Object Localisation using Textual Description

FNU Vrinda(34020582), Guangshi Xu (31498869), Tanay Joshi (33160057)

Abstract

This project attempts to create and analyse a new architecture for Object Detection based on textual prompts. We draw inspiration from SOTA architectures for Object Detection and replace the Transformer layer with a CLIP (Contrastive Language-Image Pre-training) model that uses a vision transformer paired with a transformer-based language model to learn unified representations of images and text. We work on the MS COCO dataset and attempt to create a different architecture for this problem and analyse the trade-off of this modification on the performance of the model.

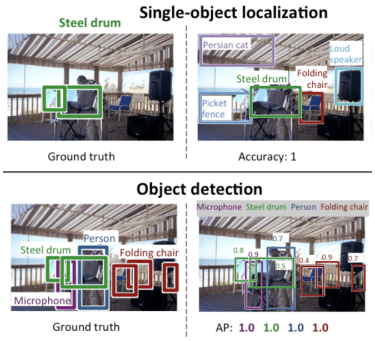

Figure 1. Object localisation vs detection

1. Introduction

We are working on an object localization problem using textual prompts. This means we are trying to identify and localize objects in images based on natural language instructions provided by a user. This is a very popular and well-researched problem with diverse real-life applications such as autonomous vehicles and surveillance.

Object Localisation architectures typically involve identifying regions of interest in the image and then using transformers to process the information in the area of interest. Transformer is a type of neural network architecture designed for sequence-to-sequence tasks, featuring self-attention mechanisms for efficient information processing. They excel at capturing long-range dependencies in sequential data, making them well-suited for tasks like natural language processing and, more recently, computer vision. The key innovation in transformers is the self-attention mechanism, allowing the model to weigh different parts of the input sequence differently during processing. This capability enables transformers to efficiently process and understand complex relationships within the data, making them increasingly popular in various machine learning applications, including object detection.

While studying existing SOTA architectures for this problem, we came across the MDETR [4] paper which used a highly complex architecture comprising models like EfficientNet and BERTA to process the text and image inputs and then used a Transformer to encode/decode positional and other information of the image such that the model can can answer several complex questions asked by the user.

Machine Learning and Artificial Intelligence Engineers are always researching the trade off between the complexity of a model and its final accuracy. Simpler models are easier to interpret, maintain and customise but may not yield results that are as good as that of complex models which are very customised to the specific problem they are designed to solve. While reading the aforementioned MDETR [4] paper, we felt that although it yielded promising results, it was very complex to implement.

There are already many promising models like YOLO [9] which work great for Object Detection but we since we wanted to do Object Localisation using Textual Descriptions, we were interested in finding a model that worked well for the same. This is when we came across the CLIP model by Open AI; it is a breakthrough in multimodal AI, bridging language and vision understanding. It creates unified embeddings for text and images in a shared space and excels in capturing intricate relationships between text and images, offering versatility for tasks like image classification and zero-shot learning. CLIP is pre-trained on large datasets and its architecture centers around a vision transformer (ViT) paired with a transformer-based language model. The ViT processes image data, while the language model handles text. Both modalities share a common embedding space, allowing direct comparisons.

Since pre-trained CLIP models are readily available for use, we wanted to experiment with the possible application of CLIP to solve the Object Localisation problem and in turn simplify the overall architecture. This intuition led us to come up with a simpler architecture of our own which we

will explain further in upcoming sections.

Through this project, we aim to implement this architecture and evaluate its performance against existing benchmark models and understand the trade-off between model complexity and accuracy.

2. Related Work

Object localization based on textual descriptions has garnered significant attention in recent times, bridging the gap between Computer Vision and Natural Language Processing. This section explores key approaches that have contributed to advancements in this domain.

Modular Approaches:

1. **MDETR (Modulated Detection for End-to-End Multi-Modal Understanding):**
   A notable model in this area is MDETR [4] . This model offers an end-to-end multi-modal understanding framework capable of detecting objects in images conditioned on raw text queries. It employs a transformer-based architecture to fuse visual and textual information at early processing stages, enabling robust object recognition guided by language. MDETR has demonstrated state-of-the-art performance on various multi-modal understanding tasks, including phrase grounding and referring expression comprehension, emphasizing its effectiveness in localizing objects based on textual descriptions.

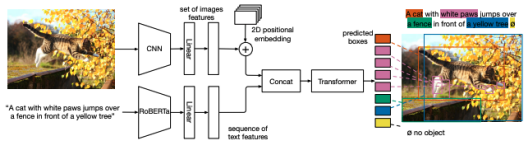

Figure 2. MDETR Architecture

2. **DETR (Deformable DETR for End-to-End Object Detection):**
   A foundational model, DETR [7] employs a transformer-based architecture for object detection, directly predicting bounding boxes and class labels for objects in an image. While not specifically designed for text-guided localization, DETR's architecture has proven adaptable for multi-modal tasks.
3. **ViLBERT (Vision-and-Language BERT):**
   ViLBERT [2] is a pre-trained model for multi-modal tasks that combines BERT for language understanding with a multi-layer vision encoder for visual understanding. It has demonstrated success in tasks like visual question answering, and its ability to integrate text and image information holds potential for text-based localization.
4. **FindIt:**
   FindIt [11] is a unified model for object localization that effectively integrates visual and textual information using a multi-scale fusion module. This module combines features extracted from different image scales with text representations, enabling the model to handle diverse natural language queries for object localization. FindIt leverages a standard object detector and uses a cross-attention mechanism to predict bounding boxes and class labels for localized objects. This approach simplifies the model architecture and training process while achieving state-of-the-art performance on localization tasks.
5. **YOLO (You Only Look Once):**
   YOLO [9] is a widely recognized model designed for generalized object detection, and while it is not explicitly tailored for text-based localization, its efficiency and speed make it noteworthy in the broader context of object detection. YOLO divides images into a grid and predicts bounding boxes and class probabilities for each grid cell in a single pass. This approach, coupled with anchor boxes, allows YOLO to accurately detect objects of various sizes within an image. While not inherently focused on text-guided localization, YOLO's ability to provide real-time object detection lays the foundation for future exploration in combining text and image modalities for precise object localization based on textual descriptions.

These approaches highlight the ongoing progress in multi-modal understanding, with models like MDETR demonstrating significant advancements in object localization based on textual descriptions. Future research directions encompass exploring model architectures to enhance cross-modal understanding further, incorporating contextual information, and addressing model interpretability.

3. Methods

In this section we focus on describing our proposed model in detail. We also describe the loss function we use to evaluate the performance of the model.

3.1. Model

In the context of existing literature such as MDETR [4], which addresses intricate queries involving detailed positional parameters, and contrasting models like YOLO, tailored for more generalized object detection tasks, we aimed to develop a model with an intermediate capability. This model strives to handle both general and specific queries, striking a balance between complexity and inclusivity.

Our exploration began with an examination of the MS COCO Dataset [6], which comprises of image descriptions, bounding boxes around identified objects and a category for each bounding box. We wanted to work with descriptions

that were detailed at the level of each bounding box such as in the RefCoco Dataset [5]. However, RefCoco [5] only identified limited objects in the image.

We recognised that at the moment there is no benchmark dataset that identifies all objects in an image like Coco [6] and gives detailed annotations for each bounding box like Refcoco [5]. As a result we decided to work with the MS COCO dataset for training cycle to optimise Object Detection and have come up with a small test dataset of up to 25 images taken from the COCO dataset where we added detailed captions for each bounding box to test our localisation capabilities.

Given a caption, extracting only the primary entity from these annotations risked losing valuable contextual information. We sought to streamline user queries, eliminating the necessity for the user descriptions needing to be as detailed as the annotations present in the dataset. For example, if the dataset describes a cat as a "white cat with a blue bell," our model should identify the same cat based on queries like "cat," "white cat," "cat with a blue bell," or "white cat with a blue bell." This implies that querying "cat" should highlight all cats in an image, while specific queries should isolate the cat fitting the given description.

To implement this, we propose leveraging Natural Language Processing (NLP) to analyze the text annotations in the input dataset. This analysis will generate multiple descriptions for the same bounding box, ensuring users can input varied descriptions of objects and obtain corresponding bounding boxes.

Due to our deliberate exclusion of complex Transformer architectures that can handle positional encoding, our model's handling capacity is constrained.

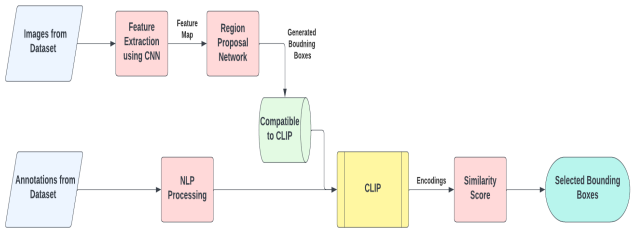

Figure 3. Data Flow

The depicted image elucidates the training-time workflow of our model, as detailed below:

1. **Feature Map Generation:** We commence by constructing a feature map from the dataset images utilizing a Convolutional Neural Network (CNN). ResNet50 [3] is our chosen CNN architecture, due to its Residual Learning property and complex depth. We also experimented with a VGG model and found that the ResNet50 model created a more extensive feature map and yielded better results. We are also experimenting the impact of using various Feature Map extraction models on the final performance.

Model	ResNet50	VGG
Size of Feature Map	548 MB	89 MB

Table 1. Comparison of Feature Map Sizes

2. **Region Proposal Generation:** Following feature extraction, the process involves passing the obtained features through a Region Proposal Network(RPN) [10]. This step generates precise region proposals (bounding boxes) for objects detected in the image, accompanied by Objectness scores for each bounding box. Leveraging the generated feature map, we implemented our custom RPN comprising the following layers:
   (a) A Convolutional layer with 512 hidden layers and a filter size of 3x3 (with added padding).
   (b) Dropout was applied with a probability of p = 0.3 to mitigate the risk of overfitting.
   (c) Another Convolutional layer with 512 hidden layers and a filter size of 1x1 was employed, utilizing nine different anchor points (comprising three different scales and three different aspect ratios) to compute the Objectness score.
   (d) Additionally, a Convolutional layer with 512 hidden layers and a filter size of 1x1 was incorporated, utilizing 36 different anchor points to generate bounding boxes.
3. **Bounding Box Filtering:** Subsequently, we use the generated Objectness scores to filter out the generated bounding boxes based on a chosen confidence threshold. We also use a Non-Maximum Suppression threshold to filter out the redundant bounding boxes.
4. **Cropping Region Proposals:** In the process of refining region proposals, we employ the bounding box coordinates of the selected proposals to crop the original image. This cropping operation is essential for obtaining image inputs so that the input is compatible with subsequent stages of the pipeline, particularly when utilizing a CLIP model. The subsequent step in the pipeline relies on image inputs, and the cropping of the original image ensures compatibility with the requirements of the CLIP model.
5. **Generating Permutations of Annotations:** Due to the limitations of the MS COCO dataset [6] described above, we have used the custom dataset with the detailed bounding box captions as the ground truth image. Employing Natural Language Processing (NLP) on these annotations, we systematically generate permutations of descriptions, ensuring the inclusion of an object/noun within each annotation. Consequently, for each image

in the custom dataset, we compile a comprehensive list of annotations, providing diverse and nuanced ways to describe all the objects present in the image. For our experiment, we use Spacy, which is renowned for its efficiency and accuracy in providing robust linguistic annotations during NLP tasks.

6. **Embedding and Similarity Evaluation using the CLIP model:** Following the acquisition of cropped regions of interest and permuted annotations, the integration of a CLIP model becomes the subsequent step. Recognizing the resource-intensive nature of training CLIP models, we opt for a pre-trained CLIP model (ViT-B/32) to fulfill our requirements. In the ViT-B/32 model, "B" signifies the model's size, and "32" denotes the input image resolution of 32 pixels per patch. The CLIP model processes the text permutations and the cropped images, to generate numerical embeddings for both. Guided by similarity scores between text and image embeddings, we identify images that surpass a specified threshold of similarity. Subsequently, bounding boxes are created around the selected images to emphasize the final objects identified by our model.

The aforementioned workflow is tailored for training and once this model is well-trained on scaled computing resources, we can build applications on top of this to take queries from users real-time to identify their desired objects in images.

This comprehensive model description provides a clear understanding of the intricate processes involved.

3.2. Loss and Evaluation Metrics

Loss:

One integral aspect of our training process involves the Region Proposal Network (RPN), a pivotal component providing bounding boxes encompassing regions of interest along with associated Objectness scores for each bounding box. To optimize the performance of our RPN, we employ two key loss metrics for model evaluation:

1. **Objectness Loss:** The Objectness loss trains the RPN to discern whether an anchor corresponds to an object or not, essentially distinguishing between foreground (object) and background. In this binary classification task, if a bounding box has class labels in the ground truth, its Objectness score is set to 1; otherwise, for negative bounding boxes without class labels, the score is 0. We utilize binary cross-entropy with logits as the loss function.
2. **Regression Loss:** The regression loss fine-tunes the shape and size of predicted bounding boxes, ensuring the network predicts boxes that closely align with the ground truth.

Evaluation Metrics:

The model output consists of bounding boxes representing objects identified based on user queries. We employ the Intersection over Union (IoU) metric to assess our overall architecture. IoU quantifies the degree of overlap between predicted and actual object regions.

The formula for IoU is given by:

IoU =

$$\frac{\text{Area of Intersection}}{\text{Area of Union}}$$

A higher IoU indicates better alignment between predicted and ground truth bounding boxes.

(CHECK THIS) For multiple images, we calculate the mean IoU (mIoU), averaging the IoU scores of all individual predictions:

Mean IoU (mIoU): =

$$\frac{1}{N}\sum_{i=1}^{N}\text{IoU}_i$$

(where N is the total number of predictions)

Ultimately, our goal is to comprehend the impact of architectural changes on the results, acknowledging that achieving comparable accuracy to state-of-the-art (SOTA) results may not be feasible.

4. Dataset

The dataset we plan to use is the MS COCO (Microsoft Common Objects in Context) [6] dataset. It comprises a collection of over 118,000 training images, 5000 validation images and 41,000 test images- each depicting diverse scenes and containing annotations for 80 object categories. These annotations include precise bounding boxes, which will help with object detection and localization, along with image captions, enabling natural language understanding tasks.

Due to the lack of strong computation resources and limited availability of GPUs, we will not be able to train our model on the entire dataset and have used 5000 images from the Training dataset.

As mentioned above, once we have trained our Region Proposal Network [10], we use a small dataset of 25 images generated by us from the COCO dataset [6], customised to our problem statement and to evaluate the working of the CLIP model.

Preprocessing the Dataset:

1. We normalised the dataset by subtracting the mean of the images from the dataset.
2. We removed the grey-scale images in the dataset to ensure we have a uniform size and category of images to work with.

5. Training

We are training our RPN [10] using Batch Gradient Descent(batch size = 10) and the Adam Optimizer [1]. The models are coded using PyTorch and we are using pre-trained CLIP (ViT-B/32) and ResNet models due to limitation of resources.

We use a learning rate schedule starting at 1e-3 and use an Exponential Learning Rate Scheduler provided by PyTorch to scale down the learning rate by 0.01 at every epoch. We use a batch size of 10 images so as to speed up the process without overloading the compute resources. We have trained our model for 100 epochs. We also added an early stopping condition which stops our training cycle when we don't see much decrease in the loss values. The loss used to train our RPN has been regularised using L2 regularisation to avoid overfitting.

During train time, we compare the generated regions of interest with the annotations of the MS COCO dataset [6]. However, during test time, we use some user-generated queries to compare the similarity to enable user-level control for object localisation.

6. Experiments and Results

6.1. Experiments

We experimented with various Convolutional Neural Networks to extract Feature Maps and compared their impact on the final performance of the model.

Model	Loss
ResNet18	10.558
ResNet50	5.23
ResNet101	5.89

Table 2. Comparison of Convolutional Neural Networks

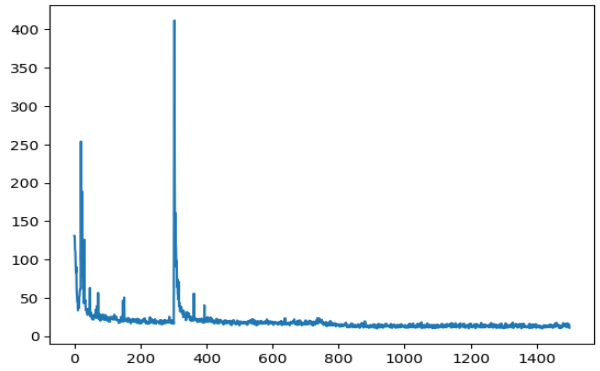

Figure 4. ResNet18 Loss

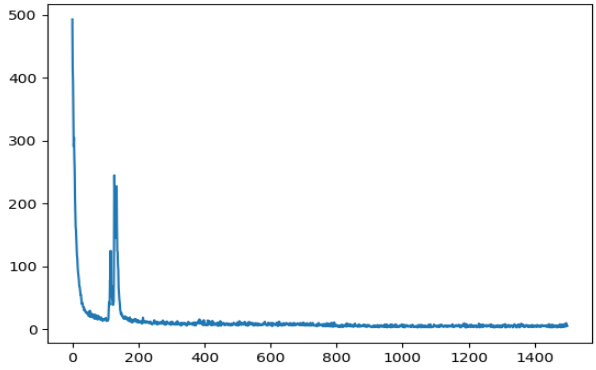

Figure 5. ResNet50 Loss

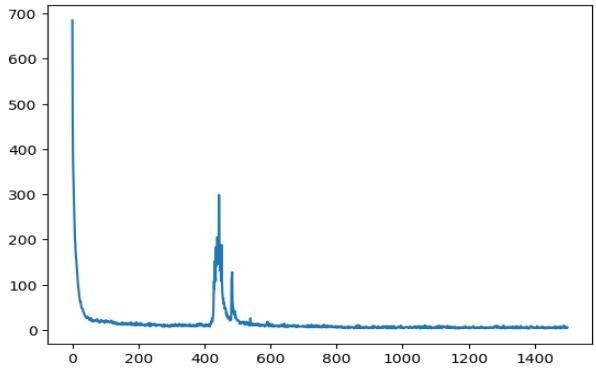

Figure 6. ResNet101 Loss

From this it is evident that the ResNet50 model was able to extract the features in the best way to optimise the final result of the model.

We also experimented with various hyper-parameters including batch size, learning rate decay, number of epochs and learning rate. We have included the results of the experimentation with various learning rates on our final performance.

Learning Rate	RPN Loss
1e-2	250.00
1e-3	5.80
5e-3	22.46

Table 3. Comparison of Learning Rates

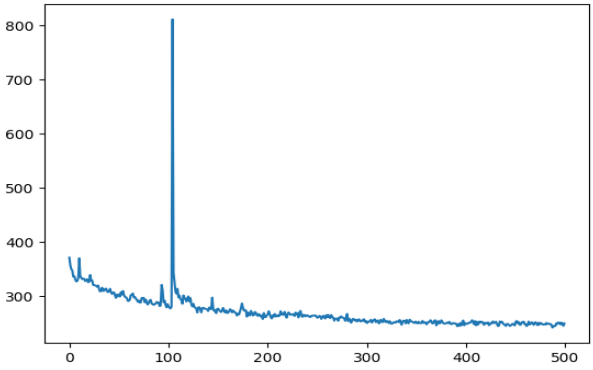

Figure 7. Learning Rate: 1e-2

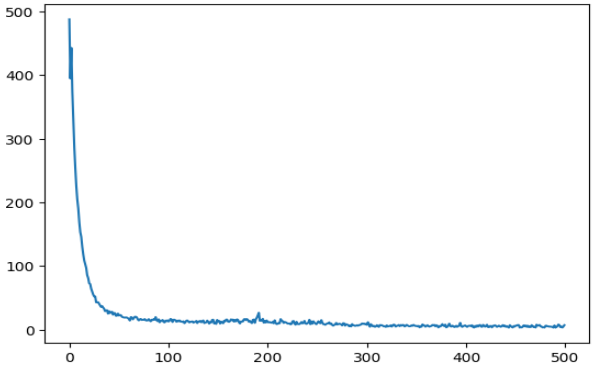

Figure 8. Learning Rate: 1e-3

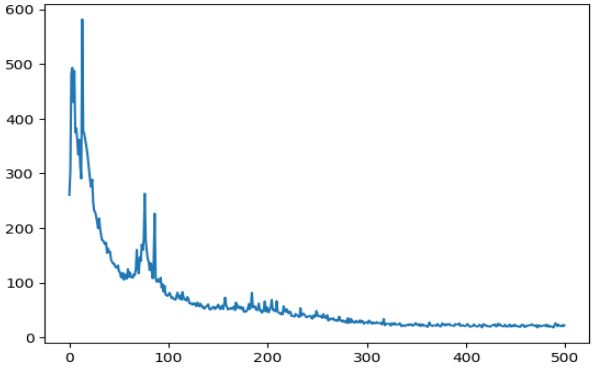

Figure 9. Learning Rate: 5e-3

From these results(trained on 100 images for 500 epochs for demonstration), it is evident that the learning rate 1e-3 worked the best, which is why it has been used to calculate the final result.

Aside from this, we also played around with the confidence and NMS thresholds in our RPN to see which combination worked best.

Few combinations have been demonstrated below:

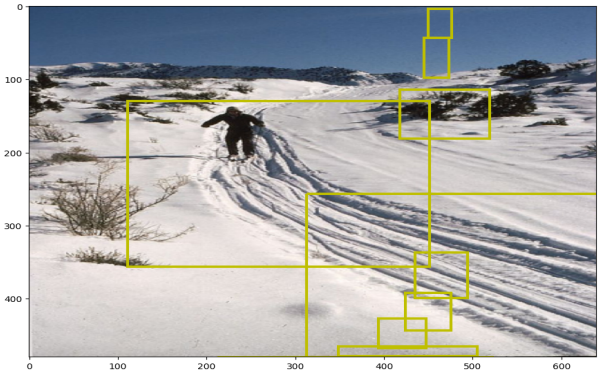

Figure 10. Confidence Threshold = 70%, NMS Threshold = 10%

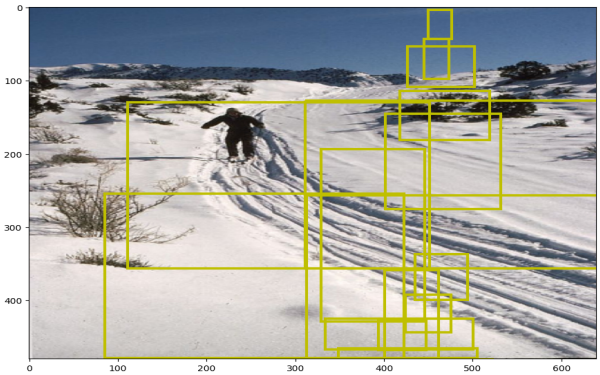

Figure 11. Confidence Threshold = 70%, NMS Threshold = 30%

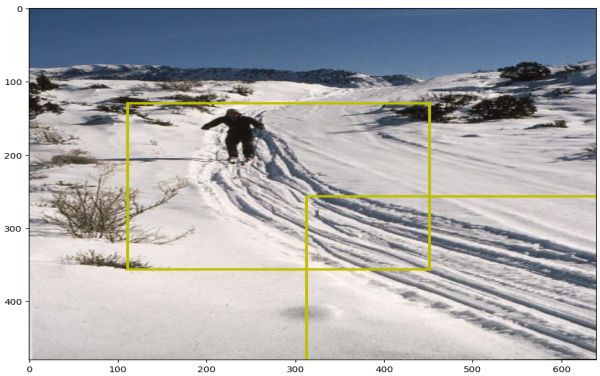

Figure 12. Confidence Threshold = 90%, NMS Threshold = 10%

The results in our experiment have been generated using Confidence Threshold = 80%, NMS Threshold = 10%.

Finally, we also moved around the threshold applied on the similarity score of CLIP used to highlight the final bounding boxes.

The result of some of these variations have been demonstrated below:

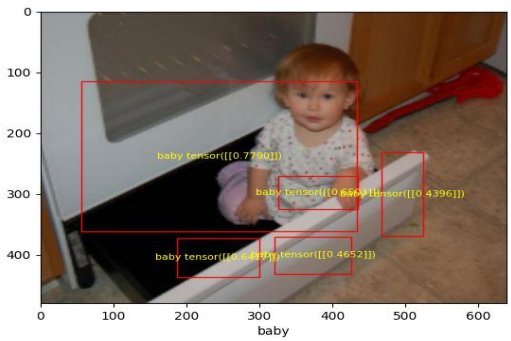

Figure 13. Similarity Threshold = 0

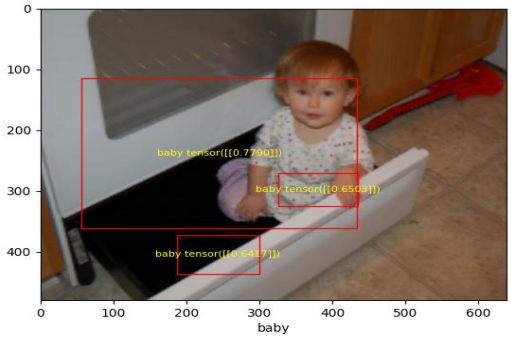

Figure 14. Similarity Threshold = 0.5

Aside from the experimentation described above, we also tried out a variety of batch sizes, decay values and Optimizers to find the best ones.

We were able to find the values of various hyperparameters that worked best for our use-case and have used the same to calculate our results.

6.2. Results

After running our Region Proposal Network, we were able to generate bounding boxes around Regions of Interest.

Demonstrated below is one of the outputs of our Region Proposal Network viz. the ground truth image:

As visible, our RPN had started to propose promising Regions of Interest, but due to a lack of computing resources we were only able to train our RPN model to a certain extent and hence could not minimise the loss to the maximum extent.

The other component of our architecture was our NLP component where we generated the permutations for every annotation associated with the bounding box of our custom dataset. Using Spacy, we were able to focus on the outputs that were focused on the objects.

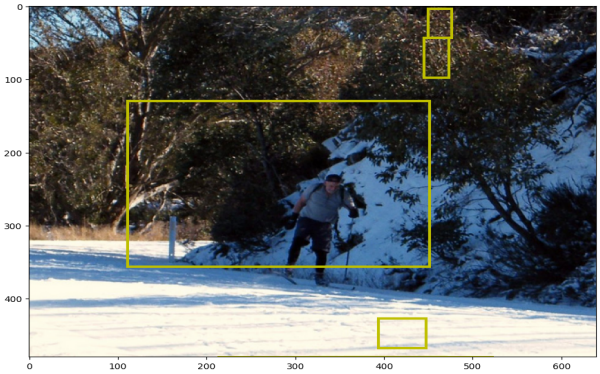

Figure 15. RPN Proposals

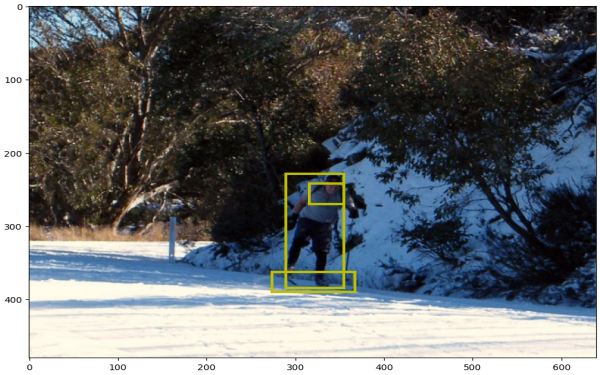

Figure 16. Ground Truth

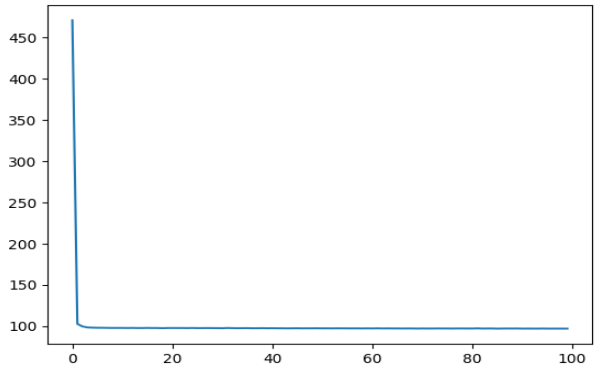

Figure 17. Training Loss

Ultimately, when we ran our CLIP [8] model, we were able to calculate the similarity scores between each proposed bounding box from our Region Proposal Network and every permuted caption. Below is a small example of the similarity scores from CLIP [8] given a region proposal generated from our RPN:

Leveraging insights garnered from the aforementioned example and additional experiments, notable patterns were observed:

```
1 permute_text("A man wearing a leather jacket")

['man',
 'leather',
 'jacket',
 'man leather',
 'man jacket',
 'leather man',
 'leather jacket',
 'jacket man',
 'jacket leather',
 'man leather jacket',
 'man jacket leather',
 'leather man jacket',
 'leather jacket man',
 'jacket man leather',
 'jacket leather man']
```

Figure 18. Demonstration of Permutations

```
Top predictions:

Value: 71.4375, Index: a person in black sitting on a motorcycle
Value: 14.75, Index: motorcycle
Value: 6.96484375, Index: bike
Value: 6.2421875, Index: red motorcycle
Value: 0.296630859375, Index: vehicle
Value: 0.227294921875, Index: a person in black
Value: 0.0515441894531125, Index: wheel
Value: 0.0162200927734375, Index: ship
```

Figure 19. Demonstration of Permutations

1. CLIP [8] exhibits a preference for detailed descriptions, where greater detail correlates with improved similarity scores.
2. Colors do not significantly impact CLIP's [8] similarity scores and, in certain instances, may even result in a decrease.
3. CLIP [8] assigns considerable importance to the primary object visible in the image.

Further validation of these observations can be conducted with an expanded dataset and enhanced computational resources.

In our experimental procedure, the computation of the final Intersection over Union (IoU) score involved the following steps:

1. For every ground truth bounding box within an image, permutations were generated based on the custom annotations.
2. Utilizing CLIP [8], similarity scores were determined between each generated permutation and all region proposal images.
3. The annotation with the highest similarity score was identified, and the associated original ground truth bounding box was retrieved.
4. Using this ground truth bounding box and the generated region proposal, IoU scores were calculated for each image.

For our experiment, we selected all region proposals that had an IoU score greater than 0.5 and found our mean IoU to be = 0.496.

This is a sample image with the predicted region proposals that have IoU greater than 0.5 with the ground truth bounding boxes and the annotations best used to describe them:

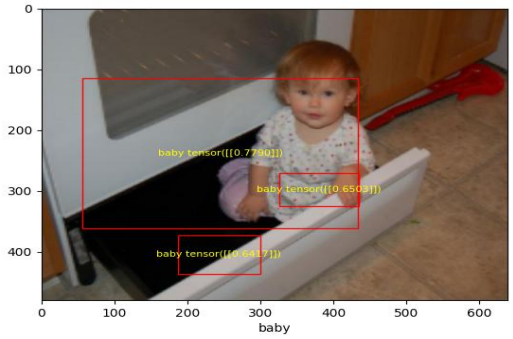

Figure 20. Region Proposals and Sample Annotation that can be used to identify it

7. Conclusion

Based on the conducted experiments, the following conclusions can be drawn

1. CLIP exhibits moderate performance in Object Detection, showcasing better results with detailed descriptions and comparatively less promising outputs with single-worded descriptions. For simpler object classification tasks, models like YOLO [9] may be more suitable.
2. Limited computational resources and dataset coverage prevented the observation of optimized results. However, it is evident that with adequate training and information, the CLIP [8] model has the potential to perform well in Object Localization based on Textual Description applications.
3. The absence of positional encoding in CLIP [8] suggests potential limitations in accuracy compared to state-of-the-art models like MDETR [4]. Nevertheless, conclusive statements are challenging due to the restricted training of our architecture.

In conclusion, the future of CLIP [8] for Object Localization looks promising, and with sufficient information and resources, promising results can be anticipated.

References

[1] Jimmy Lei Ba Diederik P. Kingma. Adam: A method for stochastic optimization, 2015. 5

[2] Devi Parikh Stefan Lee Jiasen Lu, Dhruv Batra. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks, 2019. 2

[3] Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition, 2015. 3

[4] Aishwarya Kamath. Mdetr- modulated detection for end-to-end multi-modal understanding, 2021. State of the Art Object Localisation solution. 1, 2, 8

[5] Shan Yang Alexander C. Berg Tamara L. Berg Licheng Yu, Patrick Poirson. Modeling context in referring expressions, 2016. 3

[6] Tsung-Yi Lin. Microsoft coco: Common objects in context. 2015. 2, 3, 4, 5

[7] Gabriel Synnaeve Nicolas Usunier Alexander Kirillov Sergey Zagoruyko Nicolas Carion, Francisco Massa. End-to-end object detection with transformers, 2020. 2

[8] Alec Radford. Learning transferable visual models from natural language supervision, 2021. CLIP Model. 7, 8

[9] Joseph Redmon. You only look once: Unified, real-time object detection. 2016. 1, 2, 8

[10] Shaoqing Ren. Fasterr-cnn: Towards real-time object detection with region proposal networks. 2016. 3, 4, 5

[11] Wei Li AJ Piergiovanni Mohammad Saffar Anelia Angelova Weicheng Kuo, Fred Bertsch. Findit: Generalized localization with natural language queries, 2020. 2