Transformer-Enhanced DenseFusion for 6D Object Pose Estimation

Ismail Abouelseoud
s291365

Valeria Intini
s338570

Gabriele Parisini
s345149

Edoardo Ciorra
s337049

Abstract

The accurate determination of 6D object pose is critical in robotics and augmented reality. Existing literature presents various approaches, including methods based on dense feature fusion. Our methodology follows a two-stage pipeline. First, we employ YOLO-based object detection to localize target objects in RGB images, providing bounding-box priors for subsequent pose estimation. Second, we perform pose estimation using these bounding-box outputs as spatial priors. We adopt an architecture inspired by DenseFusion [6], with our primary contribution being the replacement of the standard multi-layer perceptron (MLP) with a Transformer network, leveraging self-attention to better capture global context and complex dependencies between depth and RGB descriptors. We benchmark our approach using standard metrics—Average Distance (ADD) error—and success rates at various thresholds (2cm,5cm, and 10cm). The complete source is on GitHub: https://github.com/ismailabouelseoud/6D_POSE_GROUP_30.

1. Introduction

6D object pose estimation, which determines both the 3D position and 3D orientation of objects in space, is a fundamental task in computer vision with critical applications in robotics, augmented reality, and autonomous systems. The challenge lies in accurately estimating the six degrees of freedom (three for translation and three for rotation) from RGB or RGB-D sensor data, particularly under varying lighting conditions, partial occlusion, and cluttered environments. There have been many improvements in this field.

Feature-based methods [3] Methods in this category take advantage of local features (keypoints, grey values, edges, or intersections of straight lines) extracted from the regions of interest or all pixels in the image, and then compared with the features found on a 3D model of the object to establish 2D-3D matches [8].

Template-based methods [3] These methods include a first off-line stage which build a template database from a 3D model of the object. This database includes a set of synthetic renderings, obtained by varying position and orientation, resulting in a group of patches from different points of view. The second stage is a test phase, executed on-line to establish the 6D position. So, the current image is compared with all the database patches generated in the previous step through a sliding window algorithm.

Direct prediction or learning-based methods These methods predict the 6D pose using CNNs [7], hence needing a training phase which requires large amounts of labelled data but allows CNNs to produce significant improvements for the 3D position and rotation estimation. DL based methods can be one-stage and two-stage, depending on the use of a further step to refine pose parameters through Perspective-n-Point (PnP) algorithm [8].

2. Related Work

6D Pose from RGB images RGB images carry only information about the intensity of each colour channel per pixel. Keypoint-based approaches have proven effective in this setting. These are distinctive points in an image that represent important local features or structures, such as corners, edges, or junctions, which can be reliably detected. Given a set of keypoints in a 3D model and their corresponding counterparts in an RGB image, it is possible to retrieve the 6D pose via the PnP algorithm

Pose from RGB-D images RGB-D images contain three values per pixel that encode the intensity of each colour channel, plus an additional value representing the distance from the camera. This depth information enables the extraction of geometrical features about the object in the scene, providing richer 3D structural information compared to RGB-only approaches.

RGB-D DenseFusion and pixel wise approach [6] The key idea is the fusion of information extracted from both RGB and depth channels. Segmented depth pixels are converted into a 3D point cloud using camera intrinsics, and a PointNet architecture extracts geometric features. The geometric embedding network generates dense per-point features, converting each of the P object points into geometric feature vectors. A colour embedding network (ResNet-

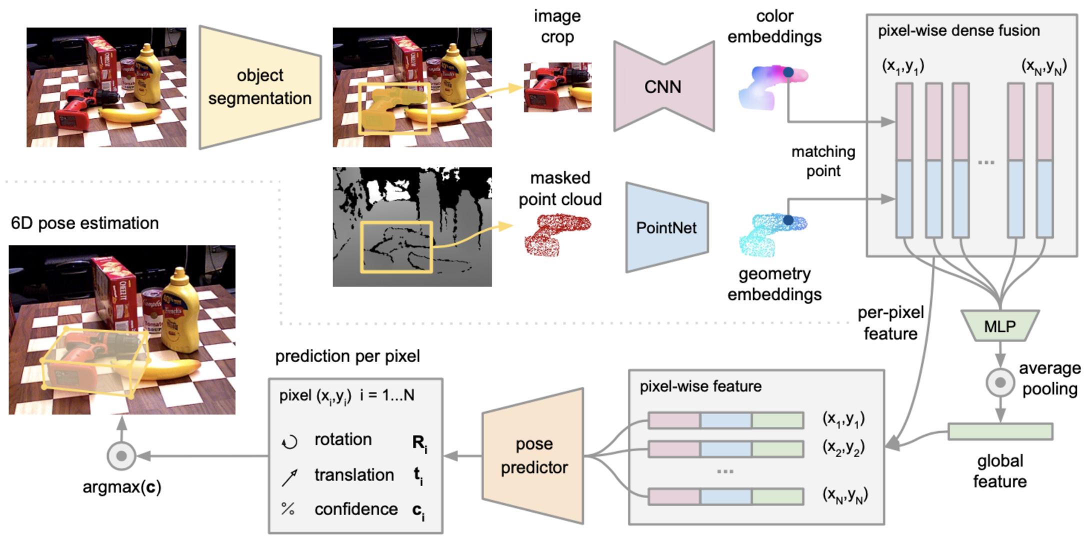

Figure 1. DenseFusion model architecture

based CNN) extracts dense per-pixel RGB features from the masked image region. The core innovation is the dense fusion mechanism, which combines these two feature streams at the point level, creating fused dense features for each 3D point. Finally, a pose regression network predicts the 6D pose from these densely fused features (Fig. 1).

3. Dataset

This project uses the LM (LineMOD [1]) dataset, which is a valuable resource in the fields of detection and pose estimation of texture-less 3D objects in moderately cluttered scenes. Some key-characteristics about it follow:

- It consists of 15 registered video sequences, each composed by over 1100 frames focusing on one target object.
- The sequences feature 15 different texture-less household objects with discriminative color, shape and size. For each of them a 3D pointcloud is provided.
- For each sequence it is provided:
  - A set of RGB frames
  - A set of depth frames
  - The 6D pose of the target object.

For this project we are using a subset of this dataset, including 13 sequences (thus, 13 objects). Thus, following standard protocols, we use the train/test split with objects 3 and 7 excluded as in DenseFusion. In order to estimate the 6D pose we exploit only data extracted from RGB images source, depth information and object pointclouds.

4. Pipeline Overview

Our approach is to implement a simplified version of DenseFusion following a five-stage pipeline:

1. A YOLOv11 detector is employed to localize target objects in RGB images, providing bounding box priors which are used to get crops of the RGB and depth images.
2. Mask R-CNN is employed on the crops to perform semantic segmentation. The produced mask is applied on the crops to make sure that only the target object is taken into account. The (masked) depth crop is projected into a 3D space using the provided camera intrinsics to recover the 3D pointcloud of the target object.
3. The RGB (masked) crop and the 3D pointcloud are passed into two distinct networks, that produce two global feature vectors encoding both color and geometric features.
4. A fusion head is used to get a single global feature vector.
5. The global feature vector is passed into a pose prediction head and a confidence estimation head to get an estimated pose along with a confidence score.

4.1. Object Detection

The model innovation lies in its single-shot detection approach: it divides the images into grids, predicts bounding boxes and class probabilities directly for each grid cell. This

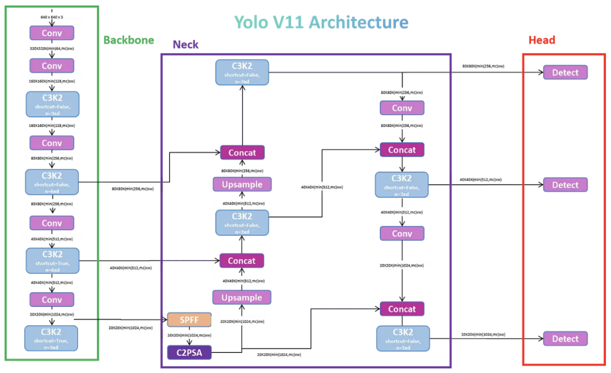

Figure 2. Yolo11 model architecture [2]

makes YOLO [2] both faster and more efficient, achieving significant speed improvements over two-stage models like Faster R-CNN. In particular YOLOv11 introduces two architectural improvements, such as the C3K2 blocks for efficient feature extraction and the C2PSA attention mechanism for enhanced focus on critical image regions. For the goal of the project we are using a pretrained version of YOLO11.

4.1.1 Model architecture

The YOLOv11s architecture (Fig. 2) we used comprises three main components:

- Backbone: A series of convolutional layers, including C3 blocks, which extract multi-scale feature maps.
- Neck: An FPN-style (Feature Pyramid Network) path that uses upsampling and concatenation to merge features from different scales, strengthening both spatial and semantic information.
- Head: Detection layers that predict bounding box coordinates and objectness scores for each anchor and class.

We chose this model for our implementation due to its simplicity, which allowed us to apply transfer learning without needing to adapt the loss function.

4.1.2 Evaluation

After fine-tuning, YOLOv11 worked well on LineMOD dataset, producing the results in Tab. 1.

Table 1. YOLOv11 Validation Metrics on the LINEMOD Dataset

Class	**Box(P)**	**Recall (R)**	**mAP@50**	**mAP@50–95**
ape	0.965	0.937	0.974	0.793
benchvise	0.998	1.000	0.995	0.904
camera	0.998	1.000	0.995	0.908
can	0.993	0.992	0.995	0.899
cat	0.963	0.849	0.963	0.781
driller	0.978	0.985	0.994	0.869
duck	0.980	0.963	0.992	0.839
eggbox	0.992	0.947	0.987	0.842
glue	0.985	0.922	0.984	0.786
holepuncher	0.998	0.991	0.995	0.865
iron	0.997	1.000	0.995	0.893
lamp	0.998	1.000	0.995	0.925
phone	0.992	0.997	0.995	0.913
All classes	0.987	0.968	0.989	0.863

4.2. Semantic Segmentation

A semantic segmentation step is performed to let the model address only for pixels belonging to the target object. The employed model is Mask R-CNN (with a ResNet50 backbone), an extension of Faster R-CNN which adds a branch to predict an object mask in parallel with the branch for object recognition. In a first stage, the Region Proposal Network (RPN) proposes candidate object bounding boxes. In the second stage, in parallel to bounding boxes refinement and classification, a binary mask for each Region of Interest (RoI) is generated using a fully convolutional network. Given the bounding box predicted by YOLO, we pick the mask with the maximum overall score, defined by a combination of original Mask R-CNN score (s), overlap ratio between the mask and the bounding box predicted by YOLO (o) and Intersection over Union (IoU) beetween the bounding boxes predicted by both YOLO and Mask R-CNN:

$$s_{\text{overall}} = 0.4 \cdot s + 0.3 \cdot o + 0.3 \cdot IoU \qquad (1)$$

4.3. RGB Feature Extraction

The RGB feature extraction module employs a convolutional neural network to encode visual appearance information from masked object regions (Fig. 3). The network follows a standard hierarchical architecture that progressively reduces spatial dimensions while increasing feature depth to capture both low-level texture patterns and high-level semantic representations.

The backbone consists of four convolutional blocks with channel dimensions expanding from 3 to 512. Each block applies 3×3 convolutions with batch normalization and ReLU activation, using stride-2 convolutions for spatial downsampling at later stages. This design achieves an 8× reduction in spatial resolution while expanding the feature representation capacity.

Global feature aggregation is performed through adaptive average pooling followed by a 1×1 convolution to produce a compact 32-dimensional embedding. The final RGB feature vector provides a discriminative representation of the object's visual appearance for subsequent multimodal fusion.

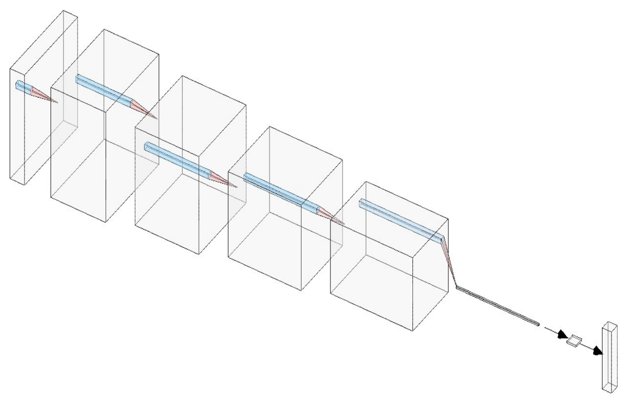

Figure 3. RGB Feature Extraction

4.4. Geometric Feature Extraction

Geometric features are extracted from 3D point clouds generated by projecting masked depth patches into world coordinates using camera intrinsic parameters. The point cloud processing follows the PointNet architecture to handle unordered 3D data, maintaining permutation invariance.

The network processes each point independently through shared multilayer perceptrons implemented as 1D convolutions. The architecture consists of three convolutional layers with progressive channel expansion (3→64→128→1024), each followed by batch normalization and ReLU activation. Point-wise features are then aggregated using global max pooling to obtain a permutation-invariant representation.

The aggregated features are processed through fully connected layers with dropout regularization to produce a 32-dimensional geometric descriptor. This descriptor encodes the 3D shape characteristics of the target object, complementing the RGB appearance features for robust pose estimation.

4.5. Feature Fusion Strategies

We investigate two approaches for combining RGB appearance and geometric features: a baseline concatenation method and a Transformer-based fusion mechanism. The fusion strategy is critical for effectively leveraging complementary information from both modalities in the pose estimation task.

4.5.1 MLP Fusion Baseline

The baseline approach concatenates the 32-dimensional RGB and geometric features, followed by multilayer perceptrons to learn cross-modal interactions. The concatenated 64-dimensional vector is processed through two fully connected layers (64→512→1024) with ReLU activations to produce the final fused representation.

Concatenation and Processing. The forward pass concatenates features from both modalities and processes them through the MLP fusion head:

$$\mathbf{F}_{\text{fused}} = \text{MLP}([\mathbf{F}_{\text{rgb}} \oplus \mathbf{F}_{\text{depth}}]) \in \mathbb{R}^{1024} \tag{2}$$

where $\oplus$ denotes the concatenation operation. This method is computationally efficient and provides a simple representation for combining the two modalities.

4.5.2 Transformer-Enhanced fusion

The MLP fusion, while simple and computationally efficient, assumes a fixed interaction pattern between RGB and geometric features. It treats all input samples in the same way, regardless of which modality is more informative with respect to the input. However, the relative importance of RGB and geometric features can vary depending on context: for instance one object's pose may be recoverable from appearance cues despite occlusion, while another may require 3D geometry due to visual ambiguity. We address this problem by replacing the original MLP fusion head with a Transformer encoder [5], which allows dynamic interactions between the two modalities via self-attention. In this way, the model can adaptively weigh features based on the specific characteristics of each input.

Features projection on a common space. Since the RGB and geometric features are learnt from two different networks, we assume that they live in different spaces. The first step consists in finding a linear projection onto a common higher-dimensional embedding space. This is achieved by a fully connected layer with an identity activation function:

$$F'_{rgb} = W_{rgb}F_{rgb} + b_{rgb} \tag{3}$$

$$F'_{depth} = W_{depth}F_{depth} + b_{depth} \tag{4}$$

Sequence preparation. The token sequence fed as input to the transformer is minimal, generated by concatenating the following three tokens:

- The learnable [CLS] token ($cls \in \mathbb{R}^{128}$) initialized with zeros, which serves as a global summary representation of the entire sequence.
- The projected RGB feature ($F'_{rgb} \in \mathbb{R}^{128}$).
- The projected geometric feature ($F'_{depth} \in \mathbb{R}^{128}$).

Since Transformers treat tokens as a set (unless otherwise instructed), a learnable positional embedding matrix ($E_{pos} \in \mathbb{R}^{3\times128}$) is added to the token sequence, leading to the following input sequence:

$$X_0 = \begin{bmatrix} cls \\ F'_{rgb} \\ F'_{depth} \end{bmatrix} + E_{pos} \tag{5}$$

Features encoding and fusion. The employed Transformer encoder is composed of 4 layers, each consisting of a multi-head self-attention block with 2 heads followed by a position-wise feedforward network. The usage of multiple heads enables the network to capture different types of relationships between tokens. For each layer ℓ, the sequence $X_{\ell-1}$ is passed to each head h, which projects it as follows:

$$Q^{(h)} = X_{\ell-1}W_Q^{(h)} \tag{6}$$

$$K^{(h)} = X_{\ell-1}W_K^{(h)} \tag{7}$$

$$V^{(h)} = X_{\ell-1}W_V^{(h)} \tag{8}$$

where the query, key and value projection matrices ($W_Q^{(h)}, W_K^{(h)}, W_V^{(h)}$) are of dimension $R^{128\times64}$. Then, the self-attention mechanism is applied as follows:

$$A^{(h)}(X_{\ell-1}) = \text{softmax}\left(\frac{Q^{(h)}K^{(h)\intercal}}{\sqrt{64}}\right)V^{(h)} \tag{9}$$

which consists in a weighted sum of values, where weights are attention scores derived from the scaled dot-product of queries and keys. The outputs of each head are then concatenated and linearly projected into a space of the same dimensionality using the (learned) matrix $W_O \in \mathbb{R}^{128\times128}$, producing the final multi-head attention output:

$$\text{MHSA}(X_{\ell-1}) = [A^{(1)}(X_{\ell-1}) \oplus A^{(2)}(X_{\ell-1})]W_O \tag{10}$$

Afterwards, a dropout layer is applied to enhance generalization, followed by a residual connection and a Layer Normalization to stabilize training:

$$Y_{(\ell)} = \text{LayerNorm}(X_{(\ell-1)} + \text{MHSA}(X_{\ell-1})) \tag{11}$$

Lastly, an MLP with 2 layers is applied to each token independently, expanding the dimensionality in the first layer and then shrinking it back:

$$\text{MLP}(Y_{(\ell)}) = \text{GELU}(Y_{(\ell)}W_1 + b_1)W_2 + b_2 \tag{12}$$

where $W_1 \in \mathbb{R}^{128\times512}$ and $W_2 \in \mathbb{R}^{512\times128}$. Again, a residual connection and Layer Normalization are applied around the MLP, returning the output sequence that will be passed to the next layer:

$$X_\ell = \text{LayerNorm}(Y_{(\ell)} + \text{MLP}(Y_\ell)) \tag{13}$$

The output sequence from the last layer is cut down to keep only the [CLS] token (which we'll mention as F_{fused} from now on), which serves as a fused global feature of RGB and geometric modalities.

4.6. Pose estimation

The fused global feature is passed through an MLP with 3 layers to predict the 6D pose of the object through a 7DoF representation (3D translation t and quaternion rotation q):

$$[t, q] = \text{ReLU}(\text{ReLU}(Y_{(\ell)}W_1 + b_1)W_2 + b_2)W_3 + b_3 \tag{14}$$

In parallel, the fused global feature is passed to another MLP with the same architecture to predict a confidence score of the related pose estimation (c). Lastly, the predicted quaternion $q = [w, x, y, z]^\intercal$ is normalized and converted into a proper rotation matrix ($R \in \mathbb{R}^{3\times3}$) by the following computation [4]:

$$R(q) = \begin{bmatrix} 1-2(y^2+z^2) & 2(xy-wz) & 2(xz+wy) \\ 2(xy+wz) & 1-2(x^2+z^2) & 2(yz-wx) \\ 2(xz-wy) & 2(yz+wx) & 1-2(x^2+y^2) \end{bmatrix} \tag{15}$$

4.7. Loss Function

Our training objective is guided by a composite loss function designed to learn 6D object pose and confidence estimation simultaneously. The total loss $\mathcal{L}_{total}$ a weighted sum of the pose loss (based on the Average Distance — ADD — metric), a confidence loss, and a regularization term that penalizes large translations ($> 2m$) , formulated as:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{ADD}}\mathcal{L}_{\text{ADD}} + \lambda_{\text{conf}}\mathcal{L}_{\text{conf}} + 0.1\mathcal{L}_{\text{reg}} \tag{16}$$

where λ_{ADD} and λ_{conf} are the weights for the pose and confidence terms, respectively. The primary pose component, $\mathcal{L}_{\text{ADD}}$, provides a strong, geometry-aware signal for accuracy:

$$\mathcal{L}_{\text{ADD}} = \frac{1}{|M|}\sum_{v\in M} ||(Rv + t) - (R_{gt}v + t_{gt})||_2 \tag{17}$$

where M is the set of 3D model vertices, while (R, t) and (R_{gt}, t_{gt}) are the predicted and ground-truth poses. The confidence loss, $\mathcal{L}_{\text{conf}}$, is implemented using Binary Cross-Entropy with Logits (BCEWithLogitsLoss), which encourages the network to produce a high score for accurate pose estimations. A minor regularization term is also added to penalize excessively large translation predictions, ensuring pose stability.

5. Experiments

5.1. Experimental Setup

For this experiment, we used Google Colab to develop our script, utilizing T4 and L4 GPUs with 16 GB and 22 GB of VRAM, respectively.

5.1.1 Implementation Details.

We implemented our method in PyTorch, exploring two distinct feature fusion strategies. For all experiments, we trained the network for 15 epochs on the LINEMOD dataset using the AdamW optimizer, with a learning rate of 1e-4 and a batch size of 12.

5.1.2 Evaluation Metrics.

To assess the performance of our pose estimation framework, we use the standard Average Distance (ADD) metric as our primary evaluation criterion. This metric is calculated as the mean pairwise distance between the 3D model vertices transformed by the predicted pose and the ground-truth pose, following the formulation for $\mathcal{L}_{ADD}$ in Equation (19).

For symmetric objects, where multiple orientations are visually identical, the standard ADD metric is insufficient. Therefore, for the two symmetric objects in the LINEMOD dataset (eggbox and glue), we use the ADD-S metric, which measures the average distance to the closest model point. It is defined as:

$$\begin{aligned} ADD\text{-}S = \frac{1}{|M|} \sum_{v_1 \in M} \min_{v_2 \in M} \|(Rv_1 + t) \\ - (R_{\text{gt}} v_2 + t_{\text{gt}})\|_2 \end{aligned} \tag{18}$$

Here, $|M|$ is the number of vertices in the model, and (R, t) is the predicted pose. A prediction is considered successful if the ADD score is below a specified threshold. We report success rates at multiple absolute distance thresholds (2 cm, 5 cm, 10 cm) and at thresholds relative to the object's diameter (5%, 10%, 20%), providing a comprehensive view of the model's accuracy.

5.2. Quantitative Results

Tab. 2 present our main experimental results, comparing the baseline MLP fusion with our Transformer-enhanced approach across different point counts, batch sizes, and optimization settings. These results indicate that the MLP slightly outperforms the Transformer—likely because the Transformer requires more extensive fine-tuning and additional training iterations, which were constrained by our limited GPU hours on Colab. An example of the training and validation loss for 6D pose in Fig. 4.

6. Discussion and Conclusion

In this paper, we present a 3D pose-estimation framework that begins with object detection via transfer learning, using the YOLOv11 model for bounding-box localization. The pose-estimation stage is inspired by DenseFusion, but whereas DenseFusion operates on pixel-wise features, our approach leverages global feature representations. Despite this architectural change, our method consistently achieves sub-decimeter accuracy. For fusing RGB information with point-cloud data, we explore two fusion strategies—a multilayer perceptron (MLP) and a transformer variant—which get very similar performance.

Limitations. The limited 6D pose accuracy of our model (approximately 6 cm ADD) constrains its suitability for real-time industrial applications. This limitation could be alleviated through model-optimization techniques—such as incorporating additional local and pixel-wise features. Furthermore, our evaluation is confined to the LineMOD dataset with minimal augmentation; expanding our experiments to include other datasets would further strengthen our conclusions. Lastly, due to computational constraints, all the trainings lasted at most 15 epochs, which is a very small number for this use case.

References

[1] 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 *Computer Vision – ACCV 2012*, pages 548–562, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. 2

[2] R. Khanam and M. Hussain. Yolov11: An overview of the key architectural enhancements, 2024. 3

[3] G. Marullo, L. Tanzi, P. Piazzolla, and E. Vezzetti. 6d object position estimation from 2d images: a literature review. *Multimedia Tools and Applications*, pages 1–39, 2022. 1

[4] K. Shoemake. Animating rotation with quaternion curves. *SIGGRAPH Comput. Graph.*, 19(3):245–254, 1985. 5

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 4

[6] 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. *CVPR*, abs/1901.04780, 2019. 1

[7] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *CVPR*, abs/1711.00199, 2017. 1

[8] Z. Zhao, G. Peng, H. Wang, H. S. Fang, C. Li, and C. Lu. Estimating 6d pose from localizing designated surface keypoints. *CVPR*, abs/1812.01387, 2018. 1

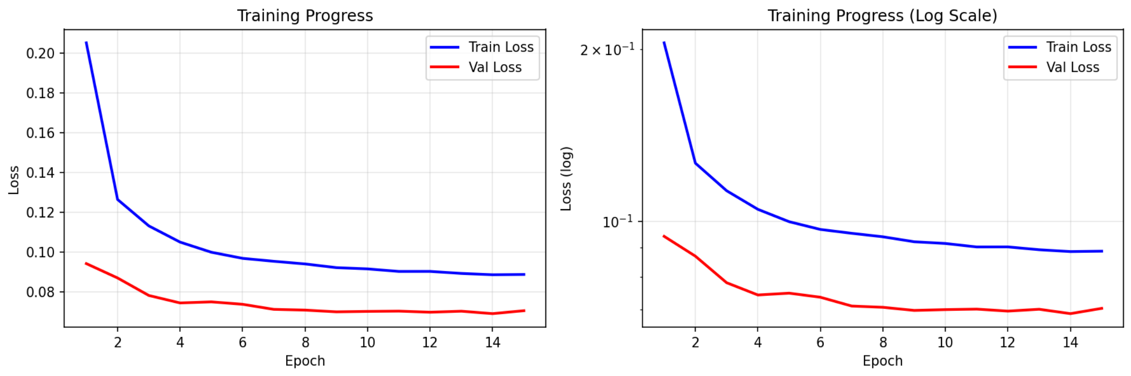

Figure 4. Training and validation loss for 6D pose estimation

(a) Transformer-enhanced results (batch=12, epochs=15)

# Points	# Patches	Mean ADD (m)	$< 2cm$	$< 5cm$	$< 10cm$	$< 5px$	$< 10px$	$< 20px$
500	512	0.064	14%	48%	71%	3%	13%	38%
500	1024	0.060	16%	56%	71%	3%	15%	44%
1000	512	0.067	12%	46%	70%	2%	11%	34%
1000	1024	0.061	13%	51%	72%	2%	13%	38%
1500	512	0.063	15%	52%	71%	3%	13%	39%
1500	1024	0.063	12%	51%	72%	2%	13%	38%

(b) MLP fusion results (batch=12, epochs=15)

# Points	# Patches	Mean ADD (m)	$< 2cm$	$< 5cm$	$< 10cm$	$< 5px$	$< 10px$	$< 20px$
500	512	0.058	21%	60%	72%	4%	18%	49%
500	1024	0.058	26%	61%	71%	5%	23%	53%
1000	512	0.062	20%	57%	71%	3%	17%	47%
1000	1024	0.060	19%	58%	72%	3%	18%	48%
1500	512	0.064	15%	55%	72%	2%	14%	42%
1500	1024	0.060	17%	58%	71%	3%	16%	47%

Table 2. Training results using MLP and transformer for 6D pose