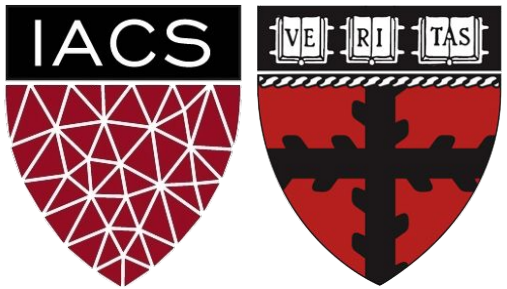
Caption This - Group BKKST

Advanced Practical Data Science, MLOps

AC215

Team Members: Al-Muataz Khalil, Ed Bayes,
Stephen Knapp, Matthew Stewart, Shih-Yi Tseng



Outline

- Project Scope
- Project Workflow
- Process Flow
- Data
- Models
- App design
- Deployment: Live Demo

Problem Definition: Image Captioning

The World Health Organization (WHO) estimated that 314 million people have visual impairment across the world, including 269 million who have low vision, and 45 million who are blind (Ono et al 2010).

Many people with visual impairments rely on screen readers in order to access the internet through audio, and thus depend on image captions (Yesilada et al 2004).

Therefore, accessibility, as well as automatic indexing and other goals, make accurate image captioning an important priority (Hossain et al 2018).

Proposed Solution

We have designed, built, and deployed at-scale an application which receives an image of an everyday activity which then **assigns a caption of the image contents**, based on state-of-art computer vision and natural language models

Project Scope

Proof Of Concept (POC)

- Set up CI/CD pipeline
- Store Flickr8k and COCO datasets in GCP bucket
- Conduct image feature extraction and EDA
- Test baseline model (InceptionV3 net + RNN)
- Improve model architecture (CLIP + transformer) and training with full dataset
- Verify models predict labels for unseen photos

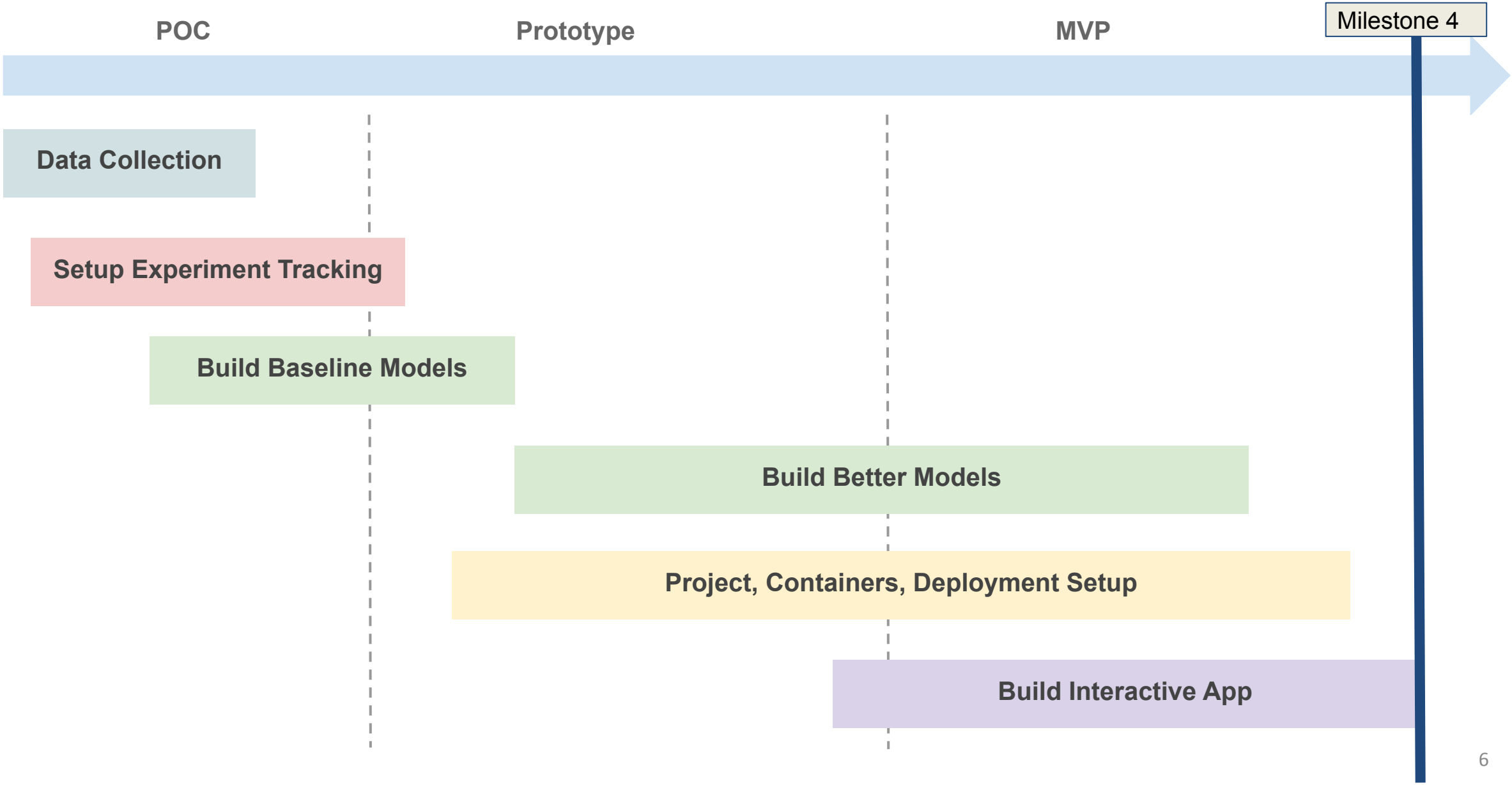
Prototype

- Create ‘looks like’ mockup of UX using figma
- Deploy one model to Fast API to service model predictions as an API

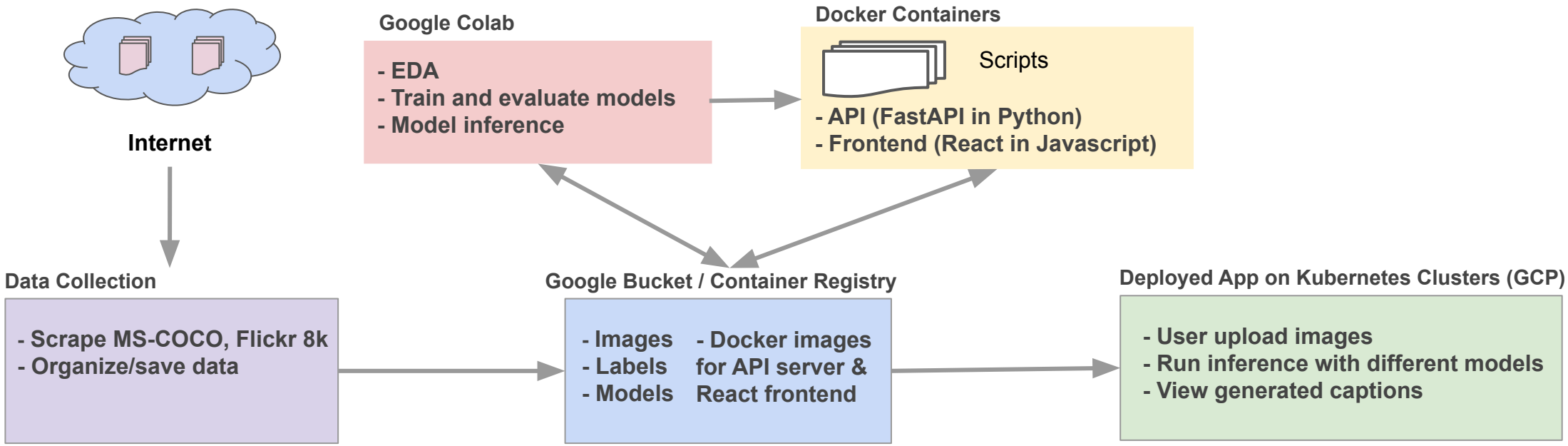
Minimum Viable Product (MVP)

- Create App that labels unseen photos
- API Server for uploading images and predicting using best model
- Deploy with Kubernetes on GCP

Project Workflow



Process Flow



Data

- Public Google bucket (link here) containing MS COCO and Flickr 8K datasets used during this project
- Flickr 8K: 8,091 images from one of six categories, each with 5 corresponding image captions (total label count:40455)
- MS COCO (2014): 164K images split into training (83K), validation (41K) and test (41K) sets, with total 616K labels
- Both datasets are standardized datasets used for benchmarking and released under CC0 license (public domain).

Data Example (Flickr 8K)

A brown dog in two black collars running through a grassy field .

a small brown and black dog lying down in a furry rug .

A man on the street standing by his bicycle .

Friends and family dance on a beach by their vehicles .

A man feels on top of the world on top of a large rock formation .

A dog leaps into the air to catch a ball in its mouth .

A dog leaps into the air to catch a ball in its mouth .

Some children watching fish in a pool .

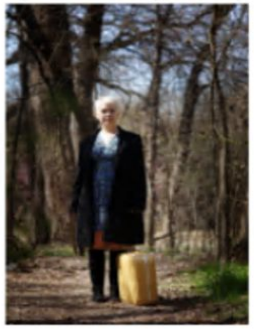
Two gray dogs jump at each other over the tall grass .

Data Example (MS COCO)

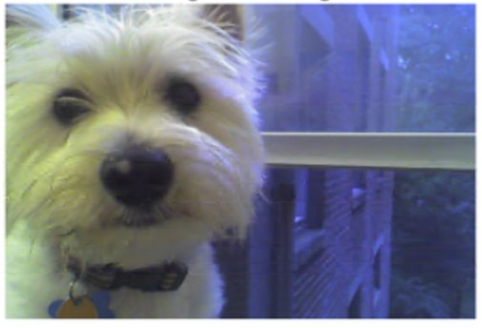
An antique style stove in a sparsely furnished room full of pots.

Two horses are grazing in a grassy valley.

A woman standing next to a yellow piece of luggage.



The small white dog is standing near a window.

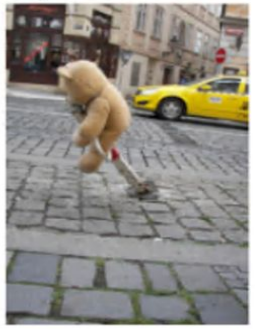


A small collection of four clear empty candlesticks.

A woman making food on the side of the road.

Various items such as sunglasses, keys, and batteries on a table

a stuffed teddy bear sits on a crooked pole



A round clock in a tan brick clock tower.

Baseline Model

RNN-based model with attention from Tensorflow Core tutorial:

https://www.tensorflow.org/tutorials/text/image_captioning

Model:

- CNN (InceptionV3) for image feature extraction
- RNN decoder that attends to image feature to predict the next word
  - Additive attention on RNN hidden state and image feature for generating context vector

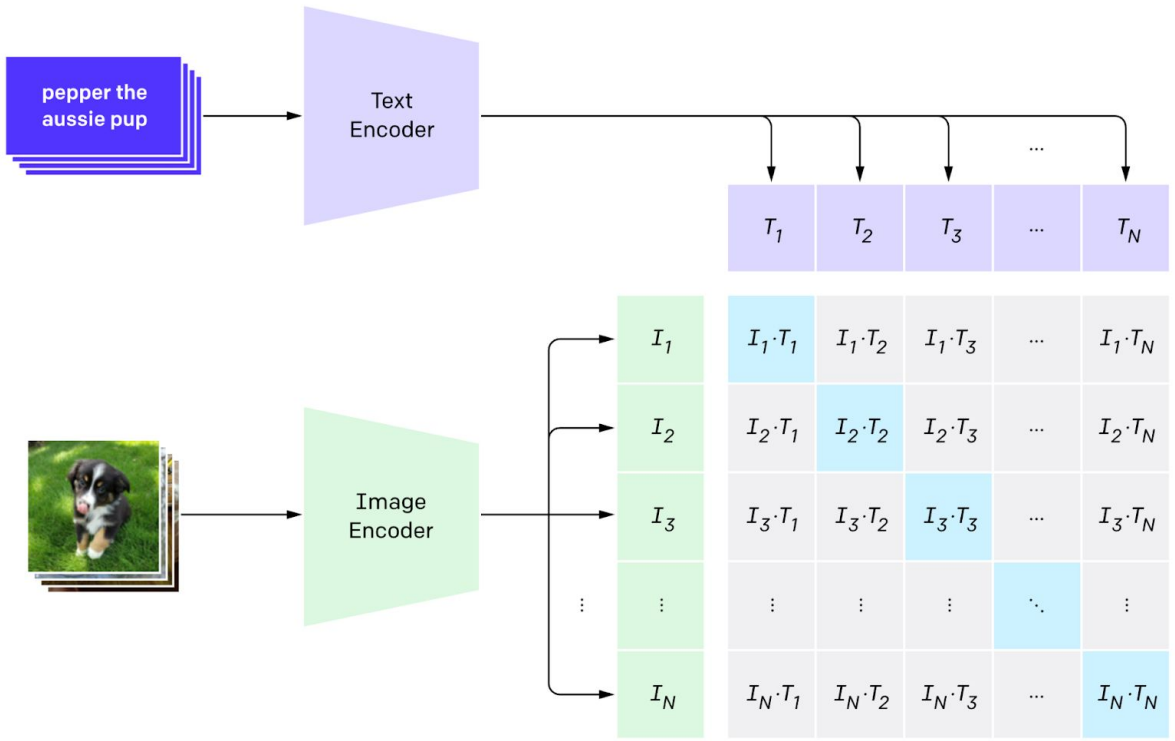
Our Models: Transformer-based Caption Models

- Transformer-based models
  - Model 1: standard encoder-decoder architecture
  - Model 2: prefix language model

- CLIP for image feature extraction

CLIP for Image Feature Extraction

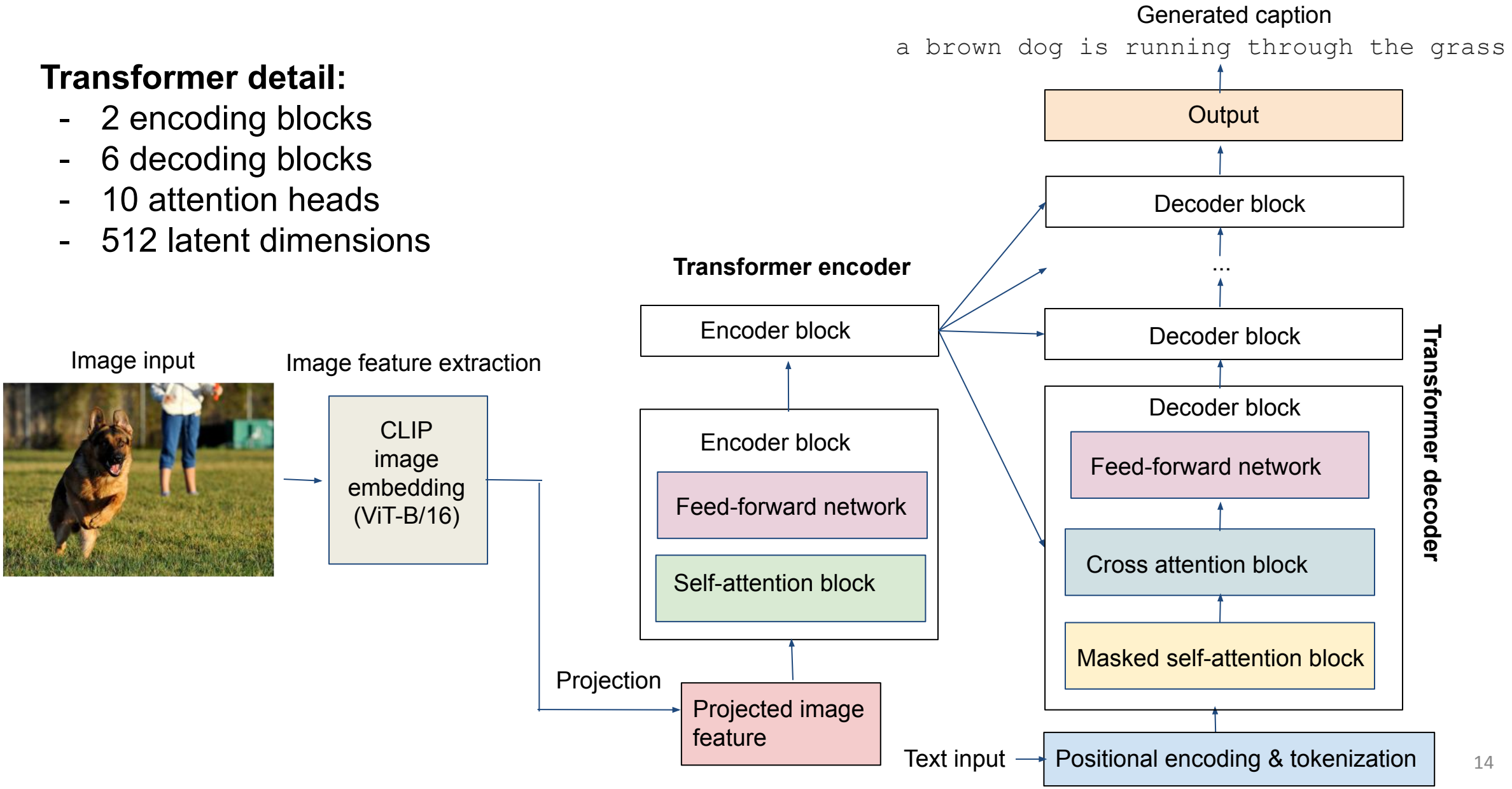
- CLIP for image embedding
  - OpenAI CLIP (*Contrastive Language–Image Pre-training*)
  - Trained to minimize contrastive loss on 4 million image-text pairs
  - Generate better embedding for representing details of images



Model 1 Architecture: Encoder-Decoder Transformer

Transformer detail:

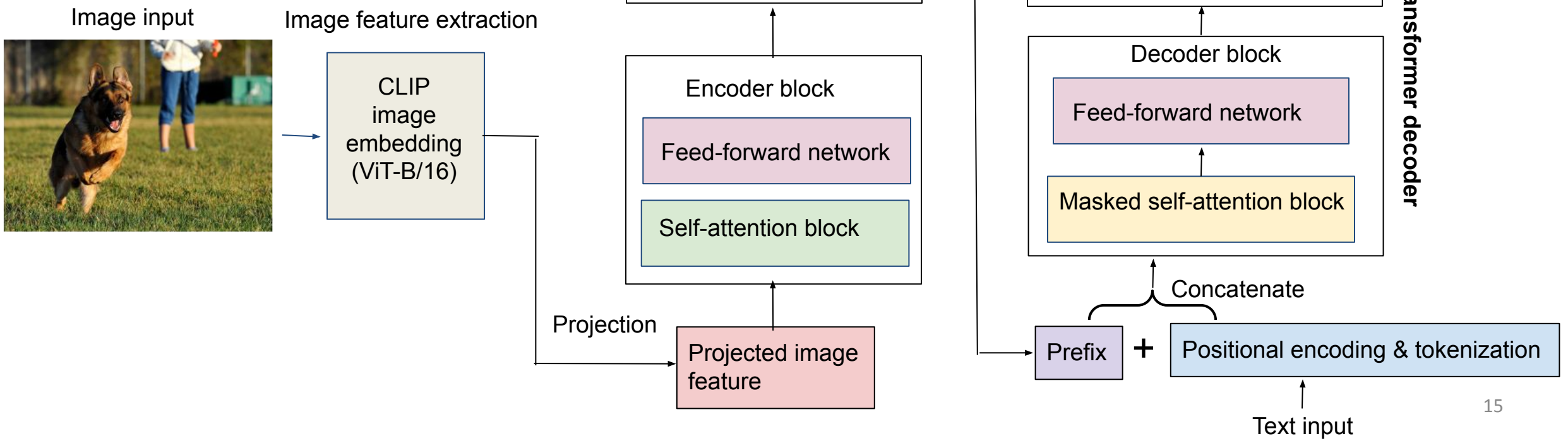
- 2 encoding blocks
- 6 decoding blocks
- 10 attention heads
- 512 latent dimensions



Model 2 Architecture: Prefix Language Model

Transformer detail:

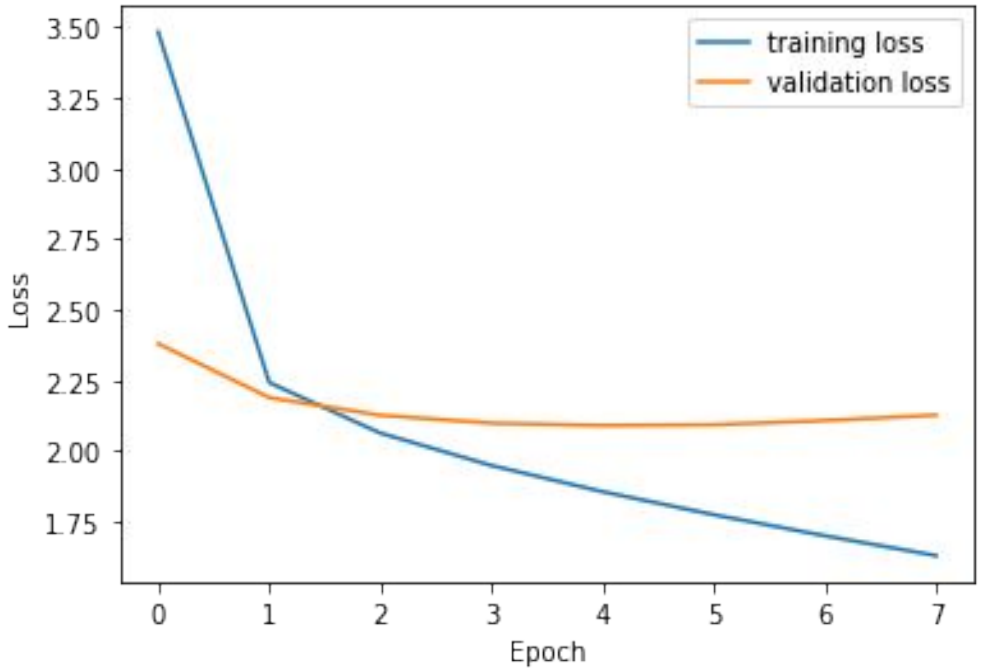
- 2 encoding blocks
- 6 decoding blocks
- 10 attention heads
- 512 latent dimensions
- prefix length = 16



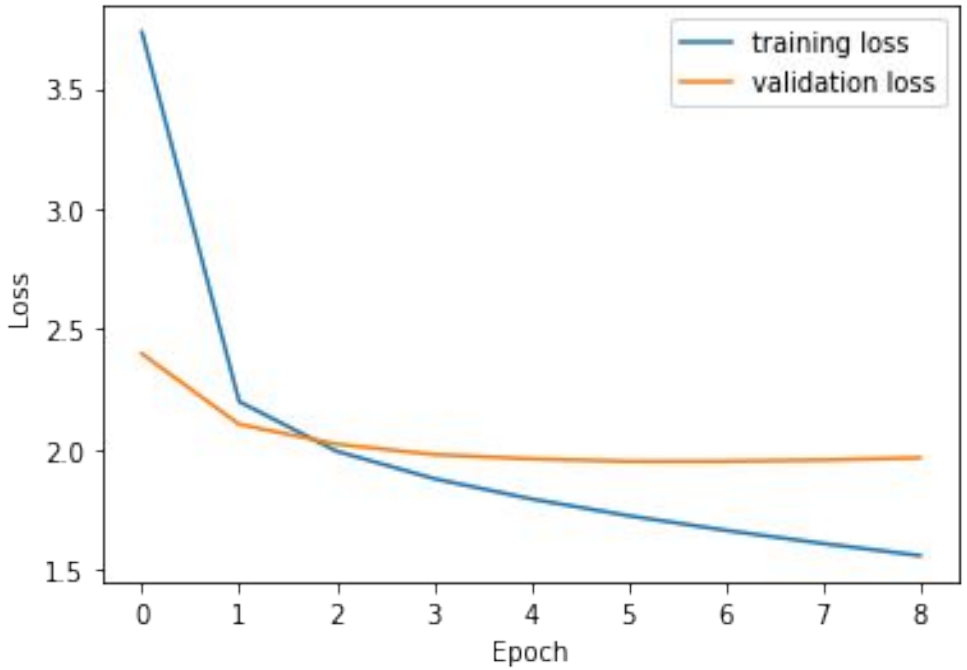
Training

Trained on 80% of Flickr 8k + MS-COCO data (591K), validated on 10% (657K), with the rest 10% was saved as test data

Model 1: encoder-decoder transformer



Model 2: prefix language model



BLEU scores on test data: BLEU-1 = 0.75; BLEU-2 = 0.58; BLEU-3 = 0.51; BLEU-4 = 0.52

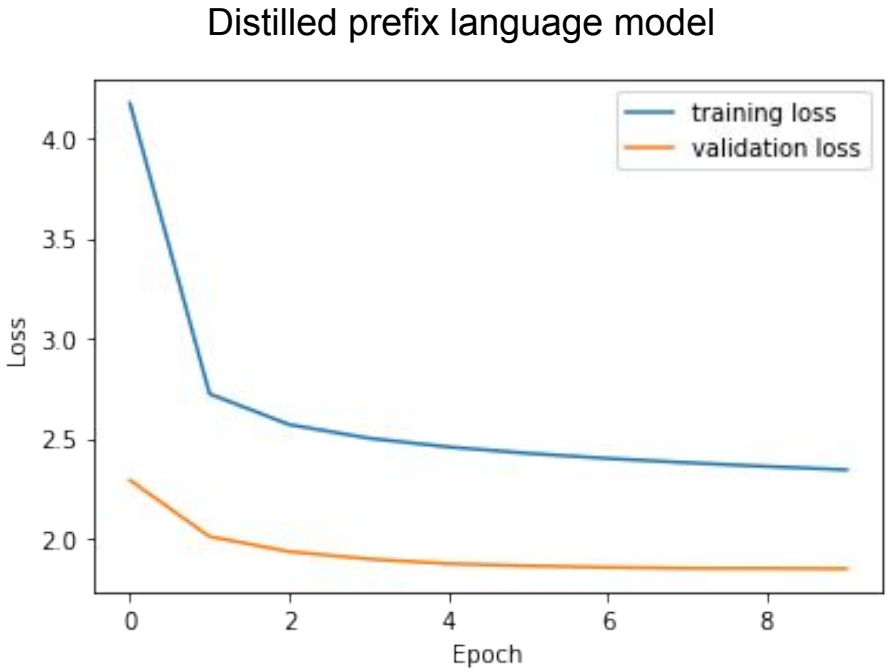
Distilled Prefix Model

We performed distillation on the prefix model with a smaller architecture:

- 1 encoder block
- 3 decoder blocks
- 8 attention heads
- prefix length = 10

Similar performance was achieved

- BLEU-1 = 0.75
- BLEU-2 = 0.58
- BLEU-3 = 0.52
- BLEU-4 = 0.53



Test Results - Generated Captions on Example Test Images (1)

a cat sitting on a bed next to a stuffed animal

a dog sitting in the back seat of a car

two people in uniform are on a boat

a man in a red jacket is skiing in the snow

two giraffes standing in a field with trees in the background

a man in a kitchen preparing a sandwich

Test Results - Generated Captions on Example Test Images (2)

a man and two children are looking at a birthday cake

a small boat in the water on a clear day

a man is doing a trick on a skateboard

a owl is sitting in a tree with leaves

a man is throwing a frisbee in a park

a cat sitting under an open umbrella on the floor

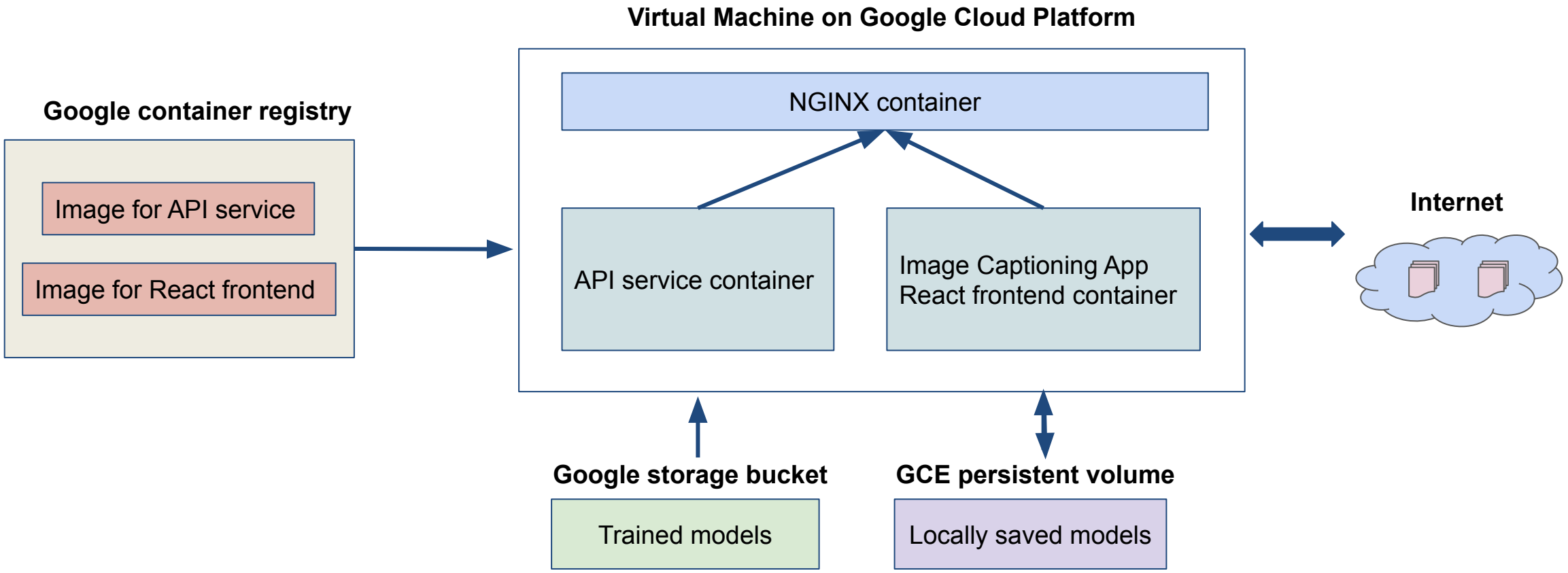
App Design

- API Server:
  - Fast API, serving 2 main models + 1 distilled model + 1 baseline model

- React Front End
  - Interacting with user for uploading images, sending image to API server, and displaying the generated caption

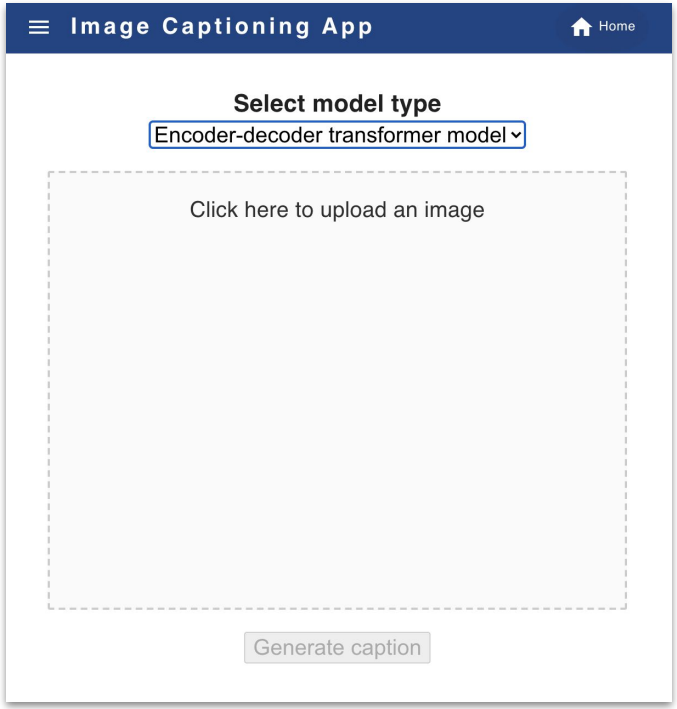
- NGINX
  - Bridging API server and react front end, exposing the App to web

App Design

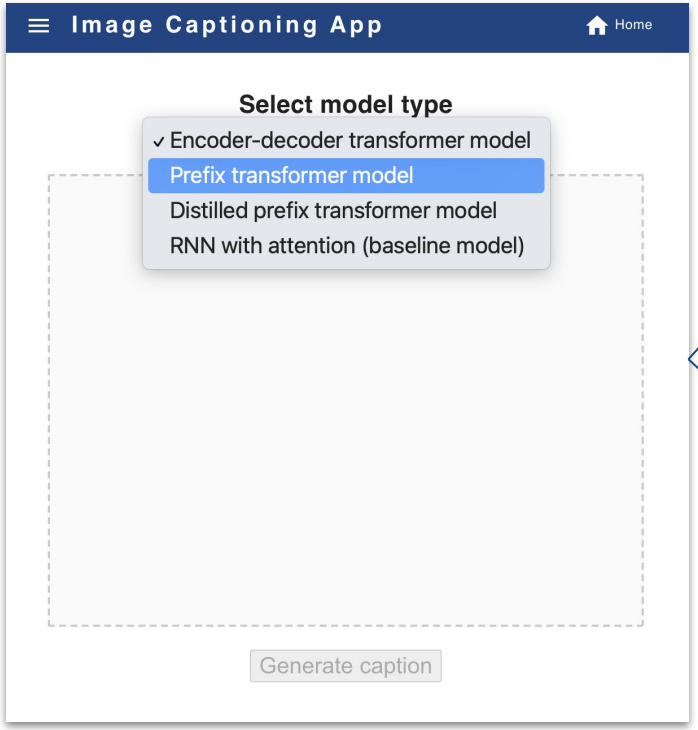

Virtual Machine on Google Cloud Platform
Google container registry
Image for API service
Image for React frontend
NGINX container
API service container
Image Captioning App
React frontend container
Internet
Google storage bucket
Trained models
GCE persistent volume
Locally saved models

App Design

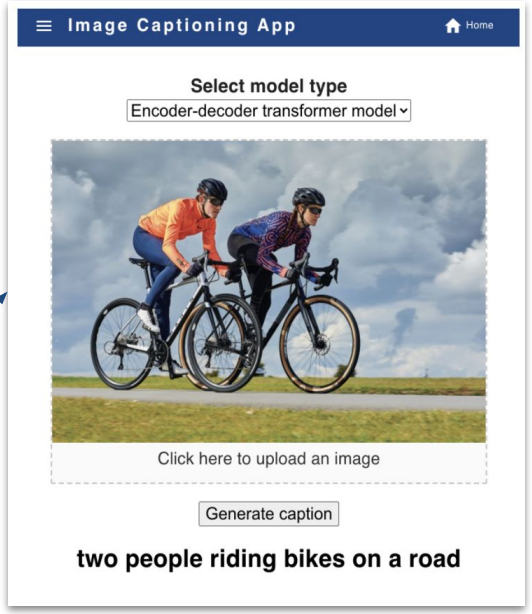
Front page

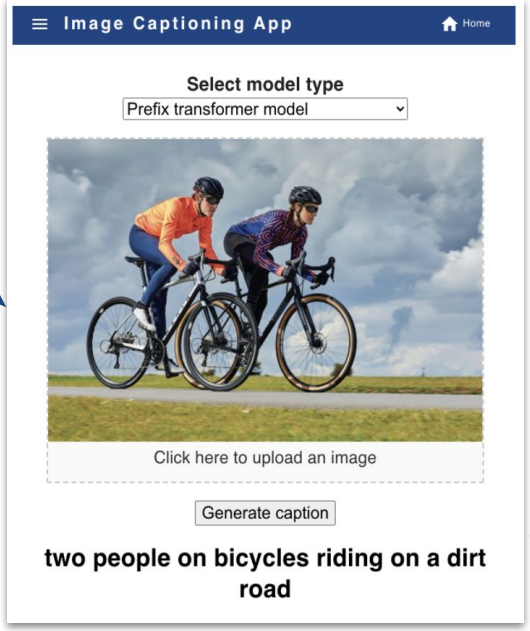


Select model type



Upload image and generator captions

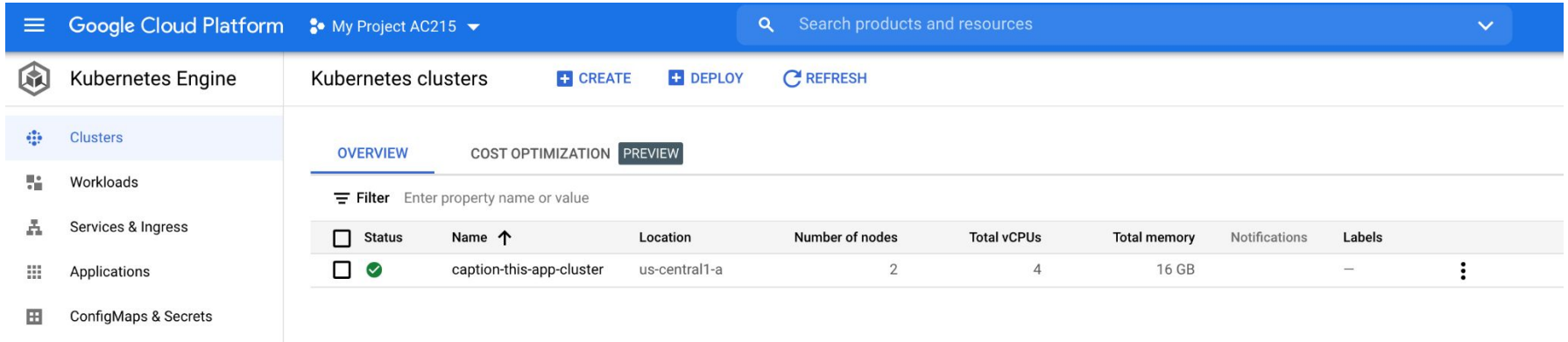


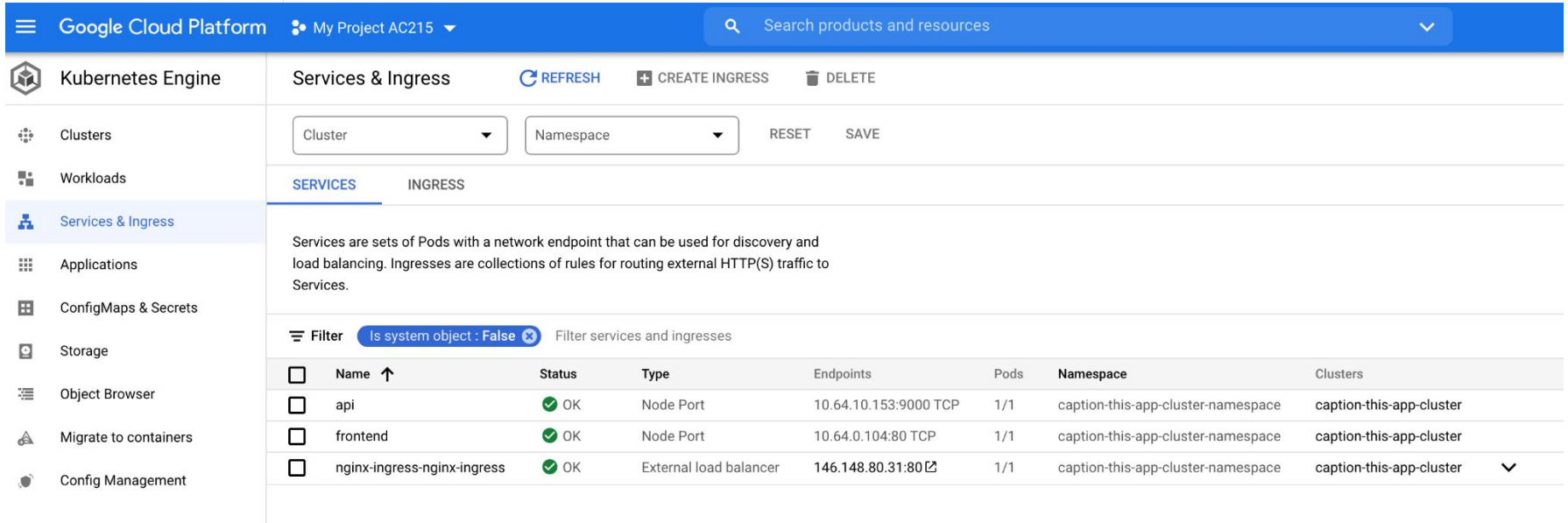


Deployment

We deployed the model-serving API, the React front end, and Nginx service with Kubernetes cluster using Ansible

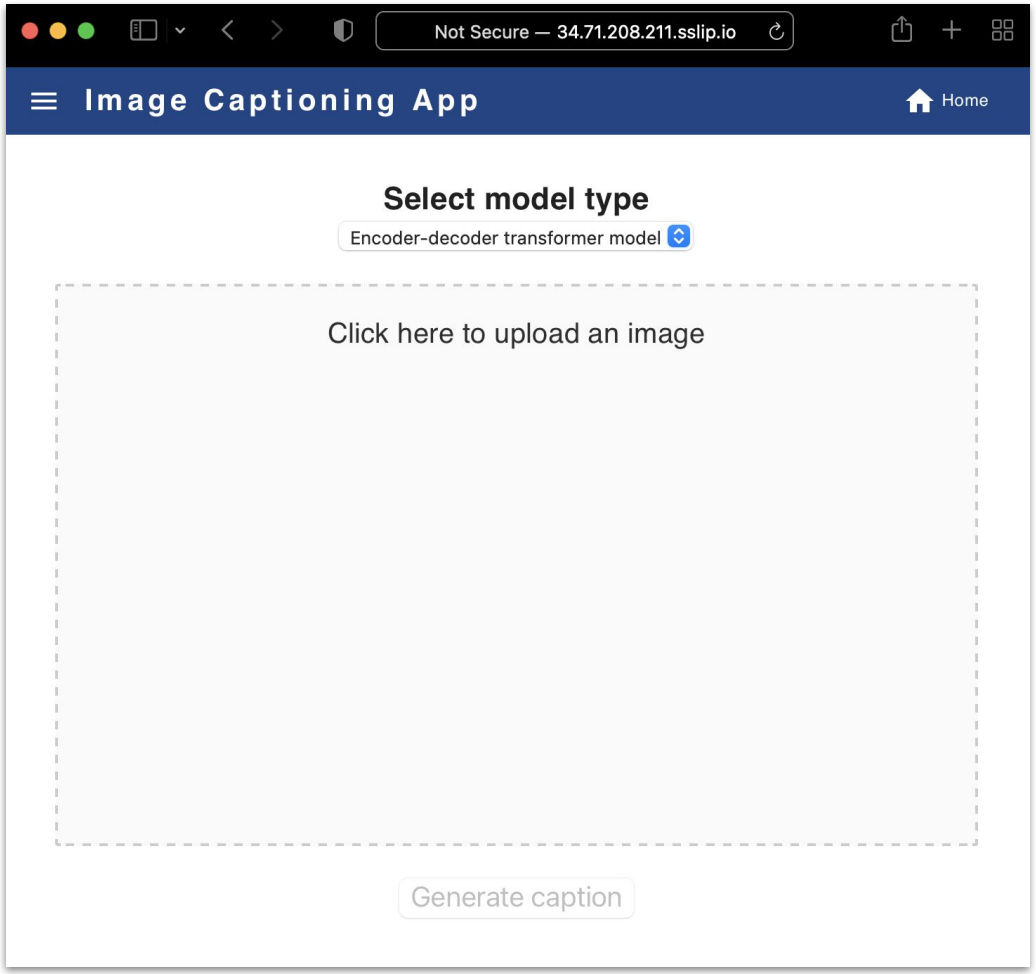
Screenshots:





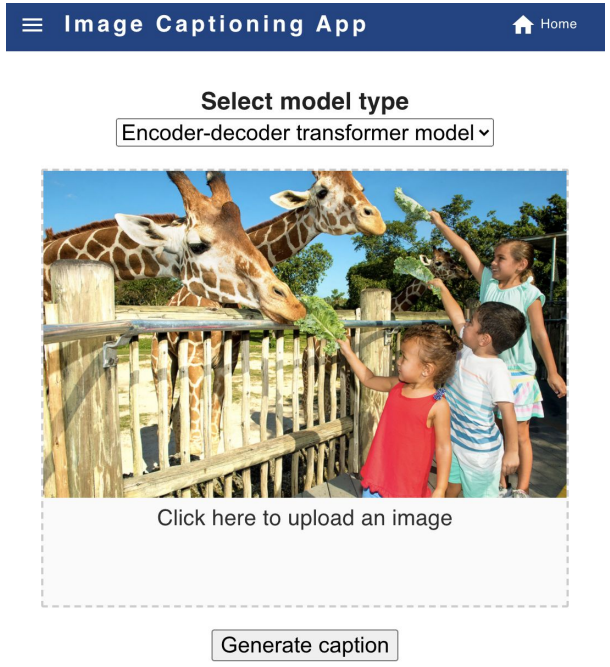
Deployment

Screenshot of the deployed App on external web:

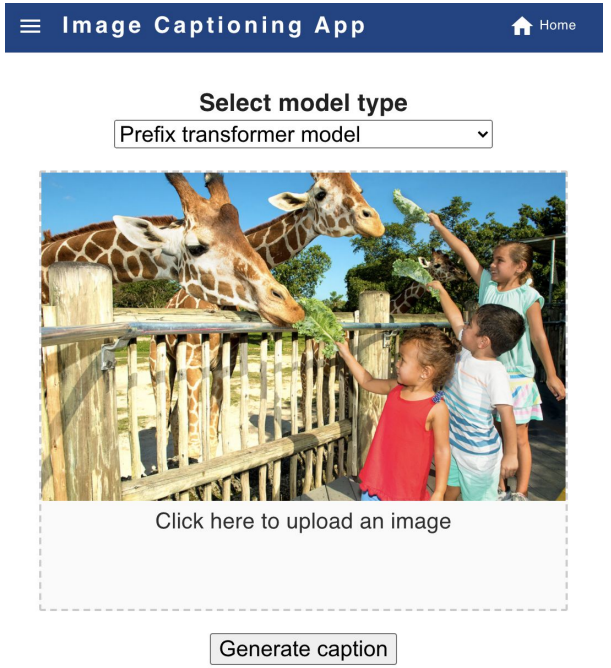


Deployment

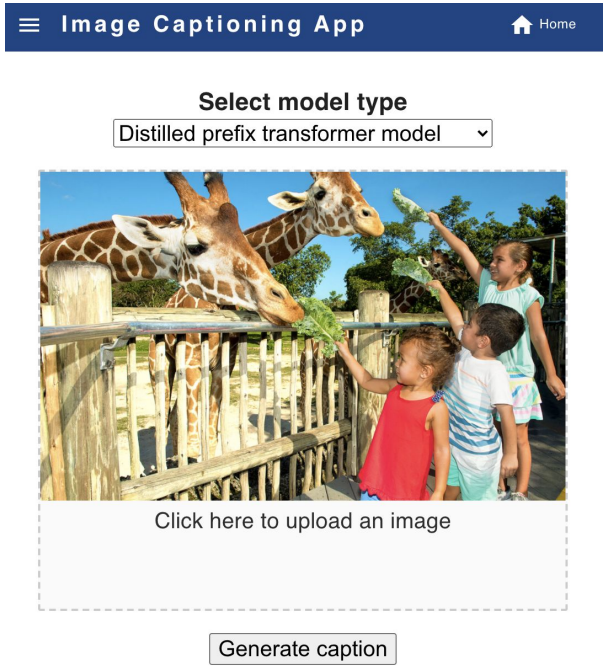
Generated captions of the 4 different models:



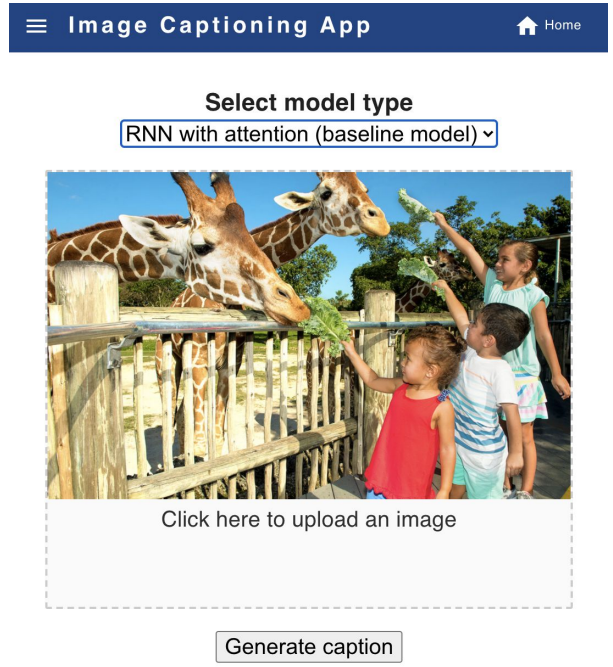
two giraffes eating from a womans hand while a child holds their hands out to the side



a woman and a child feeding a giraffe some food



a little girl feeding a giraffe some food



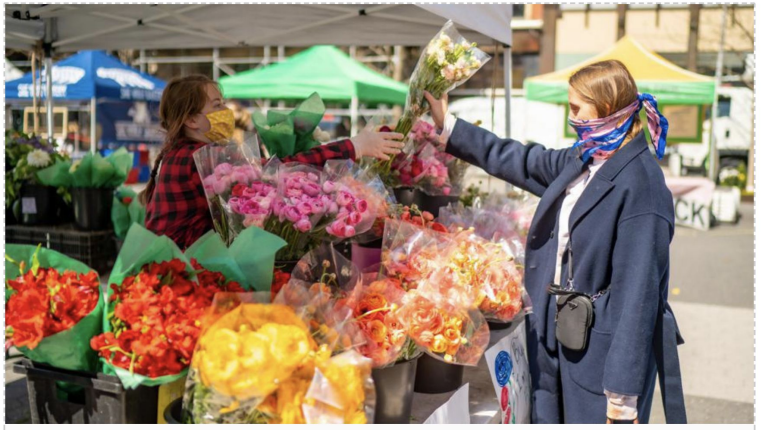
girls at the zoo

Deployment

More examples with random images downloaded from internet:

a couple of people walking across a park

a woman holding a flower in a flower shop



a group of children are sitting in a classroom

two giraffes eating from a womans hand while a child holds their hands out to the side

a plate of pasta with shrimp and vegetables

a group of people sitting at a bar

Supporting Notebooks in Github Repo

- Link to Github repo:
  https://github.com/skgithub14/AC215_KKST

- RNN-base Model (Baseline Model):
  https://github.com/skgithub14/AC215_KKST/blob/main/notebooks/Image_captioning_RNN_with_attention.ipynb
- Encoder-Decoder Transformer Model:
  https://github.com/skgithub14/AC215_KKST/blob/main/notebooks/Transformer_based_image_captioning_with_CLIP_embedding.ipynb
- Prefix Model with Distillation:
  https://github.com/skgithub14/AC215_KKST/blob/main/notebooks/CLIP_Prefix_Transformer_Image_Captioning_with_Distillation.ipynb