Project Report

CMPE 297 Section 49
Special Topics

Image Caption Generator

Team Members:

Jithesh KB
Poojashree NS
Priyanka Math
Tharun Mukka

Table of contents

Abstract

Using any natural language words to automatically generate an image's caption or description is a very difficult undertaking. The goal of our project is to put into practice an image caption generator that responds to user input to produce captions for a picture. To translate the comprehension of the image into words in the appropriate sequence, it needs both computer vision techniques to comprehend the image's content and a language model from the field of natural language processing. Due to the mutual exclusivity of images and text sequences, this job requires the use of two interconnected models, one of which is dedicated to image encoding and the other to text decoding. As part of this project, first a language model named Roberta will be trained and fine tuned on captions and then a captioning model will be initialized and trained using Vision Encoder Decoder module of hugging face library.

Introduction

Computer vision in the field of image processing has significantly advanced in recent years, including image classification and object recognition. The issue of image captioning, which involves automatically generating one or more phrases to comprehend an image's visual information, has benefited from advancements in image categorization and object detection. Large potential consequences of automatically creating detailed and natural image descriptions include news image titles, medical image descriptions, text-based image retrieval, information accessed by blind users, and human-robot interaction. These captioning-related applications have significant theoretical and real-world research significance. In the age of artificial intelligence, image captioning is a trickier but important work. An image captioning model should produce a semantic description of a new image when given one.

Related Work

To extract picture grid level information, researchers first used a pre-trained convolutional neural network (CNN) as an encoder and a recurrent neural network (RNN) as a decoder. Faster R-CNN was initially used by Anderson et al. 2018 [1] to extract region-level characteristics. The majority of following research adopted this approach because to its clear benefit, while grid-level characteristics retrieved by CNN were disregarded. However, there are still a few fundamental flaws in the object detector's region-level features and encoder: The following factors make it difficult to train an image captioning model end-to-end from image pixels to descriptions: 1) region-level features may not cover the entire image, resulting in the lack of fine-grained information; 2) extracting region features is time-consuming, and the object detector requires an additional Visual Genome dataset for pre-training. These factors also restrict potential applications in the actual scene.

The usual and prevalent method over the past several years has continued to be LSTM decoder with a soft attention mechanism. However, the limitations of LSTM's expressive capability and training efficiency also restrict the impact of pertinent models. Many researchers started integrating the Multi-head Self-Attention (MSA) mechanism into the decoder of LSTM or directly adopted Transformer architecture as the decoder of image captioning models as a result of the success of the Multi-head Self-Attention (MSA) mechanism and Transformer architecture in NLP tasks.

Transformer architecture, which offers a novel option for encoding pictures into vector characteristics, particularly progressively demonstrates amazing promise in computer vision tasks and multi-modal activities. Grid-level features, which offer a better processing efficiency and enable swiftly exploring more efficient and sophisticated designs for image captioning, are features retrieved by a visual transformer as opposed to Faster R-CNN.

Data

Flickr8k Dataset, a benchmark collection for sentence-based picture description and search, is the dataset utilized in this project. It consists of 8,000 images that are each matched with five different captions that clearly describe the key items and events. The photographs were painstakingly chosen to represent a diversity of scenarios and circumstances from six different Flickr groups, and they often don't feature any famous individuals or places.

Exploratory Data Analysis and Visualization

The below image is a result of data preprocessing.

Figure 1. Dataset image with five different captions. Source: Author

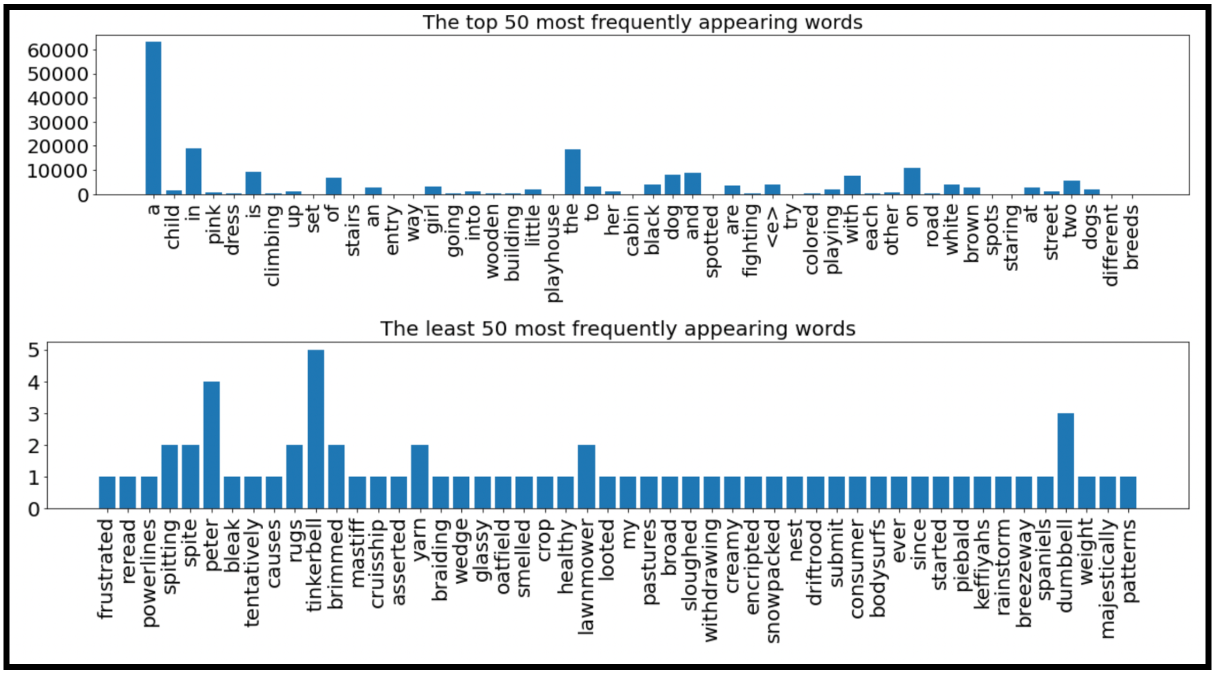

Figure 2. EDA and visualization of Flickr 8k dataset. Source: author

Methodology

The task of captioning an image can be thought of as an end-to-end sequence-to-sequence embedding task, where the input sequences are the image pixels and the output is a caption that describes the image. Due to the mutual exclusivity of image and text sequences, this task requires the use of two interconnected models, one of which is dedicated to image encoding and the other to text decoding. This concept is equivalent to conventional encoders and decoders, which are used to handle the processing and transmission of electrical signals.

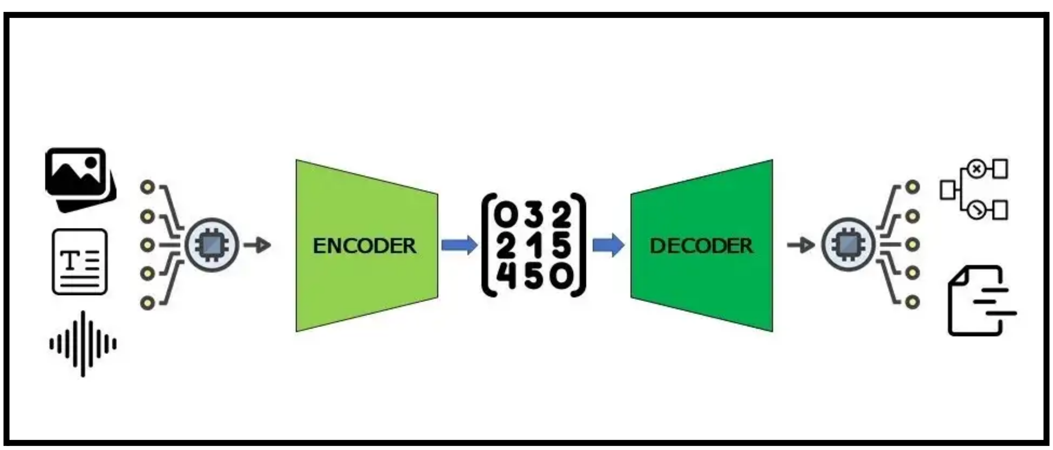

Figure 3. Encoder-Decoder architecture. Source:
https://paperswithcode.com/method/vision-transformer

The basic idea of an encoder-decoder architecture involves transforming a signal using an encoder and then translating it back into the desired form using a decoder. State-of-the-Art Transformers like BERT, Roberta, and GPT2 are used for state-of-the-art Sequence to Sequence learning tasks like language translation and text summarization. The Encoder is essentially a transformer model that encodes the information in the text in a mathematical-tensor using Self-Attention heads that uses Key, Value, and Query to solve the problem of traditional RNN's & RNN architecture.

Using a Cross-Attention layer, this mathematical tensor from the encoder is coupled to the layers of the decoder (which helps in encoding the previously generated output ex. in machine translation every next token is chosen using the previous tokens generated and the Encoder tensor) The Decoder tensors are mapped to the Encoder tensors with the aid of this Cross-Attention layer, which also creates intermediate embedding states that aid in producing the output.

The main difference between image captioning and other Sequence to Sequence tasks is that, instead of translating from one language to another, we translate from an image to a language. which, theoretically speaking, is an identical task because both are just three-dimensional vectors.

An idea that every image can be read as a chunk of images as tokens, similar to words in a sentence, was recently flashed in the paper [2] "An Image is Worth 16x16 Words" on Vision Transformers. This study suggests that any CV job can employ this 3D image tensor token input. This overcomes CNN's problem of limited association because the fundamental principle of CNN is to connect just local pixels using a kernel which convolutes the entire image.

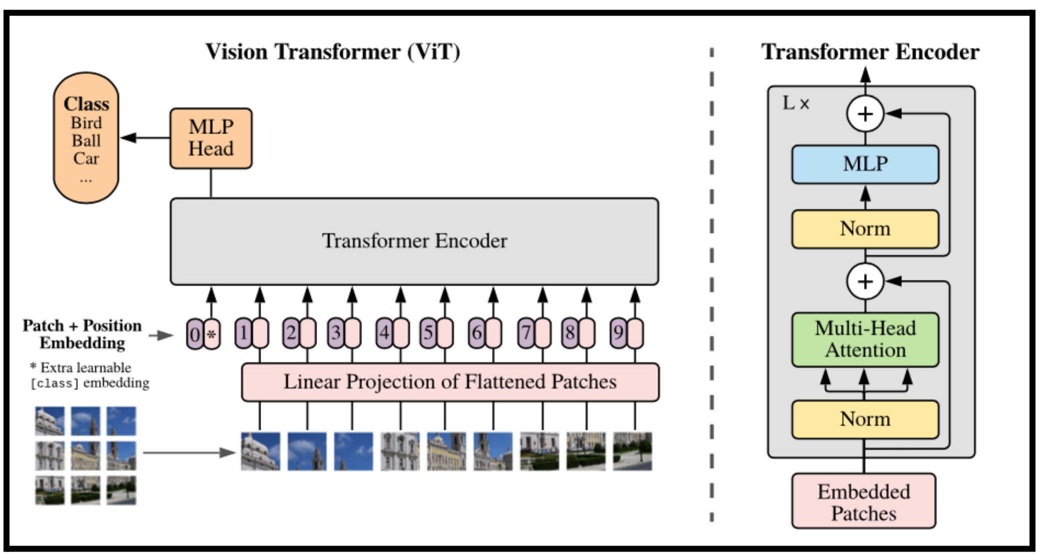

Figure 4. ViT (Vision Transformer). Source: https://paperswithcode.com/method/vision-transformer

In light of the aforementioned two concepts, the task of image captioning can be approached as Encoder & Decoder where, an image serves as the decoder's input and can be processed by a vision transformer (ViT) and the output from the decoder is text that describes the objects in the image. This task can be completed by Roberta, BERT, or any other cutting-edge language model.

Firstly, a language model that has been fine-tuned on a text corpus will enable the decoder to pick up new words (if any) and produce brief captions. In order to create better sentence context, this will fine-tune the transformer's self-attention weights. Additionally, because the self-attention layers would already be primed to have better sentence (caption) representation, the optimizer could concentrate on fine-tuning the cross-attention layers while training the captioning model, saving training time and complexity on the actual task of captioning.

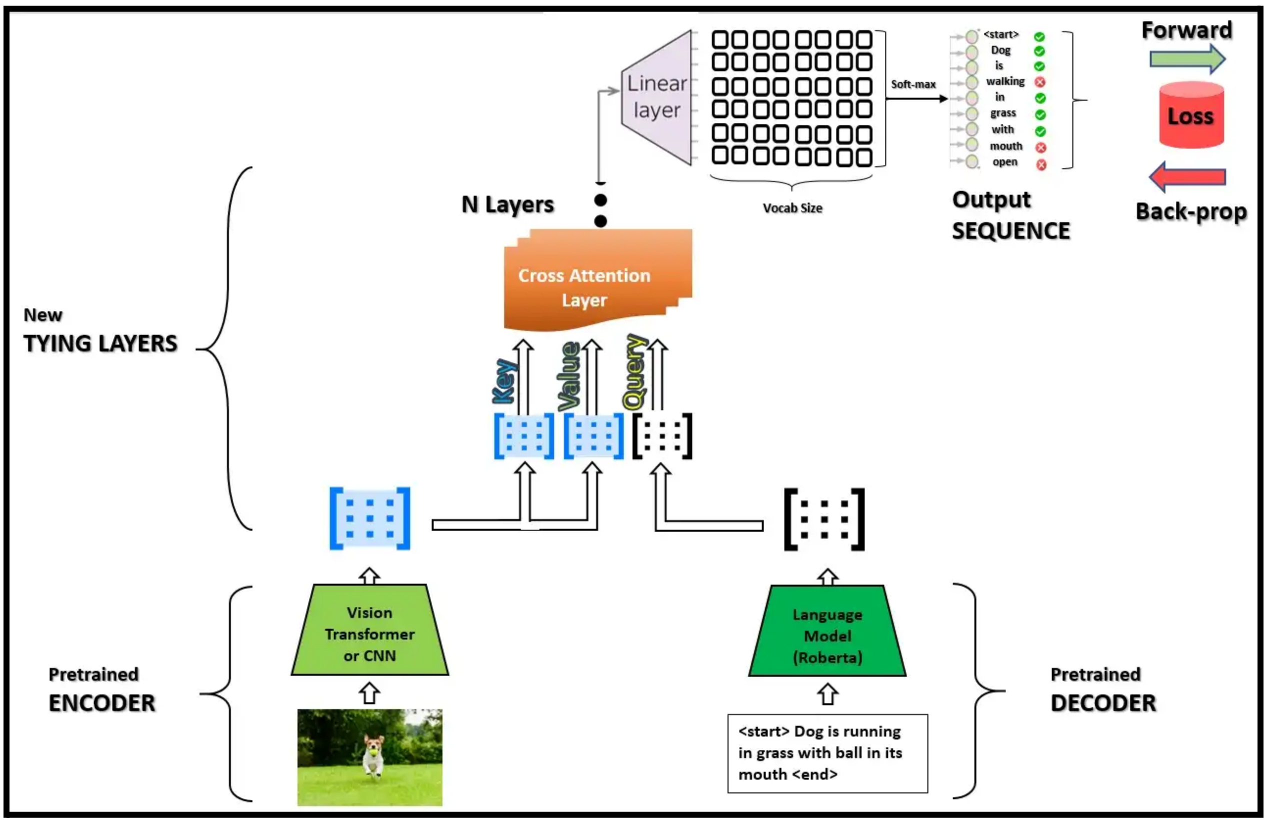

Figure 5. Vision Encoder Decoder Architecture. Source: https://paperswithcode.com/method/vision-transformer

When pre-trained models (in the example above, Vision Transformer and Roberta) initialize the vision encoder decoder, they build an instance of an image encoder and a language decoder and link their embeddings via a cross attention layer. In the cross-attention head, the encoder embeddings are employed as KEY & VALUE and the decoder embeddings as QUERY.

The model is fed both the image and the intended caption during training. This is an illustration of teacher-forcing instruction. By forcing the model to produce the same caption using these inputs, it is possible to learn how the words in the captions relate to the objects in the input images. The vector of size Length of sequence X Vocabulary size is the output of the linear layer (LM head). The elements of a vector matrix (after SoftMax) are probabilities of the word occurring at a given position in the sequence. Every time a training step is completed, the model predicts a word sequence that is compared to the real caption and the loss is sent back into the learning process.

Performance evaluation

The number of matching n-grams between the sequence produced by the model and the required sequence is measured using ROUGE (Recall-Oriented Understudy for Gisting Evaluation) and perplexity is used as a metric for measuring a language model's performance.

We conducted experiments with different values for Epoch, batch size, learning rate, etc and the resultant metrics are compared using MLFlow as shown below.

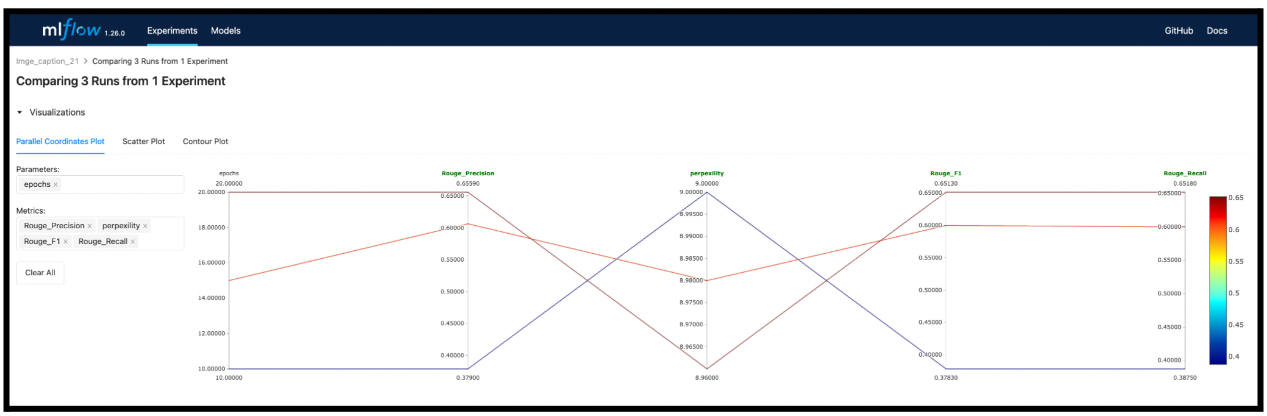

Figure 6: Comparison of experiments on trained models. Source: Author

MLOps

CI/CD pipeline for this project is generated using MLflow, DagsHub and Huggingface Spaces. MLflow is an open source platform for managing the end-to-end machine learning lifecycle. Dagshub integrates with github and MLflow and helps us in versioning datasets & models, track experiments, label data, and visualize results and Huggingface Spaces is used to deploy the application.

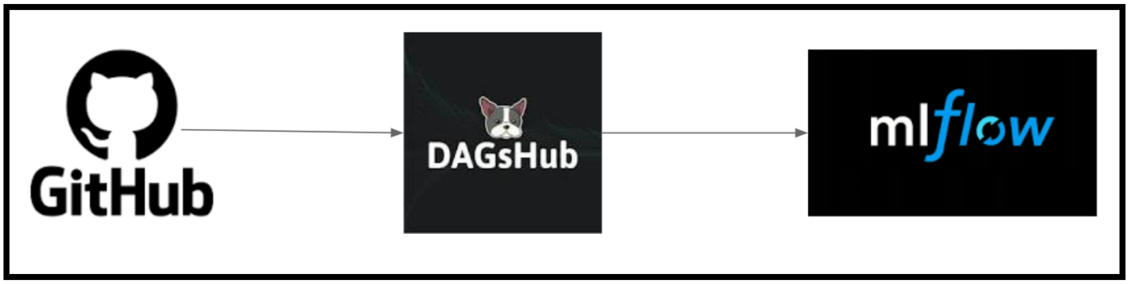

Figure 7: MLOps pipeline for Image Caption Generator. Source: Author

As a crucial component of the ML project, the GitHub project repository is linked to DagsHub utilizing GitHub Actions for CI/CD.

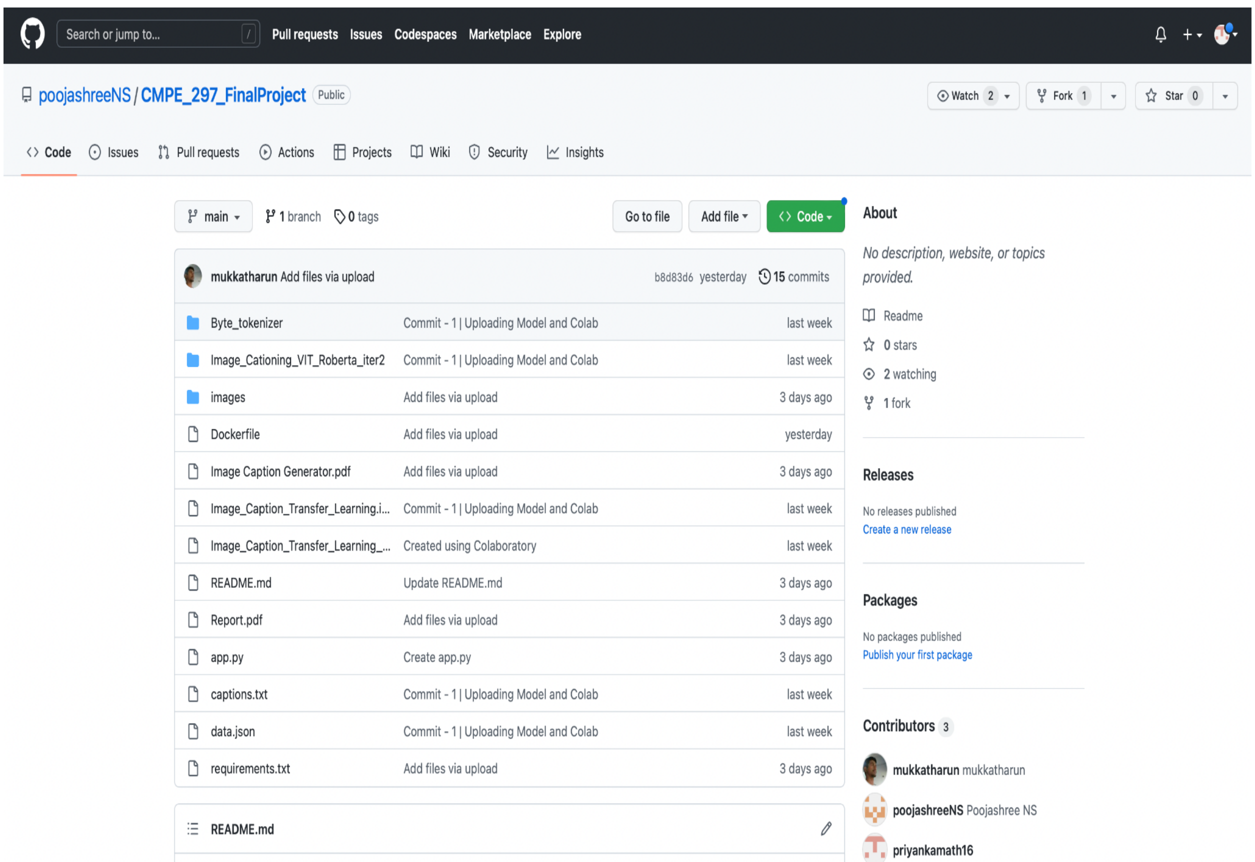

Figure 8: Project GitHub Repository Source: Author

The workflow for DagsHub's GitHub-connected repositories is much more efficient. Additionally to automatically syncing the repository on push and enabling management of PR and Issues from both platforms with the aid of GitHub Actions, DagsHub also syncs the git-tracked files.

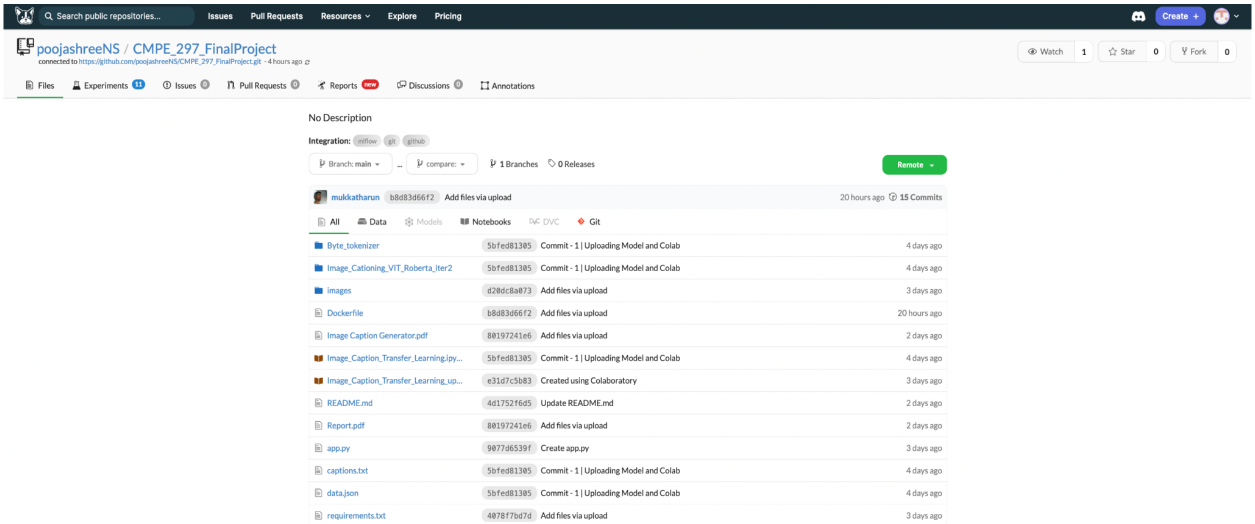

Figure 9: DagsHub workspace that is linked to the project github repository. Source: Author

When code is pushed, the GitHub connected repositories that have subscribed to the GitHub webhooks are automatically synchronized. The project is now completely configured with a remote object storage and experiment tracking server, while git-tracked files continue to reside on GitHub.

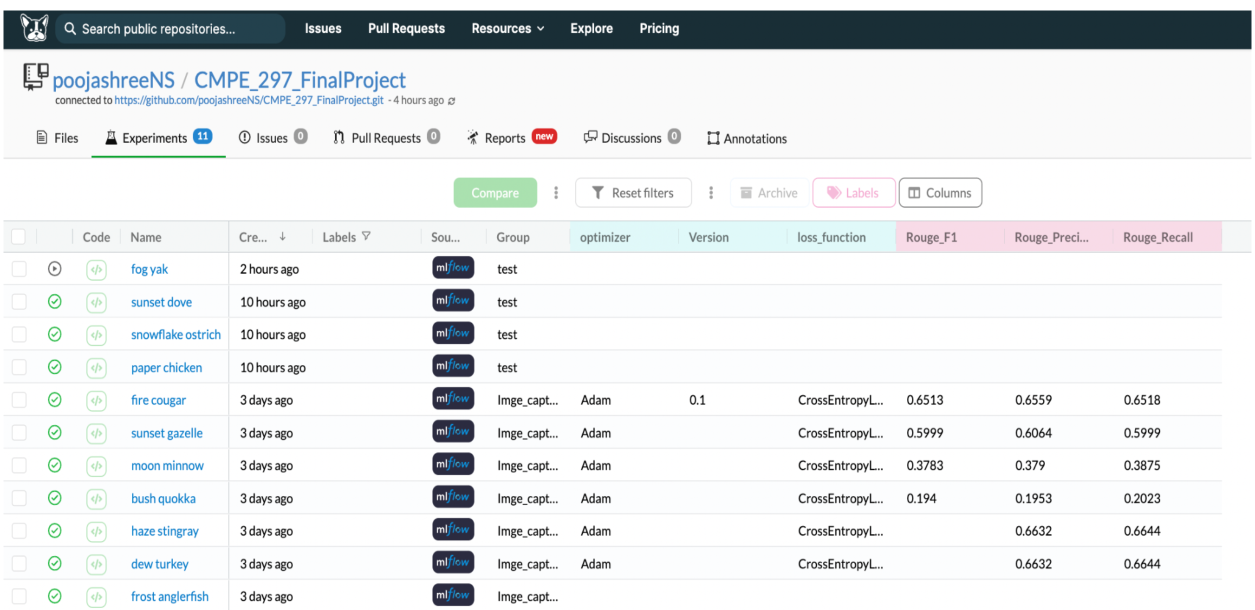

Figure 10: Pipeline of the model training. Source: Author

Four elements are offered by MLflow to assist in managing the ML workflow:

- MLflow Tracking
- MLflow Projects
- MLflow Models
- MLflow Registry

A new experiment is established each time a pipeline is triggered, tracking all the parameters, metrics, datasets, and model artifacts. The figures that depict the experiments and the information they track are provided below.

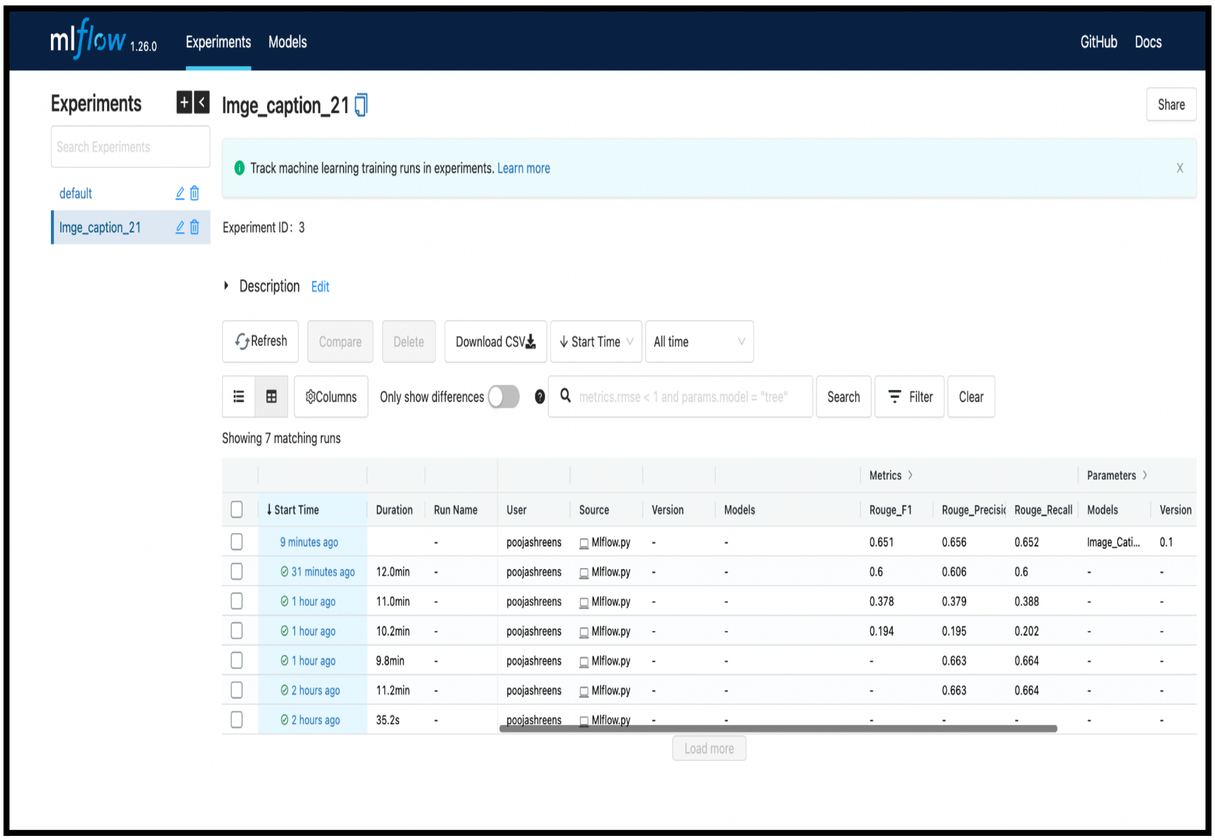

Figure 11: MLflow dashboard. Source: Author

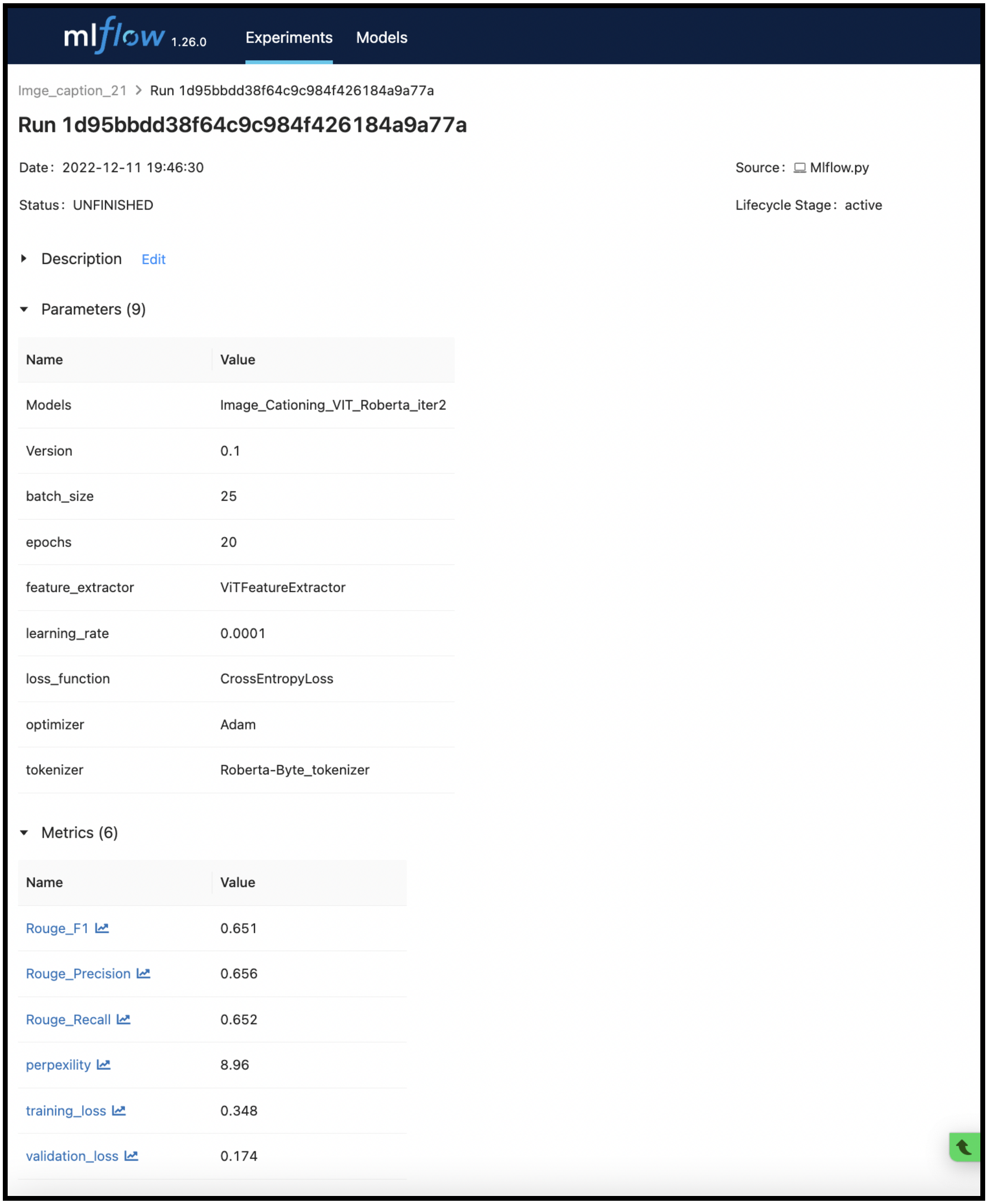

Figure 12: MLflow experimental model parameters and metrics. Source: Author

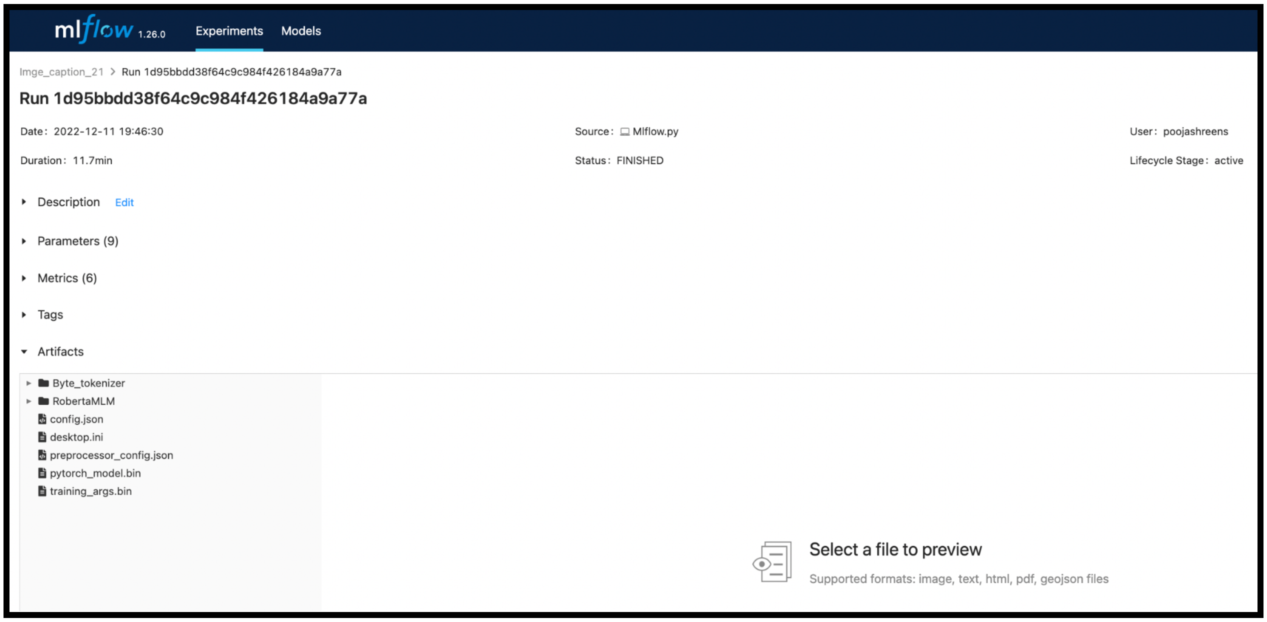

Figure 13: MLflow model artifacts. Source: Author

We conducted experiments with different values for Epoch, batch size, learning rate, etc and the resultant metrics are compared using MLFlow as shown below.

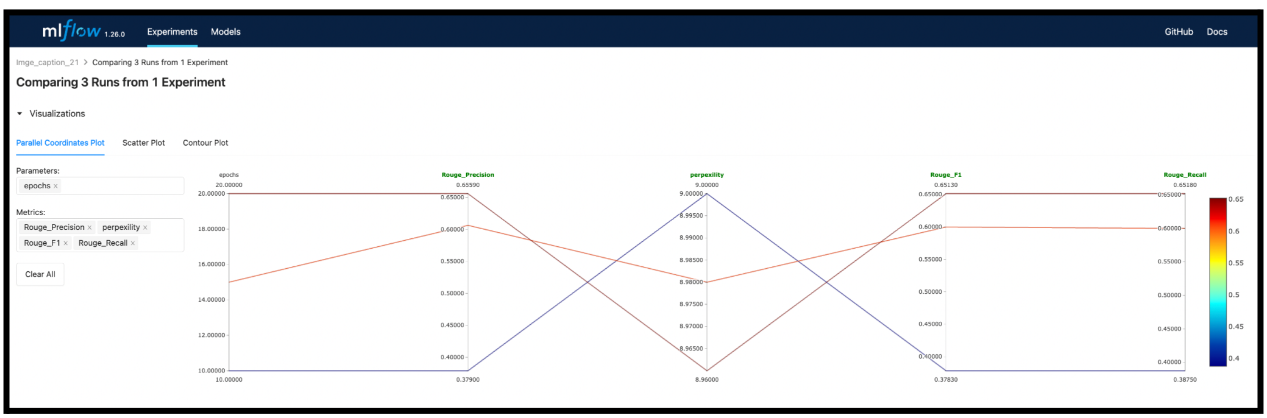

Figure 14: Comparison of experiments on trained models. Source: Author

One stage may at any time be allocated to each unique model version. For typical use-cases like Staging, Production, or Archived, MLflow offers predefined stages. A model version can be moved from one stage to another.

Figure 15: MLflow Model registry. Source: Author

The code is dockerized and deployed on hugging-face spaces once the best model has been chosen.

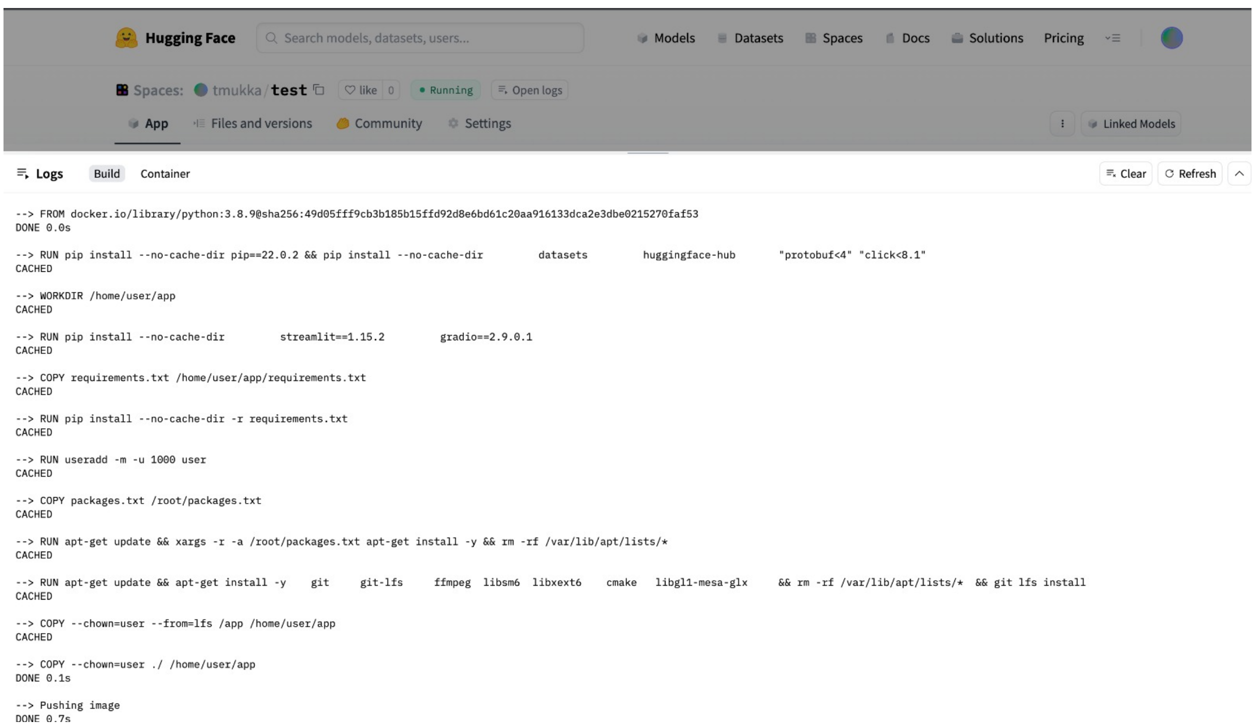

Figure 16: Deployment of dockerized image on hugging face spaces. Source: Author

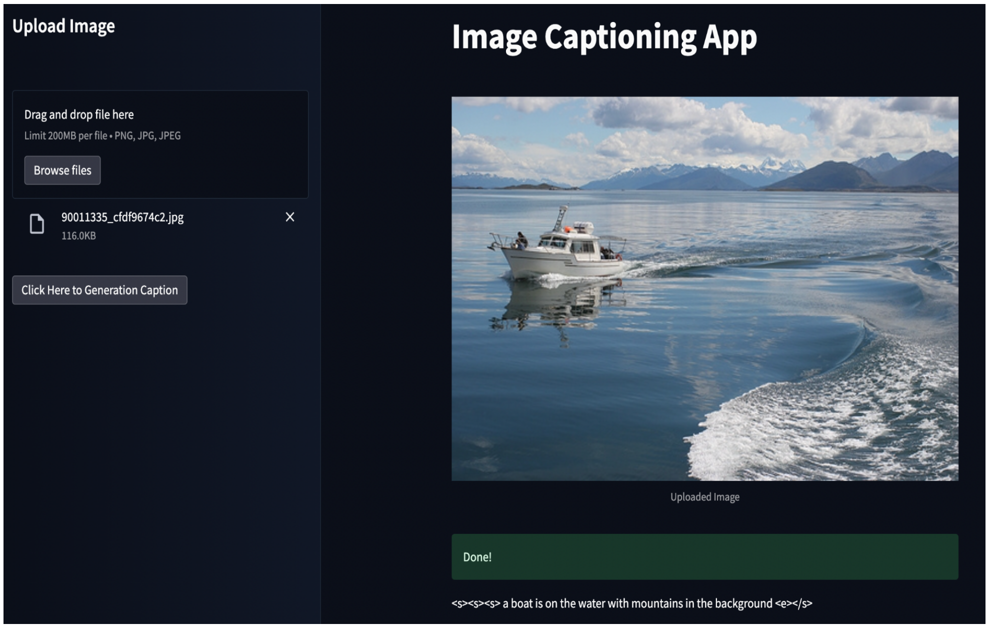

Figure 17: Deployed web application for image caption generator. Source: Author

Conclusion

As part of this project, first a language model named Roberta is trained and fine tuned on captions and then a captioning model is initialized and trained using the Vision Encoder Decoder module of hugging face library. MLOps pipelines were created using MLflow, to control the entire machine learning lifecycle and DagsHub to track trials, label data, visualize outcomes, and update datasets.

References

1. Anderson, P.; He, X.; Buehler, C.; Teney, D.; Johnson, M.; Gould, S.; and Zhang, L.. Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. In Proceedings of the CVPR, 6077–6086, 2018.

2. Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. Conference paper at ICLR 2021

3. https://huggingface.co/docs/transformers/model_doc/vit