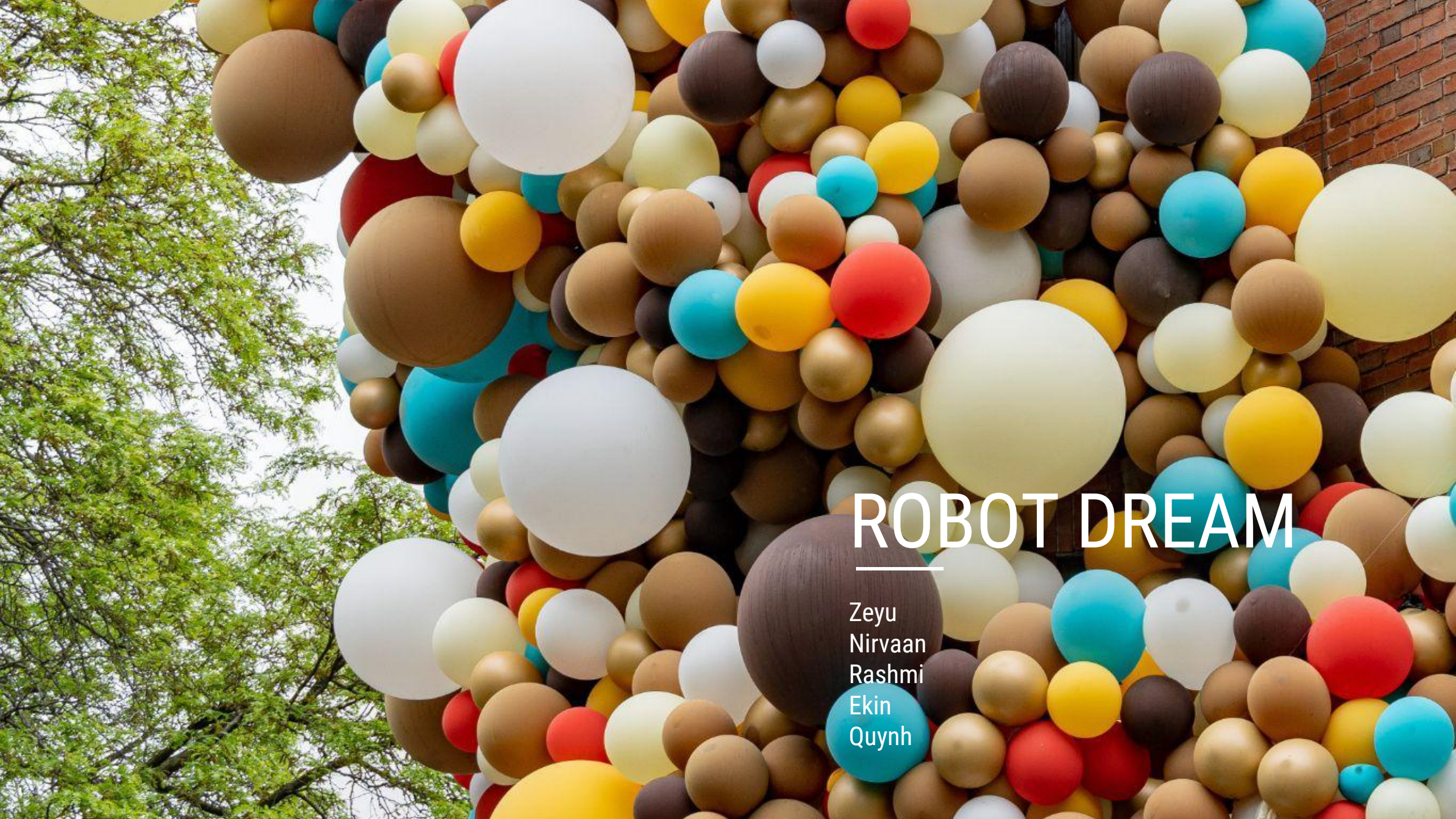


ROBOT DREAM

Zeyu
Nirvaan
Rashmi
Ekin
Quynh

Agenda

- Project Overview
- Milestones
- Deciding End Model
  - YOLOv5
  - DETR
- Deployment Overview
  - Flow chart, hardware requirement, service used for deployment
  - Problems during deployment
- Deployment Result
- Deployment Demonstration
- Future steps

Project Overview

Team: Robot Dream

Project Goal:

- Build a model to
  - detect balloon and count number of balloons
  - detect happy face
- Build a Flask App to upload a picture and receive the same image with
  - a text stating the number of balloon(s) and happy face(s) presenting in the photo
  - a copy of the uploaded image is saved
  - a copy of the result image with bounding box(es) around the balloon(s) and happy face(s) is saved

Final Dataset:

- Dataset of 2376 images with 2018 training and 358 validation images
- Image size: at least 600 x 600 pixels
- 2 classes: balloon (1150 images) and N/A (1226 images)

Project Milestones

SESSION 2

Labelling Guideline & Decision of Data Collection

SESSION 3

Model Research & Comparison of human baseline using (IoU)

SESSION 4

Modelling and Evaluation

- Yolov5 and DETR for balloon on dataset of 2645 images
- DCNN for Happy Faces

SESSION 6

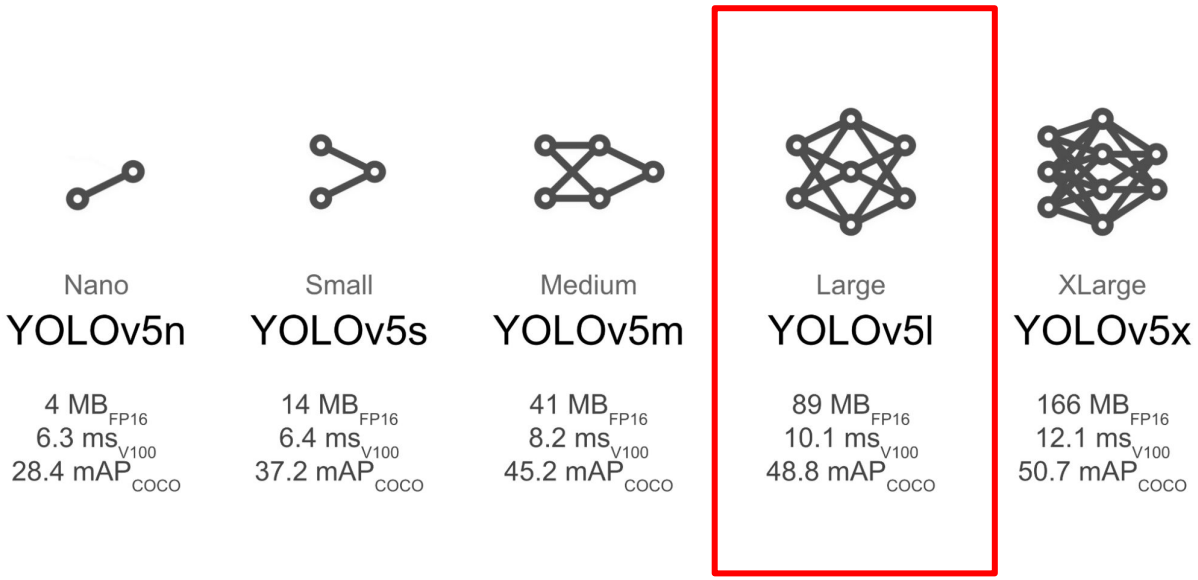
Deployment

- Re-train Yolov5 and DETR for balloon on dataset of 2376 images
- Building Flask App

Deciding the End Model

Yolo v5

Best Result with **YOLO v5l**



- Fine-tuning a pre-trained YOLOv5 model on a our balloon dataset
  - Model used - YOLOv5l
  - structured the data as per the model requirements

- Our training with dataset 2376 images
  - dataset of 2018 training images
  - dataset of 358 validation images
  - 50 epochs
  - trained on google colab
  - training time 2 hours

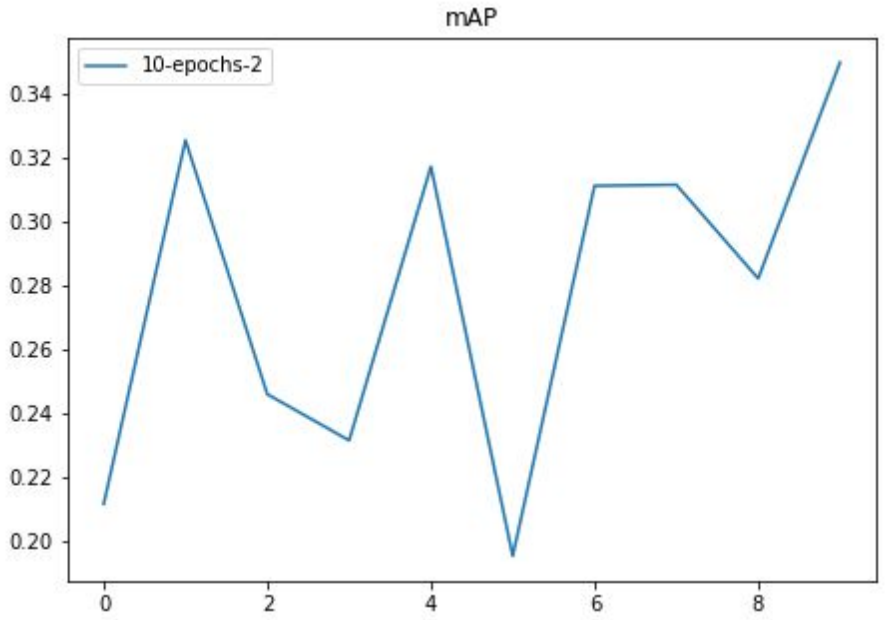
Mean Average Precision (mAP) - 79%
Recall - 79.2%
Precision - 80.7

Deciding the End Model

DETR

- Fine-tuning a pre-trained DETR model on a our balloon dataset
  - DETR RC 50
  - change the model-building structure to enable any amount of classes (in our case, num_classes = 1)
  - replace the classification head
  - transform the images to have maximum dimension of 600 x 1333 pixels

- Our model training with
  - dataset of 2376 photos (2018 training images, 358 validation images)
  - 10 epochs
  - free GPU on Google Collab
  - 40 mins training time



Mean Average Precision (mAP) - 35%

Model Comparison

	YOLO v5l	DETR

Dataset	- Train on final dataset of 2376 images
	  - 2018 training images
	  - 358 validation images
		- Train on final dataset of 2376 images
		  - 2018 training images
		  - 358 validation images

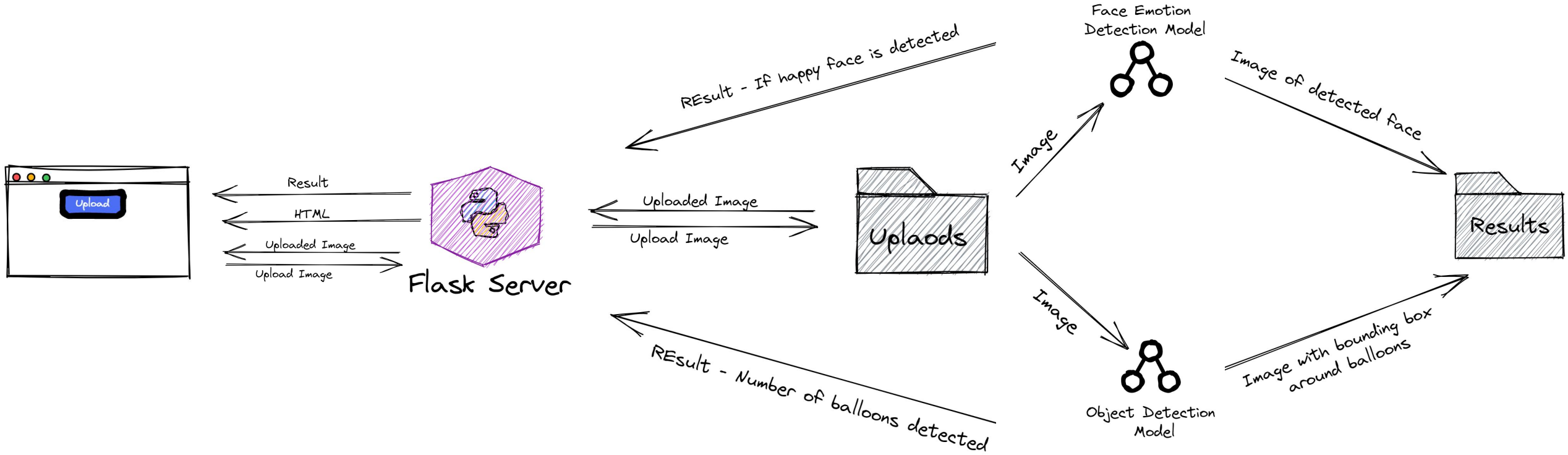
Training Process	- Fine-tuning a pre-trained Yolov5 model on a our balloon dataset
	  - Model used - YOLOv5l
	  - structured the data as per the model requirements
	- 50 epochs
	- trained on google colab
	- training time 2 hours
		- Fine-tuning a pre-trained DETR model on a our balloon dataset
		  - DETR RC 50
		  - change the model-building structure to enable any amount of classes (in our case, num_classes = 1)
		  - replace the classification head
		  - transform the images to have maximum dimension of 600 x 1333 pixels
		- 10 epochs
		- trained on google colab
		- training time 40 mins

Results	Mean Average Precision (mAP) - 79% Recall - 79.2 Precision - 80.7	Mean Average Precision (mAP) - 35%
	FINAL MODEL FOR BALLOON DETECTION	

Deployment Overview

System Requirements

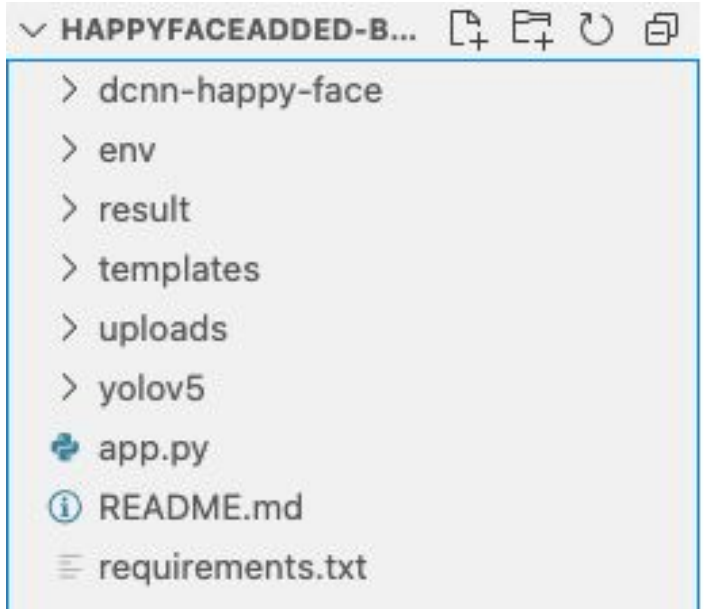
- Python 3.8 or 3.9 (other versions would cause error)
- Required packages - Flask, Modelling packages, etc
- Virtual environment



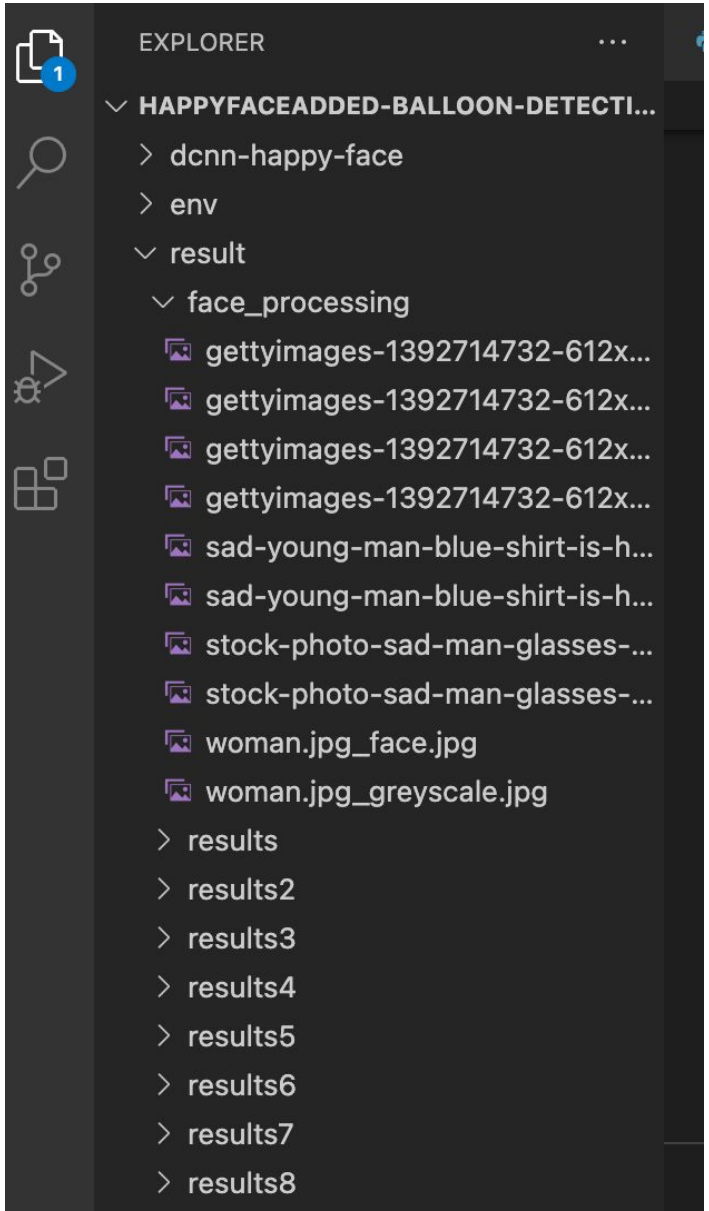
Problems during Deployment

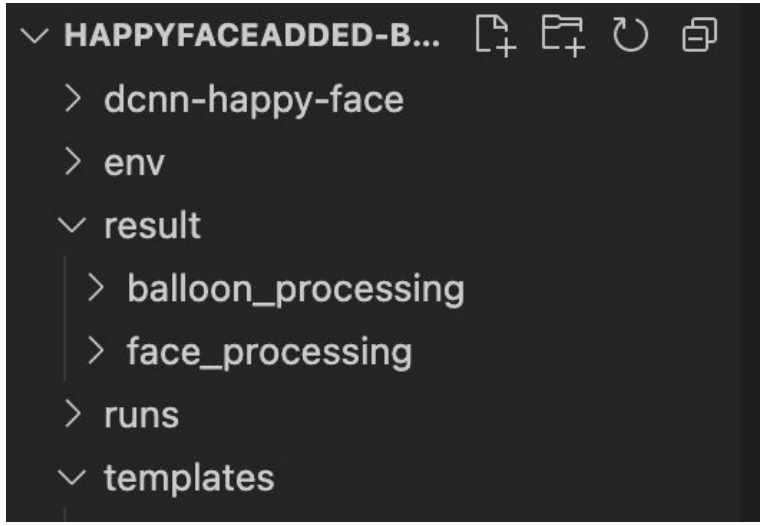
No.	Problems	Solutions	Results
1	269 duplicates in both training and validation sets in the original dataset of 2645 photos which might have affected the accuracy in model training	- Locate and remove these 269 duplicates → new final dataset of 2376 photos - Retrain both YOLO v5 and DETR using the new dataset and compare the end result again	Contrast to the results in previous session, YOLO v5 gives better results with mAP up to 79% while DETR achieve maximum mAP of 35% **→ Switch to YOLO v5 as final model for balloon detection**
2	Problem importing libraries for face detection	Trying out different python versions and download all necessary libraries for the deployment.	Python 3.8 is finally considered as the final version for our model deployment. Solved it on windows using a specific version of dlib for Python 3.9
3	Files too large to share on github	Share files using other platforms	Shared files using Google Drive

Deployment - File Structure, Incorporate Happy Face Model



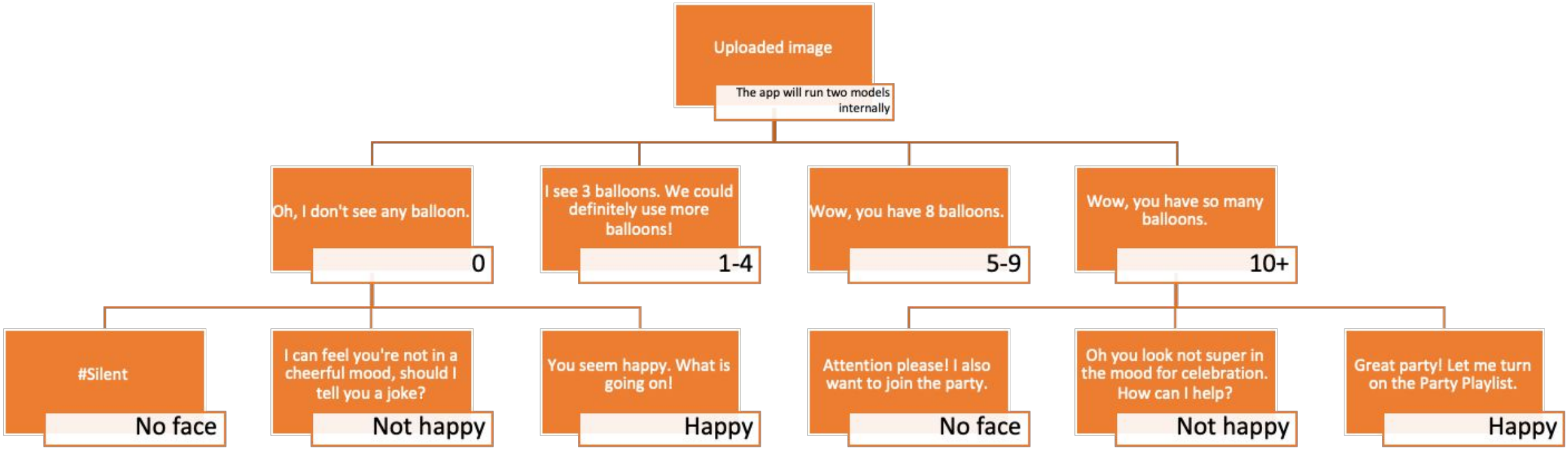
- **env** folder was used to set up project env for the deployment;
- **requirements.txt** stores all necessary library info;
- **app.py** is the actually app to run when deploying;
- **yolov5** and **cnn-happy-face** store the two models;
- uploaded photos will be cached in **uploads**;
- **result** stores all the processing files produced by the prediction model





- Initially, the processing files were not properly organized in the result folder
- for locating the result easier and better documentation of processing files, we improved the deployment code in app.py

Deployment - Output Context Design



Deployment - Sample Output

Entering Robot Vision™

Upload Image and Interact §°.°§

Choose File | no file selected | Upload

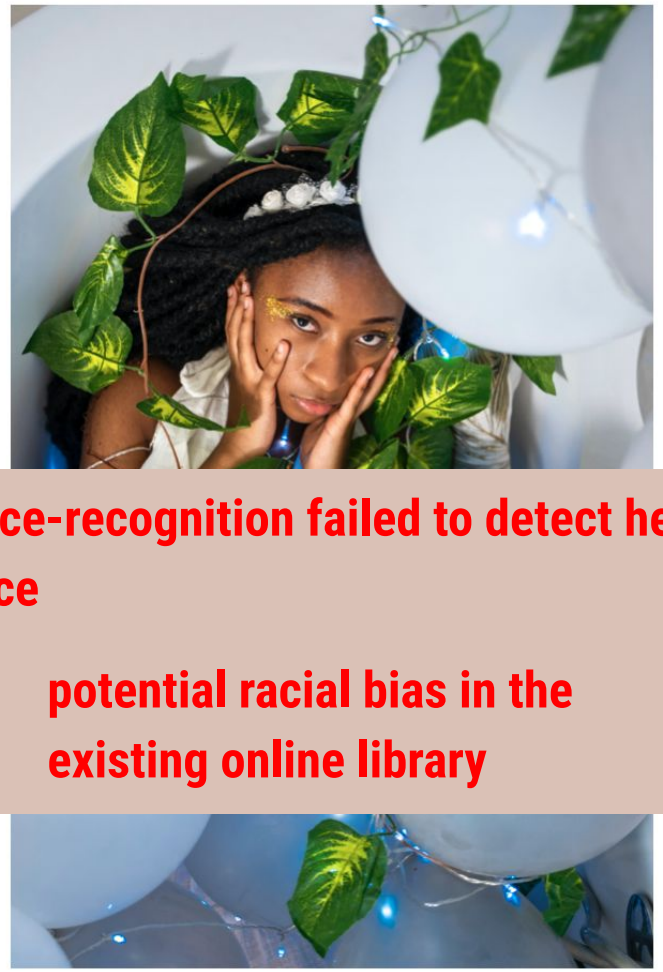
Wow, you have 5 balloons. Great party! Let me turn on the Party Playlist ^_^

I see 2 balloons. We could definitely use more balloons! Oh you look not super in the mood for celebration. How can I help ~_~?

I see 2 balloons. We could definitely use more balloons! Great party! Let me turn on the Party Playlist ^_^

Deployment - More outputs and Limitations

Oh, I don't see any balloon.



Face-recognition failed to detect her face

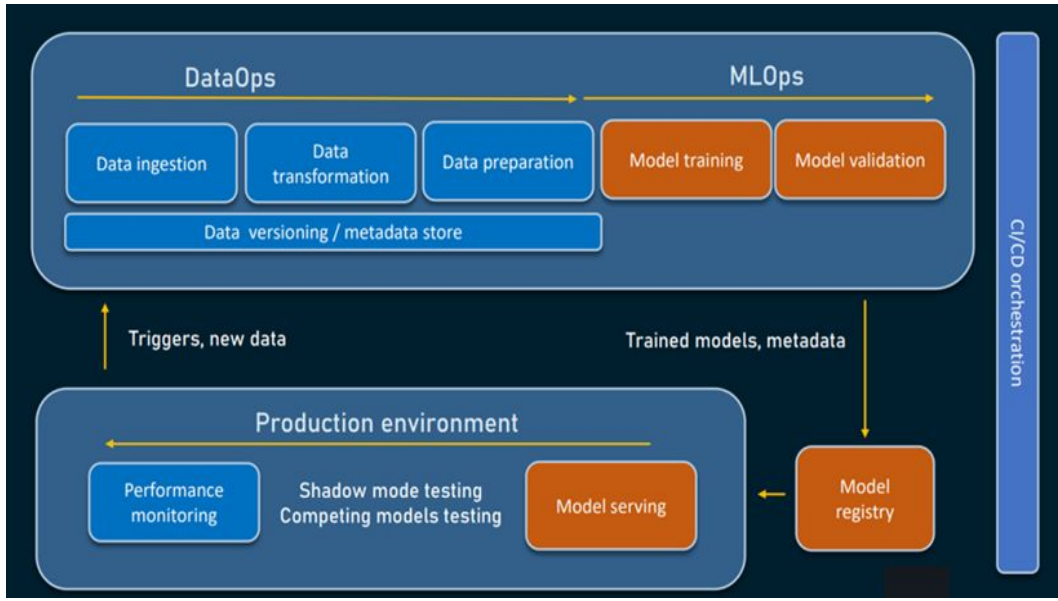
- **potential racial bias in the existing online library**

Wow, you have 6 balloons. Attention please! I also want to join the party.

DEPLOYMENT DEMONSTRATION

Future Steps

- Containerize the machine learning model in an environment where the model can be reproducible for better collaboration among ML teams
- Continue with data collecting and labeling to build a bigger dataset which include a good variation of different viewpoints, illumination conditions, and objects in different backgrounds to train our models.
- Optimise the transformer models (DETR/YOLO) using the optimum library by computing and storing parameters by their quantised counterparts
- Perform automated testing such as Smoke testing, Unit testing, Regression testing especially in early stages of development to reduce the number of bugs
- Perform CI/CD orchestration with automated pipelines for better streamlining of model deployment



Q & A