RLIO: Reliable Learned Inertial Odometry

Qi Zhang, Junxiang Ji, ShengTeng Ong, Yuhui Chen, Lizhi Yang

School of informatics Xiamen University
{23320231154459, 23020231154142, 31520231157797, 23020231154173, 31520231154323}@stu.xmu.edu.cn

Abstract

RILO presents a novel framework aiming to enhance the precision of Inertial Measurement Unit (IMU)-based motion estimation. Leveraging a backbone ResNet18 architecture, RILO integrates an auxiliary neural network (NET2) to discern training-real world motion disparities. NET2 learns basic weights (w_b) initially, capturing model error patterns. Subsequent fine-tuning refines weights (w_c) to align with realworld motion features. Employing TLIO dataset and visualinertial filter supervision, experiments showcase RILO's superiority over TLIO, achieving a reduced RMSE of 1.305 with MSE loss, showcasing the efficacy of the proposed auxiliary network in improving IMU-based motion estimation.

Introduction

In recent decades, Inertial Measurement Units (IMU) have played a pivotal role in various fields, including robotics, augmented reality, and autonomous vehicles, offering crucial information for navigation, motion tracking, and spatial awareness(Chen and Pan 2023). While IMU are widely used to measure acceleration, angular velocity, and orientation, this paper adopts the standard definition of IMU as a device that integrates accelerometers and gyroscopes to estimate the object's motion in three-dimensional space(Huang et al. 2022)(Li et al. 2022). The significance of IMU lies in their ability to provide real-time data that aids in navigation, object tracking, and environmental mapping.

Despite extensive research in IMU technology, addressing the inherent uncertainties has remained a challenge, and no single study has comprehensively tackled this issue to date. This paper explores the application of deep learning techniques to mitigate the uncertainties associated with IMU. By leveraging the power of neural networks, we aim to enhance the accuracy and reliability of IMU-based systems. This research, for the first time, investigates the use of deep learning models for IMU data, offering a potential solution to the longstanding problem of uncertainty in motion and orientation estimation. The primary issues addressed in this paper include a) the challenges posed by IMU uncertainty,and b) the application of deep learning to improve IMU data.

Related work

PDR. Pedestrian dead reckoning(Harle 2013) (PDR) forms a dead-reckoning model by detecting gait information, which is collected by IMU. This approach consists of several submodules: step detection, step length estimation and step orientation estimation. Hand-designed parameters are required by these modules, which means that it's difficult to cover the great mass of motion patterns. In short, All of traditional gait model-based methods cannot deal with complex but normal scenes.

TLIO. Tight Learned Inertial Odometry(Liu et al. 2020) (TLIO) uses Extended Kalman Filter (EKF) and ResNet18 model to reduce drift and uncertainty for EKF. The network offers the displacement estimates and their uncertainties which parametrize the diagonal entries of the covariance. The EKF propagates with raw IMU samples and uses network outputs for measurement updates. However, in cases where the datasets do not cover certain motion patterns, the model's predicted output can exhibit significant deviations from the expected behavior. This approach needs high-quality datasets to establish the relationship between the IMU and pose, which is difficult. Similar to TLIO, RONIN which is alse data-driving inertial odometry face the same problem.

Uncertainty Estimation and Evaluation of DNN. According to previous work, there exist some way to estimate the uncertainty.

Single deterministic methods give the prediction based on one single forward pass within a deterministic network. They can be roughly categorized into approaches where one single network is explicitly modeled and trained in order to quantify uncertainties (Sensoy, Kaplan, and Kandemir 2018) (Malinin and Gales 2018) (Możejko, Susik, and Karczewski 2018) (Nandy, Hsu, and Lee 2020) (Oala et al. 2020) and approaches that use additional components in order to give an uncertainty estimate on the prediction of a network (Raghu et al. 2019) (Ramalho and Miranda 2020) (Oberdiek, Rottmann, and Gottschalk 2018) (Lee and AlRegib 2020) .Single deterministic methods are computationally efficient in training and evaluation.But this method rely on a single opinion and can therefore become very sensitive to the underlying network architecture, training procedure, and training data.

Bayesian methods cover all kinds of stochastic DNNs, i.e. DNNs where two forward passes of the same sample generally lead to different results. Bayesian Neural Networks (BNNs) (Denker et al. 1987) (Tishby, Levin, and Solla 1989) (Buntine 1991) have the ability to combine the scalability, expressiveness, and predictive performance of neural networks with the Bayesian learning as opposed to learning via the maximum likelihood principles.

Ensemble methods combine the predictions of several different deterministic networks at inference. (Lakshminarayanan, Pritzel, and Blundell 2017) are often referenced as a base work on uncertainty estimations derived from ensembles of neural networks. Ensemble methods are very easy to apply since no complex implementation or major modification of the standard deterministic model have to be realized. Furthermore, ensemble members are trained independently from each other, which makes the training easily parallelizable. But this method was limited by the computation power or memory , since the application is time-critical, or very large networks with high inference time are included (Malinin, Mlodozeniec, and Gales 2019).

Also, there exists some ways to evaluate uncertainty in regression tasks. For data uncertainty, regression tasks only predict a pointwise estimation without any hint of data uncertainty. A common approach to overcome this is to let the network predict the parameters of a probability distribution, for example, a mean vector and a standard deviation for a normally distributed uncertainty (Lakshminarayanan, Pritzel, and Blundell 2017) (Kendall and Gal 2017). For model uncertainty, The most common measures for this are mutual information (MI) and the expected Kullback–Leibler Divergence (EKL).

Proposed Solution

Backbone Network

Architecture and Loss Function Design The backbone network adopts a one-dimensional version of the ResNet18 architecture proposed in (He et al. 2016). Input dimensions of N × 6, representing N IMU samples in the gravity-aligned frame, lead to the output of two 3D vectors: displacement estimates $\hat{d}$ and their uncertainties u, with u parameterizing the covariance's diagonal entries. These vectors incorporate independent fully-connected blocks, extending the ResNet architecture.

Two distinct loss functions, Mean Square Error (MSE) and Gaussian Maximum Likelihood(GML), are employed during training.

Data Source We utilize the TLIO paper's dataset(Liu et al. 2020), featuring an IMU (Bosch BMI055) mounted on a headset rigidly attached to cameras. The dataset comprises 400 sequences, totaling 60 hours of pedestrian data, capturing diverse activities. Position estimates, obtained at 1000 Hz using a visual-inertial filter(Mourikis and Roumeliotis 2007), serve as supervision data for training.

IMU Model The IMU sensor directly measures non-gravitational acceleration a and angular velocity ω in the IMU frame. Measurements are subject to noise $n_{g;a}$ and bias $b_{g:a}$, modeled as:

$$\omega = \omega_{true} + b_g + n_g$$
$$a = a_{true} + b_a + n_a$$

Here, n_g and n_a follow zero-centered Gaussian distributions, and biases evolve as a random walk process over the IMU sampling period δt with parameters η_{gd} and η_{ad}.

Auxiliary Network

In this study, we introduce an auxiliary neural network, referred to as the NET2 model, designed to understand the differences between training and actual motion features. The primary goal is to adjust the parameters of the primary NET1 model to improve overall performance. Below are the detailed steps for training and adjusting the auxiliary NET2 model. Fig 1 shows the architecture.

NET2 Model Architecture The auxiliary NET2 model adopts a multilayer perceptron (MLP) structure, aiming to comprehend the disparities between training and actual motion features. The model comprises an input layer, hidden layers, and an output layer. The input layer receives the previous second's motion features (d) and uncertainty assessment (u) from the primary NET1 model.

Learning Basic Weights w_b Firstly, the auxiliary NET2 model undergoes training using the outputs of the NET1 model (motion features and uncertainty assessment) to learn basic weights (w_b). These weights capture the error patterns of the general model, intending to capture features during the training process.

Input Data: Utilize the outputs of the NET1 model as input data, including motion features (d) and uncertainty assessment (u).

Loss Function: Define a loss function that measures the difference between the predicted values of NET2 (d_e) and the actual error (difference between GT and d).

$$loss = \frac{(d_e - (GT - d))^2}{log(\Delta d_t)}$$

Optimization Algorithm: Employ an optimization algorithm, such as stochastic gradient descent (SGD) or Adam, to learn the basic weights (w_b).

In summary, the learning of basic weights (w_b) is a fundamental step in the training process, providing the auxiliary NET2 model with the foundation to comprehend general motion characteristics and uncertainties. The thoughtful design of the loss function and the application of optimization algorithms contribute to the effective acquisition of these basic weights, setting the stage for subsequent fine-tuning and adaptation to real-world motion features.

Weight Adjustment during Fine-tuning (w_c) After the initial phase of learning basic weights (w_b), the auxiliary NET2 model undergoes a crucial fine-tuning process to further refine the weights denoted as w_c. This stage is pivotal in adapting the NET2 model to real-world motion features, with a specific focus on minimizing the disparities between

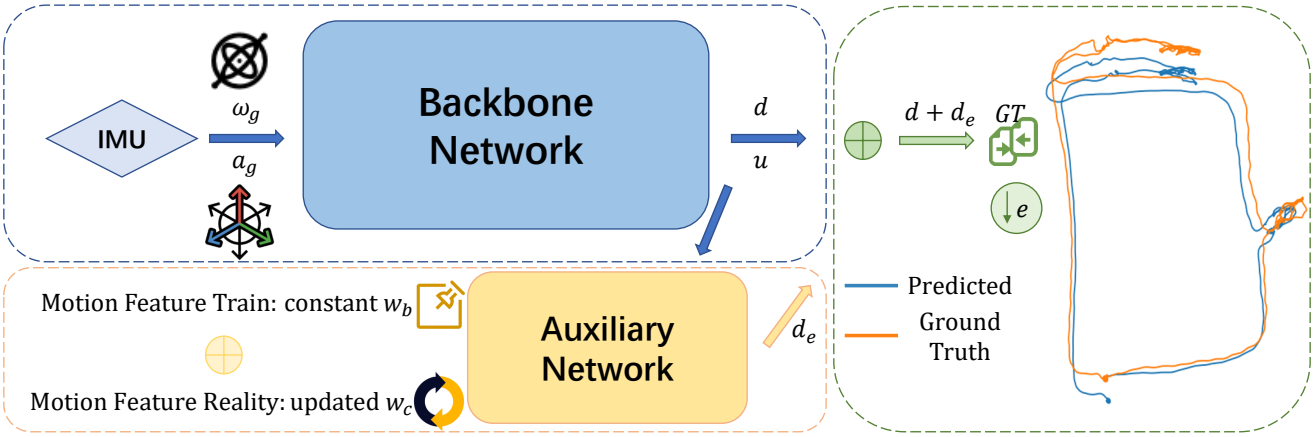

Figure 1: RLIO architecture.

the NET1 model predictions and the actual errors observed during system operation.

Similar to the basic weight learning phase, the input data for the fine-tuning process comprises motion features (d) and uncertainty assessments (u) derived from the NET1 model's outputs. This ensures that the fine-tuning is informed by the motion characteristics observed during the training phase.

The loss function used during fine-tuning is designed to measure the disparity between the predicted values of NET2 (d_e) and the actual error, represented by the difference between the ground truth (GT) and the motion features (d). The formulation of this loss function ensures that the fine-tuning process actively corrects any deviations from the actual motion patterns.

In summary, the fine-tuning process with the adjustment of weights (w_c) is a crucial step in the training pipeline. It enables the auxiliary NET2 model to adapt specifically to actual motion features, contributing to the overall improvement of the system's predictive accuracy and robustness.

Data Distribution Shift The concept of Data Distribution Shift plays a pivotal role in the training and adjustment processes of the auxiliary NET2 model. Through the iterative phases of learning and adapting weights, the model aims to grasp and comprehend the distinctions between the training dataset and the actual motion features encountered during real-world scenarios. A nuanced exploration of this phenomenon sheds light on the model's adaptability and its ability to generalize effectively to real-world motion dynamics.

Data Distribution Shift refers to the inevitable differences between the data distribution encountered during training and the distribution observed in real-world scenarios. In the context of the auxiliary NET2 model, these disparities may arise from variations in motion patterns, environmental conditions, or other factors not fully encapsulated in the training dataset.

To specifically address the Data Distribution Shift, the fine-tuning process becomes particularly relevant. During fine-tuning, the model refines its understanding of actual motion features, further narrowing the gap between the learned representations and the real-world distributions encountered during system operation.

In summary, the understanding and management of Data Distribution Shift are integral components of the auxiliary NET2 model's training process. The model's adaptability, gained through continuous learning and fine-tuning, positions it to effectively generalize and make accurate predictions in real-world scenarios with varying data distributions.

The above outlines the auxiliary NET2 model's training and adjustment process, providing details on its architecture, training steps, and how it adapts to actual motion features to enhance the overall system's performance.

Benefits of Adding the Auxiliary Network

In the TLIO framework, the localized gravity-aligned frame defines measurements to decouple global yaw information from relative state measurements. These techniques compensate for errors and reduce drift, successfully enhancing the network's robustness to sensor biases and rotation inaccuracies. Building upon the RESNET backbone used in TLIO, the introduction of the Auxiliary Network brings several advantages:

Improved Robustness to Sensor Variabilities The Auxiliary Network, trained in conjunction with the Backbone Network, facilitates the disentanglement of global and local motion characteristics. This aids in enhancing the network's resilience to sensor biases and inaccuracies in rotational measurements.

Guiding Backbone Network Based on Motion Characteristics The Auxiliary Network serves as a guiding mechanism for the Backbone Network, enabling it to perceive differences between training and real-world motion characteristics. By iteratively optimizing the results in the Backbone Network through the loss function, the Auxiliary Network contributes to refining the main branch.

Feature Alignment with Ground Truth Through the additional task handled by the Auxiliary Network, the Back-

bone Network gains the capability to exhibit a feature tendency toward the ground truth (GT). This ability is crucial for aligning the predicted results with the actual motion characteristics.

Minimal Computational Overhead The simplicity of the Auxiliary Network design ensures that it does not introduce excessive computational overhead. By using d and u as inputs and directly adding its output to the Backbone Network's output, the Auxiliary Network offers a straightforward means to capture motion characteristic differences.

Enhanced Prediction Accuracy The output of the Auxiliary Network, integrated into the final prediction of the Backbone Network, contributes to refining the overall prediction accuracy. This two-branch architecture helps in capturing and incorporating subtle motion pattern nuances.

In summary, the inclusion of the Auxiliary Network complements the TLIO RESNET architecture, providing a mechanism to enhance robustness, guide the backbone's understanding of motion characteristics, align features with ground truth, and improve overall prediction accuracy without introducing significant computational complexity.

Experiment

Dataset

The dataset we use comes from the TLIO paper dataset. The full dataset contains more than 400 sequences totaling 60 hours of pedestrian data that pictures a variety of activities including walking, standing still, organizing the kitchen, playing pool, going up and down the stairs etc. It was captured with multiple different physical devices by more than 5 people to depict a wide range of individual motion patterns and IMU systematic errors. A state-of-the-art visual-inertial filter based on provides position estimates at 1000 Hz on the entire dataset. We use these results both as supervision data in the training set and as ground truth in the test set. The dataset is split into 80% training, 10% validation and 10% test subsets randomly.

Experiment detail

For network training, we use an overlapping sliding window on each sequence to collect input samples. Each window contains N IMU samples of total size N×6. In our final system we choose N=200 for 200 Hz IMU data. We want the network to capture a motion model with respect to the gravity-aligned IMU frame, therefore the IMU samples in each window are rotated from the IMU frame to a gravity-aligned frame built from the orientation at the beginning of the window. We use visual-inertial ground-truth rotation for that purpose. The supervision data for the network output is computed as the difference of the ground-truth position between two instants expressed in the same headset-centered, gravity-aligned frame.

During training, because we assume the headset can be worn at an arbitrary heading angle with respect to the walking direction. In our final estimator, the network is fed with potentially inaccurate input from the filter, especially at the initialization stage. We simulate this at training time by random perturbations on the sensor bias and the gravity direction to reduce network sensitivity to these input errors. To simulate bias, we generate additive bias vectors with each component independently sampled from uniform distribution in $[-0.2, 0.2]m/s^2$ or $[-0.05, 0.05]rad/s$ for each input sample. Gravity direction is perturbed by rotating those samples along a random horizontal rotation axis with magnitude sampled from $[0, 5^o]$.

Optimization is done through the Adam optimizer. We used an initial learning rate of 0.0001, zero weight decay, and dropouts with a probability of 0.5 for the fully connected layers. The GPU we use for training is NVIDIA RTX 2080, and the system environment is Ubuntu.

Result

We evaluate our system on the testing segmentation of the dataset. Each of these 37 trajectories contains 3 to 7 minutes of human activity. Our proposed RLIO is based on an improved TLIO model, so we will compare the results of RLIO with those of TLIO to demonstrate that our improved method has improved the results. To evaluate our system performance, for each dataset of length n, calculate the root mean square error (RMSE) between these value sets. In order to present the output results of our system more intuitively, we plot the trajectory between the predicted and true values.

We first compared the performance of models with different parameters and scenarios vertically, and found that when using the absolute values of predicted and true values in loss and without real-time tuning, the RMSE was the smallest and the model performance was the best.

RPE refers to Relative Pose Error, which is used to evaluate the error between the estimated pose or position measured in robot or camera positioning tasks and the true value.In general, for relative pose error (RPE), the lower the error value, the better. In other words, the smaller the RPE, the closer the estimated pose or position is to the true value, which means the better the performance of the system. Therefore, when using root mean square error (RMSE) as a metric, smaller RMSE values typically correspond to more accurate relative pose estimation.We can use the root mean square error to calculate this error and obtain a population value.

Taking a simple trajectory as an example, we first compare and select different losses to choose the optimal model structure. The results obtained from complex trajectories remain consistent. Fig 2 shows the results of the RLIO model when the loss functions are MSE and GML.The RMSE of the model is 1.305 and 1.775, respectively. Explanation: When loss takes MSE is reached, the model performs the best.

Then, with a loss of MSE, we compare the results obtained by using a simple trajectory to compare the non real-time tuning and real-time tuning of NET2. Fig 3 shows the results of the RLIO model for NET2 without real-time tuning and for real-time tuning.The RMSE of the model is 1.305 and 1.631, respectively. So the model that does not

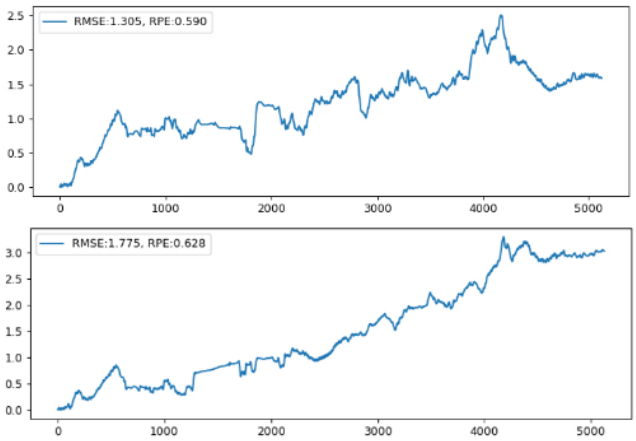

Figure 2: For MSE (top) and GML (bottom), the results of RLIO.

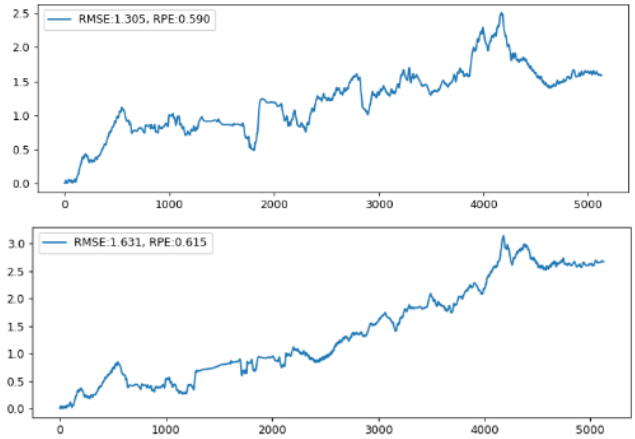

Figure 3: The results of RLIO for NET2 without real-time parameter tuning (top) and real-time parameter tuning (bottom).

perform real-time parameter tuning on NET2 performs the best.

Then, we compared the performance of the models horizontally, and compared the results with TLIO in simple trajectories and complex trajectories. The results (Fig 4 and 5) showed that our RLIO model was superior to the TLIO model, which proves that the network structure we added is effective in improving prediction performance.

In summary, our proposed RLIO model performs better than the TLIO model, and the best performance is achieved when the loss is MSE, without real-time tuning of NET2.

Conclusion

In conclusion, the study focused on addressing uncertainties associated with IMU-based neural networks and enhancing IMU-based navigation systems. The research aimed to analyze the sources of uncertainty in IMU neural networks and experiment with state-of-the-art IMU-based navigation

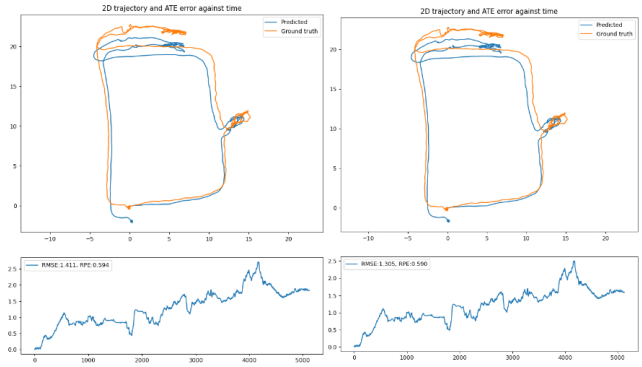

Figure 4: For simple trajectories, the results of TLIO (left) and RLIO (right).

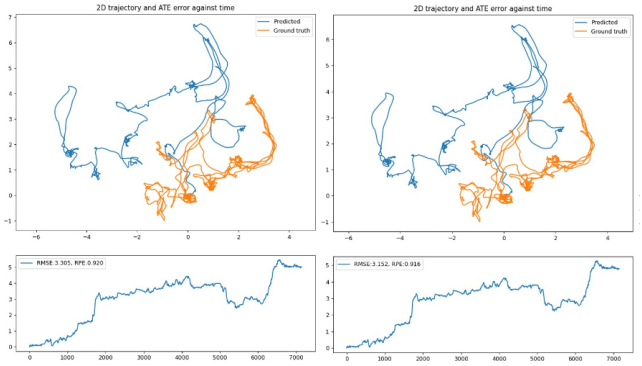

Figure 5: For complex trajectories, the results of TLIO (left) and RLIO (right).

methods such as RoNIN and TLIO. The primary goal was to develop an improved approach that reduces uncertainties and drift in IMU-based inertial navigation, utilizing neural networks and advanced data processing techniques.

The study's proposed solution involved the implementation of a backbone network architecture and an auxiliary network, referred to as the NET2 model, to understand the differences between training and actual motion features. The inclusion of the Auxiliary Network complemented the TLIO RESNET architecture, providing a mechanism to enhance robustness, guide the backbone's understanding of motion characteristics, align features with ground truth, and improve overall prediction accuracy without introducing significant computational complexity.

The experiments conducted on the dataset from the TLIO paper demonstrated that the proposed RLIO model outperformed the TLIO model, particularly when employing the Mean Square Error (MSE) loss function and without real-time tuning of NET2. The results showcased the improved performance of the RLIO model in comparison to TLIO, validating the effectiveness of the added network structure in enhancing prediction accuracy and reducing uncertainties associated with IMU-based navigation systems.

References

Buntine, W. L. 1991. Bayesian backpropagation. *Complex systems*, 5: 603–643.

Chen, C.; and Pan, X. 2023. Deep Learning for Inertial Positioning: A Survey.

Denker, J.; Schwartz, D.; Wittner, B.; Solla, S.; Howard, R.; Jackel, L.; and Hopfield, J. 1987. Large automatic learning, rule extraction, and generalization. *Complex systems*, 1(5): 877–922.

Harle, R. 2013. A Survey of Indoor Inertial Positioning Systems for Pedestrians. *Communications Surveys & Tutorials, IEEE*, 15: 1281–1293.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Huang, F.; Wang, Z.; Xing, L.; and Gao, C. 2022. A MEMS IMU Gyroscope Calibration Method Based on Deep Learning. *IEEE Transactions on Instrumentation and Measurement*, 71: 1–9.

Kendall, A.; and Gal, Y. 2017. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30.

Lakshminarayanan, B.; Pritzel, A.; and Blundell, C. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30.

Lee, J.; and AlRegib, G. 2020. Gradients as a measure of uncertainty in neural networks. In *2020 IEEE International Conference on Image Processing (ICIP)*, 2416–2420. IEEE.

Li, R.; Fu, C.; Yi, W.; and Yi, X. 2022. Calib-Net: Calibrating the Low-Cost IMU via Deep Convolutional Neural Network. *Frontiers in Robotics and AI*, 8.

Liu, W.; Caruso, D.; Ilg, E.; Dong, J.; Mourikis, A. I.; Daniilidis, K.; Kumar, V.; and Engel, J. 2020. TLIO: Tight Learned Inertial Odometry. *IEEE Robotics and Automation Letters*, 5(4): 5653–5660.

Malinin, A.; and Gales, M. 2018. Predictive uncertainty estimation via prior networks. *Advances in neural information processing systems*, 31.

Malinin, A.; Mlodozeniec, B.; and Gales, M. 2019. Ensemble distribution distillation. *arXiv preprint arXiv:1905.00076*.

Mourikis, A. I.; and Roumeliotis, S. I. 2007. A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 3565–3572.

Możejko, M.; Susik, M.; and Karczewski, R. 2018. Inhibited softmax for uncertainty estimation in neural networks. *arXiv preprint arXiv:1810.01861*.

Nandy, J.; Hsu, W.; and Lee, M. L. 2020. Towards maximizing the representation gap between in-domain & out-of-distribution examples. *Advances in Neural Information Processing Systems*, 33: 9239–9250.

Oala, L.; Heiß, C.; Macdonald, J.; März, M.; Samek, W.; and Kutyniok, G. 2020. Interval neural networks: Uncertainty scores. *arXiv preprint arXiv:2003.11566*.

Oberdiek, P.; Rottmann, M.; and Gottschalk, H. 2018. Classification uncertainty of deep neural networks based on gradient information. In *Artificial Neural Networks in Pattern Recognition: 8th IAPR TC3 Workshop, ANNPR 2018, Siena, Italy, September 19–21, 2018, Proceedings 8*, 113–125. Springer.

Raghu, M.; Blumer, K.; Sayres, R.; Obermeyer, Z.; Kleinberg, B.; Mullainathan, S.; and Kleinberg, J. 2019. Direct uncertainty prediction for medical second opinions. In *International Conference on Machine Learning*, 5281–5290. PMLR.

Ramalho, T.; and Miranda, M. 2020. Density estimation in representation space to predict model uncertainty. In *Engineering Dependable and Secure Machine Learning Systems: Third International Workshop, EDSMLS 2020, New York City, NY, USA, February 7, 2020, Revised Selected Papers 3*, 84–96. Springer.

Sensoy, M.; Kaplan, L.; and Kandemir, M. 2018. Evidential deep learning to quantify classification uncertainty. *Advances in neural information processing systems*, 31.

Tishby; Levin; and Solla. 1989. Consistent inference of probabilities in layered networks: predictions and generalizations. In *International 1989 joint conference on neural networks*, 403–409. IEEE.