Dual Fusion PoseNet

Mozhdeh Hajiani
Politecnico di Torino
s330007@studenti.polito.it

Abstract

This work presents a deep learning pipeline for 6D object pose estimation using RGB-D data from the LineMOD dataset. The approach employs a dual-stream structure that processes RGB and depth data separately through ResNet-based networks before integrating them with a dual fusion method. A pretrained ResNet-50 extracts RGB features, while depth features come from a ResNet-18. These representations are combined and passed through a multilayer perceptron that predicts weights. This setup allows the model to focus on the most useful data type for each input. The weighted features are then combined and directed into two separate regression heads to predict object rotation and translation. The model trains with a combined loss that includes quaternion distance and mean squared error for translation. Experiments on the LineMOD dataset show that the dual fusion approach improves pose accuracy by balancing the contributions from RGB and depth information. The method achieves strong results, reducing the ADD metric from a baseline of 110.3 mm to 33.2 mm,, demonstrating a substantial gain in estimation precision.

The source code is publicly available at https://github.com/mozhdeh-hajiani/Dual-Fusion-PoseNet.

1. Introduction

Recognizing objects and estimating their 3D poses has a wide range of applications in robotics, such as object manipulation and human-robot interaction tasks. However, this problem is challenging due to the variety of object shapes and appearances, which can be affected by lighting conditions, scene clutter, and occlusions.

Traditionally, 6D object pose estimation has been approached by matching feature points between 3D models and images. These methods rely on rich object textures, making them ineffective for texture-less objects. With the advent of depth cameras, several approaches have been proposed to recognize texture-less objects using RGB-D data. However, template-based methods suffer from performance degradation under occlusions, and methods that regress image pixels to 3D object coordinates often struggle with symmetric objects.

As industrial applications become more complex, pose estimation in environments with overlapping objects, confusion, and severe occlusion has become increasingly difficult for traditional methods. This has driven the shift toward deep learning-based pose estimation algorithms, which offer improved robustness and are emerging as a key direction in the field. Applications include augmented reality, autonomous driving, and general computer vision tasks.

In this work, we propose a deep learning-based 6D pose estimation pipeline that combines object detection using YOLO8 with pose regression. Our approach is evaluated on the LINEMOD dataset, a widely used benchmark for instance-level 6D pose estimation. We train our pose regression model using ground truth object poses and evaluate performance using the ADD .

2. Related Work

2.1. 6D Pose Estimation

Traditional pose estimation techniques have been widely implemented in a variety of fields. This approach can be divided into three main categories: template-based matching methods, feature-based approaches, and 3D descriptor-based methods [1].

(a) Matching methods based on templates involve creating a 2D image template using a 3D model of the object. This template typically represents a projected image of the object from a specific angle or perspective. Subsequently, the created template is compared with real-time images of the scene. Similarity measures are calculated between the template and different parts of the scene image for each possible position and orientation. These measures determine the most suitable location and orientation of the template within the scene, providing an estimate of the object's position and orientation in three dimensions. [2]. While these methods excel at detecting texture-less objects, they struggle with occlusions since partially visible objects yield low similarity scores with complete templates.

(b) Feature-based method refers to the process by which the feature points are extracted from an original picture to form a set and compared with the target pictureś feature point set to get the position of the object. This approach is appreciated for its simplicity and user-friendly nature. When dealing with target objects with strong texture characteristics, feature extraction methods like SIFT [3], SURF successfully work. Although these methods handle occlusions better than template-based approaches, they require sufficient object textures for reliable feature computation.

(c) Direct prediction or learning-based methods: These methods predict the 6D pose using CNNs, 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.

In recent years, the research mainly focused on Learning-based methods, which allow the training of a neural network tailored for a specific task. For this reason, most of the analyzed methods belong to this category, and they have been in turn classified into three sub-categories: Bounding box prediction and Perspective-n-Point (PnP) algorithm-based methods, Classification-based methods, and Regression-based methods. These methods can reach very high levels of precision but need many data to train the network accurately and to be able to work well in real cases. Learning-based methods can be classified into three categories [4]:

1. Bounding box prediction and PnP algorithm-based methods: These approaches use a pipeline for the 6D pose prediction composed of a CNN architecture for the object category detection and the object projected bounding box vertices prediction.

2. Classification-based methods: These approaches attempt to simplify 6D pose estimation into a one-step classification problem by discretizing the pose space. They rely on CNNs to obtain a probability distribution across the pose space, which is then correlated with 3D model information to determine the 3D position and rotation [5]. For instance, the SSD-6D [6] algorithm extends object detection algorithms for 6D pose estimation by discretizing the rotation space into discrete viewpoints and intra-plane rotations, thereby treating rotation estimation primarily as a classification problem before performing the complete pose estimation

3. Regression-based methods: These systems solve 6D pose as a regression problem and use CNNs to estimate the position. They directly regress the 6D pose parameters of the target object from the input image [5]. Usually, there is a preliminary stage of object detection to simplify the position estimation process . These systems belong to the category of one-stage methods, i.e. they design a neural network which receives an input image for training and solves the posed problem by learning the rotation and 3D translation of the object represented in it . Yu Xiang et al. [7] proposed the PoseCNN, a novel CNN that decomposes pose estimation into semantic segmentation, 3D rotation, and 3D translation tasks. PoseCNN estimates 3D translation by locating the object center and predicting the distance to the center while predicting 3D rotation using quaternion regression, demonstrating robustness in estimating poses of occluded objects. Wang et al. introduced a method called DenseFusion [8] , which estimates the 6D pose of known objects from RGB-D images. This approach employs a heterogeneous architecture to process two data sources and utilizes a novel dense fusion network to extract pixel-level dense feature embeddings for pose estimation. The method integrates an end-to-end iterative pose optimization process, enhancing accuracy while maintaining near-real-time inference speed.

2.2. Object Detection

Object detection is a computer vision task that involves identifying and locating objects within an image or video by drawing bounding boxes around them and classifying them into predefined categories.

R-CNN is a two-stage object detection model that first generates around 2,000 region proposals, then processes each region to extract features and classify them. It ranks regions by classification confidence and removes overlapping ones. While accurate, R-CNN is slow and computationally expensive. Fast R-CNN and Faster R-CNN [9] are improved versions that reduce processing time and improve accuracy.

YOLO [10] and its successors (YOLOv5, YOLOv8) have become popular for real-time object detection due to their speed and accuracy. YOLO (You Only Look Once) is a fast, single-stage object detection model designed for real-time detection. It divides an image into a grid and predicts bounding boxes and class probabilities for each grid cell. Instead of using separate networks for feature extraction and classification like R-CNN, YOLO does both in a single pass.

In 6D pose estimation pipelines, YOLO can serve as a robust front-end to localize object bounding boxes prior to pose estimation. Integrating YOLO with pose regression networks enables end-to-end pipelines that are both fast and effective.

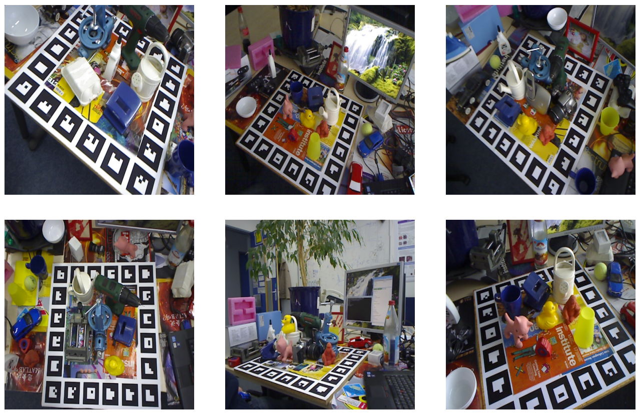
Figure 1. Example of dataset images

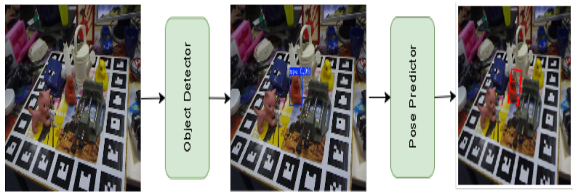

Figure 2. Baseline Pipeline. (1)The RGB image is passed through a YOLO8 object detector to generate bounding boxes. (2) The object is cropped from the RGB images based on the detected bounding box. (3) The cropped RGB representation is passed directly to a PoseNet-style network. (4) The network regresses the 6D pose, outputting a 4D quaternion for rotation and a 3D vector for translation

3. Methods

3.1. Dataset

We evaluate our proposed method on the LINEMOD dataset [11], a widely used benchmark for 6D pose estimation. It consists of RGB-D image sequences for 15 objects, with annotations for the central object's bounding box, rotation, and translation matrices. The dataset contains 18,273 test images and provides textured 3D models. we skip the third and seventh objects class.

3.2. Extract Object Patches

The first step in the pipeline is to detect objects of interest in the image. Object detection involves identifying and locating objects by drawing bounding boxes around them. Unlike image classification, which only labels objects, detection includes both classification and localization.

We use YOLOv8 as the object detector to localize objects within each frame. YOLO (You Only Look Once) frames detection as a regression problem, predicting bounding boxes and class probabilities directly from full images. A single convolutional neural network processes the entire image in one evaluation, allowing the pipeline to be optimized end-to-end for detection performance. YOLO offers several advantages over traditional methods. It is extremely fast, requiring only one network evaluation per image without a complex multi-stage pipeline. It also reasons globally about the image, reducing background errors compared to region proposal-based approaches like Fast R-CNN. Furthermore, YOLO learns generalizable object representations and performs well even when applied to new domains, outperforming methods like R-CNN.

To adapt YOLO for object detection on the LINEMOD dataset, we first fine-tuned the model using the LINEMOD training set. A custom preprocessing pipeline was implemented to convert the original LINEMOD annotations into the YOLO format. Each object class (e.g., ape, cat, driller, etc.) was processed individually. For each class, the corresponding gt.yml file was parsed to extract ground truth 2D bounding boxes, which were then converted to the YOLO format—normalized coordinates in the form (x_center, y_center, width, height) relative to the image dimensions. The images and their corresponding YOLO-format annotation files were organized into structured directories in accordance with YOLO conventions. Additionally, a data.yaml configuration file was generated, specifying class names and paths to the training and validation sets.

This preprocessing pipeline enabled us to train and evaluate YOLO models on the LINEMOD dataset by framing it as a standard object detection task. Once trained, the YOLO model was used to detect objects of interest in the images. The resulting bounding boxes were then used to crop image patches, which served as input for subsequent 6D pose estimation.

3.3. Baseline assessment

To establish a baseline for 6D pose estimation, we adopt a regression-based approach . The objective is to directly predict the full 6D pose, comprising 3D rotation and 3D translation, using only RGB images of cropped object instances from previous stage. Our implementation builds on a ResNet-50 backbone pretrained on ImageNet.

The network accepts as input an RGB crop of a detected object and outputs a pose vector: a 3D translation (x, y, z) and a 4D unit quaternion (q_x, q_y, q_z, q_w) representing the object's orientation. Internally, the ResNet-50 architecture is modified by removing its final classification layer and applying global average pooling to extract a 2048-dimensional feature vector from the last convolutional layer. This feature vector is passed through two separate fully connected heads: one head outputs a 4D quaternion that is normalized to lie on the unit hypersphere, while the other produces a 3D translation vector. This results in a complete pose vector of the form $[q_x, q_y, q_z, q_w, x, y, z]$ for each object crop.

The network is trained using a Mean Squared Error

(MSE) loss applied to both the predicted rotation and translation components. leading to the following total loss function:

$$L_{\text{pose}} = \text{MSE}(q_{\text{pred}}, q_{\text{gt}}) + \text{MSE}(t_{\text{pred}}, t_{\text{gt}})$$

where $q_{\text{pred}}, q_{\text{gt}}$ are the predicted and ground-truth quaternions, and $t_{\text{pred}}, t_{\text{gt}}$ are the predicted and ground-truth translations.

Each object instance is cropped from the input RGB images using bounding boxes predicted by a YOLOv8 detector. These image patches are resized to 224×224 pixels, conforming to the input requirements of the ResNet50 backbone. To maintain consistency with the pretrained model, the RGB values of these crops are normalized using ImageNet statistics: the mean is set to $[0.485, 0.456, 0.406]$ and the standard deviation to $[0.229, 0.224, 0.225]$. To represent object rotations, the dataset-provided 3×3 rotation matrices('cam_R_m2c') are converted into unit quaternions. This transformation improves numerical stability and facilitates efficient regression. The translation vectors('cam_t_m2c'), originally in millimeters, are rescaled to meters. The dataset is then randomly split into 80% training and 20% validation subsets.

Training is conducted using the Adam optimizer with a learning rate of 1×10^{-4} and a batch size of 16. The model achieving the lowest validation loss during training was saved as the best-performing checkpoint for subsequent evaluation. After each epoch, the model's performance was assessed on the validation set using the same loss function applied during training. Evaluation was conducted exclusively on the held-out 20% validation subset to monitor generalization and prevent overfitting.

3.4. Pose Estimation Network

The proposed pipeline for 6D object pose estimation begins with data preprocessing, where each scene provides paired RGB and depth images along with corresponding 6D object poses defined by rotation and translation values. The RGB images are resized to a uniform resolution of 224 by 224 pixels and normalized. Depth maps are similarly resized and then normalized to scale them into the range between 0 and 1. This preprocessing ensures consistency across samples and facilitates convergence during training.

The dataset is organized such that each object class has its own directory containing RGB images, depth maps, and pose annotations stored in gt.yml files. A custom PyTorch dataset class is implemented to load these multimodal inputs. During training, light augmentations such as horizontal flipping and color jitter are applied to the RGB inputs, while the depth inputs remain unaugmented to preserve geometric integrity.

The proposed model consists of two distinct backbone networks. The RGB modality is processed through a ResNet-50 network pretrained on ImageNet, which extracts high-level visual features and outputs a 512-dimensional embedding. In parallel, the depth modality is passed through a modified ResNet-18 network initialized from scratch, adapted to accept single-channel input and also producing a 512-dimensional feature vector. These two branches operate independently until their outputs are concatenated to form a unified 1024-dimensional feature vector.

At this point, a mechanism is introduced to adaptively fuse the RGB and depth features. A two-layer multilayer perceptron (MLP) is used to process the concatenated features and produce two weights via a sigmoid activation function. These weights represent the model's learned confidence in each modality. The weights are then applied to scale the RGB and depth features separately. The features are summed/concatenated to produce a final fused representation, which effectively integrates appearance and geometric information.

From this fused feature representation, the model branches into two heads responsible for predicting pose components. One head regresses a four-dimensional vector representing a unit quaternion for object rotation, and the other predicts a three-dimensional translation vector. To ensure the quaternion represents a valid rotation, it is L2-normalized before loss computation.

The model is trained using a composite loss function. The rotation loss is defined as one minus the squared inner product of the predicted and ground truth quaternions, which encourages angular alignment. The translation loss is calculated using the mean squared error between the predicted and ground truth translation vectors. The total loss is a direct sum of these two components, guiding the network to learn both orientation and position simultaneously.

Training is conducted using the AdamW optimizer with a fixed learning rate of 1e-4 and a batch size of 16 over the course of 100 epochs. The training is accelerated using GPU hardware, such as an NVIDIA Tesla T4, to facilitate faster computation. For evaluation, the model's performance is assessed using the Average Distance of Model Points ADD metric, which measures the discrepancy between the predicted and ground truth poses by averaging the distance between corresponding 3D model points after transformation.

This pipeline integrates deep visual and geometric learning through a dual fusion strategy that dynamically leverages the strengths of each modality, resulting in enhanced pose estimation accuracy even in complex real-world environments

3.5. Evaluation Metrics

We adopt the Average Distance (ADD) metric as proposed in [12] for evaluation. Given the ground truth rota-

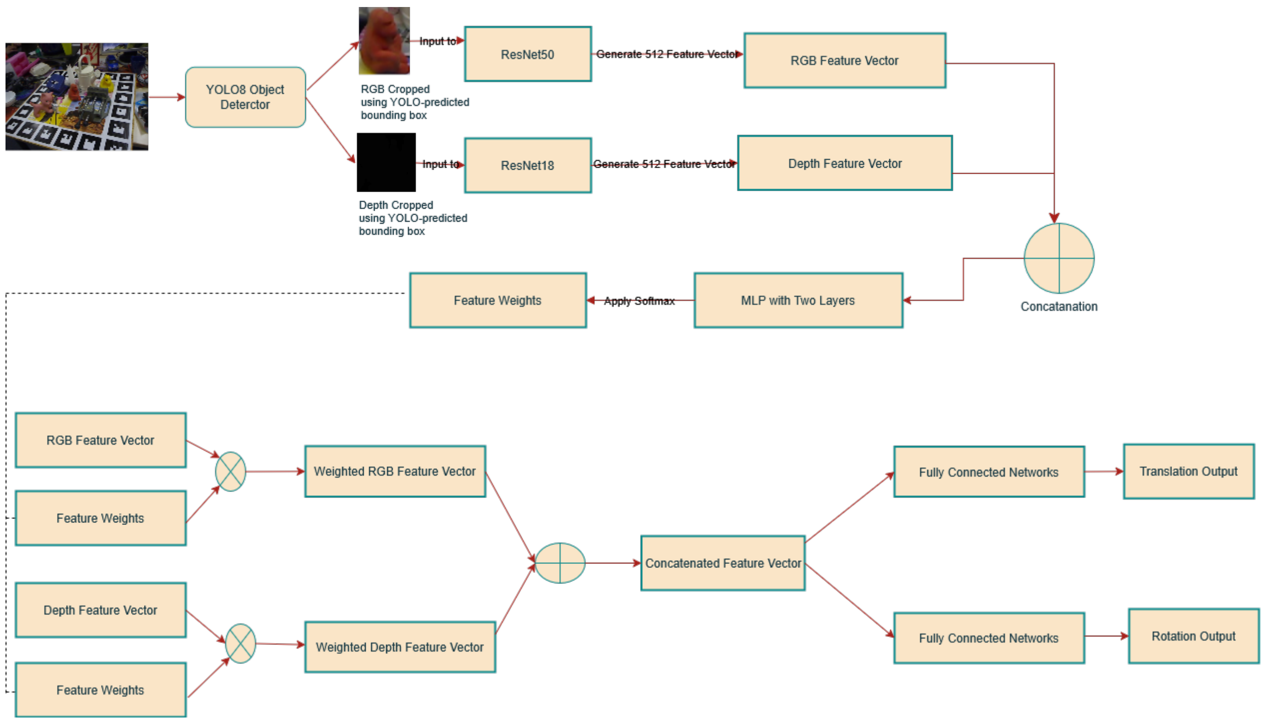

Figure 3. Dual Fusion PoseNet pipeline for 6D pose estimation . (1) An RGB image is processed by a YOLO8 object detector to localize the object. (2) The detected bounding box is used to crop both RGB and corresponding depth images. (3) The cropped RGB image is passed through a ResNet50 encoder and th depth image through a ResNet18 encoder, each producing a 512-dimensional feature vector. (4) These features are concatenated into a 1024-dimensional vector and passed through a two-layer MLP with softmax activation to compute attention weights. (5) The weighted concatenation of the original feature vectors produces a fused 1024-dimensional representation. (6) This final feature is fed into two separate fully connected heads to regress a 4D quaternion for rotation and a 3D translation vector.

tion $\mathbf{R}$ and translation $\mathbf{T}$, and the estimated rotation $\hat{\mathbf{R}}$ and translation $\hat{\mathbf{T}}$, the ADD metric computes the mean pairwise distance between the 3D model points transformed by the ground truth pose and those transformed by the estimated pose. Evaluation is conducted on the LINEMOD test split.

$$\text{ADD} = \frac{1}{m} \sum_{\mathbf{x} \in \mathcal{M}} \left\| \mathbf{R}\mathbf{x} + \mathbf{T} - \hat{\mathbf{R}}\mathbf{x} - \hat{\mathbf{T}} \right\|, \tag{1}$$

where $\mathcal{M}$ denotes the set of 3D model points and m is the number of points.

4. Experiments and results

This section discusses additional extensions applied to the baseline model along with their corresponding results. The original training dataset was divided into training and validation subsets. The model was trained on the training subset while its performance was monitored on the validation subset by tracking the validation loss. The model checkpoint with the lowest validation loss was selected as the best-performing model, which was subsequently evaluated on the test set to assess its generalization performance.

4.1. Setup

We evaluate our method on the LINEMOD dataset using the Average Distance of Model Points (ADD) metric, which measures the average point-wise distance between the predicted and ground-truth object poses.

Our baseline model utilizes only RGB features, which are directly passed to the pose regression heads. In the proposed extension, we introduce a feature fusion module that computes adaptive weights over RGB and depth embeddings. These weights are used to generate a weighted sum (512-dimensional) and a weighted concatenation (1024-dimensional) of the RGB and depth features. The resulting fused representations are then fed into the pose regression module for 6D pose estimation.

4.2. Result

We compare the baseline and our proposed extension using per-class ADD . As shown in Tables 1, our extension significantly improves the overall pose estimation quality, reducing the average ADD from 110.3 mm to 61.4 mm.

Table 1. Mean ADD (mm) per object class for Baseline and Dual Fusion PoseNet.

Class	Baseline	Dual Fusion PoseNet (Feature Summation)	Dual Fusion PoseNet (Feature Concatenation)
01	123.6	68.1	71.3
02	100.9	61.0	51.0
04	100.5	66.3	57.0
05	103.3	69.0	57.7
06	110.9	60.04	63.9
08	110.5	61.2	54.9
09	117.4	81.0	73.8
10	112.7	63.0	67.2
11	110.0	71.07	65.2
12	116.4	65.5	57.9
13	112.4	65.5	61.4
14	106.6	70.6	59.8
15	107.9	57.0	56.8
Overall Mean	**110.3**	**66.2**	**61.4**

4.3. Optimizer

This section explores the use of different optimization algorithms during model training, each applied with its default parameter settings

- **SGD**: SGD is an iterative method for optimizing a differentiable objective function. It updates the weights and bias terms by computing the gradient of the loss function over minibatches of data.

- **Adam**: Adam (Adaptive Moment Estimation) [13] is an optimization algorithm that computes adaptive learning rates for each parameter by estimating both the first and second moments of the gradients. The update rule is given by:

$$w_{t+1} = w_t - \frac{\eta\, \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

  where $\hat{m}_t$ represents the first moment (mean) of the gradients, $\hat{v}_t$ is the second moment (uncentered variance), and ϵ is a small constant to prevent division by zero.

- **AdamW**: AdamW [14] is a variant of the Adam optimizer that decouples weight decay from the gradient-based update. Unlike Adam, where weight decay is implemented as L2 regularization, AdamW applies an explicit penalty on the weights during the update step. The update rule is:

$$w_{t+1} = w_t - \frac{\eta\, \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} - \lambda \eta w_t$$

  Here, λ is the weight decay coefficient, $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected first and second moment estimates, and ϵ is a small constant to ensure numerical stability.

- **RMSProp**: RMSProp (Root Mean Square Propagation) [15] is an adaptive learning rate optimization algorithm that maintains an exponential moving average of the squared gradients. Unlike Adam, it does not compute first-order moment estimates. This approach helps normalize the updates and improves convergence stability, especially in non-stationary settings.

  The results are summarized in Table 2.

Table 2. Optimizer comparison on Dual Fusion PoseNet (LR = 10^{-4})

Optimizer	Dual Fusion PoseNet (Feature Concatenation)	Dual Fusion PoseNet (Feature Summation)
SGD	104.3	109.3
Adam	84.5	86.4
AdamW	61.4	66.2
RMSProp	85.2	81.7

4.4. Learning Rate

Additionally, for the best-performing optimizer, AdamW, a range of learning rates lr $\in \{10^{-4}, 10^{-3}\}$ and weight decay values wd $\in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$ were tested to determine the most suitable configuration. The results are summarized in Table 3.

Table 3. Performance of AdamW optimizer with different learning rates and weight decay values

Learning Rate / Weight Decay	10^{-2}	10^{-3}	10^{-4}	10^{-5}
10^{-4}	34.0	42.1	41.3	57.1
10^{-3}	104.8	118.6	117.7	103.8

4.5. Losses

Additionally, using the best-performing optimizer, AdamW, with a learning rate of 10^{-4} and a weight decay of 10^{-2}, we investigated various loss functions to identify the most suitable configuration. The results of this analysis are summarized in Table 4.

Mean Squared Error (MSE) Pose Loss This loss computes the mean squared error between the predicted and ground truth quaternion rotations and translations:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N} \|\hat{q}_i - q_i\|^2 + \frac{1}{N}\sum_{i=1}^{N} \|\hat{t}_i - t_i\|^2 \quad (2)$$

where $\hat{q}_i, \hat{t}_i$ are the predicted rotation quaternion and translation vector, and q_i, t_i are the ground truth values.

Angle-Based Pose Loss The predicted and ground truth quaternions are normalized, and the rotational loss is defined using the cosine similarity:

$$\mathcal{L}_{\text{angle}} = \frac{1}{N}\sum_{i=1}^{N}(1 - |\hat{q}_i \cdot q_i|) + \frac{1}{N}\sum_{i=1}^{N}\|\hat{t}_i - t_i\|^2 \quad (3)$$

where $\cdot$ denotes the dot product.

Smooth L1 Pose Loss The Smooth L1 loss (Huber loss) is applied separately to the rotation and translation components:

$$\mathcal{L}_{\text{Smooth L1}} = \text{SmoothL1}(\hat{q}, q) + \text{SmoothL1}(\hat{t}, t) \quad (4)$$

Quaternion Dot Product Pose Loss The rotational loss is based on the squared dot product between predicted and ground truth quaternions:

$$\mathcal{L}_{\text{rot}} = \frac{1}{N}\sum_{i=1}^{N}\left(1 - (\hat{q}_i \cdot q_i)^2\right) \quad (5)$$

The total pose loss combines rotation and translation terms:

$$\mathcal{L}_{\text{pose}} = \mathcal{L}_{\text{rot}} + \frac{1}{N}\sum_{i=1}^{N}\|\hat{t}_i - t_i\|^2 \quad (6)$$

Table 4. ADD (mm) on the test dataset for different loss functions. The model was trained using AdamW optimizer with learning rate 1×10^{-4} and weight decay 1×10^{-2}.

Loss Type	**ADD (mm)**
Angle-Based Pose Loss	33.2
Smooth L1 Pose Loss	37.2
Mean Squared Error (MSE) Pose Loss	45.6
Quaternion Dot Product Pose Los	34.0

4.6. Best Performing Configuration

We evaluate the per-class average ADD (Average Distance of model points) in millimeters using the optimal configuration identified through extensive experimentation. This setup leverages the AdamW optimizer with a learning rate of 1×10^{-4}, a weight decay of 1×10^{-2}, and the angle-based loss function which yielded the best performance in terms of pose accuracy is selected. The results, summarized in Table 5.

Table 5. Per-class average $\overline{\text{ADD}}$ in millimeters. Results are obtained using the best-performing configuration: the AdamW optimizer with a learning rate of 1×10^{-4}, weight decay of 1×10^{-2}, and the best loss function.

Class	$\overline{\text{ADD}}$ (mm)
Class 01	34.8
Class 02	29.9
Class 04	34.0
Class 05	29.7
Class 06	31.0
Class 08	31.3
Class 09	36.7
Class 10	35.2
Class 11	35.9
Class 12	33.3
Class 13	33.6
Class 14	34.0
Class 15	32.1
Overall Mean	**33.2**

5. Conclusion

We propose DualFusion-PoseNet, a method for 6D pose estimation. Our approach extracts features using two separate backbones—ResNet50 for RGB and ResNet18 for depth. These features are concatenated and passed through a two-layer MLP with a softmax that learns to weigh each modality based on its contribution to the final prediction. The fused 1024-dimensional representation is then used to regress both the object's rotation (as a 4D quaternion) and translation (as a 3D vector) via two fully connected branches. This dual fusion framework enables adaptive integration of color and depth cues, improving robustness to occlusion and noise. Experimental results on the LINEMOD dataset demonstrate that DualFusion-PoseNet outperforms the baseline, achieving lower ADD errors.

References

[1] Chen Yuanwei, Mohd Hairi Mohd Zaman, and Mohd Faisal Ibrahim. A review on six degrees of freedom (6d) pose estimation for robotic applications. 2023. Supported by Universiti Kebangsaan Malaysia under Grant GUP-2023-016. 1

[2] R. Vock, A. Dieckmann, S. Ochmann, and R. Klein. Fast template matching and pose estimation in 3d point clouds. *Comput. Graph.*, 79:36–45, Apr 2019. 1

[3] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.*, 60(2):91–110, Nov 2004. 2

[4] W. Zhao, S. Zhang, Z. Guan, W. Zhao, J. Peng, and J. Fan. Learning deep network for detecting 3d object keypoints and 6d poses. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14122–14130, 2020. 2

[5] W. Zhao, S. Zhang, Z. Guan, H. Luo, L. Tang, J. Peng, et al. 6d object pose estimation via viewpoint relation reasoning. *Neurocomputing*, 389:9–17, 2020. 2

[6] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again. pages 1530–1538, Oct 2017. 2

[7] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. pages 1–10, Jun 2018. 2

[8] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. DenseFusion: 6D object pose estimation by iterative dense fusion. pages 3338–3347, Jun 2019. 2

[9] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015. 2

[10] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2016. 2

[11] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 858–865, 2011. 3

[12] 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*, pages 548–562. Springer, 2012. 4

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[14] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2019. 6

[15] T. Tieleman and G. Hinton. Lecture 6.5 - rmsprop: Divide the gradient by a running average of its recent magnitude. Coursera: Neural Networks for Machine Learning, 2012. Accessed May 2025. 6