```
caption: a couple of giraffe standing next to each other .
```

Caption: A couple of giraffe standing next to each other.



```
caption: a giraffe standing next to a tree in a field .
```

Caption: A giraffe standing next to a tree in a field.

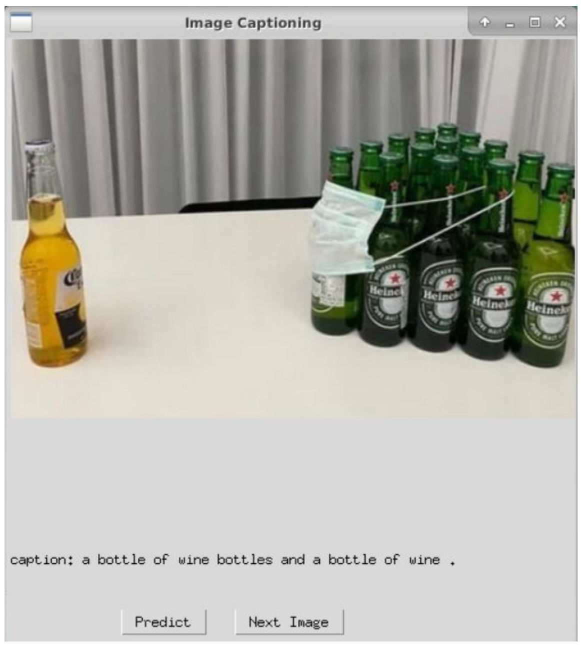
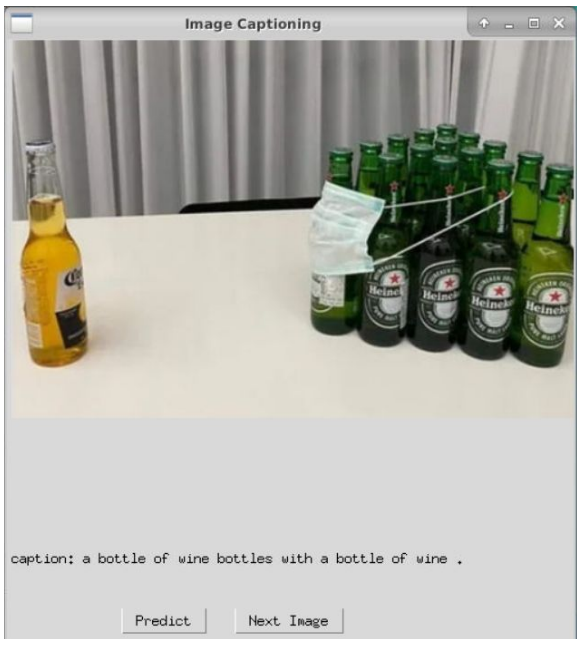
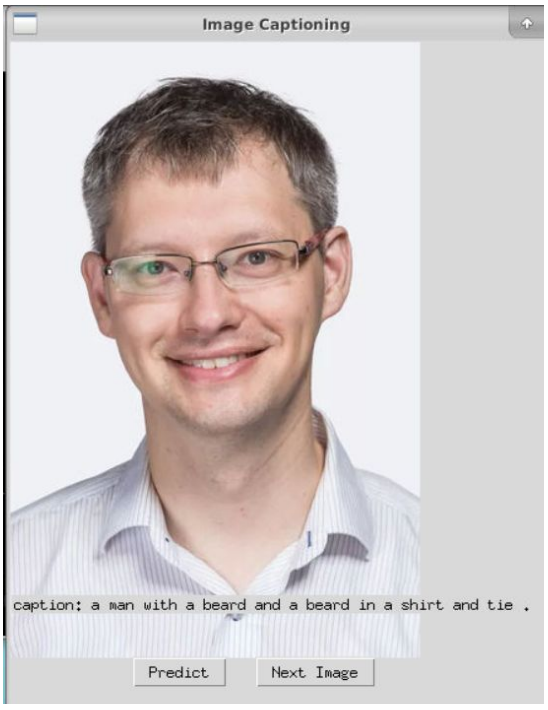
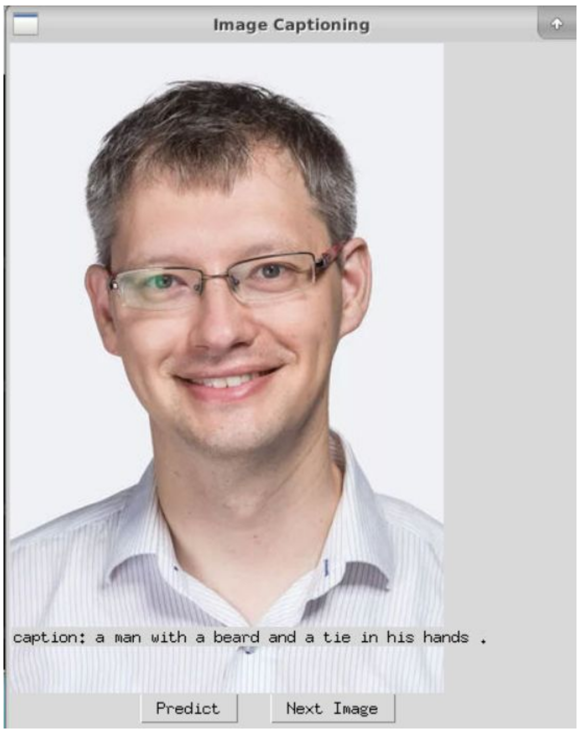
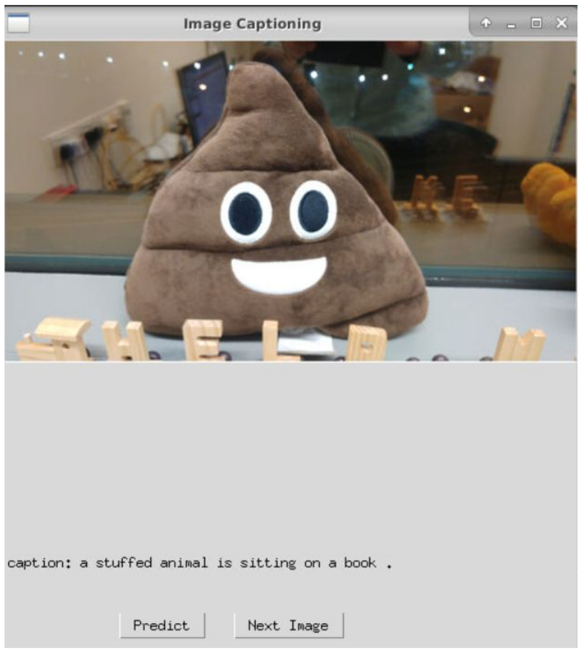
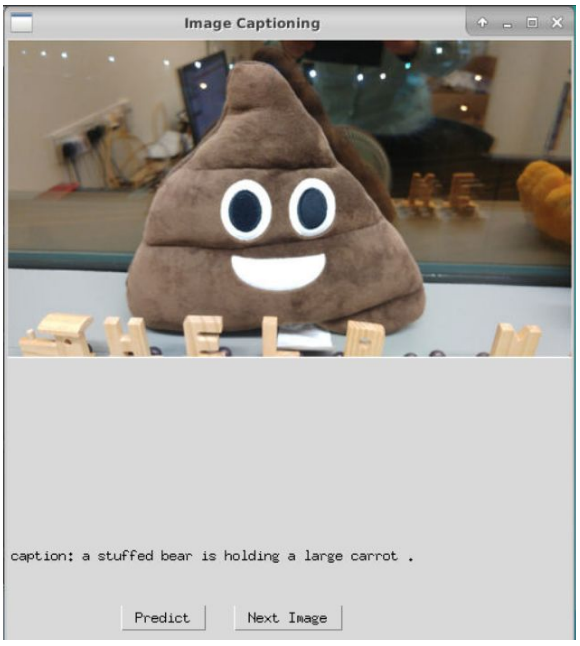
 caption: a bottle of wine bottles and a bottle of wine . Caption: A bottle of wine bottles and a bottle of wine.	 caption: a bottle of wine bottles with a bottle of wine . Caption: A bottle of wine bottles with a bottle of wine.

Table 2: GRU vs LSTM

Results of GloVe vs without GloVe (5 epochs each with 3230 iterations) with GRU

GloVe	Without GloVe
 caption: a man with a beard and a beard in a shirt and tie . Caption: A man with a beard and a beard in a shirt and tie.	 caption: a man with a beard and a tie in his hands . Caption: A man with a beard and a tie in his hands.
 caption: a stuffed animal is sitting on a book . Caption: A stuffed animal is sitting on a book.	 caption: a stuffed bear is holding a large carrot . Caption: A stuffed bear is holding a large carrot.

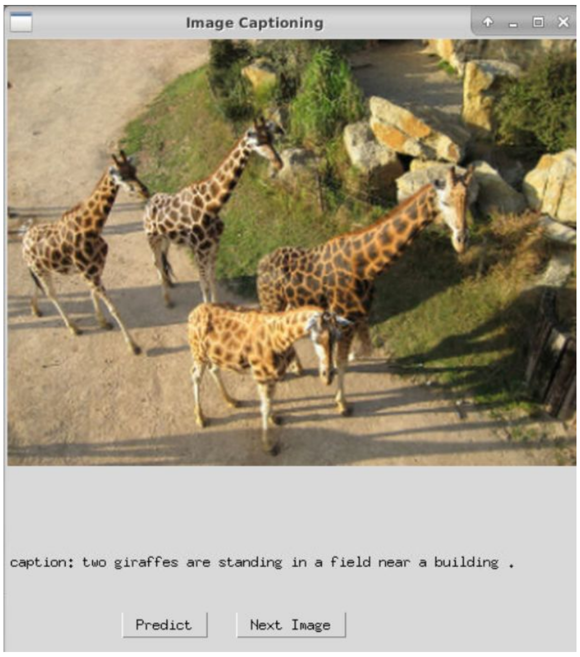
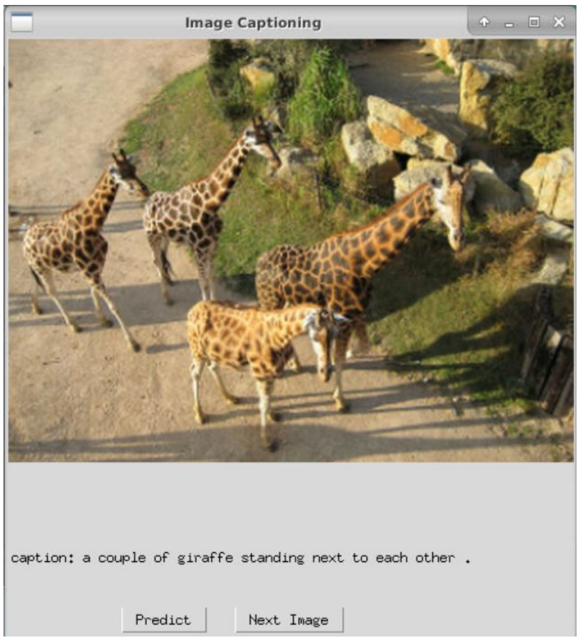
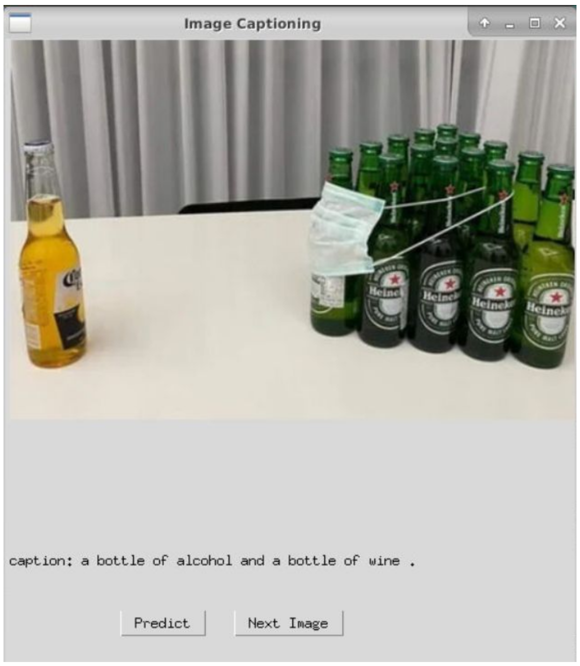
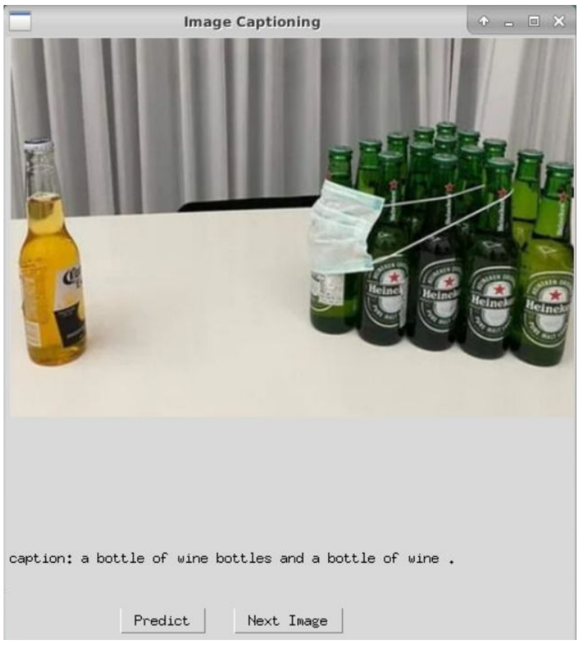
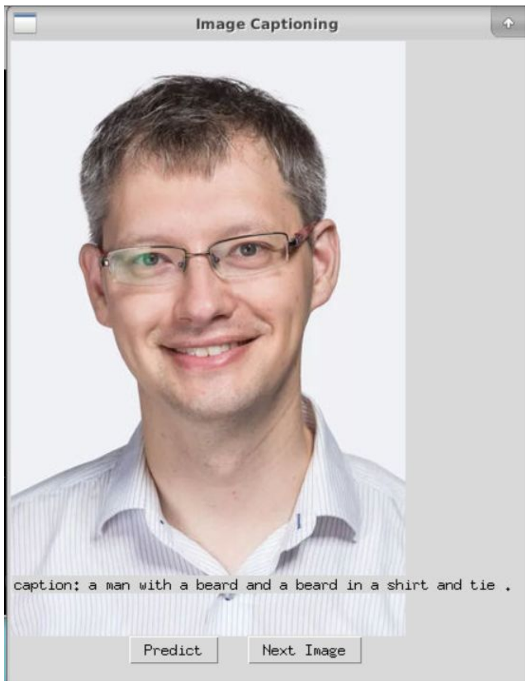
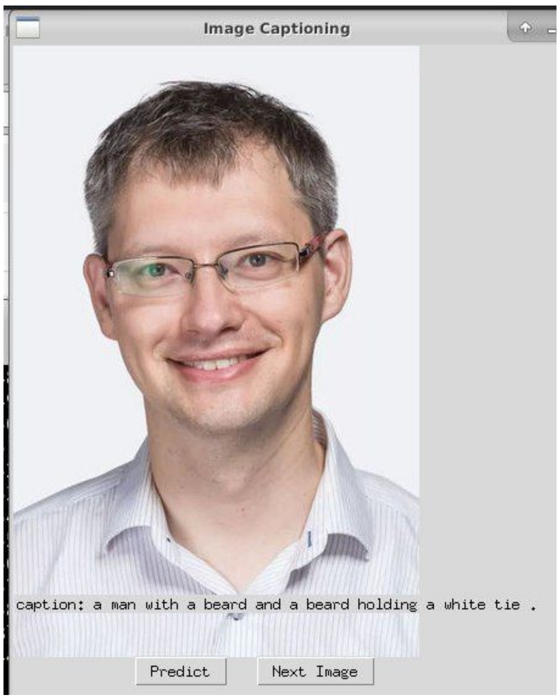
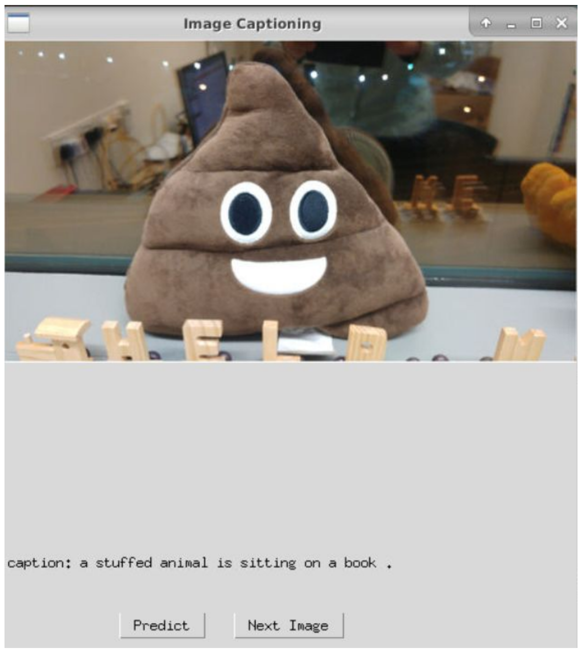
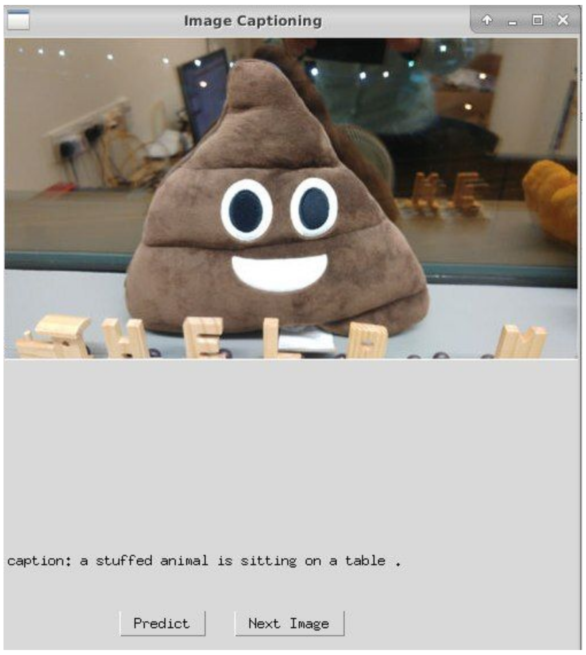
 caption: two giraffes are standing in a field near a building . Caption: Two giraffes are standing in a field near a building.	 caption: a couple of giraffe standing next to each other . Caption: A couple of giraffe standing next to each other.
 caption: a bottle of alcohol and a bottle of wine . Caption: A bottle of alcohol and a bottle of wine.	 caption: a bottle of wine bottles and a bottle of wine . Caption: A bottle of wine bottles and a bottle of wine.

Table 3: GloVe vs Without GloVe

Results obtained of 5 epochs vs 10 epochs (each with 3230 iterations) with GloVe and GRU

5 epochs	10 epochs
 Image Captioning caption: a man with a beard and a beard in a shirt and tie . Predict Next Image Caption: A man with a beard and a beard in a shirt and tie.	 Image Captioning caption: a man with a beard and a beard holding a white tie . Predict Next Image Caption: A man with a beard and a beard holding a white tie.
 Image Captioning caption: a stuffed animal is sitting on a book . Predict Next Image Caption: A stuffed animal is sitting on a book.	 Image Captioning caption: a stuffed animal is sitting on a table . Predict Next Image Caption: A stuffed animal is sitting on a table.

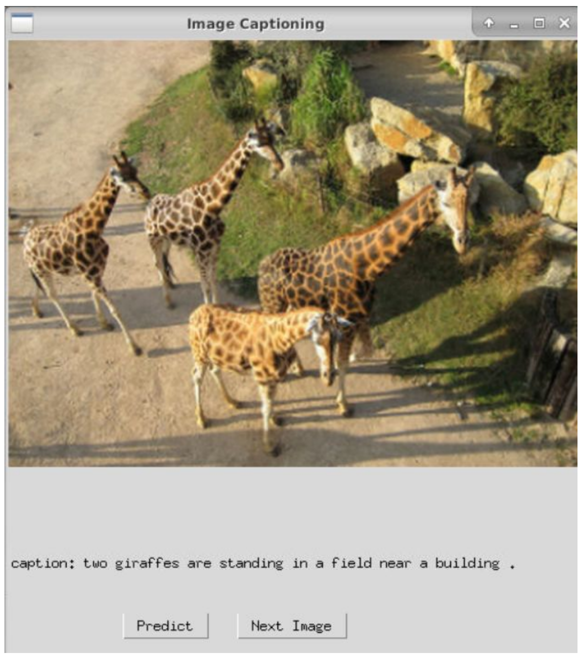
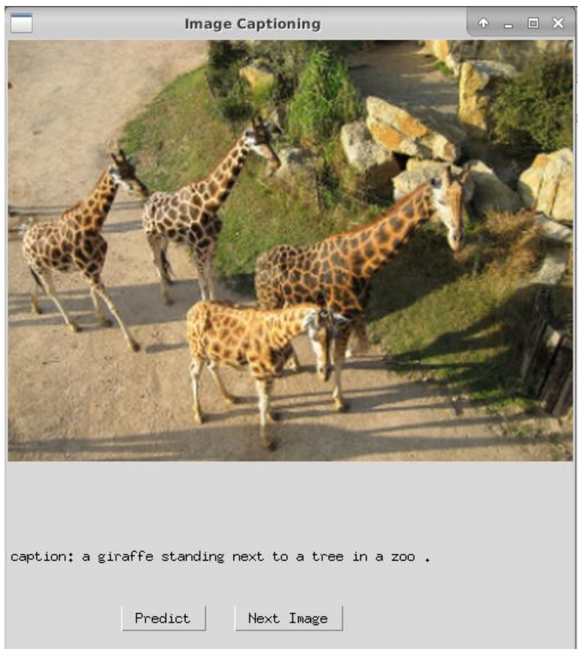
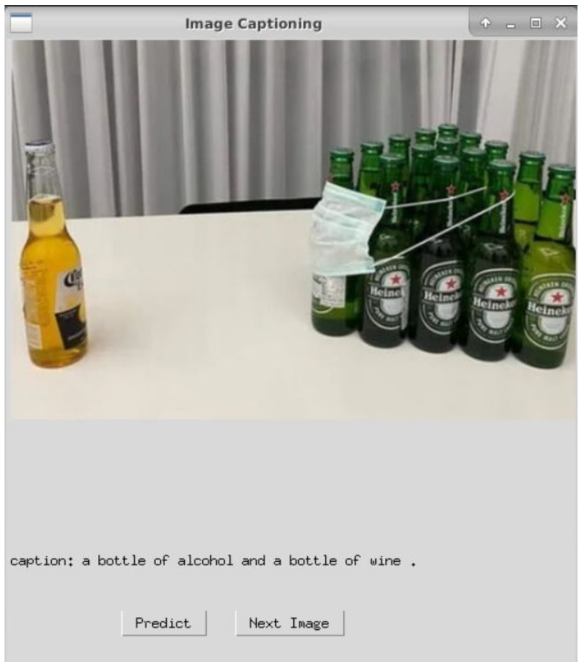
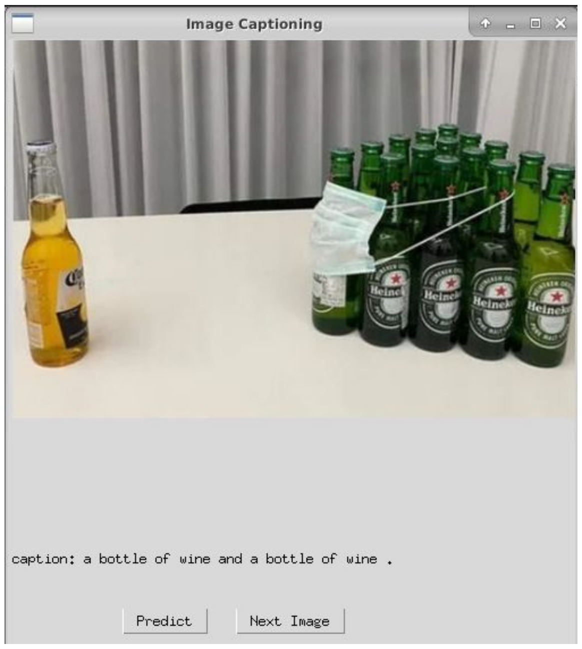
 caption: two giraffes are standing in a field near a building . Caption: Two giraffes are standing in a field near a building.	 caption: a giraffe standing next to a tree in a zoo . Caption: A Giraffe standing next to a tree in a zoo.
 caption: a bottle of alcohol and a bottle of wine . Caption: A bottle of alcohol and a bottle of wine.	 caption: a bottle of wine and a bottle of wine . Caption: A bottle of wine and a bottle of wine.

Table 4: 5 vs 10 epochs

Amusing screenshots and highlights for running with GloVe

Preliminary Testing (5 epochs with 20 iterations each)

We have decided to use the model on some of our own pictures (they were not included in the training data when we trained the model), as well as the ones provided by DL course. The results are rather comedic, to say the least.

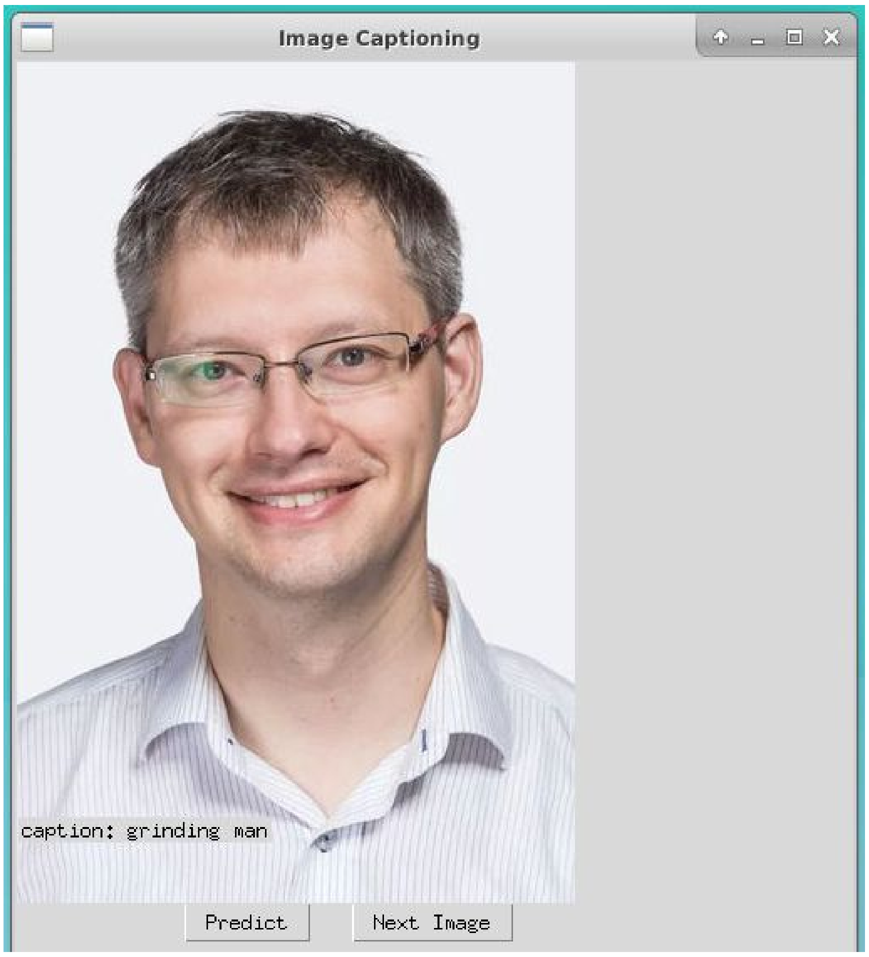
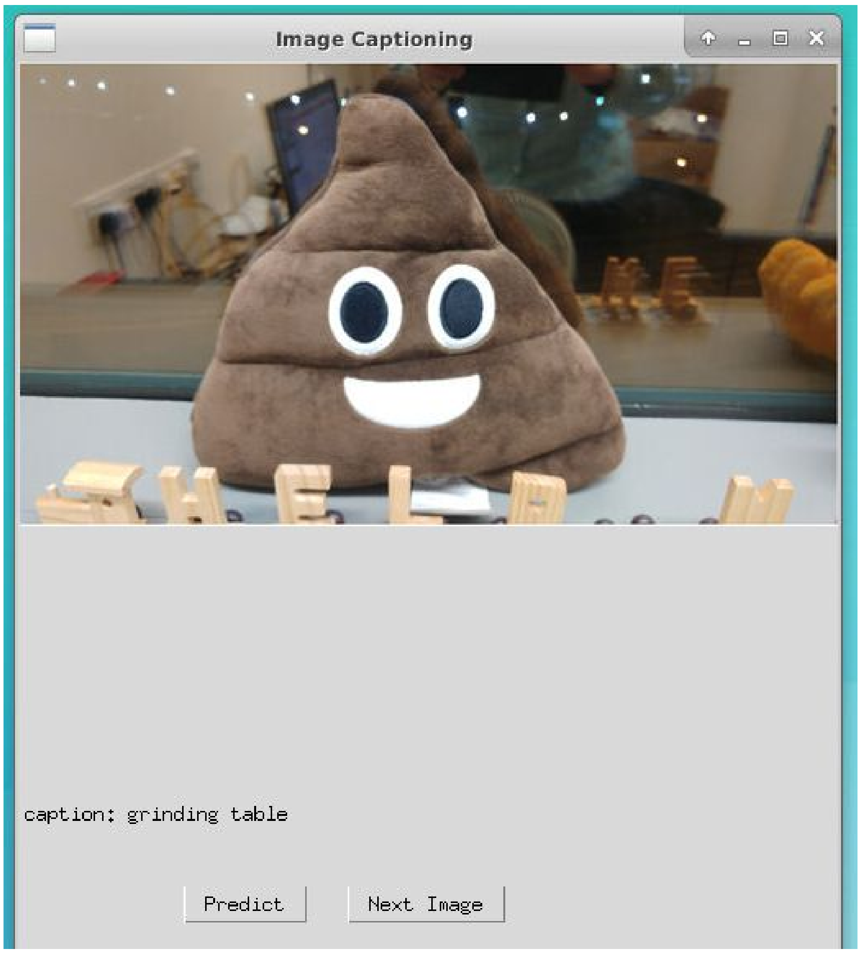


Figure 11.
Caption: Grinding man
(It’s not so wrong. This “man” made us “grind” the entire term)

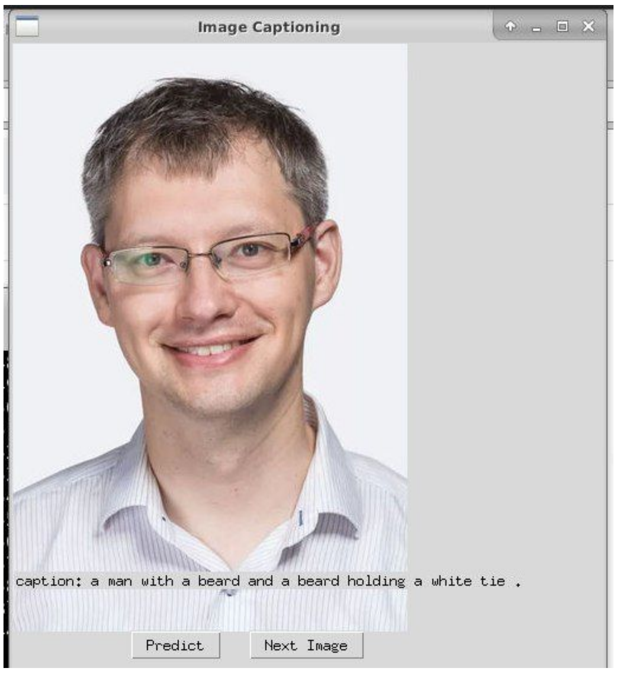


```
caption: grinding table
```

Figure 12.
Caption: Grinding table.
(Note the lack of acknowledgement on the obvious element)

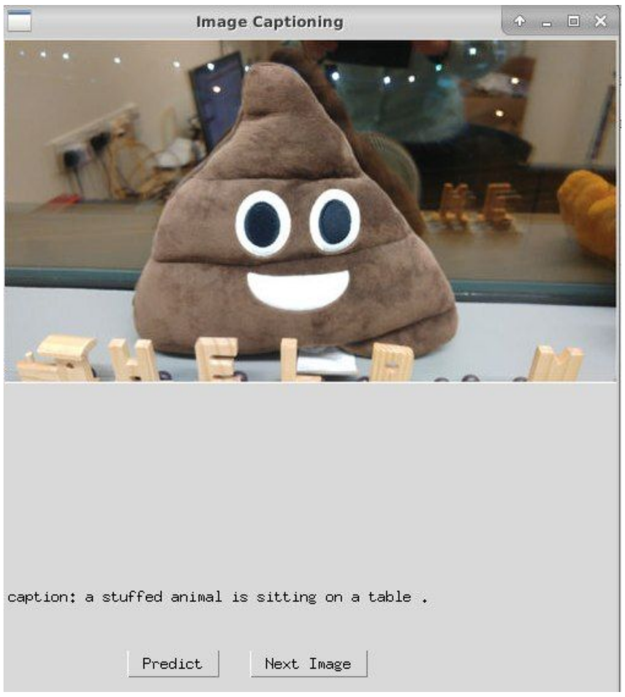
Final Testing (10 epochs)

Along the way, we decided to train the model on 10 epochs. After that was done, the model then began to show an increase in accuracy when it comes to identifying the photos that we have fed into it. We used a mix of different photos, along with the original photos that we have used for the Preliminary Test:



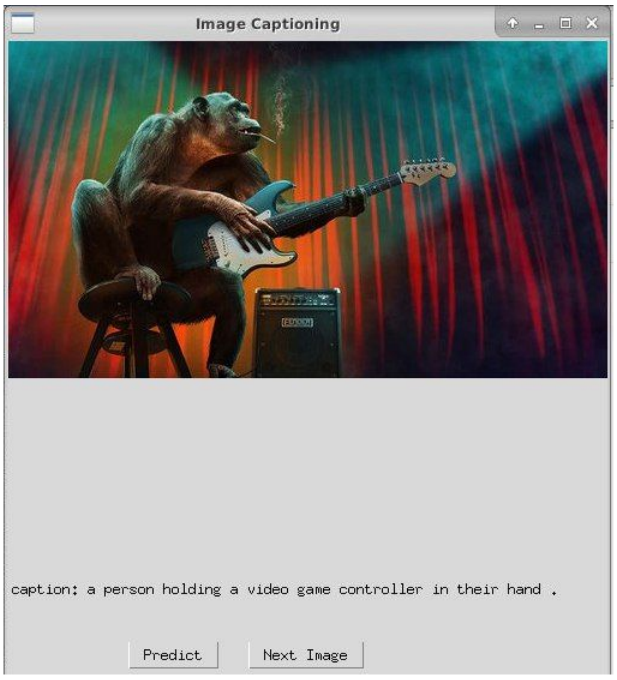
```
caption: a man with a beard and a beard holding a white tie .
```

Figure 13.
Caption: A man with a beard and a beard holding a white tie.
(I see no beard on this Handsome face!)



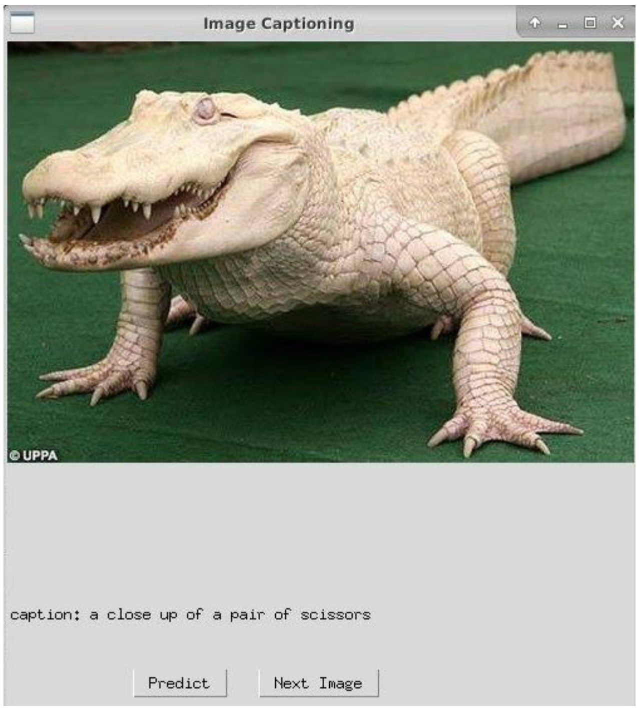
```
caption: a stuffed animal is sitting on a table .
```

Figure 14.
Caption: A stuffed animal is sitting on a table.
(For this time, the plush is correctly identified of its nature i.e. “stuffed”)



```
caption: a person holding a video game controller in their hand .
```

Figure 15.
Caption: A person holding a video game controller in their hand.
(This is a new photo that we have used for this test)



```
caption: a close up of a pair of scissors
```

Figure 16.
Caption: A close up of a pair of scissors
(Technically it's not so wrong)

Future Works with Attention Model

Attention model is proposed as a method for both alignment and for translations.

Alignment is the problem in machine translation that identifies which parts of the input sequence are relevant to each word in the output, whereas translation is the process of using the relevant information to select the appropriate output. In other words, attention models ensure that the captions generated are translated properly, along with aligning properly (i.e. the sequence).[4]

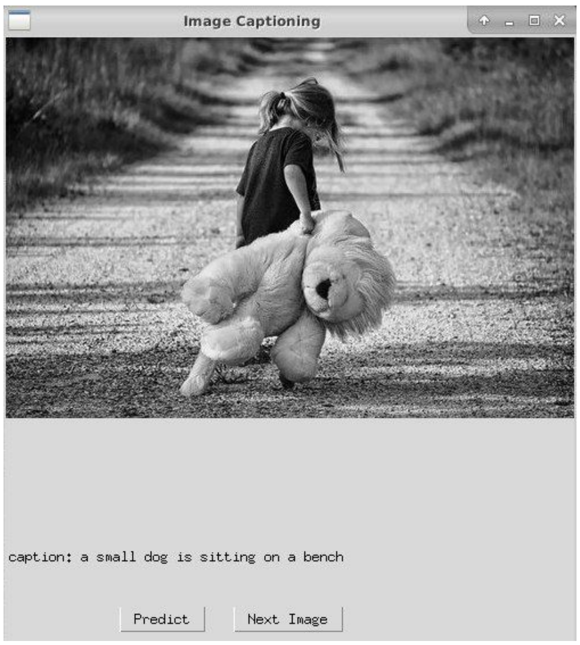
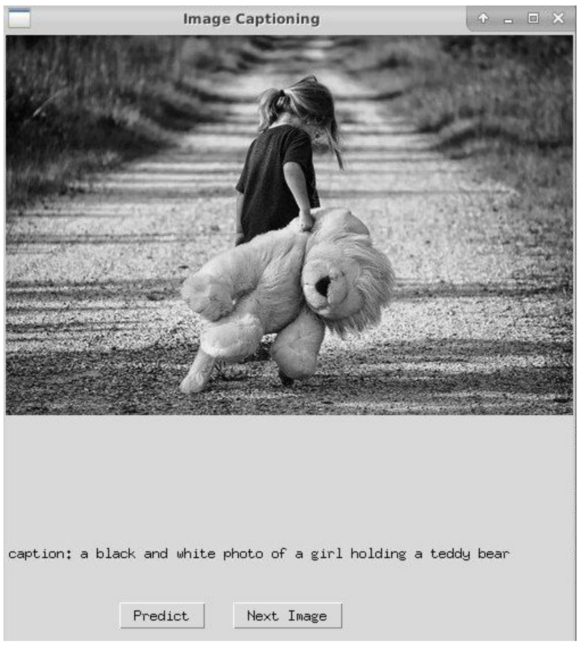
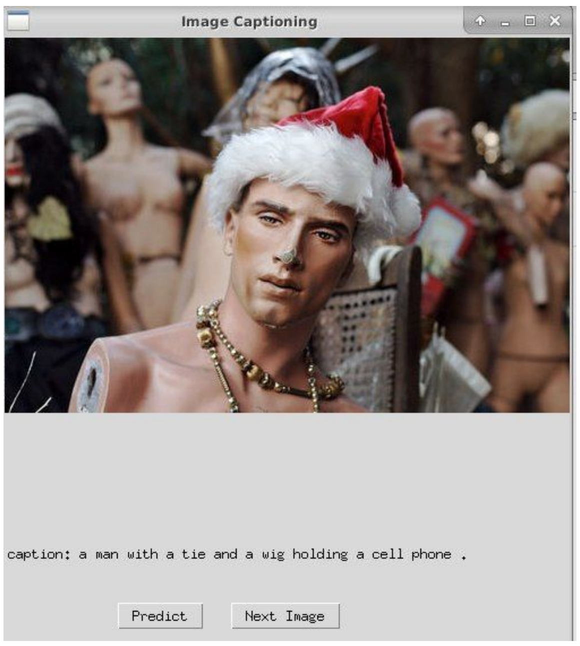
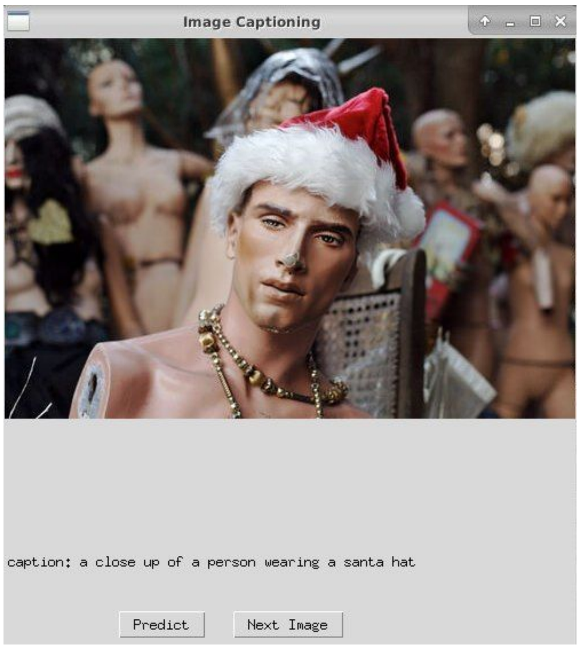
That is, to weigh and estimate the importance of a certain part of an image. We would need to be aware of the sequence that we have generated so far such that we can look at the image and decide what needs describing next. For example, after mentioning a person, it is logical to declare that he is holding a football.[5]

The attention network, in essence, considers the sequence generated thus far, and attends to the part of the image that needs describing next. It basically ignores the background bits and information that is not as necessary, and focuses on the main part.

We tried to implement an attention model to this project, but due to shortage of time we failed to do so. Instead we compared our results with this image captioning code which is implemented with an attention model. We added a GUI this code and the modified code can be found here. Due to the shortage of time, we used pretrained weights to run this model.

In comparison with predicting the same images with and without attention model, we realise that Attention Model can further improve the accuracy of the captions generated. A few of the results and comparisons are shown in the table below:

[4] "How Does Attention Work in Encoder-Decoder Recurrent ...." 13 okt.. 2017, https://machinelearningmastery.com/how-does-attention-work-in-encoder-decoder-recurrent-neural-networks/. Geopend op 27 apr.. 2020.
[5] "a PyTorch Tutorial to Image Captioning ...." https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Image-Captioning. Geopend op 27 apr.. 2020.

Without Attention Model	With Attention Model
 caption: a small dog is sitting on a bench Caption: A small dog is sitting on a bench.	 caption: a black and white photo of a girl holding a teddy bear Caption: A black and white photo of a girl holding a teddy bear.
 caption: a man with a tie and a wig holding a cell phone . Caption: A man with a tie and a wig holding a cell phone.	 caption: a close up of a person wearing a santa hat Caption: A close up of a person wearing a santa hat.

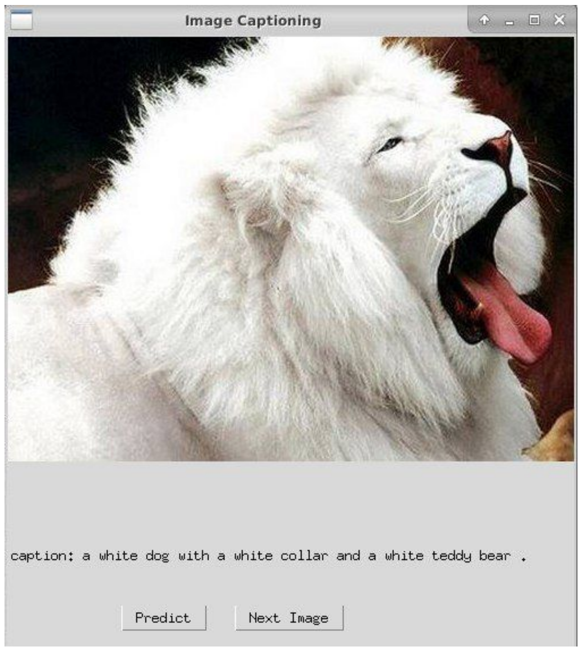
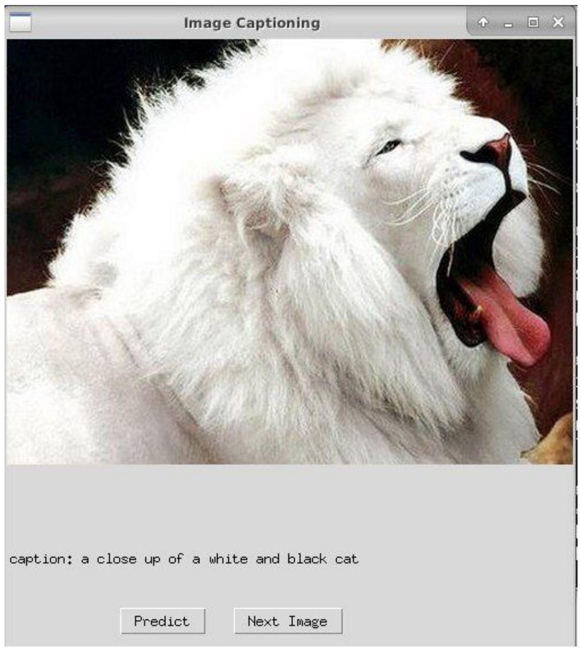
 Image Captioning caption: a white dog with a white collar and a white teddy bear . Caption: A white dog with a white collar and a white teddy bear.	 Image Captioning caption: a close up of a white and black cat Caption: A close up of a white and black cat.

Table 5: Attention vs Without attention

The captions generated without the attention model were rather inaccurate, though it is not that far off. For instance, looking at the second picture (man with a Santa hat), the model without the attention model indeed identified that the picture contains a man, but it also gave too much of a weight to insignificant details, e.g. the bench / chair on the left hand side of the man being identified as a cell phone.

On the other hand, the attention model, as the name suggests, gives a lot of attention to the foreground object, and has not only identified it properly (man with a Santa hat), but the model also went out of the way to indicate that it is a "close up" picture, all the while ignoring the background objects (such as the "phone" that was captioned by the model without attention).

Setbacks

Computational time and server response isan issue. Due to the circuit breaker measure in effect when the project was ongoing, only one of our group members was able to stay in school in order to connect to the VPN and use the machine provided for this DL Project. While the rest of us were also able to connect to the machine, there would be timeouts and downtimes when it comes to exceptionally long training times.

Even running 5 epochs can have moments whereby the machine stopped abruptly and as a result, necessitating a rerun.

As it was common and typical of this team, the COVID-19 situation is also very detrimental to our project progress and research because of the inability to meet outside (and thus having more immediacy when it comes to helping each other out or brainstorming), as well as not being to be in school to use the VPN-enabled machines as aforementioned.

Teammate Contributions

Grand - Testing with attention model, modified training code and video demo presentation. Report.
Nigel - Coding and refinements. Implementing GloVe. Video presentation and production. Report.
Andri - Coding and refinements. GUI coding. Report.
Valentine Zwe - Adjusting parameters and models of code; working on the backbone of the report.

Citations in footnotes.