Robot Grasp Detection using Detection Transformer

Srinidhi Kalgundi Srinivas
Department of Electrical and Computer Engineering
A59010584

Sumukh S Badam
Department of Electrical and Computer Engineering
A59013970

Abstract

In robotics, having a robot grasp an object in a cluttered environment is a challenging problem. With the advent of Convolutional Neural Networks (CNNs), many work such as [3, 4] has been proposed to tackle the problem of detecting the right grasp for an object. Several works such as [1] combine grasp detection and object detection to predict grasp candidates that are assigned to specific objects in the scene. In our project, we propose to use Detection Transformers(DETR)[2] to perform grasp detection.

1 Introduction

Simple task like walking, kicking a ball and Grasping a object is very trivial for humans but it's daunting to get robots to perform. Robotic grasping is highly researched topic and there are many literature for this simple task. We define the task as given the parallel arm we predict the bounding box. The bounding box is normal to the arm and only one side of the bounding side is used to grasp the object. Reaching the co ordinates is not part of the problem statement. We further simply this task by having single object in the scene. Data we are using is from monocular camera and we do not have any depth information.

Even thought this task is similar to the object detection task where object is detected and the bounding box is applied around the box or pixel wise segmentation is performed. Robotic grasp is similar to object detection but it differ in one point. Bounding box predicted in the object detection task is always perpendicular where as in the object grasping task the bounding box also has a rotation component. So the network has to predict two things bounding box and rotation transformation of the bounding box.

2 Related work

This project is primarily based on the utilizing transformers architecture for object detection in a robotic environment.

The ability to locate objects in the scene and successfully determine the grasping pose is essential for robust robotic grasping. Grasp detection used images taken from the camera mounted overhead and infers pose for the grippers from the image. There have been several different attempts at successfully determining the pose. Geometry driven methods [8][9] focus on the geometrical shapes which helps in prediction of grasping points. The main drawback of these works is that they rely on the CAD models of the graspable objects which may not always be available.

Many Deep Learning based methods have been proposed to detect grasping. [1] utilizes a Resnet-101[7] backbone with individual heads for bounding box detection and orientation classification.

Feature Pyramid Network (FPN)[10] is used on top of Resnet. The Bounding box detection head uses Faster R-CNN [11] object detector which has Region Proposal Networks(RPN).

Prior to this work, other data driven methods have been used in grasp detection. Methods in [12] [13] took a supervised learning approach to predict multiple grasp candidates for a single object. [18] proposed a network for grasp detection based on two stage object detectors which used the RPN approach.

[14] uses a generative architecture that predicts grasps for individual pixels in the form of *quality, angle width*. Approaches using Reinforcement Learning [15] [16] have also been considered on both real and simulated robots to perform multiple grasp attempts and improve grasp detection with closed loop analysis.

2.1 Transformers

The field of Natural Language Processing was taken over by Transformers introduced by Vaswani *et al*[17]. It was introduced as attention-based building block for machine translation. Transformers used self-attention layers that are extensions to attention layers that were introduced in [17]. Attention layers aggregate information from the entire input sequence and helps in establishing long range relationship between the words in the sequences. Attention based models perform global computations and have perfect memory which has made them more suitable than RNNs for long sequences.

3 Dataset

- **Cornell Grasp Dataset:**[2]
  The Cornell dataset contains 885 RGB-D images of graspable objects and each iamge of size 640x480px. The Cornell Dataset is considered to be a gold standard dataset for Grasp detection. Since the size of the dataset is small, we will be augmenting the dataset with random crops and rotations.
- **OCID Dataset:**6
  The authors of [1] propose an extension to their work using OCID dataset. We will compare our results on OCID dataset as well. OCID grasp consists of 1763 selected images with over 11.4k segmented object masks and more than 75k hand-annotated grasp candidates, and each object is classified into one of 31 classes.

Note: Due to low time, we were only able to run experiments on Cornell Grasp Dataset. However, we have written the Dataloader for OCID dataset as well and the same is submitted along with the code.

4 Architecture

We propose the robotic grasping architecture using Detection Transformers [2]. We use similar approach as [2] where our decoder predicts bounding box, orientation and classifies the object. First two outputs are similar to other networks mentioned in the Related Work section and for robot grasping we add the object classification part similar to DETR where the classes we use are a binary *present* or *absent*. We used cornell dataset which lacked information about the object in the scene so we only used two output instead of predicting for every class and no object class.

4.1 Data format

The output of our dataloader is in the format as specified below to enable easier handling of data by the network.

$$\{orientation(\theta), x1, y1, x2, y2\}$$

For the orientation, Cornell Dataset divides the possible orientations into 20 bins and provides them as θ which is learned by the network as classes making it a 20+1(background) class classification problem. Each of these 20 intervals are represented by their mean values. The output of the network for θ are logits for softmax layer represented as probability distribution of the output classes.

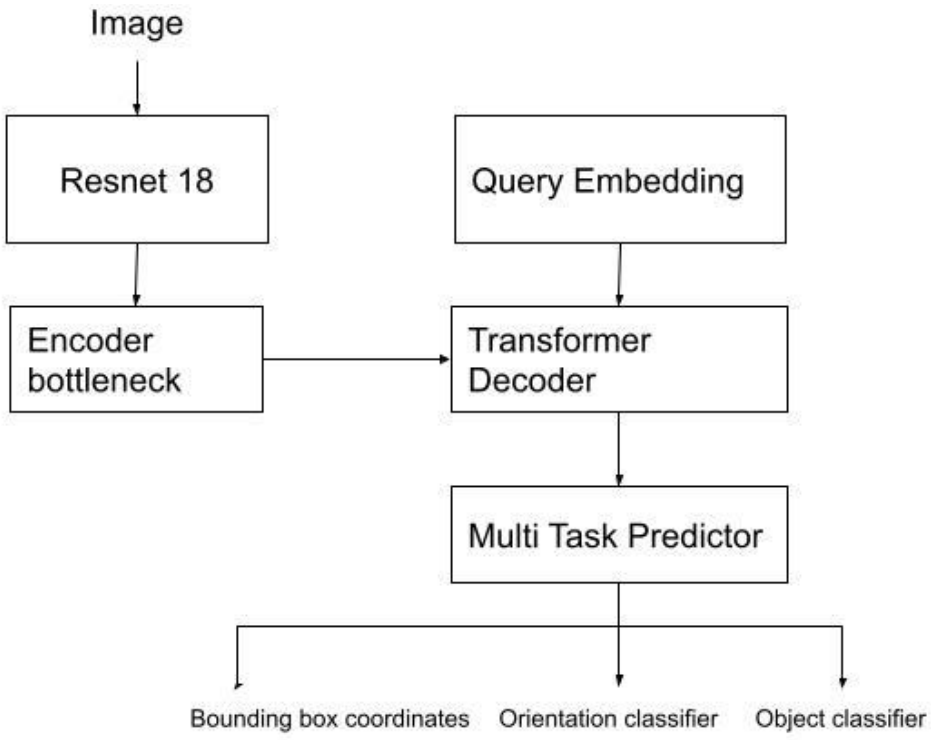

Figure 1: Architecture for the Object Grasping network

4.2 Encoder

Pre-trained Resnet-50[7] network was used as the encoder part of the Transformer network. In our architecture, we do not use self-attention mechanism for the encoder head which is potentially a future work. Input to the encoder are images of size 224x224x3. The dimensions of the output of Resnet is reduced to avoid potential overfitting. We experimented with different Resnet backbones which are explained in the Experiments section.

4.2.1 Resnet

We use the pre-trained Resnet-50 as our encoder. The final linear layer of the Resnet is removed and embedding from the penultimate linear layer was used as the embedding for the decoder. The embedding output from the encoder has a spatial component of 7*7 with 512 channels. We have combined spatial information as our embedding for the next layers. We experimented with training the model by preserving the spatial information but the performance of the network was sub-par. This network is not trained and gradients for this network is not updated.

4.2.2 Encoder Bottleneck

The channel output of Resnet is 2048. The channel dimension is reduced by passing the output of Resnet through a sequence of convolution layers and bring down the channel size to 512 and subsequently, 64. We then pass this embedding through a linear layer which converts the 49-dimensional vector into a 128-dimensional vector which is provided as an input to the decoder.

4.3 Decoder

Similar to the Encoder, the Decoder was split into two sub networks. THe first network is the transformer decoder layer and the second network utilizes the embedding of the transformer decoder to predict the bounding box and the orientation.

4.3.1 Decoder Transformer

Similar to the DETR model we have query embedding that are learned and used as the input for predicting the bounding box and orientation. We experimented with different sizes of embedding starting with 16 and inputs from the encoder. The output of which is then passed through a layer of transformer decoder layer. The output is then sent to the multi task predictor.

4.3.2 Multi task predictor

We used a small network with linear layers for bounding box prediction, orientation prediction and classify the image between "present" and "absent" object classes. Bounding box outputs points consists of center points, width and height which are then converted to the

$$x1, y1, x2, y2$$

format.. Output of the orientation is logits which then passed through a softmax layer. Resolution was set to about 10 degrees.

4.4 Loss functions

We took two different approaches for calculation loss. Both of the below mentioned loss calculations resemble in other in the sense of the loss functions used but differ greatly in the loss calculated.

4.4.1 Vanilla Loss

The output of the model is *n* queries as defined by the decoder transformer. Each of these 16 queries provide bounding box predictions and orientation logits. We used the Intersection Over Union(IoU) metric which is defined by the below formula to choose a query whose result closely matched with the ground truth data.

$$IoU = \frac{A \cap B}{A \cup B}$$

Calculation of loss in this Vanilla loss model is outlines below:

- Pick the query whose bounding box output matches closely with that of the ground truth data. This was achieved by using IoU as the metric
- Get the orientation and bounding box output values from the selected query
- Use cross entropy loss to calculate the orientation loss
- Use Smooth L1 loss to calculate the regressed bounding box loss

$$\mathbb{L}_{total} = \mathbb{L}_{orientation} + \mathbb{L}_{bbox} \tag{1}$$

$$\mathbb{L}_{orientation} = -\sum t_i log(p_i) \tag{2}$$

$$\mathbb{L}_{bbox} = \begin{cases} |x|, & \text{x} > \alpha \\ \frac{1}{|\alpha| x^2}, & \text{x} \leq \alpha \end{cases}$$

4.4.2 Bipartite Matching Loss

Similar to DETR [2] we uses bipartite matching loss, with the key difference of having a single object in the image. For cost function calculation, we included the L1 distance of bounding box and intersection over union. Since each of the images have only one object, we select the query with the least cost and use it calculate the loss. We include four losses. Bounding box and orientation loss is similar to the vanilla loss. We add Intersection over loss and binary cross entropy loss for predicting object/no object within the bounding box.

$$\mathbb{L}_{total} = \mathbb{L}_{orientation} + \mathbb{L}_{bbox} + \mathbb{L}_{bce} + \mathbb{L}_{iou} \tag{3}$$

$$\mathbb{L}_{bbox} = L1_{distance}(prediction, groundtruth) \tag{4}$$

$$\mathbb{L}_{orientation} = -\sum t_i log(p_i) \tag{5}$$

$$\mathbb{L}_{bce} = -[t_i log(p_i) + (1 - t_i) log(1 - p_i)] \tag{6}$$

$$\mathbb{L}_{iou} = 1 - [iou - (area - union)/area] \tag{7}$$

5 Experiments

As described in the previous section, we conducted experiments based on different loss criteria, varied number of decoder heads, varied encoder backbone, multiple batch sizes. Results of these experiments are summarized below.

We trained the model on an NVIDIA GeForce RTX 2080 with 11GB of GPU computational power. For all the batch sizes that we experimented with, we trained the model for 300 epochs, with initial learning rate α set to 1e-4. Learning rate scheduler was added to kick in when the validation loss was plateaued.

Since the Cornell Dataset only has 5k images, data augmentation was performed where we used Rotate, Translate and Horizontal Flip as augmentation criteria. We used 5568 training images and 128 validation images.

5.1 Vanilla Loss Model Results

The graphs for training and validation loss was generated using tensorboard and are as shown below. We noticed severe overfitting during the training. This was due to the fact that the size of the dataset was small. The overfitting problem was mitigated to an extent using data augmentation techniques but it was not enough to completely overcome the problem. The fact that the loss model was trivial also caused the network to overfit. It can also seen from 2 that the learning rate scheduler kicked in when the learning rate plateaued.

5.1.1 Running Inference

Once the model was successfully trained, we used the predicted logits and passed it through the softmax layer. We then picked the query whose probability was the highest. This is in contrast to the loss calculation where we used IoU of the bounding box as the criterion to choose the best query. We then obtained the bounding box result from the corresponding query. Some of the successful and failure cases are as shown below.

5.2 Bipartite Matching

In this experiment we included the bipartite loss explained in section above. We experimented with different weights for different losses. When Bounding box loss was less, we noticed the network was not able to learn to predict bounding box so bounding box had cost of 5, generalised intersection over union loss had cost of 2, orientation loss had cost of 1 and BCE loss had cost of 4. Due to class imbalance of BCE loss the weight of object has was high compared to the no object case.

5.3 Number of Query embedding

In this experiment we train network with different number of query embedding. Since our dataset is small and each image has only one object, the network with only 4 embedding was able to predict the bounding box. We experimented with 8 and 16 embeddings and but increasing the number of embedding did not improve the performance.

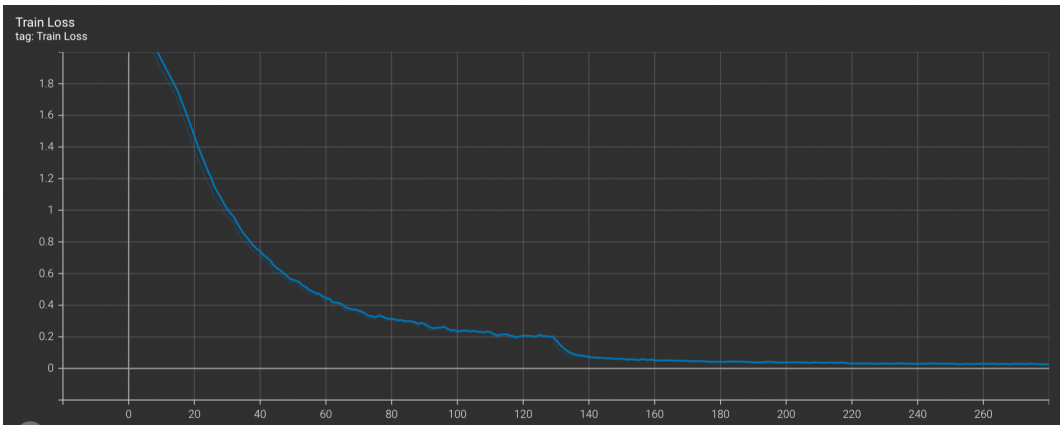

Figure 2: Training Loss for Vanilla loss model

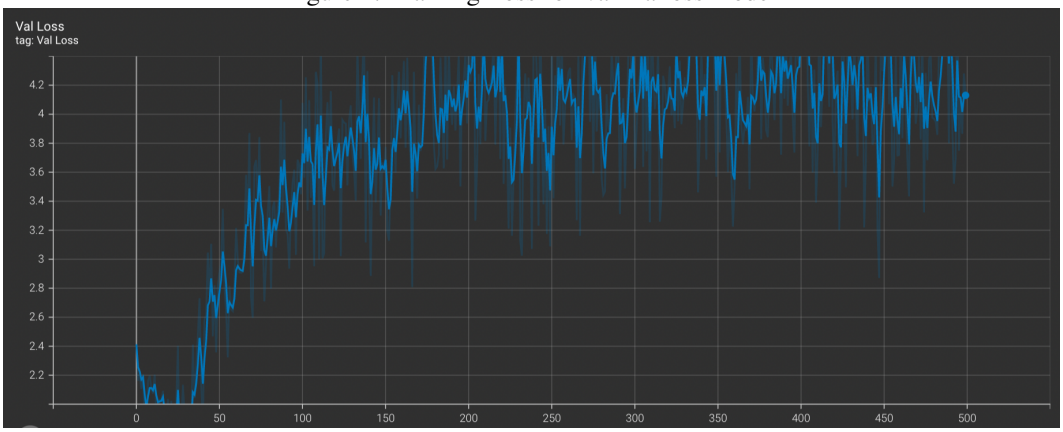

Figure 3: Validation Loss for vanilla loss model

6 Results

Below table summarizes the IoU achieved for different loss functions and the effect of batch size and dropouts on the ability of the model to perform well. We also show the training and validation accuracy for the network.

Model	**Batch Size**	**Dropout**	**mIoU**	**Train Acc**	**Val Acc**
Vanilla	32	Yes	0.49	56.2%	43.1%
Vanilla	32	No	0.47	**55.46%**	**34.7%**
Vanilla	64	Yes	0.5	58.1%	45.77%

It can be seen in the above table clearly that the network overfits on the training data when the vanilla loss model is used.

6.1 Bipartite Matching

The network successfully learnt to predict correct bounding boxes but failed to predict for few cases and figure 6.1 shows different prediction for object. Since grasping is a multi-modal problem, we observed that even though the grasp was right, comparing it with a single ground truth image resulted in reduced IoU. We also observed that missing depth information causes predictions to fail as well. Due to lack of depth information, the bounding box are over some edges shown in figure 6.1.2.

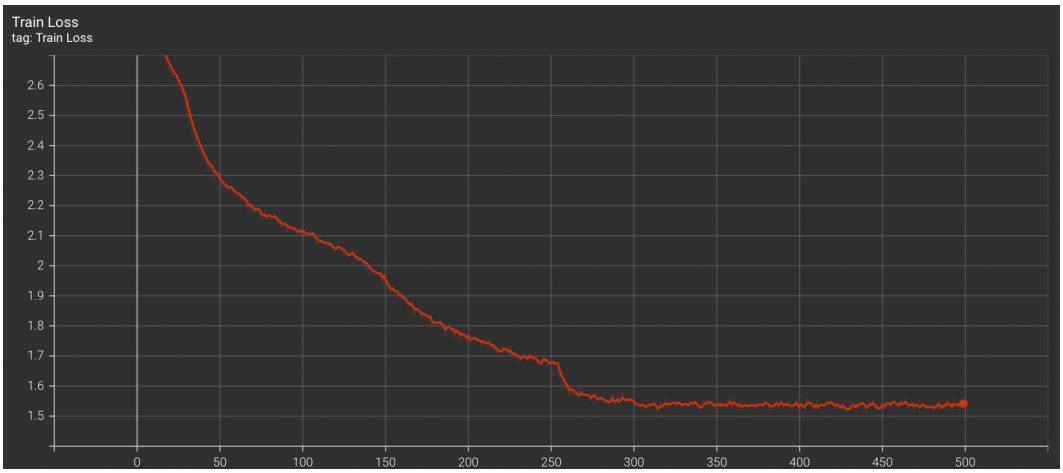

Figure 4: Training Loss for Bipartite matching loss model

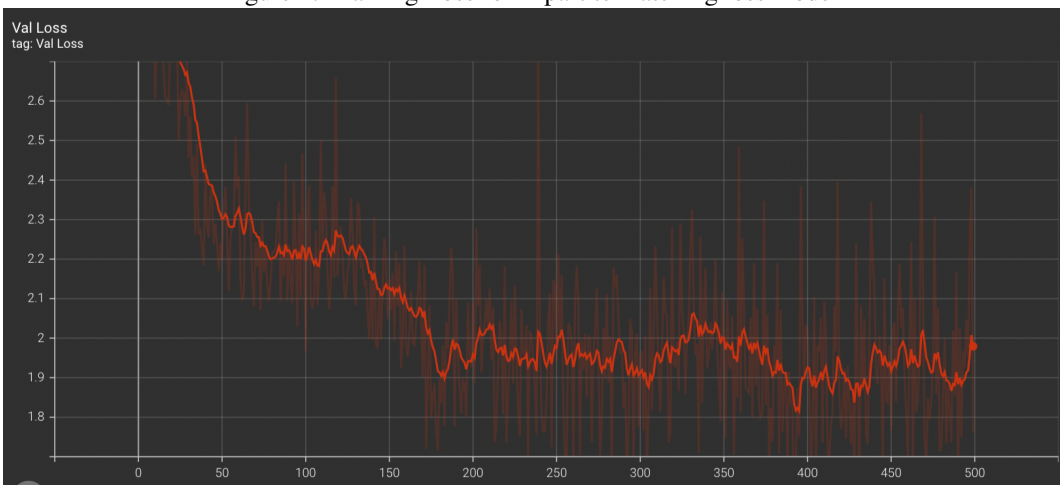

Figure 5: Validation Loss for Bipartite matching loss model

Figure 6: Successful Grasp for a Spatula

Figure 7: Successful grasp for a candy

Figure 8: Failed Grasp for Sandals

Figure 9: Failed grasp for a toy

Figure 10: Grasp prediction for regular shape object

Figure 11: Grasp prediction for irregular shape object

Figure 12: Grasp prediction for regular shape object

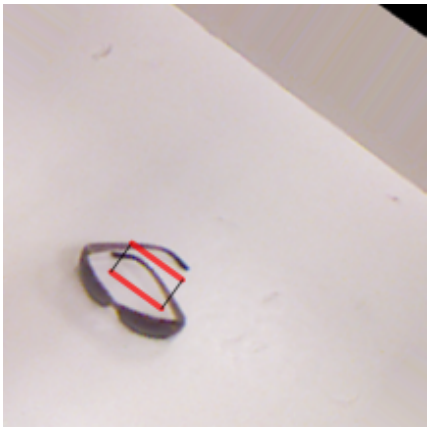

Figure 13: Grasp prediction for irregular shape object

6.1.1 Correct bounding box

For few cases the network prediction is different from the possible combinations. This would lead to lesser accuracy but the output can be used to pick up object. 6.1.1 shows few cases where the network output are different from possible combinations yet the bounding box prediction is correct.

Figure 14: Correct Bounding box for Spectacle according to dataset

Figure 15: Correct Bounding box for Spectacle according to network

6.1.2 Distortion due to projection

Currently, we are using only monocular images which makes it hard for the network to learn depth information from the image. 6.1.2 shows image where oracle has bounding box on top where as in prediction the bounding box extended to side face and remaining on top face this is mainly due to lack of depth information.

Figure 16: Bounding box on Top of the charger

Figure 17: Bounding box on the side of the charger

6.2 Number of Query embedding

As show in table 6.2 we see that there is no significant improvement with more query embeddings which may be due to small and simple dataset with single object per scene.

Embeddings	mIoU
4	0.51
8	0.513
16	0.512

7 Conclusion and Future Work

Robotic grasp detection task involves the network to predict region of grasp for the different object using parallel gripper. We have proposed a transformer based network to detect and predict the grasp

for the object of interest. We demonstrated that the network is able to learn to predict the grasp with good accuracy. As an extension, more experiments can be run on different datasets as mentioned in the Dataset section and different loss functions could be explored.

8 Acknowledgements

We would like to thank Prof. Wang and the instructional team for this wonderful course and all the help extended along the way. We also would like to thanks our peers for their contributions on Piazza.

References

[1] Ainetter, S. and Fraundorfer, F., 2021, May. End-to-end trainable deep neural network for robotic grasp detection and semantic segmentation from rgb. In 2021 IEEE International Conference on Robotics and Automation (ICRA) (pp. 13452-13458). IEEE.

[2] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A. and Zagoruyko, S., 2020, August. End-to-end object detection with transformers. In European conference on computer vision (pp. 213-229). Springer, Cham.

[3] H. Zhang, X. Lan, S. Bai, L. Wan, C. Yang, and N. Zheng, "A multitask convolutional neural network for autonomous robotic grasping in object stacking scenes," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 6435–6442.

[4] D. Park, Y. Seo, D. Shin, J. Choi, and S. Y. Chun, "A single multitask deep neural network with post-processing for object detection with reasoning and robotic grasp detection," in IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 7300–7306.

[5]I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," The International Journal of Robotics Research, vol. 34, no. 4-5, pp. 705–724, 2015

[6]M. Suchi, T. Patten, D. Fischinger, and M. Vincze, "Easylabel: A semi-automatic pixel-wise object annotation tool for creating robotic rgb-d datasets," in IEEE International Conference on Robotics and Automation (ICRA), 2019, pp. 6678–6684.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in European Conference on Computer Vision (ECCV), 2016, pp. 630–645.

[8] R. M. Murray, Z. Li, and S. S. Sastry, A mathematical introduction to robotic manipulation. Boca Raton, FL, USA: CRC, 1994, 2017.

[9] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review," in Proc. IEEE Int. Conf. Robot. Autom., San Francisco, CA, USA, Apr.2000, pp. 348–353.

[10] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, ´ "Feature pyramid networks for object detection," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2117– 2125.

[11] Ren, S., He, K., Girshick, R. and Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28.

[12] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 1316–1322.

[13] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," The International Journal of Robotics Research, vol. 34, no. 4-5, pp. 705–724, 2015.

[14]] S. Kumra, S. Joshi, and F. Sahin, "Antipodal robotic grasping using generative residual convolutional neural network," in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020.

[15] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 3406– 3413. [16] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," The International Journal of Robotics Research, vol. 37, no. 4-5, pp. 421–436, 2018.

[17] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. Advances in neural information processing systems, 30.

[18] X. Zhou, X. Lan, H. Zhang, Z. Tian, Y. Zhang, and N. Zheng, "Fully convolutional grasp detection network with oriented anchor box," in IEEE/RSJ International Conference

9 Individual Contribution

- Cornell Dataloader: Srinidhi
- Vanilla Loss Model: Srinidhi
- Loss with generalized IoU: Sumukh
- Training and Validation loop: Sumukh and Srinidhi (Equal Contribution)
- Models: Sumukh and Srinidhi (Equal Contribution)
- Experiments: Sumukh and Srinidhi (Equal Contribution)

Pretrained Models used: Resnet

Reused code PyTorch data augmentation code:

https://github.com/Paperspace/DataAugmentationForObjectDetection/blob/master/data_aug/data_aug.py

The code above was modified for our needs.

Generalized IoU from https://giou.stanford.edu/