# Cohort19--Group 12

| ***Guide*** | ***Prof. Vineeth Gandhi*** |
| --- | --- |
| ***Mentor*** | ***Brahmani Nutakki*** |
| ***Team Members*** | Rashmi Gandra (2218437)  Bhanu Theja Tenneti ([2203514](https://aiml.iiith.talentsprint.com/dashboard))  Ujwal Reddy (22184  Kumar Paritosh (2218229) |

Contents

[Cohort19--Group 12](#_heading=h.gjdgxs) **1**

[Problem Statement](#_heading=h.30j0zll) **3**

[Literature Review](#_heading=h.3znysh7) **3**

[Methodologies](#_heading=h.tyjcwt) **4**

[Block wise steps for the execution of the Project](#_heading=h.3dy6vkm) 4

[Figure 1 Methodology flow diagram](#_heading=h.6077caxf63qz) 4

[Considering Data set for pre-processing.](#_heading=h.1t3h5sf) 4

[Load pre trained model](#_heading=h.26in1rg) 5

[● Resnet 50 :](#_heading=h.9en5i6bg47q7) 5

[Figure 2 The proposed ResNet50 model after dropping the classifier layer](#_heading=h.8q2a7pbj4fbe) 6

[Figure 3 A representative figure of LSTM model architecture](#_heading=h.np8ri5st2aa5) 6

[Tokenize the captions](#_heading=h.fbr4zjvl8w0i) 7

[● Glove](#_heading=h.zbr3o3ajc8qt) 7

[Input the features (extracted) and Captions to LSTM Model](#_heading=h.rs4ss2a2utu7) 7

[Model Architecture](#_heading=h.98gg2eidble3) 7

[Figure 4 8k architecture](#_heading=h.k2lqxkgf8j3) 8

[Figure 5 30k architecture](#_heading=h.2w6lfve17giz) 9

[Transformer model:](#_heading=h.ig7v9jxh23cz) 9

[Train the model with data](#_heading=h.lnxbz9) 10

[Evaluation of Test Data](#_heading=h.7ns7bykhe9fo) 10

[Figure 6 Bleu Score with 8k LSTM Model](#_heading=h.124cikdyry1a) 10

[Figure 7 Bleu Score with 30k LSTM Model](#_heading=h.mbe8izda871s) 10

[Deployment](#_heading=h.cwqjdnaxm4sl) **11**

[Figure 8 Template form for getting data from user](#_heading=h.jgrec1ddni25) 11

[Figure 9 Model define for image prediction](#_heading=h.k4sb139ic3fq) 11

[Figure 10 image request function for collecting and processing the loaded images based on the method selected by the user](#_heading=h.6acmz1adggkx) 12

[Captioning:](#_heading=h.ey1zk54408ok) **13**

[Figure 11 Captions generated from the loaded images](#_heading=h.mi7uvhe03hz7) 13

[Enhanced Phase](#_heading=h.1ksv4uv) **13**

[Future Scope](#_heading=h.913ajpi9iq4e) **14**

[Challenges](#_heading=h.2jxsxqh) **14**

[Applicability in the real world](#_heading=h.z337ya) **14**

[References:](#_heading=h.3j2qqm3) **15**

**Title - Automatic Image Captioning**

# **Problem Statement**

To generate the caption of a given image automatically based on Deep Neural Networks. The project involves developing an automatic image captioning system that can generate captions that are linguistically plausible and semantically truthful to the content of the given image.

# **Literature Review**

Image captioning—the task of providing a natural language description of the content within an image—lies at the intersection of computer vision and natural language processing. As both these research areas are highly active and have experienced many recent advances, progress in image captioning has naturally followed suit. On the computer vision side, improved convolutional neural network and object detection architectures have contributed to improved image captioning systems. On the natural language processing side, more sophisticated sequential models, such as attention-based recurrent neural networks, have similarly resulted in more accurate caption generation. We have investigated some of the methods implemented by various research papers for image captioning.

From [[1],](#_heading=h.3j2qqm3) the Resnet-LSTM model was used for the image captioning process. Here Resnet Architecture is used for encoding and LSTM is used for decoding. Once when the image is sent to Resnet (Residual Neural Network) it extracts the image features then with the help of vocabulary that is built using training captions data, then the model is trained with those two parameters as input. The results show the ResNet-LSTM model has higher accuracy compared to CNN-RNN and VGG Model.

From [[2],](#_heading=h.3j2qqm3) we find a modification of the conventional Transformer, specifically adapted to the task of image captioning. The proposed Transformer encodes 2D position and size relationships between detected objects in images, building upon the bottom-up and top-down image captioning approach. SO basically, they first use an object detector to extract appearance and geometry features from all the detected objects in the image, thereafter, they use the Object Relation Transformer to generate the caption text. Their results on the MS-COCO dataset demonstrate that the Transformer does indeed benefit from incorporating spatial relationship information.

From [[3],](#_heading=h.3j2qqm3) research on the field is mostly focused on deep learning-based methods, where attention mechanisms along with deep reinforcement and adversarial learning appear to be at the forefront of this research topic. Recent methodologies such as [[7]](#_heading=h.3j2qqm3) UpDown, [[5]](#_heading=h.3j2qqm3) OSCAR, [[6]](#_heading=h.3j2qqm3) VIVO, [[8]](#_heading=h.3j2qqm3) Meta Learning and a model that uses [[9]](#_heading=h.3j2qqm3) conditional generative adversarial nets were used for the image captioning. Although the GAN-based model achieves the highest score, UpDown represents an important basis for image captioning and OSCAR and VIVO are more useful as they use novel object captioning.

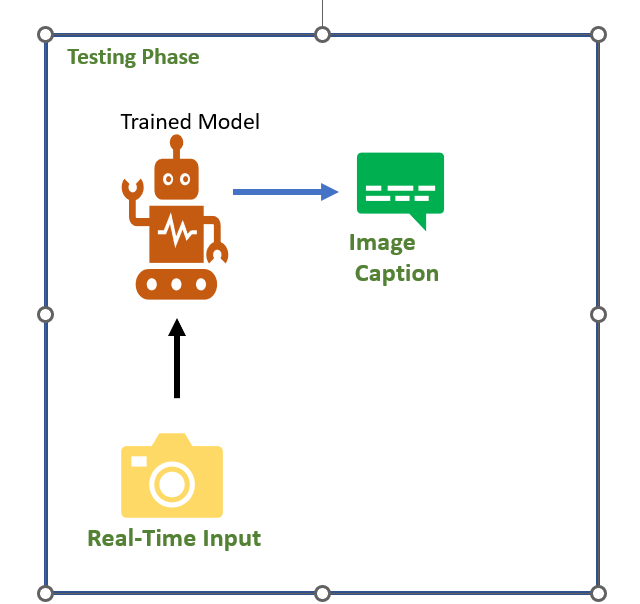
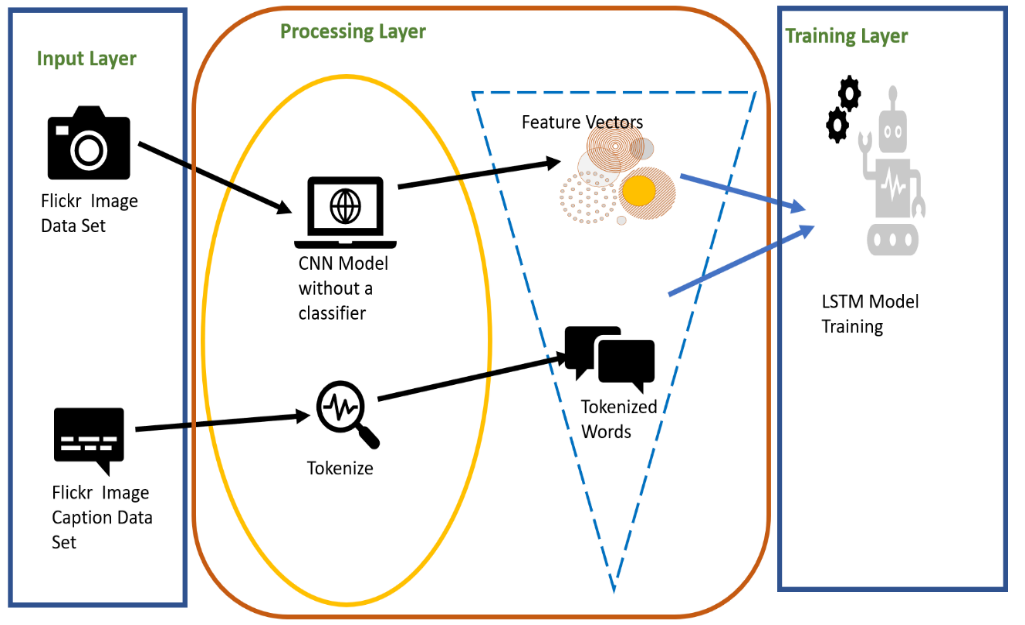
# **Design Overview**

## **Block wise steps for the execution of the Project**

1. Considering Data set for pre-processing.

* In this project we have taken the Flicker8k and 30k dataset.

1. Processing the Data set
2. Feature extraction (use the existing pre-trained model to extract Feature Vectors from the Images)
3. Input the features (extracted) and Captions to LSTM Model
4. Train the model with the data
5. Test the predictions with real-time Images



##### Figure 1 Methodology flow diagram

### Considering Data set for pre-processing.

There are many open-source datasets available for this problem, like Flickr 8k (containing images), MS COCO (containing 180k images), Flickr 30k (containing 30k images), etc. The dataset consists of 8k/30k images, each image having 5 captions in the text file(s). We are considering Flickr 8k and Flickr 30k datasets for this project.

**Processing the Data set**

* Deep Lake API[[11]](#_heading=h.3j2qqm3) is **optimized for maximizing throughput to GPU processing**. It includes CPU prefetching, decompression or decoding, transformations, and GPU memory transfer in a deep learning framework's expected layout.
* A standard train-test split will be done with 80:20 ratio.
* The captions will be tokenized and fed to the LSTM. Captions are treated like target variables that the model is learning to predict. Since the caption will be predicted as one word at a time, each word is encoded into a fixed-sized vector. All the unique words will be mapped to an index and vice versa including the tokens added.

### 

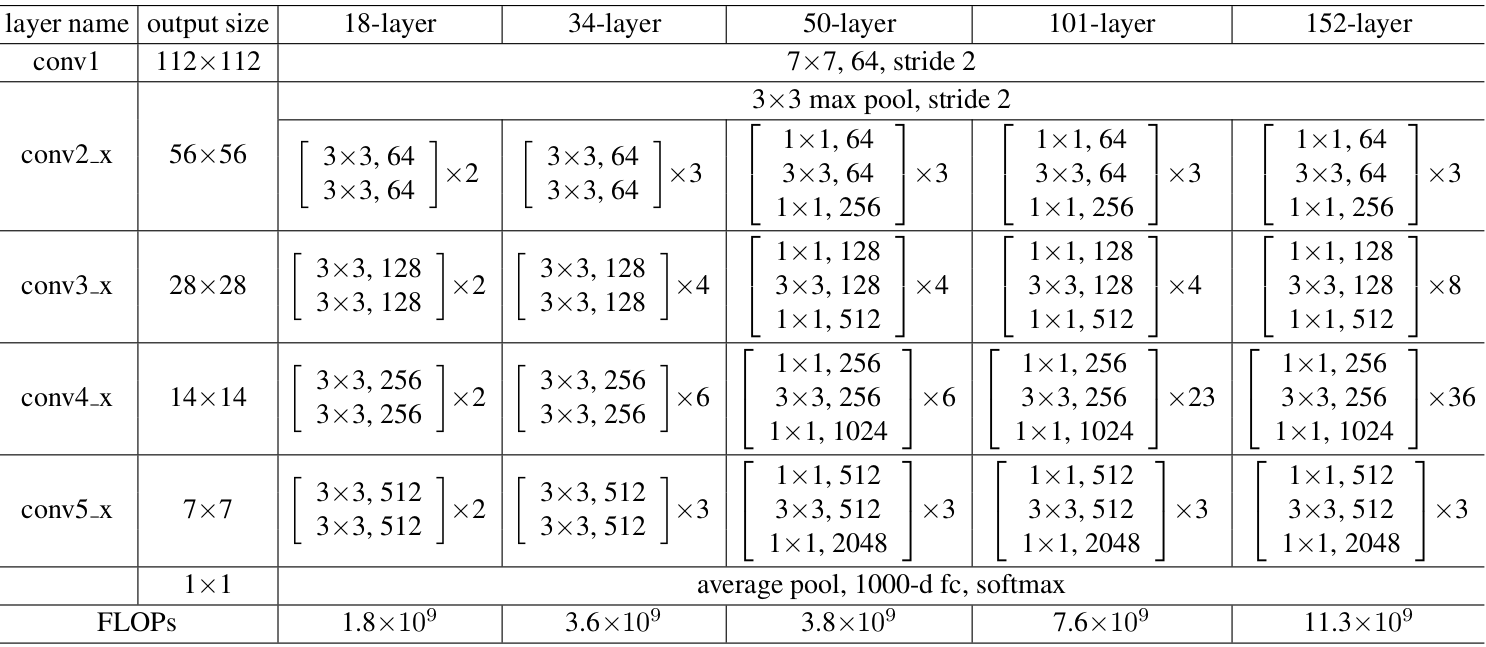
### **Load pre trained model**

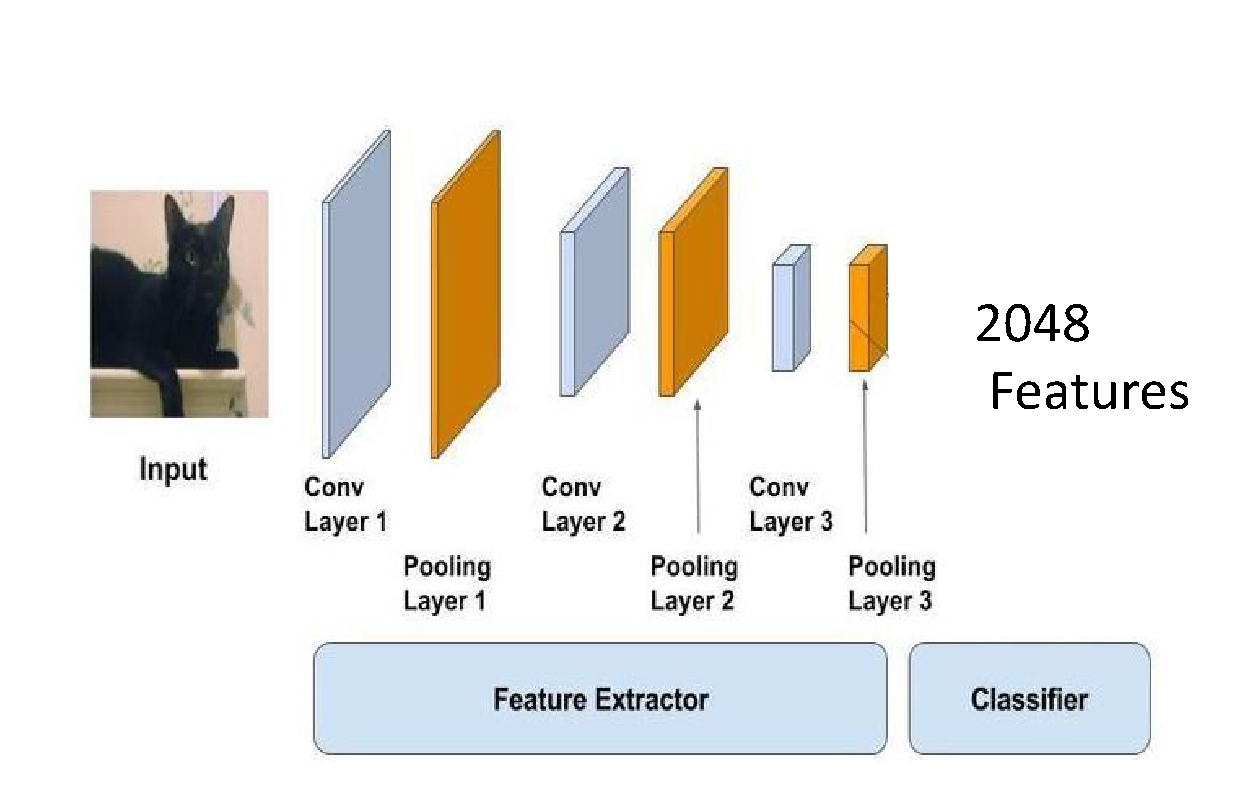
#### Resnet 50 :

ResNet-50[[12]](#_heading=h.3j2qqm3) is a convolutional neural network that is 50 layers deep. You can load a pre-trained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224. This architecture involves using Convolutional Neural Network (CNN) layers for feature extraction on input data combined with LSTMs to perform sequence prediction on the feature vectors.

We want to extract the features of the images using RESNET50. ResNet50 is a variant of [ResNet model](https://iq.opengenus.org/resnet/) which has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer. It has 3.8 x 10^9 Floating points operations. It is a widely used ResNet model.

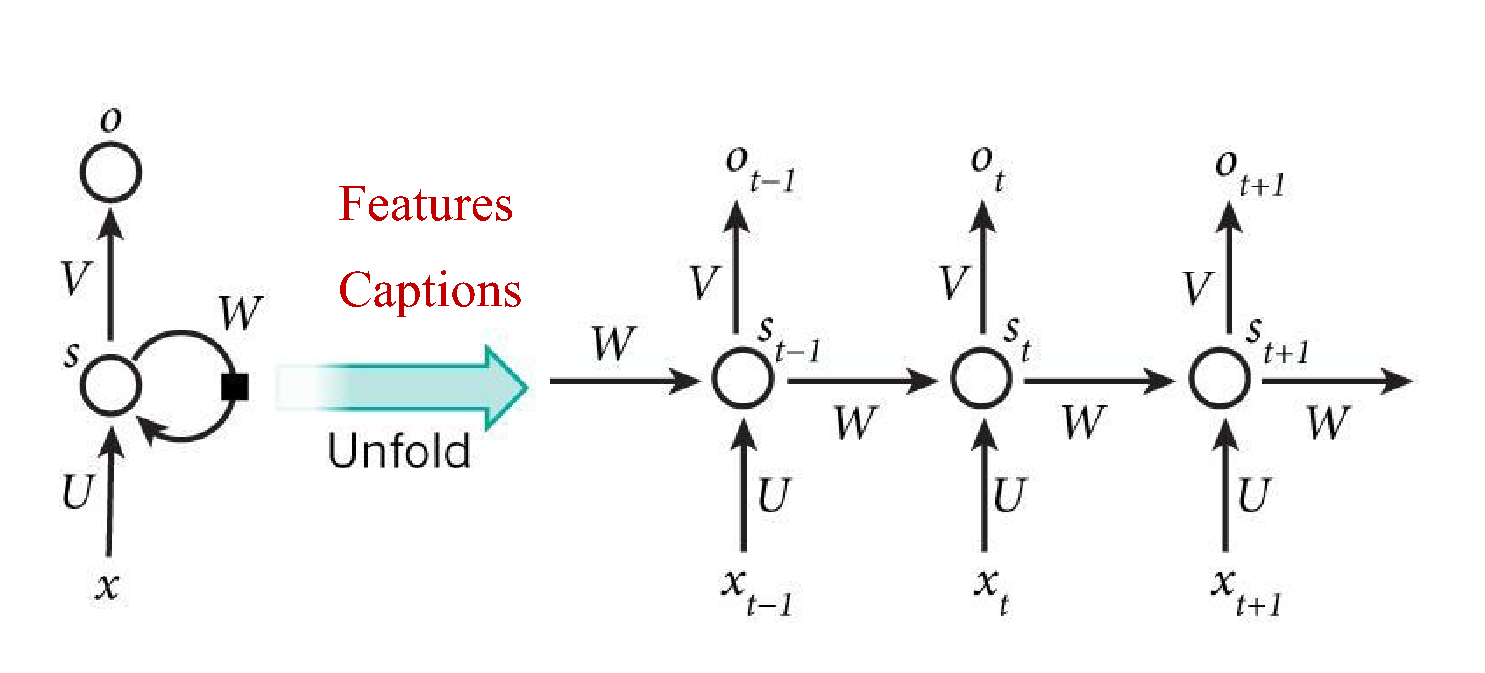
The model has been selected based on the project requirements and the computing capacity we have. We will use the deep CNN architecture to extract features.





##### Figure 2 The proposed ResNet50 model after dropping the classifier layer

Recurrent neural networks (RNN) are a class of neural networks that is powerful for modelling sequence data such as time series or natural language. After producing the output, it is copied and sent back into the recurrent network. For making a decision, it considers the current input and the output that it has learned from the previous input (Denny, Britz 2015). The Long Short-Term Memory (LSTMs) are a class of RNNs to process the sequence input.



##### Figure 3 A representative figure of LSTM model architecture

The LSTM based model will be used to predict the sequences of words, called the caption, from the feature vectors obtained from the ResNet50 model. An LSTM layer requires a three-dimensional input and LSTMs by default will produce a two-dimensional output as an interpretation from the end of the sequence.

### Tokenize the captions

#### Glove

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. [[10]](#_heading=h.3j2qqm3)The advantage of GloVe is that, unlike Word2vec, GloVe does not rely just on local statistics (local context information of words), but incorporates global statistics (word co-occurrence) to obtain word vectors.

### Caption Generation Model

#### Model Architecture

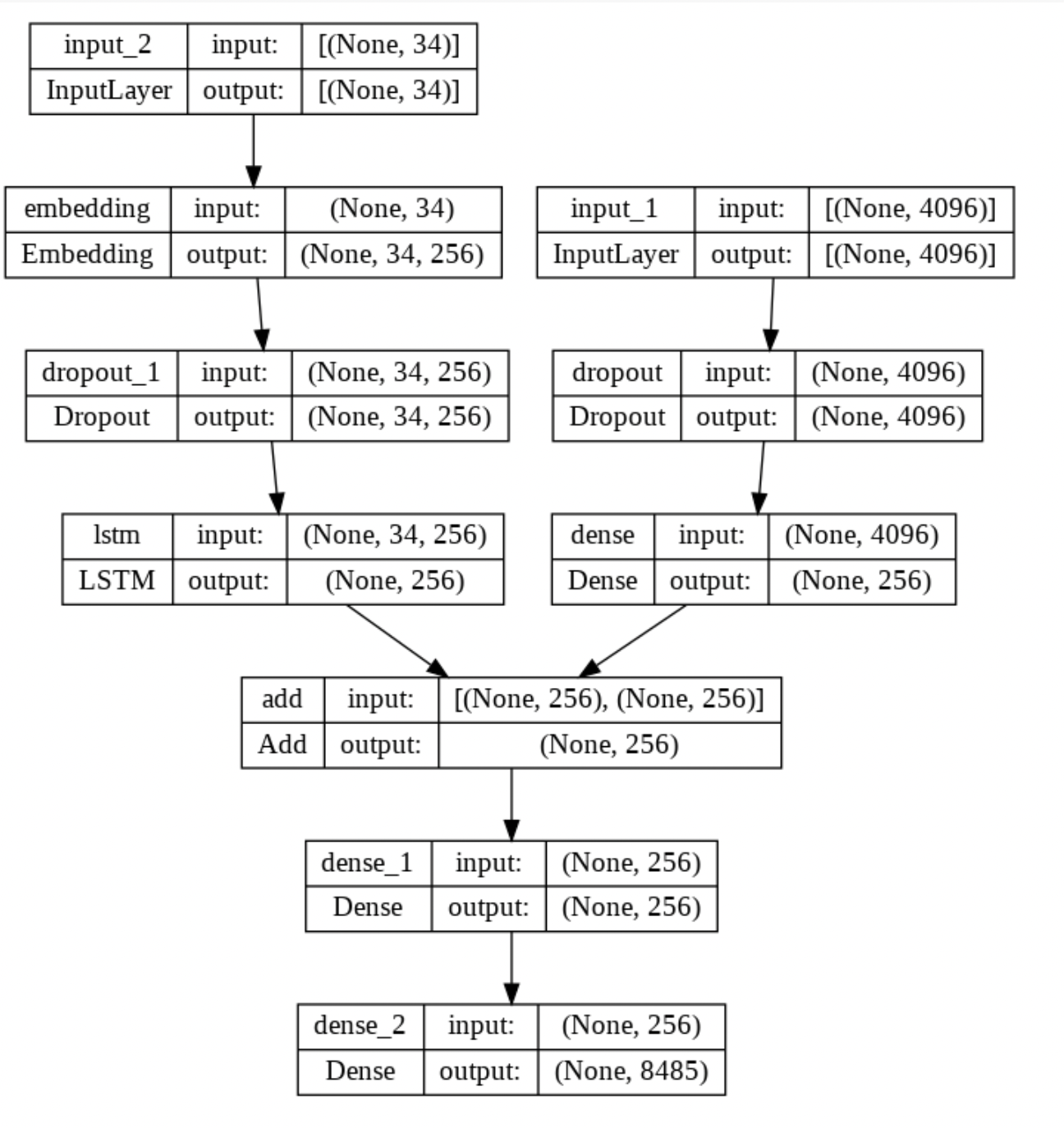
- Defining the CNN-LSTM model

To define the structure of the model, we will be using the Keras Model from Functional API. It will consist of three major parts:

**Feature Extractor** – The feature extracted from the image has a size of 4096, with a dense layer, we will reduce the dimensions to 256 nodes.

**Sequence Processor** – An embedding layer will handle the textual input, followed by the LSTM layer.

**Decoder** – By merging the output from the above two layers, we will process the dense layer to make the final prediction. The final layer will contain the number of nodes equal to our vocabulary size.



##### Figure 4 8k architecture

### 

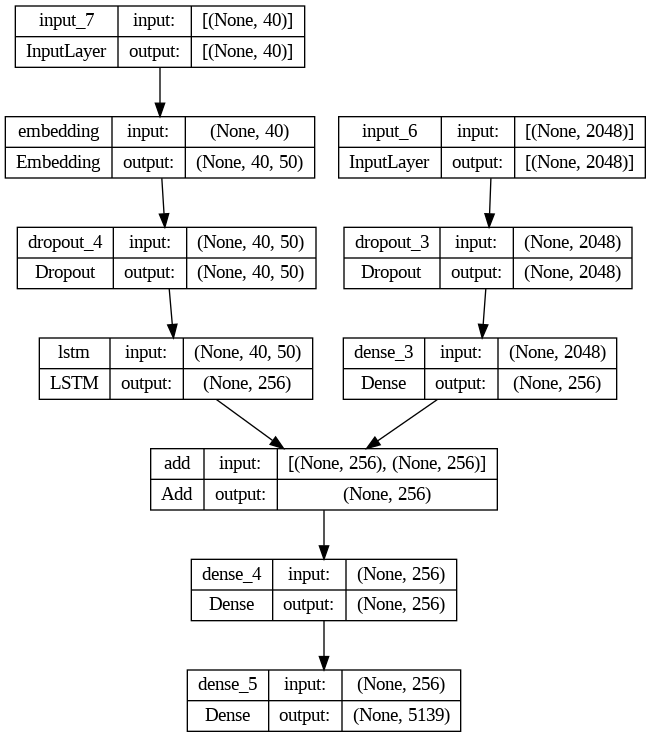


Figure 5 30k architecture

#### 

#### 

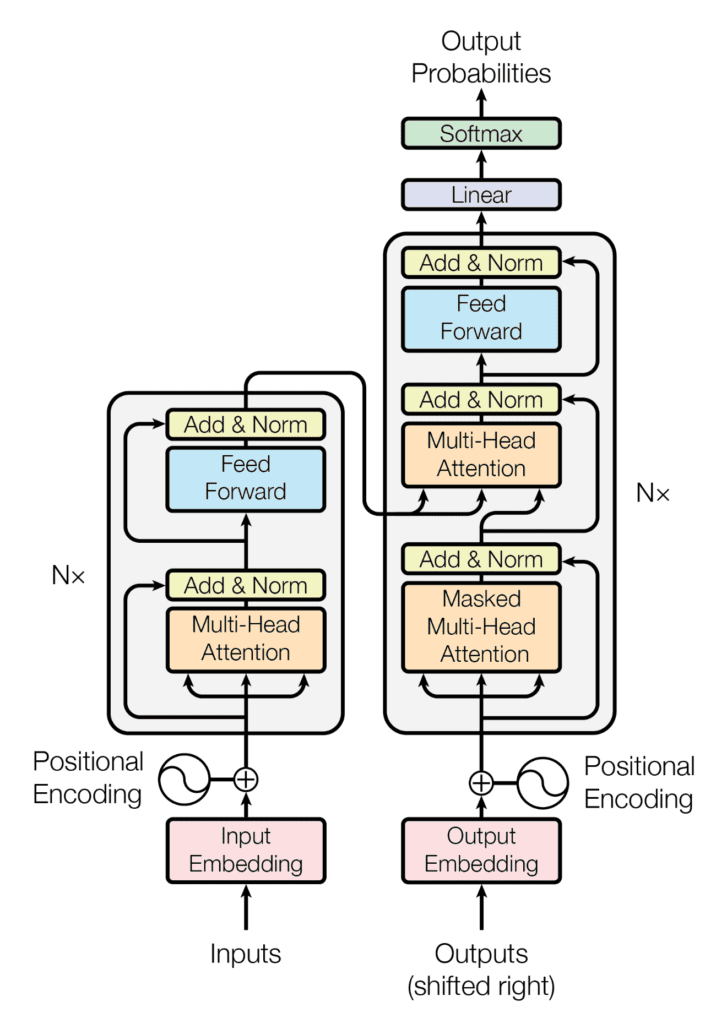
#### 

#### 

#### 

#### Transformer model:

The attention mechanism is a complex cognitive ability that human beings possess. When people receive information, they can consciously ignore some of the main information while ignoring other secondary information. This ability of self-selection is called attention. The attention mechanism allows the neural network to have the ability to focus on its subset of inputs to select specific features.



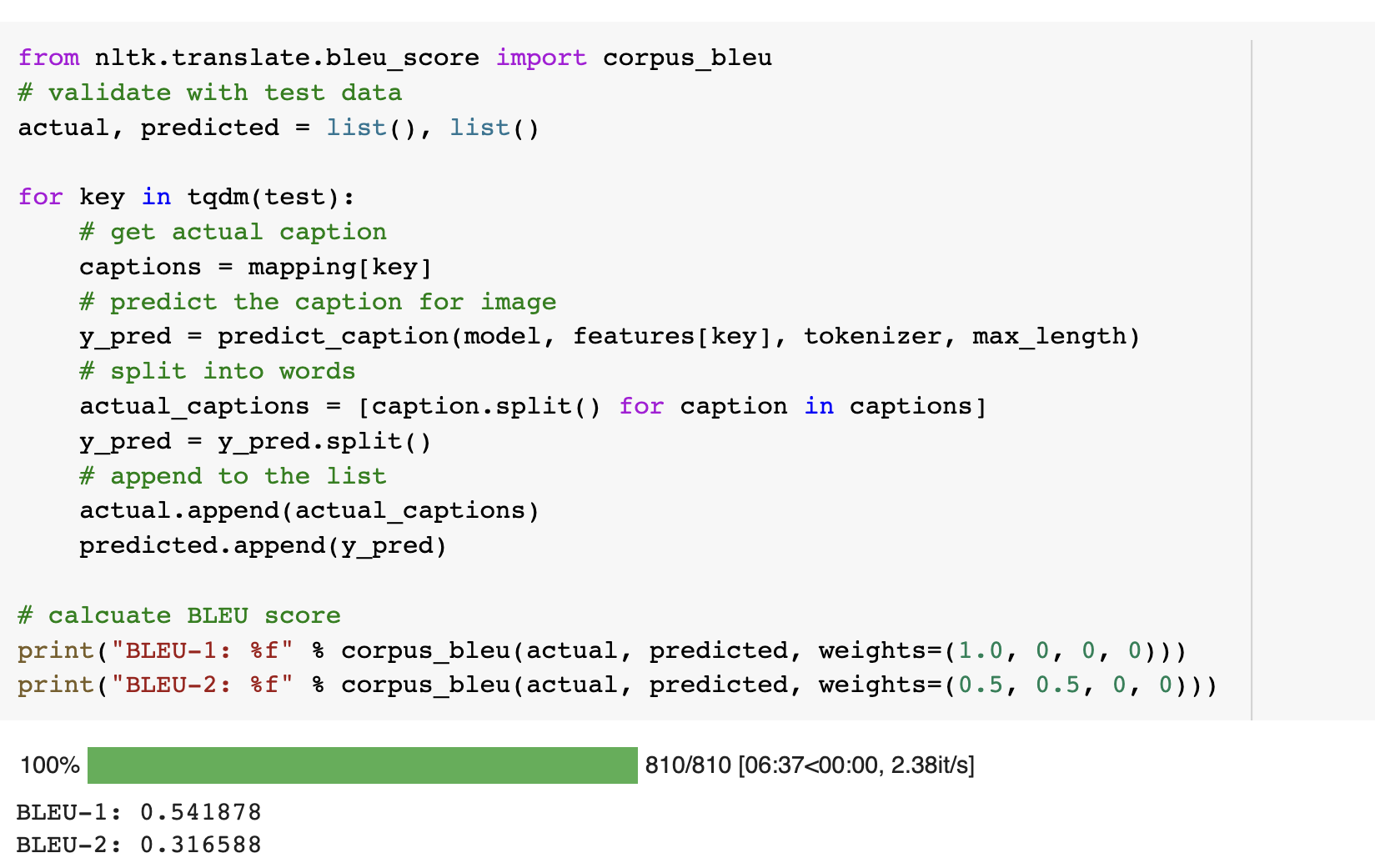
The Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. The Transformer architecture allows for significantly more parallelization and can reach new state of the art results in translation quality. This Transformer model is based solely on attention mechanisms, dispenses with recurrence and convolutions entirely.

### **Train the model with data**

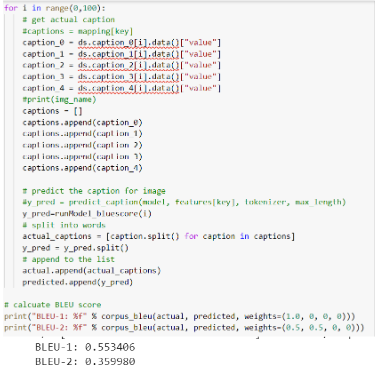
For training the model, the captions for each of the images were processed and one-hot encoding of the words in the five captions given for each image. An indexed list of each unique word was developed. The LSTM model was developed using a one layer architecture having 256 cells.

We will make an indexed list of each unique word present in the sentences. Thus, we will represent every word and symbols in the text data as a unique one hot vector which contains numerical data (1 and 0) as its constituent elements. So, we end up with athree-dimensional tensor that can be fed to the Neural network.

### Evaluation of Test Data



##### Figure 6 Bleu Score with 8k LSTM Model



##### Figure 7 Bleu Score with 30k LSTM Model

# 

# 

# 

# **Deployment**

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

Django helps you write software that is:

* Complete
* Versatile
* Secure
* Scalable
* Maintainable
* Portable

For deploying the model, we developed the frontend form by extending the form class in the template.

{% block content %}

<div align="center">

<form method="post" enctype="multipart/form-data">

{% csrf\_token %}

{{ form.as\_p }}

<button *type*="submit">Upload</button>

</form>

##### Figure 8 Template form for getting data from user

The model was developed in the models.py file, extending the Model class in django architecture. Three fields were defined in the model. Selected method of prediction was defined from a choice of one of three dropdown menus. “lstm 8k” was used as a default value.

*class* UploadImage(models.*Model*):

caption = models.CharField(max\_length=200, null=*True*, blank=*True*)

image = models.ImageField(upload\_to='images')

method= models.CharField(max\_length=20, default="lstm8k")

*def* *\_\_str\_\_*(self):

return *self*.caption

##### Figure 9 Model define for image prediction

The model was developed in the models.py section, where all the data obtained from the form was checked and analyzed. Based on the selected method of image prediction, the uploaded image was processed and one of the three deployed pipelines was used .

*def* image\_request(request):

if request.method == 'POST':

form = UserImageForm(request.POST, request.FILES)

if form.is\_valid():

form.save()

# Getting the current instance object to display in the template

img\_object = form.instance

image\_loaded = "/Users/paritoshkumar/Desktop/python\_training/captioning/caption" + img\_object.image.url

*print*(image\_loaded)

caption=""

if img\_object.method == "lstm8k":

caption = image\_8k(image\_loaded)

elif img\_object.method == "lstm30k":

caption = image\_30k(image\_loaded)

return render(request, 'projects/image\_form.html', {'form': form, 'img\_obj': img\_object, "caption":caption})

else:

form = UserImageForm()

return render(request, 'projects/image\_form.html', {'form': form})

##### 

##### Figure 10 image request function for collecting and processing the loaded images based on the method selected by the user

# 

# 

# 

# 

# 

# 

# **Results:**

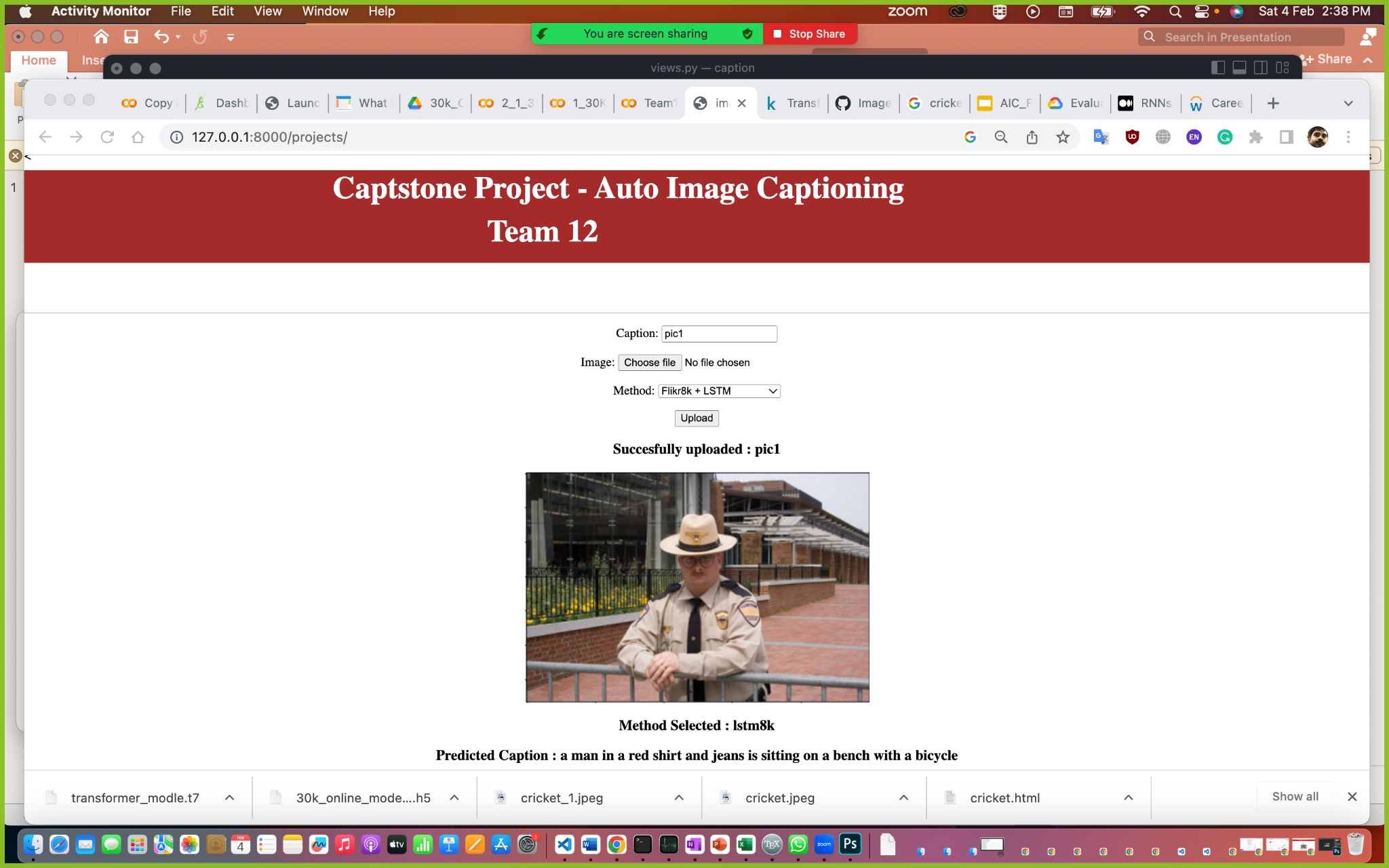
Image was processed and captioning was done on the deployed model for random images downloaded from internet:

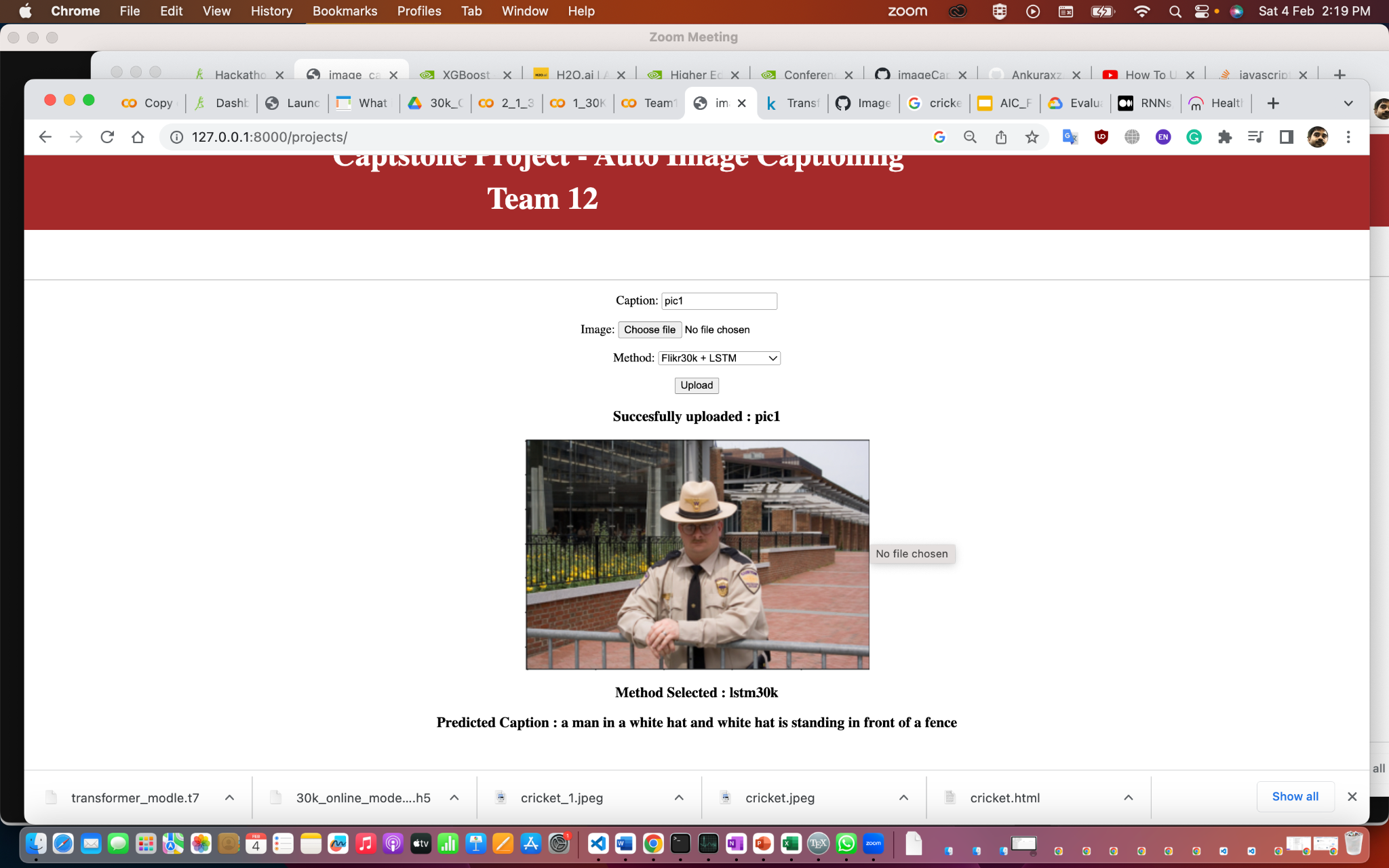


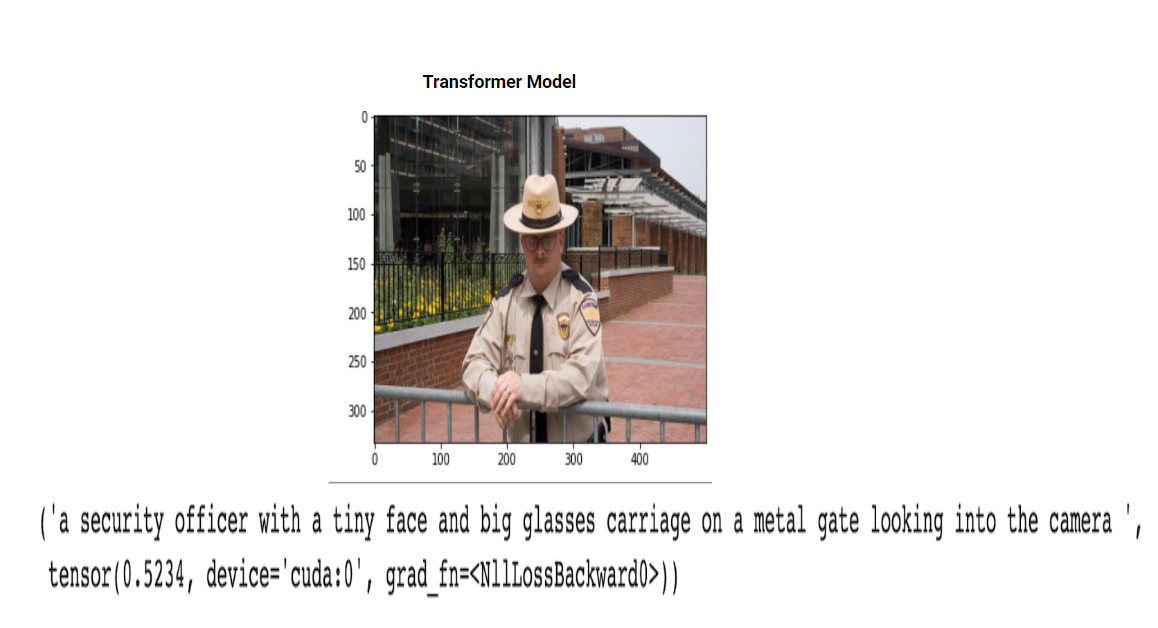
##### Figure 11 Captions generated from the loaded images

* **Caption from 8k trained model**: A man in a red jacket is sitting on a red horse
* **Caption from 30k trained model**: A **woman** in a red hat is riding a horse in the **snow.**

We could generate the captions using all the three models we developed. Earlier we tried with Agrmax for the image predictions but later we generated five captions for each of the images loaded by randomly selecting one word out of five best predictions. Since the LSTM-8k model was trained on ~6000 training samples, the prediction was not the best but the model could identify specific objects in the loaded images. BELU score for 8k trained models was 54%. Models with a training set of ~30k could formulate the sentences better (BELU score ~56%). The Transformer model, though trained with ~6k training sets, worked best for us.







# **Enhanced Phase**

### Deployment using FAST API:

Some of the advantages of FAST API

1. Framework is fast compared to other Python APIs

2. Support for asynchronous code

3. Relatively short development time

4. Easy testing

5. Easy deployment

We have added the below additional features to our model:

### Model can generate 5 best possible captions to the given image.

* + As a part of caption generation, instead of considering the word with maximum probability, consider the one random word from the best probable 3 words. This will generate different captions.
  + Considered only best of 3 because the data set is limited and the diversity is more across the words.

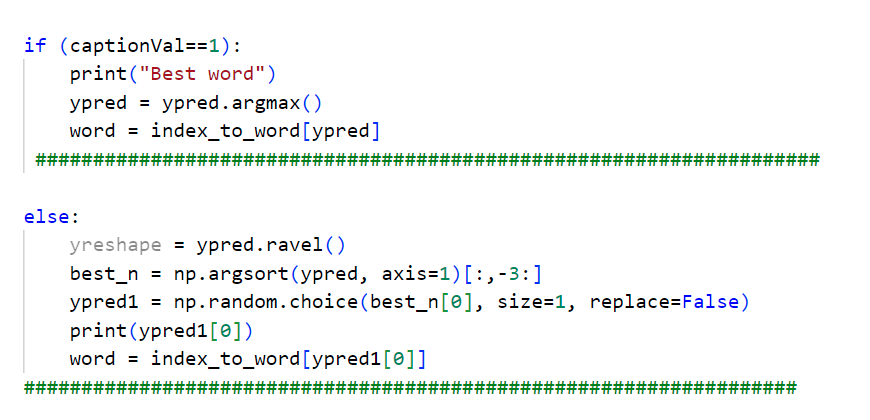
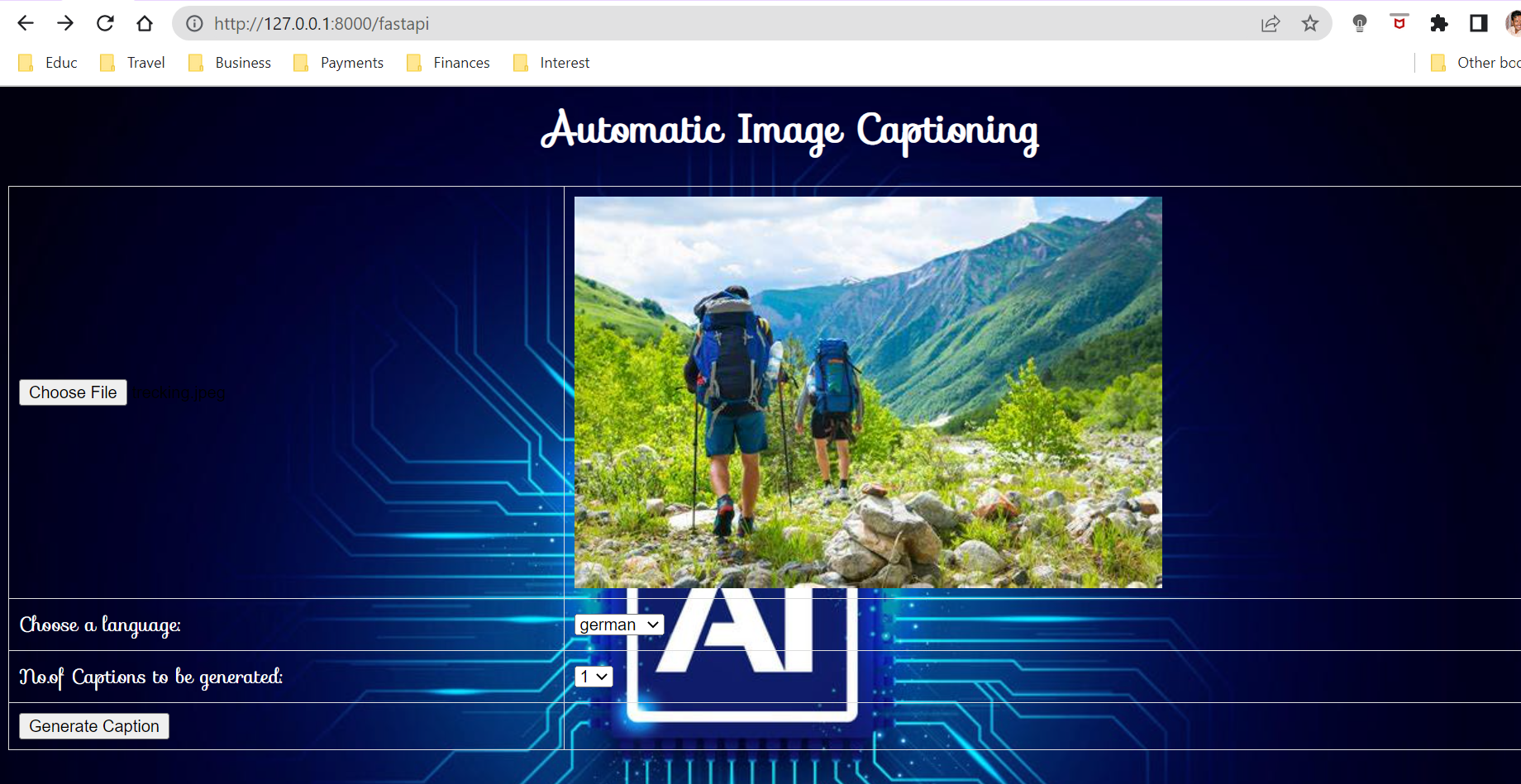
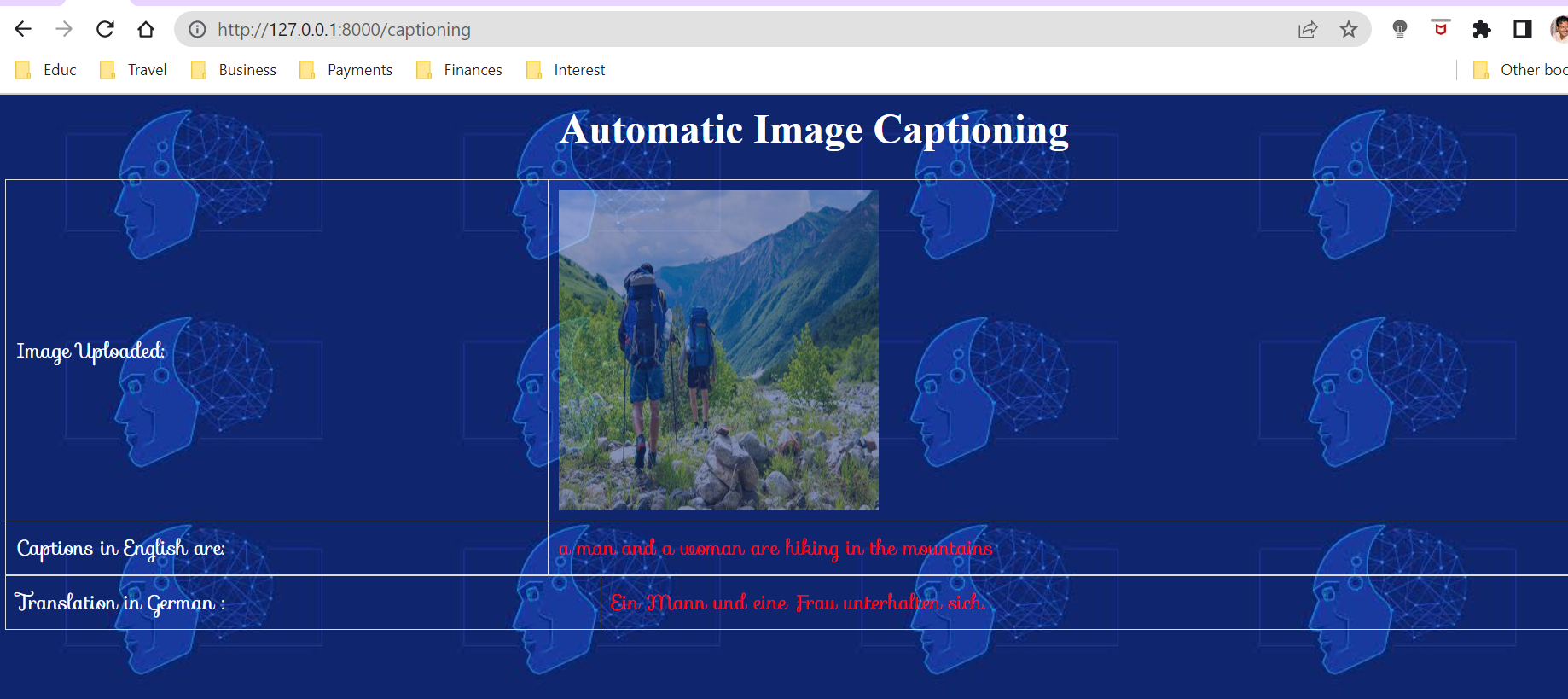
]

Figure 12 Code Snippet for generating multiple captions

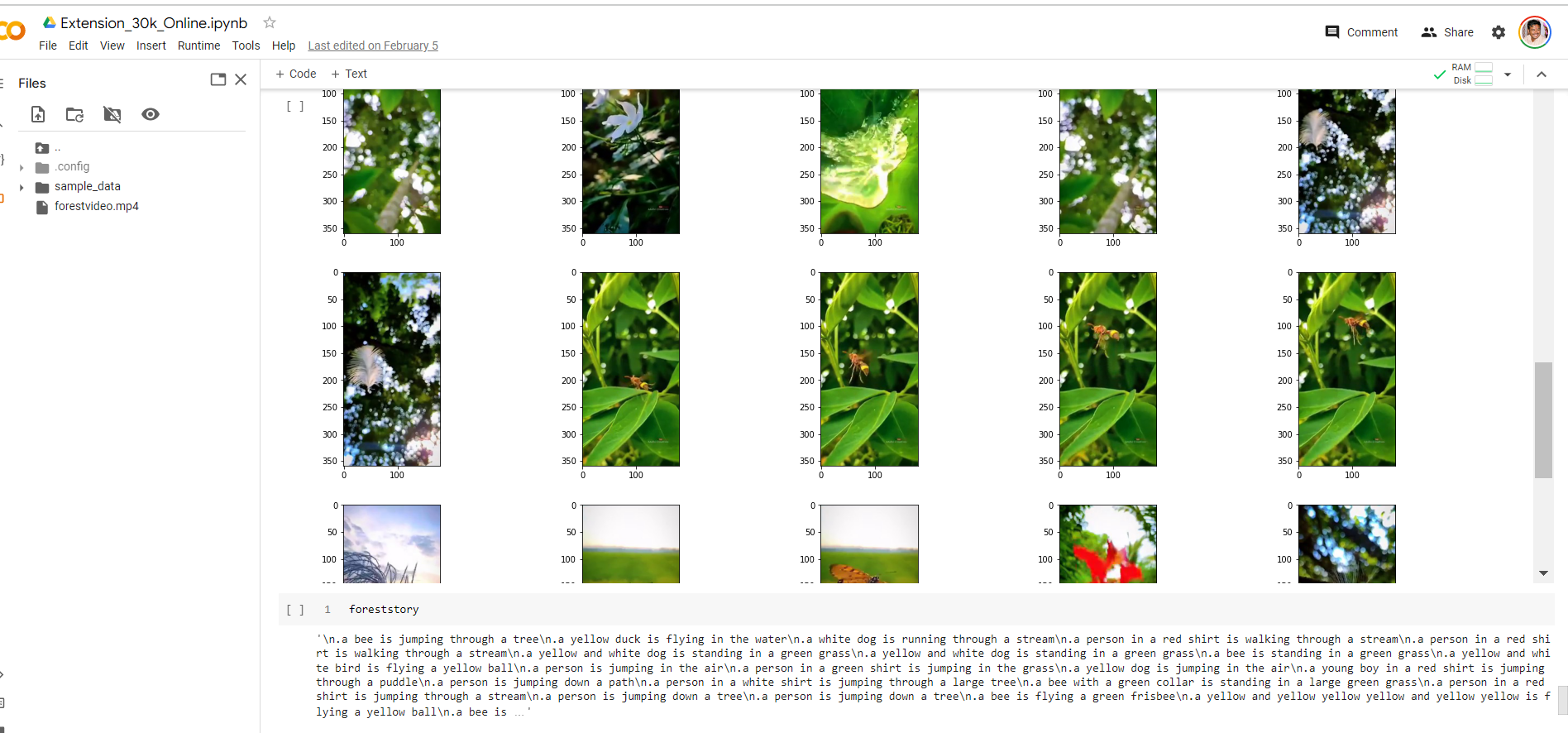
### Translate the caption into another language from english.

* + Using the Python translate API, the captions could be translated into the other languages.





### Describing the uploaded video in the form of a story.

* + Load the video.
  + Slice the video in a specific frequency to get the images.
  + Feed the slices to generate the captions and finally the story is extracted.
  + 

### 

# **Challenges**

Traditional captioning systems will often be more syntactically correct but the appropriate mood of the image is not captured correctly. Yet at times the sentence could make a sense which is not the actual essence of the image.

It is more of robotic sentences rather than the captions conveying the real information of the image.

For Example:



The above image could be captioned as:

Caption 1: Cow grazing grass.

Caption 2: A herd of cows grazing grass on a sunny day.

Caption 2 captures more info and better linguistic representation of image rather than the caption 1.

We need to develop the model in such a way that it not only makes the caption grammatically right but also conveys the appropriate context and detailed captions.

# **Applicability in the real world**

Generating complete and natural image descriptions automatically has large potential effects

* Would serve as a huge help for visually impaired people.
* Image indexing.
* Recommendation in editing applications.
* Explaining what happens in a video, frame by frame.
* Forensics (for detailed analysis of the crime scene)
* Real time commentary of sports
* Captions attached to news images.

# **Future Scope**

1. Train the model with Higher volume data sets like COCO Dataset and generate the captions with more accuracy.
2. Could start working and enhancing the story telling concept further using Visual Storytelling Dataset
3. Work on datasets containing other language captions and train the model to generate captions in other languages.
4. Add the readability ability to the captions generated using text to speech API

# **Conclusions:**

# In this project three different models ( 8k LSTM, 30K LSTM, 30k Transformer models ) for image captioning were developed and deployed on Django/FastAPI.There are also options for selecting the number of captions the model needs to generate and translate the generated caption into other languages. We have also tried to generate a story out of a video by slicing the video frames into images.

# **References:**

1. Aishwarya Maroju , Sneha Sri Doma , Lahari Chandarlapati Department Of Electronics and Computer Engineering J.N.T.U, Hyderabad. Image Caption Generating Deep Learning Model.

<https://www.ijert.org/research/image-caption-generating-deep-learning-model-IJERTV10IS090120.pdf>

1. Simao Herdade, Armin Kappeler, Kofi Boakye, Joao Soares. Yahoo Research. Image Captioning: Transforming Objects into Words.

<https://proceedings.neurips.cc/paper/2019/file/680390c55bbd9ce416d1d69a9ab4760d-Paper.pdf>

1. Ahmed Elhagry, Karima Kadaoui. A Thorough Review on Recent Deep Learning Methodologies for Image Captioning.

<https://arxiv.org/abs/2107.13114>

1. Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan. Show and Tell: A Neural Image Caption Generator.

<https://arxiv.org/abs/1411.4555>

1. X. Li, X. Yin, C. Li, P. Zhang, X. Hu, L. Zhang, L. Wang, H. Hu, L. Dong, F. Wei, et al. **Oscar**: Object-semantics aligned pre-training for vision-language tasks. In European Conference on Computer Vision, pages 121–137. Springer, 2020.
2. X. Hu, X. Yin, K. Lin, L.Wang, L. Zhang, J. Gao, and Z. Liu.Vivo: Visual vocabulary pre-training for novel object captioning, 2021.
3. P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and VQA. CoRR, abs/1707.07998, 2017.
4. N. Li, Z. Chen, and S. Liu. Meta learning for image captioning. Proceedings of the AAAI Conference on Artificial Intelligence, 33:8626–8633, 2019.
5. C. Chen, S. Mu, W. Xiao, Z. Ye, L. Wu, and Q. Ju. Improving image captioning with conditional generative adversarial nets. Proceedings of the AAAI Conference on Artificial Intelligence, 33:8142–8150, 2019
6. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. [pdf] [bib]
7. Sasun Hambardzumyan, Abhinav Tuli, Levon Ghukasyan, Fariz Rahman, Hrant Topchyan, David Isayan, Mark McQuade, Mikayel Harutyunyan, Tatevik Hakobyan, Ivo Stranic, Davit Buniatyan. 2022, Deep Lake: a Lakehouse for Deep Learning. arXiv:2209.10785
8. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. 2015. arXiv:1512.03385