university of groningen / faculty of science and engineering

University of Groningen

DenseTransformer: Direct 6D OPE using self-attention on dense representations

MASTER'S THESIS
Artificial Intelligence

Author: **Nachiket Desai**

Primary supervisor: **Prof. Lambert Schomaker**
Secondary supervisor: **Asst. Prof. Hamidreza Kasaei**

November 17, 2022

Abstract

This paper proposes a framework for single shot object pose estimation which leverages the power of transformers using joint RGB and point cloud features. Our key study is the use of transformers on a joint embedding that is produced using a bidirectional encoder-decoder network. This way we study the application of Transformers and self-attention on intermediary features produced by an independent network. Also, our approach uses point-cloud networks (PCNs) to extract geometric information and hence the model has lower complexity. Furthermore, it opens the path for future research in using transformers(which are currently outperforming previous mechanisms in a variety of image processing tasks) on unified representations learnt from different networks.

In our model architecture, we first use the aforementioned encoder-decoder pair to create a joint representation, which also utilizes a mapping of 2D-3D features using KNN. The learnt representation is then fed to a transformer module to infer the spatial relevance of features in the joint embedding. This is followed by a set of simple convolutional modules to estimate class and pose. We evaluated our model on the LineMod and YCB datasets using the average distance metric (ADD/ADD(s)). The results show that the performance is competitive with existing state of the art.

Acknowledgments

This thesis would not come to fruition without the constant guidance and help of my professors and colleagues. I would like to expressly thank Prof. Hamidreza Kasaei for sharing his immense knowledge of existing work and methods in the field of computer vision, for helping me formulate the idea into milestones, and providing constant technical and emotional support. I also want to thank Prof. Lambert Schomaker for providing effective and practical ideas and feedback that helped formulate the framework and analysis. This project would not be completed without their support.
I also want to thank the Center of Information Technology at the University of Groningen for providing me storage and computational capacity through the Peregrine cluster, without which efficient training and analysis of the proposed idea would not have been possible. An additional thanks to the support staff at both the Center of Information Technology and Student Administration at the University of Groningen for prompt assistance and support.
Lastly, a nod of gratitude to my fellow students and colleagues for sharing with me their thoughts on my ideas and work.

Contents

List of Figures

1 Introduction

Autonomous service robots heavily depend on localizing and manipulating objects in a real world scenario. With the recent advances in CNNs and processing power, and a continuously rising collection of training datasets, a lot of research is being done to create and improve service robots. An important part of this process is that of estimating the relative pose of objects from a cluttered scene. One can imagine a lot of tasks in everyday life where we require to move/collect/adjust an object in the environment. For this reason, finding objects and their properties in 3D environments is a field of research which is fast gaining momentum. Using neural networks, I intend to create a system that can recognize objects from 3D point clouds and then estimate its 6D pose. A 6D pose refers to the posture of an object defined by a translation vector and a rotation vector, and estimating it is a very important task in a multitude of problems and applications such as driving, bin picking, recognition, self localization and mapping etc.
A variety of explicit feature extraction methods, using templates to learn texture, color and geometric information, have been proposed to find the pose of an object in a scene ([1],[2],[3]). Yet, this task is plagued with problems, apart from requiring verified templates, the results still suffer from generic real world problems such as noisy data, occlusion from clutter and insufficient information regarding the objects/scenes. CNNs ([4]) on the other hand, have shown robustness to these factors. Due to their ability to learn intrinsic properties and relations given a large set of parameters and a good dataset, many researchers have tried to solve object segmentation and 6D object pose estimation using deep networks ([5],[6],[7],[8]). Additionally, depth sensitive cameras provide an easy supply of training data, so there has been improvements in the use of point clouds for OPE tasks. [9], [10] also introduce data-models to deal with point cloud data, which further prompts the use of RGB-D data. This is good as RGB based techniques often suffer from a lack of geometric perspective, seeing as hypothesis are made based on projections of RGB data. Some solutions use 3D-CNNs ([11],[12]) to directly incorporate depth data into models similar to 2D architecture, however that greatly increases complexity. Point-cloud based networks or PCNs ([13],[14]) reduce some of this strain, however both these methods suffer from lack of texture information and sparsity of datapoints.
Using variations like affine regressors ([15]), regional bounding boxes ([16]), ShapeLoss ([17]), domain transformation ([18]) etc, multiple improvements have been suggested for end-to-end model architectures for both CNNs and PCNs. [19], [20], [21] show that the use of information from both RGB(texture) and RGB-D(rotation) data to generate features helps improve accuracy.
The introduction of attention ([22]) and transformers initially thoroughly outperformed existing methods in NLP and proved to be efficient to a variety of tasks there. Not only are these networks very efficient at capturing spatial relations in input data, they are also easy to train even for very large sets of parameters. Different kinds of attention based frameworks ([23], [24], [25]) have also been applied to image data with successful results.

While the different architectures discussed above focus on different issues and use different approaches, there appears to have been a general shift towards the use of deep learning to address 6D-OPE. Furthermore, attention based approaches have performed significantly better for image data with recent improvements. Furthermore, they are mechanisms that find relation between input features irrespective of dataset. This means that transformers can be used for a variety of datasets, and also, aggregated features.
In our paper, we evaluate the performance of transformers for the problem of pose estimation. We leverage the use of features with transformers, we go one step further, and devise a model that takes both 2D and 3D data as input. These inputs are individually fed to a well performing CNN model,

however we share information after each step between the two networks, resulting in a joint embedding that should contain information regarding 2D-3D correspondences. We then use this set of points as input to a transformer, where the model will learn spatial (and if need be, temporal) relations within the data. This is then fed to separate networks that will each focus on predicting segment, position and orientation for recognized objects in the scene. Figure 1.1 shows the abstracted pipeline of our model.

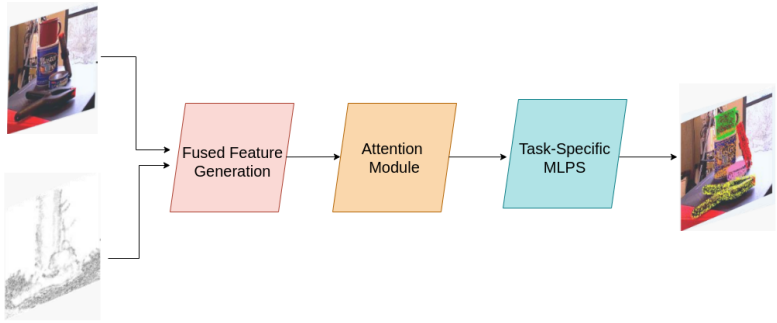


Figure 1.1: Abstract Network Architecture

We show that this can provide even better results than the individual CNN based methods and previous dense representation based methods. Furthermore, this will show that the use of dense representations of point clouds as input for Transformers results in very good inferences on the spatial relations in the data.

2 Background Literature

We will now discuss the different frameworks in the field of 6D-OPE and attention networks. Prevalent methods involve computing 2D projections of 3D images and evaluating keypoints, followed by template matching. Since initial algorithms were inefficient in presence of occlusion and change of lighting, various techniques were proposed to improve the 3D-to-2D correspondences, for example, using template matching ([1],[26]) where the image data was compared to a set of templates and the closest matching template was selected as the resulting pose. A popular alternate to this was to use sparse feature matching, where only selective features such as edges were matched ([16],[27],[28],[29]). While these were outperformed on multiple aspects by deep-CNNs over the last decade due to their capacity to learn inherent features on its own, they provide the basis for some of the CNN based architectures. The primary models which have gained traction use CNN based models and can broadly be classified as keypoint based models and holistic regression based models. We will now discuss some of these frameworks that lead to our idea.

2.1 RGB Object Pose Estimation

Holistic/direct-regression based approaches attempt to define and end-to-end framework that directly produces output from input, and the feedback in propagated all the way back. This is also an attempt to design a network that learns the data for different parameters together directly based on input and output.
BB8 [7], for example, used CNNs to find the 2D projections of the 8 corners of the bounding box detected in the image. In order to reduce errors from object symmetry, ti hey restricted the rotation of objects in the training dataset. However it was unable to efficiently estimate poses of objects with occluded parts. To deal with this problem a segmentation technique similar to older template matching techniques was applied by Crivallero et al. [30] and Hu et al. [31], where the image is segmented into multiple parts and each segment is assigned a local pose. The SSD framework [32], an improved multi-box detecting network like YOLO, Faster-RCNN, tries to directly solve the BB finding task by evaluating multiple convolutional patches, each resulting in a prediction of label. SSD6D [6], extends the SSD by incorporating the task of 6D OPE with the SSD framework, providing a single-shot RGB-based object detection and pose estimation system. An important addition made by the SSD6D paper is the additional calculated probability and loss for known viewpoints and in-plane rotations, which requires depth information. This kind of approach fails to improve after a point as some of the geometric information is only assumed. This flaw is exacerbated in the presence of occlusion. [17] also decouples the translation and rotation components, and while it is an end-to-end framework, it first find BBs using traditional methods, then decouples the network to find pose parameters. [33], [16] further reduces complexity of this method by using a pixel wise or regional voting schemes within estimated keypoints. Kleeberger et al. [34] also proposed a single-shot network with focus on object symmetry, clutter, occlusion, and multiple instances of the same object. They also use a novel loss function to deal with symmetry and similar objects. This shows that the task of 6D OPE can be solved directly by using appropriate frameworks, however all these models make hypothesis regarding the geometric information and perform relatively poorly for estimating rotation of objects. To this end we look at how point clouds based networks address the task of OPE.

2.2 RGB-D and Point Cloud Object Pose Estimation

RGB-D based networks include depth data into the input matrix, this is usually done by either adding pixel-wise depth information to complementary RGB data ([35], [36], [37], [38]), or by creating point-clouds from given information ([14], [9], [39], [40], [13], [41]).
In the first category, initial attempts simply concatenated input features directly from RGB and depth sensors respectively [35], this is then forwarded through a deep network usually ending up in a very large set of parameters. Other approaches use both these features separately, i.e, the RGB information is first used to extract features, which are then refined using depth information ([17], [42], [6], [33]). SSD6D [6] implements the SSD [32] framework and then uses PnP to incorporate depth information to give good results. PoseCNN [17] decouples the translation and rotation components, and also introduces Shape-Loss for point matching. Methods like these, first performs the task of finding BB center using RGB and estimating the distance from the camera, then use these BB to guess other pose and label parameters. Wang et al. [43] proposed Geometric-guided Direct Regression network (GDR-Net) which uses a new continuous 6D representation of the geometric space. The network first uses an efficient multi-block object segmentation network like YOLO to find ROIs, then the GDR-Net calculates dense correspondences for them using a 3D graph convolution network, and then applies Patch-PnP (a proposed variant of PnP that works with regression models) to regress 6D pose for the output.
The costly PnP algorithm is avoided by Point Cloud Networks which directly work on point-clouds generated from RGB-D data. The PointNet framework [9] in particular has been modified by a number of papers to effectively deal with point clouds. Dong et al. [39] proposed an end-to-end framework called the PPR-Net for 3D data where the technique is similar to the point-wise voting used in PVNet [33]. The Point-wise Pose Regression Network (PPR-Net) regresses a 6D pose for the corresponding object for each point in the point cloud. A clustering method is used to find segments in the point cloud, followed by an aggregation of all the pose predictions for points in that segment. [13] introduce RandLA-Net which effectively reduces the computational load by using effective sampling and local feature aggregation to learn spatial relations for each point. These and many other 3D-CNN and PCN based techniques suffer from sparse points and overhead from PnP refining. More importantly, they evaluate 2D and 3D data in different networks due to their dimensions, and at a later stage attempt to use the features from these modules to regress pose. However it would be beneficial if we could help these separate networks share information.

2.3 Joint Representation

Ku et al. [19] first introduce AVOD which uses point clouds and RGB inputs, each of which generate features which are fed to two sub-modules; a region proposal network(RPN) and a detector network. They propose an efficient method for multimodal feature fusion in the RPN that aggregates RGB and PCN features to generate proposals for objects in the scene. Xu et al. [44] propose an application agnostic framework that uses a single fusion network to predict BBs. Liang et al. [20] propose an end-to-end framework that uses a fusion of 2D and 3D features to directly detect multiple objects. MoreFusion [45] propose a framework which only fuses features at at the end of individual networks to predict BBs and pose. They use additional information from 3D-CNN features for pose regression, and split the network for individual tasks at the end.
DenseFusion [46] first proposes a generic framework to use both features together. The RGB and PCN module are allowed to process each data source individually, and uses a separate parallel module that fuses RGB and point cloud information for each pixel. While it is the framework used by

current SOTA architectures, it still does not try to learn contextual information in the neighbourhood of the pixels. Zhou et al. [47] propose a method that first extracts information from RGB data and then directly combine these features to the point cloud using a modified version of PointNet++ [10]. Several 'region-level' filters are obtained in the form of point sets, and each makes a prediction for pose. He et al. [21] use the DenseFusion architecture with RandLA-Net [13] to further reduce the size and complexity of fused features.

2.4 Transformer Networks

Since the introduction of Transformer networks [22], which use attention, and their success in various NLP tasks. While attention was already replacing other memory based networks for image processing ([23], [24], [25]), the transformer module has recently been with excellent results ([48], [49], [50], [51], [52]).
Providing pixel-wise Parmar et al. [53] applied attention to images only to select pixels in a local neighborhood. Further research showed that such local multi-head attention can perform as well as CNNs ([54], [55], [56]). Hu et al. [25] add focus on the relation between channels by adding a channel attention module that studies the relationships between channels. Separate 'SE'-blocks then explicitly model channel inter-dependency. Sparse-Transformers [57] use global self attention by creating scalable regional approximations on the transformer matrix. DANet [58] propose a dual attention module method that integrate local features with global dependencies, making similar features related to each other irrespective of distance. These networks still use CNN modules to extract features before applying attention. DETR [59] uses RGB feature maps and transformers with sparse sampling for multi-object detection with a much faster convergence rate. [60] refute the need for any convolutional modules while using transformer modules, by simply representing the image as a series of smaller patches. Since transformers are known to perform well with sentences. They show that this is true and present significant results for smaller patch sizes. [61] propose the use of spatial attention in 6D OPE with RGB images and got successful results on various datasets. [62] propose the use of color and geometry information alongside attention modules and achieve competitive results. They utilize a schema similar to AVOD [19], however this can be improved upon by incorporating more recent fusion techniques.
The rising use and success of attention networks for image processing DNNs prompts the use of transformers with the fused features architecture discussed above. To this end we will design a network that extracts features based on the framework suggested by DenseFusion and FFB6D ([46],[21]), and then test these features with an attention network to capture contextual information, before regressing pose parameters.

3 Methods

The goal is to create an end-to-end framework that outputs object class and pose for the recognized objects in the scene. Given an RGB image and it's corresponding point cloud, we want to output label and 6D pose, i.e, translation matrix $T \in \mathbb{R}^3$ and rotation matrix $R \in SO(3)$ w.r.t camera/POV. We wish to leverage both image and point cloud information for complete texture and rotation data, use the features extracted by such a network to test its compatibility with transformer networks, and we want the different modules to learn from each other directly without external refining or transformations.
An overview of the model is presented in Fig 3.1. Image and point-cloud inputs are fed into individual neural networks. The upper layers of encoder-decoder pair represent the CNN, while the lower represent the PCN. However instead of directly passing output of a layer to the next, it is first shared and transformed by a fusion layer(the set of layers in between). At the end of the networks, the resulting fused features are passed to a transformer module to incorporate spatial attention to get relative information on a local and global scale. The resulting representations which are output by the Transformer are used as inputs to a set of MLPs, each focusing on one task. Since the features learnt at the end of the transformer module are expected to contain information relevant to multiple tasks, this is a multi-task learning network that will also be using different losses for some of the tasks. We will now discuss the details of each section of our architecture.

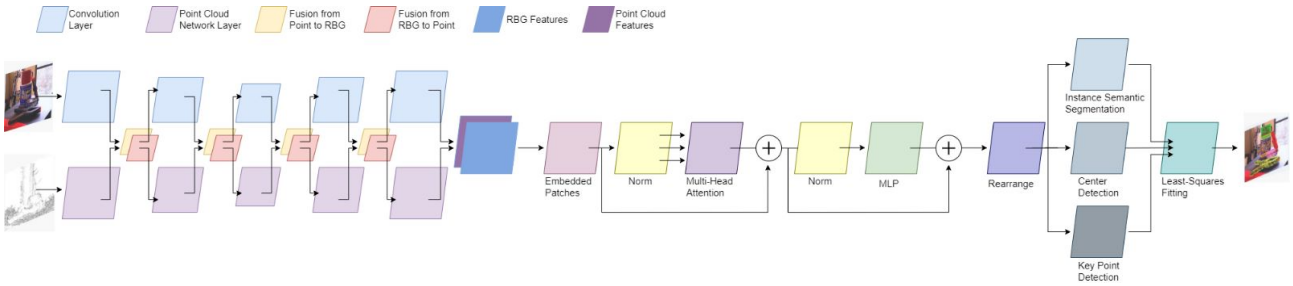


Figure 3.1: Detailed Network Architecture

3.1 Preparing Joint Representation

The network attempts to create a jointly learnt representation ([63], [21]) as opposed to a mere concatenation of individually learnt representations ([64], [65], [33]). The task again is to create a fused-features representation using both image and point cloud data. However instead of trusting the aggregated features at the end of an individual network, we will let the networks interact with each other as they progress. As proposed by [21], we will add an additional layer whose sole purpose is to collate information from one network and prepare it for the other, so as to share information between CNN and PCN at each step of convolution. The input image(3x640x480) and point-cloud(9x12800) are fed into individual CNN and PCN networks respectively, after each step of the respective process, the set of resulting CNN features and PCN features are first transformed by the corresponding nodes in the fusion layer.
In the fusion layer, we want to map the color information from the image to the point cloud, and similarly, geometric information from point cloud to RGB. Instead of evaluating a global feature for both sets of features individually, or using sparse regional features, we will follow the pixel-wise information gathering methods discussed in [8], where only the pixels in the region of the point cloud will be considered. Since the input data is aligned geometrically, we can use the point cloud to map

the corresponding pixels and vice versa [21]. We can use this information to collect relevant information for each point/pixel. This will also avoid noise from pooling over areas where the image is mostly background between two different objects. We will need two different pipelines to deal with the different kind of data for Pixel-to-Point and Point-to-Pixel information mapping.
During Pixel-to-Point mapping, we want to incorporate texture data from pixels to corresponding points. We use the camera intrinsic matrix, to add perspective adding depth for each pixel to get an XYZ map aligned to the RGB channels. Since the point cloud is generated from RGBD data in similar fashion, so further steps are needed to align the map. Then a KNN search is used to find k nearest corresponding points between the point cloud and XYZ map [21]. The CNN data from the RGB channels is then aggregated using max pooling, following [10]. This aggregated feature is then added to the point cloud data. An MLP is used to adjust the dimensions back to the required dimension of the next layer of PCN 1. During Point-to-pixel mapping, we follow the same procedure, however this time, feature data is aggregated for k nearest points in point cloud corresponding to XYZ mapping for each pixel in CNN feature. The result is passed through an MLP to prepare it for the complementary network 2. This way additional geometric information is shared with the CNN layers.

$$F_{r2p} = MLP(max_{i=1}^{k} F_{r_i}) \tag{1}$$

$$F_{p2r} = MLP(max_{i=1}^{k} F_{p_i}) \tag{2}$$

After the information from the *other* network is prepared, a concatenation operation is used, followed by another small MLP to fit the shape of next layer(3, 4).

$$F_{fused\_pcn} = MLP(concat(F_{p2r} + F_{r2p})) \tag{3}$$

$$F_{fused\_cnn} = MLP(concat(F_{r2p} + F_{p2r})) \tag{4}$$

After three encoding and decoding steps in this fashion, the final output is concatenated based on the mapping used during fusing. The resulting features should contain information on both texture and geometry. We will use this set of *enhanced* features as input for a transformer module, so we do not need more layers in encoder-decoder pair.

3.2 Transformer Module

We want to test the performance of the fused features with transformer modules, and let the entire network learn together. Attention should also help refine features from noise due to occlusion of shared pixels. We leverage the results found by [60], where an image is treated as a set of patches with positional encoding. It is then treated like words tokens in a sentence and fed to a standard transformer encoder. [60] does this by splitting the image into *mxn* patches and gives each patch a positional encoding. While a point cloud lies in a different geometric space, the fused parameters, and the transformers inherent ability to learn relations between positions irrespective of *distance* can

be leveraged to use the same architecture for point clouds.
While attention based methods have previously been effective with image tasks relating segmentation and captioning, Transformer based methods have recently attained state of the art results. We believe this is in part due the nature of self-attention as compared to attention. While attention is based on an encoder-decoder mechanism that learns relation between an output and input sequence, self attention focuses more on learning representations based on the query itself. This is particularly useful for images and point clouds in context to pose estimation, and even more so with a keypoint based approach like ours. This is because of two reasons, the first being that the orientation and texture of neighbouring pixels play an important role in both pose estimation and overall image processing. Secondly, the task requires to learn a dependable representation, thereby making a task specific decoder part of traditional attention unnecessary. The improved performance of Transformers is mainly due to the 'multi-headed self attention' that is applies. To understand why self-attention is relevant to image processing, let us first understand key, query and value as formulated by [22].
Traditionally the terms key, query, and value (Q,K,V) in attention refer to the query-response kind of applications that they was initially introduced for. Then in translation tasks, while the terms are not explicitly used, the implementation of storing information regarding *neighbours* usually involved an encoder-decoder pair. Essentially the encoder creates a key for a set of values, and when the decoder receives a query, it outputs the corresponding set of values. Whereas with *self-attention* ([22]) in Transformers, all three, key, query and value come from the same place, which is simply the previous layer of the encoder. In this manner, each position in a Transformer layer attends to every other position of the previous layer. Additionally, Transformers implement *multi-head attention* (5) where attention is applied to h linear projections of the *input* containing Q,K,V in parallel. The resulting outputs are then concatenated and passed forward. This allows the model to jointly learn from different representation subspace at different points [22].

$$MultiHead(Q,K,V) = MLP(Concat(head_1,...,head_h)) \tag{5}$$

where $head_i$ refers to the attention output of the i_{th} projection.
Furthermore the paper introduces Scaled Dot-Product Attention which deals with larger dimensions by scaling the dot-products by a factor of the dimension (6). This is also compatible with our model which has high dimensional dense features.

$$Attn = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{6}$$

3.3 Pose Parameter Regression

Now that we have is features that contain texture and geometric information, and local and global spatial awareness. We will use these features with simple FFNs and squeeze dimensions to our output dimensions. While some models ([7]) directly regress object labels, centers and poses using a single MLP, research has shown that regression of rotation parameters is more effective if handled separately ([66],[67]). Additionally, we want to design the network so that the design represents the task. So we will use 3 separate sets of FFNs to find label, centers and keypoints respectively.
As discussed before, we implement different losses for different tasks. However up to the transformer

module, our network is learning a multi-task representation, in this case for segmentation and keypoint reduction. However the representation learnt by the transformer module is then used as input to different parallel MLPs, each of which is task specific, so as to increase efficiency for each task.
For the segmentation task, we use Focal Loss, a weighted loss function that increases weight adaptation when encountering cases with lower prediction confidence (7).

$$Focal\ Loss = \alpha(-(1-p)\gamma log(p)) \tag{7}$$

For the task of regressing center and keypoints(two separate MLPs in model), we use L1 Loss or mean absolute error(MAE)(8). The reason for this is, that while segmentation/labelling, the regression is on a set of categorical points, whereas for center and keypoints regression, the output space is continuous.

$$L1\ Loss = \frac{1}{N}(\sum_{n=1}^{N} |x_n - x_n^*|) \tag{8}$$

As the L1-loss for keypoints will involve a set of points instead of a single point. So for a set of N object instances, each containing a set of K keypoints, the L1-Loss equation will be as shown in (9), where * denotes ground truth values. To estimate relative position from object center, the object coordinates are instead replaced by offset from the corresponding predicted center.

$$L1_{kpts} = \frac{1}{N}(\sum_{n=1}^{N}\sum_{k=1}^{K} |x_n^k - x_n^{k*}|) \tag{9}$$

The net loss is a weighted loss as shown in (10), where λ_1, λ_2, λ_3 are weight parameters.

$$Net\ Loss = \lambda_1 L_{class} + \lambda_2 L_{ctr} + \lambda_3 L_{kpts} \tag{10}$$

Instead of directly regressing pitch, yaw and roll, we instead opt to use the keypoint voting module introduced in PVN3D [63]. The two sets of MLP finding center and keypoint estimates will fetch us an estimated center and set of keypoints. We will then use this information to compare with ground truth center and keypoints, in order to estimate pose. Not only is this more compatible with point clouds, it is also more efficient than bounding boxes as all output points lie on the object body. This method is more coherent for objects with disproportionate dimensions, and also provides a more precise estimate of object boundaries. Given a center and set of keypoints on the ground truth model, and an estimated center and set of keypoints in the output parameters, we calculate the pose parameters. This is calculated by minimizing the least squared fitting loss as shown in (11) [21].

$$L_{lsf} = \sum_{i=1}^{N} ||p_i^* - (R \cdot p_i + T)||^2 \tag{11}$$

Here R and T denote the rotational and translational estimates respectively. p_i is the estimated keypoints corresponding to ground truth p_i^*.

4 Experimental Setup

We evaluate the performance of our model(s) on two benchmark datasets, the LineMod and YCB Video datasets. We use a fixed model of CNN and PCN across all models to keep comparisons accurate. We do not focus on testing different CNNs or PCNs, the focus of these experiments is to ascertain if transformer modules enhance the learnt representations and improve performance on the 6D pose estimation. To this effect, we will test the hypothesis using different datasets and small variants in our basic model.

4.1 Dataset

We evaluate our models on two benchmark datasets.

4.1.1 LineMOD

The LineMod dataset is widely used for various image processing algorithms including those dealing with 6D pose estimation on objects. The original dataset consists of 18 object models and over 15000 RGBD images annotated with ground truth 6D poses. The images contain objects in a cluttered scene from a variety of viewpoints and in different lighting. However a cleaner more widely used version of the dataset contains the same images with robust ground truths for only 13 of these object classes. We use this version of the dataset which is publicly available at https://bop.felk.cvut.cz/datasets/. Additionally, the raster triangle repository is used to create synthetic training data in a range of new backgrounds and object poses. Masks are added over object boundaries to prepare labels for both networks. Figure 4.1 shows samples of RGB and corresponding label images from the dataset. However the resulting dataset only contained label masks for one object class per image, for this reason, we also decided to test out model on the YCB-Video Dataset for multi-object detection.

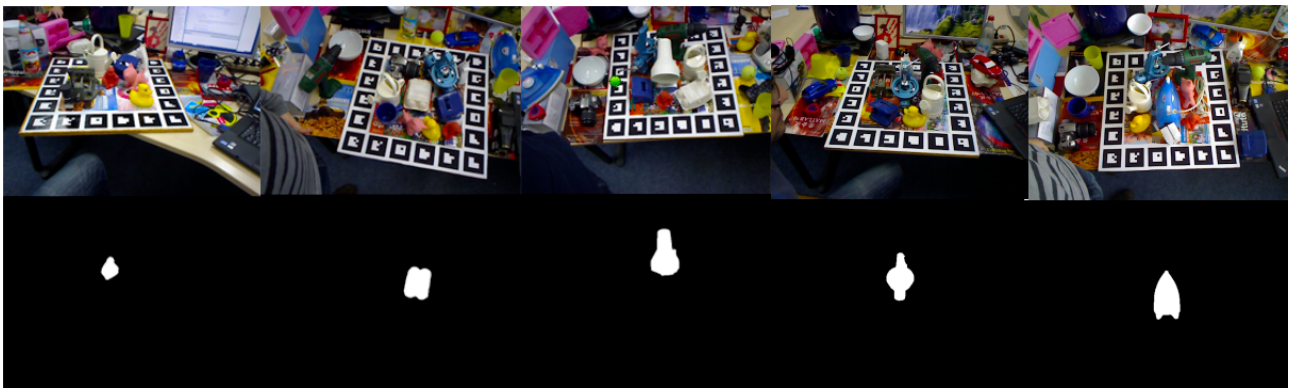


Figure 4.1: LineMOD: Samples of RGB images with corresponding label masks. (L-R: ape, eggbox, lamp, benchvise, iron)

4.2 YCB-Video

In order to test the model's performance for multi-object pose estimation, we use the YCB-Video dataset. It is a commonly used dataset in the field of 6D object pose estimation, having information regarding 21 object classes. The dataset consists of 92 video sequences, each of a random subset of

object classes, saved as a set of RGBD images. The objects in the images are annotated with 6D pose parameters and semantic segmentation masks. Again the different images are include different settings of lighting, occlusion and perspectives. Figure 4.2 shows a sample from the YCB-Video dataset, showing RGB image, depth image and class label mask for the sample.

Figure 4.2: YCB-Video: Sample RGB image with corresponding depth image and label mask. (L-R: large_clamp, gelatin_box, tuna_fish_can, wood_block, cracker_box)

While the datasets contains RGBD images and ground truths, we require point clouds for our model architecture. This is done by projecting the points using depth information and camera intrinsic matrix, following which, a threshold is applied to remove background points from the point cloud.

The data processing, and model creation and training were handled using Python3.7. The model architecture and training pipeline are implemented in PyTorch.1.10 and Nvidia apex support. Additionally, the Peregrine cluster provided by University of Groningen was used to train the models.

4.3 Model Variants and parameters

As discussed before, our model consists of three distinct segments:

1. Fusion module: consisting of a CNN and a PCN architecture and above-mentioned fusion layers
2. Transformer module: consisting of part of the Transformer architecure introduced in [68], or a variant.
3. Task specific MLPs: consisting of three sets of MLPs, respectively dealing with estimating segment, center and keypoint per object.

We were particularly impressed by the framework introduced by DenseFusion [46] which allows us to use our choice of CNN and PCN to extract features. This architecture was further improved on by FFB6D [21], we borrow this fusion architecture, albeit with lesser layers of convolution, as we will be achieving even better spatial knowledge from the transformer module. The objective of this module is to create a representation that contains both texture and geometric information without compromising detail through projections. We want to use RandLA-Net [13] as PCN for its reduced complexity and parameters, alongside ResNet(encoder)-PSPNet(decoder) for RGB input, as these networks have performed well on most well-known dataset and has publicly available pre-trained models. This pair has shown to work well with our dataset in previous papers ([63], [21]) and we will use this configuration for our tests. To reduce complexity and redundancy, the number of layers between fusion have been reduced by computing two convolutions before merging for the peripheral layers.

We will then use the obtained fused features with a self-attention network [60] to learn spatial relations. The points are given a positional embedding and used as input for the Transformer. We only leverage the encoder part of the network as the goal is to create an embedding that has learnt representations useful for multiple tasks. This representation will then be used as input to the task specific MLPs, which are simply 1D FFNs.
Each set of task specific MLP is a simple set of 4 1D convolutional layers that regress to the required parameters. The losses implemented for each task are as discussed in Section 3. The object labels are learnt from the semantic segmentation MLPs. The predicted label, along with the predicted center for each object found are used to distinguish between different objects. These point-wise offsets of each point from center are learnt by the center prediction MLPs. The points then vote for to select keypoints using a MeanShift [69] clustering algorithm. In the keypoint voting module specifically, for each object segment, we follow the point voting mechanism introduced in [63]. The ground truth keypoint selection is carried out using the SIFT-FPS as introduced in [21]. The algorithm leverages the additional texture information which has been added to the point clouds. As the name suggests, a SIFT algorithm is applied to find points with distinctive texture components. These points are projected to 3D space using depth information. Then amongst these points, a furthest point sampling(FPS) algorithm is implemented to pick a subset of points that span across the object model. The number of keypoints used in our models is 8. The predicted keypoints are compared to the ground truth using the least squares fitting algorithm [63]. Then the distance between corresponding keypoints is used to calculate the rotation parameters using the 3D rotation group notation ($SO(n)$).
For all models and datasets, the input image dimensions for the CNN architecture are 480x640x3 (height, width, channels), and the PCN input dimensions are 12800x9 (no. of points, dimension per point). The 12800 points are sampled randomly using the depth image, the position, color and normal data is stored having 3 dimensions each. During the fusion stage the network uses max-pooling between neighbors to aggregate data for point-to-pixel and vice versa. For either operation, 16 neighbours are used to calculate the aggregated information for the corresponding pixel/point.
The model was trained for 15 epochs, while 10 was enough for the LineMOD dataset, the YCB dataset model was retrained for 5 more epochs. This is reasonable as each sample alone provides feedback collected from multiple object. Twice every epoch, the model is evaluated with test data to study the prediction accuracy over time. Additionally a small batch size of 3 was used for every configuration. The model uses dynamic learning gradient with ADAM optimizer with the initial learning rate being set at $1e^{-5}$ with the minimum threshold being $1e^{-5}$ and the maximum threshold is set at $1e^{-3}$. The weight parameters are all randomly initialized and no pretrained models are used. They are updated using the net loss, which is a weighted sum of losses from the respective ends of MLPs. The segmentation module uses Focal Loss (7) with the balance parameter α being decided based on the balance of the dataset. Since we use synthetic data to balance out the number of training samples, this is set to 1 for all classes. The focus parameter γ is set to 2 in order to reduce weight adjustments for examples where the prediction confidence is high. The center and keypoint regression modules are optimized using $L1$ loss, however since they are both regressing for a set of points, we balance the effect on weight by setting $\lambda_1, \lambda_2, \lambda_3$ in (10) as $2, 1, 1$ respectively. Additionally, while a full Transformer encoder would be more effective, we believe that a miniature model with lesser blocks would suffice to verify the effectiveness of the module. Furthermore this reduces the complexity of the model by a lot as the self-attention mechanism as implemented in [22] is computationally expensive. With this in mind, we reduce the *size* of the Transformer by only using two attention-heads and two layers of depth. The input dimensions for the Transformer module are $64x12800$ and we find that increasing the depth of the Transformer quickly increases model size.

To study the effect of the Transformer module better, we train an additional model that uses the concatenation of representations learnt by each prior module as input for the task specific MLPs. That is to say, the fused representation, which is the input for the Transformer, is used as additional input for the final layer (resupply model). We will compare the results and behaviour of the networks in the next section.

4.4 Evaluation Metrics

To evaluate our performance on the datasets, we use the average distance metrics, ADD for non-symmetric objects and ADD-S for symmetric objects. The ADD metric calculates the average distance between ground truth and predicted values for sets of points. This could be object centers or a set of keypoint sets. The ADD metric is denoted as shown in (12),

$$ADD = \frac{1}{m}(\sum_{v \in O} ||(Rv+T) - (R^*v_2 + T^*)||) \tag{12}$$

here, v represents a vertex $\in$ object O. R,T are the predicted rotation and translation vectors, and R^*, T^* are the ground truth vectors.
The ADD-s metric is shown in (13), with the same denotions, and v_1, v_2 representing two different points from the same object model.

$$ADD = \frac{1}{m}(\sum_{v \in O} \min_{v_2 \in O} ||(Rv_1+T) - (R^*v_2 + T^*)||) \tag{13}$$

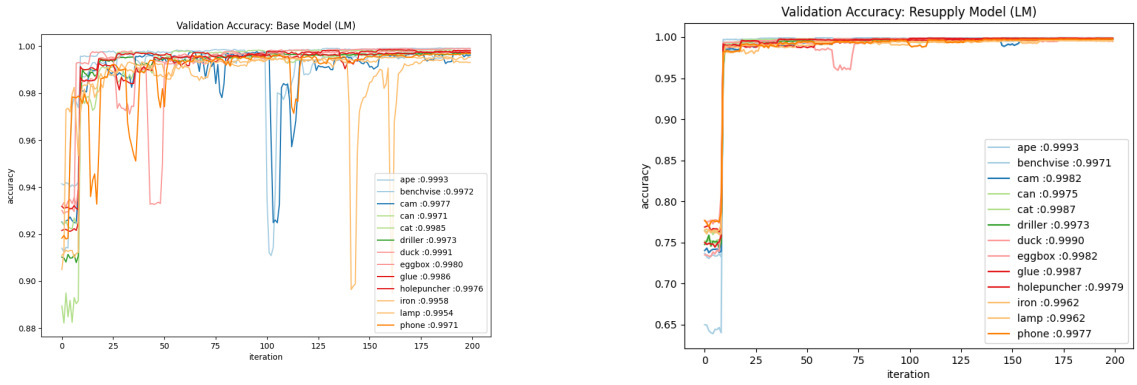
This is a commonly used pair of evaluation metrics and we compare our results with previous noteworthy models (Fig. 5.5, Fig. 5.6).

5 Results

In this section, we report and discuss the evaluated results of our models on the LineMOD and YCB-Video datasets. We present the training accuracy and loss, and test accuracy for different models from data saved during training and evaluation. We also compare our quantitative results, arrived at by using ADD/ADD-s metrics with the results of previous architecture, including the state of the art algorithms for object 6D pose estimation [21].
Additionally, we will project the predicted keypoints on the test image, so as to perform a qualitative analysis of the results obtained by our models.

5.1 Training and Validation

We will first study the training results for the LineMOD dataset for both the base model and Resupply model. Fig. 5.1 shows the prediction accuracy over time for each of the models. The model is tested twice or more every epoch using a test set. We see that for this dataset, where there is only one labelled object per test image, the model has learnt considerably in just a few epochs for both models. Due to the small batch-size, some classes like *lamp* and *cam* have some drops in prediction accuracy where the model needs to adjust more significantly even at later epochs. However this is not as prevalent for the resupply model, implying that the fused representation helped reduce confusion during training. This trend is also seen in the training accuracy and loss observed during the training process (Fig. 5.2). With the YCB dataset, we observe relatively lower testing accuracy of 93% during the training



(a) Base Model

(b) Resupply Model

Figure 5.1: Class-wise testing accuracy over time

phase. However to test the model through different hyper-parameters, only a portion of the training data is used. We can see the prediction accuracy of the model over time in Fig. 5.3. We observe that the resupplied representations from the fusion module do not cause much improvement this time, may be this is because it does not help significantly when dealing with multiple ROIs at the same time. We also observe that the model has not really stabilized and while it has reached decent performance levels, it still has a lot of room for training and adjustment. This can be confirmed in Fig. 5.4 where we can see that the model is still adjusting at every iteration, we can also see that the loss is adjusting at a higher rate, showing that the model has the capacity to do even better.

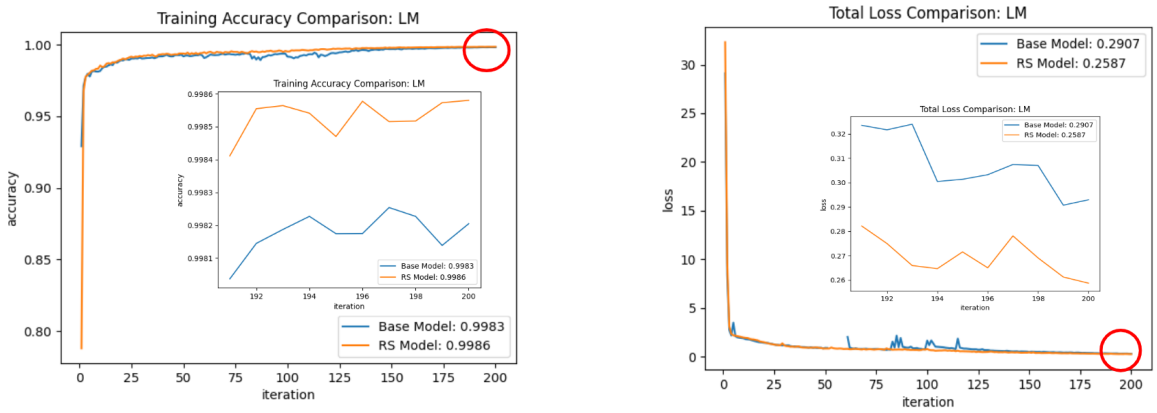
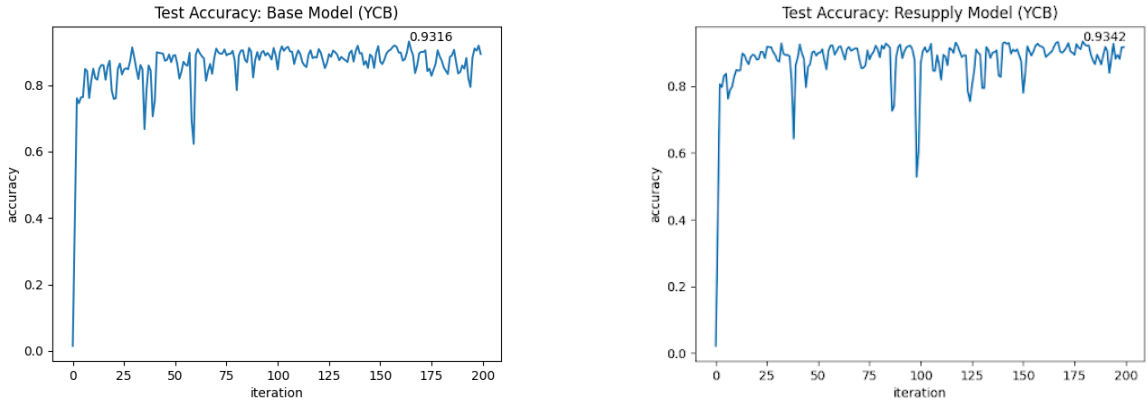


Figure 5.2: Mean Training Accuracy and Total Loss over time



(a) Base Model

(b) Resupply Model

Figure 5.3: Class-wise testing accuracy over time

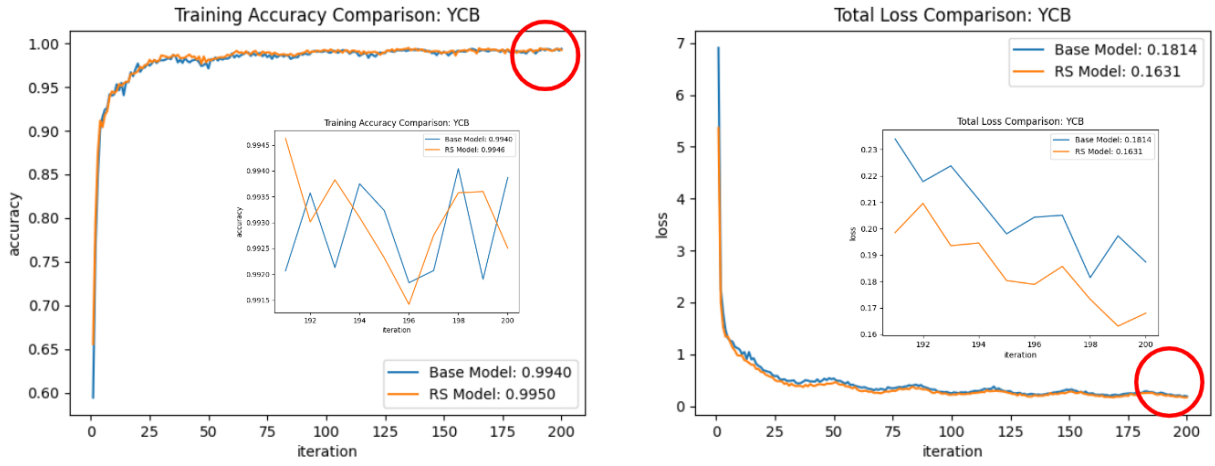


Figure 5.4: YCB: Training Accuracy and Total Loss over time.

5.2 Quantitative Analysis

Below, we can see the comparison of our model's performance to previous works in the field of 6D OPE. The tables in figures 5.5, 5.6 show the prediction accuracy as observed using the ADD/ADD(s) metric for non-symmetric and symmetric objects respectively (12, 13). We can see that for the LineMOD dataset, our model outperforms most models, and has a higher average performance across all classes than all previous works (Fig. 5.5). However for the YCB dataset (Fig. 5.6) the test results

	RGB				RGB-D						
Object	PoseCNN DeepIM	PVNet	CDPN	DPOD	Point-Fusion	Dense-Fusion	G2L-Net	PVN3D	FFB6D	Base	Resupply
ape	77	43.6	64.4	87.7	70.4	92.3	96.8	97.3	98.4	**99.92**	**99.93**
benchvise	97.5	99.9	97.8	98.5	80.7	93.2	96.1	99.7	**100**	99.71	99.71
camera	93.5	86.9	91.7	96.1	60.8	94.4	98.2	99.6	**99.9**	99.76	99.81
can	96.5	95.5	95.9	99.7	61.1	93.1	98	99.5	**99.8**	99.71	99.75
cat	82.1	79.3	83.8	94.7	79.1	96.5	99.2	99.8	**99.9**	99.84	99.86
driller	95	96.4	96.2	98.8	47.3	87	99.8	99.3	**100**	99.72	99.73
duck	77.7	52.6	66.8	86.3	63	92.3	97.7	98.2	98.4	**99.9**	**99.89**
eggbox	97.1	99.2	99.7	99.9	99.9	99.8	100	99.8	**100**	99.79	99.82
glue	99.4	95.7	99.6	96.8	99.3	100	100	100	**100**	99.85	99.86
holepuncher	52.8	82	85.8	86.9	71.8	92.1	99	99.9	99.8	99.76	99.79
iron	98.3	98.9	97.9	**100**	83.2	97	99.3	99.7	99.9	99.57	99.61
lamp	97.5	99.3	97.9	96.8	62.3	95.3	99.5	99.8	**99.9**	99.53	99.61
phone	87.7	92.4	90.8	94.7	78.8	92.8	98.9	99.5	**99.7**	**99.7**	**99.77**
MEAN	88.6	86.3	89.9	95.2	73.7	94.3	98.7	99.4	99.7	**99.75**	**99.78**

Figure 5.5: LineMOD: Class-wise test accuracy comparison

from the best saved model did not reflect the results seen during training. This is because only 30% of the training data was used for training, and 10% of the test data was used for the calculating the test scores during training. When tested on the entire test dataset, we see a drop in prediction accuracy of roughly 10%. This is not so bad and shows that the model can still improve if trained over the entire dataset and for a longer time, preferably with a larger batch-size. Also, we see that our models do not perform as well for the multi-object dataset with more classes, perhaps a deeper Transformer module can address this problem by allowing more evaluations from sub-space representations during multi-headed self-attention. We still see approximately 84% accuracy on both models, with room for improvement as the model hasn't properly converged. We see that our model shows competitive performance for single object pose estimation on the LineMOD dataset and also shows promise with multi-object pose estimation on the YCB dataset.

5.3 Qualitative Analysis

Following [63], we have used keypoints to estimate pose parameter, our model estimates a class label for each point, and when we use this to visualize the predictions of our models, we get a more accurate view than one would with bounding boxes. In Fig. 5.5 and Fig. 5.8 we see a subset of points from the object model projected onto the original image. The projected points are color coded by class, and we can observe that the resulting projections are quite accurate, implying our model successfully predicts the pose for the single object and multi-object cases in the presence of occlusion and noise. As discussed before, there is still room for training and improvement for the multi-object scenario.

5.4 Additional Points

While we see that the use of dense representations with self-attention can help directly solve pose estimation problems using deep learning, we also see that it improves the result over existing configurations for single object pose estimation. We also see that using the resupply model can improve the stability of the model and provide a small boost in performance.

Object	PVN3D	FFB6D	**Ours (Base)**	**Ours (Resupply)**
master chef can	96	96.3	91.88	**93.69**
cracker box	96.1	96.3	**86.79**	86.15
sugar box	97.4	97.6	**92.92**	91.84
tomato soup can	96.2	95.6	85.25	**92.05**
mustard bottle	97.5	97.8	**94.3**	94.25
tuna fish can	96	96.8	69.79	**88.03**
pudding box	97.1	97.1	**92.49**	89.38
gelatin box	97.7	98.1	**94.03**	91
potted meat	93.3	94.7	**73.25**	71.24
banana	96.6	97.2	**91.59**	90.87
pitcher base	97.4	97.6	**94.02**	90.89
bleach cleanser	96	96.8	92.52	**93.6**
bowl	90.2	96.3	**61.33**	58.3
mug	97.6	97.3	**95.04**	90.53
power drill	96.7	97.2	91.41	**91.43**
wood block	90.4	92.6	**72.21**	71.22
scissors	96.7	97.7	**84.21**	51.33
large marker	96.7	96.6	**81.33**	78.1
large clamp	93.6	96.8	**44.45**	43.29
extra large clamp	88.4	96	71.34	**79.9**
foam brick	96.8	97.3	**92.55**	90.75
ALL	95.5	96.6	83.46	**84.58**

Figure 5.6: YCB: Class-wise test accuracy comparison

Figure 5.7: LineMOD: Visualization of predicted points

Seeing as we only use a small set of convolutions to regress pose after the Transformer module, we can say that the Transformer learns reliable representations for the training data. Additionally, using a more gradually paced MLP at the end of transformer may help reduce some of the training noise,

Figure 5.8: YCB: Visualization of predicted points

as this would provide an additional layer of joint multi-task learning before the model splits into task specific parts.
We observe that larger objects have lesser accuracy, this is also true for objects with textureless parts. In the case of *extra large clamp* in the YCB dataset for example, we see a drastically weaker performance as compared to other classes (Fig. 5.6). This could also possibly be because a similar model exists called *large clamp* which is very similar. Since the model is supposed to account for perspective, this is probably what caused the large dip in performance.
We see from our results that the Transformer layer improves the representation learnt by the model, and helps achieve higher accuracy. This is true even for shallow Transformers. Due the the limited computational capacity, we were unable to test the model for more depth or batch size, both these factors could in our opinion, greatly improve the representations learnt.

6 Conclusion

In this work, we present a new model for direct 6D pose estimation of objects from RGBD data in cluttered scenes. The model focuses on creating highly informative features using fusion architecture coupled with a transformer. We are the first to attempt the use of fused features with multi-headed self attention for direct pose regression. Our innovation includes the use of well augmented features that learn multi-task oriented representations with help of self-attention Transformers with a small amount of self-attention blocks.

We show that the use of small transformers successfully performs the task of 6D pose estimation and also improves direct pose regression from RGBD data, as we achieve higher average results than most leading architecture. Our results show that using Transformers further improves the stability of learnt weights in case of single object pose estimation. However when the confusion increases due to multiple objects, the resupply model fails to show the same stability. It is possible that with further training, the additional information will once again help improve stability. Also, the Transformer module does not have to be as large as proposed in the initial paper, however for multi-object estimation and datasets with more class labels, it would be better to use a larger model. Also, only a few simple convolutions are required for each of the tasks, showing that the attention module does improve the quality of learnt representations. While the model learns to recognize all the object classes, it does struggle relatively more with large and shiny objects.

The current model can still be reduced in size and complexity by using smaller input images, as the size we use are slightly larger than standard. Furthermore, the layers of the fusion network can further be reduced, and replaced with attention layers instead. However we retain this module to a good extent to ensure interpolation of texture and geometric data for each point. While the current model is not usable in real-time applications, we believe that it can be achieved by reducing the layers and size of input, without sacrificing significant performance.

7 Scientific Relevance for Artificial Intelligence

As deep networks are good at learning intrinsic properties in data, they have shown to be more robust, and various architectures have been proposed to deal with data so as to be more true to the task definition. Making end-to-end frameworks more efficient by using task-oriented designs can help improve the efficiency of these systems. In this paper, we evaluate the legitimacy of transformers with the use of dense representations. From the results, we have shown that the use of Transformer modules with dense representations for point clouds can improve results even with a few layers of encoding. The representations learnt did not require a lot of additional processing to regress task parameters. Furthermore, the entire pose estimation process does not require external input. While point clouds are not an ordered set, the Transformer's inherent capacity to learn spatial relations across large distances allows it to learn representations that can be used with simple convolution to regress pose parameters. This may be useful for testing with 3D datasets with more versatile object models (human organs,), those with subtle differences between states/classes (3D-fMRI, study of bee-hives). This approach may also be useful for learning geometrical structure based graph models, such as intra-molecular geometry prediction where data has high dependence on spatial relations and geometry.
Furthermore, recent papers ([70, 51]) have attempted to reduce the complexity of self-attention and Transformer related networks. This improvement could help improve our model to more practical parameter size, tackling some of the problems we faced with the computational load.

I believe this model can also improve performance for multi-object pose estimation if trained with more layers. There is further space evaluate the possibility of using of Transformers for pose refining, and investigate more efficient ways to represent point clouds to transformers.

Bibliography

[1] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient response maps for real-time detection of textureless objects," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 5, pp. 876–888, 2011.

[2] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," in *European conference on computer vision*, pp. 536–551, Springer, 2014.

[3] J. Liebelt, C. Schmid, and K. Schertler, "independent object class detection using 3d feature maps," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, 2008.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[6] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," in *Proceedings of the IEEE international conference on computer vision*, pp. 1521–1529, 2017.

[7] M. Rad and V. Lepetit, "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3828–3836, 2017.

[8] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann, "Segmentation-driven 6d object pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3385–3394, 2019.

[9] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

[10] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[11] S. S. J. Xiao, "Sliding shapes for 3d object detection in depth images,"

[12] S. Song and J. Xiao, "Deep sliding shapes for amodal 3d object detection in rgb-d images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 808–816, 2016.

[13] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "Randla-net: Efficient semantic segmentation of large-scale point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11108–11117, 2020.

[14] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4490–4499, 2018.

[15] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proceedings of the IEEE international conference on computer vision*, pp. 2938–2946, 2015.

[16] T.-T. Do, M. Cai, T. Pham, and I. Reid, "Deep-6dpose: Recovering 6d object pose from a single rgb image," *arXiv preprint arXiv:1802.10367*, 2018.

[17] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.

[18] G. Gao, M. Lauri, Y. Wang, X. Hu, J. Zhang, and S. Frintrop, "6d object pose regression via supervised learning on point clouds," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3643–3649, IEEE, 2020.

[19] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3d proposal generation and object detection from view aggregation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–8, IEEE, 2018.

[20] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3d object detection," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 641–656, 2018.

[21] Y. He, H. Huang, H. Fan, Q. Chen, and J. Sun, "Ffb6d: A full flow bidirectional fusion network for 6d pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3003–3013, 2021.

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

[23] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, pp. 2048–2057, PMLR, 2015.

[24] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3156–3164, 2017.

[25] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

[26] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Asian conference on computer vision*, pp. 548–562, Springer, 2012.

[27] D. Li, H. Wang, Y. Yin, and X. Wang, "Deformable registration using edge-preserving scale space for adaptive image-guided radiation therapy," *Journal of applied clinical medical physics*, vol. 12, no. 4, pp. 105–123, 2011.

[28] D. G. Lowe *et al.*, "Fitting parameterized three-dimensional models to images," *IEEE transactions on pattern analysis and machine intelligence*, vol. 13, no. 5, pp. 441–450, 1991.

[29] M. Rad and V. Lepetit, "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3828–3836, 2017.

[30] A. Crivellaro, M. Rad, Y. Verdie, K. Moo Yi, P. Fua, and V. Lepetit, "A novel representation of parts for accurate 3d object detection and tracking in monocular images," in *Proceedings of the IEEE international conference on computer vision*, pp. 4391–4399, 2015.

[31] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann, "Segmentation-driven 6d object pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3385–3394, 2019.

[32] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[33] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4561–4570, 2019.

[34] K. Kleeberger and M. F. Huber, "Single shot 6d object pose estimation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6239–6245, IEEE, 2020.

[35] M. Schwarz, H. Schulz, and S. Behnke, "Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features," in *2015 IEEE international conference on robotics and automation (ICRA)*, pp. 1329–1335, IEEE, 2015.

[36] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab, "Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation," in *European conference on computer vision*, pp. 205–220, Springer, 2016.

[37] C. Choi and H. I. Christensen, "Rgb-d object pose estimation in unstructured environments," *Robotics and Autonomous Systems*, vol. 75, pp. 595–613, 2016.

[38] T. Do, T. Pham, M. Cai, and I. Reid, "Real-time monocular object instance 6d pose estimation," 2019.

[39] Z. Dong, S. Liu, T. Zhou, H. Cheng, L. Zeng, X. Yu, and H. Liu, "Ppr-net: point-wise pose regression network for instance segmentation and 6d pose estimation in bin-picking scenarios," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1773–1780, IEEE, 2019.

[40] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3d object detection in point clouds," in *proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9277–9286, 2019.

[41] K. Park, T. Patten, and M. Vincze, "Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7668–7677, 2019.

[42] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6d object pose prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 292–301, 2018.

[43] G. Wang, F. Manhardt, F. Tombari, and X. Ji, "Gdr-net: Geometry-guided direct regression network for monocular 6d object pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16611–16621, 2021.

[44] D. Xu, D. Anguelov, and A. Jain, "Pointfusion: Deep sensor fusion for 3d bounding box estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 244–253, 2018.

[45] K. Wada, E. Sucar, S. James, D. Lenton, and A. J. Davison, "Morefusion: Multi-object reasoning for 6d pose estimation from volumetric fusion," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14540–14549, 2020.

[46] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, "Densefusion: 6d object pose estimation by iterative dense fusion," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3343–3352, 2019.

[47] G. Zhou, Y. Yan, D. Wang, and Q. Chen, "A novel depth and color feature fusion framework for 6d object pose estimation," *IEEE Transactions on Multimedia*, vol. 23, pp. 1630–1639, 2020.

[48] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 11–19, 2017.

[49] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, and T.-S. Chua, "Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5659–5667, 2017.

[50] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[51] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou, "Going deeper with image transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 32–42, 2021.

[52] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International Conference on Machine Learning*, pp. 10347–10357, PMLR, 2021.

[53] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran, "Image transformer," in *International conference on machine learning*, pp. 4055–4064, PMLR, 2018.

[54] H. Hu, Z. Zhang, Z. Xie, and S. Lin, "Local relation networks for image recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3464–3473, 2019.

[55] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens, "Stand-alone self-attention in vision models," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[56] H. Zhao, J. Jia, and V. Koltun, "Exploring self-attention for image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10076–10085, 2020.

[57] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[58] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3146–3154, 2019.

[59] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *arXiv preprint arXiv:2010.04159*, 2020.

[60] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[61] S. Stevšič and O. Hilliges, "Spatial attention improves iterative 6d object pose estimation," in *2020 International Conference on 3D Vision (3DV)*, pp. 1070–1078, IEEE, 2020.

[62] H. Yuan and R. C. Veltkamp, "6d object pose estimation with color/geometry attention fusion," in *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pp. 529–535, IEEE, 2020.

[63] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun, "Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[64] S. Zakharov, I. Shugurov, and S. Ilic, "Dpod: 6d pose object detector and refiner," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1941–1950, 2019.

[65] J. Zhou, M. Hao, D. Zhang, P. Zou, and W. Zhang, "Fusion pspnet image segmentation based method for multi-focus image fusion," *IEEE Photonics Journal*, vol. 11, no. 6, pp. 1–12, 2019.

[66] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.

[67] W. Chen, X. Jia, H. J. Chang, J. Duan, L. Shen, and A. Leonardis, "Fs-net: Fast shape-based network for category-level 6d object pose estimation with decoupled rotation mechanism," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1581–1590, June 2021.

[68] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[69] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[70] D. Zhou, B. Kang, X. Jin, L. Yang, X. Lian, Z. Jiang, Q. Hou, and J. Feng, "Deepvit: Towards deeper vision transformer," *arXiv preprint arXiv:2103.11886*, 2021.

Appendices