Rheinische
Friedrich-Wilhelms-
Universität Bonn

Institute for Computer Science
Department VI
Autonomous Intelligent Systems

Rheinische
Friedrich-Wilhelms-Universität Bonn

Master Thesis

Fast Training of Convolutional Neural Networks For Scene Understanding

Author:
Arul Selvam Periyasamy

First Examiner:
Prof. Dr. Sven Behnke

Second Examiner:
Prof. Dr. Angela Yao

Date:

Declaration

I hereby declare that I am the sole author of this thesis and that none other than the specified sources and aids have been used. Passages and figures quoted from other works have been marked with appropriate mention of the source.

Place, Date

Signature

Abstract

The ability of the robots to autonomously manipulate objects in cluttered real-world environments depends on the ability to perceive and understand the scene. In this thesis, we propose a pipeline for fast training of Convolutional Neural Networks (CNNs) for scene understanding. The pipeline consists of a quick and efficient turntable capture of new objects with automatic segmentation using background subtraction, alignment of multiple capture sequences, scene generation for pose estimation using turntable data and previously captured backgrounds, a semantic segmentation module and a pose estimation module. The semantic segmentation module is based on RefineNet architecture, and the pose estimation model we propose is a CNN that takes fixed sized crops of the segmented objects and estimates the 5D pose (x and y of the translation component, and a unit quaternion for the rotation component). We propose a method for dealing with invariance in the object's appearance in order to facilitate the training of pose estimation network. We also train the network with synthetically occluded objects to deal with real-world occlusions.

Some of the components of the pipeline were used by Team NimbRo at the Amazon Robotic Challenge (ARC), where the robots have to perform picking and stowing of objects from a cluttered bin. One of the important aspects of the Team NimbRo's approach for ARC is the ability to fast train on new objects in a short span of 45 minutes. We evaluate the complete pipeline on the dataset collected for ARC, as well as on a disaster response scenario dataset, and assess different design choices for the pose estimation module. Finally, we show a real-world setting where the pipeline is used in successful grasping of a novel object.

Acknowledgements

I thank Max Schwarz, Autonomous Intelligent Systems (AIS) group, for his help, support, and collaboration for the entirety of my time at AIS as a student assistant, and during this thesis work. I extend my gratitude to Prof. Dr. Sven Behnke, head of AIS, for allowing me work as a student assistant at AIS and do this thesis work at AIS. I would also like to thank Stefan Oßwald, Humanoid Robotics Lab, for his support during my Master's studies.

Contents

1 Introduction

Scene understanding is a vast visual problem, and the fact that the ability to understand a scene from visual inputs comes very natural to humans makes it harder to formulate the problem of scene understanding precisely. Thus the formulation of scene understanding is quite general, and a more specific formulation is often application dependent. In simple words, the problem of scene understanding can be defined as extracting as much semantic information as possible from the visual input of scene. The kind of semantic information that needs to be extracted depends on the application. For example, the information needed for an autonomous driving system differs from that of an indoor bin picking system. Despite these differences, some information is ubiquitous. This information includes what objects are in the scene, and where the objects are in the scene. In this thesis, we focus on scene understanding for object manipulation. To enable object manipulation, in addition to the above-said information, full 6D pose including orientation is required.

The pipeline we use in this thesis consists of a semantic segmentation module and a pose estimation module. The input scene is represented as RGB-D images. The semantic segmentation module estimates the probability of each pixel in the image belonging to one of the object classes and the pose estimation module estimates the position and orientation of the objects in the scene. We use Convolutional Neural Networks (CNNs) to perform both semantic segmentation and pose estimation. While Convolutional Neural Networks are the state-of-the-art in many visual perception tasks, the data hungry nature and time needed for training the CNNs pose challenges in deploying CNNs in many real-time robotic applications. In this thesis, we propose faster data acquisition and training mechanisms to enable efficient usage of CNNs in scene understanding for robotics object manipulation tasks and provide the results of evaluating the proposed methods at Amazon Robotics Challenge and in a disaster response scenario.

In chapter 2, we discuss the works related to the individual modules in the pipeline and the approaches by the teams that performed well in Amazon Picking Challenge 2016. In chapter 3, we elaborate the data acquisition pipeline along with the calibration process of the data acquisition setup. In chapter 4 and 5 we discuss in details the architecture of the semantic segmentation module and the pose

estimation module respectively; and the metrics used to evaluate them. In chapter 6, we provide an overview the implementation details reviewing the libraries used. The chapter 7 is divided into three sections: evaluating different aspects of the pose estimation module, application in cluttered bin-picking scenario, and in a disaster response scenario[1].

Finally, in chapter 8 we discuss the possible future works and conclude the contributions of the thesis.

The major contributions of this thesis are:

1. A high throughput data acquisition pipeline.
2. A pose estimation module.
3. Using these above modules along with the semantic segmentation module proposed by Schwarz, Lenz, et al. (2018) to implement a scene understanding pipeline.
4. Evaluating the pipeline in two different scenarios: random bin-picking and disaster response.

[1] `https://www.centauro-project.eu/`

2 Related Work

We first describe the works related to each of the individual components in the scene understanding pipeline, then we describe the works related to the pipeline as a whole, emphasizing on Amazon Picking Challenge 2016 (APC)[1] and Amazon Robotics Challenge 2017 (ARC)[2].

2.1 Semantic Segmentation

Deep learning methods are the most common approach for many computer vision tasks like image classification, object detection, semantic segmentation, etc. Before we review the literature of deep learning methods for semantic segmentation, we briefly discuss the methods used for semantic segmentation before the advent of deep learning methods.

Graph cut methods that formulate the problem of segmentation as energy minimization on a graph representation of an image with pixels as nodes and affinity measure between the pixels as edges are one of the early methods used for semantic segmentation. The solution is approximated with maximum flow in the graph (Shi and Malik (2000), Rother, Kolmogorov, and Blake (2004), Y. Boykov and Funka-Lea (2006), Y. Y. Boykov and Jolly (2001)). Other most prevalent methods are the probabilistic graphical models such as Conditional Random Fields (CRF). CRFs works by assigning a class for each pixel (from a classifier), a unary cost for changing the class assignment, and a binary pairwise potential if two neighboring pixels are not of same class. The inference algorithm tries to find the setting of labels that minimizes total cost (Russell, Kohli, and Torr (2009) Gould, Fulton, and Koller (2009), Kumar and Koller (2010), Müller and Behnke (2014), Lempitsky, Vedaldi, and Zisserman (2011), Müller and Behnke (2013)).

Deep learning methods are the state-of-the-art in the task of semantic segmentation (Schulz and Behnke (2012), Girshick et al. (2014) A Milan et al. (2017), Couprie et al. (2013), He et al. (2017)). Unlike the image classification task where the spatial information lost due to the pooling layers has no impact on the per-

[1] `https://www.amazonrobotics.com/#/roboticschallenge`
[2] `https://www.amazonrobotics.com/#/roboticschallenge/past-challenges`

formance of the CNNs, the semantic segmentation performance suffers due to loss of spatial information. Many recent works address this problem. Most notably, Fully Convolutional Networks (FCNs) use skip connections to propagate information from early layers to the final layers of the network (Long, Shelhamer, and Darrell (2015)), L.-C. Chen et al. (2016) use atrous convolutions to enlarge the receptive field if the convolutions, and Pohlen et al. (2016) use full-resolution residual network with two stream information flow to make the low level features available to the all the higher layers.

2.2 Pose Estimation

The most common approach for 6D object pose estimation is to perform object detection or semantic segmentation to extract the points belonging to object, use the centroid of the points as the origin and initialize the pose using the result from singular value decomposition of the points (Besl and McKay (1992), Haehnel, Thrun, and Burgard (2003), Granger and Pennec 2002). Machine learning approaches like deformable parts-based model for object pose estimation from a single RGB image (Zhu et al. (2014)), voting-based approach employing color point pair feature Choi and Christensen (2016) for pose estimation from RGB-D image, regression forest for pose estimation from RGB image(Hara and Chellappa (2014)), Hough forests based pose estimation from synthetic RGB-D images(Badami, Stückler, and Behnke (2013)) (Kouskouridas et al. (2016)), CNN based feature extraction for estimating the yaw angle of objects placed in a table (Schwarz, Schulz, and Behnke (2015)) have been proposed. Kendall, Grimes, and Cipolla (2015) used CNNs based regression to predict 6D pose of camera for visual localization from RGB images collected from a hand-held mobile phone camera in a large urban scene. Koo et al. (2017) used CNN based regression the pose of individual chain links from a pile using RGB-D images.

A different approach compared to the above mentioned methods for 6D pose estimation based on analysis-by-synthesis framework by (Yuille and Kersten (2006)) using deep learning methods was proposed in Krull et al. (2015). The pipeline proposed by the authors is shown in Fig. 2.1: It uses random forests to generate an object probability distribution as in semantic segmentation and 3D object coordinates corresponding to each object for each pixel in the image. Then for each detected object, using a RANSAC-like approach by randomly selecting 3-tuples of pixels and computing affine transformation with their predicted 3D object coordinates, the authors generate a set of hypothesis H using metropolis algorithm and render RGB and depth images from the 3D model for each of the hypothesis. Then

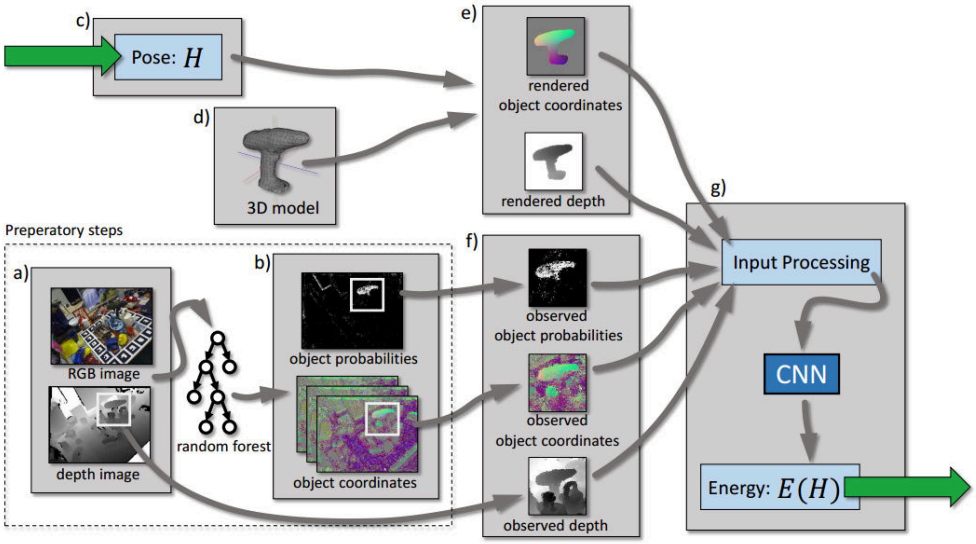

Figure 2.1: Learning analysis-by-synthesis. Source: Krull et al. (2015).

the authors train a CNN to take the rendered and observed images as input and output an energy value such that the energy value is high if inputs are similar and low when dissimilar. Finally, the hypothesis with maximum energy is selected. In contrast, our work needs one froward pass per segmented object.

2.3 APC 2016 Perception Solutions

APC 2016 included two tasks; picking and stowing. In the picking task, the teams needed to pick the specified items from the bins of the shelf and place them in a tote. Team NimbRo's robot is shown performing the picking task in Fig. 2.2. The stowing task was about stowing the objects from the tote into the shelf. Both the task involved picking/stowing of 12 objects in 15 minutes. A set of 40 objects were provided to the participating teams a couple of months prior to the challenge in order to train the perception systems.

The standard approach by most of the teams in Amazon Picking challenge, notably Team Delft (Hernandez et al. (2016)), NimbRo Picking from Universität Bonn (Schwarz, Milan, et al. (2017)), and Team MIT and Princeton (Zeng et al. (2016)), was a pipeline consisting of object detection and/or semantic segmentation, extracting relevant point cloud from the scene, and using ICP to align the observed point cloud with the 3D model of the object

Figure 2.2: Team NimbRo's robot picking objects from the shelf and placing them in the red tote. Source: Schwarz, Milan, et al. (2017).

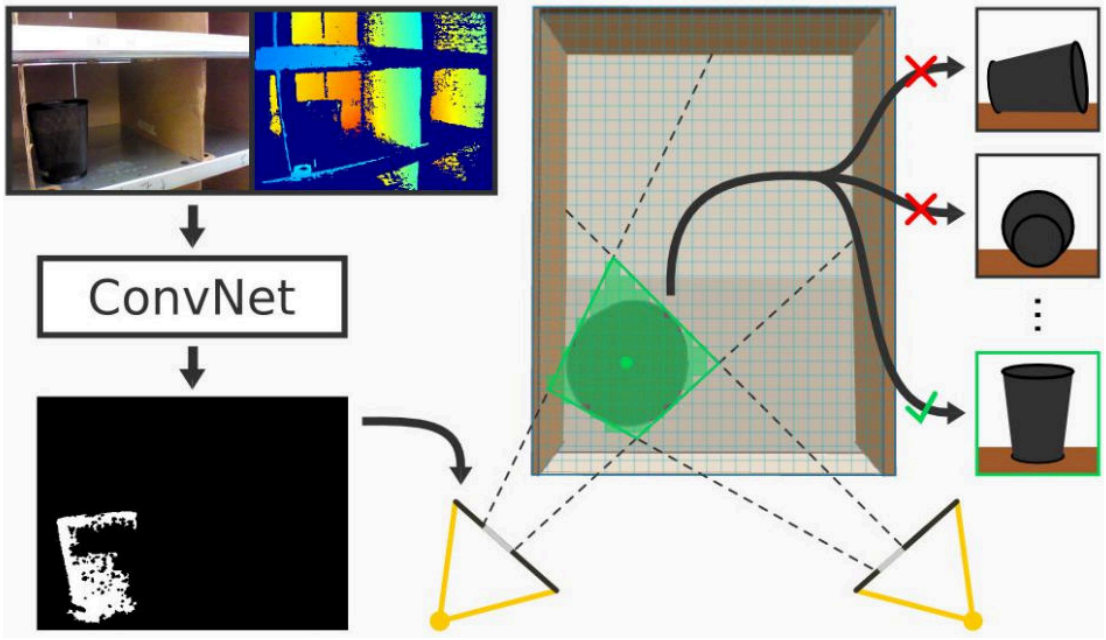

Figure 2.3: Perception approach by team MIT and Princeton. Source: Zeng et al. (2016).

2.3.1 Team Delft and Team MIT/Princeton

Team Delft localized the objects from as RGB image using Faster R-CNN (Ren et al. (2015)) and used the Super 4PCS algorithm (Mellado, Aiger, and Mitra (2014)) for aligning the observed point cloud with a 3D model. Team MIT and Princeton did instance segmentation from RGB using Fully Convolutional Networks (FCN) instead of object detection (Long, Shelhamer, and Darrell (2015)). An interesting point about their system is the use of multiple views of the scene and fusing the segmentation results from each view to estimate the orientation of the object as shown in Fig. 2.3.

2.3.2 Team NimbRo Picking

The perception system used by Team NimbRo consists of an object detection module using CNN based on DenseCap (Johnson, Karpathy, and Fei-Fei (2016)), a semantic segmentation module using CNN proposed by Husain et al. (2017), registering the segmented point clouds with the 3D model to determine the 6D pose(only for 3 objects that needed specific grasping locations), and dense 3D modeling using Prankl et al. (2015).

We will discuss the object detection and semantic segmentation models in detail.

The object detection model is based on DenseCap network used for dense captioning of the RGB images. It consists of a CNN that takes RGB images and gen-

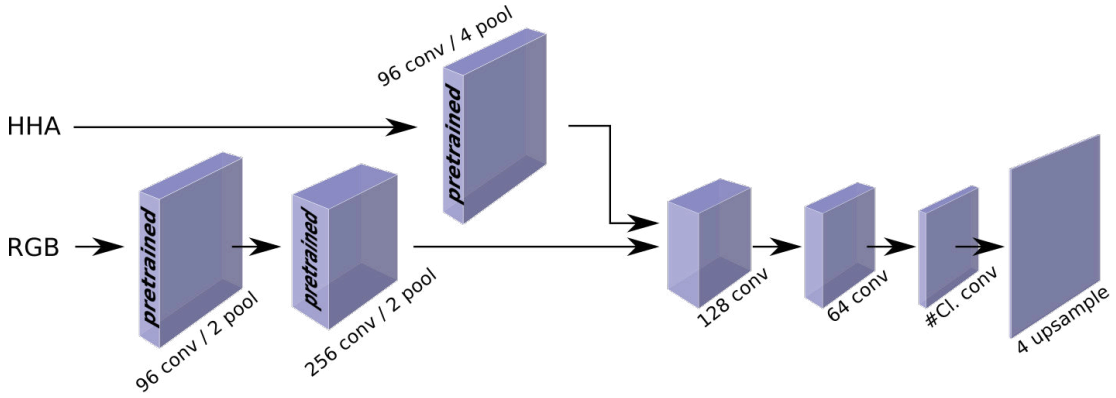

Figure 2.4: Semantic segmentation model used by team NimbRo. Source: Schwarz, Milan, et al. (2017).

erates regions proposals (bounding boxes and corresponding 14×14 dimensional 1024 features maps). These regions proposals are fed to a language model that generates captioning for each proposal. The complete model is trained end-to-end. For APC 2016, the language model was replaced by an SVM trained on-line to classify the objects present in the predicted bounding boxes. Semantic segmentation was done using the CNN shown in Fig. 2.4. The initial two layers were copied from OverFeat Network Sermanet et al. (2013). The last three layers were tuned for APC dataset. The resulting images from the CNN were up-sampled to the size of the input images. Both networks were modified to accept RGB and depth encoded as HHA features Gupta et al. (2014) as input.

2.4 ARC 2017 Challenge Description

ARC 2017 was designed to increase the complexity compared to APC 2016. Among the several changes included, we will discuss the changes that impact the perception system and these changes motivates the contributions of this thesis. To start with, like APC 2016, the teams were provided with a set of 40 objects a couple of months prior to the competition which we call training set. Just 45 minutes before each challenge, the teams received a set of 32 objects named competition set. Half the objects in the competition set are new. These new objects are similar to objects in the training set but not the exact same ones. Thus the time available for training the CNN models to include the new objects is limited. Also the time available for collecting the training data is minimal. This makes any manual data annotation process impractical.

3 Data Acquisition

3.1 Data Acquisition Pipeline

Training convolutional neural network usually needs lots of training data. Using a pre-trained network trained on larger dataset and fine-tuning the network for the task-in-hand alleviates the need for huge training dataset but even for performing fine-tuning we need significant amount of training data. Acquiring labeled datasets is a labor-intense operation. For many applications, including the Amazon robotics challenge, the time available for collecting the training is limited. Thus we need an automated data acquisition setup. Our design goals for a data acquisition setup were straightforward:

- The time required for acquiring images of a new object should be minimal.
- Completely automated setup is not quite possible because of the need to support a large variety of object shapes/textures but human labor needed in the process should be scalable.

The data acquisition setup is shown in Figure 3.1. It consists of a turntable (B in Figure 3.1), an Intel RealSense SR300 camera (B in Figure 3.1) and a Nikon D3400 camera (D in Figure 3.1), background panel with monotonic color (A in Figure 3.1), and a pair of LED panels for additional light (E in Figure 3.1). Intel RealSense camera is an RGB-D camera suitable for depth perception up to 1.5m while Nikon D3400 is a consumer scale RGB camera capable of significantly higher resolution RGB images (24 MP).

The data acquisition pipeline consists of the following steps:

1. Data capture,
2. Background subtraction,
3. View alignment, and
4. Synthetic data generation.

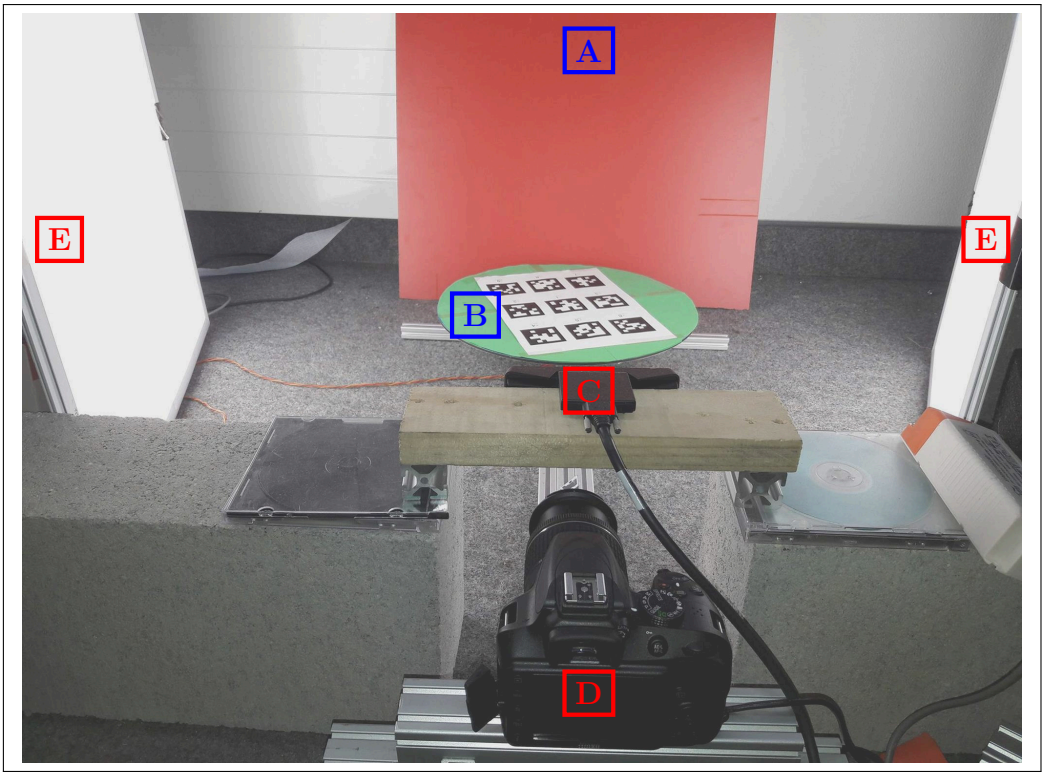

Figure 3.1: Data capture setup. A: Background panel; B: Turntable; C: SR300 camera; D: Nikon camera; E: LED lights.

3.1.1 Data Capture

The RealSense and Nikon cameras operate at different frame rates and we do not employ any hardware synchronization. Images from RealSense and Nikon are matched based on their timestamp. The frames that are more than the specified time difference apart are discarded. For every RGB image from Nikon camera, the closest SR300 image is found. This matching process is done greedily. The data acquisition setup is capable of capturing 2 frames per second, mostly limited by the Nikon camera capture and download rate.

3.1.2 Background Subtraction

The frames acquired from the data acquisition setup need to be processed to extract the pixels belonging to the object, and the pixels belonging to the turntable and other background objects need to be discarded. This is done a using background subtraction mechanism. During the beginning of the data acquisition process, a background frame without any objects is captured. Then a frame with the object is compared with the background frame. The pixels that differ by a predefined threshold or more are considered as belonging to the object. This step is quite simple but due to wide range of objects with different reflectance properties, finding a correct threshold for background subtraction that works for all objects needs complex mechanisms. Also, accuracy of the background subtraction step impacts all the modules in the rest of the pipeline. Thus, to achieve high-quality background subtraction without complex mechanisms, we opted for a manual background subtraction step but with minimal human effort. We designed a GUI to aid human operator in try out different thresholds, see the effect of the threshold, and then choose a threshold for the background subtraction. The human operator can also specify if the object has holes, or is convex. This information is used as a constraint producing either a filled or convex polygon as the final object region, thus making the segmentation more robust when allowed by the object shape. An example of the background subtraction process with the GUI is shown in Figure 3.2.

On a large scale deployment scenario, one can imagine this module being learned using machine learning techniques as well. But for ARC 2017, we used the above explained manual background subtraction step.

3.1.3 View Alignment

When objects are placed on the turntable, not all the faces are visible. Thus, we need to capture multiple sequences of objects with different faces pointing

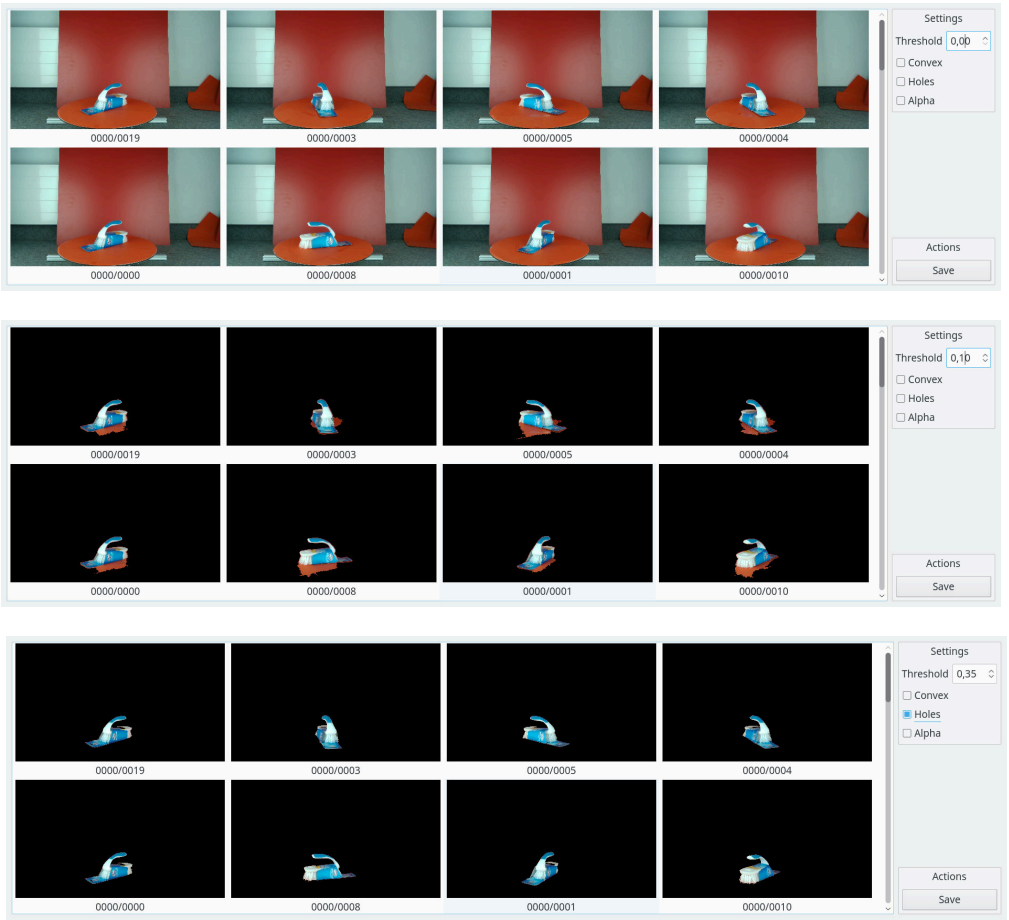

Figure 3.2: Background subtraction GUI. Top: Frames captured without any threshold for background subtraction. Middle: Effect of background subtraction with a threshold of 0.1 Bottom: Effect of human operator chosen final parameters(threshold= 0.1 and holes enabled).

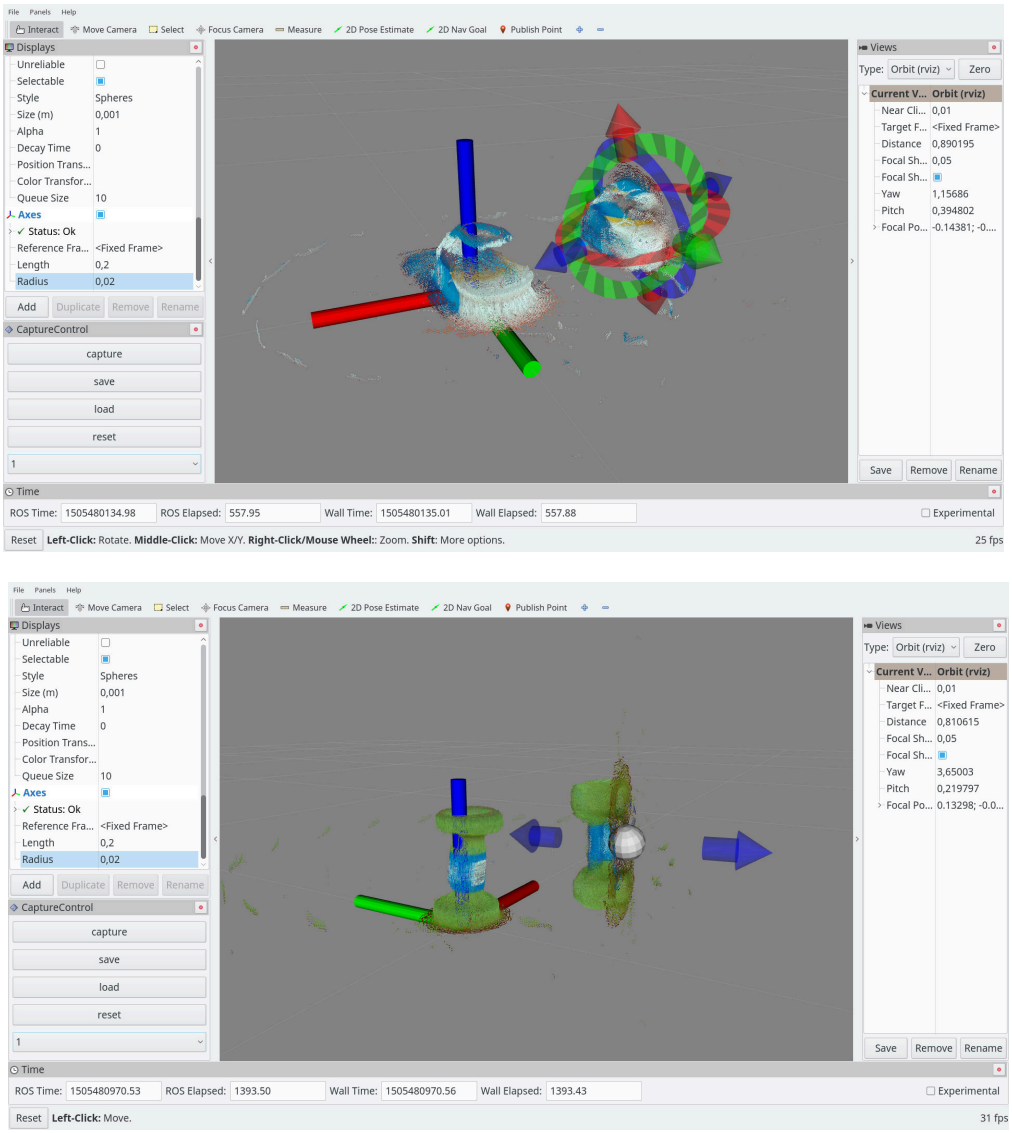

Figure 3.3: Alignment tool in usage. The aggregated point cloud with coordinate system is the frame of reference of the object. The transformation from the other point cloud to the frame of reference is specified using the 6D marker.

upwards. For generating data for training the segmentation network, no additional information is required. But for generating ground truth for training the pose estimation network, we need to know the transformation between the different sequences.

We created a tool to help with specifying the transformation between different sequences of an object. This tool was created as an RViz plug-in[1]. The transformations between the frames of a single sequence can be simply obtained from forward kinematics of the turntable. The initial frame of a sequence is considered as the reference frame for that sequence. All other frames are transformed to the reference frame and are aggregated. This aggregated point clouds for each placing is displayed in the RViz with 6D markers to manipulate them. A human operator can specify the transformation between different sequence by manipulating the 6D markers. The first frame of the first sequence is considered as the frame of reference of object and all other sequences are transformed into this frame. Some examples of the alignment step using the RViz plug-in is shown in Figure 3.3

3.1.4 Synthetic Data Generation

The semantic segmentation pipeline used in this thesis was augmented by our team with a scene generator operating on previously annotated background scenes. From the images acquired from the turntable setup, we generate labeled data for training semantic segmentation and pose estimation networks. Generating training data for semantic segmentation network starts with manually annotating a number of real scenes. Then we place the images of the objects obtained from the data capture setup on top of the scene to create synthetic training data. Examples of synthetically generated scenes along with the manually annotated scenes are shown in Figure 3.4

For pose estimation, we decided to follow this approach and implement a scene generator in the following way. We create synthetic training data following the steps shown in Figure 3.5. We extract the images of the objects from turntable setup by the background subtraction step. We sample a rotation angle along the optical axis of the camera frame and apply the transformation to the image of the object (step d in Figure 3.5) and extract a crop whose dimensions are computed based on the expected size of the object in the image. The motivation for this way of extracting the crop is to ensure a constant sized crop even when the object is partially occluded. We then place the extracted crop on top of randomly cropped image patches of the real scenes and shift the pixels that does not the object

[1] http://wiki.ros.org/rviz

Figure 3.4: Generated synthetic scenes. All scenes were generated with the same annotated background frame (left column) for easier comparison. Top row: RGB. Bottom row: Color-coded generated segmentation ground truth. Source: Schwarz, Lenz, et al. (2018)

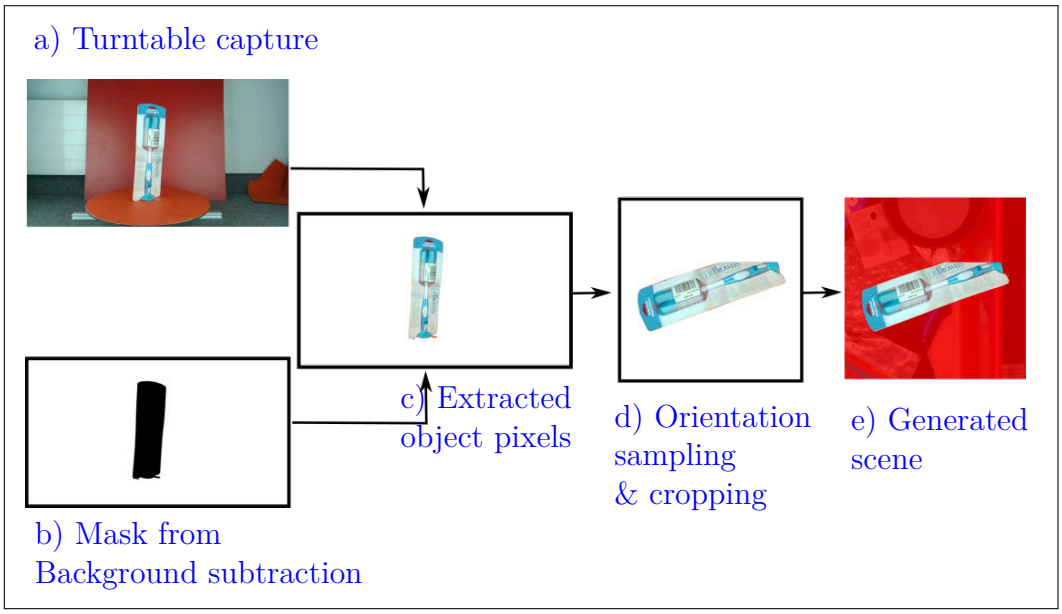

Figure 3.5: Steps in synthetic image generation.

towards red (step e in Figure 3.5) to emphasize the pixels belonging the object in focus.

The coordinate frames used in the data capture setup are shown in Figure 3.6. The *Camera Frame* is the frame of reference of the object. The *Base Frame* is the center of the turntable but fixed; the *Turntable Frame* is the center of the turntable but rotates along with the turntable. The *Object Frame* is the origin of the object. During the data capture, for an object, between the frames of same sequence the transformation $T^{Base}_{Turntable}$ changes according the angle of turntable rotation; $T^{Turntable}_{Object}$ changes between sequences.

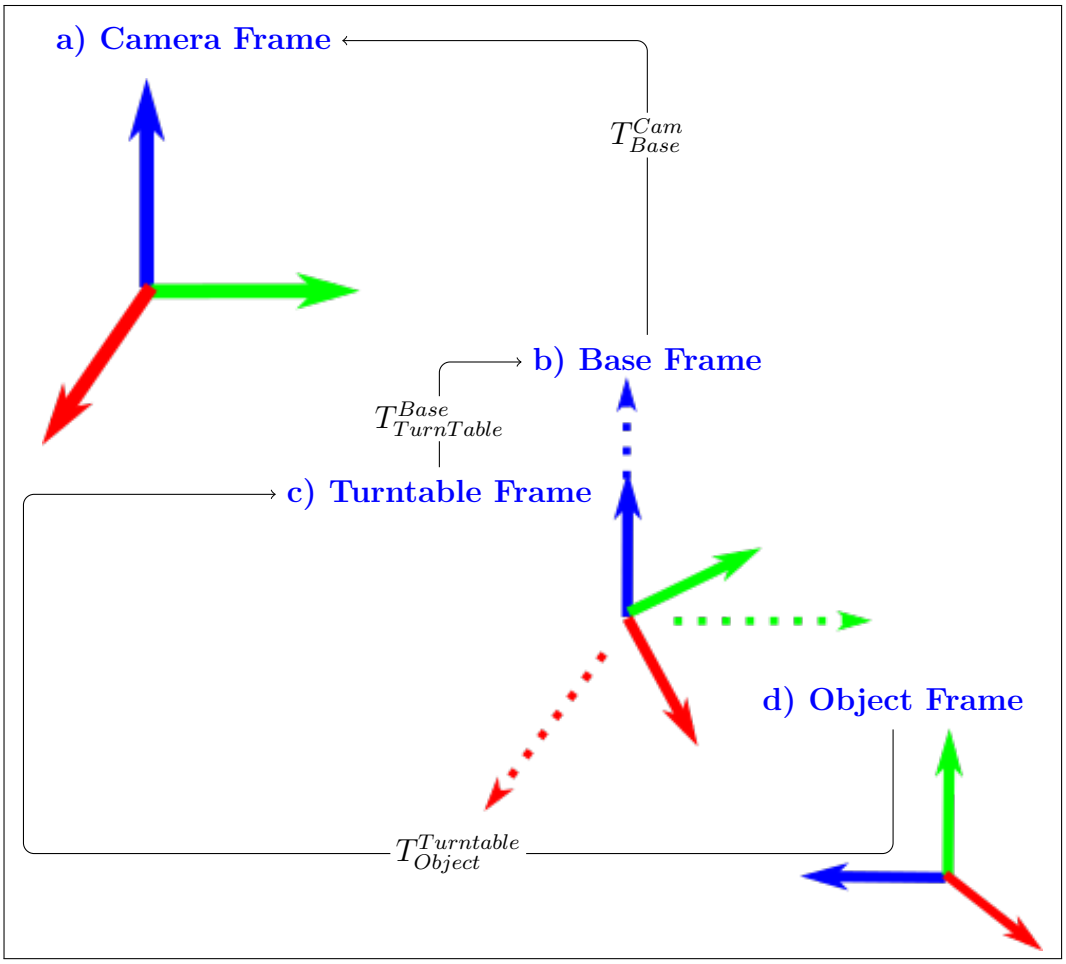

Figure 3.6: Coordinate frames used in the data capture setup.

3.2 Calibration

Obtaining ground truth for the pose estimation needs good calibration between the camera and the turntable. We use AprilTags, a fiducial marker based technique proposed by Olson (2011). A set of nine ArpilTags are placed on the turntable and frames are captured for one complete rotation of the turntable. To make use of conventional SLAM pipelines, we assume the turntable was fixed and the camera was moved around the turntable. For each frame, the AprilTags are detected. We use graph optimization approach to compute 6D pose of camera using g2o graph optimization package described by Kümmerle et al. (2011) for estimating the pose of the camera at different time frames. The graph optimization results in a set of camera pose that lie in a circle and the center of the turntable is the center of the circle on which the camera poses lie. To compute the center from a given set points on a circle, Hough transformation can be used. While Hough transformation is a simple process, exploiting the fact that the points roughly span the whole circumference and the points lie at roughly equal intervals, we can estimate the center of the circle in a more efficient way. The idea is motivated by the property of the circle that the distance from any point on the circle to the center of the circle is constant (radius). We compute the centroid $\hat{c}$ of the points. We search a small rectangular window space around the centroid by discretizing the search space into finer cells. For each cell, we assume the center of the circle to be the cell and compute the distance from each point, p_i, to the cell. We call each of the distance as r_i and we compute the mean of all r_i, $\bar{r}$. A cost for each cell is defined as

$$cost_{x,y} = \sum_i abs(\bar{r} - r_i)$$

The cell that has the minimal cost is the center we need. The points, centroid, and the computed center are shown in 3.7.

Note that this problem of finding the center of the circle given some points on the circle can also be formulated as a least-square problem and can be solved very efficiently using existing implementations. We used the above-explained method since the number of points is small and calibration is an one time process thus does not need to scale for more number of points.

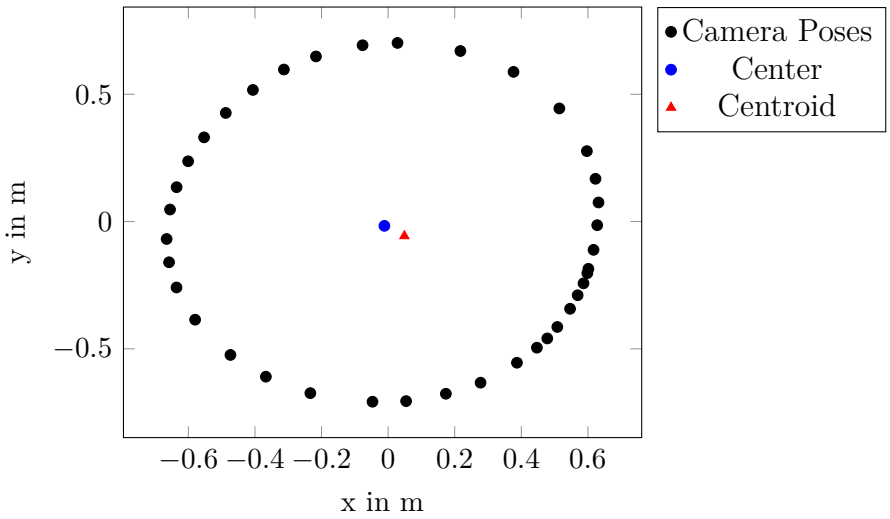

Figure 3.7: Estimated camera poses, centroid of the poses, and the estimated center of the turntable.

4 Semantic Segmentation

Semantic segmentation is the task of understanding what is in an image at the level of pixels. Semantic segmentation is often formulated as pixelwise classification but the task is much complex that an a simple image classification task. The research community also uses terms like scene labeling (Farabet et al. (2013)), scene parsing (Grangier, Bottou, and Collobert (2009)), etc., to emphasize the importance of understanding the scene to perform semantic segmentation.

4.1 Network Architecture

In the recent years, the algorithms that perform well on the most commonly used benchmarking datasets such as PASCAL VOC 2012 (Everingham et al. (2010)), Cityscapes (Cordts et al. (2016)), SUN RGB-D (Song, Lichtenberg, and Xiao (2015)), etc., uses deep learning techniques. Training CNNs from scratch is a time and resource consuming process, thus the common approach to train a CNN on a new task is to perform transfer learning. A detailed study by Yosinski et al. (2014) on the effectiveness of transfer learning showed that initialing weights by transfer learning and performing fine tuning leads to better generalization.

In our previous work for APC 2016, we used this approach to perform image segmentation for cluttered bin picking. While this approach provides good results, often the segmentation is poor along the object boundaries. This is mainly due to fact that after repeated down-sampling, the features in the final layers lack fine-grained spatial information need for performing semantic segmentation. One of the state-of-the-art methods, RefineNet proposed by Lin et al. (2017) tackles this problem by using the features also from the earlier layers. Figure 4.1 shows the semantic representation of the multi-path RefineNet where the features from different stages of ResNet are combined using learned convolutions.

A detailed representation of the RefineNet architecture is shown in Figure 4.2. The pipeline of the RefineNet is as follows: features from different stages of the ResNet are fed to an Adaptive Convolution layer with RCU blocks and the resulting features of different resolution are fused into features of same dimensions as that of the highest resolution using convolutions and upsampling. Then a Chained

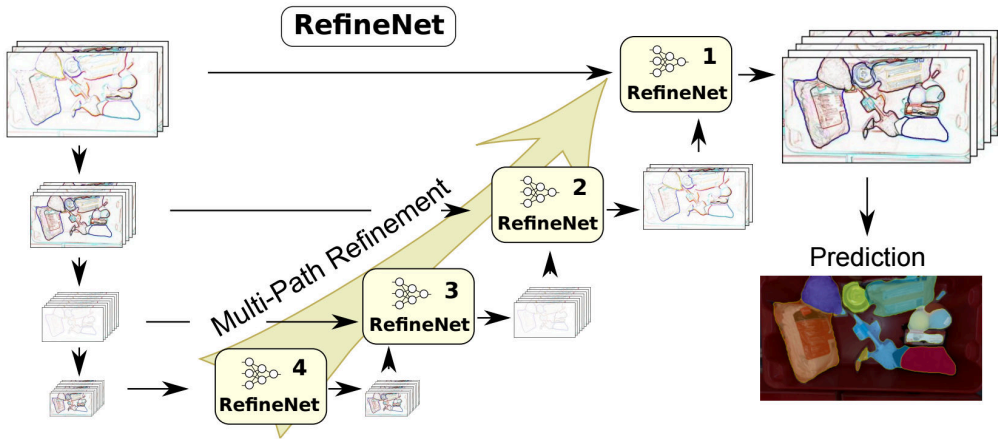

Figure 4.1: RefineNet architecture: The features from the different stages of ResNet are combined using learned RefineNet blocks into refined high-resolution feature maps. Source: A. Milan (2017).

Residual Pooling layer with a sequence of convolutions with different kernel size are employed and finally the features are again refined with an Residual Convolutional Units(RCU) block. All the convolutions performed in the RCU and Chained Residual Pooling blocks follows the spirit of ResNet architecture in having residual connections. This ensure the gradient flow during learning process. In the following paragraph, we discuss RCU and Chained Residual Pooling blocks in detail.

The RCU shown in Figure 4.2(b) is simply a ReLU non-linearity followed by 3×3 convolution applied as a Residual connection. The Chained Residual Pooling shown in Figure 4.2(c) consists of ReLU followed by a sequence of 5×5 pooling and 3×3 convolutions where each of the convolutions are applied as residual connections.

We use RefineNet as the backbone network for semantic segmentation and pose estimation.

4.2 Evaluation Metric

The most common metric used to measure the accuracy of the semantic segmentation models is Intersection over Union (IoU). For example, the PASCAL Visual Object Challenge, one of the major challenges in the computer vision community (Everingham et al. (2010)), uses IoU as the evaluation metric. The IoU score is

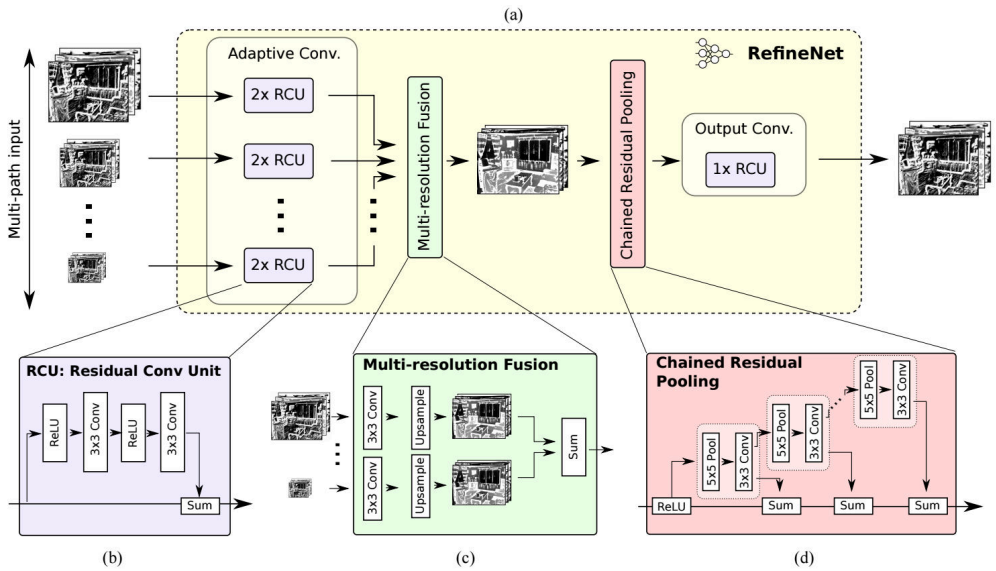

Figure 4.2: A RefineNet block combining feature from different stages of the backbone model. Source: Lin et al. (2017).

defined as,

$$IoU = \frac{True\,Positive}{True\,Positive + False\,Positive + False\,Negative}$$

The positive and negative counts are aggregated over all the images; and the IoU score is averaged over the classes to obtain the mean IoU.

5 Pose Estimation

Pose Estimation is predicting the 6D pose of an object in the world coordinate frame. Estimating 6D pose of the objects is usually done using variants of Iterative Closest Point (ICP) algorithm. ICP is only an refinement algorithm and the quality of the refinement depends on the initial guess. ICP works with two point clouds: the observed point cloud and the reference point cloud and computes the transformation that minimizes the distance between them. Finding the correspondences between two point clouds is a difficult process. Many ICP variants have been proposed for finding better correspondences, from which point-to-point and point-to-plane are the most widely used. For finding the initial estimation R, the most commonly used approach is to initialize with the result of performing singular value decomposition (SVD) of the observed point cloud into UDV^T as,

$$R = UV^T$$

Since the quality of the final estimation depends on the initial estimate, this method of initialization does not always guarantee the best results. Inspired by the recent use of convolutional neural networks for 6D pose estimation (Kendall, Grimes, and Cipolla (2015) , and Koo et al. (2017)), we propose a CNN to estimate the pose of the object.

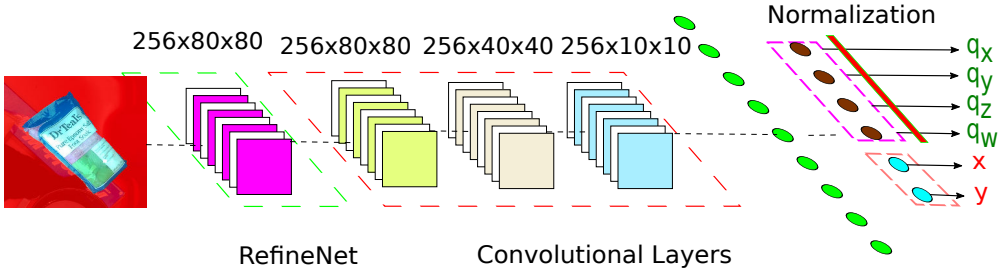

Figure 5.1: Pose Estimation network architecture.

5.1 Network Architecture

The pose estimation network is trained with the features extracted from RefineNet model as input. The last layer of the RefineNet model has 256 feature maps. We use RGB images of dimension 320 × 320 as input for RefineNet and the resulting feature maps is of dimension 80 × 80. We use the result of the semantic segmentation network to crop a rectangular region containing pixels belonging to the target object. It is beneficial to feed the segmentation results into the pose estimation network. We do this by pushing the pixels in the crop that does not belong to the target object towards red. We want the objects to be maximal in the crop, but also have constant scaling for all views. We achieve this by specifying the size of the crop to be the expected size of the object in the image given the camera parameters and the size of the object; such that the biggest view of the object would fit. The architecture of the network is shown in Figure 5.1. It consists of three convolutional layers followed by two fully connected layer and the final output layer. The pose estimation network predicts 5D pose of the object(x and y of the translation component, and a unit quaternion for the rotation component). We do not train the network to predict the z component (depth) of the translation since generating synthetic images with depth needs sophisticated rendering pipeline and the training process is time consuming. Also the depth can be predicted from by projecting the predicted x and y component into the 3D space using the camera parameters. We evaluated two variants of the network that differ in the number of neurons in the final output layer. One of the variants has 6 neurons in the output layer—an unit quaternion representing the rotation component and the translation component. Output of the neurons predicting the unit quaternions are normalized explicitly alleviating the networks to learn the normalization. The first variant is trained with a modified version of mean-square-error(MSE) as:

$$Error(\hat{Y}, Y) = \sum_{i}^{n} (\hat{Y}_i - Y_i)^2,$$

where $\hat{Y}_i$ is the ground truth pose and Y_i is the predicted pose for an image i. Note that the error is not averaged over the batch as in mean-square-error.

The second variant has $6N$ neurons, where N is the number of objects. Each set of 6 neurons is associated with an object. This allows the network to adapt to different object classes. While training, only the predictions corresponding to the object is subjected to the error metric and the other predictions are simply ignored. Like the first variant, quaternion predictions are normalized explicitly. The error functions is defined as

$$Error(\hat{Y}, Y) = \sum_{i}^{n} (\hat{Y_i} - Y_{ij})^2,$$

where $\hat{Y_i}$ is the ground truth pose and Y_{ij} is the predicted pose for an image i containing object j.

Since the translation component and the rotation component are of different scales, we need weighting factors for each of the component in the error function. The weighted error function is defined as

$$Error(\hat{Y}, Y) = \sum_{i}^{n} \frac{1}{\beta} (\hat{Y_i^{xy}} - Y_{ij}^{xy})^2 + (\hat{Y_i^{quaternion}} - Y_{ij}^{quaternion})^2.$$

5.2 Evaluation Metric

The above-defined error metric consisting of two different components balanced by an explicit weighting is used only as a metric for the network learning process. The performance of the network in estimating the pose of the objects is measured individually for each of the component.

5.2.1 Translation error

The translation error is simply the Euclidean norm between the predicted and the ground truth pixel.

5.2.2 Rotation error

The rotation error metric is measured by two metrics: the Euclidean norm and angle between the predicted and the ground truth quaternion.

Quaternion, Q, is represented as $[q_w, q_x, q_y, q_z]$; angle, θ, between two quaternions, q_1 and q_2 is computed as,

$$\begin{aligned} q_2^{-1} &= [q_{2w}, -q_{2x}, -q_{2y}, -q_{2z}] \\ q &= q_1 * q_2^{-1} \\ \theta &= 2 * arccos(q_w) \end{aligned} \tag{5.1}$$

6 Implementation

An overview of the pipeline implemented is shown in Figure 6.1. The input RGB-D image is provided to the Semantic Segmentation model and from the resulting segmented image, a crop around the object that we need pose estimation is extracted. The red component for the pixels in the crop that does not belong to the object is increased while the blue and green components are decreased. This resulting image is passed though the RefineNet to extract a set of feature maps which is fed to the pose estimation model.

In this thesis, we use an existing semantic segmentation pipeline implemented using the Torch framework[1], which was developed for the APC 2016. We adapt it to extract high-level features from synthetic scenes and implement the pose estimation model using the newer PyTorch framework[2].

While training the pose estimation network, for an input image, the RefineNet features needs to be extracted just once and can be stored to the SSD hard-drive. However, loading the data from the SSD hard-drive is slower than the time needed for GPU to process the data. Thus we implemented multi-threaded data loading as shown in Figure 6.2. The training thread at the beginning of an epoch pushes the list of directories to be loaded into the directories queue. The loader threads are waiting for directories in the directories queue and when it appears, the loader threads pops the directories from the queue, loads the data in the directory, and pushes the data to the data queue. The training thread, after pushing the list of the directories, waits for data in data queue and once it appears, pops the data form the data queue and starts training. All the thread synchronization is done using the Python's[3] and Torch's [4] thread-safe queue libraries implicitly and not handled explicitly.

[1] `http://torch.ch`
[2] `http://pytorch.org`
[3] `https://docs.python.org/3/library/queue.html`
[4] `https://github.com/torch/threads`

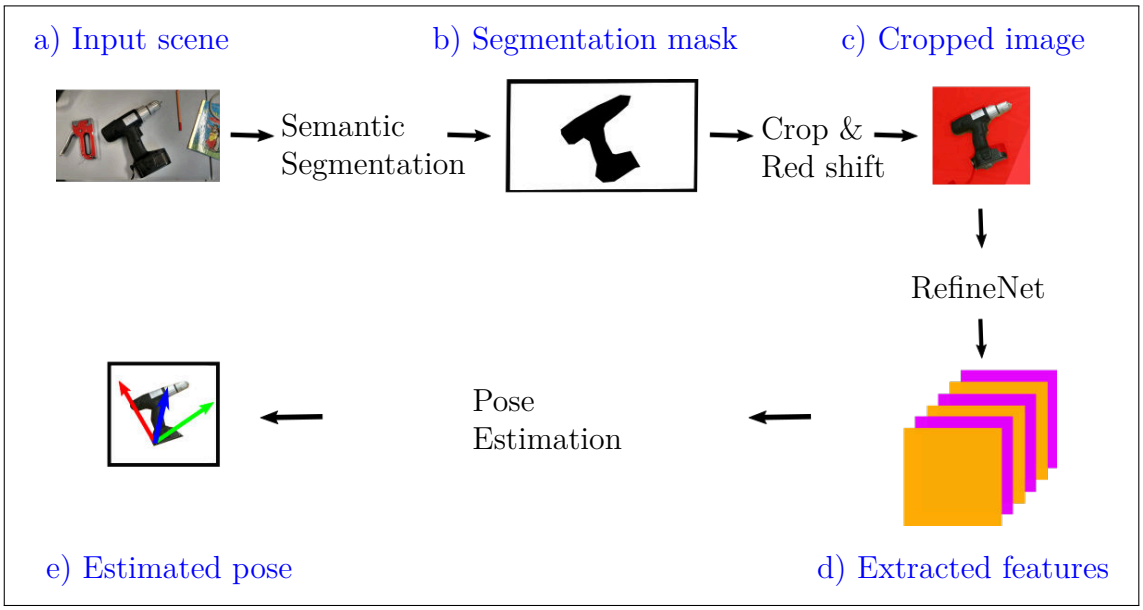

Figure 6.1: Scene understanding pipeline.

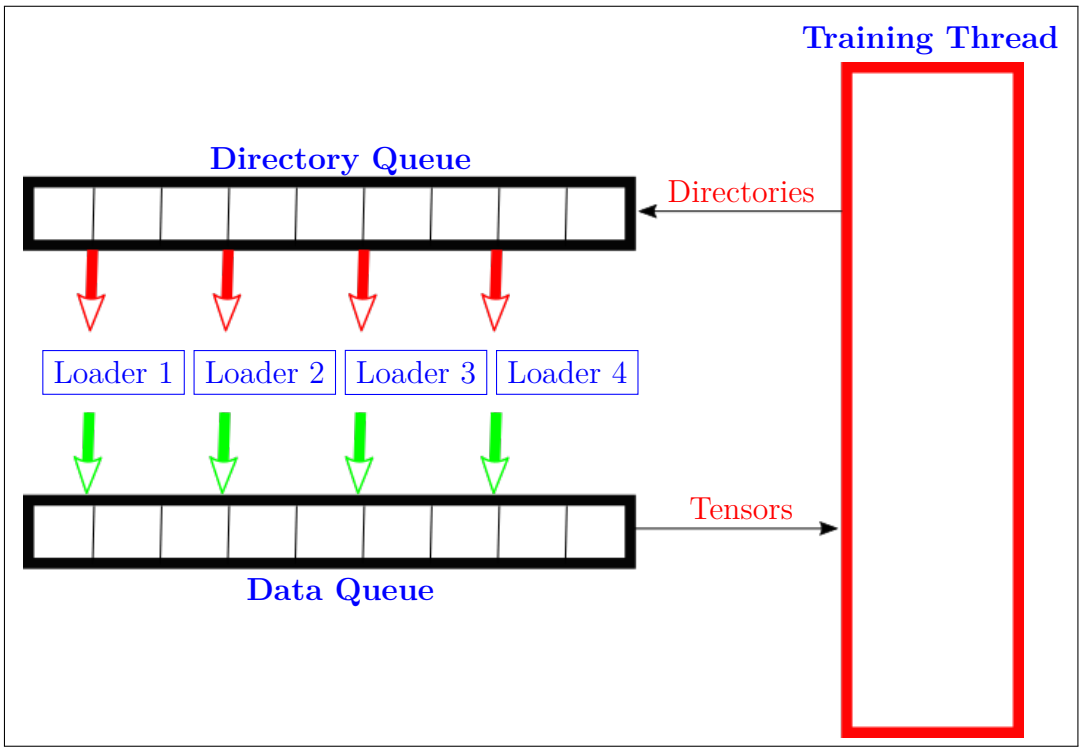

Figure 6.2: A schematic representation of the multi-threaded data loading.

7 Evaluation

7.1 Pose Estimation

We evaluated the pose estimation model trained on synthetically generated training data as discussed in the Section 3.1.4. For each object, depending on the shape of the object, we capture two or three turntable sequences. Each sequence consists of 20 frames. We augment each frame with 60 new sampled orientations. Thus the dataset generated consists of 2400 or 3600 images with ground truth poses. We use this dataset in the following evaluations.

7.1.1 Single Vs. Multi Block Output Comparison

We evaluated both variants of the model: multi-block output; single-block output on three objects. We also evaluated the variants on synthetic images with and without occlusion. To generate occlusion, we randomly sample a portion of the bounding box that contains the object and set the pixels in the portion as background.

The Table 7.1 and Table 7.2 shows the comparison of single-block vs. multi-block variants with no occlusion in training data and Table 7.3 and Table 7.4 shows the comparison of single-block vs. multi-block variants with synthetic occlusion in training data. From these results, we observe that single block output variant works slightly better with occlusion while the multi block output works comparatively better when the there is no occlusion in the input image. One of the reasons for the multi-block variant performs in the absence of occlusion could be that the training objective for the multi-block variant does not penalize for wrong object recognition; we simple discard the poses estimated in the blocks that does not correspond to the objects under consideration. i.e. we do not force the multi-block variant to perform object recognition as a part of post estimation. On the other hand, the reason for single-block variant performs better when occlusion is present could be that the single-block variant has less danger of overfitting, because the output is forced to adapt to different objects all the time—like a regularizer that introduces noise on the labels.

Table 7.1: Pose Estimation Errors - Translation component single-block vs. multi-block output without any occlusion.

	Training		Validation	
	Translation [pixel]		Translation [pixel]	
	Multi-block	Single-block	Multi-block	Single-block
Epsom salts	2.28	4.85	3.32	5.69
Toilet paper	2.41	5.08	3.79	5.98
Yellow windex	2.25	4.07	3.41	4.75
Average	2.31	4.67	3.51	5.47

Table 7.2: Pose Estimation Errors - Rotation component single-block vs. multi-block output without any occlusion.

	Training				Validation			
	Quaternion [norm, $\cdot 10^{-2}$]		Angular [degrees]		Quaternion [norm, $\cdot 10^{-2}$]		Angular [degrees]	
	Multi	Single	Multi	Single	Multi	Single	Multi	Single
Epsom salts	1.63	2.88	1.80	3.32	3.83	4.43	3.19	4.92
Toilet paper	1.94	3.06	1.68	3.38	4.75	5.73	3.09	4.98
Yellow windex	2.04	3.03	1.86	3.48	3.23	4.71	2.78	4.57
Average	1.87	2.99	1.78	3.39	3.93	4.97	3.02	4.82

Table 7.3: Pose Estimation Errors - Translation component single-block vs. multi-block output with occlusion.

	Training		Validation	
	Translation [pixel]		Translation [pixel]	
	Multi-block	Single-block	Multi-block	Single-block
Epsom salts	6.95	6.29	8.31	7.44
Toilet paper	5.89	6.14	7.72	7.73
Yellow windex	5.29	5.33	6.34	6.33
Average	6.04	5.92	7.46	7.17

Table 7.4: Pose Estimation Errors - Rotation component single-block vs. multi-block output with occlusion.

	Training				Validation			
	Quaternion [norm, $\cdot 10^{-2}$]		Angular [degrees]		Quaternion [norm, $\cdot 10^{-2}$]		Angular [degrees]	
	Multi	Single	Multi	Single	Multi	Single	Multi	Single
Epsom salts	4.94	2.92	4.64	3.38	5.81	4.85	6.26	4.98
Toilet paper	3.37	3.04	3.83	3.44	6.51	6.15	6.13	5.56
Yellow windex	4.06	2.86	5.61	3.27	6.59	4.66	6.82	4.52
Average	4.12	2.99	4.69	3.36	6.03	5.22	6.04	5.02

7.1.2 Data Requirement Evaluation

The accuracy of the model heavily dependents on the volume of training data used for training the model. But training with more volume of data requires more computational resource and time. Thus finding the least volume of data that is suffice to achieve good accuracies is vital for the model to be deployed in real world—particularly with flexibility of adding new objects incrementally. We empirically found the minimum volume of data needed for training the pose estimation model. As discussed in Section 3.1.4, for a captured frame, we create synthetic scene by sampling new orientation (described in step d) of Figure 3.5) . In Figure 7.2, Figure 7.3, and Figure 7.4, the training process of training with 24, 36, and 60 sampled orientations for each of captured frame is shown respectively. In Figure 7.2, and Figure 7.3 we can observe that while training loss looks good, the validation loss is high suggesting insufficiency of the training data. Figure 7.4 shows validation loss is comparable to training loss. The final accuracies of these training process are compared in Figure 7.1. In the rest of the experiments we create training dataset by generating 60 synthetic images for each of the captured frame.

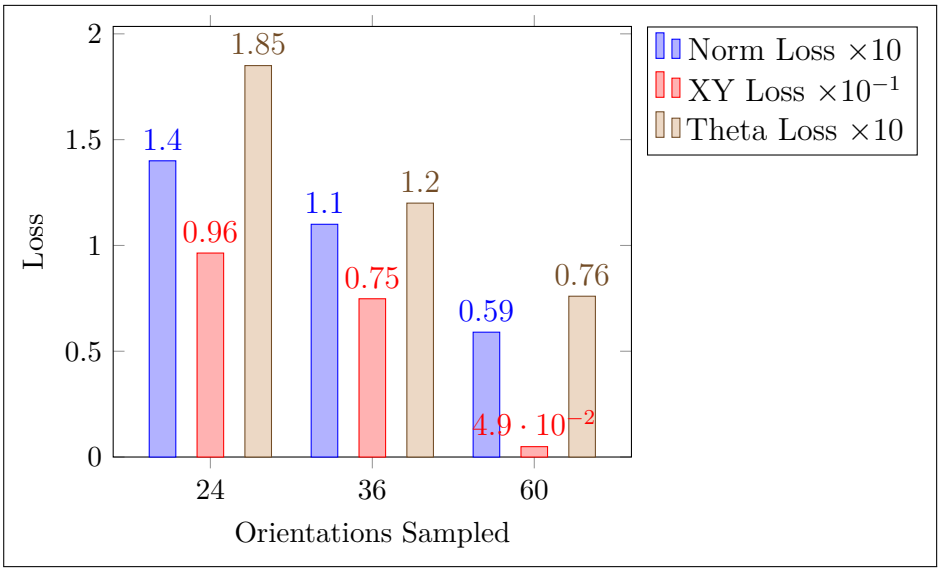

Figure 7.1: Accuracies of the pose estimation model trained with different number of orientations sampled.

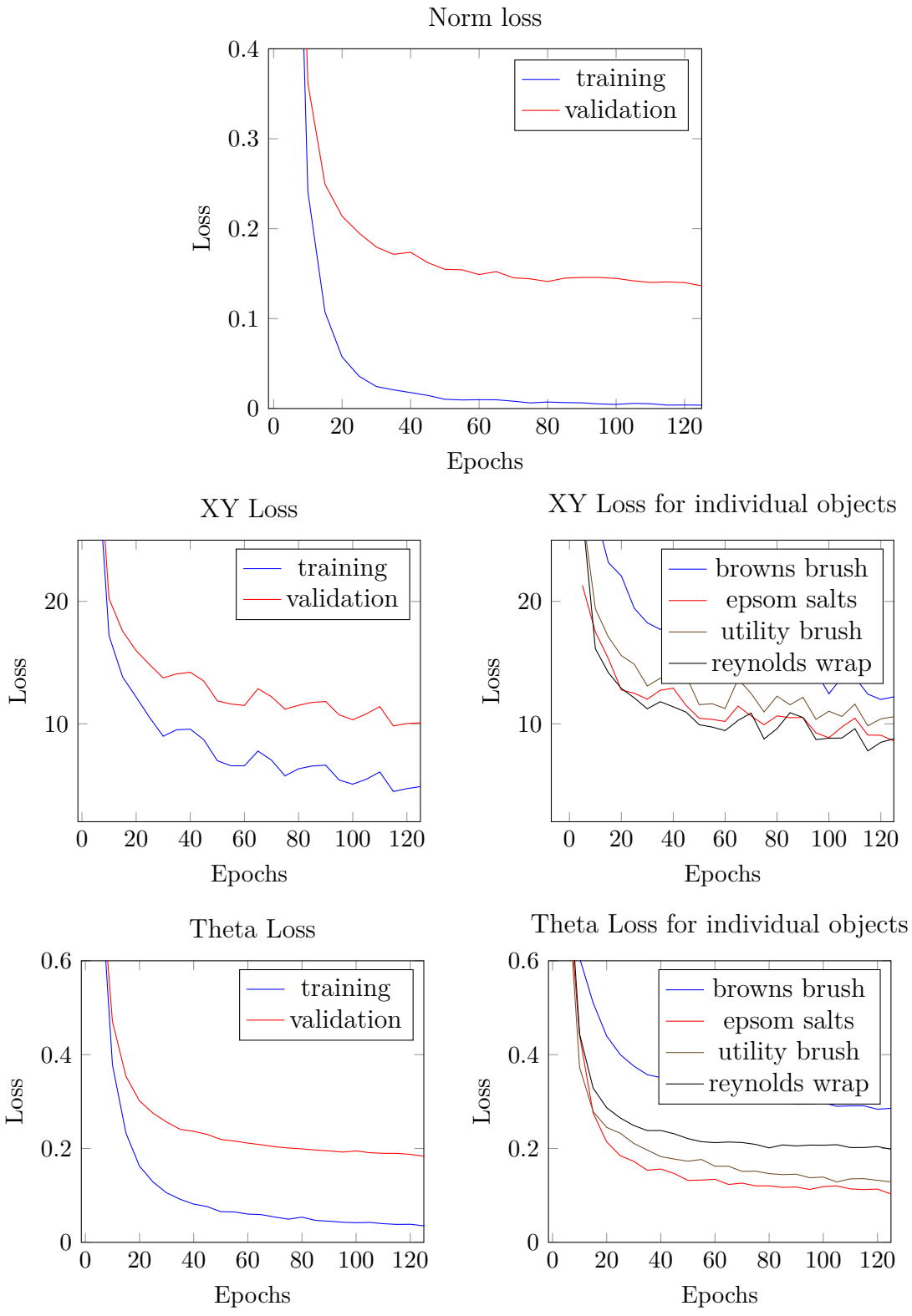

Figure 7.2: The training process of training with 24 sampled orientations for each of the captured frames.

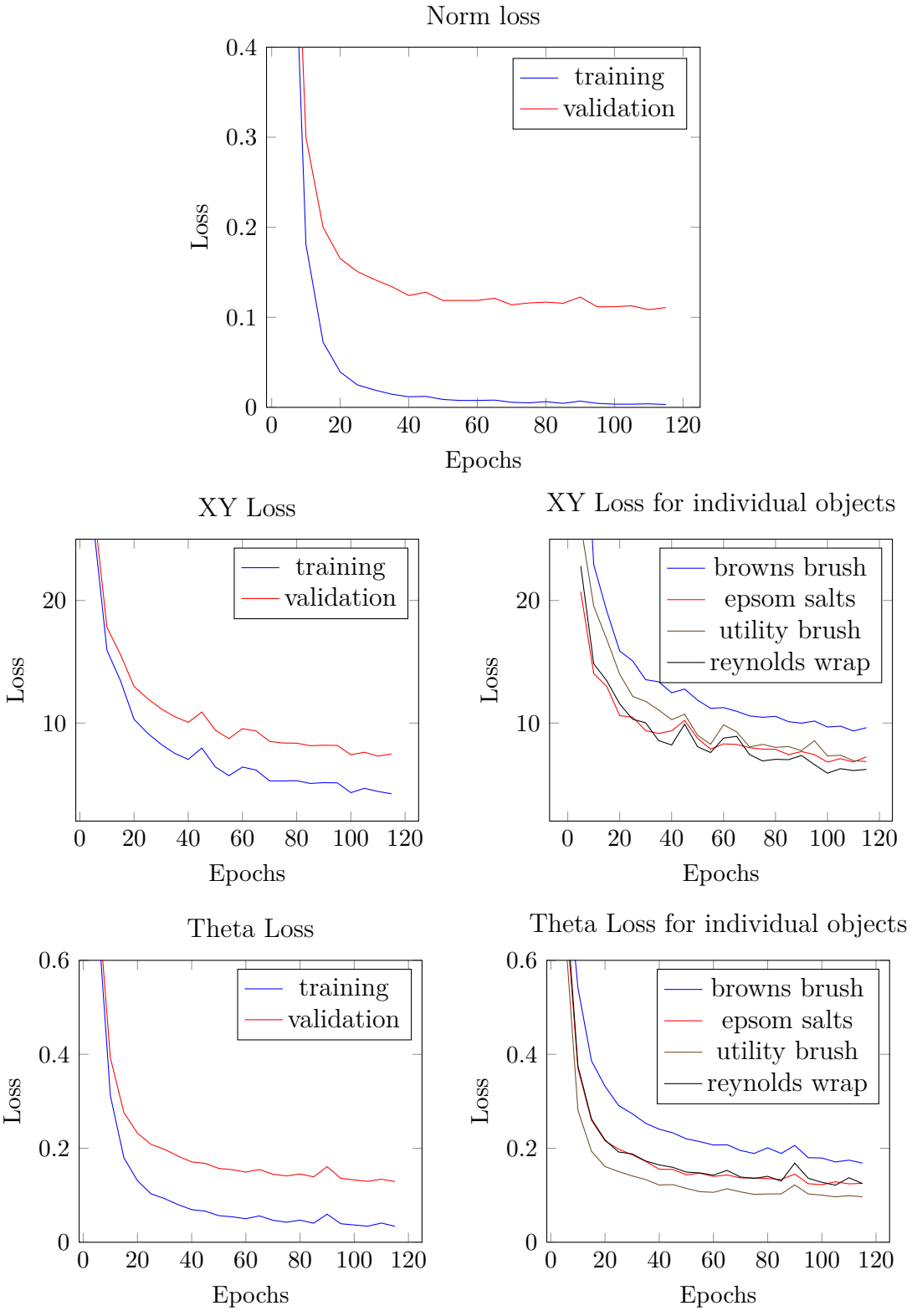

Figure 7.3: The training process of training with 36 sampled orientations for each of the captured frames.

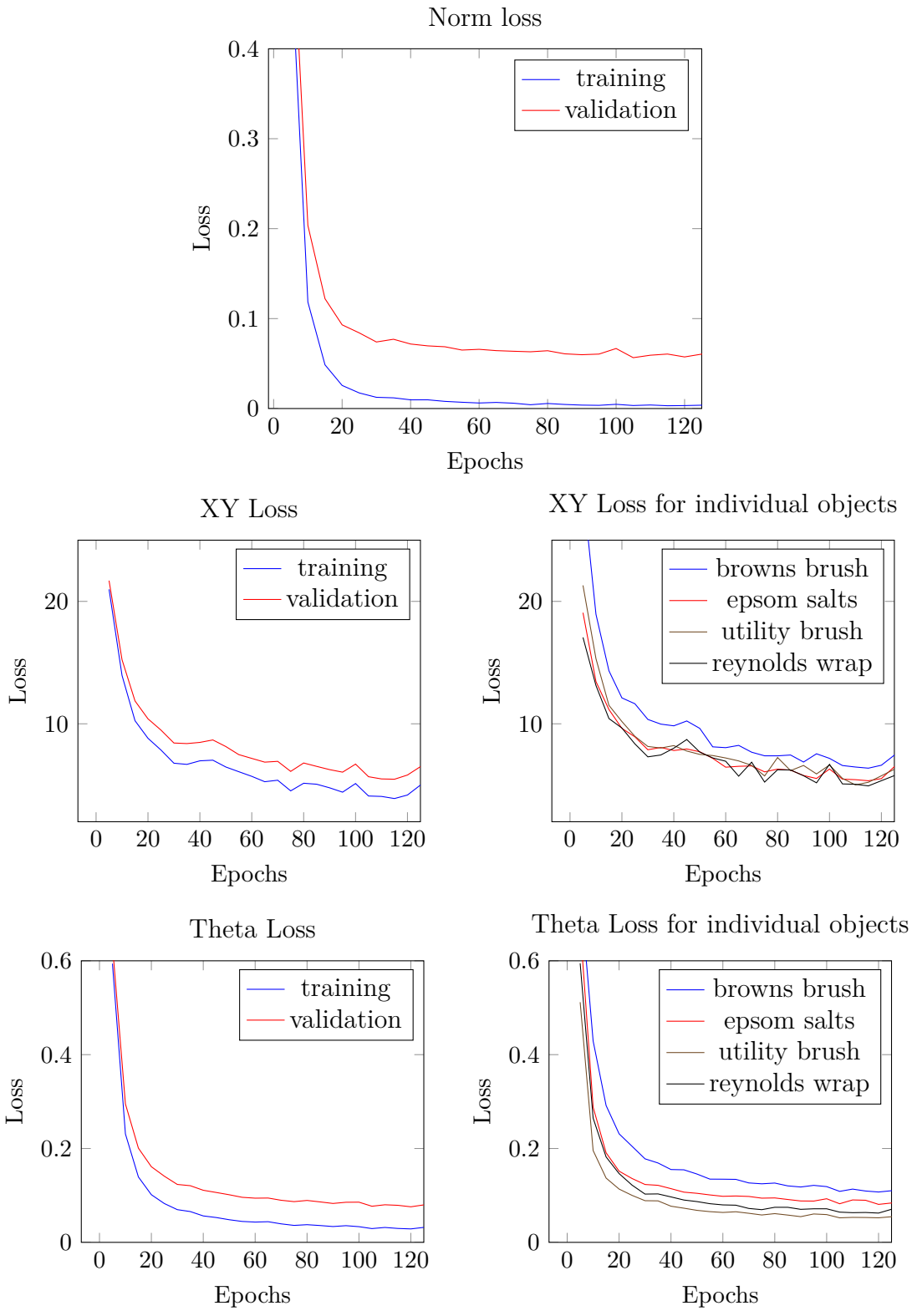

Figure 7.4: The training process of training with 60 sampled orientations for each of the captured frames.

7.1.3 Scenes Requirement Evaluation

In Section 7.1.2 we empirically found the minimum number of synthetic images needed for training the pose estimation model. We concluded that for each of the captured frame, we need to generate 60 new images by sampling along z axis and superimposing the generated scene in a realistic background. We experimented with generating more synthetic scenes by superimposing the images of sampled orientation (described in step d) of Figure 3.5) on different backgrounds (described in step e) of Figure 3.5).

The Figure 7.5 compares the training process of generating five synthetic scene and one synthetic scene for each of sampled orientation respectively. The final accuracy of the two resulting model were similar, and thus we can conclude that we need only one synthetic scene for each of sampled orientation. One probable explanation for not needing multiple synthetic scenes is that network learns to ignore the background pixels of the scene that has dominant red component from just one synthetic scene of each sampled orientation —suggesting that our encoding of the segmentation results is easily understandable by the network.

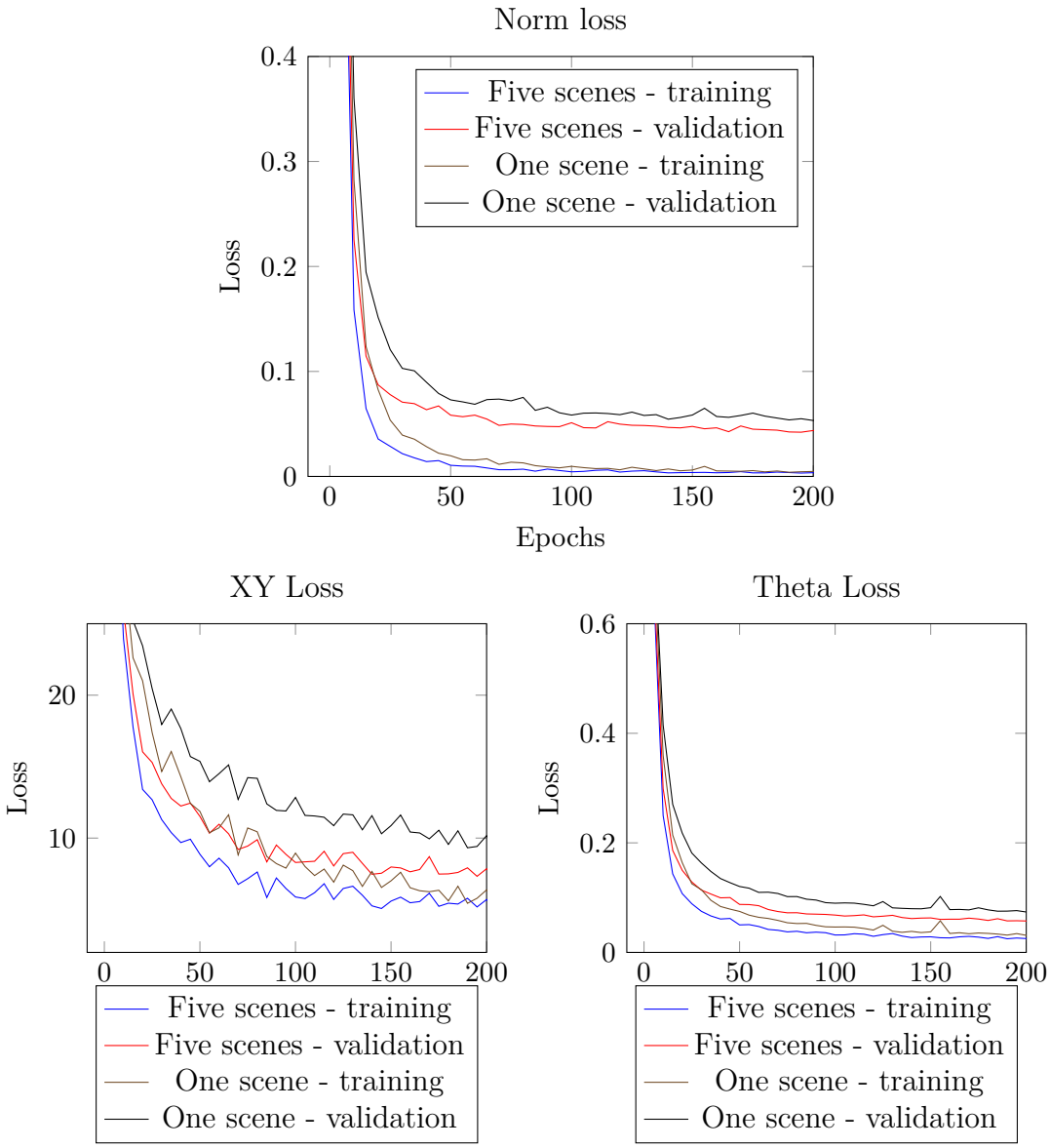

Figure 7.5: The comparison of the training processes of training with five and one synthetic scene for each the sampled orientations.

7.1.4 Effect of Occlusion

In the Section 7.1.1, we showed that the pose estimation network can learn to estimate the pose of the object, despite the object being partially occluded. We generate occlusion by sampling as rectangular portion close to one of the four sides of the bounding box containg the object in the image. We set all the pixels in the portion as the background pixels. We then compute the precent of occlusion and discard the iamge if the percentage of occlusion is more than 50. Examples of the occluded images are shown in Figure 7.6. We further investigated the effect of occlusion in the accuracy of the pose estimation. In Figure 7.7, and Figure 7.8 we,respectively, present the norm loss in predicting the translation component, and the rotation component when object is occluded between 0 and 50 percentage. As we discussed in Section 7.1.1, the pose estimation model variant with single-block output is less susceptible to occlusion. From these charts we can see the loss increases with the increase in percentage of occlusion almost linearly.

Figure 7.6: Examples of occlusion. Top row: RGB image of the objects. Bottom row: Occluded objects.

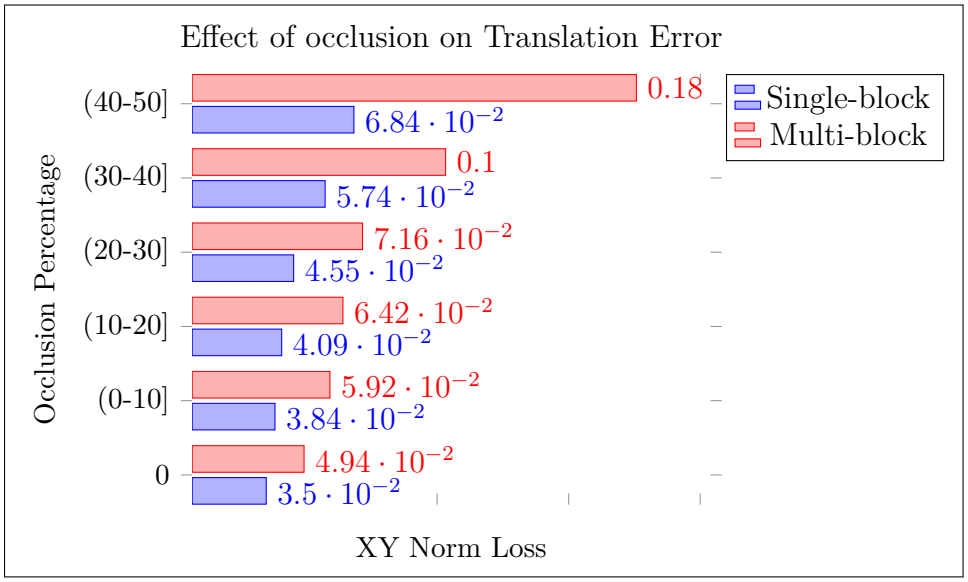

Figure 7.7: Effect of occlusion on Translation error.

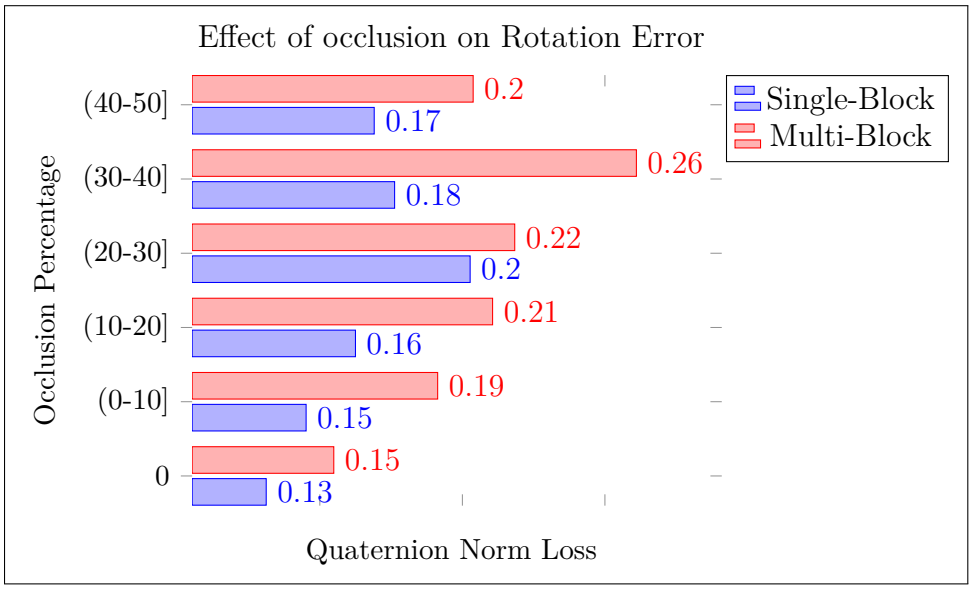

Figure 7.8: Effect of occlusion Rotation error.

7.1.5 Effect of Output Scaling

In many machine learning problems where the input features and output of the models are of different scale, the model learned preforms better when the input features and the output of the models are scaled to have zero mean and unit standard deviation (Juszczak, Tax, and Duin (2002), Aksoy and Haralick (2001), Youn and Jeong (2009)). In the case of the pose estimation network, the output of the network has two components; translation, and the rotation components. While the translation component represented as quaternions range between 0 and 1, the translation component representing the location of the origin of the object in the image space ranges from 0 to 320. We experimented with two different variants of pose the estimation model; the translation component of the model is scaled between -1 and 1, and unscaled version ranging between 0 and 320.

Both variants needs a balancing factor β to balance the difference in the scale of the translation and rotation losses as discussed in Chapter 5. But in the case of scaled variant, the two losses are of a very similar scale and differ only in learning process(one of the losses might be easier to minimize than the other). Thus β is needed only to make the network emphasize more on one of the losses. To allow this, the loss function can be reformulated as follows,

$$Error(\hat{Y}, Y) = \sum_{i}^{n} \beta * (\hat{Y_i^{xy}} - Y_i^{xy})^2 + (1 - \beta) * (\hat{Y_i^{quaternion}} - Y_i^{quaternion})^2$$

Table 7.5: Effect of output scaling.

	Translation [pixel]		Rotation [degree]	
	train	val	train	val
scaled	4.6	6.8	3.2	8.2
unscaled	3.24	6.6	2.9	6.9

The hyper-parameter search for β is easier in the case of scaled variant. The training process of the two variants are shown in Figure 7.9, and Figure 7.10. While the orientation loss curves (theta loss) in both variants looks very similar, translation loss(XY loss) is smoother in the case of scaled variant suggesting that the learning is faster. The final accuracy shown in Table 7.5 is similar. One of the reasons that unscaled version performed comparable to scaled version is the usage of ReLU activations whose activations are unbounded when active.

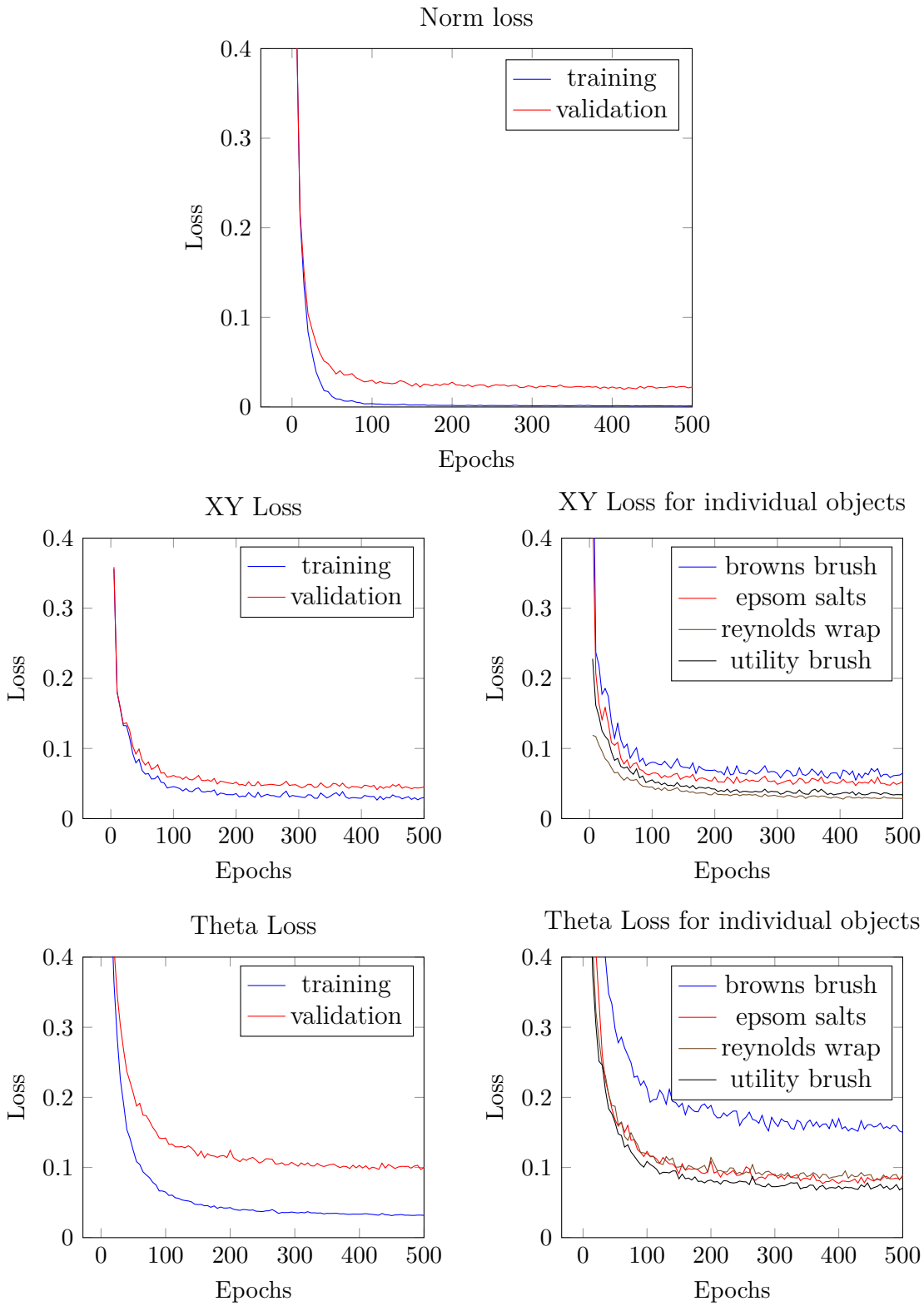

Figure 7.9: Training process when x and y outputs are scaled between -1 and 1.

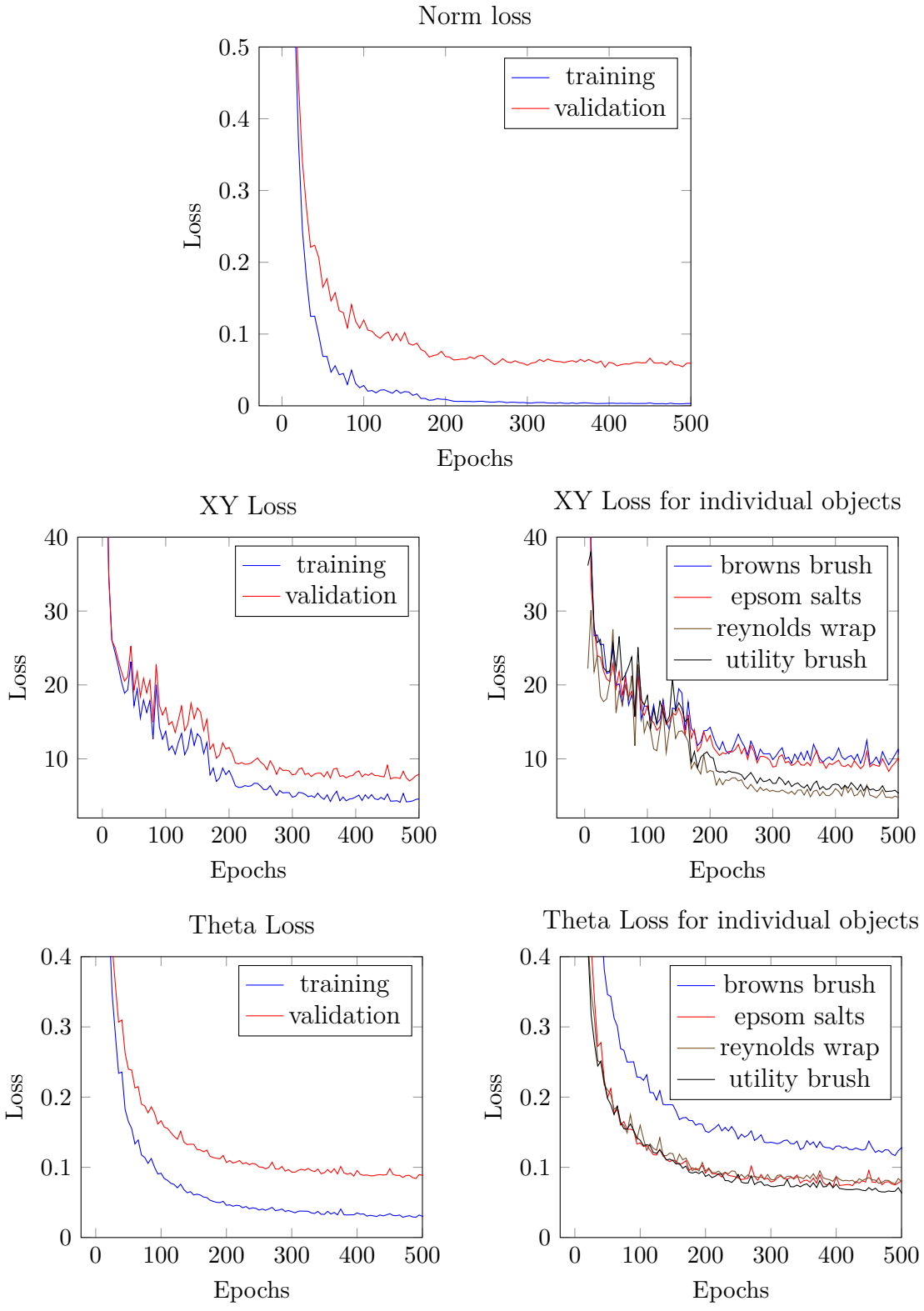

Figure 7.10: Training process when x and y outputs are not scaled.

7.1.6 Evaluating the Generalization Properties

The objects that we encounter in the real world are often instances of some category of object. We humans need not to learn to recognize/operate each single instance of the category of the object. We can transfer the knowledge that we acquire in one of the instances to other instances of the same category. With enough training data, CNNs are shown to generalize to new instances of same category. In this section we evaluate the generalization properties of the pose estimation network. We trained the pose estimation networks on a set of four drills shown in Figure 7.11. The origin and the pose of a drill is shown in Figure 7.12. We then evaluated the generalization properties of the networks by testing on an unseen drill shown in Figure 7.13. The accuracy of the pose estimation network in predicting the pose of the unseen drill with and without occlusion is shown in Table 7.6. We investigated the pose estimation for the unseen cases further by displaying the image of the seen drill that is closest to the predicted orientation in the entire seen dataset. The unseen input images and the image closest to the predicted orientation are juxtaposed in Figure 7.14, Figure 7.15, Figure 7.16, and Figure 7.17. In these figures, the ground truth translation and the predicted translation are displayed as small white and green dots in the input image displayed on the left-hand side.

The Figure 7.14 shows some example cases where input is not occluded and the predicted pose is relatively good. The Figure 7.15 shows some typical failure cases. In most cases where the model performed poorly was when the unseen drill was lying with its base facing the camera. Similarly, the Figure 7.16 shows the cases where predicted pose was good even though the unseen drill in input image is occluded and the Figure 7.17 shows the failure cases. The top right image of Figure 7.17, we can see the effect of the occlusion in predicting the translation. Due to the base of drill being completely occluded—despite the orientation predicted is quite good—the translation prediction fails.

In both occluded and non occluded cases, the translation estimation is off by a few pixels and from this observation, we could infer that the pose estimation model doesn't learn the definition of the origin for a novel drill based on the seen drills. Rather the model tries to estimate the translation of the novel drill from the set of features learned on the seen drills and this doesn't generalize perfectly.

Table 7.6: Generalization properties of the Pose Estimation network.

	without occlusion	with occlusion
Translation [pixel]	36.34	39.52
Quaternion [norm, $\cdot 10^{-2}$]	0.362	0.397
Angular [degrees]	33.6	38.21

Figure 7.11: The drills on which the model was trained.

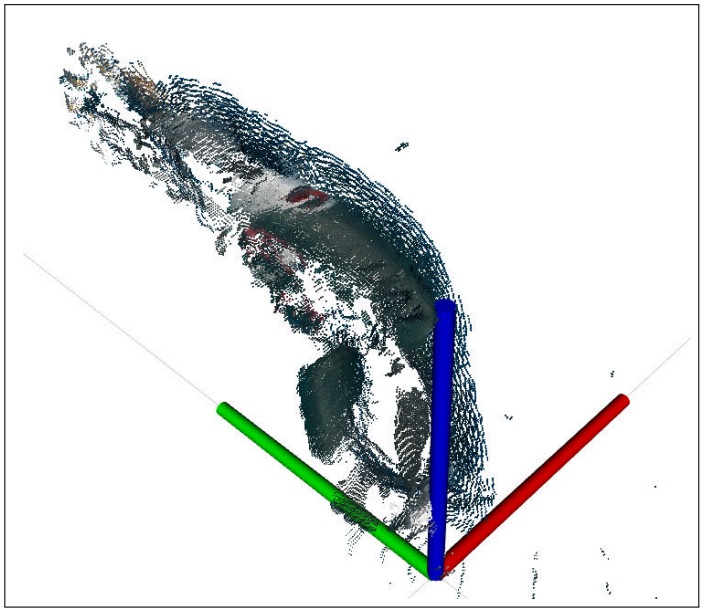

Figure 7.12: The origin and the coordinate frame of a drill

Figure 7.13: The new drill on which the model was evaluated.

Figure 7.14: Generalization without occlusion; working cases.

Figure 7.15: Generalization without occlusion; typical failure cases.

Figure 7.16: Generalization with occlusion; working cases.

Figure 7.17: Generalization with occlusion; typical failure cases. Top: While the orientation estimation is relatively good, the translation estimation is poor due the base being occluded.

7.1.7 Dealing With Invariance

Many objects have the inherent property of being invariant to viewing angles along one or more axis. Consider, for example, the dumbbell shown in Figure 7.18 is looks very similar when rotated along the z axis(shown in blue) except for a small difference the positioning of the label close to the origin which is negligible. Also, in terms of manipulation, the pose of object is invariant along z axis and also invariant to 180°rotations along y axis (shown in green). But the automatic ground truth generation process discussed in Section 3.1 does not treat these invariances. Forcing the pose estimation network to estimate the precise pose from the images with little variance will hamper the learning process. To deal with this, we experimented by assigning the same ground truth for all the invariant poses along an axis as shown in Figure 7.19. The learning process with variant poses for visually similar images is shown in Figure 7.20 and Figure 7.21 shows the learning process with dataset containing invariant poses for visually similar images.

We can observe that the translation and rotation error of the object "Hand weight" (dumbbell) drops significantly faster in the invariant poses case compared to variant poses case and the final accuracy is slightly better. The final accuracies are compared in Table 7.7.

Table 7.7: Invariance experiment: Accuracy comparison.

	Variant Poses				Invariant Poses			
	Translation [norm, $\cdot 10^{-2}$]		Rotation [degrees]		Translation [norm, $\cdot 10^{-2}$]		Rotation [degrees]	
	train	val	train	val	train	val	train	val
Epsom salts	3.48	5.04	3.39	6.53	3.22	4.76	3.16	6.65
Hand weight	2.67	2.67	1.67	1.72	1.44	1.52	0.74	0.80
Utility brush	1.92	2.90	3.49	5.68	1.87	3.09	3.36	5.67
Average	2.68	3.54	2.81	4.71	2.18	3.16	2.41	4.47

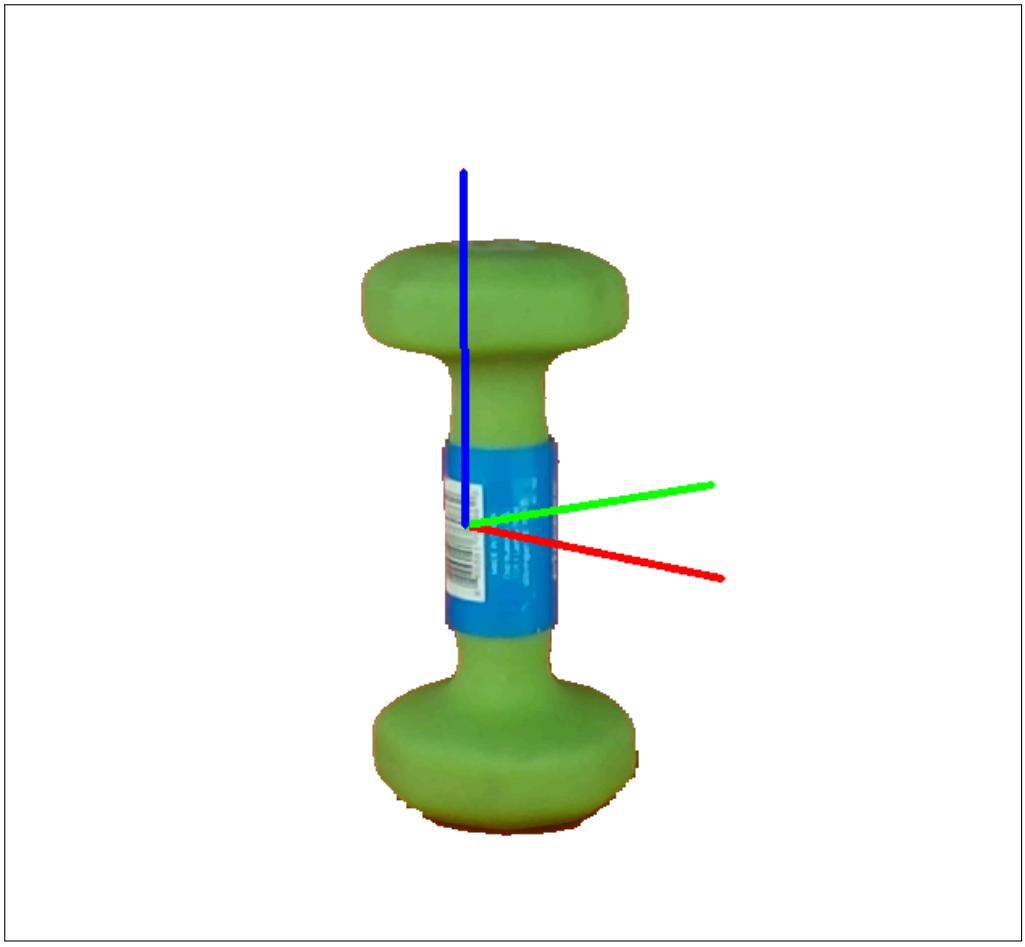

Figure 7.18: Dumbbell with origin and coordinate system.

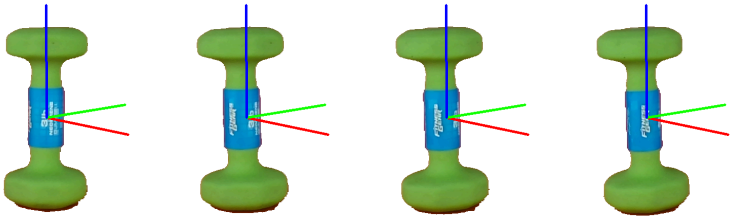

Figure 7.19: Poses with little invariance are assigned the same ground truth.

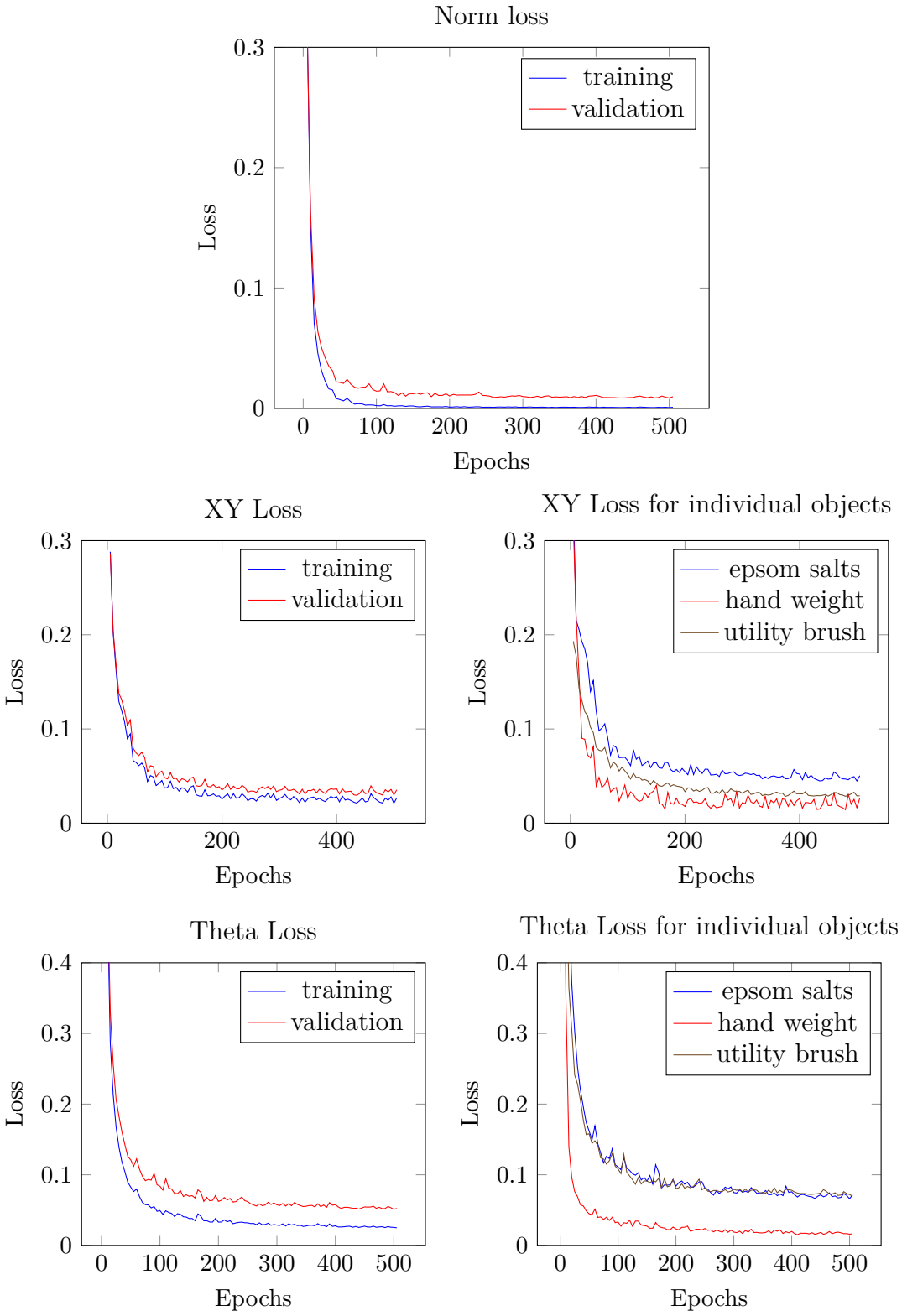

Figure 7.20: Invariance experiment: Variant Poses.

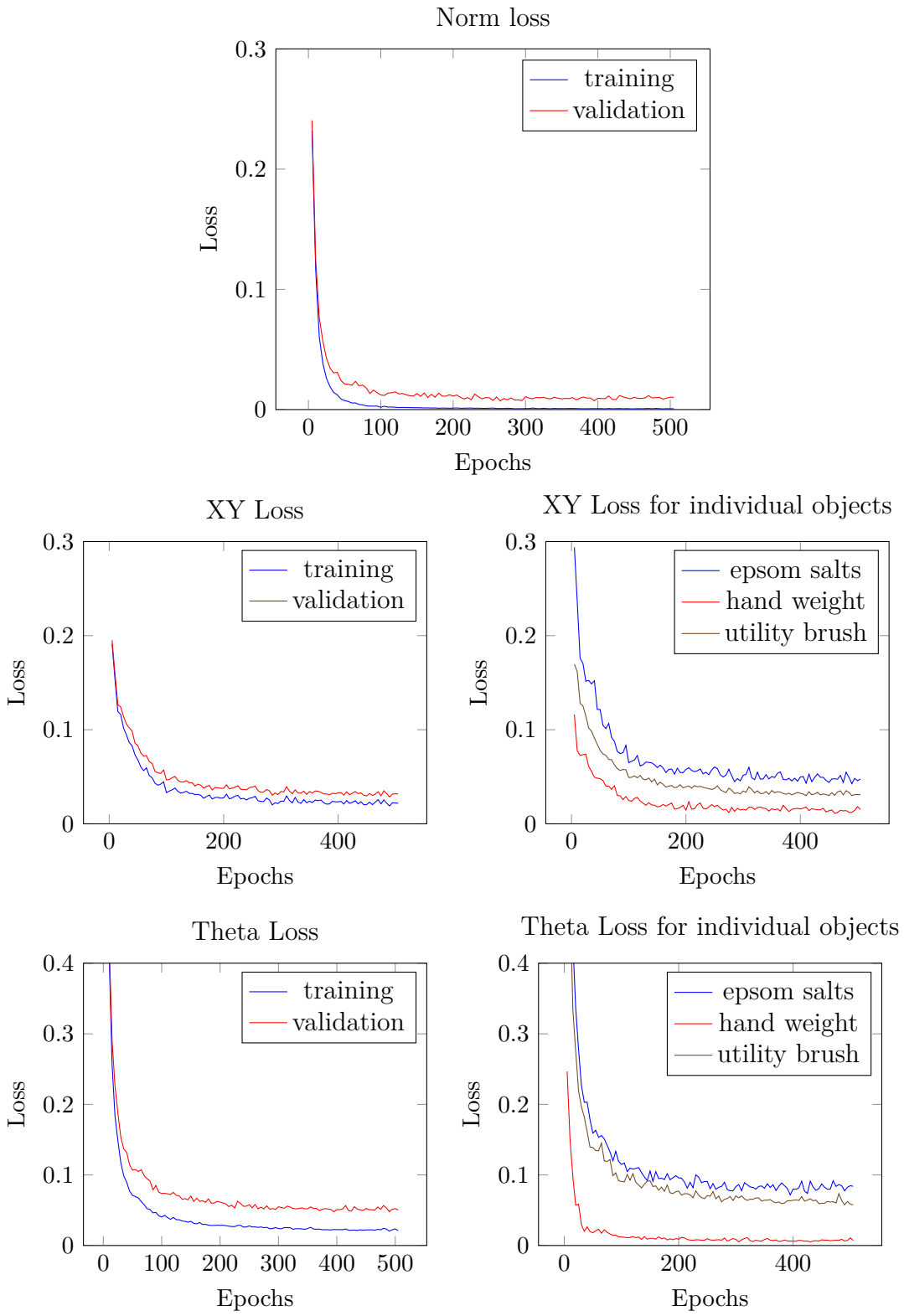

Figure 7.21: Invariance experiment: Invariant Poses.

7.1.8 Copy/Retain Weights experiment

The conventional machine learning work-flow involving a fixed dataset to train a model does not suit real-world applications; for a model to be pragmatic, the ability fast retrain on new data is crucial. Some of the recent works (T. Chen, Goodfellow, and Shlens (2015), Li and Hoiem (2016)) address the problem of modifying the architecture of the network without destroying the functionality learned. One of our goals in designing the the pose estimation network was to have the ability to fast train the model on new objects. Our design of multi-block output was motivated by this necessity. We evaluated the ability of the two variants of the pose estimation network to fast learn a new objects. We perform this evaluation by training both variants on three objects initially, and then adding one more object to the dataset. While learning on the new object, we experimented with retaining/copying weights of the neurons from the initial run and assess the benefit of retaining/copying the weights. Figure 7.22, and Figure 7.24 shows the training process of training single-block and multi-block variant with the initial three objects respectively. Figure 7.26 shows the training process of training the multi-block variant from scratch and Figure 7.23 shows the training process of training single-block variant by retaining the weights from the initial run and Figure 7.25 shows the training process of training multi-block variant by retaining weights of the convolutional layer and the three output blocks corresponding the the initial three objects. The output block corresponding to the new object (a driller in our case) was initialized by copying the weights corresponding to the driller in the initial set of objects. While training the new object, the number of the training images corresponding the old set of objects were halved. Note that the plots shows the training process only the first 150 epochs, since, eventually, the final accuracies of the variants are similar and we are interested not in the final accuracy but in the rate of learning process. From these results we can observe that multi-block variant learns faster when the weights are copied and the single-block variant also learns equally faster without forgetting the older objects. We did not observe any particular benefit of having multi-block output and single-block variant not forgetting the initially learned objects while learning the new object.

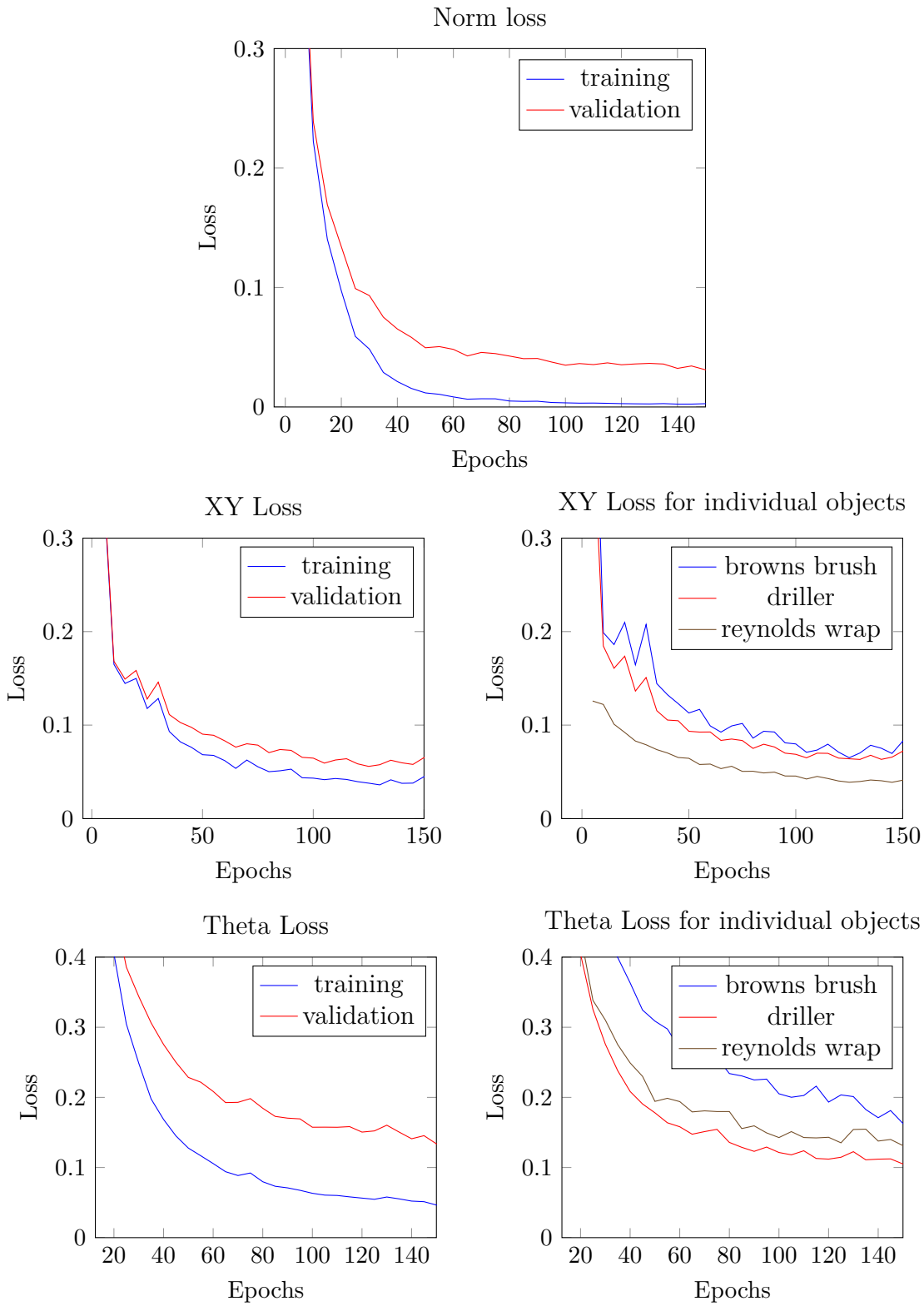

Figure 7.22: Copy weights experiment: Single-block output; initial run with three objects.

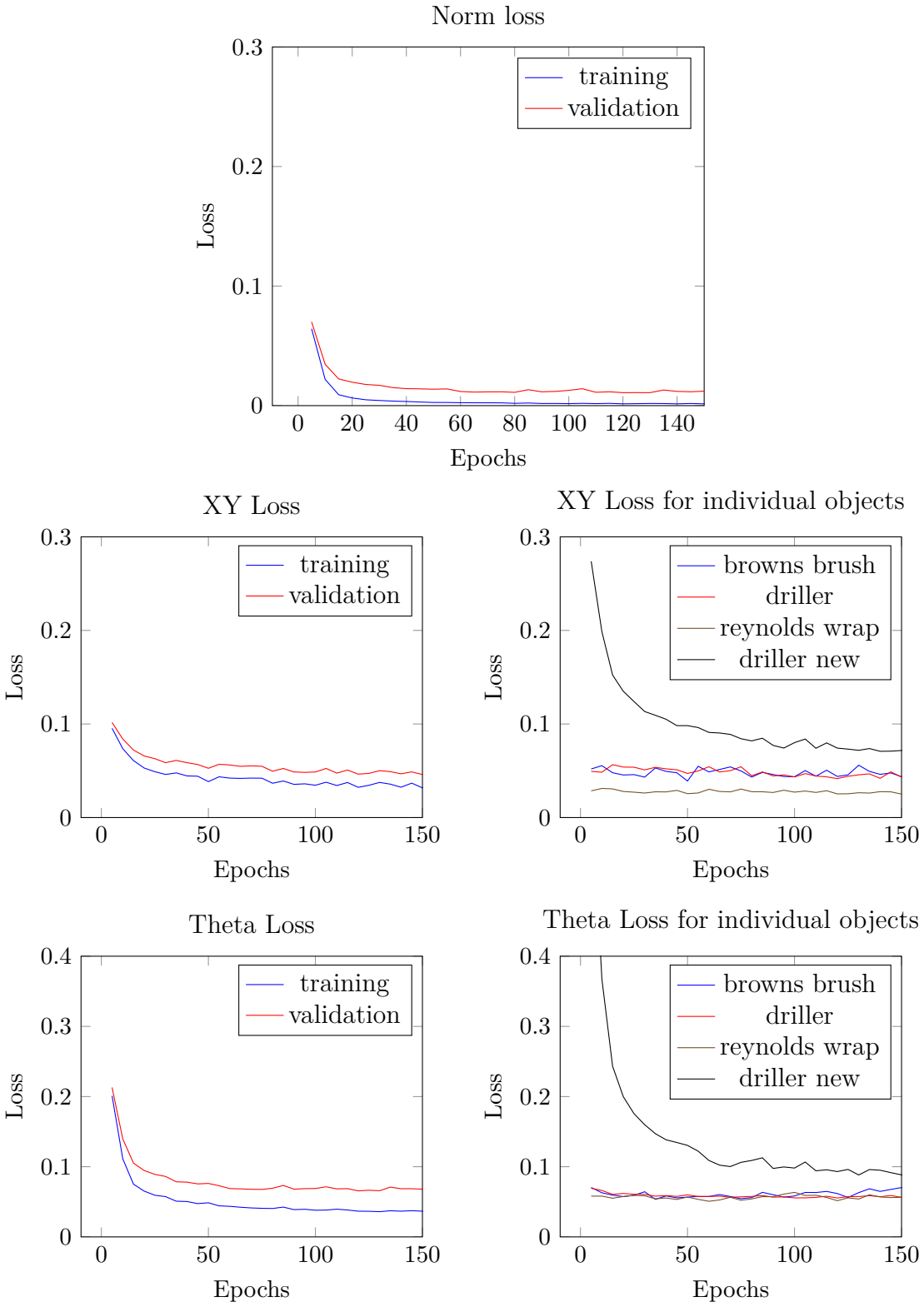

Figure 7.23: Copy weights experiment: Single-block output; training with an additional object by retaining weights from the initial run.

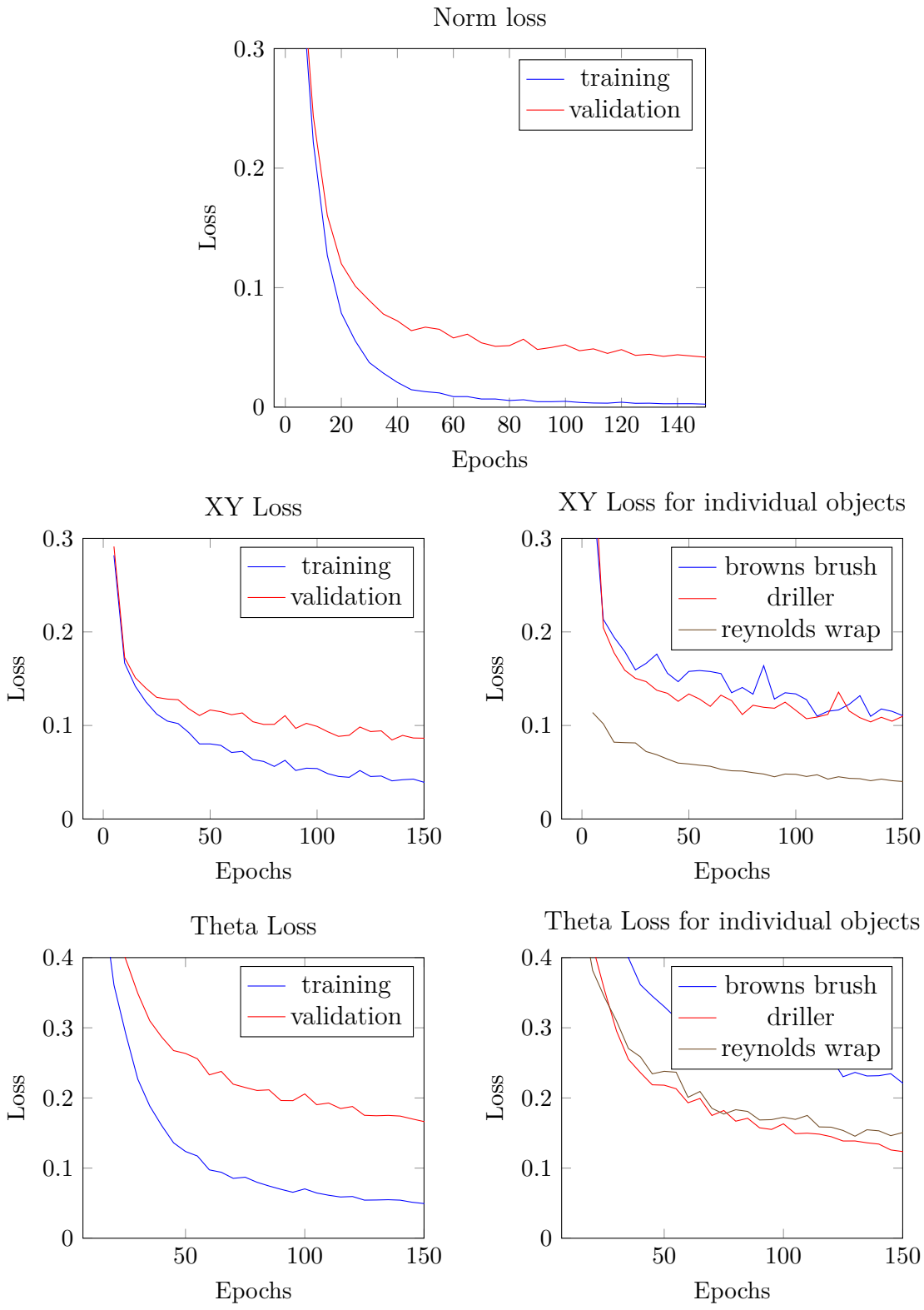

Figure 7.24: Copy weights experiment: Multi-block output; initial run with three objects.

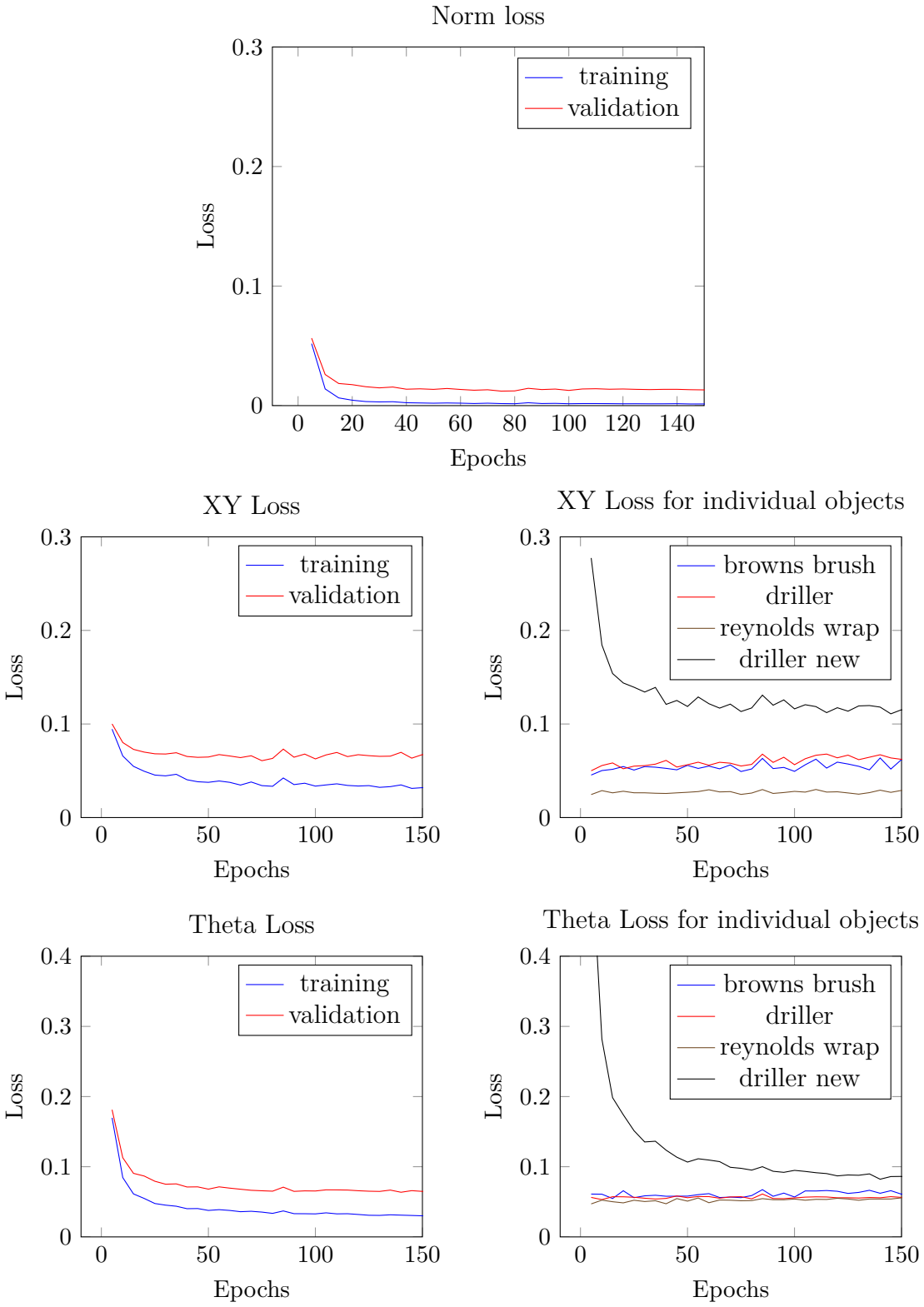

Figure 7.25: Copy weights experiment: Multi-block output; training with an additional object by retaining weights from the initial run.

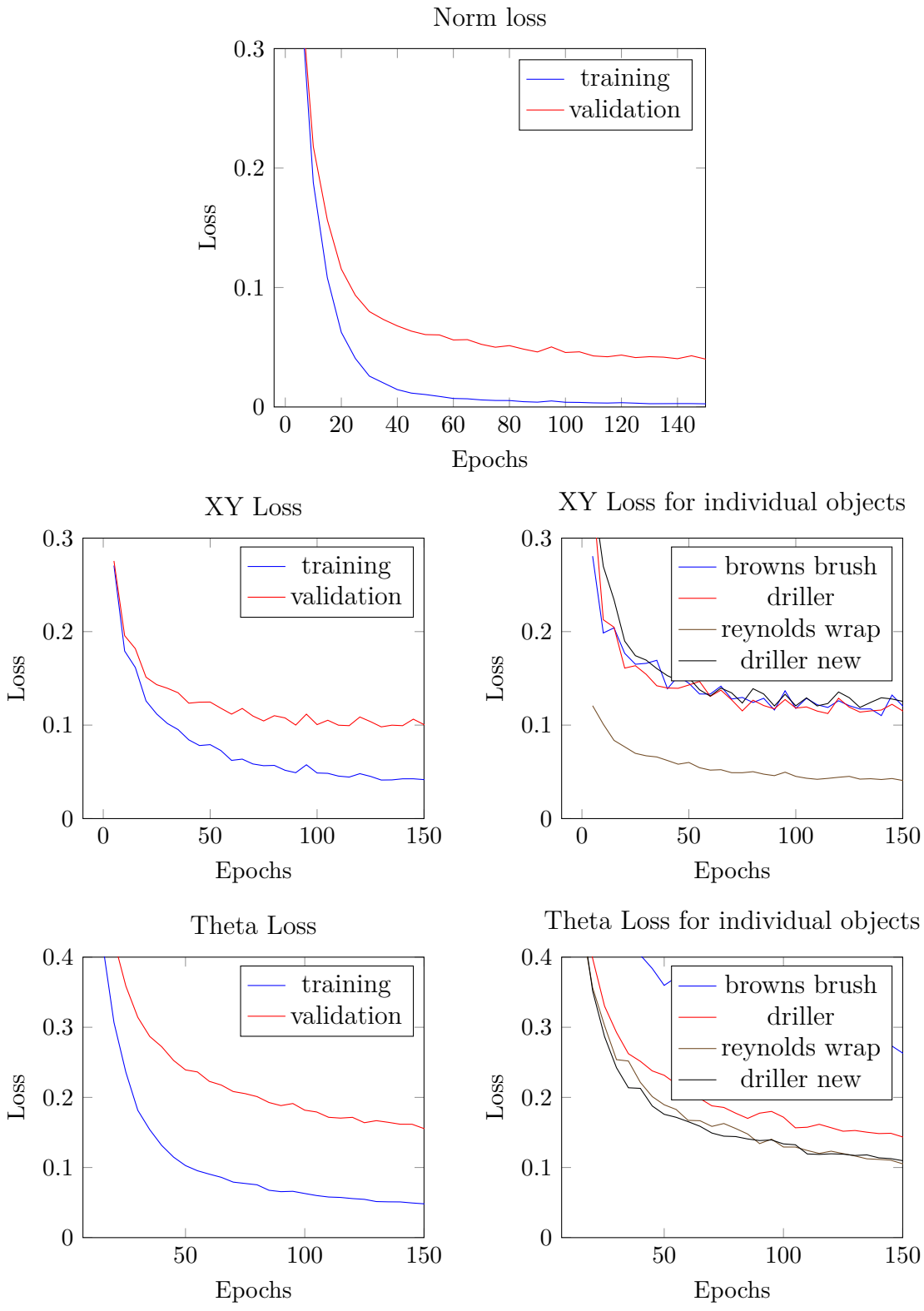

Figure 7.26: Copy weights experiment: Multi-block output; training with an additional object from scratch.

7.2 ARC Dataset

The scene understanding pipeline was evaluated with the data collected before and during Amazon Robotics Challenge, 2017. The Challenge consists of picking and stowing objects from and into the boxes. A set of 40 objects called *the training set* were given to the teams a couple of months before the competition and just 45 minutes before the competition, the teams receive a set of 32 objects called *the competition set* to be used in the pick or stow task. Half the objects in the competition set are new, and the rest are from the training set. We start with acquiring data for new objects using the pipeline discussed in Section 3.1. In the meantime, we start the training of the semantic segmentation and the pose estimation networks with the data we have and add the new data as we acquire.

7.2.1 Semantic Segmentation

The semantic segmentation network in trained in parallel over of multiple GPUs. The training starts with the network trained on the objects in the training set. The training process scans for new the new turntable captures in the filesystem, and generated synthetic images by placing images of five objects on top of manually annotated scenes. After every epoch, the structure of the network is adopted to accommodate the new objects. Thus the structure of the final classification layer of the network do not stay constant during the training and changes according to the number of objects. Figure 7.27 shows examples of the segmentation results. The input RGB image is shown in the left followed by ground truth, uninformed, and informed case of segmentation results. In the informed case, the set of objects in the bin is known thus the argmax operation on the final softmax classification layer is done only on the feature maps corresponding to those set of objects, whereas in the uninformed case, the set of objects are not known and the argmax is performed over all the feature maps. Figure 7.27 shows the mean IOU during a typical training process. The learning saturates after 5000 to 10000 images and with four GPUs, it takes 15 to 30 minutes to reach the saturation point.

7.2.2 Pose Estimation

The pose estimation network is trained on the five objects shown in Figure 7.29. Pose estimation is done only for objects that needs to be grasped in a specific poses due to its shape and other physical constraints. From the evaluations done in the Section 7.1, we observed single-block variant performed better when the objects are occluded and can learn faster when the weights are retained from the

Figure 7.27: ARC semantic segmentation results. In each row: RGB input on the left; ground truth to the left, followed by segmentation results of uninformed and informed cases.

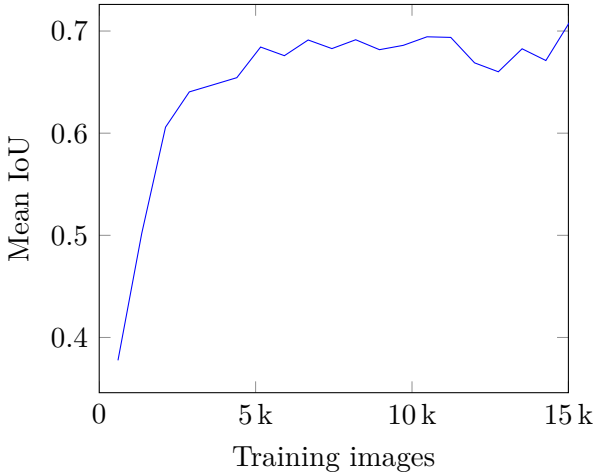

Figure 7.28: Semantic Segmentation experiments; Mean IoU during training.

Figure 7.29: ARC objects used in pose estimation evaluation. From left to right: Browns brush, epsom salts, hand weight, reynolds wrap, and utility brush.

Table 7.8: ARC objects pose estimation results.

	Translation [pixel]		Rotation [degree]	
	train	val	train	val
Browns brush	9.1	11.2	3.7	11.4
Epsom salts	8.6	9.8	3.6	6.8
Hand weight	5.5	7.3	0.9	1.1
Reynolds wrap	5.9	7.6	3.8	9.1
Utility brush	8.0	8.9	3.6	6.9

previous training process. We decided to use singe-block variant for the evaluation of pose estimation in favor of the multi-block variant. A training epoch consists of batches of 32 images. One training epoch takes approximately 40 seconds on a single GPU. When the training is resumed with the weights from a previous training run, it takes 50-75 epochs to achieve a reasonable accuracy. Thus the pose estimation network needs 30 to 50 minutes to train on a new set of objects. The final accuracy of the network is shown in Table 7.8.

7.3 Disaster Response

On a high-level, random bin picking and disaster response appears to be very different application scenarios but the scene understanding pipeline we developed can be applied to disaster response as well. To that extent, we evaluated our pipeline on Centauro dataset[1]. The Centauro project aims to build a human-robot symbiotic system with sophisticated autonomous and teleoperation capabilities. In the context of autonomous behavior, the robot should be capable of recognizing the known tools in a disaster environment and manipulate the recognized objects. To recognize the tools we perform semantic segmentation of the scene and estimate the pose of the objects to aid manipulation.

7.3.1 Semantic Segmentation

The semantic segmentation network is evaluated on the Centauro Tools dataset [2]. The dataset consists of 129 frames RGB-D frames of seven tools. Additional to the 129 manually annotated frames, we generated more training images using our

[1] `https://www.centauro-project.eu/`
[2] `https://www.centauro-project.eu/data_multimedia/tools_data`

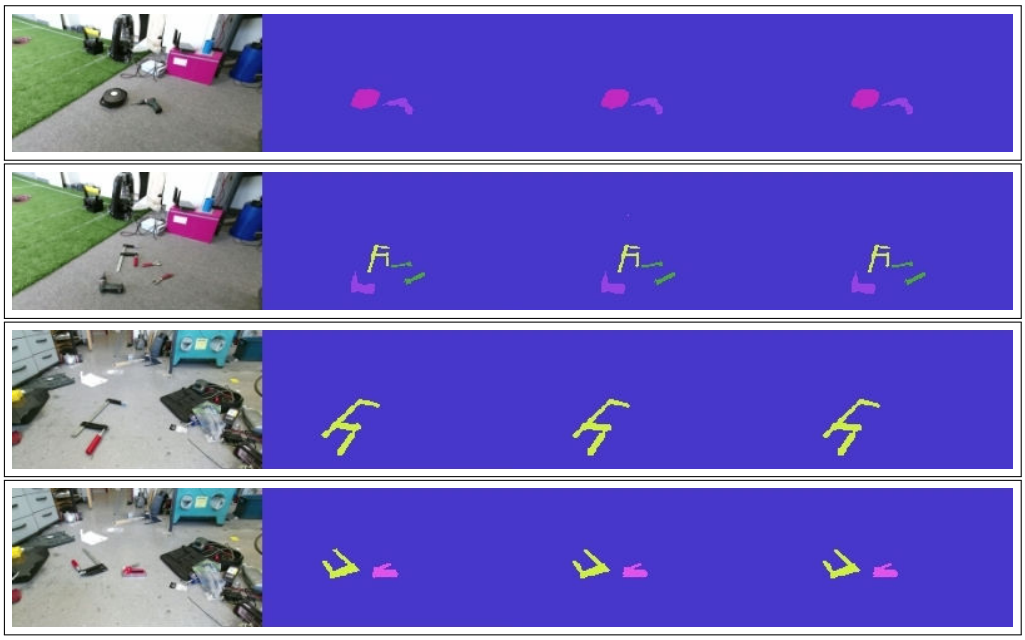

Figure 7.30: Centauro semantic segmentation results. In each row: RGB input on the left; ground truth to the left, followed by segmentation results of uninformed and informed cases.

Table 7.9: Centauro objects: semantic segmentation results.

	Box	Clamp	Driller	Door Handle	Extension Box	Stapler	Wrench
Intersection	767021	3389	7144	112	7627	1580	1751
Union	768443	3859	7510	129	7881	1725	1920
IoU	1.0	0.88	0.95	0.97	0.97	0.92	0.91

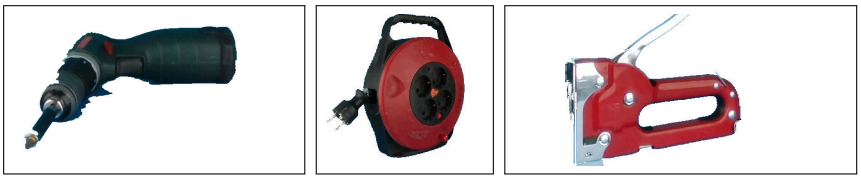

Figure 7.31: Centauro objects used in pose estimation evaluation. From left to right: Driller, extension box, and stapler.

Table 7.10: Centauro objects pose estimation results.

	Translation [pixel]		Rotation [degree]	
	train	val	train	val
Driller	8.3	9.9	7.6	10.2
Extension box	9.8	12.8	7.8	9.9
Stapler	6.3	8.1	4.3	6.4

synthetic data generation pipeline.Figure 7.30 shows the RGB image on the left, ground truth, uninformed, and informed segmentation results of example scenes during the training. The final IoU of the segmentation is shown in Table 7.9.

7.3.2 Pose Estimation

We evaluated the pose estimation network on the three objects shown in Figure 7.31. As in the ARC dataset pose estimation, we used the single-block variant. The objects that needed pose estimation are known in advance and thus eliminating the need for learning new objects. The final accuracies of the pose estimation model is shown in Table 7.10.

7.3.3 Application In Real-World Scenarios

The trained models were used in real-time on Centauro setup. A novel drill is placed on the table and the task for the robot is to perceive the environment, localize the driller and grab it to preform some predefined manipulation actions. The scene understanding pipeline developed in this thesis was put to test in the real-world environment in real-time. The results are shown in Figure 7.32. An unseen drill is placed in the table and is observed from the Kinect V2 camera mounted on the table. The predicted orientation is displayed at the bottom of the Figure 7.32. The translation is not used in this case and the center of the point cloud is used as the translation component in the visualization. The estimated

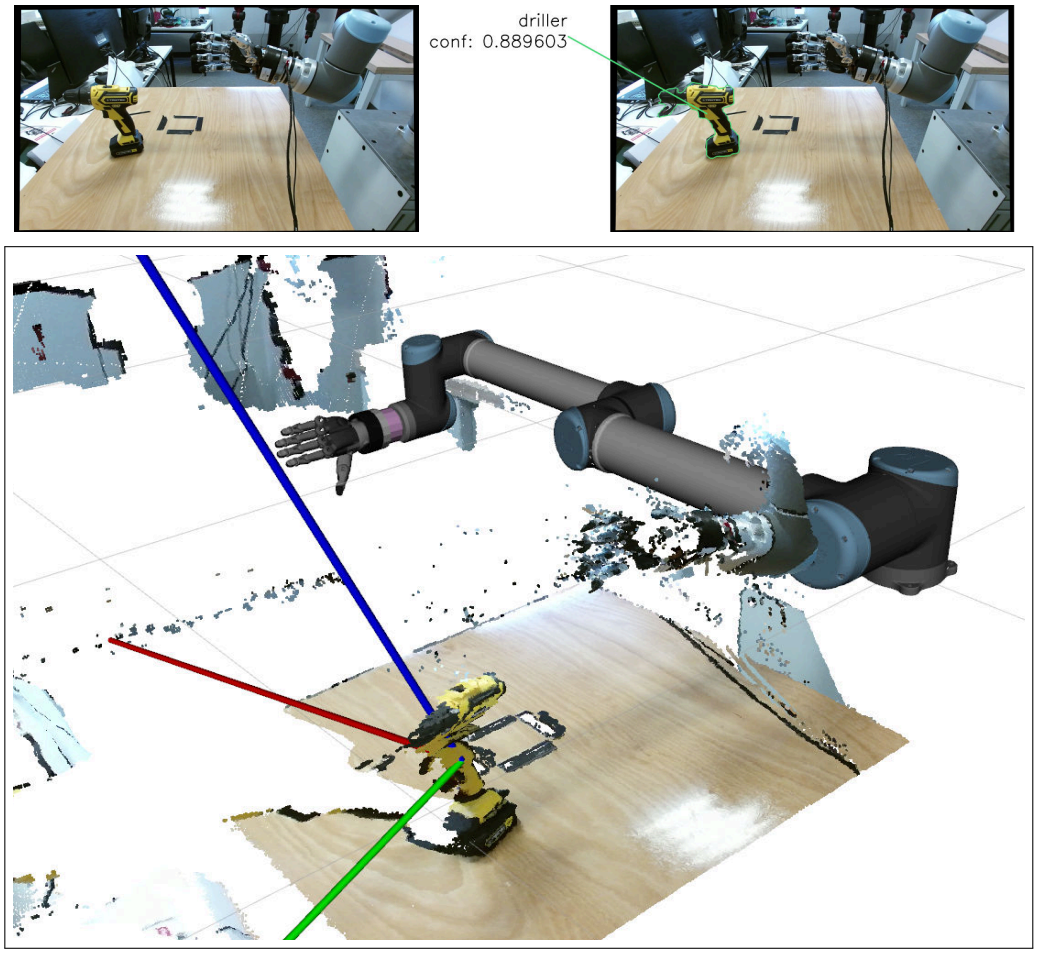

Figure 7.32: Semantic segmentation and pose estimation model evaluated on a unseen object. Top Left: Input image of the scene. Top Right: Result of semantic segmentation. Bottom: Visualization of predicted orientation.

Figure 7.33: Centauro robot grasping novel drill using the results of the pipeline proposed in this thesis.

orientation is good enough for the manipulation using an UR5 [3] arm to succeed. Figure 7.33 shows pictures captured during the demonstration where the Centauro robot is grasping a novel drill using the results proposed in this thesis. A video capture of the demonstration is present in the DVD attached with this thesis.

[3] `https://www.universal-robots.com/products/ur5-robot/`

8 Conclusion

We conclude by discussing the contributions of the thesis, limitations, and the possible future works to address the shortcomings of our work. We presented a pipeline for fast training of convolutional neural networks for scene understanding. Our pipeline consists of a high throughput data acquisition setup to capture images of new objects in short span of time, synthetic data generation module to generate training images needed for training segmentation and pose estimation networks, a semantic segmentation network, and a pose estimation networks. We evaluated different design choices of the pose estimation network and selected the best performing variant suitable for fast retraining on new images. We also proposed a method for dealing with invariance in the appearance of the objects to ease the training process of the pose estimation network. We then evaluated the pipeline in a random bin picking and a disaster response scenario and showed that our model performed well in both scenarios. Finally, we discussed the real-world scenarios where the pipeline was used to recognize an object and estimate its pose, thus aiding a robot arm in the autonomous manipulation of the object.

While this pipeline performs well in real-world scenarios, one of the early design choices of the pipeline to decouple the semantic segmentation and pose estimation networks resulted in the inability to perform joint learning. We believe both semantic segmentation and pose estimation can benefit mutually and thus needs to be trained jointly. To achieve joint learning, the architecture of the pose estimation network needs to be modified by replacing the final fully connected layers with convolutional ones, thus obtaining a fully convolutional network. Additional improvements can be done in post-processing the estimated 5D pose to obtain 6D pose, and generating 3D scenes with meshed models to directly train the pose estimation network to estimate 6D poses.

List of Figures

List of Tables

Bibliography

Aksoy, Selim and Robert M Haralick (2001). "Feature normalization and likelihood-based similarity measures for image retrieval". In: *Pattern recognition letters* 22.5, pp. 563–582.

Badami, Ishrat, Jörg Stückler, and Sven Behnke (2013). "Depth-enhanced hough forests for object-class detection and continuous pose estimation". In: *Workshop on semantic perception, mapping and exploration (SPME)*.

Besl, Paul J, Neil D McKay, et al. (1992). "A method for registration of 3-d shapes". In: *IEEE transactions on pattern analysis and machine intelligence* 14.2, pp. 239–256.

Boykov, Yuri Y and M-P Jolly (2001). "Interactive graph cuts for optimal boundary & region segmentation of objects in nd images". In: *Computer vision, 2001. iccv 2001. proceedings. eighth IEEE international conference on*. Vol. 1. IEEE, pp. 105–112.

Boykov, Yuri and Gareth Funka-Lea (2006). "Graph cuts and efficient nd image segmentation". In: *International journal of computer vision* 70.2, pp. 109–131.

Chen, Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille (2016). "Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *Arxiv preprint arxiv:1606.00915*.

Chen, Tianqi, Ian Goodfellow, and Jonathon Shlens (2015). "Net2net: accelerating learning via knowledge transfer". In: *Arxiv preprint arxiv:1511.05641*.

Choi, Changhyun and Henrik I Christensen (2016). "RGB-D object pose estimation in unstructured environments". In: *Robotics and autonomous systems* 75, pp. 595–613.

Cordts, Marius, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele (2016). "The cityscapes dataset for semantic urban scene understanding". In: *Proceedings of the IEEE conference on computer vision and pattern recognition, (CVPR)*, pp. 3213–3223.

Couprie, Camille, Clément Farabet, Laurent Najman, and Yann LeCun (2013). "Indoor semantic segmentation using depth information". In: *Arxiv preprint arxiv:1301.3572*.

Everingham, Mark, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman (2010). "The pascal visual object classes (voc) challenge". In: *International journal of computer vision* 88.2, pp. 303–338.

Farabet, Clement, Camille Couprie, Laurent Najman, and Yann LeCun (2013). "Learning hierarchical features for scene labeling". In: *IEEE transactions on pattern analysis and machine intelligence* 35.8, pp. 1915–1929.

Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik (2014). "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition, (CVPR)*, pp. 580–587.

Gould, Stephen, Richard Fulton, and Daphne Koller (2009). "Decomposing a scene into geometric and semantically consistent regions". In: *Computer vision, 2009 IEEE 12th international conference on*. IEEE, pp. 1–8.

Granger, Sébastien and Xavier Pennec (2002). "Multi-scale em-icp: a fast and robust approach for surface registration". In: *European conference on computer vision*. Springer, pp. 418–432.

Grangier, David, Léon Bottou, and Ronan Collobert (2009). "Deep convolutional networks for scene parsing". In: *Icml 2009 deep learning workshop*. Vol. 3.

Gupta, Saurabh, Ross Girshick, Pablo Arbeláez, and Jitendra Malik (2014). "Learning rich features from rgb-d images for object detection and segmentation". In: *European conference on computer vision*. Springer, pp. 345–360.

Haehnel, Dirk, Sebastian Thrun, and Wolfram Burgard (2003). "An extension of the icp algorithm for modeling nonrigid objects with mobile robots". In: *Ijcai*. Vol. 3, pp. 915–920.

Hara, Kota and Rama Chellappa (2014). "Growing regression forests by classification: applications to object pose estimation". In: *European conference on computer vision*. Springer, pp. 552–567.

He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick (2017). "Mask r-cnn". In: *Arxiv preprint arxiv:1703.06870*.

Hernandez, Carlos, Mukunda Bharatheesha, Wilson Ko, Hans Gaiser, Jethro Tan, Kanter van Deurzen, Maarten de Vries, Bas Van Mil, Jeff van Egmond, Ruben Burger, et al. (2016). "Team delft's robot winner of the amazon picking challenge 2016". In: *Arxiv preprint arxiv:1610.05514*.

Husain, Farzad, Hannes Schulz, Babette Dellen, Carme Torras, and Sven Behnke (2017). "Combining semantic and geometric features for object class segmentation of indoor scenes". In: *IEEE robotics and automation letters* 2.1, pp. 49–55.

Johnson, Justin, Andrej Karpathy, and Li Fei-Fei (2016). "Densecap: fully convolutional localization networks for dense captioning". In: *Proceedings of the IEEE conference on computer vision and pattern recognition, (CVPR)*, pp. 4565–4574.

Juszczak, P, D Tax, and Robert PW Duin (2002). "Feature scaling in support vector data description". In: *Proc. asci*. Citeseer, pp. 95–102.

Kendall, Alex, Matthew Grimes, and Roberto Cipolla (2015). "Posenet: a convolutional network for real-time 6-dof camera relocalization". In: *Proceedings of the IEEE international conference on computer vision*, pp. 2938–2946.

Koo, Seongyong, Grzegorz Ficht, Germán Martın Garcıa, Dmytro Pavlichenko, Martin Raak, and Sven Behnke (2017). "Robolink feeder: reconfigurable bin-picking and feeding with a lightweight cable-driven manipulator". In:

Kouskouridas, Rigas, Alykhan Tejani, Andreas Doumanoglou, Danhang Tang, and Tae-Kyun Kim (2016). "Latent-class hough forests for 6 dof object pose estimation". In: *Arxiv preprint arxiv:1602.01464*.

Krull, Alexander, Eric Brachmann, Frank Michel, Michael Ying Yang, Stefan Gumhold, and Carsten Rother (2015). "Learning analysis-by-synthesis for 6d pose estimation in rgb-d images". In: *The IEEE international conference on computer vision (iccv)*.

Kumar, M Pawan and Daphne Koller (2010). "Efficiently selecting regions for scene understanding". In: *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*. IEEE, pp. 3217–3224.

Kümmerle, Rainer, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard (2011). "G 2 o: a general framework for graph optimization". In: *Robotics and automation (icra), 2011 IEEE international conference on*. IEEE, pp. 3607–3613.

Lempitsky, Victor, Andrea Vedaldi, and Andrew Zisserman (2011). "Pylon model for semantic segmentation". In: *Advances in neural information processing systems*, pp. 1485–1493.

Li, Zhizhong and Derek Hoiem (2016). "Learning without forgetting". In: *European conference on computer vision*. Springer, pp. 614–629.

Lin, G., A. Milan, C. Shen, and I. Reid (2017). "RefineNet: Multi-path refinement networks for high-resolution semantic segmentation". In: *Cvpr*.

Long, Jonathan, Evan Shelhamer, and Trevor Darrell (2015). "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition, (CVPR)*, pp. 3431–3440.

Mellado, Nicolas, Dror Aiger, and Niloy J Mitra (2014). "Super 4pcs fast global pointcloud registration via smart indexing". In: *Computer graphics forum*. Vol. 33. 5. Wiley Online Library, pp. 205–215.

Milan, A. et al. (2017). "Semantic segmentation from limited training data". In: *Arxiv:1709.xxxxx [cs]*. URL: `https://arxiv.org/abs/1709.07665`.

Milan, A, T Pham, K Vijay, D Morrison, AW Tow, L Liu, J Erskine, R Grinover, A Gurman, T Hunn, et al. (2017). "Semantic segmentation from limited training data". In: *Arxiv preprint arxiv:1709.07665*.

Müller, Andreas C and Sven Behnke (2013). "Learning a loopy model for semantic segmentation exactly". In: *Arxiv preprint arxiv:1309.4061*.

– (2014). "Learning depth-sensitive conditional random fields for semantic segmentation of rgb-d images". In: *Robotics and automation (ICRA), 2014 IEEE international conference on*. IEEE, pp. 6232–6237.

Olson, Edwin (2011). "Apriltag: a robust and flexible visual fiducial system". In: *Robotics and automation (icra), 2011 IEEE international conference on*. IEEE, pp. 3400–3407.

Pohlen, Tobias, Alexander Hermans, Markus Mathias, and Bastian Leibe (2016). "Full-resolution residual networks for semantic segmentation in street scenes". In: *Arxiv preprint arxiv:1611.08323*.

Prankl, Johann, Aitor Aldoma, Alexander Svejda, and Markus Vincze (2015). "Rgb-d object modelling for object recognition and tracking". In: *Intelligent robots and systems (iros), 2015 IEEE/RSJ international conference on*. IEEE, pp. 96–103.

Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun (2015). "Faster r-cnn: towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*, pp. 91–99.

Rother, Carsten, Vladimir Kolmogorov, and Andrew Blake (2004). "Grabcut: interactive foreground extraction using iterated graph cuts". In: *Acm transactions on graphics (tog)*. Vol. 23. 3. ACM, pp. 309–314.

Russell, Chris, Pushmeet Kohli, Philip HS Torr, et al. (2009). "Associative hierarchical crfs for object class image segmentation". In: *Computer vision, 2009 IEEE 12th international conference on*. IEEE, pp. 739–746.

Schulz, Hannes and Sven Behnke (2012). "Learning object-class segmentation with convolutional neural networks". In: *Proceedings of the european symposium on artificial neural networks (esann)*. Bruges, Belgium.

Schwarz, Max, Christian Lenz, Germán Martín García, Seongyong Koo, Arul Selvam Periyasamy, Michael Schreiber, and Sven Behnke (2018). "Fast object learning and dual-arm coordination for cluttered stowing, picking, and packing". In: *Submitted to IEEE international conference on robotics and automation (ICRA)*.

Schwarz, Max, Milan, Christian Lenz, Aura Munoz, Arul Selvam Periyasamy, Michael Schreiber, Sebastian Schüller, and Sven Behnke (2017). "Nimbro picking: versatile part handling for warehouse automation". In: *IEEE international conference on robotics and automation (ICRA)*.

Schwarz, Max, Hannes Schulz, and Sven Behnke (2015). "Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features". In: *Robotics and automation (ICRA), 2015 IEEE international conference on*. IEEE, pp. 1329–1335.

Sermanet, Pierre, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun (2013). "Overfeat: integrated recognition, localization and detection using convolutional networks". In: *Arxiv preprint arxiv:1312.6229*.

Shi, Jianbo and Jitendra Malik (2000). "Normalized cuts and image segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 22.8, pp. 888–905.

Song, Shuran, Samuel P Lichtenberg, and Jianxiong Xiao (2015). "Sun rgb-d: a rgb-d scene understanding benchmark suite". In: *Proceedings of the IEEE conference on computer vision and pattern recognition, (CVPR)*, pp. 567–576.

Yosinski, Jason, Jeff Clune, Yoshua Bengio, and Hod Lipson (2014). "How transferable are features in deep neural networks?" In: *Advances in neural information processing systems*, pp. 3320–3328.

Youn, Eunseog and Myong K Jeong (2009). "Class dependent feature scaling method using naive bayes classifier for text datamining". In: *Pattern recognition letters* 30.5, pp. 477–485.

Yuille, Alan and Daniel Kersten (2006). "Vision as bayesian inference: analysis by synthesis?" In: *Trends in cognitive sciences* 10.7, pp. 301–308.

Zeng, Andy, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker Jr, Alberto Rodriguez, and Jianxiong Xiao (2016). "Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge". In: *Arxiv preprint arxiv:1609.09475*.

Zhu, Menglong, Konstantinos G Derpanis, Yinfei Yang, Samarth Brahmbhatt, Mabel Zhang, Cody Phillips, Matthieu Lecce, and Kostas Daniilidis (2014). "Single image 3d object detection and pose estimation for grasping". In: *Robotics and automation (ICRA), 2014 IEEE international conference on*. IEEE, pp. 3936–3943.