PROJECT REPORT

The purpose of this report is to assess my contribution and learning through implementing a project in Object Detection with transformers. I worked with team “Robot Dream,” which consisted of 5 individuals with diverse backgrounds, skills and experiences. The project encountered several challenges, for which I put in lots of efforts to research and made various attempts to resolve them. Although not all my efforts were successful, this report describes the process and challenges I faced during this project as well as the strategies I employed to identify and resolve problems, ultimately contributing to the team's success.

Project Overview

The team worked on a comprehensive end-to-end project, including data collection, model development, and deployment. We established the following project objectives:

- Develop 2 models for
    - detecting and counting balloons
    - detecting happy faces
- Create a web application where users can upload a picture receive a modified version of the image with bounding boxes around the identified balloons and happy faces, as well as the number of balloons present in the photo

The project was divided into 3 distinct phases:

- **Data collection and labeling**
    - created guidelines for balloon labeling consistency among team members.
    - collected data using a hybrid method, utilizing both pre-labeled public datasets and labelling our own photos
    - compare team members’ performances (human baseline) using Intersection over Union (IoU) to ensure adherence to labeling guidelines
    - to construct the final dataset, team members each added 200 images per week (100 for positive examples of balloons and 100 for negative examples of non-balloons) to Roboflow over a period of 3 weeks.
- **Model development**
    - Stopped data collection and used the current dataset of 2,645 images (2,245 for training, 302 for validation, and 98 for testing) to train the balloon detection model.
    - balloon detection model:
        - Research various options for balloon detection, including YOLO v7, DETR, and Detectron 2, but ultimately settled on using DETR and YOLO v5
        - Evaluated performance between the two models and chose DETR as the final model
    - Utilized a pre-built library called Pillow to detect faces in images and thereafter classify the faces using Convolutional Neural Networks and evaluated its performance.

- **Deployment**
    - Incorporated feedback from our supervisor and eliminated duplicates from the dataset, resulting in a final dataset of 2,376 images (2,018 for training, 358 for validation, and 38 for testing).
    - Retrained and re-evaluated both balloon detection models and switched to YOLO v5 as the final model
    - Built a Flask web application that allows users to upload an image and receive the following results:
        - a text displaying the number of balloons and happy faces present in the image
        - The uploaded image, and the result image with bounding boxes around balloons and happy faces are also saved
        - Utilized YOLO v5 for balloon detection and CNN-happy-face for happy face detection.

My role and accountabilities in phase 1 - Data collection and labeling

In this phase of the project, I contributed to building a dataset for balloons, while my other team member was responsible for creating guidelines for labeling. I suggested that our team use Roboflow as the tool for collaboration in checking, labeling, and building our database. Roboflow offers several advantages, such as

- Create dataset by
    - upload images and annotation from a local machine
    - browse and pull images from public datasets on Roboflow
    - upload regular images and add bounding boxes and annotations
- Divide the dataset in set of … ratio of your choice and export to various format (YOLO, COCO, Detectron 2) for use in building the model
- Provide statistics on the dataset including total images, number of images per class, size distribution, and histogram of Object Count by image.

As the team agreed to use Roboflow, I demonstrated how to use it by showing how to upload pre-labeled images, re-label them according to our guidelines, set up a collaborative workspace, and export the final dataset in different formats.

One team member found a public dataset of 4,000 images and divided that dataset into small batches, assigning each team member around 300-400 images per week to check, relabel, and add to our final dataset. Our goal was to have each team member add 600 images over 3 weeks. I personally contributed 706 images to the final dataset. In addition to the images from this public dataset, I also collected 64 unlabeled balloons and began labelling them myself, which later served as the baseline for comparing the IoU of other team members' performance. I also create an Xxcel sheet for other team members to add their results and compare the IoU.

Based on the rough guidelines created by my teammate, I began collecting data to add to Roboflow workspace. During this process, I had many questions that led to changes in the labeling guidelines. These questions included:

- Whether the file extension (avif, jfif, jpeg, pgn, jpg) made a difference in our dataset?
  → modification of guidelines to accept only jpg images
- Whether to add black and white or only color photos?
  → modification of guidelines to accept images in RGB, not greyscale
- Whether it was okay to include images with copyright/watermarks?
  → modification of guidelines to avoid watermarked images, download from proper sources and not steal any intellectual property
- Notice different labels from different dataset (Balloon, balloons, balloon)
  → modification of guidelines to set label name as "balloon", not Balloon or balloons or BalloonS
- Whether the balloons can have a pattern/print on them or just solid color?
  → modification of guidelines to allow for print/pattern on party balloons
- Noticing mis-labeled reflections on mirrors and shadows on walls in some images on public datasets
  → modification of guidelines to ignore balloons' reflection on other surfaces such as wall, mirror, etc

During my data collection process, I noticed some risks that might make it tricky to detect balloons, including: balloons with colors closely matching the background, such as white balloons on a white background; balloons on the edge of the photo; reflections of balloons on other surfaces; balloons that are too blurry; groups of balloons where there are too many overlapped balloons. Later, I prepared a draft presentation for the team to review and provide feedback on and included these risks in the presentation.

My role and accountabilities in phase 2 – Model Development

Prior to this project, I had no experience in working with Object Detection and limited proficiency in Python coding. This project was challenging for me as it required advanced programming knowledge. I faced many challenges but did my best to contribute to the team's progress. In this phase, I was tasked with working on the DETR model for balloon detection.

I began by testing the code using a temporary dataset of 1000 images. This was done to ensure that the code was running correctly before applying it to our larger dataset of 2645 images. The temporary dataset was downloaded from Roboflow in COCO format. To use DETR, I had to make some modifications in this dataset such as creating and renaming folders, modifying and rearranging files into the appropriate folders.

Before: Original structure of the dataset folder after downloading in COCO format from Roboflow:

- 1000 photos
  - train
    - _annotations.coco
    - all .jpg files for training images
  - valid

        - _annotations.coco
        - all .jpg files for training images

After: To prepare the dataset for use with DETR, the following modifications need to be made:

- Create a new folder called 'annotations' within the ‘1000 photos’ folder.
- Open and modify the _annotations.coco file in both the 'train' and 'valid' folders
    - Change the 'categories' field from "categories":[{"id":0,"name":"balloon","supercategory":"none"},{"id":1,"name":"balloon","supercategory":"balloon"}] to "categories":[{"id":0,"name":"balloon","supercategory":"N/A"}]
    - Replace all instances of “category_id":1 with “category_id":0
    - Save both files as custom_train.json and custom_val.json in the new 'annotations' folder
- Rename folders: ‘1000 photos’ to ‘custom’, ‘train’ to ‘train2017’ and ‘valid’ to ’val2017’
- The final structure of the dataset folder before using DETR should be as follows:
    - custom
        - annotations
            - custom_train.json
            - custom_val.json
        - train2017
            - all .jpg files for training images
        - val2017
            - all .jpg files for validation images

I utilized the code from https://github.com/woctezuma/finetune-detr/blob/master/finetune_detr.ipynb as a guide and adapted it to fit the requirements of our project. Initially, I faced several challenges during the implementation process.

- **Code didn’t run on my local machine:** It took me some time and multiple references to realize that my local machine didn’t have a GPU enabled graphics card. After realizing this, I decided to run the code on Google Colab to take advantage of the free GPU availability.
- **Run out of GPU free credit:** I have to run my models on several days as I ran out free GPU on Google Colab.
- **Successfully loading random image but fail to load annotations:** I realized that DETR required modifications to the dataset folders, so I made changes to the folder structure and file names, and was able to successfully load a random image from the training set and its annotations.
- **Model didn’t make any prediction on test images:** After quickly training the model for 3 epochs, it didn’t make any predictions on test images. I fixed this issue by modifying the 'categories' field and 'category_id' in the annotation files as mentioned above.
- **Results and models weren’t saved:** As I was new to using Google Colab, I wasn't aware that my work was saved on temporary session and any results obtained during that

session wouldn't be saved. However, since I was still trying to understand the code and get it to work correctly, it wasn't a major issue at first, as I could simply re-run the code if I made mistakes without affecting the actual dataset. Once the code executed properly and the model was able to make predictions, I mounted my Google Drive on Google Colab to ensure that all models and results were automatically saved for future use and evaluation.

- **Incorrect image size:** As per our guidelines, each image in the dataset must be at least 600 x 6000 pixels in size. However, DETR resized the images to have a minimum of 800 pixels. This resulted in the model missing the balloons when the image size was less than 800 pixels. I resolved this issue by setting the T.Resize (600) in my code and rerunning the code. This allowed the model to correctly detect the balloons.

Additionally, I modified the function 'plot_finetuned_results' to count the number of balloons in the image as follows:

```
def plot_finetuned_results(pil_img, prob=None, boxes=None):
    plt.figure(figsize=(16,10))
    plt.imshow(pil_img)
    ax = plt.gca()
    count_balloon = 0
    colors = COLORS * 100
    if prob is not None and boxes is not None:
      for p, (xmin, ymin, xmax, ymax), c in zip(prob, boxes.tolist(), colors):
          cl = p.argmax()
          label = finetuned_classes[cl]
          if label == "balloon":
            count_balloon +=1
            text = f'{finetuned_classes[cl]}: {p[cl]:0.2f}'
            ax.add_patch(plt.Rectangle((xmin, ymin), xmax - xmin, ymax - ymin, fill=False, color=c, linewidth=3))
            ax.text(xmin, ymin, text, fontsize=10, bbox=dict(facecolor='yellow', alpha=0.5))
    print("Number of balloons detected: {}".format(count_balloon))
    plt.axis('off')
    plt.show()
```

After resolving all issues and ensuring the code was functioning correctly, I began training DETR on our dataset of 2645 images. I ran the code multiple times with various epochs to find the optimal results. I initially trained the model for 20 epochs and observed that the mAP began to converge around 6 epochs, so I re-ran the model with 7 epochs and saved it as my final model. The results from this model were quite good, with mAP of around 0.68.

I set the threshold for accuracy for 75%, 85% and 95% respectively to see how the model performs. At 95% threshold, the model performed well in detecting balloons in small groups (2-7) and avoiding non-balloon objects, but had some difficulty accurately detecting multiple balloons in large groups or in less common positions, such as when balloons were blurry or far away, or when the balloon colors were too similar to the background.

I then discussed with my other team members to compare the results of our model predictions using DETR and YOLO v5. The team evaluated both models and agreed that, while the statistics of both models were similar, DETR performed better than YOLO v5 on our test images, especially in detecting difficult balloon situations. As a result, we selected DETR as our final

model for deployment. Additionally, I prepared a draft presentation for the team to add and modify their inputs

My role and accountabilities in phase 3 – Deployment

After receiving feedback from our supervisor about the possibility of duplicates in our dataset, which could affect the performance of YOLO v5, we decided to clean up the dataset and rerun the models. This could happen because many people were contributing to the same dataset and mistakes were bound to occur. I was assigned the task of preparing the final dataset by removing duplicates and images of less than 600 x 600 pixels. I had to go through 2645 images and removed 265 duplicates, which took me 1.5 days to complete. The new and final dataset was reduced to 2376 images.

After that, I reran my code for DETR on the new dataset, but observed that the model performed worse than before. I trained the model for 20 epochs and noticed that the mAP began to converge around 10 epochs and was around 0.35. The results for this model on test images were also much poorer than before.

As DETR did not perform well, YOLO outperformed DETR on this new dataset. Therefore, we decided to switch to using YOLO v5 as our final model for deployment. My other team member was responsible for building a Flask App in this phase.

Reference:

GitHub Repository for modeling - https://github.com/rashmi-carol-dsouza/robot-dream

GitHub Repository for deployment - https://github.com/rashmi-carol-dsouza/balloon-detection

Data Collection and Labeling Policy - https://docs.google.com/document/d/1JgQjHEJilRJTLoLUZWgXyAeWul9PTRYJto2xC7jVK6I/edit?usp=sharing

Roboflow Projects -

Week 1 - https://universe.roboflow.com/balloon-project/robot-dream-week1-positive-negative

Week 2 - https://universe.roboflow.com/balloon-project/robot-dream-week2

Week 3 - https://universe.roboflow.com/balloon-project/robot-dream-week3

Final dataset - https://universe.roboflow.com/balloon-emler/final-dataset-mhybw

Human baseline - https://docs.google.com/spreadsheets/d/1XLJ8ptX7vjaUYb5vDgWv5_sFxp4Fr5mU/edit#gid=2132659847

DETR model - https://github.com/rashmi-carol-dsouza/robot-dream/tree/main/DETR

App Demo - https://www.youtube.com/watch?v=R5XeVGh6ZoI