[[1]](#footnote-1)

**Object Detection and Labeling in Satellite Images**

Edward Simpson

May 2022

***Abstract*—** In this paper, I explore the DETR transformer model and application of satellite images to it. This includes how the DETR works and how I plan on implementing it with custom satellite datasets. Then the process for implementing a small, customized dataset directly onto the DETR model by fine tuning its already generated parameters.

***Index Terms*—** Transformers, Object Detection, Object Labeling, DETR, Encoder, Decoder, Fine Tuning

# **INTRODUCTION**

Object detection and labeling has been a very influential field in machine learning for quite some time. Allowing computers to correctly identify and label objects in images is extremely important in many real-life situations. A prominent technology utilizing object detection and labeling is self-driving cars. Being able to correctly detect and label objects is an essential function that makes this technology work.

This concept of detection and labeling can also be applied to satellite images. With a computer detecting and labeling objects, either generically or specifically will allow for countless applications across many fields.

Seeing how satellite cameras currently have coverage over the entire planet, using this technology will allow for data collection on an entirely different level. Information that could only be roughly estimated can now be way more precisely collected. Information like animal migration habits, street traffic, or preferred vehicle color can be more precisely estimated.

Although, there are still limitations on processing power and time. Searching every inch of the planet to count every vehicle is technically possible but would require a massive amount of time and processing power. Nevertheless, object detection and labeling are a crucial technology. This paper will explore the DETR transformer model developed by Facebook in May of 2020 [1].

# **Transformer Model**

The DETR model is a transformer model, in order to understand how it generally operates. A basic knowledge of transformers is required.

Essentially a transformer consists of an encoder and a decoder [3]. The encoder takes input, in our case image features, and generates a numerical representation of the image in the form of a tensor. The decoder takes the output of the encoder and the previous output of itself to then generate a prediction output.

Specifically, the encoder and decoder use a combination of layers to do this task. These include Multi-Head Self-Attention, Multi-Head Attention, and Feed-Forward layers.

# **Dataset**

After extensive research into various open-source datasets in satellite images. The optimal one found for object detection and labeling in satellite images is the DOTA dataset, created by various researchers [5]. This dataset includes images collected from Google Earth and CycloMedia, each image includes bounding boxes around objects and their respective labels. In this dataset there are around 18 categories for labeling. In total this dataset consists of 11,286 images containing 1,793,658 instances.

Unfortunately, due to compatibility and time constraints on training time, the DOTA dataset could not directly be used for the project. Instead, a branch off the dataset was used, this smaller dataset is called planes. This dataset was developed by an individual on Kaggle three years ago [8]. Inside of this dataset is solely images of planes and their detection boxes. Because of this the dataset is significantly smaller and easier to work with. It only contains around a thousand images, consisting solely of planes and their surroundings. This makes the dataset ideal for a small project that can be done on an average consumer computer.

# **Software**

The DETR transformer model, DEtection TRansformer, was developed by Facebook in May of 2020 [1]. The basic process for this model is to first send an image through a convolutional neural network. Then take the results from that and send it into the encoder. After this the information is processed through the decoder to determine bounding boxes and labels.

# **Training**

In this project the code to generate and run the model is written on google collab and a customized DETR model is pulled from GitHub. The code itself is based on fine-tuning code written by Woctezuma [9]. To clarify this model is not trained from scratch, as there are quite a lot of parameters. Because of this I will fine tune already generated parameters to try and fit my data. This process saves quite a lot of time and does not require such a large dataset, compared to training from the ground up.

The first step to fine tuning any model is to prepare the dataset that will be used. In this project I will using an augmented version of the planes dataset [8]. In order to properly run the dataset with DETR it must first be in the COCO format. In order to convert the dataset to this format I used roboflow, a database of datasets for the conversion [7]. Then manually uploaded the zipped planes-coco dataset into google collab. Once unzipped I did a quick test on the internals of the dataset to double check.

Once the dataset has been properly uploaded into the notebook, google collab, the next step is to import the customized version of the DETR dataset. As seen on the notebook this is done as an upload from GitHub. The reason the basic DETR model needed to be customized was due to compatibility issues with various libraries and their versions. The customization was kept to a minimum as to try and not degrade the overall quality of the model.

Once all the appropriate files are uploaded all that’s left is to fine tune the model. In this project I used 10 epochs of training, which took approximately 31 minutes. Once trained the fine-tuned parameters are saved as a checkpoint, which is referred to later when validating.

# **Results**

Chart, line chart

Description automatically generated Once the model is trained the results achieved are quite nice. In the diagrams representing the statistics of the training, the dashed line represents the testing set and the full line represent the training set. The diagram below shows the overall loss achieved while training, final values being around 4.5.

Chart, line chart

Description automatically generated The next diagram shows the loss of the bounding boxes generated, throughout training.

As seen by the diagrams, across 10 epochs the loss values significantly decrease. Ten epochs appears to be optimal for time constraints, as the decrease of loss appears to slow down after the 10th epoch.

# **Future**

This project has been a very fulling experience at all stages. Throughout this project I have learned a ton about transformer models, datasets, and implementation. Overall, with setbacks in consideration, I think I did well on this project. Even though I was only able to classify one type of object in satellite images, the foundation is there to apply to larger and larger datasets.

For future work on this project, I would go about increasing categories detected in each image. Of course, this would mean finding a larger and better dataset to train on. Also, this would require a better computer, in order to reduce the length of time required for training. Other than that, for future work, increasing overall model accuracy would be beneficial.

**References**

[1] "DETR". *Huggingface.Co*, 2022, https://huggingface.co/docs/transformers/model\_doc/detr.

[2] Carion, Nicolas et al. "End-To-End Object Detection With Transformers". *Corr*, 2020. *Arvix*, https://arxiv.org/pdf/2005.12872.pdf. Accessed 31 Mar 2022.

[3] Cristina, Stefania. "The Transformer Model". *Machine Learning Mastery*, 2022, <https://machinelearningmastery.com/the-transformer-model/>.

[4] Ding, Jian κ.ά. ‘Object Detection in Aerial Images: A Large-Scale Benchmark and Challenges’. *arXiv [cs.CV]* 2021. Web.

[5] Xia, Gui-Song κ.ά. ‘DOTA: A Large-Scale Dataset for Object Detection in Aerial Images’. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. N.p., 2018. Print.

[6] ‘Learning RoI Transformer for Detecting Oriented Objects in Aerial Images’. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. N.p., 2019. Print.

[7] *How to convert openimages CSV to coco JSON*. Roboflow. (n.d.). Retrieved May 5, 2022, from <https://roboflow.com/convert/openimages-csv-to-coco-json>

[8] aceofspades914. (2019, April 22). *CGI planes in satellite imagery w/ bboxes*. Kaggle. Retrieved May 5, 2022, from https://www.kaggle.com/datasets/aceofspades914/cgi-planes-in-satellite-imagery-w-bboxes?resource=download

[9] Woctezuma. (n.d.). *WOCTEZUMA/finetune-detr: Fine-tune Facebook's detr (detection transformer) on colaboratory.* GitHub. Retrieved May 5, 2022, from https://github.com/woctezuma/finetune-detr

1. . [↑](#footnote-ref-1)