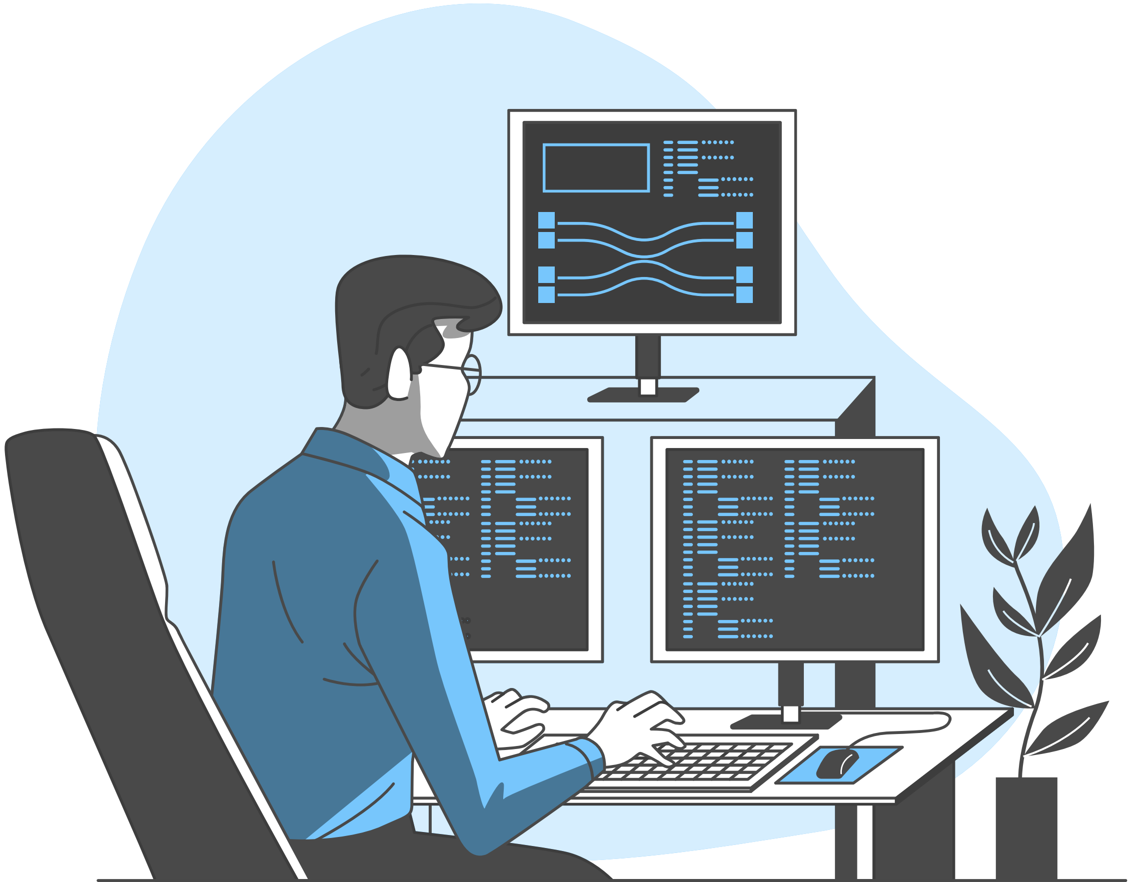
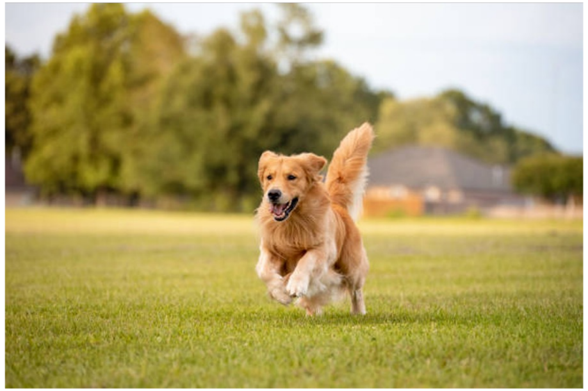


Pixels To Phrases

The Problem: Image Captioning

We have an image and we want to have a caption or description of what is happening in the image.



Caption: A dog is running in the park

We built an image caption generator to recognize the context of an image and annotate it with relevant captions using deep learning. It includes the labeling of an image with English keywords with the help of datasets provided during model training.

Loading the dataset

- Dataset used: Flickr8k (One folder contains images, another contains the captions)
- Image and its corresponding numericalized caption are sent to the Neural network for training.
- To numericalize the caption, the caption string was split into individual words using spacy tokenization and mapped to corresponding index values.

- Vocabulary dictionary is initialized and built with the numericalized captions list.
- Pytorch dataset was used to load the dataset to get a single image with the corresponding caption.
- Every batch of images and corresponding captions were padded to be of the same sequence length.
- Data loader function (get_loader) was created to load the dataset and the vocabulary into the model.

Models used

To build an image caption generator model, we need 2 models:

CNN

- Specialized deep neural networks used for the recognition and classification of images.
- Used to process the data represented as a 2D matrix like images.
- Analyzes the visual imagery by scanning them and extracting relevant features from that.
- Combines all the features for image classification.

LSTM

- A type of RNN, LSTM (Long short-term memory) is capable of working with sequence prediction problems.
- Mostly used for the next word prediction purposes.
- Throughout the processing of inputs, LSTM is used to carry out the relevant information and to discard non-relevant information.

- To build an image caption generator model we have to merge CNN with LSTM.
- We will use CNN as an encoder and LSTM as a decoder.
- CNN- To extract features from the image. A pre-trained model called ResNet-50 is used for this.
- LSTM- To generate a description from the extracted information of the image.

...

Our Project: Training Parameters

01 **Batch Size**

32 images per epoch

02 **Learning Rate**

Initially 3e-4
Then 5e-4

03 **Epochs**

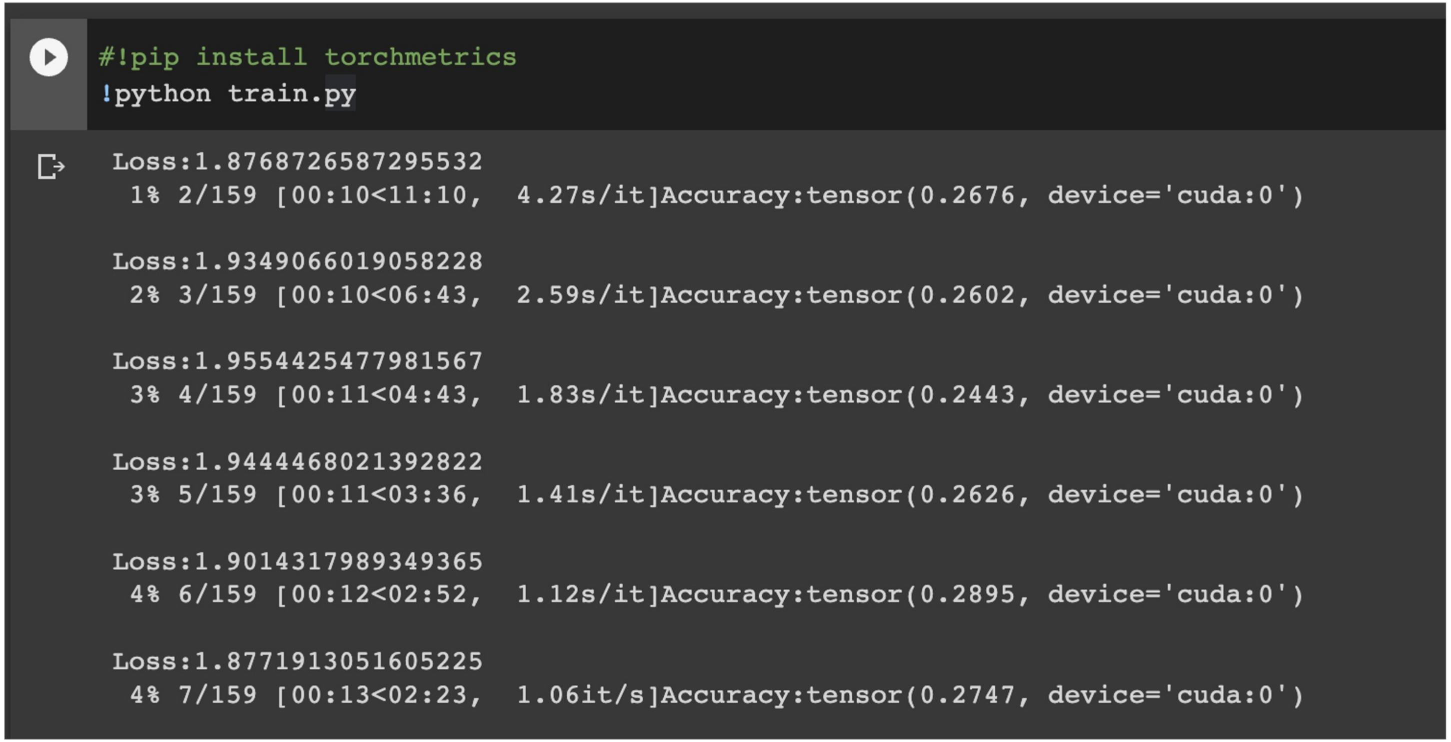
50 epochs

04 **Optimization Parameters**

Size of Embedding= 256

Number of hidden layers = 256

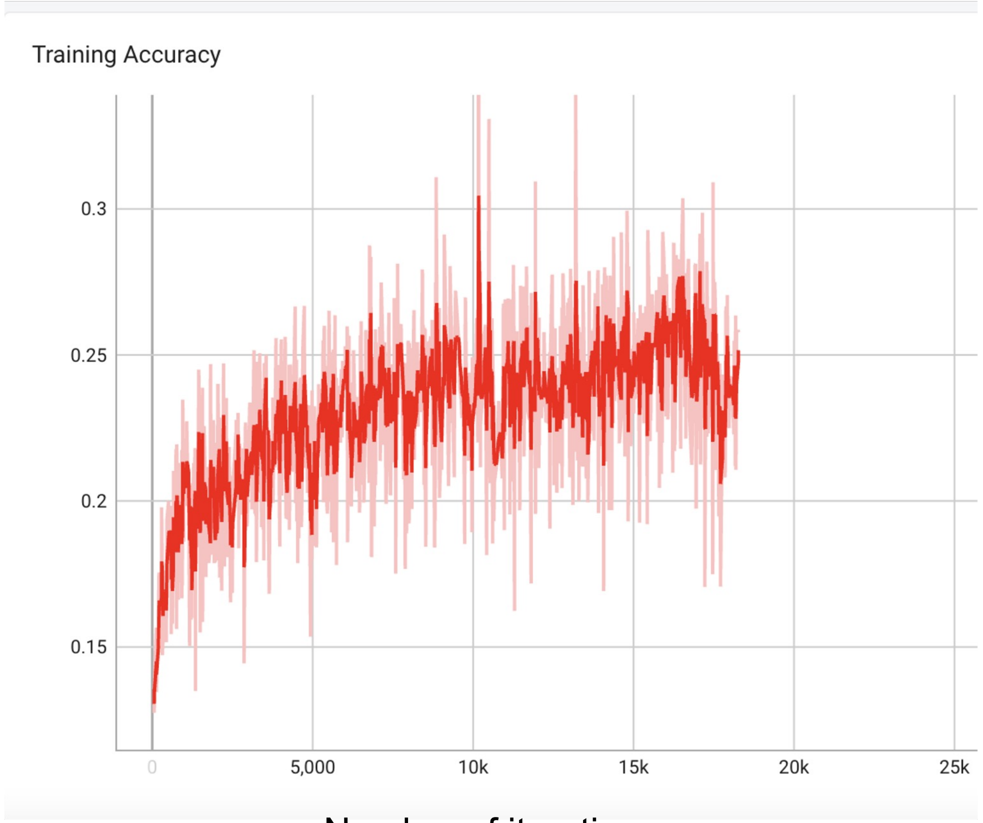
Accuracy & Loss Counting during Training



```
#!pip install torchmetrics
!python train.py
```

```
Loss:1.8768726587295532
 1% 2/159 [00:10<11:10,  4.27s/it]Accuracy:tensor(0.2676, device='cuda:0')

Loss:1.9349066019058228
 2% 3/159 [00:10<06:43,  2.59s/it]Accuracy:tensor(0.2602, device='cuda:0')

Loss:1.9554425477981567
 3% 4/159 [00:11<04:43,  1.83s/it]Accuracy:tensor(0.2443, device='cuda:0')

Loss:1.9444468021392822
 3% 5/159 [00:11<03:36,  1.41s/it]Accuracy:tensor(0.2626, device='cuda:0')

Loss:1.9014317989349365
 4% 6/159 [00:12<02:52,  1.12s/it]Accuracy:tensor(0.2895, device='cuda:0')

Loss:1.8771913051605225
 4% 7/159 [00:13<02:23,  1.06it/s]Accuracy:tensor(0.2747, device='cuda:0')
```

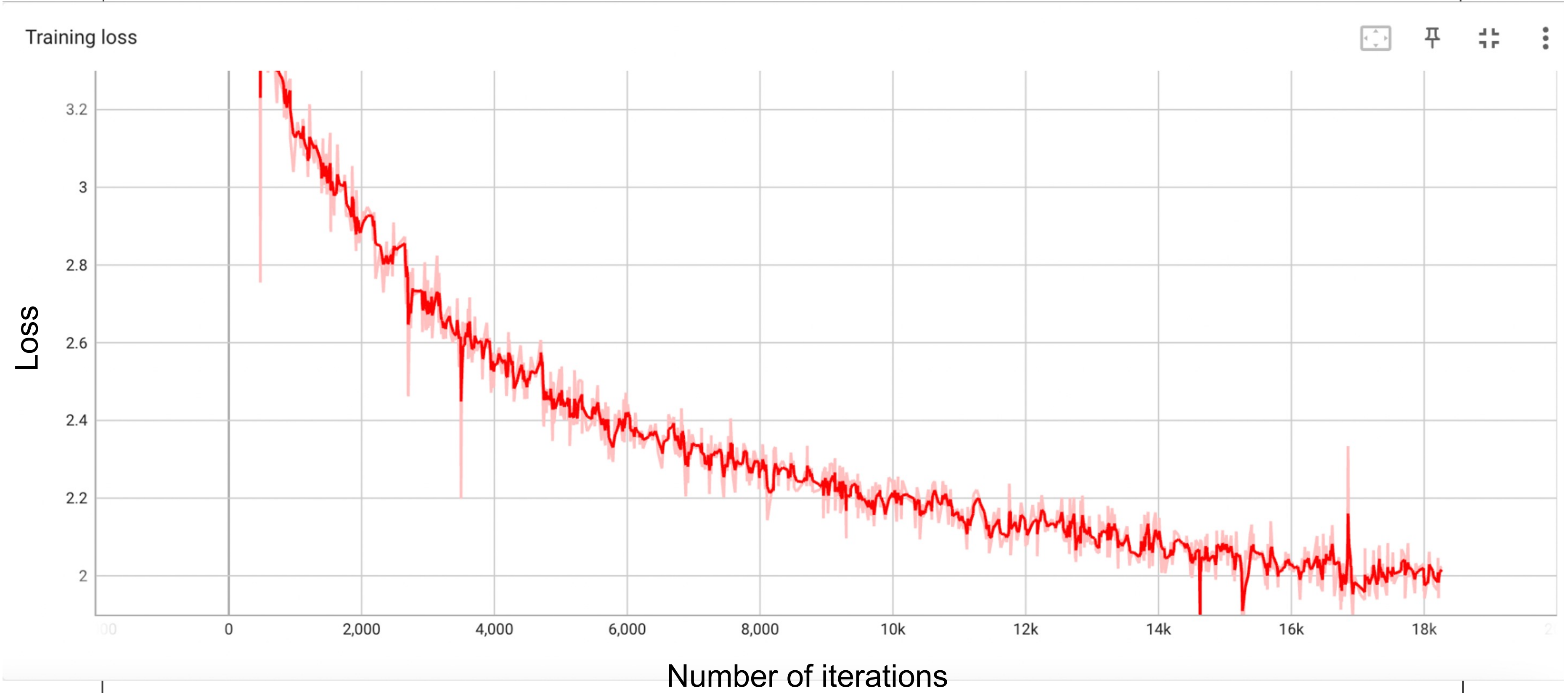
Training Accuracy



Accuracy

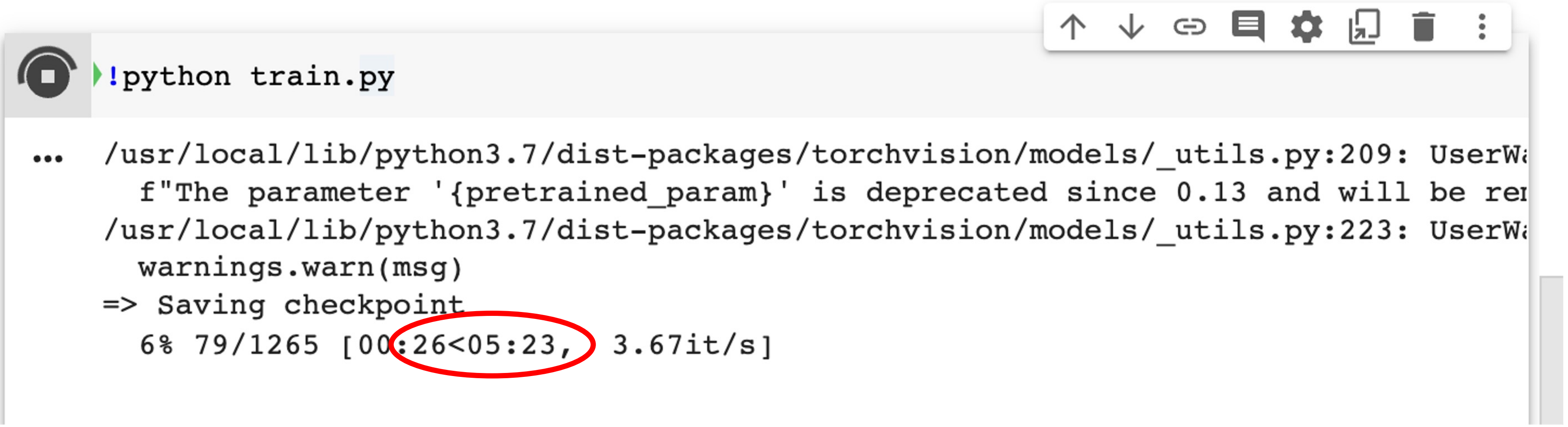
Number of iterations

Training Loss



Time required for epochs

According to the Colab/PC config, each epoch took 5 minutes. So, the total training took around 4 hours.



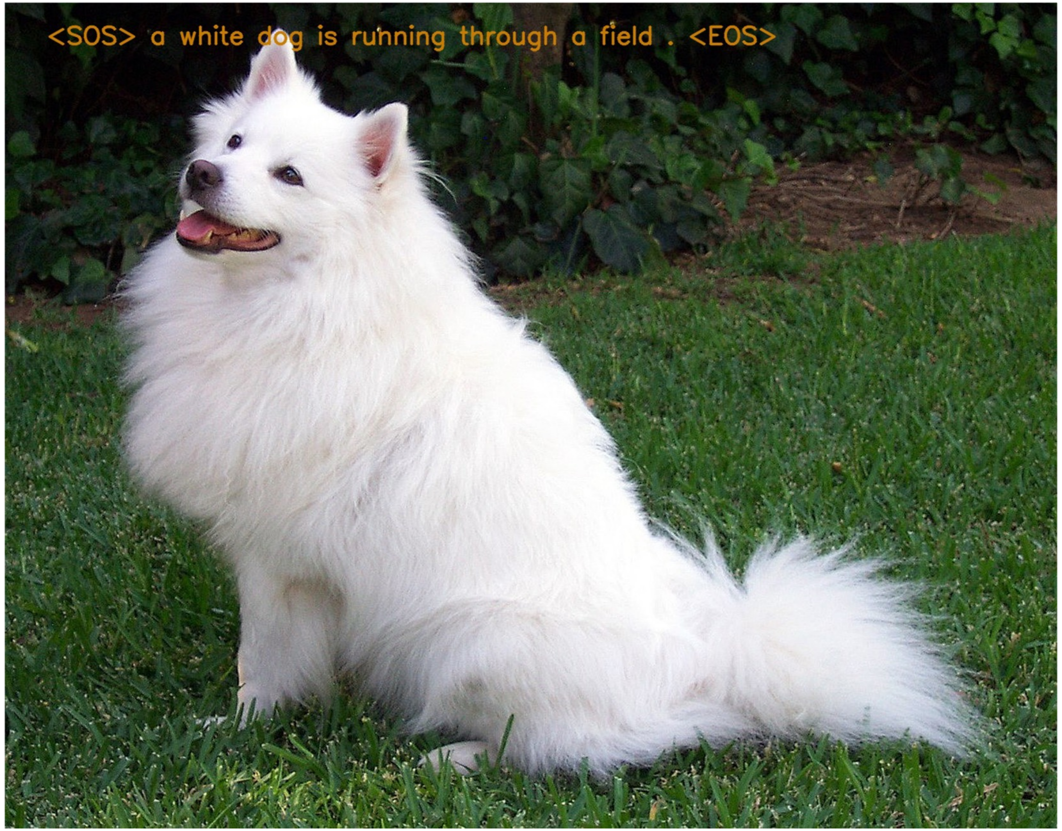
```
!python train.py
```

```
/usr/local/lib/python3.7/dist-packages/torchvision/models/_utils.py:209: UserWa
  f"The parameter '{pretrained_param}' is deprecated since 0.13 and will be ren
/usr/local/lib/python3.7/dist-packages/torchvision/models/_utils.py:223: UserWa
  warnings.warn(msg)
=> Saving checkpoint
  6% 79/1265 [00:26<05:23,  3.67it/s]
```

Test images

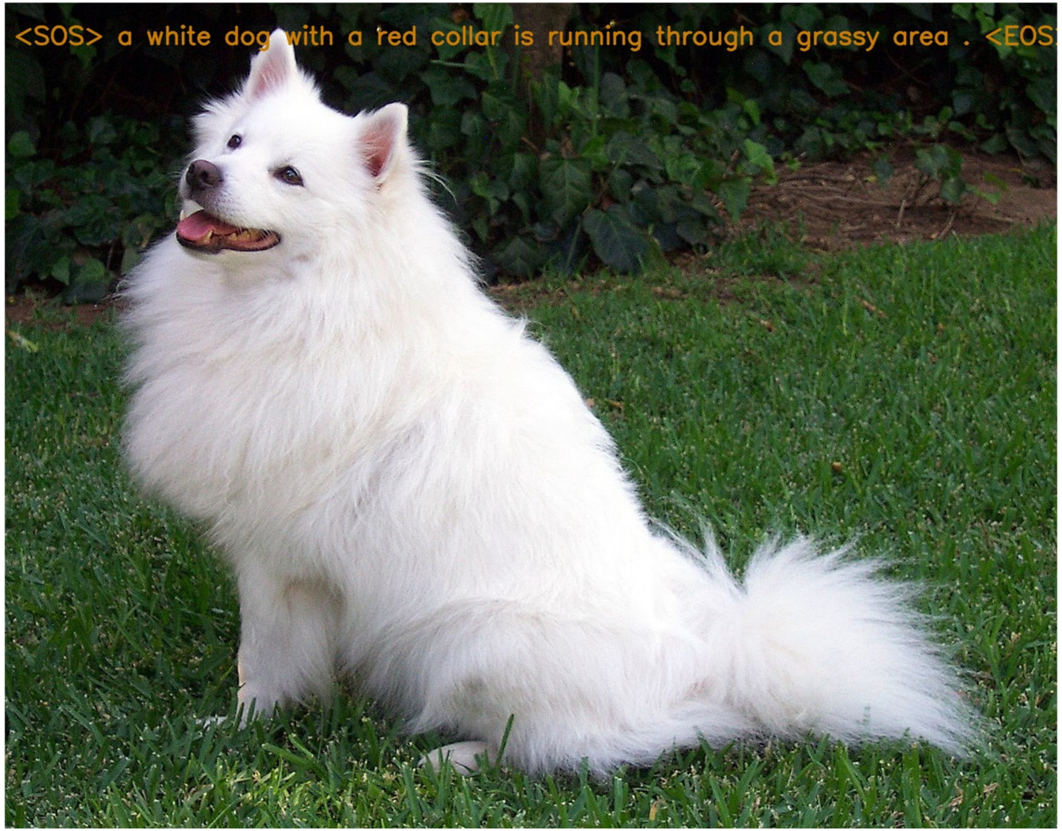
Model Test Result on Random Image

For 10 Epoches



Model Test Result on Random Image

For 50 Epoches


<SOS> a white dog with a red collar is running through a grassy area . <EOS

Test results

Example 1 CORRECT: Dog on a beach by the ocean
Example 1 OUTPUT: <SOS> a white dog with a red collar is running through a grassy area . <EOS>
Example 2 CORRECT: Child holding red frisbee outdoors
Example 2 OUTPUT: <SOS> a little girl in a red shirt is running through a field of grass . <EOS>
Example 3 CORRECT: A cowboy riding a horse in the desert
Example 3 OUTPUT: <SOS> a man is standing on a hill with a dog in the background . <EOS>

Aspects we found interesting

- As number of epochs increase, statement becomes more muddled.
- ReLu was used in the last layer instead of softmax, to prevent the exponential growth in the computation required to operate the neural network.
- For LSTM, it has a tendency to forget a long sequence.