LightCBAM-ResNet: A Lightweight Attention-Enhanced Backbone for Camera Pose Estimation

Yuer Tang, Junze He, Andrea Baretta, Zhiyi Chen, McKay McFadden, YuSheng Li

Abstract

Accurate camera pose estimation is critical for a wide range of robotic applications, particularly in environments where GNSS signals are unreliable or unavailable. Traditional visual odometry methods rely heavily on feature tracking and probabilistic filtering, which can be fragile under challenging visual conditions. PoseNet introduced a learning-based alternative, demonstrating that Convolutional Neural Networks (CNNs) can directly regress 6-DoF camera poses from single images. However, PoseNet suffers from limited generalization and suboptimal accuracy. In this work, we propose an improved architecture that integrates the Convolutional Block Attention Module (CBAM) into a ResNet backbone for Absolute Pose Regression (APR). Experimental results indicate that our CBAM-augmented network achieves improved accuracy and generalization over the baseline PoseNet architecture, offering a promising direction for attention-based learning in camera pose estimation.

1 Introduction

Knowing the pose, position, and orientation of a camera is fundamental to a wide range of robotic applications. In many real-world scenarios, Unmanned Aerial Vehicles (UAVs) and humanoid robots cannot rely on GNSS (Global Navigation Satellite Systems) for positioning. In these cases, UAVs must rely on alternative onboard sensors—most notably, cameras—to estimate their own pose. Given the central role of cameras in robotic perception, the natural question arises: how does one estimate the pose from a camera feed? This is a well-established area of study, typically falling under the umbrella of Visual Odometry (VO). Traditional approaches to visual odometry originate from the theory of stochastic dynamic systems, where pose estimates are generated through variations of the Kalman filter.

However, these pipelines have limitations. Feature tracking can be unreliable, especially in texture-poor environments or under abrupt motion, leading to degraded pose estimates. A seminal paper called PoseNet [6] proposed a novel idea: What if the mapping from image to pose could be learned directly? In other words, rather than tracking features and computing pose analytically, a deep neural network could be trained to regress the pose of a camera directly from an image. This end-to-end learning approach eliminates the need for fragile feature tracking and enables real-time inference given sufficient computational resources.

PoseNet successfully demonstrated that Convolutional Neural Networks (CNNs) can be trained to produce 6-DoF camera poses from single images. While the results were promising, the method had significant limitations. Notably, the model struggled to generalize to new scenes outside the training dataset and often produced relatively low-accuracy pose estimates compared to traditional methods.

This work aims to improve upon PoseNet by incorporating a more powerful base network and enhancing it with a Convolutional Block Attention Module (CBAM). CBAM introduces lightweight attention mechanisms that help the network focus on the most informative spatial and channel-wise features in the image, improving both accuracy and generalization. By

integrating CBAM into a residual network architecture, this research explores how attention-enhanced deep learning can push the boundaries of camera pose estimation.

2 Background

In this section, we provide an overview of the key concepts relevant to our approach. Specifically, we outline the fundamentals of Convolutional Neural Networks (CNNs), the Convolutional Block Attention Module (CBAM), and Residual Networks (ResNet), which are the core architectural components leveraged in addressing the 6-DoF camera pose regression problem.

2.1 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) have emerged as a powerful class of deep learning models, particularly suited for computer vision tasks and the process of image datasets. Introduced by LeCun et al. [8], CNNs leverage spatial hierarchies in data through local connectivity, learnable kernels with weight sharing, and pooling operations, enabling them to capture local patterns in the early layers, and more complex structures in the deeper layers.

One of the key advantages of CNNs is their ability to exploit the 2D structure of images, reducing the number of parameters compared to fully connected networks and making them more efficient and generalizable to image-based tasks [2]. We use VGG16, which is a widely-used deep CNN architecture, as the baseline model.

2.2 Residual Networks (ResNet)

As researchers grew CNNs deeper to improve performance, they encountered the degradation problem: adding more layers often led to higher training error and difficulty in optimization, contrary to expectations. To address this, He et al. [3] proposed Residual Networks (ResNet), which introduced the concept of residual learning. Instead of directly learning a desired mapping $H(x)$, ResNet pivots the objective into learning the residual function $F(x) = H(x) - x$, so that the original mapping becomes $H(x) = F(x) + x$.

Adding the input x back to the learned function provides the network with the flexibility to approximate the identity mapping, by setting the residual function $F(x) = 0$, resulting in $H(x) = x$. This acts as a form of performance safeguard, allowing the model to skip unnecessary transformations at intermediate layers. Consequently, it mitigates the risk of performance degradation in deeper networks by enabling the model to preserve information across layers when deeper transformations are not beneficial [3]. The residual framework enables better feature reuse and improves optimization, making ResNet a foundational architecture in modern deep learning and computer vision pipelines.

2.3 Convolutional Block Attention Module (CBAM)

Attention mechanisms have become a cornerstone in deep learning, enabling models to dynamically focus on the most relevant parts of the input data. Originally popularized in sequence modeling tasks such as machine translation [1, 11], attention has been widely adopted in computer vision for its ability to enhance feature representation by selectively emphasizing informative regions and suppressing irrelevant ones [12].

In the context of CNNs, attention mechanisms can be used to guide the network to focus on important spatial locations and important feature channels. The Convolutional Block Attention Module (CBAM), proposed by Woo et al. [10], is a lightweight and effective attention module. Sequentially, CBAM applies channel attention to emphasize meaningful feature maps, followed by spatial attention to localize critical regions in the image.

By refining intermediate feature representations, CBAM improves the ability of the model to capture discriminative features, leading to improved performance. Our model, which will be discussed further in the Model section, integrates the CBAM module with ResNet architecture.

3 Model

3.1 Convolutional Block Attention Module

In CBAM, a feature map is defined as $\mathbf{F} \in \mathbb{R}^{C \times H \times W}$, where C is cardinality, H is height, and W is width [10]. Sequentially, the **Channel Attention Module (CAM)** transforms the feature map F into a 1D channel attention map, denoted $\mathbf{M_c}(F) \in \mathbb{R}^{C \times 1 \times 1}$. CAM then performs element-wise multiplication between $\mathbf{M_c}(F)$ and F along the channel dimension to produce an intermediate output. This output is passed to the **Spatial Attention Module (SAM)**, which generates a 2D spatial attention map, denoted as $\mathbf{M_s}(F) \in \mathbb{R}^{1 \times H \times W}$ [10]. Finally, SAM multiplies this spatial attention map with the input feature map (from CAM) element-wise to produce the final output. The complete CBAM can be described sequentially as follows:

$$
\begin{aligned}
F_1 &= M_c(F) \otimes F \\
F_2 &= M_s(F_c) \otimes F_1
\end{aligned}
$$

where $F_1 \in \mathbf{R^{C \times W \times H}}$ is the output of CAM, and $F_2 \in \mathbf{R^{C \times W \times H}}$ is the output from SAM, $\otimes$ denotes the element-wise multiplication [10].

3.2 ResNet with CBAM

We implemented a simplified ResNet block composed of three convolutional layers, each followed by batch normalization and ReLU activation, except for the final layer which omits activation [10]. Given an input tensor $\mathbf{x}$, we store it as *residual* for the skip connection. The block transforms the input as follows:

$$
\begin{aligned}
\mathbf{z}_1 &= \mathrm{ReLU}(\mathrm{BN}_1(\mathrm{Conv}_1(\mathbf{x}))) \\
\mathbf{z}_2 &= \mathrm{ReLU}(\mathrm{BN}_2(\mathrm{Conv}_2(\mathbf{z}_1))) \\
\mathbf{z}_3 &= \mathrm{BN}_3(\mathrm{Conv}_3(\mathbf{z}_2)) \\
\mathbf{z}_4 &= \mathrm{CBAM}(\mathbf{z}_3) \\
\mathbf{y} &= \mathbf{z}_4 + \mathbf{residual} \\
\mathbf{y}_{\text{out}} &= \mathrm{ReLU}(\mathbf{y})
\end{aligned}
$$

3.3 ResNet with CBAM for Absolute Pose Regression

Since the ResNet with CBAM is specifically for image classification and detection, we have to modify the ResNet model to achieve our purpose. The changes include the following:

1. Add a **Pose Regression Module** as the output layer to the ResNet with CBAM
2. Add a **Pose Loss Module** to evaluate generalization of the ResNet with CBAM

3.3.1 Pose Regression Module

Pose regression module is a simple multi-layer perceptron. Our ResNet with CBAM for APR consists of position and orientation output layers. We cannot regress position and orientation simultaneously because they are in different units. Furthermore, ResNet with CBAM can use

the previous pose as part of its prediction logic by setting the parameter **use_prior** = True in the pose regression module [9]. The whole structure of pose regression module is defined as follows:

$$h = GELU(W_{hidden}x + b_{hidden})$$
$$\hat{y} = W_{output}h + b_{output}$$

where $x \in \mathbf{R}^{input\_dim}$, $W_{hidden} \in \mathbf{R}^{1024 \times input\_dim}$ and $b_{hidden} \in \mathbf{R}^{1024}$ are the weights and bias of the hidden layer, and $W_{output} \in \mathbf{R}^{output\_dim \times 1024}$ and $b_{output} \in \mathbf{R}^{1024}$ are the weights and bias for the output layer. If **use_prior** = True, W_{hidden} is replaced with $W_{hidden,prior} \in \mathbf{R}^{1024 \times 2input\_dim}$ [9]. After processing image features through the last ResNet layer, we define two heads. One head is for position regression with output dimension 3 and one head is for orientation regression with output dimension 4.

3.3.2 Pose Loss Module

To evaludate model's performance, we can implement **L2 loss** to assess the prediction of camera position and orientation of our model. The pose of the ground-truth camera is defined as (p, o), where p is the position and o is the orientation, and the estimated camera pose denotes ($\hat{p}$, $\hat{o}$) for the estimated position and the orientation, respectively [9]. We define the position loss $L_p = ||p - \hat{p}||_2$ and the orientation loss $L_o = ||o - \frac{\hat{o}}{||\hat{o}||_2}||_2$.

Since the camera position is unit meters and the orientation is unit quaternions, we have to balance these two types of loss function. The first solution is that we employ a static loss function applying a static scaling weight β to orientation. This is sufficient when one component is more important than the others. For example, we may consider position to be more important than orientation, so we may set a larger scaling weight to reduce the influence of orientation. The second solution is to employ two learnable scaling weights s_p and s_o to adjust the uncertainty of two loss values [9]. Instead of a fixed scaling weight, we let our model learn the scaling weights. This not only scale orientation, but also scale position. The two learnable scaling weights are optimized during backpropagation by Adam. The trainable and static pose loss function are given by:

$$L_{static\_pose} = L_p + L_s \times \beta$$
$$L_{trianable\_pose} = L_p e^{-s_p} + s_p + L_o e^{-s_o} + s_o$$

In the trainable pose loss function, s_p and s_o are treated as regularization terms to prevent the weights from collapsing to zero, ensuring stable learning. Moreover , the intuition behind taking the logarithmic inverse of s_p and s_o is that when uncertainty is high, the model downweights the loss using e^{-s_x}; when uncertainty is low, it emphasizes the loss, letting the model focus more on confident predictions.

4 Dataset

The King's College dataset is one of several outdoor scene datasets introduced in the original PoseNet paper to evaluate **Absolute Pose Regression** (APR) models. It consists of RGB images captured on the campus of King's College in Cambridge, UK, using a handheld smartphone camera[7]. Each image is paired with a ground-truth 6-DoF camera pose, comprising a 3D position vector and a 4D orientation quaternion. These ground-truth poses were generated using structure-from-motion (SfM) techniques.

The dataset was created by recording a video of a person walking through the King's College courtyard. Individual frames were extracted from the video, and VisualSfM was applied to compute the corresponding 6-DoF pose for each frame. The resulting dataset consists of frame–pose

pairs, where each RGB image serves as the input feature, and the associated pose acts as the learning target.

This dataset reflects many real-world challenges inherent to outdoor environments, including dynamic elements such as pedestrians and vehicles, variations in lighting and shadows, and regions with low visual texture.

5 Methodology

In this project, we used the King's College dataset to maintain continuity with the original PoseNet work. We randomly split the dataset 70-30 for training and testing, respectively. It served as a fine-tuning dataset for our ResNet+CBAM model. While our ResNet backbone was initially pre-trained on large-scale image classification datasets such as ImageNet or CIFAR, we leveraged only the resulting feature extractor through a transfer learning approach. The goal was to adapt the pre-trained model to the task of estimating the camera pose using the King's College data.

We followed a structured training pipeline using the King's College dataset, which already provides predefined training, validation, and test splits. Therefore, no additional data separation was necessary. We trained and compared three models: VGG16, ResNet50, and ResNet50 with CBAM. Each model was trained for 50 epochs using the Adam optimizer. For the loss function, we experimented with both static and trainable pose loss variants. When using the trainable pose loss, two learnable scaling weights were optimized alongside the model parameters to balance translation and orientation components.

We took great care to avoid any form of data leakage or knowledge leakage. The test set was strictly held and was not used in any part of the training, validation, or model selection process, including during exploratory data analysis or preprocessing steps to prevent human or data bias. To evaluate model performance, we used standard metrics for pose estimation: median translation error (in meters) and median rotation error (in degrees). These metrics were computed solely on the test set after training was complete.

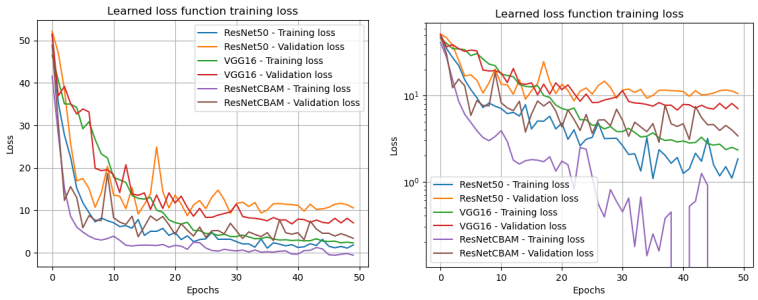
6 Results

Figure 1 plots every loss curve we recorded for all three backbones under two loss regimes. Sub–figures **(a)** (1a) and **(b)** (1b) correspond to the learned-s_p and s_o CameraPoseLoss, reported on linear and $\log_{10}$ scales, respectively. Sub–figures **(c)** (1c) and **(d)** (1d) show the same curves when a fixed-weight MSE loss is used. **Code:** `https://github.com/YuerTang/Math-156-Project`

Speed of convergence. All three models drop their loss a lot in the first five epochs, but ResNet-CBAM hits the "single-digit" zone first (purple line in Fig. 1a). The log plot (Fig. 1b) looks almost like a straight line, which means the loss is shrinking by the same factor each epoch —close to exponential decay.

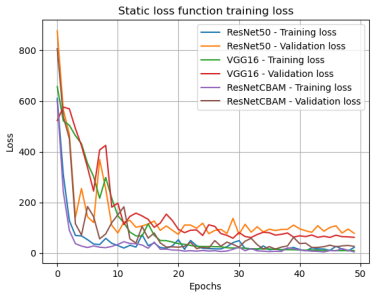
Generalisation gap. Throughout training, ResNet-CBAM keeps its training and validation losses very close together, which is a good sign for generalisation. By contrast, the validation loss for plain ResNet-50 (orange) stops coming down early and even bumps up a bit later, showing mild over-fitting; this gets worse when we use the static-β loss (Figs. 1c–1d).

Learned vs. fixed $\boldsymbol{\beta}$. When training models, especially those with multiple output components like translation and rotation, the method of balancing each part's contribution to the total loss is critical. Our observations clearly demonstrate a significant benefit to allowing the



(a) Learned Loss

(b) Learned Loss Log



(c) Static Loss

(d) Static Loss Log

Figure 1: Loss curves under linear and logarithmic scales for learned and static settings.

network to **learn its own balance factors** (s_q and s_o) within the loss function, as opposed to employing a fixed β (typical of a plain Mean Squared Error, or MSE).

Comparing the results in Figures 1a and 1c reveals two key points:

- Firstly, loss values are substantially lower when the network determines its own balancing. Specifically, using a fixed β results in loss values that are roughly **ten times larger** on the linear plots.
- Secondly, and more critically, models trained with a fixed β show their validation curves halting their descent earlier and stabilizing at a higher error level. This indicates less effective training convergence and, consequently, poorer generalization performance.

In essence, granting the network the flexibility to dynamically weigh the importance of translation versus rotation leads to **smoother training convergence** and ultimately **superior generalization**. This strongly suggests that the optimal balance is not static, and enabling the model to discover it autonomously is crucial for achieving peak performance.

Open question. We do not yet have a full theoretical explanation for why the self-tuning s_q and s_o works so well. A leading intuition—echoing [5, 4]—is that the network treats the translation and rotation errors as sharing a single amount of noise and keeps re-scaling the two loss terms to match it. This is only a first guess; future work will include targeted experiments and deeper analysis to confirm (or refute) this idea.

7 Discussion and Conclusion

Our experiments show that equipping a ResNet backbone with Convolutional Block Attention Modules (CBAM) delivers two clear benefits for learning-based camera-pose estimation:

(i) **Faster and more stable convergence.** Loss curves in Fig. 1 reveal that CBAM accelerates convergence in the first five epochs and maintains the narrowest train–validation gap, indicating reduced over-fitting.

(ii) **Improved representation focus.** By selectively emphasising salient channel- and spatial cues, CBAM helps the network attend to image regions that are most informative for 6-DoF pose, boosting both accuracy and robustness in visually cluttered scenes.

These gains are further amplified when the translation/rotation trade-off parameter s_o and s_p are learned rather than fixed β, yielding smoother loss trajectories across all backbones. A full quantitative assessment on the held-out test split—reporting median translation and rotation errors—is left to future work once the pose-error annotations become available.

In summary, replacing PoseNet's original VGG-16 with a **ResNet + CBAM** architecture advances absolute pose regression by coupling deeper feature extraction with attention-driven relevance filtering, setting a solid baseline for subsequent research on attention-enhanced localization.

8 Acknowledgment

This project was completed as part of MATH 156 under the guidance of Professor Lara Kassab, whose insights and feedback were invaluable throughout the term. Additional troubleshooting and model exploration benefited from publicly available discussions and tutorials on Stack Overflow and GeeksforGeeks.

Author Contributions

- **Yuer Tang** – Team management and task assignment; main author of the *Results*, *Discussion*, and overall *Proposal* documents; first to suggest replacing PoseNet's VGG16 backbone with a CNN variant for camera-pose estimation; liaison with Prof. Lara and team-mates.
- **Junze He** – Proposed and implemented the CBAM attention module and the swap from VGG16 to a ResNet backbone; primary author of the *Methodology* section.
- **Andrea Baretta** – Built the unified training pipeline, making model selection and hyperparameter tuning easier; executed all training runs and generated Fig. 1 loss-curve plots.
- **Zhiyi Chen** – Wrote production-quality code for CBAM and the `data_processing` class; main author of the *Background* section.
- **McKay McFadden** – Originated the camera-pose estimation topic; first to suggest replacing PoseNet's VGG16 backbone with a CNN variant for camera-pose estimation; drafted the *Abstract*, *Introduction*, and *Dataset* subsections.
- **Yusheng Li** – Performed the literature review to ground the chosen algorithm; authored the *Conclusion*, *Acknowledgment*, and *End-note* sections.

9 End Note

Below are some components the project could touch upon, but did not explore due to scale or time limitations. Future work might focus on these aspects.

The CBAM–ResNet model was trained and tested only on the King's College scene, so its improved accuracy may not carry over to other environments. Evaluating the pipeline on additional benchmarks such as 7 Scenes, TUM RGB–D, and DeepLoc would provide vital cross-scene validation. Collecting new datasets with varied lighting, noise, and motion would further test

whether CBAM consistently highlights informative image regions. Training and evaluation were limited to this single dataset because obtaining additional image collections online involves lengthy downloads and substantial storage requirements.
Adding CBAM improves pose accuracy but increases GPU memory usage and inference time, which can introduce latency on edge devices in time-critical applications. Future work will explore lighter ResNet variants such as ResNet 18 and ResNet 34, together with pruning or depthwise separable convolutions, to identify the smallest model that preserves accuracy. A compact, attention-enhanced network could then deliver real-time 6-DoF localization on lightweight GPUs and embedded platforms. This efficiency optimization was not pursued because it lay outside our original scope, and proper evaluation would require deployment on actual robotic platforms with the relevant embedded chips, which were unavailable to us.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `https://www.deeplearningbook.org`.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–778, 2016.

[4] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[5] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in Neural Information Processing Systems*, 30, 2017.

[6] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2938–2946, 2015.

[7] Grimes M. Cipolla-R. Kendall, A. Research data supporting "posenet: A convolutional network for real-time 6-dof camera relocalization": Kings college. 2015.

[8] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[9] Jae-Pil Heo Miso Lee, Jihwan Kim. Activating self-attention for multi-scene absolute pose regression. 2024.

[10] Joon-Young Lee In So Kweon Sanghyun Woo, Jongchan Park. Cbam: Convolutional block attention module. 2018.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, volume 30, 2017.

[12] Fei Wang, Meng Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164, 2017.