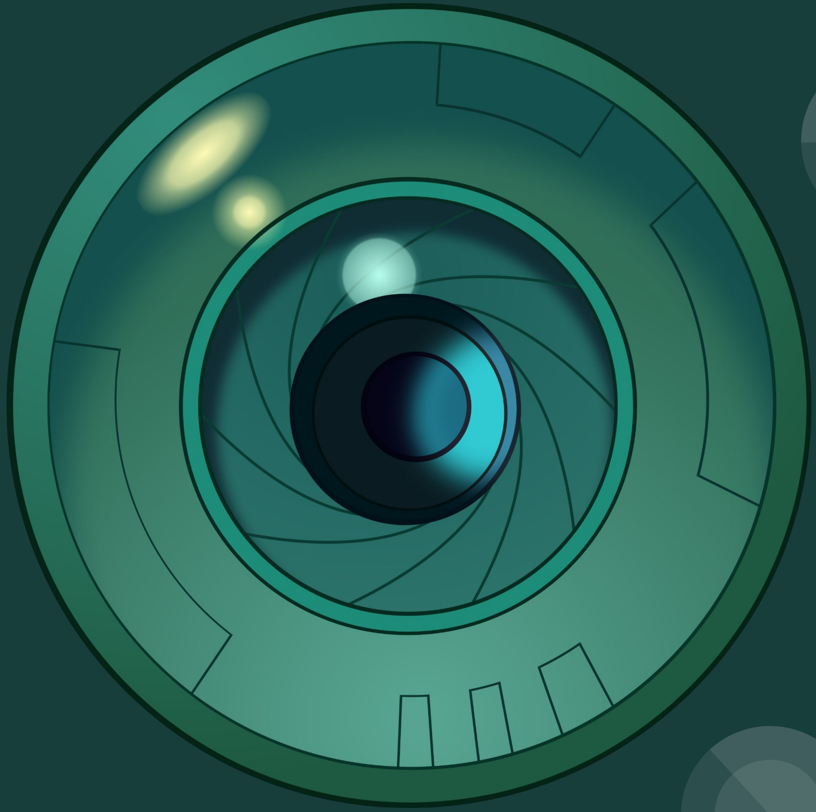
A-Eye App
For Image Captioning



AC 215: Advanced Practical Data Science, MLOps
Group Name: Pinkdrink
Heather Liu, Anita Mahinpei

Outline

Background

- In 2020, approximately 43.3 million people were blind and 295 million people had moderate to severe vision impairment worldwide (Bourne et al 2021).
- 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).
- Studies have revealed that most websites are not accessible to visually impaired individuals due to missing image captions/alt text (McEwan et al 2007).

Background

Therefore, with the ever-increasing prevalence of online services, accurate image caption is an important priority for improving accessibility.

Proposed Solution

A-Eye App for Image Captioning

" A man in the surf "
" A man in surfing "
" A man on a surfboard "

User Input Image

Image Caption Translation

Read Out

Proposed Solution

To tackle this problem, we developed a web application that allows users to upload images and have them be captioned by three different machine learning models. We specifically focused on creating captions for images of objects and scenery. Our app generates and displays three possible captions for a user provided image. Since the intended audiences are visually impaired individuals, our app provides a text-to-audio functionality so that the generated captions can be read out loud to the user if desired. To support a broader range of audiences, we also allow users to select the language they wish to use for the generated captions.

Project Scope (A-Eye App)

Proof Of Concept (POC)

- Download MSCOCO data
- Perform EDA to verify data
- Set up data pipeline (i.e. resize all images to a fixed size, normalize pixel values, tokenize the captions, store in tf dataset)
- Experiment with some baseline models trained on a subset of the dataset
- Validate captioning results on unseen images

Prototype

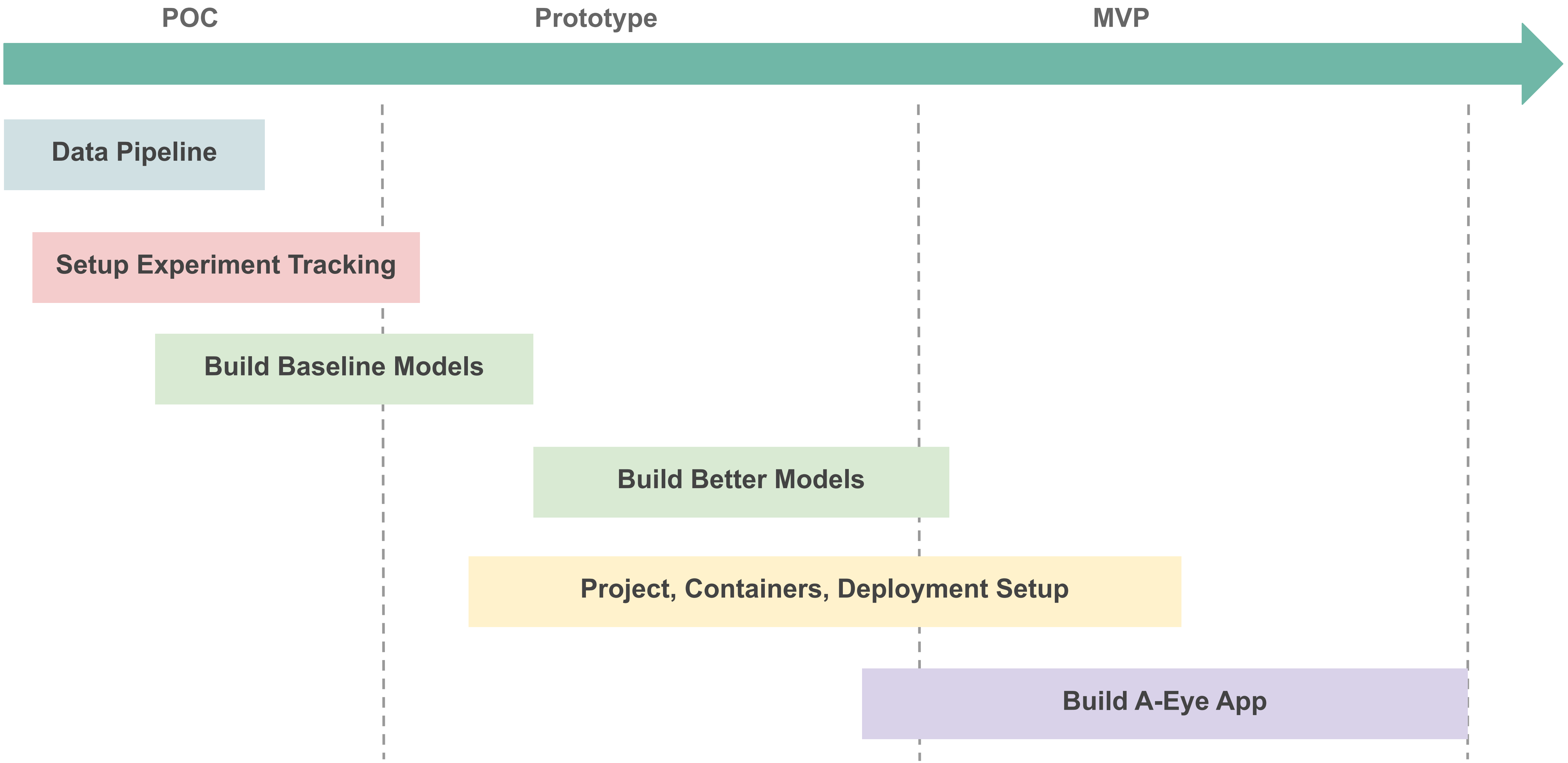
- Create a mockup of screens to see how the app could look like
- Deploy one model to Fast API to service model predictions as an API

Minimum Viable Product (MVP)

- Create App to caption images
- API Server for uploading images and predicting using best model

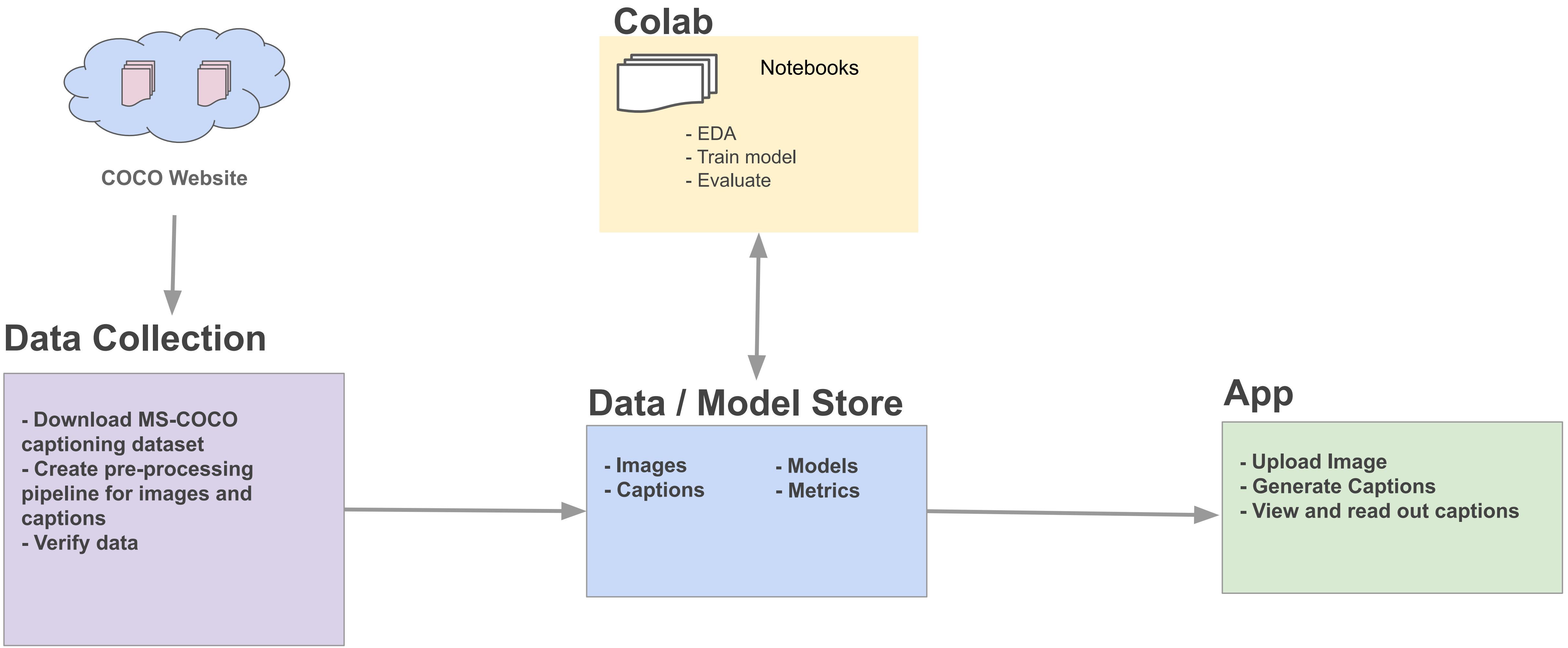
Project Workflow

A-Eye App



Process Flow

A-Eye App



Data: Overview

We used the Microsoft Common Objects in Context (COCO) data for our project. COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features:

- Object segmentation
- Recognition in context
- Superpixel stuff segmentation
- 330K images (>200K labeled)
- 1.5 million object instances
- 80 object categories
- 91 stuff categories
- 5 to 7 captions per image
- 250,000 people with keypoints

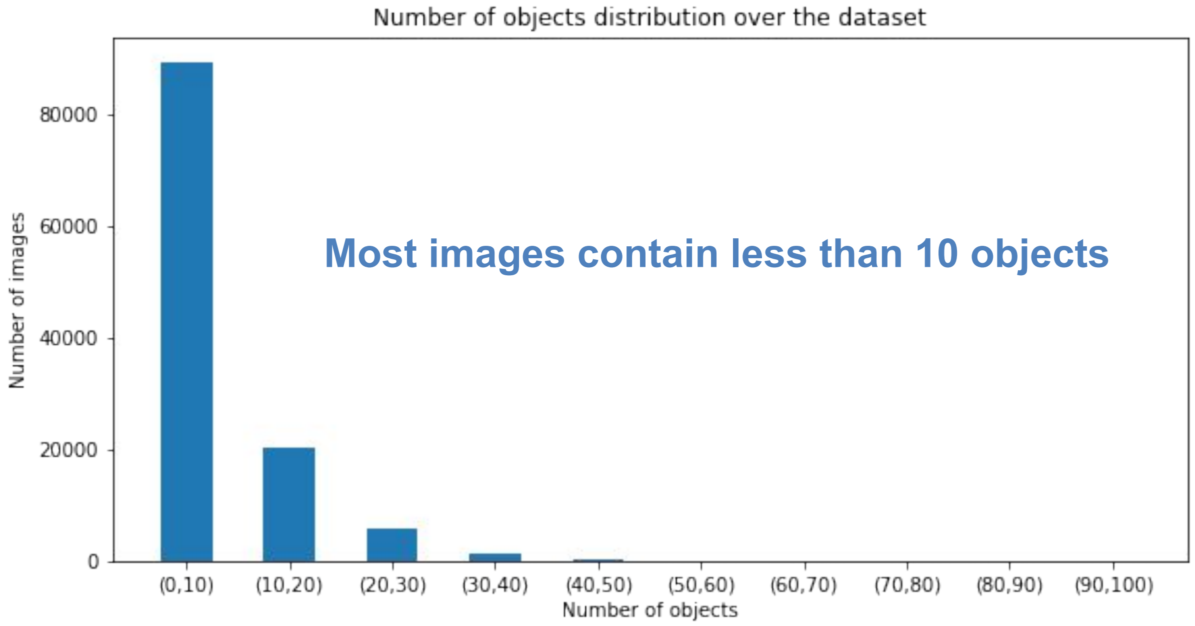
Data: Overview

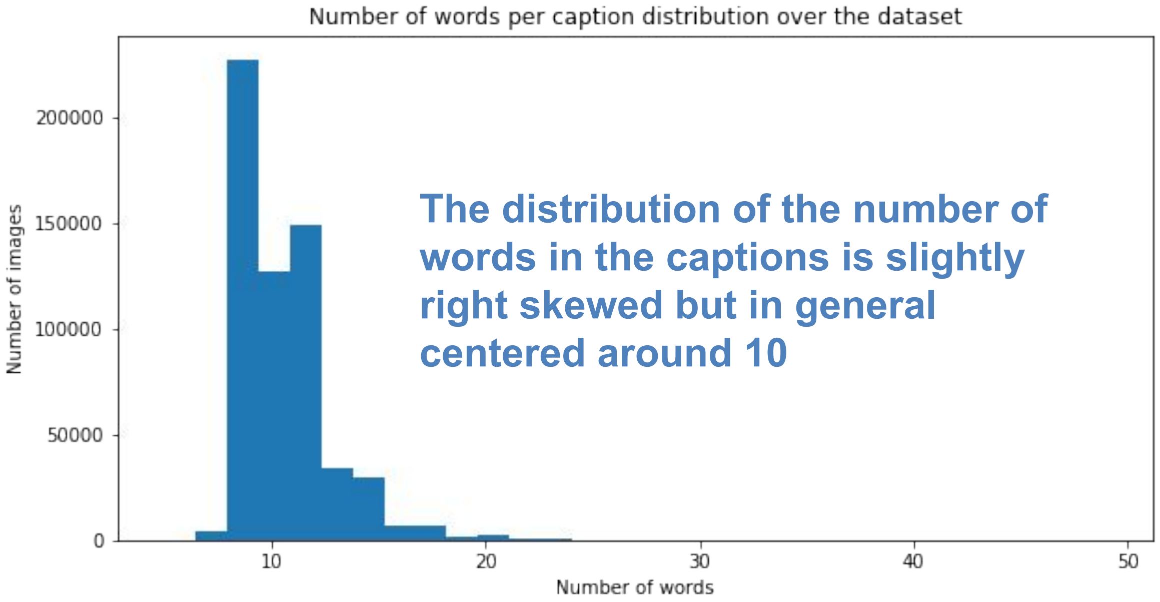
We used the images and the caption labels for training models.

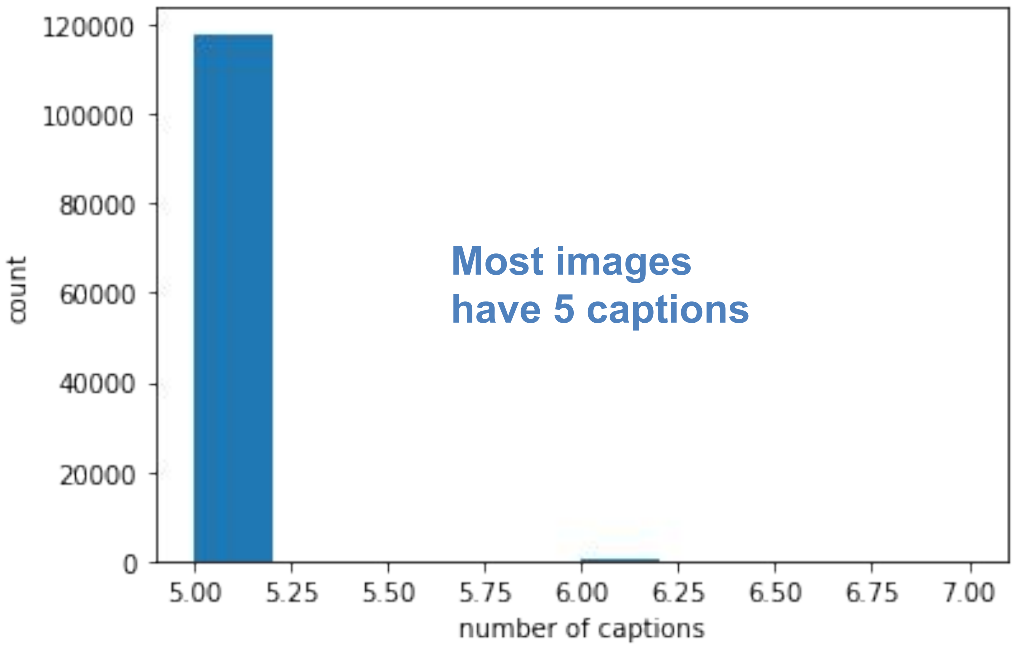
Sample image with 5 gold captions:

```
A couple of people riding on top of a wave on surfboards.
A man rides a white surfboard near another person in the ocean.
one person surfing one person laying on a surfboard
The guy is riding the wave as a girl watches.
Surfers surfing in the ocean on a clear day.
```

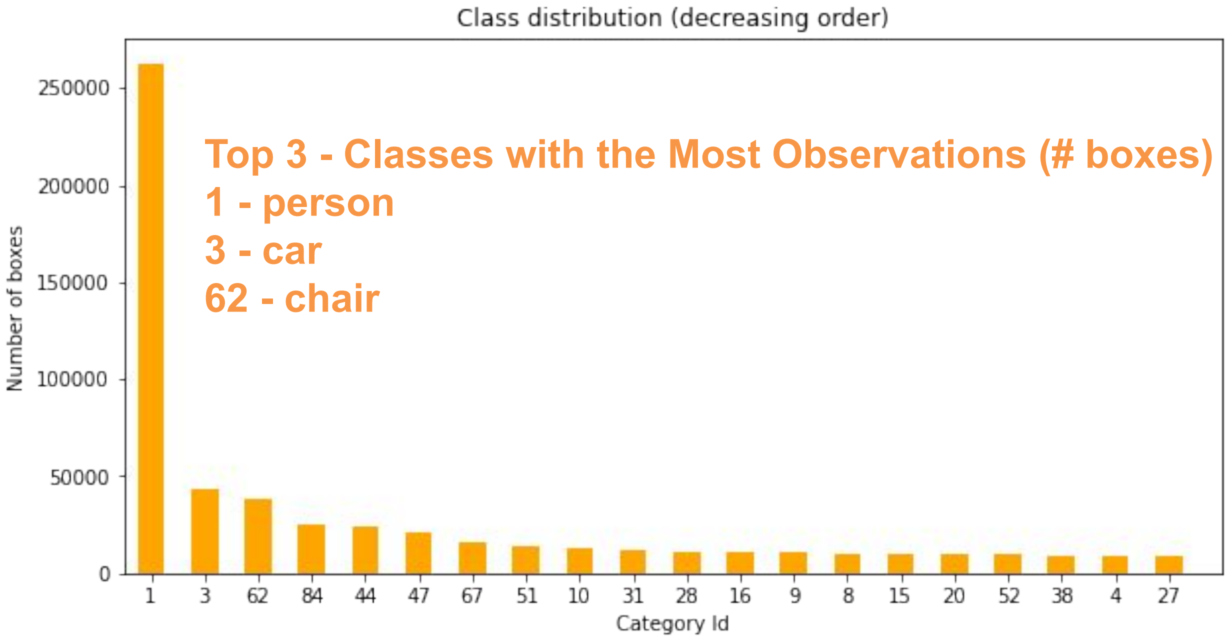
Data: EDA Results

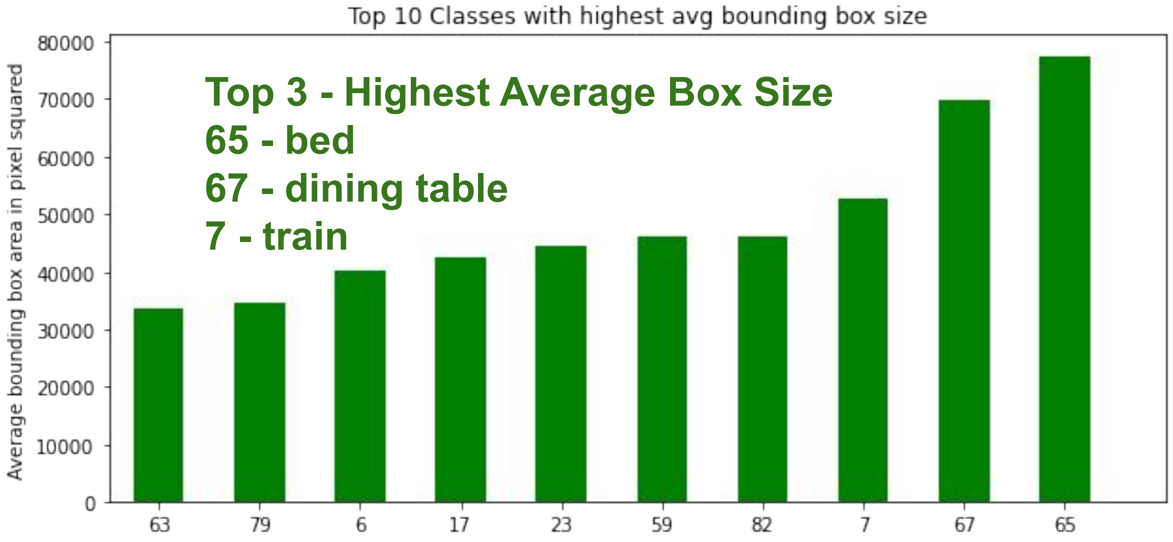


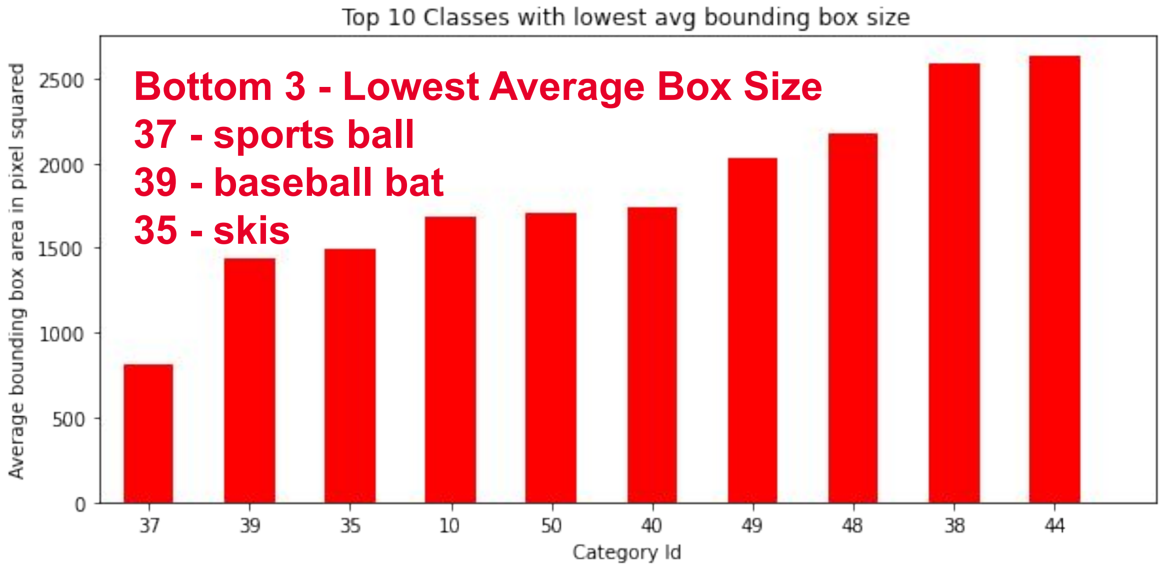




Data: EDA Results







Models

In terms of deep learning models, both computer vision and natural language processing models will be used.

- **Computer Vision:** Pre-trained, frozen CNN architectures (e.g. VGG, Inception) will be used to extract features from the images. (Note: we decided to use frozen encoders trained on ImageNet because we ran a trial without freezing layers and observed a lot of over-fitting.)
- **Language:** RNN and LSTM structures
- **Attention:** As described in the paper, "Show, Attend, and Tell", models that incorporate attention to the image feature map perform, have improved performance. We will use the attention mechanism described in this paper in one of our models.

Models

We have trained the following models:

1. **Inception-GRU:**
   - Extract feature map from the last CNN layer of frozen InceptionV3 with ImageNet weights
   - Attend to feature map
   - Decoder with an embedding layer (not pre-trained), a GRU layer and 2 fully-connected layers
2. **Inception-LSTM:**
   - Extract feature map from the last CNN layer of frozen InceptionV3 with ImageNet weights
   - Attend to feature map
   - Decoder with an embedding_layer(not pre-trained), a LSTM layer and 1 fully-connected layer
3. **VGG-LSTM:**
   - Extract feature map from the last CNN layer of frozen VGG16 with ImageNet weights
   - Language feature extractor with an embedding_layer(not pre-trained), a dropout layer, and an LSTM layer. (This extracts features from the previous caption words to predict the next word)
   - Combine the output of the language and image feature extractors and feed it into a fully-connected layer to predict the next word of the caption

Models: Evaluation

We compared the performance of our models using the loss function and BLEU-4 scores on held-out test data. The BLEU score is a method of comparing generated texts against a set of golden text labels (in our case image captions) as described in this paper.

Models: Evaluation Results

We trained with 82783 images each with 5 captions. We used 500 unseen image-caption pairs from the COCO dataset to calculate the BLEU-4 score.

We can see that adding an attention mechanism helps a lot with performance. Both the Inception-GRU and Inception-LSTM models which used attention have a higher BLEU score than the model that didn't use attention. Using an LSTM layer in the decoder also helps improve performance significantly as opposed to using a GRU layer.

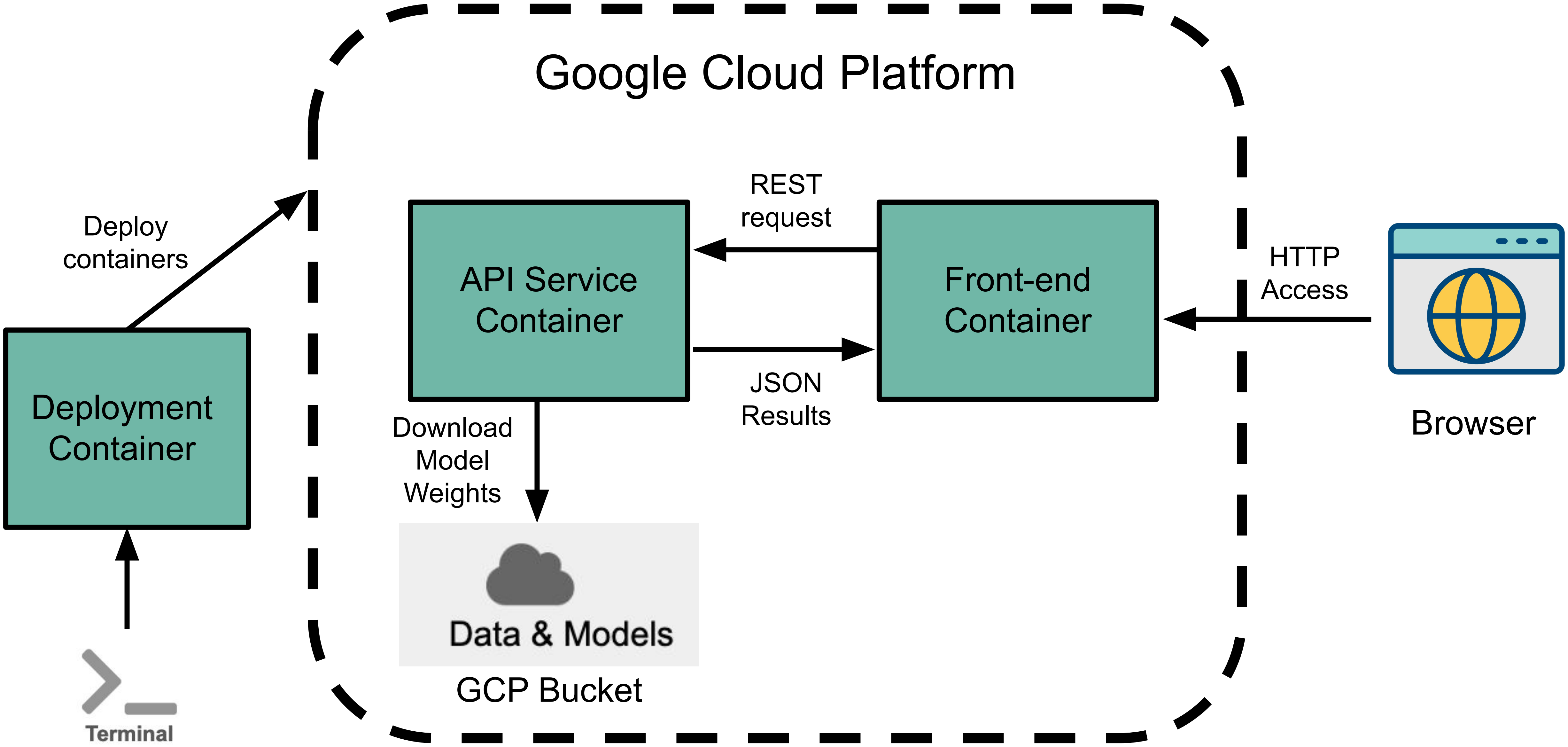
Model	Test BLEU-4
Inception-GRU	0.47476
Inception-LSTM	0.60926
VGG-LSTM	0.43329

App Architecture

Our app constitutes three docker containers:

1. api-service:
   a. one endpoint to generate captions and one that returns the caption audios
   b. written in Python and uses the Google Cloud APIs
2. frontend-react:
   a. Allows users to upload an image, view the captions and caption audios and change the language settings of the app.
   b. Written in Javascript and CSS and uses the React library.
3. deployment:
   a. Used to deploy the app to GCP using Ansible scripts.

App Architecture



App Architecture

API Service Process Flow:

Image-to-Text:
- Trained Encoder Decoder Models
- Generate English Captions

→

Text-to-Text:
- Google Translate API
- Translate English to User-Specified Language

→

Text-to-Audio:
- Google Cloud Text-to-Speech API
- Read Caption in User-Specified Language

Deployment

We used Ansible scripts to deploy our app on GCP and Kubernetes.

- Deployment to GCP
  - http://35.222.164.164/

- Deployment to K8s Cluster
  - http://35.202.124.222.sslip.io/

Note: we have shut them down for now to save GCP credit.

Deployment: GCP

a-eye-app-project

Filter Enter property name or value

Name ↑	Hostname	Visibility
a-eye-app-api-service	gcr.io	Private
a-eye-app-frontend-react	gcr.io	Private

VM instances CREATE SCHEDULE DELETE RESET OPERATIONS HELP ASSISTANT SHOW INFO PANEL LEARN

INSTANCES INSTANCE SCHEDULE

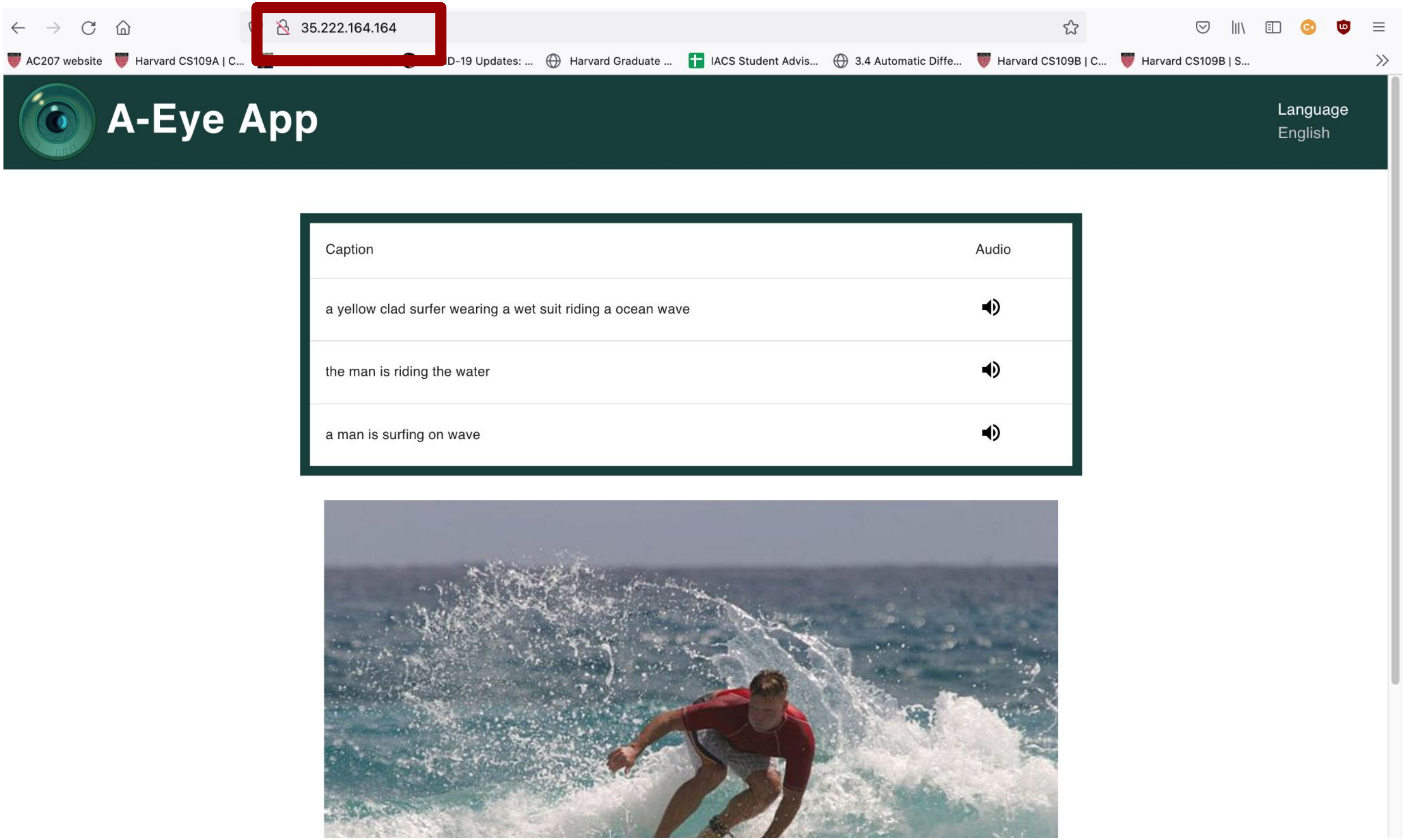
VM instances are highly configurable virtual machines for running workloads on Google infrastructure. Learn more

Filter Enter property name or value

Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
✓	a-eye-app	us-central1-a			10.128.0.4 (nic0)	35.222.164.164	SSH
✓	gke-a-eye-app-cluster-default-pool-74a0105f-2tf4	us-central1-a		gke-a-eye-app-cluster-default-pool-...	10.128.0.8 (nic0)	35.202.124.222	SSH
✓	gke-a-eye-app-cluster-default-pool-74a0105f-xrvj	us-central1-a		gke-a-eye-app-cluster-default-pool-...	10.128.0.7 (nic0)	35.184.212.54	SSH

Deployment: GCP

A-Eye App



Deployment: Kubernetes

A-Eye App

Kubernetes clusters

CREATE DEPLOY REFRESH OPERATIONS

OVERVIEW COST OPTIMIZATION PREVIEW

Filter Enter property name or value

Status	Name ↑	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
✓	a-eye-app-cluster	us-central1-a	2	2	7.5 GB		–

Services & Ingress

REFRESH CREATE INGRESS DELETE

Cluster Namespace RESET SAVE

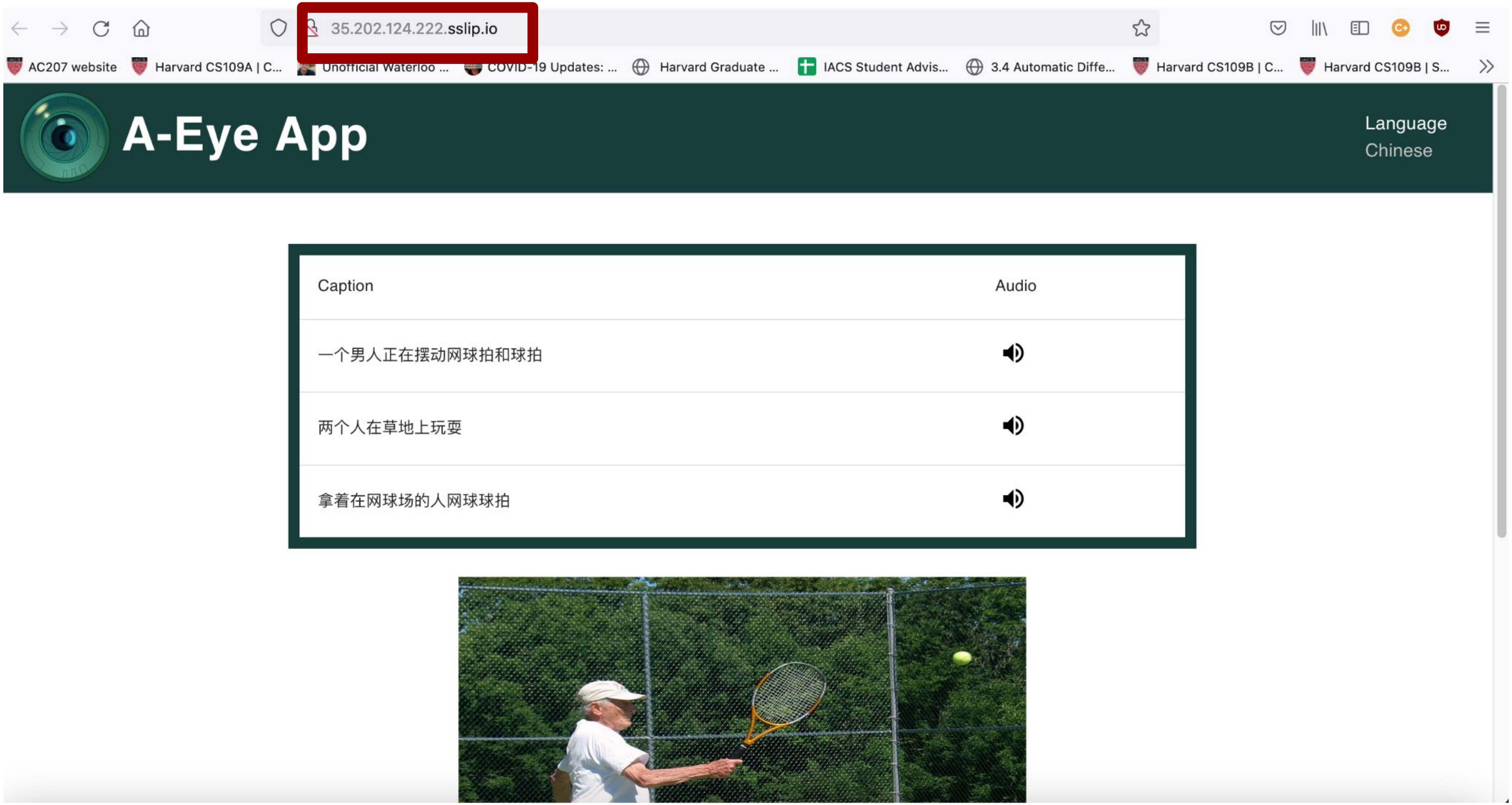
SERVICES INGRESS

Services are sets of Pods with a network endpoint that can be used for discovery and load balancing. Ingresses are collections of rules for routing external HTTP(S) traffic to Services.

Filter Is system object : False Filter services and ingresses

Name ↑	Status	Type	Endpoints	Pods	Namespace	Clusters
api	OK	Node Port	10.24.14.220:9000 TCP	1/1	a-eye-app-cluster-namespace	a-eye-app-cluster
frontend	OK	Node Port	10.24.12.145:80 TCP	1/1	a-eye-app-cluster-namespace	a-eye-app-cluster
nginx-ingress-nginx-ingress	OK	External load balancer	35.202.124.222:80	1/1	a-eye-app-cluster-namespace	a-eye-app-cluster

Deployment: Kubernetes



App Demo

App Demo - English

Caption	Audio
several people that are playing frisbee on a field	🔊
two children are playing with ball	🔊
two children are playing soccer on field	🔊

App Demo - French

A-Eye App

Caption	Audio
Une adolescente portant sur le bord du paddle dans l'océan	🔊
Jeune garçon en chemise rouge joue dans l'eau	🔊
Deux enfants jouent dans de l'eau avec leurs planches de surf	🔊

App Demo - Chinese

Caption	Audio
冲浪板上的一个男人在海洋中穿着黑色潜水服的冲浪板	🔊
那个男人骑着水	🔊
一个男人在波浪上冲浪	🔊

App Demo - Spanish

A-Eye App

Caption	Audio
Una pequeña pizza sentada en la parte superior de una sartén.	
Dos chicas están jugando en la hierba.	
Pizza con queso y verduras en él.	

Conclusions

- We created a simple web application that automatically captions images in four different languages (English, French, Spanish, Chinese).

- Future extensions can:
  - Support more languages
  - Combine multiple image captioning datasets (MS-COCO, Flickr30k, etc) to train better models with more data
  - Train on and support other types of images that are commonly found on websites such as infographics
  - Use pre-trained language embeddings (e.g. GloVe or Word2Vec) in the decoder

Project Files

- Github Repo: https://github.com/pinkpinkdrink/AC215_pinkdrink
- Medium Post Draft: https://medium.com/@amahinpei/a-eye-image-captioning-app-3bf7c1d11e91
- Video Recording: https://youtu.be/1GRi85gAUbw

References

1. Alkhathlan, Mallak, et al. "'Honestly I Never Really Thought about Adding a Description': Why Highly Engaged Tweets Are Inaccessible." *Human-Computer Interaction—INTERACT 2021*, 2021, pp. 373–395., doi:10.1007/978–3–030–85623–6_23.
2. Bourne, Rupert, et al. "Trends in Prevalence of Blindness and Distance and near Vision Impairment over 30 Years: An Analysis for the Global Burden of Disease Study." *The Lancet Global Health*, vol. 9, no. 2, 2021, doi:10.1016/s2214–109x(20)30425–3.
3. Brownlee, Jason. "How to Develop a Deep Learning Photo Caption Generator from Scratch." *Machine Learning Mastery: How to Develop a Deep Learning Photo Caption Generator from Scratch*, 23 Dec. 2020, machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/.
4. Lin, Tsung-Yi, et al. "Microsoft Coco: Common Objects in Context." *Computer Vision—ECCV 2014*, 2014, pp. 740–755., doi:10.1007/978–3–319–10602–1_48.
5. McEwan, Tom, and Ben Weerts. "Alt Text and Basic Accessibility." *Electronic Workshops in Computing*, 2007, doi:10.14236/ewic/hci2007.64.
6. Papineni, Kishore, et al. "Bleu." *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics—ACL '02*, 2001, doi:10.3115/1073083.1073135.
7. Sharma, Piyush, et al. "Conceptual Captions: A Cleaned, Hypernymed, Image Alt-Text Dataset for Automatic Image Captioning." *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, doi:10.18653/v1/p18–1238.
8. Xu, Kelvin, et al. " Show, Attend and Tell: Neural Image Caption Generation with Visual Attention." *Internation Conference on Machine Learning,* 2015.
9. Yesilada, Yeliz, et al. "Screen Readers Cannot See." *Lecture Notes in Computer Science*, 2004, pp. 445–458., doi:10.1007/978–3–540–27834–4_55.