Low-Power Localization Systems with Hardware-Efficient Deep Neural Networks

by

Yu Chen

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical and Computer Engineering)
in The University of Michigan
2023

Yu Chen

unchenyu@umich.edu

ORCID iD: 0000-0002-3008-9208

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to everyone who has helped me in the completion of this dissertation. First and foremost, I would like to thank my advisor and committee chair, Professor Hun-Seok Kim, for his guidance and support throughout my Ph.D. journey. We first met when I was pursuing my master's degree, and he introduced me to research in this field. He has been an exceptional mentor with great patience and dedication to provide assistance whenever I encountered difficulties in my research. I am also grateful to the members of my dissertation committee, Professor Maani Ghaffari Jadidi, Professor Andrew Owens, and Professor Dennis Sylvester, for their insightful comments and suggestions that helped me to improve the quality of my dissertation.

I would like to thank all the talented lab members: Mingyu Yang, Pierre Abillama, Bowen Liu, Ziyun Li, Minchang Cho, Seokhyeon Jeong, Changwoo Lee, and Sara Shoouri, for all the successful collaborations and valuable discussions. I have learned a lot from you, and I feel so lucky to be part of such a supportive and productive research team.

Lastly, I want to express my gratitude to my parents and friends for their unwavering support and encouragement throughout this long journey. Their love and support have been my source of strength during all the challenging times.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Localization systems solve the problem of identifying the location of the agent or surrounding objects with the information gathered from various sensors. It enables a wide range of practical applications, such as autonomous navigation, self-driving cars, virtual reality, augmented reality, and enhanced surveillance. In recent years, deep neural networks have achieved great success in various computer vision and machine learning tasks, including more accurate localization systems with extensive computation complexity and power consumption. However, deploying such systems on energy-constrained mobile Internet-of-Things (IoT) platforms remains a big challenge due to the contradiction between system performance and power consumption. This thesis presents several practical approaches to develop energy-efficient localization systems for real-world applications. First, a real-time visual based simultaneous localization and mapping system is investigated and optimized for hardware implementation, which is ported on a low-power, application specific integrated circuit accelerator. The second work focuses on reducing the complexity of deep learning based visual-inertial odometry systems by finding the most efficient network architecture through neural architecture search and adaptively disabling visual sensor modality on the fly. The third work proposes an accurate learning based end-to-end audio source separation and localization framework with only low-power microphone sensor array, taking advantage of self-supervised learning and adversarial learning techniques. Finally, a new hardware-efficient heterogeneous transform-domain neural network is introduced to reduce computation complexity by replacing convolution operations with element-wise multiplications, learning sparse-orthogonal weights in heterogeneous transform domains, and non-uniform quantization with canonical-signed-digit representation. These works explore four different yet effective ways to balance the system performance and power consumption for mobile IoT platforms, namely reducing deep neural network complexity, adaptively selecting and fusing sensor modalities, employing lower power sensors, and developing hardware-efficient systems for specialized accelerators.

CHAPTER I

Introduction

1.1 Localization Systems and Applications

Localization systems rely on various kinds of sensors, such as camera, lidar, radar, inertial measurement unit (IMU), sonar, and microphone array, to obtain environmental information. These systems continuously estimate the position information of the agent or surrounding objects by analyzing the collected sensor data. Localization is an important part of various applications, including self-driving cars, autonomous navigation, virtual reality (VR) and augmented reality (AR). In recent years, there has been a significant focus on developing reliable and fast localization systems to advance the development of truly autonomous and intelligent systems.

An autonomous and intelligent system demands the localization subsystem to perform two main tasks, estimating the agent's position in an unknown environment, and detecting and localizing other stationary or moving objects around the agent. Figure 1.1 provides the overview of a typical perception and localization system in an autonomous car platform [158], which processes the acquired raw sensor data to determine its own location and ego-motion, the number of surrounding objects, as well as their locations and motions. Typically, the two tasks are accomplished separately with different algorithms and different sets of sensors.

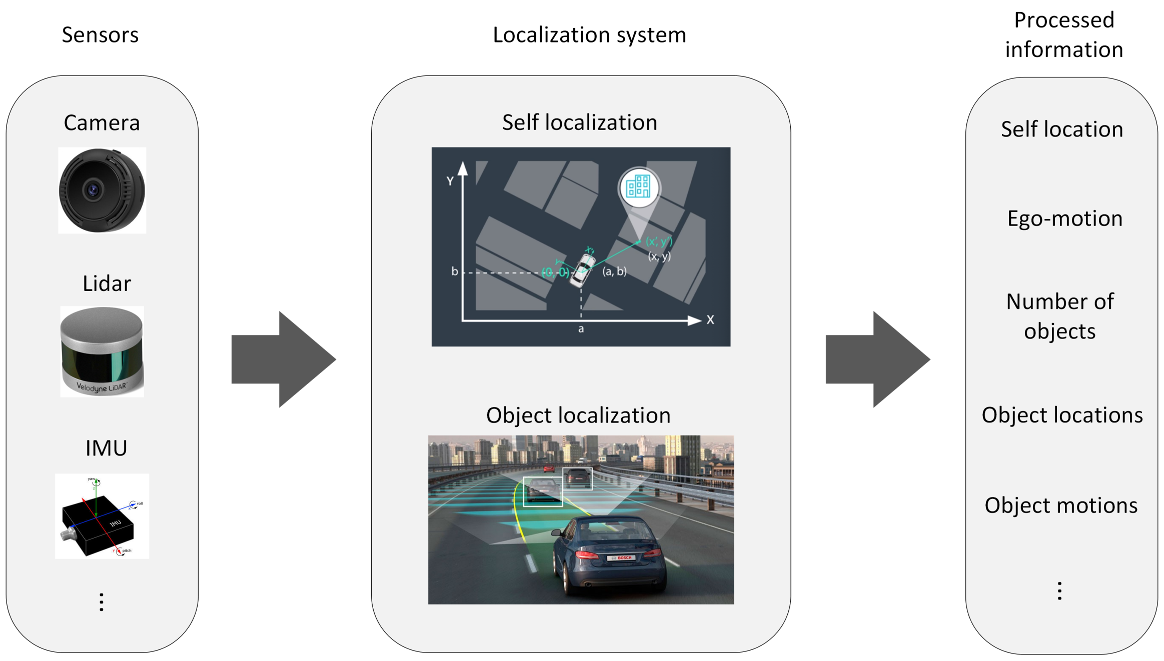


Figure 1.1: An overview of a localization system for autonomous vehicle that localizes the car itself as well as other cars or objects around.

1.1.1 Self-Localization

Localizing the agent itself in an unknown or partially known environment is the most important step before performing autonomous navigation, path planning, and other complex tasks. While simple odometry can estimate the position change over time using data collected from various motion sensors, such as wheel encoder and IMU, measurement errors accumulate and cause the estimated position to drift. In recent years, visual odometry (VO) [93] has emerged as a promising alternative to simple odometry due to its low cost and ability to capture useful information from the images. And it is proven to outperform simple odometry in most cases [4].

When building the map of the environment is required at the same time while localizing the agent, the problem is often referred to as simultaneous localization and mapping (SLAM) [6, 33] (Figure 1.2). SLAM has been extensively studied since the mid-1980s [115] and plays an important role in many real-world applications, includ-

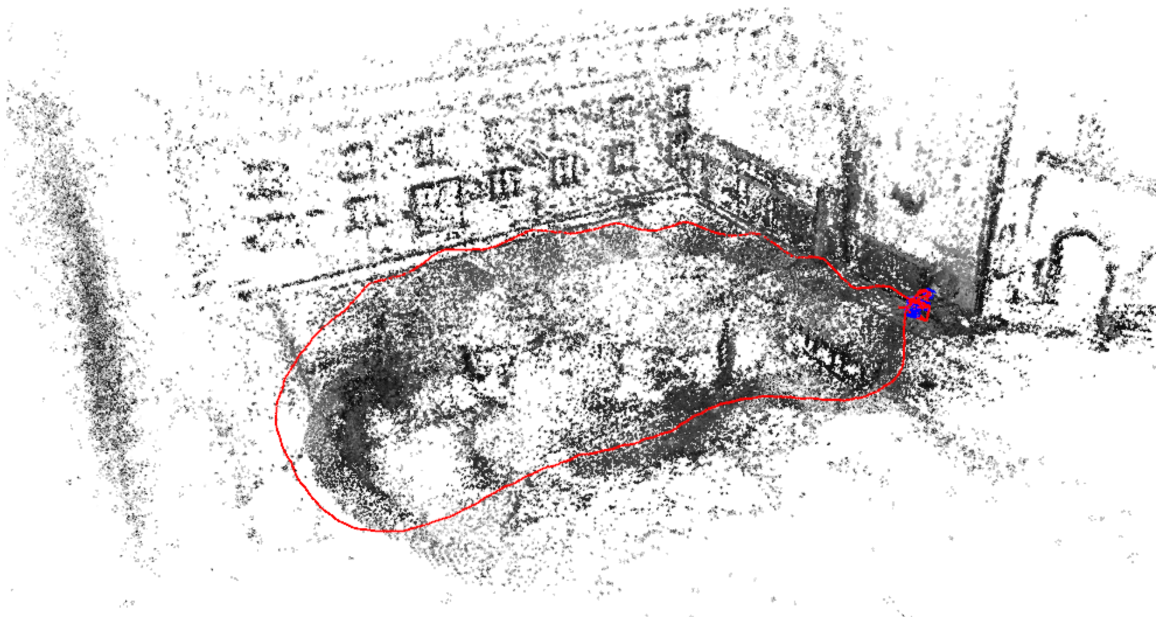


Figure 1.2: The estimated agent trajectory (red track) and sparse map (black point cloud) from SLAM system LDSO [40].

ing self-driving cars, unmanned aerial vehicles (UAVs), and autonomous underwater vehicles. Modern SLAM systems achieve relatively good performance thanks to the global optimization and loop closure [3]. SLAM usually employs multiple sensors of different types, such as camera, lidar, radar, and IMU, leading to different research directions with different sets of sensors.

With the rapid expansion of energy-constrained mobile devices and Internet-of-Things (IoT) platforms, developing hardware-efficient visual odometry and SLAM algorithms with low-power sensors and fewer sensor types has become one of the major research focuses in recent years.

1.1.2 Surrounding Object Localization

Localizing other objects within the same space is another critical task for autonomous and intelligent systems since the interaction between agent and environment requires a good knowledge of the surrounding incidents. Obtaining accurate

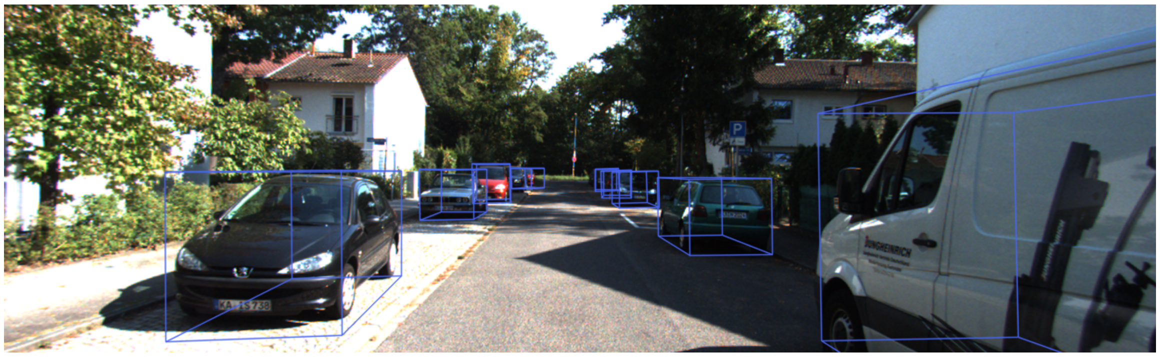


Figure 1.3: 3D surrounding object detection and localization (3D bounding box estimation) for autonomous driving cars, MVX-Net [114].

positions of other objects helps to avoid collision and entering danger zones, which is a precondition for safe and efficient robot navigation and movement. Figure 1.3 is an example of 3D surrounding object detection and localization for an autonomous driving car platform with camera and lidar sensors. Besides, surrounding object detection and localization is also useful in surveillance systems [28], such as 24-hour monitoring in industrial facilities and critical environments.

Depending on the characteristic of objects, it is possible to use different sensors with different algorithms to detect and localize them. For example, surrounding acoustic sources can be localized with low-power microphones or microphone arrays, while transparent objects can be efficiently detected and localized using radar instead of camera or lidar. In the realm of mobile IoT platforms, studying energy-efficient localization algorithms using passive and low-power sensors is a new trend.

1.2 Low Power Challenges

Recent advances of deep neural networks (DNN) in computer vision and machine learning enable localization systems to achieve excellent performance in various applications. However, most of the state-of-the-art DNNs focus on superior performance regardless of system energy consumption, making it impossible to be directly deployed on energy sensitive platforms, such as battery-powered VR headsets, micro UAVs and small mobile robots. The challenges of developing energy-efficient localization systems are summarized in this section.

1.2.1 Complexity of Deep Neural Network

Recent deep neural networks report superb performance in various computer vision and machine learning tasks. However, while the performance of DNNs evolved drastically, their complexity has also grown super-linearly. Take the image classification task and some famous convolutional neural network (CNN) structures as an example, the LeNet [74] for handwritten digit classification (MNIST) only requires 0.3 million (M) operations while the winners of the ImageNet classification challenge (ILSVRC) [104] need 0.7 giga (G) (AlexNet [68]) and even 13.5 giga (G) operations (VGG [113]) [125]. Later models such as ResNeXt-101 [149] and EfficientNet [126] achieve higher accuracy on ImageNet classification with more complex structures. Figure 1.4 shows the comparison of several famous CNN architectures regarding their computational complexity and classification accuracy on ImageNet.

Modern DNNs are unsuitable to be directly deployed for real-time applications on low-power and low-cost hardware platforms that cannot afford powerful and energy-intensive computing devices. Thus, reducing network complexity while maintaining network performance has become a main premise of the successful adoption of DNN based algorithms on power- and cost-constrained mobile devices. Although several different paths have been explored in the literature, the solutions are still far from perfect.

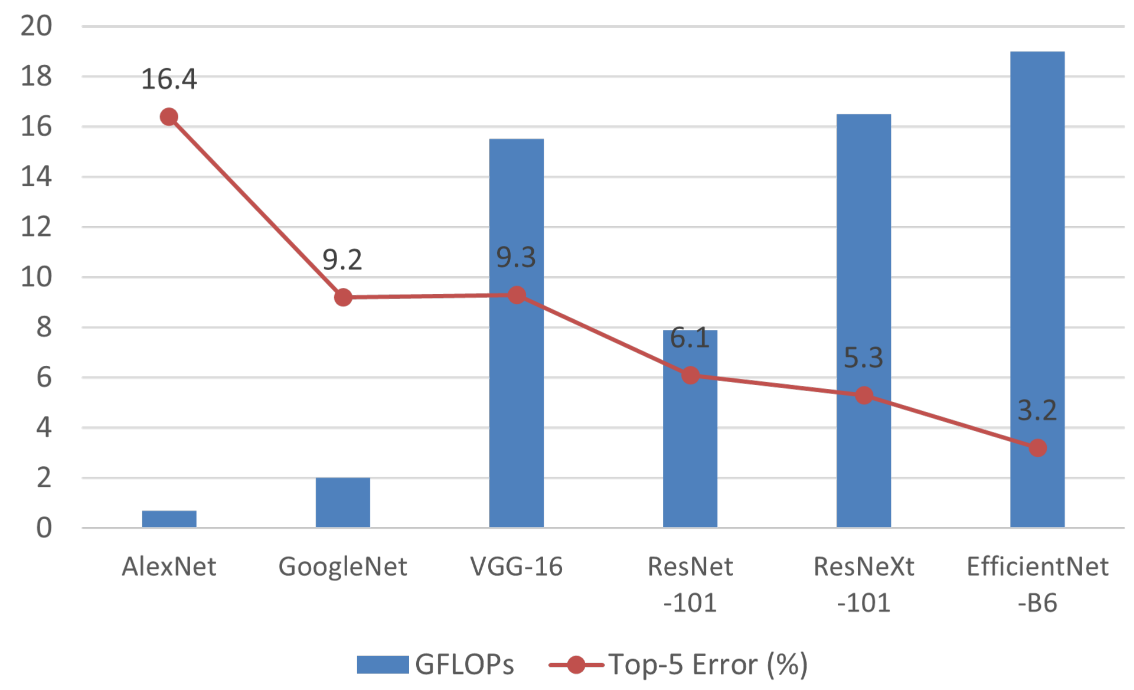


Figure 1.4: Comparison of the network complexity of some famous CNN architectures, as well as their classification accuracy on ImageNet classification challenge (ILSVRC).

1.2.2 Sensor Selection

Sensors are indispensable parts of the localization system, acting as the eyes to sense the surrounding environment and collect useful information. Various kinds of sensor modalities are employed to collect visual, sound, and other forms of data and information. And the system fuses and processes all the data to estimate the location of surrounding objects or itself.

State-of-the-art localization systems rely on different kinds of sensors to achieve the best performance. For example, autonomous vehicle platforms typically incorporate multiple sensors, including camera, lidar, radar, IMU and GPS, with camera, lidar, IMU and GPS primarily used for self-localization, and camera, lidar and radar for surrounding object detection and localization. However, mobile platforms cannot afford too many sensors due to space and power limitations. Moreover, different types

Sensor	Power consumption	Commercial product
Camera	120mW	OmniVision OVM7692
IMU	2.5mW	TDK ICM-20948
Microphone	0.12mW	Knowles SPW2430HR5H-B
Lidar	2W	Hitachi HLS-LFCD2

Table 1.1: Power consumption of different types of sensors.

of sensors consume different amounts of power, as listed in Table 1.1. Some sensors, such as lidar, may be too expensive for mobile platforms. Furthermore, raw sensor data processing and sensor fusion modules can consume more energy if multiple sensors are present, putting additional pressure on the energy budget of mobile IoT platforms. Therefore, developing more energy-efficient localization algorithms that utilize fewer sensors with lower power consumption is a major focus.

1.2.3 Sensor Fusion

Modern localization systems contain multiple sensors of different types to overcome the potential limitations of using a single sensor. Sensor fusion is the process of integrating signals from multiple sensors to reduce the amount of uncertainty that may be involved in a localization system [130], thus helping the system to utilize all the collected data effectively and produce more accurate results. Figure 1.5 is an illustration of sensor fusion in a typical autonomous driving car system. The sensor fusion module processes the IMU, camera, lidar and other sensor inputs for self-localization. And data collected from camera, lidar and radar are fused for surrounding object detection and localization.

Sensor fusion requires additional computation to process and fuse data from multiple sensors, which leads to increased system energy costs and brings high pressure on

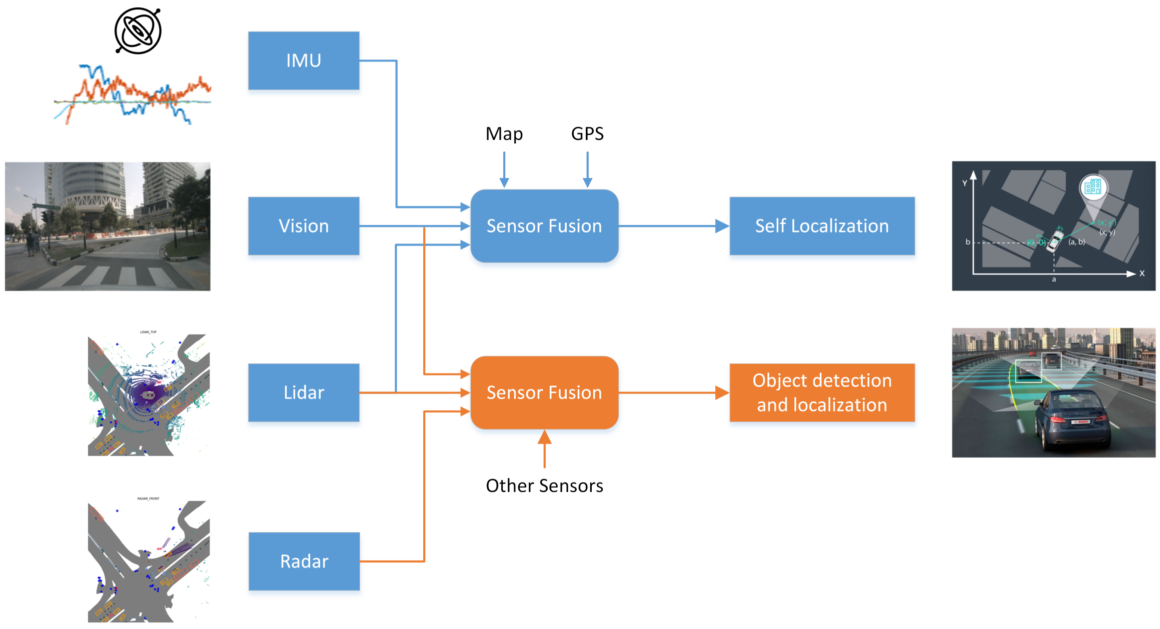


Figure 1.5: An illustration of sensor fusion in an autonomous driving car system for self-localization and surrounding object detection and localization.

energy-constrained mobile IoT platforms, especially when using more recent neural network based fusion methods [19, 114]. To reduce the computational complexity and lower the power consumption, several practical approaches have been proposed for energy-efficient sensor fusion [8, 44]. However, these approaches are often limited and lack robustness due to their reliance on static fusion algorithms with all available sensor modalities. As a result, there is growing interest within the research community in more flexible approaches like adaptive sensor modality selection and fusion.

1.2.4 Deployment on Hardware

Most localization algorithms are designed to run on general purpose processors like central processing units (CPUs) and graphics processing units (GPUs). Such computing platforms are too big and consume too much energy, making them unsuitable for mobile IoT applications. To address this issue, one practical approach is to design specialized hardware accelerators using application specific integrated circuits

(ASICs) [64] with much lower energy consumption. However, it is not feasible to run state-of-the-art localization systems directly on these specialized accelerators. Therefore, localization systems need to be optimized in terms of computational complexity and memory usage before deploying on dedicated hardware accelerators.

Although optimizing existing state-of-the-art algorithms by reducing the system's computational complexity can be beneficial, it may not always result in optimal solutions. For example, pruning neural network weights and activations [49, 50, 159] can significantly reduce the number of operations and memory usage, but the pruned sparse network still requires substantial hardware overhead to implement the sparse operations [48, 96, 161]. A more effective approach is to design new hardware-efficient systems through hardware-software co-design [30].

1.3 Dissertation Organization

This dissertation demonstrates four different yet effective approaches to build low-power localization systems with balanced system performance and power consumption, enabling the potential application to energy-constrained mobile IoT platforms. The proposed approaches focus on DNN complexity reduction for deep learning based systems, adaptive sensor modality selection and fusion, utilization of low-power sensors, and hardware-software system co-design for specialized accelerators. The detail of each work is presented in the following chapters.

In Chapter II, a real-time visual based SLAM system for six degrees of freedom (6DoF) ego-motion estimation is investigated. The algorithm is optimized for low-power mobile autonomous navigation and VR/AR applications. Visual features are extracted by a simple CNN feature descriptor with efficient parallel execution on specialized DNN hardware accelerators, which outperforms hand-crafted feature extraction algorithms like SIFT [84] and ORB [103]. An aggressive region-prediction based matching technique is applied to eliminate unnecessary feature matching computations in the tracking phase. A new keyframe decision scheme is developed to reject ineffective keyframes and 3D landmarks for global optimization, thus significantly reducing on-chip memory requirements. The system achieves good accuracy with much lower power consumption and smaller memory usage compared to the state-of-the-art SLAM algorithms, and it is already ported on a low-power, very large-scale integrated (VLSI) ASIC accelerator.

In Chapter III, an energy-efficient deep learning based visual-inertial odometry (VIO) system is proposed. First, the neural architecture search (NAS) technique is adopted to search for the most efficient VIO network architecture, targeting the lowest number of operations and lowest inference latency. The searched model achieves up to 97.4% complexity reduction with no performance degradation and allows the VIO system to run in real-time on a single laptop CPU core. Then, a policy network is learned to opportunistically disable the visual modality on the fly from the current motion state and IMU measurements. A Gumbel-Softmax trick is adopted to make the decision process differentiable for end-to-end system training. The learned strategy is interpretable, and it shows scenario-dependent decision patterns for adaptive complexity reduction. The proposed modality selection method achieves similar performance compared to the full modality baseline with up to 72.0% computational complexity reduction. Overall, the model complexity can be reduced up to 99.1% when combining the searched efficient VIO network and the dynamic visual modality selection technique.

In Chapter IV, an end-to-end deep learning framework is presented to separate and localize multiple audio sources at the same time with a low-power microphone sensor array. The proposed framework jointly estimates the separated sources and their time difference of arrival (TDOA) at different microphones, then obtains the direction-of-arrival (DOA) of each source. Source separation and TDOA estimation are jointly optimized by adding a similarity loss between the reconstructed mixture and the original mixed signal. In addition, a discriminator network is added during the training phase to further improve the separation quality. The system achieves state-of-the-art accuracy on source separation as well as DOA estimation.

In Chapter V, a new class of transform domain DNNs, the HTNN, is introduced, where convolution operations are replaced by element-wise multiplications in heterogeneous transform domains. The network computational complexity is reduced by learning sparse-orthogonal weights in heterogeneous transform domains and co-optimizing with a hardware-efficient accelerator architecture to minimize the overhead of handling the sparse weights. Furthermore, sparse-orthogonal weights are non-uniformly quantized with canonical-signed-digit (CSD) representations to substitute multiplications with simpler additions/subtractions. The proposed framework achieves up to 6.8$\times$ complexity reduction without compromising the DNN accuracy compared to equivalent CNNs that employ sparse (pruned) weights.

CHAPTER II

Real-Time Simultaneous Localization and Mapping for Energy-Constrained Mobile IoT Applications

2.1 Introduction

In recent years, simultaneous localization and mapping (SLAM) draws lots of attentions in computer vision and robotics communities due to its importance in various tasks. It has wide applications in autonomous navigation, self-driving cars, unmanned aerial vehicles, mobile robots, autonomous underwater vehicles and planetary rovers. SLAM solves the problem of building a consistent map of an unknown environment while simultaneously localizing the agent within this map. A wild variety of sensors are used in SLAM systems to catch different kinds of environmental information, including camera, inertial measurement unit (IMU), lidar and radar. Due to the low cost of the camera and rich information provided through images, visual based SLAM systems, where the main sensors are cameras, are in favor nowadays.

The simplest setup for visual SLAM involves only a single camera, which is referred to as monocular SLAM. Since the depth information cannot be directly inferred from just one camera, monocular SLAM often encounters lots of problems. First, monocular SLAM suffers from scale drift and may fail if the camera is doing pure rotations. Besides, monocular SLAM requires a careful and accurate initialization of the map to avoid system failure. These disadvantages can be easily overcome by using a stereo or an RGB-D camera, which leads to much more reliable visual SLAM solutions.

Most SLAM algorithms are designed to run on general purpose processors like CPUs and GPUs, which are usually too big and require too much energy for mobile IoT applications, such as battery-powered virtual reality (VR) headsets, micro unmanned aerial vehicles (UAVs), and small moving robots. For this kind of application, the power constraint of SLAM processing is only hundreds of micro watts, much less than the power consumption of the general purpose processors which is typically more than 10 watts. Such mobile IoT platforms are not powerful enough to run state-of-the-art SLAM algorithms in real time. To enable SLAM applications in such energy-constrained situations, a practical approach is to design specialized hardware accelerators using application specific integrated circuits (ASIC) [64] with lower energy consumption. Apparently, state-of-the-art SLAM systems cannot be directly deployed on this sort of specialized accelerators. Thus, it is important to investigate new SLAM algorithms that are more suitable for hardware implementation while still maintaining good system performance.

This work presents a visual SLAM system that is optimized for hardware VLSI implementation, using only a stereo or RGB-D camera. A simple convolutional neural network (CNN) is trained as the visual feature descriptor, which improves the feature matching accuracy and can be efficiently executed on a dedicated neural engine. The region-prediction based feature matching technique and a new keyframe decision scheme help to significantly reduce computation cost and on-chip memory requirement. Moreover, the proposed system has been ported on the first large-scale low-power SLAM ASIC accelerator [78].

2.2 Related Work

There are two mainstreams of state-of-the-art visual SLAM algorithms, namely the keypoint based SLAM and the direct SLAM, depending on whether the image pixels are directly used or not.

Keypoint based SLAM methods [29, 91] make use of the images indirectly. First, important visual features are extracted from the images, then these features are used for keypoint tracking and camera ego-motion estimation, and the map is built from the tracked sparse keypoints. The visual features can be either simple geometric features like corners and edges, or extracted by more sophisticated hand-crafted feature descriptors, such as SIFT [84], ORB [103] and FAST [102]. For example, the SOFT-SLAM [29] uses corner-like features while the ORB-SLAM2 [91] employs more complex ORB features to improve accuracy. Since only indirect and sparse feature information is used, the performance of keypoint based methods is greatly affected by the quality of extracted features and the accuracy of feature matching.

Direct SLAM algorithms [36, 40] propose to directly utilize the image pixel intensities for both tracking and mapping, rather than extracting intermediate visual features and keypoints. The camera poses are estimated by direct image alignment and the 3D environment is reconstructed with associated semi-dense depth maps. Direct methods can obtain a much denser map than keypoint based methods thanks to the use of more image areas and pixels. However, direct SLAM systems are usually computationally demanding and consume more power due to the need of processing large amounts of data. Hence, it is not a good choice for this work since the goal is to develop hardware-efficient SLAM systems for energy-constrained applications.

Recently, some hardware implementations of visual odometry / visual-inertial odometry (VO/VIO) [31, 58], which often serves as the front-end of visual SLAM

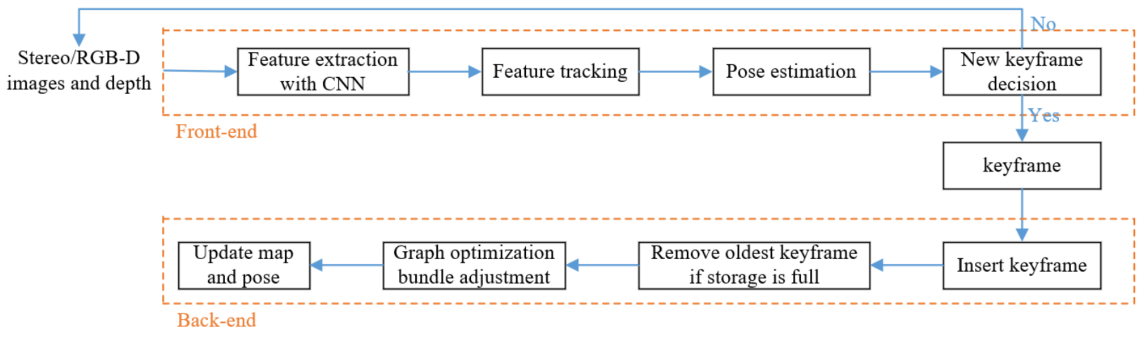

Stereo/RGB-D images and depth
Feature extraction with CNN
Feature tracking
Pose estimation
New keyframe decision
No
Yes
Front-end
keyframe
Insert keyframe
Remove oldest keyframe if storage is full
Graph optimization bundle adjustment
Update map and pose
Back-end

Figure 2.1: The proposed complete system with visual odometry front-end and graph optimization back-end.

systems, have been proposed, including a simple low-power implementation for augmented reality (AR) applications [58]. However, these implementations only target on the general purpose embedded hardware, which is less efficient than the specialized ones. Zhang et al. [160] attempt to design a VIO system for specialized hardware with desired resource-performance trade-off through hardware and algorithm co-design process. Inspired by their work, this work aims to design and implement a complete low-power SLAM system through a similar hardware and software co-design approach. The proposed low-power SLAM system can do long-term and globally consistent localization and mapping in real-time operation.

2.3 A SLAM System Optimized for Hardware

The proposed SLAM system takes the keypoint based approach and can be divided into two parts, the visual odometry front-end and the optimization back-end. The front-end consists of 4 blocks, i.e., feature extraction, feature tracking, pose estimation, and new keyframe decision. The back-end collects all keyframes and 3D landmarks and applies graph optimization to reduce the accumulated drift in the front-end estimation. The system inputs are images taken from stereo or RGB-D cameras where the depth information can be easily acquired through existing algo-

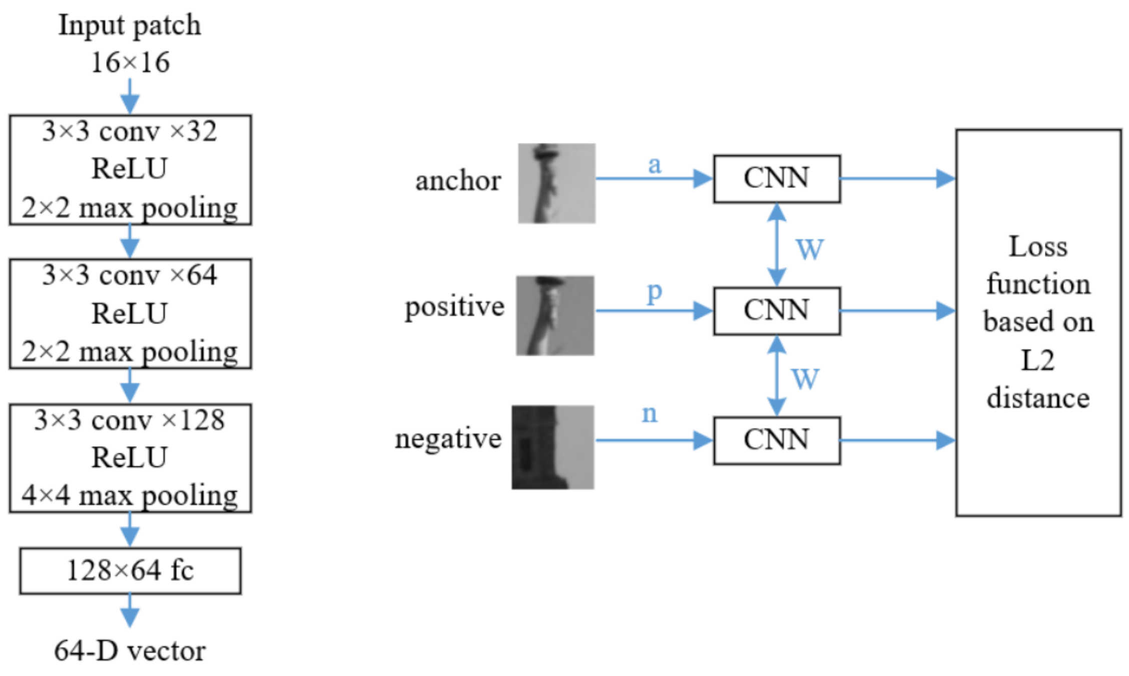


Figure 2.2: Left: CNN architecture of the proposed visual feature descriptor. Right: Training with the triplets, all three CNNs in the figure are identical and share the same weights.

rithms such as semi-global block matching [55]. Figure 2.1 is the diagram of the proposed system, the detail of each part is described in the rest of this section.

2.3.1 Visual Odometry Front-End

Recent works have demonstrated that deep learning based local visual feature descriptors can significantly improve the feature matching performance [9, 112]. In the proposed SLAM system, instead of using hand-crafted feature descriptors, a simple CNN is utilized to extract the visual features. It contains three convolutional layers, and each convolutional layer is followed by a ReLU layer and max pooling layer. Before the final output, there is a 128×64 fully connected layer to convert the feature vector into the desired dimension. Figure 2.2 left is the network structure of the proposed CNN based feature descriptor. Given the small image patch (16×16

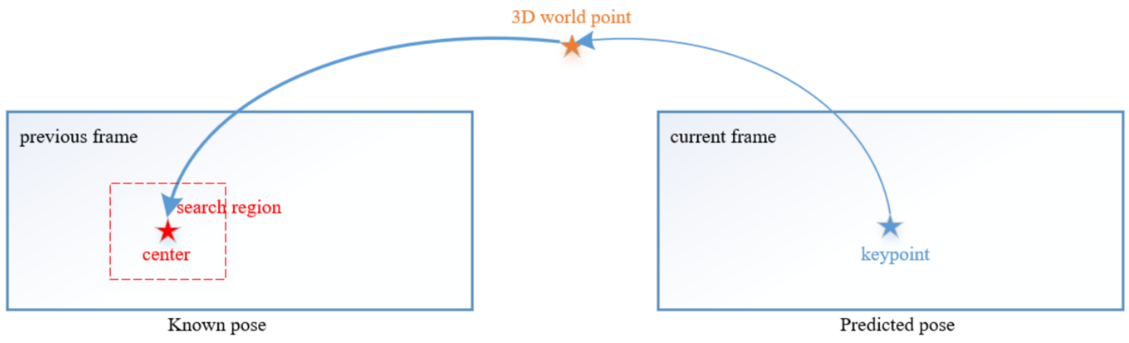


Figure 2.3: Region-prediction based matching technique. The search region in the previous frame (red square) is greatly reduced.

in pixels) around the keypoint obtained by difference of Gaussian (DoG), the CNN generates a 64-dimension vector as the feature descriptor. The network is trained with triplets described in [9] and is illustrated on the right of Figure 2.2. In each training step, three different training samples, the anchor patch, the positive patch (a different sample from the same class) and the negative patch (a sample from a different class) are passed through the network to generate three feature vectors. Then the L2 distance between the anchor vector and the negative vector is maximized, and the L2 distance between the anchor vector and the positive vector is minimized. Hence, the loss function can be defined as

$$L(\mathbf{A}, \mathbf{P}, \mathbf{N}) = \max(0, \mu + \|f(\mathbf{A}) - f(\mathbf{P})\|_2 - \|f(\mathbf{A}) - f(\mathbf{N})\|_2), \tag{2.1}$$

where $\mathbf{A}, \mathbf{P}, \mathbf{N}$ are anchor, positive and negative image patches, respectively, and μ is a margin parameter. Equation 2.1 is a convex approximation to the 0-1 ranking error loss, which measures the violation of the feature descriptor ranking order inside the triplets.

The feature tracking step finds the corresponding 2D image keypoints between the current frame and the previous frame, which are then used for estimating the relative camera pose between the two frames. Conventionally, after obtaining all the keypoints and their feature descriptors on the current frame, these keypoints are matched by calculating and minimizing the L2 distance of all the feature descriptors on the previous frame. However, this process wastes too much time and computation due to the redundant comparison of all the feature descriptor pairs. Instead, a region-prediction based matching technique that can significantly reduce the computation is introduced, which is shown in Figure 2.3. First, the current camera orientation $\hat{\mathbf{R}}_{\mathbf{c}}$ and translation $\hat{\mathbf{t}}_{\mathbf{c}}$ are predicted using the relative pose of the previous two frames, assuming the camera has no sudden movement between any two consecutive frames. Next, for each keypoint (x, y) on the current frame, its corresponding 3D world point (X, Y, Z) can be obtained by

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{R}}_{\mathbf{c}} & \hat{\mathbf{t}}_{\mathbf{c}} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f_x \\ 0 & 0 & f_x/b & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix}, \tag{2.2}$$

where d is the disparity that can be obtained from depth information, f_x, (c_x, c_y) and b are the focal length, the principal point in pixels and the baseline, all known from camera calibration. Finally, this 3D world point is reprojected back onto the previous frame to get the center of the search region (u, v):

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}\,[\mathbf{R}_{\mathbf{p}}\ \mathbf{t}_{\mathbf{p}}] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{2.3}$$

where $\mathbf{K}$ is the camera intrinsic matrix, $\mathbf{R}_{\mathbf{p}}$ is the previous orientation and $\mathbf{t}_{\mathbf{p}}$ is the previous translation. Finally, only feature descriptors in this search region are compared, which is a $n \times n$ square in pixels around the predicted center. Typi-

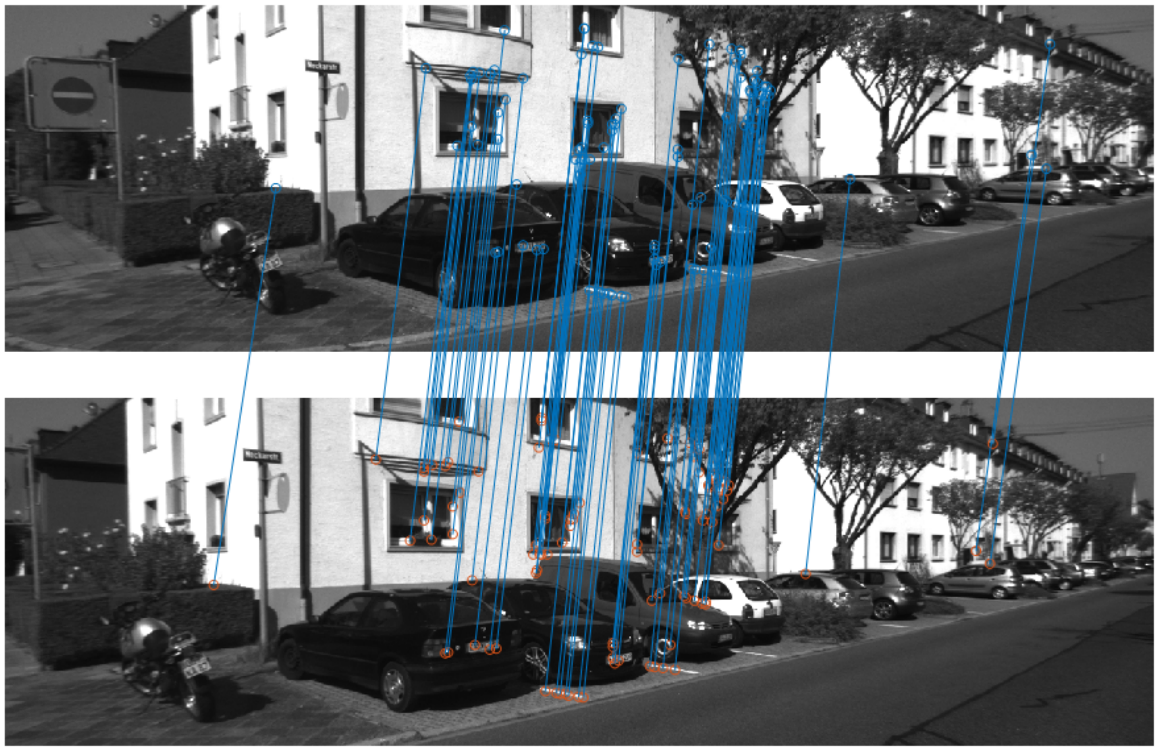


Figure 2.4: Example of the feature tracking and matching between two consecutive frames with proposed CNN feature descriptor and region-prediction based matching technique.

cally, thousands of keypoints are extracted on each frame, but the search region only contains dozens of them. By using this matching technique, the computation can be reduced by around 100X while maintaining the same matching accuracy. Figure 2.4 shows an example of the feature tracking and matching with the proposed CNN feature descriptor and region-prediction based matching technique.

The proposed SLAM system performs motion-only bundle adjustment (BA) to estimate the relative camera pose between two consecutive frames. Bundle adjustment [129] is originally employed as the last step of structure from motion problem [108, 134], which simultaneously refines the 3D coordinates describing the scene geometry and the parameters of the relative motion according to an optimization criterion involving the corresponding image projections of all keypoints. Motion-only BA optimizes the relative camera orientation $\mathbf{R}$ and translation $\mathbf{t}$ of the current frame by minimizing the total reprojection errors between the matched 3D world points $\mathbf{X_j}$ and the 2D image keypoints $\mathbf{x_j}$:

$$\{\mathbf{R}^*, \mathbf{t}^*\} = \arg\min_{\mathbf{R},\mathbf{t}} \sum_{j=1}^{M} \rho(\|\mathbf{K}(\mathbf{R}\mathbf{X_j} + \mathbf{t}) - \mathbf{x_j}\|_2^2), \tag{2.4}$$

where ρ is the robust Huber cost function [62] and $\mathbf{K}$ is the camera intrinsic matrix. To solve this nonlinear least squares optimization problem, Levenberg–Marquardt method [69, 83] is often employed. It is also noticeable that the accuracy of relative pose estimation is greatly affected by the matching quality of the 3D and 2D keypoint pairs used in motion-only BA. In all experiments, it is observed that doing motion-only BA twice, where bad matching pairs with large reprojection errors are eliminated during the first run, could improve the overall system performance, especially in an environment with lots of moving objects.

The last block of the visual odometry front-end, the new keyframe decision block, is the preparation step for the back-end optimization, which is critical for dealing with the accumulated drift and reducing the pose estimation errors. A good keyframe selection strategy can lead to better back-end optimization results, hence improving the overall SLAM system performance. An efficient keyframe decision and update scheme is developed to avoid storing less useful keyframes and landmarks when there is limited on-chip memory of the mobile platforms. At system startup, the first frame is set as a keyframe, its pose is set to the origin and an initial map is created. Then the second frame is also registered as a keyframe and all the 3D landmarks that can be tracked between the first and second keyframes are stored. During the system operation, it keeps tracking the keypoints between the current frame and the previous keyframe. Once the number of tracked keypoints is below 50% of the total keypoints in the previous keyframe, the current frame is selected as a new keyframe.

Besides, only landmarks that can be tracked in multiple keyframes are stored since only these landmarks are effective during graph optimization. Moreover, the current frame is registered as a keyframe if there is no keyframe in the past 10 frames. The proposed keyframe decision and update strategy saves the system's on-chip memory significantly, and enables the storage of more keyframes and associated landmarks for back-end optimization on a larger scale.

2.3.2 Graph Optimization Back-End

For a long time, the SLAM systems are optimized through filtering methods, such as the well-known extended Kalman filter [17] and its different variants [56, 124]. However, filtering methods are not the optimal back-end optimization strategies for visual SLAM algorithms. On the other hand, keyframe bundle adjustment is first introduced for visual SLAM back-end optimization by Klein et al. [66], and is proven to outperform filtering methods by Strasdat et al. [118]. Given a set of N keyframes and M associated 3D landmarks appeared in these keyframes, keyframe BA jointly optimizes all the keyframe poses (rotation $\mathbf{R_i}$ and translation $\mathbf{t_i}$) and 3D landmarks $\mathbf{X_j}$ by minimizing the total reprojection errors. This optimization problem can be expressed as:

$$\{\mathbf{R_i}^*, \mathbf{t_i}^*, \mathbf{X_j}^*\} = \arg\min_{\mathbf{R_i},\mathbf{t_i},\mathbf{X_j}} \sum_{i=1}^{N}\sum_{j=1}^{M} w_{ij}\|\mathbf{K}(\mathbf{R_i}\mathbf{X_j}+\mathbf{t_i})-\mathbf{x_{ij}}\|_2^2, \tag{2.5}$$

where w_{ij} is an indicator function on whether landmark j appears in the keyframe i, $\mathbf{K}$ is the known camera intrinsic matrix, and $\mathbf{x_{ij}}$ is the actual 2D projection from landmark j to keyframe i. Again, Levenberg–Marquardt method is employed to solve this nonlinear optimization problem.

For best system performance, all the keyframes and associated landmarks should be kept and used in the keyframe BA, which is also called the global BA. However, global BA is not practical in the proposed low-power SLAM system due to the limited storage availability. It is important to avoid using any off-chip dynamic random access memory (DRAM) since off-chip DRAM could cause much higher power consumption and chip cost. Hence, only a reasonable amount of keyframes and landmarks can be kept in the on-chip memory with the help of the proposed keyframe decision and update scheme. Every time before inserting a new keyframe and its associated landmarks, the system checks if there is enough space in the memory. If not, the oldest keyframes and their associated landmarks are removed to make room for the new ones. Typically, if 4 megabytes of on-chip memory is available, around 50 keyframes and 4000 landmarks can be stored, which ensures a quite large-scale keyframe BA in the back-end.

2.4 Experiments

First, this section shows the performance of the proposed CNN based visual feature descriptor. Then the entire low-power SLAM system is evaluated on one of the most influential SLAM benchmarks, the KITTI Visual Odometry / SLAM Evaluation dataset [41].

2.4.1 CNN Feature Descriptor

The proposed CNN feature descriptor is evaluated on one of the most popular benchmarks in the field of local descriptor matching, the Multi-view Stereo Correspondence dataset [12]. It consists of 64×64 grayscale image patches sampled from 3D reconstructions of the Statue of Liberty (New York), Notre Dame (Paris), and Half Dome in Yosemite (Yosemite). All image patches are divided into three subsets: Liberty, Yosemite and Notredame, with each containing more than 500k patch pairs. Image patches are downsampled to 16×16 in all experiments. Following the

Training subsets Testing subset		Lib & Yos Not	Lib & Not Yos	Yos & Not Lib	
Descriptor	dim				mean
SIFT [85]	128	22.53	27.29	29.84	26.55
ImageNet CNN [38]	128	9.64	30.22	14.26	18.04
DeepDesc [112]	128	4.54	16.19	8.82	9.85
Proposed CNN	64	10.59	18.70	18.02	15.77
Proposed CNN (8 bits)	64	10.76	18.92	18.15	15.94

Table 2.1: Results from the Multi-view Stereo Correspondence dataset. Numbers are reported in terms of FPR95. Lib: Liberty, Yos: Yosemite, Not: Notredame.

Descriptor	SIFT [120]	Proposed CNN (8 bits)	DeepDesc	ImageNet CNN
Parameters	N/A	92,448	280,096	3,722,592
Power	52.5 mW	150 mW	7.3 W	96.6 W

Table 2.2: Comparison of the network complexity and power consumption among different descriptors.

prior works [9, 12, 112], any of the two subsets are used for training and the rest for testing. The evaluation metric is the false positive rate at 95% true positive rate (FPR95) of all three training and testing combinations, as well as the mean across all combinations, same as many previous state-of-the-art works.

Table 2.1 summarizes the results of the proposed CNN descriptor with all three training and testing combinations, as well as its quantized version with 8-bit weights and activations. The quantized network achieves similar image patch matching performance as the original floating-point network while reducing the network computational complexity and power consumption by around 18× according to Horowitz [59]. The FPR95 values of SIFT [85] and some of the existing deep learning based feature descriptors [38, 112] are also reported in Table 2.1 for comparison. Besides, Table 2.2 lists the estimated power consumption of SIFT, the proposed CNN descriptor, DeepDesc and ImageNet CNN, and the network complexity (number of parameters) of the three learning based descriptors. Although the proposed method consumes 150 mW of power, which is higher than SIFT, it outperforms SIFT in all three cases with acceptable power consumption for mobile applications. Compared to other learning based feature descriptors, the proposed network performs slightly worse in some cases, but with much lower computation cost during the inference stage thanks to the simpler network structure and 8-bit weights & activations quantization. In addition, the feature vector dimension of the proposed CNN descriptor is smaller than all other feature descriptors (64D vs 128D), which could reduce the computation in the feature matching phase.

2.4.2 SLAM System Evaluation

The entire low-power SLAM system is evaluated on the popular KITTI Visual Odometry / SLAM Evaluation dataset [41]. It contains stereo video sequences recorded from a calibrated stereo camera on a driving car platform in real-world environments including urban, highway and other conditions. The stereo camera works at 10Hz with a resolution of around 1240×376 in pixels after rectification. There are 22 sequences in total, and 11 of them (sequences $00 - 10$) have ground truth trajectories. The proposed system is tested on sequences $00 - 10$ and uses only grayscale stereo images as input. Two different metrics are used for performance evaluation, the absolute translation root mean square error (RMSE) t_{rmse} proposed in [119], and the average relative translation error t_{re} and relative rotation error r_{re} over various subsequence path lengths in $100, 200, \ldots, 800$ meters as proposed in [41].

Detailed results on each sequence (00 to 10) of the proposed system, as well as two state-of-the-art stereo visual SLAM systems, ORB-SLAM2 [91] and Stereo LSD-SLAM [36], are reported in Table 2.3. The proposed system achieves relatively good pose estimation accuracy on sequences 03, 04, 06, 07 and 10, and acceptable

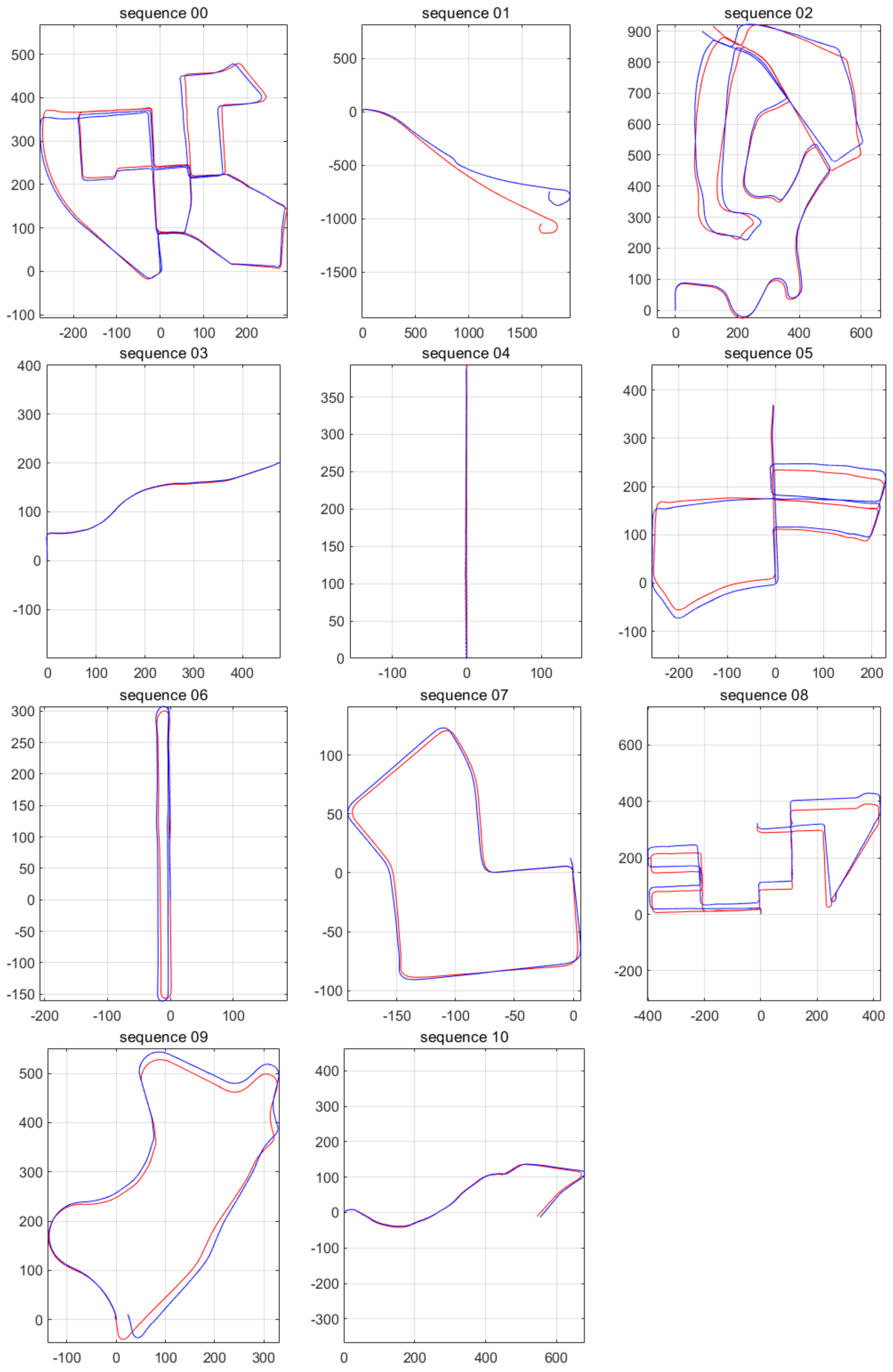


Figure 2.5: Estimated trajectories (blue) and ground truth trajectories (red) on KITTI dataset sequences 00 to 10.

Seq.	Proposed system			ORB-SLAM2 [91]			Stereo LSD-SLAM [36]		
	t_{re} (%)	r_{re} (°/100m)	t_{rmse} (m)	t_{re} (%)	r_{re} (°/100m)	t_{rmse} (m)	t_{re} (%)	r_{re} (°/100m)	t_{rmse} (m)
00	1.78	0.73	22.3	0.70	0.25	1.3	0.63	0.26	1.0
01	5.07	1.06	105.3	1.39	0.21	10.4	2.36	0.36	9.0
02	1.66	0.62	39.5	0.76	0.23	5.7	0.79	0.23	2.6
03	2.03	0.64	0.9	0.71	0.18	0.6	1.01	0.28	1.2
04	2.18	0.58	2.9	0.48	0.13	0.2	0.38	0.31	0.2
05	2.26	0.57	12.5	0.40	0.16	0.8	0.64	0.18	1.5
06	1.80	0.68	2.6	0.51	0.15	0.8	0.71	0.18	1.3
07	1.69	0.64	1.7	0.50	0.28	0.5	0.56	0.29	0.5
08	2.29	0.49	20.6	1.05	0.32	3.6	1.11	0.31	3.9
09	3.03	0.65	12.2	0.87	0.27	3.2	1.14	0.25	5.6
10	2.38	1.01	4.7	0.60	0.27	1.0	0.72	0.33	1.5

Table 2.3: The absolute translation RMSE t_{rmse}, relative translation error t_{re} and relative rotation error r_{re} of the proposed low-power SLAM system, ORB-SLAM2 [91] and Stereo LSD-SLAM [36] on KITTI dataset sequences 00 to 10.

performance on sequences 00, 02, 05, 08 and 09. However, it gets the worst result on sequence 01 because of the more challenging highway scenario. In sequence 01, fewer close points can be correctly tracked due to the high speed of the vehicle and low frame rate of the recorded video, hence camera translation is much harder to estimate by the proposed system. Besides, some sequences contain a lot of turns, such as sequences 00, 02, 05 and 08, which could lead to severe drift that is hard to be dealt with by the proposed system. Moreover, the estimated trajectories versus the ground truth trajectories of all the 11 sequences are plotted in Figure 2.5 for quality evaluation.

Although the proposed low-power SLAM system performs slightly worse than ORB-SLAM2 and Stereo LSD-SLAM in terms of absolute translation RMSE t_{rmse}, relative translation error t_{re} and relative rotation error r_{re} on most of the sequences, the most important advantages of the proposed system are the low power consumption and small memory usage. Table 2.4 summarizes the power consumption and

	Proposed system	ORB-SLAM2 [91]	Stereo LSD-SLAM [36]
Frame rate (KITTI)	72.5 FPS	16.7 FPS	14.3 FPS
Platform	Specialized VLSI ASIC	Intel i7-4790 2 cores @ >3.5 GHz	Intel i7-4900MQ 1 core @ 2.8 GHz
Power	200 mW	42 W	11.75 W
Memory	13 MB	4 – 6 GB	1 – 3 GB

Table 2.4: Frame rate on KITTI, deployed platform, system power consumption and memory usage of the proposed low-power SLAM system, ORB-SLAM2 [91] and Stereo LSD-SLAM [36].

memory requirement of ORB-SLAM2, Stereo LSD-SLAM and the proposed SLAM system. To achieve real-time operation, i.e. $\geq$ 10 FPS (frame per second), ORB-SLAM2 and Stereo LSD-SLAM require desktop CPUs with more than 10 watts of power, while the proposed low-power system runs on specialized ASIC accelerator with only 200 mW power. Moreover, the proposed system only requires 13 MB of on-chip memory, while ORB-SLAM2 and Stereo LSD-SLAM need at least gigabytes of memory. The proposed low-power SLAM system achieves much lower power consumption and memory usage with reasonable performance, which is more capable for mobile IoT applications.

2.5 Summary

This work shows a complete low-power visual SLAM system with stereo or RGB-D cameras for real-time applications. The work mainly focuses on optimizing the system for hardware-efficient VLSI implementation through hardware and algorithm co-design. In the visual odometry front-end, a simple CNN based visual feature descriptor is introduced to help on improving the feature matching and tracking accuracy. To reduce the system's overall computation cost and on-chip memory usage, a new region-prediction based feature matching technique and a new keyframe selection and update scheme are proposed. By running the proposed system on the KITTI Visual Odometry / SLAM Evaluation dataset, satisfactory results are obtained on most of the evaluation sequences compared to the state-of-the-art algorithms. With much lower power consumption and memory usage, the proposed system is capable for mobile IoT applications, and it is already ported on a low-power SLAM ASIC accelerator.

CHAPTER III

Efficient Deep Visual-Inertial Odometry with Neural Architecture Search and Adaptive Visual Modality Selection

This work is collaborated with Mingyu Yang. Thanks for his contribution in deep VIO with adaptive visual modality selection (Section 3.3.3).

3.1 Introduction

Visual-inertial odometry (VIO) continuously determines the position and orientation of the agent from captured sequential images and inertial measurement unit (IMU) readings. It has a wide range of applications in robotics and computer vision related tasks, such as localization, navigation, autonomous driving, augmented reality (AR) and virtual reality (VR). Compared to visual odometry (VO) systems [13, 35, 39, 90, 151, 152, 164], VIO systems [18, 47, 77, 80, 99] incorporate additional IMU measurements and are more robust in texture-less environments and under extreme lighting conditions.

With the rapid development and superior performance of deep learning in various computer vision tasks [32, 68], deep learning based VO/VIO systems [18, 47, 80, 151, 152, 164] have received growing attention from the community in recent years. Compared with conventional geometric based methods [35, 39, 77, 90, 99], data-driven

VIO methods achieve competitive performance in both accuracy and robustness while avoiding manual interventions for system initialization and parameter tuning (e.g., number of features per frame, the threshold of feature matching, keyframe selection and so on). Moreover, learning based methods extract better features and learn more efficient fusion mechanisms through deep neural networks (DNNs). However, the excessive complexity of DNNs [125] incurs a significant burden on computing platforms. It is impractical to directly deploy learning based VIO models in real-time without powerful GPUs, which are typically not available on energy-constrained mobile platforms operating with low-cost, energy-efficient cameras and IMU sensors.

Reducing the complexity of DNNs has drawn significant attention in the community, and various kinds of methods have been proposed to combat fast-growing DNN model sizes. One of the efficient approaches is network weight pruning and quantization [49, 159], where less important weights or channels are iteratively pruned without loss of accuracy. Designing compact models specialized for certain mobile platforms [105] is another efficient way, but it is a time-consuming process, heavily relying on computer architecture expertise. More recently, neural architecture search (NAS) [162, 163], and its more practical variant one-shot NAS [14, 150] are proposed for automating the DNN model design, making it possible to find the optimal network structure given certain hardware-specific constraints (e.g., MACs, latency and/or memory). However, prior NAS works mainly focus on image classification [14, 21, 150, 154], depth estimation [154], object detection [21] and semantic segmentation [79]. Applying one-shot NAS to VIO is not straightforward because each task has its own unique structure.

On the other hand, the computation of learning based VIO models heavily lies on the sensor data encoder (for data processing and feature extraction), especially

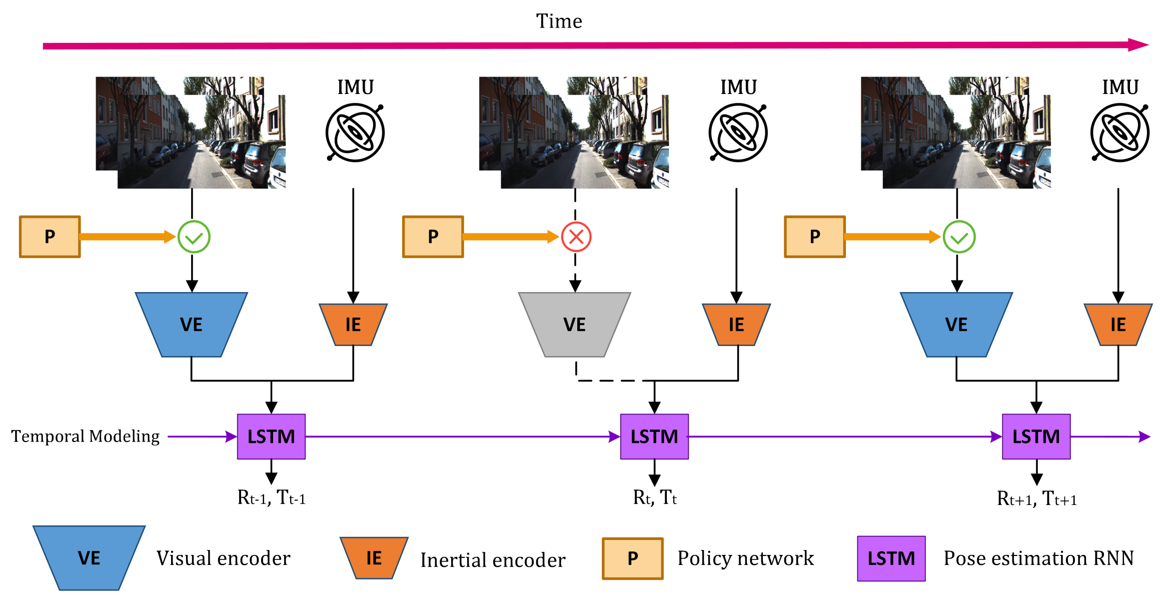


Figure 3.1: An overview of the adaptive visual modality selection technique. A policy network is learned to adaptively disable the visual encoder on the fly without harming the VIO system accuracy, thus saving huge computation from the visual encoder.

the visual modality encoder. The visual encoder is usually much more computation demanding than the inertial encoder due to the big image dimension and its rich information. It is observed that some sensor data is not always helpful for accurate pose estimation, thus the system computational complexity can be reduced by intentionally disabling the visual modality on the fly while keeping IMU always available. Learning a good policy is the biggest challenge to achieve this strategy.

This work proposes two techniques to reduce the overall computation of the learning based VIO systems [24, 153]. First, one-shot NAS is applied to identify the most efficient visual encoder network with low complexity and low latency, which brings down the system complexity to 2.6% of the baseline model without performance degradation. The VIO system with searched efficient visual encoder can run in real-time on a laptop CPU (Intel i7-7700HQ) at the rate of 83.3 frames per second (FPS). Motivated by recent works on temporal adaptive inference for efficient action recognition [87, 88, 95, 148] and fast text classification [16, 51, 109], the second technique relies on a lightweight policy network to adaptively disable the visual (image) modality on the fly to alleviate the high computational cost of learning based VIO methods, as illustrated in Figure 3.1. Since visual information does not always contribute to accurate motion estimations, especially when the ego-motion is relatively small over time, occasionally skipping unimportant images could save lots of computation with almost no performance degradation. According to the experimental results, visual modality can be disabled up to 72% of the time without compromising VIO accuracy. Furthermore, these two techniques can be combined together to further reduce the system complexity down to only 0.9% of the baseline for KITTI dataset evaluation. With combined techniques, the proposed framework is suitable for mobile platforms with limited computation resources and energy budgets.

3.2 Related Work

3.2.1 Visual-Inertial Odometry

Conventional VO/VIO systems typically consist of four steps: feature detection, feature matching and tracking, motion estimation, and local optimization [107]. They often serve as the front-end of simultaneous localization and mapping (SLAM) systems [90, 91, 99], followed by 3D environment mapping, global optimization and loop closure steps to form the complete SLAM systems. The performance of conventional systems is largely affected by visual feature matching and tracking accuracy, as well as sensor fusion strategy. Existing state-of-the-art systems rely on superior handcrafted feature descriptors [84, 103] for better feature matching and tracking accuracy, and different sensor fusion schemes, such as adaptive filtering [77] and nonlinear optimization [57, 76] for better performance.

End-to-end deep learning based VO/VIO models are first introduced in a supervised learning manner [18, 26, 151, 152] that regress the six degrees of freedom (6-DoF) camera relative poses from consecutive image frames and IMU measurements to minimize the difference from the ground-truth poses. A long short-term memory (LSTM) network is first introduced by VINet [26] to model the temporal motion correlation. Later, different fusion mechanisms of the visual and inertial features have been proposed, for example, soft and hard masking techniques proposed by Chen et al. [19], and attention based fusion module introduced in ATVIO [80]. More recently, self-supervised learning based frameworks [2, 47, 110, 164] propose to train the model without ground-truth poses and achieve more accurate motion estimations. These works utilize additional information for self-supervision, such as optical flow [47], depth information [2] and multi-level online error correction [110]. However, these learning based VIO systems focus on improving system performance without considering the system complexity and power consumption, making them unsuitable for energy-constrained mobile platforms.

3.2.2 Neural Architecture Search

Pioneer NAS works [7, 162] attempt to search for high performance models in a large sequential search space, which is more representative but computationally expensive to cover. Later, one-shot NAS [14, 150] treats all architectures as subgraphs of a supergraph with shared weights [34]. Only the 'super-network' (with supergraph) needs to be trained once and all subnets (subgraphs) can be directly sampled from the super-network reusing its weights, which greatly speeds up the architecture evaluation and search process. VONAS [15] is the first work to apply NAS to VO network design, but it targets on finding the best performance model when the system complexity is unconstrained. Nevertheless, its performance is far from satisfactory compared to state-of-the-art VO/VIO systems. Different from VONAS, this work proposes to apply one-shot NAS to find the most hardware-efficient (low complexity, low latency) VIO network from a carefully designed search space.

3.2.3 Adaptive Inference

Adaptive inference scheme aims to allocate computing resources dynamically based on the input instance of each task, and eliminates the redundant computation for relatively easy task inputs. Several different techniques have been proposed, including early exiting [10, 61, 127], layer skipping [46, 137, 141], and dynamic channel pruning [60, 157]. Recently, the idea of adaptive inference has been extended to sequential data (e.g., texts [16, 51, 109] and videos [87, 88, 95, 148]) that involves recurrent neural networks (RNNs). The proposed adaptive modality selection technique is closely related to adaptive video recognition [148], which introduces a memory-augmented LSTM to adaptively select relevant frames for fast action recognition. Similarly, AR-Net [87] learns a policy through the Gumbel-Softmax trick to dynamically select the optimal resolution of each video frame for efficient action recognition. Later, this idea was extended to modality selection [95] and image region (patch) selection [142] on the same task. Motivated by these prior works, the proposed technique is the first to apply a similar framework for adaptive sensor modality selection on deep learning based VIO models.

3.3 Method

This section first presents the baseline deep VIO model architecture and its training strategy. Then the steps to search for the most efficient visual feature encoder through one-shot NAS are discussed. Finally, the adaptive visual modality selection technique along with Gumbel-Softmax training trick is introduced to further reduce the model's overall computational complexity.

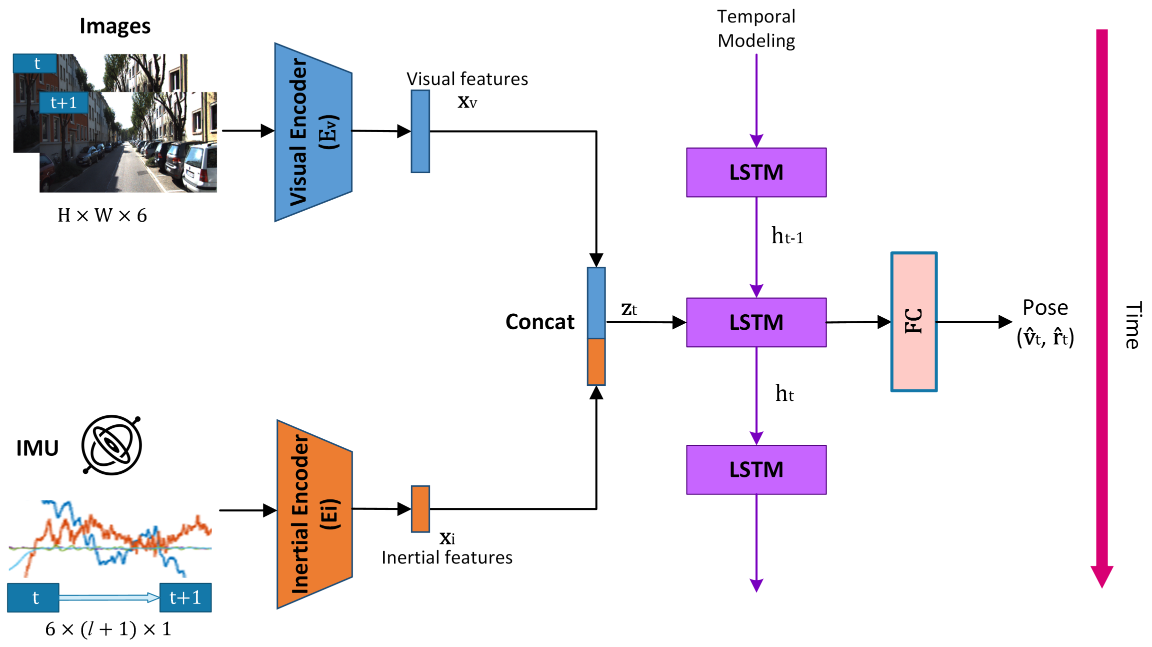


Figure 3.2: The architecture of the deep VIO model that serves as the baseline. Visual feature and inertial feature are extracted by the visual encoder and inertial encoder, respectively, then combined by direct concatenation to get the fused future $\mathbf{z}_t$. Finally, translation $\hat{\mathbf{v}}$ and rotation $\hat{\mathbf{r}}$ are estimated through the LSTM network.

3.3.1 Deep Visual-Inertial Odometry

Figure 3.2 shows a typical deep learning based VIO system, which involves a visual encoder E_v to extract features from two consecutive images $\mathbf{I}_t$, an inertial encoder E_i to extract features from IMU measurements $\mathbf{X}_t$ between two images, a fusion function f to fuse both features, and an RNN to perform 6-DoF pose regression. This process can be expressed as:

$$\mathbf{z}_t = f(E_v(\mathbf{I}_t), E_i(\mathbf{X}_t)), \tag{3.1}$$

$$\hat{\mathbf{v}}_t, \hat{\mathbf{r}}_t, h_t = \mathrm{RNN}(\mathbf{z}_t, h_{t-1}), \tag{3.2}$$

where $\mathbf{z}_t$ is the fused feature vector, h_{t-1} and h_t are the RNN hidden states, $\hat{\mathbf{v}}_t$ and $\hat{\mathbf{r}}_t$ are the estimated relative translation and rotation of the camera at time t. Different sensor fusion methods have been explored by prior works, such as direct concatenation [26], masking [19] and attention module [80]. This work proposes to simply concatenate visual and inertial features and let the pose RNN learn their importance for estimating the pose output. This simple fusion strategy is also beneficial to the proposed adaptive visual modality selection technique. Since the image dimension is much larger than that of the IMU measurement, the visual encoder is usually much bigger than the inertial encoder and the rest of the network. Hence one of the main focuses is to find a more efficient visual encoder with lower complexity through NAS while maintaining its compatibility with the rest of the system. Due to the nature of sequential motion estimation, temporal information such as the estimated motions and states of previous frames is important for accurate pose estimation of the current frame. Hence RNNs are typically employed to learn the temporal correlation within frames.

This VIO system can be trained by minimizing the combined translation and rotation mean squared error (MSE):

$$\mathcal{L}_p = \|\hat{\mathbf{v}} - \mathbf{v}\|_2^2 + \alpha\|\hat{\mathbf{r}} - \mathbf{r}\|_2^2, \tag{3.3}$$

where $\mathbf{v}$ and $\mathbf{r}$ denote the ground-truth translation and rotation vectors. α is a scale factor for balancing the translation error and rotation error. In all the experiments, α is set to 100, same as the previous supervised learning VO/VIO methods [18, 26, 80, 140, 152].

3.3.2 Neural Architecture Search on Visual Encoder

VONAS [15] searches for the VO architecture with best performance from scratch. Thus it has a big and unstructured search space without any prior knowledge. However, many preceding VO/VIO systems [18, 47, 151, 152, 164] have proven the effectiveness of adapting Flownet [32], a set of convolutional neural networks (CNNs)

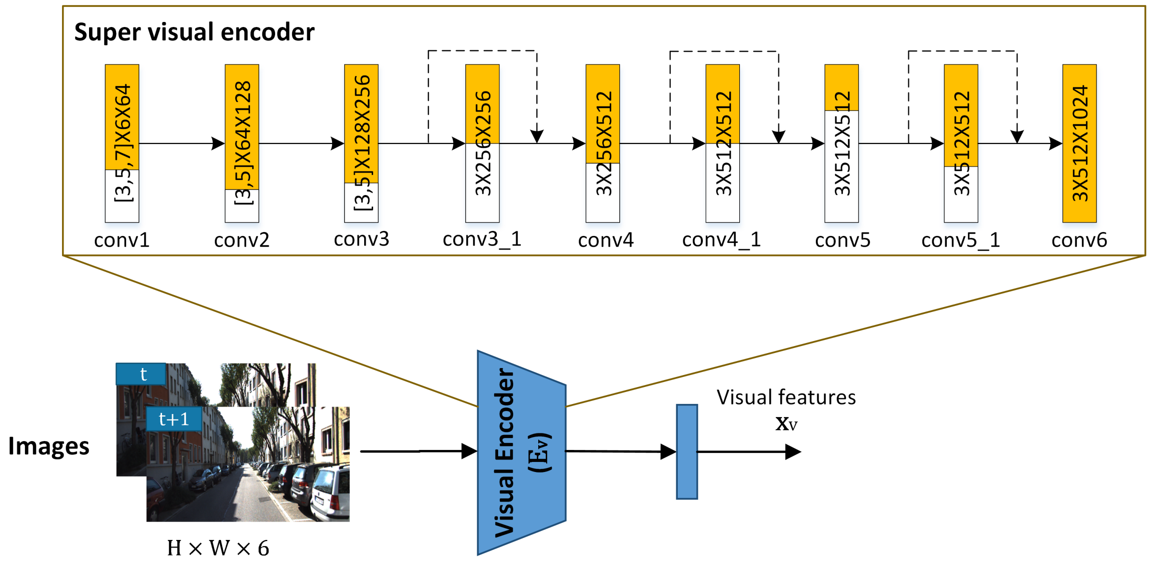


Figure 3.3: The super visual encoder and its search space. The biggest subnet is equivalent to FlowNetS [32] with 9 convolutional layers. Layer *conv1*, *conv2* and *conv3* can search for different kernel sizes. The first 8 layers can search for different numbers of channels. And layer *conv3_1*, *conv4_1* and *conv5_1* can be skipped via direct connection.

proposed for optical flow estimation, as the visual encoder. Therefore, taking the advantage of this, the architecture of a simple version, the FlowNetS [32] (without the last layer), is chosen as the biggest visual encoder network here. The FlowNetS contains 9 convolutional layers as shown in Figure 3.3.

The search space includes convolution kernel sizes, number of filters and network depth, as shown in Figure 3.3. The original kernel sizes are 7, 5, 5 for the first 3 layers and 3 for the rest layers in FlowNetS. To avoid adding additional complexity, the kernel size is selectable in $\{7, 5, 3\}$ for the first layer, and in $\{5, 3\}$ for the second and third layers while no selections are provided for the rest layers. The number of channels is reduced by multiplying FlowNetS's original channel sizes by scaling multipliers, which are selected from $\{\frac{1}{8}, \frac{2}{8}, \ldots, \frac{7}{8}, 1\}$ for the first 8 layers. The number of channels of the last layer remains the same to maintain the final visual feature dimension, which can be directly fed to the fusion network without additional operation. For network depth, instead of directly deciding the total number of layers of the visual encoder, skipping via direct connections for layer *conv3_1*, *conv4_1* and *conv5_1* is allowed as they do not contain dimension reduction operations and their input feature maps can be directly passed to the next layer. The search space size of the proposed method is around 10^9, which is much smaller than that of VONAS (10^{13}) [15]. The smaller search space guarantees faster super-network training and architecture search.

The super visual encoder is jointly trained with the inertial encoder and pose RNN, and is evaluated by pose estimation accuracy of the entire VIO system. Although NAS is only performed on the visual encoder, training and evaluating it in isolation is impractical. Besides, good subnet candidates should produce similar visual feature vectors, leading to accurate pose estimations of the entire VIO system. To warm up the whole system, the VIO model with the biggest subnet (same as FlowNetS) is trained for several epochs first, which is initialized with weights pre-trained on the FlyingChairs dataset [32]. Then following the progressive shrinking algorithm [14], selections on kernel sizes, number of channels and depth are gradually added and optimized, with new subnets generated for joint optimization. Although progressive shrinking helps prevent the 'interference' between subnets, noticeable performance degradation of bigger subnets is still observed during the optimization of newly sampled subnets. Inspired by the sandwich rule [156], the biggest subnet is sampled every N iterations ($N = 500$ in all experiments) for optimization to alleviate such degradation.

Once the super visual encoder is well-trained, the next step is searching for the optimal architecture given certain constraints. Suppose the super visual encoder with trained shared weights is $\mathbf{W}$. Denoting the set of architectures of all subnets by A, the neural architecture search algorithm solves the following problem:

$$a^* = \arg\max_{a \in A} \Psi(\varphi(\mathbf{W}, a)), \\ \text{subject to } g_i(a) \leq C_i, \tag{3.4}$$

where a denotes a subnet architecture in A, φ returns the subnet specified by a from the super visual encoder network, Ψ is the system performance evaluation function, and C_i is the i_{th} system constraint such as computation cost, system latency, and memory usage.

The goal is to search for the most efficient visual encoder architectures without performance degradation. Instead of finding the best performance model under explicit efficiency constraints (complexity, latency and/or memory usage), the proposed technique sets a small tolerance range on VIO performance degradation as a constraint and searches for the models with the lowest FLOPs (floating-point operations) or lowest inference latency within this constraint. Reinforcement learning [7, 15, 162] and evolution search [14, 21, 100] are the most commonly used search algorithms and are proven to be efficient and effective in previous works. Since the evolution search is more friendly with hard system constraints as in this case, the evolution search technique proposed by [100] is adapted to find the targeting efficient visual encoder network structure.

3.3.3 Deep VIO with Adaptive Visual Modality Selection

The overview of the proposed visual modality selection technique is illustrated in Figure 3.4. For this adaptive method, the goal is to learn a binary decision d_t to determine whether the visual modality can be disabled without a significant motion estimation accuracy drop. A decision module is introduced to make the decision d_t by sampling from a Bernoulli distribution, whose probability $p_t \in \mathbb{R}^2$ is generated by

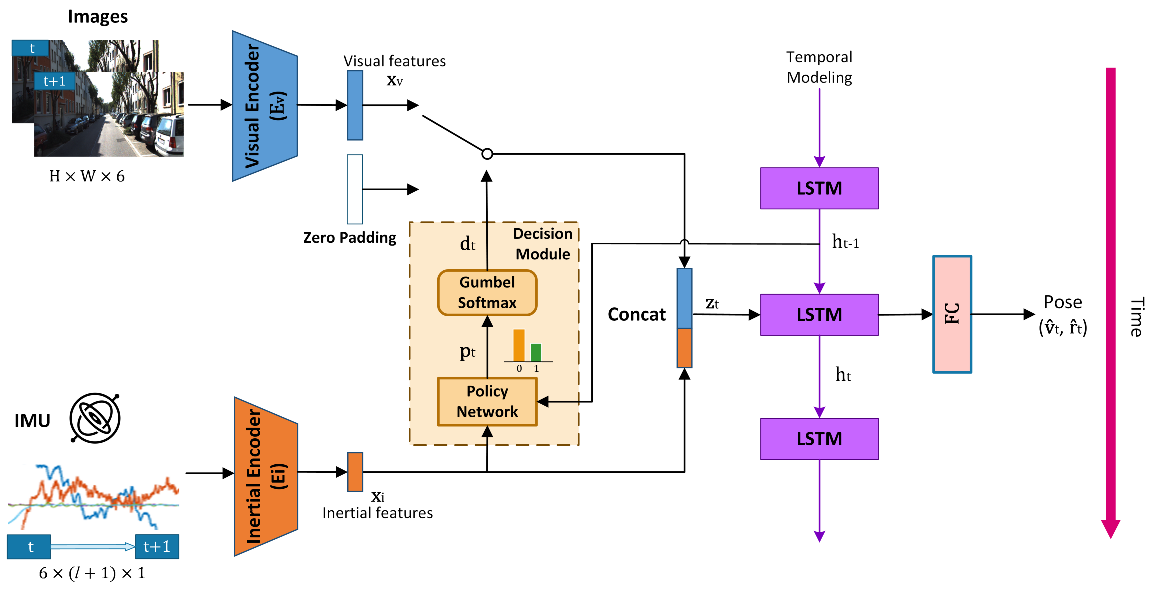


Figure 3.4: Deep VIO system with proposed adaptive visual modality selection technique. At the current time stamp, the policy network takes the current inertial feature and the previous hidden state of the LSTM to decide whether to use the visual modality or not. Once the policy network decides not to use visual modality, the visual encoder is disabled to save the computation and the visual feature is zero-padded.

a lightweight policy network P. The policy network takes the previous RNN hidden state h_{t-1} and the current inertial feature x_t as inputs, where x_t is extracted by the inertial extractor E_i from the current IMU measurements $\mathbf{X}_t$. This can be expressed as:

$$p_t = P(h_{t-1}, x_t). \tag{3.5}$$

However, sampling from a Bernoulli distribution is discrete and non-differentiable. One practical solution is to use a score function estimator, such as REINFORCE [42, 145] or other policy gradient methods. But REINFORCE often suffers from slow convergence and high variance [147] in many applications. Alternatively, Gumbel-Softmax scheme [63], a reparametrization trick for categorical distributions [65, 89, 101], is adapted to enable end-to-end training through back-propagation. This kind of reparameterization trick is easier to implement and exhibits lower variance during training.

Consider a categorical distribution with K categories where the probability of the k_{th} category is p_k for $k = 1, \dots, K$. Following the Gumbel-Max trick [63], a discrete sample q that follows the target distribution can be drawn by:

$$q = \arg\max_k(\log p_k + g_k), \quad k \in \{1, 2, \dots, K\}, \tag{3.6}$$

where $g_1, \dots, g_K$ are i.i.d samples drawn from Gumbel$(0, 1)$. And Gumbel$(0, 1)$ distribution can be obtained by $g_k = -\log(-\log u)$ with $u \sim$ Uniform$(0, 1)$. Then, the softmax function is applied as a continuous and differentiable approximation to $\arg\max$. The sample vector $y \in \mathbb{R}^K$ can be obtained by

$$y_k = \frac{\exp((\log p_k + g_k)/\tau)}{\sum_{j=1}^{K} \exp((\log p_j + g_j)/\tau)}, \quad k = 1, 2, ..., K, \tag{3.7}$$

where τ is a temperature parameter that controls the 'discreteness' of y. Gumbel-Softmax distribution is smooth and has a well-defined gradient when $\tau > 0$. When $\tau \approx 0$, y is close to a one-hot vector for sampling the discrete variable. For the proposed decision module, there are only two categories ($K = 2$) because of the binary decision. During policy network training, the decision is sampled from the target Bernoulli distribution through Equation 3.6 in the forward pass whereas Equation 3.7 is used for gradient approximation in the backward pass.

With Gumbel-Softmax, the binary decision $d_t \in \{0, 1\}$ can be sampled by:

$$d_t \sim \text{GUMBEL}(p_t). \tag{3.8}$$

When $d_t = 1$, the visual encoder is enabled, and visual and inertial features are concatenated before RNN pose regression. When $d_t = 0$, the visual encoder is disabled, and the visual feature is replaced by a zero-padded vector of the same dimension.

This operation can be expressed as:

$$\mathbf{z}_t = \begin{cases} E_v(\mathbf{I}_t) \oplus E_i(\mathbf{X}_t) & \text{if } d_t = 1 \\ \mathbf{0} \oplus E_i(\mathbf{X}_t) & \text{if } d_t = 0 \end{cases}, \tag{3.9}$$

where $\oplus$ denotes the concatenation operation. The combined feature $\mathbf{z}_t$ is then fed to the RNN (a simple two-layer LSTM) for 6-DoF pose regression ($\hat{\mathbf{v}}_t$ and $\hat{\mathbf{r}}_t$) as in Equation 3.2. Notice that the decisions are sampled from the Gumbel-Softmax distribution only during the training phase to make the system end-to-end trainable. During the inference, decisions are sampled from the Bernoulli distribution parameterized by the policy network outputs.

For training the policy network, an additional penalty factor λ is applied for using the visual encoder, thus encouraging the system to disable the visual modality. During the training process, the averaged penalty is calculated and denoted as the efficiency loss defined by:

$$\mathcal{L}_e = \lambda d_t. \tag{3.10}$$

Finally, the deep VIO with adaptive visual modality selection is trained by combining the pose estimation loss (Equation 3.3) and efficiency loss (Equation 3.10) to strike a balance between good accuracy and computational efficiency. The joint loss is defined as:

$$\mathcal{L} = \mathcal{L}_p + \mathcal{L}_e. \tag{3.11}$$

3.4 Experimental Setup

3.4.1 Dataset and Metrics

The proposed approach is evaluated on one of the most popular VO/VIO benchmarks, the KITTI Visual Odometry / SLAM Evaluation dataset [41]. The KITTI

Layer	Total number of FLOPs
2D Convolution	$C_o \times C_i \times k \times k \times \frac{H}{s} \times \frac{W}{s}$
1D Convolution	$C_o \times C_i \times k \times L$
2D BatchNorm	$C_o \times H \times W$
1D BatchNorm	$C_o \times L$
LSTM (each layer)	$4 \times (L + L_h + 1) \times L_h + 4 \times L_h$
Fully connected	$L_h \times L$

C_o, C_i: output and input channel
k: kernel size, s: stride
H, W: 2D input height and width
L: 1D input length, L_h: hidden state size

Table 3.1: FLOPs calculation for each layer type involved in the proposed framework. Non-linear activation functions are ignored.

dataset consists of 22 sequences of stereo videos, where only sequences 00 – 10 contain the ground-truth trajectory. Following previous works [18, 80], sequences 00, 01, 02, 04, 06, 08 and 09 are selected for training, and sequences 05, 07 and 10 are reserved for testing. Sequence 03 is excluded due to the lack of IMU data. The monocular color images (from the left camera) and ground-truth poses are sampled at 10 Hz while IMU is sampled at 100 Hz. Since IMU is not synchronized with other sensors, raw IMU data is linearly interpolated to time-synchronize with the images and ground-truth poses. The input images are resized to 512×256, and 11 IMU measurements (input IMU data dimension is 6×11) are used between two consecutive frames.

For accuracy evaluation, the most common metrics are used, i.e., the root mean square error (RMSE) on translation and rotation, and the relative rotation error r_{rel} and relative translation error t_{rel} for subsequences whose lengths span in $100, 200, \ldots,$ 800 meters [41]. To evaluate system complexity reduction with searched efficient visual encoder, the model FLOPs and the inference speed (frame per second) on a single laptop CPU core (Intel i7-7700HQ) are calculated. To evaluate the policy network and adaptive visual modality selection, the average visual encoder usage rate and average model FLOPs are computed. Table 3.1 lists the FLOPs calculation of each layer type involved in the proposed framework. Computations of all non-linear activation functions are ignored.

3.4.2 Model Architecture and Training Strategies

The baseline VIO model consists of three main parts, visual encoder, inertial encoder and RNN pose regression network. The visual encoder is adopted from FlowNetS network [32] (without the last layer) as described in Section 3.3.2. A fully connected layer is attached at the end of the visual encoder to produce a 512-dimensional visual feature vector. The inertial encoder is a 1D CNN with three convolutional layers followed by a fully connected layer to generate a 256-dimensional inertial feature vector. Due to richer information of the visual modality, the dimension of the visual features is deliberately set to be larger than that of the inertial features. The RNN pose regression network is a two-layer LSTM each with 1024 hidden units. Two fully-connected layers are connected to the LSTM to generate the final 6-DoF pose estimation. The policy network for adaptive visual modality selection is a lightweight three-layer multi-layer perceptron (MLP) network.

First, the visual encoder of the baseline VIO network is initialized with weights pretrained on the FlyingChairs dataset [32], and the entire VIO network is trained for 100 epochs. The learning rate is set to 5×10^{-4} in the first 40 epochs, then 5×10^{-5} for the next 40 epochs, and 1×10^{-6} for the last 20 epochs for fine-tuning. Next, VIO with the super visual encoder reuses the weights of the baseline VIO model and is trained through progressive shrinking. The progressive shrinking training involves three stages, with selection on kernel sizes, number of channels, and network depth added, respectively. Each progressive shrinking training stage takes 100 epochs, where the learning rate is 5×10^{-5} for the first 80 epochs and 1×10^{-6} for the last 20 epochs.

Model	Trans. RMSE (m)	Rot. RMSE (°)	GFLOPs	CPU FPS
Biggest VE	**0.0337**	0.0476	7.766	5.9
FLOPs target	0.0339	**0.0474**	**0.198**	79.1
Latency target	0.0339	0.0476	0.205	**83.3**

Table 3.2: Evaluation results of the VIO models with biggest visual encoder and searched encoders targeting low FLOPs and low latency on KITTI testing sequences 05, 07 and 10.

The evolution search algorithm is applied to the super visual encoder to search for the most efficient architecture given performance constraints. And the searched network is fine-tuned for 20 epochs to regain accuracy. Finally, VIO with searched efficient visual encoder is jointly trained with the policy network for 40 epochs with a learning rate of 5×10^{-5}, and fine-tuned for additional 20 epochs with a learning rate of 1×10^{-6}. The initial temperature of Gumbel-Softmax is set to 5 and is decayed exponentially for each epoch with a factor of -0.05. The visual modality is always used in the first frame to guarantee a qualified initial pose estimation for both training and inference.

For all the models, a weight decay of 5×10^{-6} is applied to avoid overfitting. Adam optimizer with $\alpha = 0.9$ and $\beta = 0.999$ is used, and the training batch size is set to 16. The training subsequences are extracted from the original long sequences with an overlap of 1 frame between subsequences, and its length is set to 11. During the training stage, horizontal flipping is applied to images with 50% probability, and ground-truth poses and IMU data are adjusted accordingly.

3.5 Experimental Results

In this section, the results of the searched efficient visual encoder are presented first. Then an ablation study on the penalty factor is discussed to compare the proposed adaptive visual modality selection scheme with the full modality baseline. Finally, the optimal models obtained in this work are compared to state-of-the-art

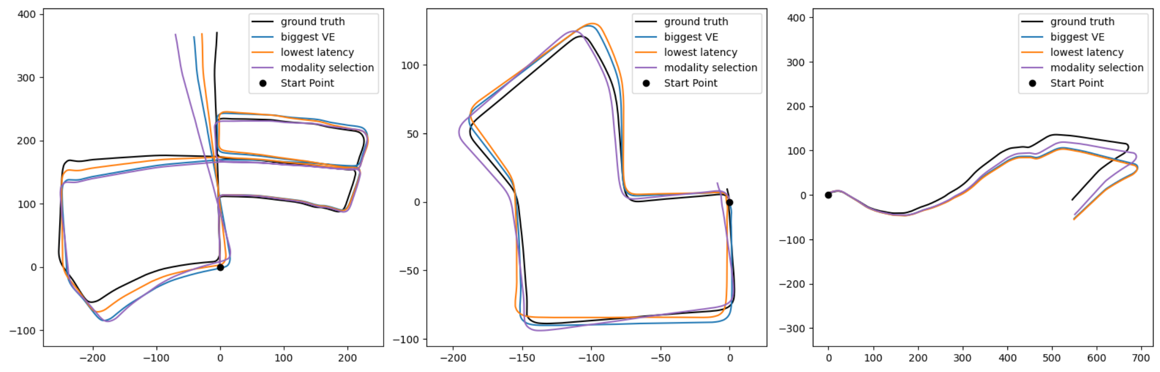


Figure 3.5: Trajectories of 1) ground-truth, 2) full model (baseline model), 3) searched model with low latency target, and 4) searched model with adaptive visual modality selection. Left / center / right plot is the evaluation of KITTI sequence 05 / 07 / 10.

learning based VO/VIO systems to show the huge computation saving without system performance degradation.

3.5.1 Search for Efficient Visual Encoder

When executing the evolution search, the performance degradation tolerance range is set to be at most 10% as the hard constraint, then searching for the model with the lowest FLOPs and lowest latency, respectively. The performance is evaluated by the combined rotation and translation mean squared error.

Table 3.2 summarises the translation and rotation RMSE, model GFLOPs and CPU inference frame rate of the VIOs with the biggest visual encoder and two searched visual encoders (one for the lowest FLOPs and the other for the lowest latency). The searched model targeting the lowest latency achieves 14.1× higher frame rate with only 2.6% FLOPs compared to the full model while maintaining similar accuracy. The proposed search strategy is efficient on reducing most of the redundancy in the original visual encoder and it consequently allows the optimized VIO model to run in real-time ($\geq$ 30 FPS) on a single laptop CPU core. For visualizing the accuracy, Figure 3.5 shows the trajectories of the model with the biggest

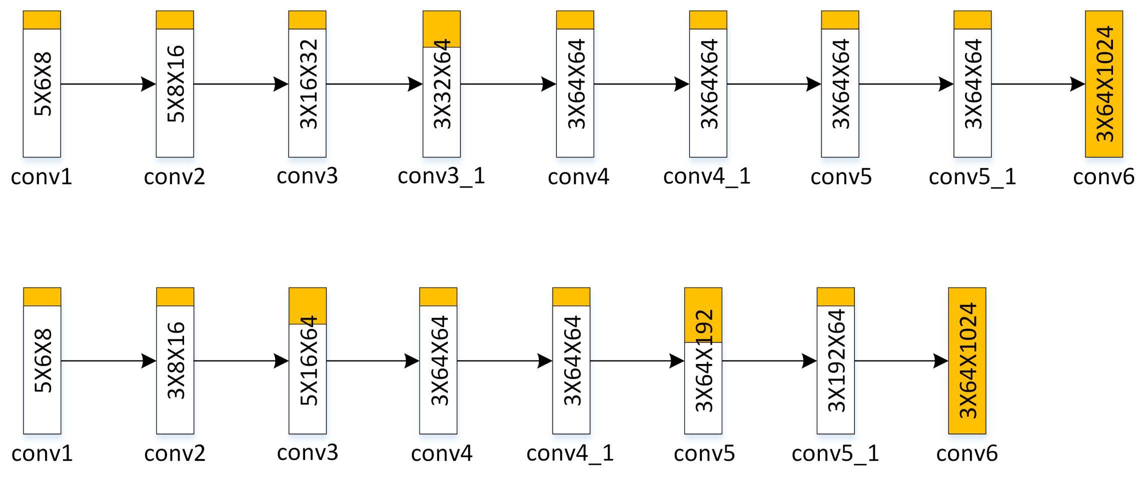


Figure 3.6: Architectures of searched efficient visual encoders through NAS. Top: Lowest FLOPs target. Bottom: Lowest latency target. Notice that the searched model with the lowest latency target makes a trade-off to skip the *conv3_1* layer at the cost of slightly degraded accuracy.

visual encoder (equivalent to FlowNetS) and the searched low latency model, along with the ground-truth trajectories, on KITTI sequence 05, 07 and 10.

The searched network architectures also show some interesting patterns that are closely related to the search targets. Figure 3.6 depicts the searched visual encoder architectures with the lowest FLOPs target and lowest latency target, respectively. The optimal kernel sizes of the first three layers are 5, 5, 3 and 5, 3, 5 for the lowest FLOPs model and lowest latency model, respectively, showing that a bigger kernel size is still critical for the first several layers even on smaller models. Big kernels are capable of extracting features from large receptive fields in the shallow layers, which helps on improving the pose estimation accuracy. But an even bigger kernel size, such as 7, has limited performance improvement while incurring higher computation cost and inference latency, thus it is discarded during the search. It is also observed that skipping layers results in noticeable accuracy drops. Hence the searched model with the lowest FLOPs target does not skip any layer. However, as skipping layers could greatly reduce the computation latency, the searched model with the lowest latency

Method	Trans. RMSE (m)	Rot. RMSE (°)	Visual encoder usage	GFLOPs
Full Modality	**0.0339**	0.0476	100%	0.205
$\lambda = 1 \times 10^{-5}$	0.0417	0.0474	57.5%	0.126
$\lambda = 2 \times 10^{-5}$	0.0431	0.0443	28.0%	0.071
$\lambda = 3 \times 10^{-5}$	0.0475	0.0428	15.1%	0.047
$\lambda = 4 \times 10^{-5}$	0.0596	**0.0421**	**7.5%**	**0.033**

Table 3.3: Evaluation of the full modality baseline with the searched low-latency visual encoder and visual modality selection models with various penalty factors λ on the KITTI testing sequences 05, 07 and 10. Translation and rotation RMSE, visual encoder usage and average model GFLOPs are reported.

target makes a trade-off to skip the *conv3_1* layer at the cost of slightly degraded accuracy.

3.5.2 Adaptive Visual Modality Selection

After obtaining the efficient visual encoder through NAS, the original FlowNetS visual encoder is replaced by the searched visual encoder with low latency target in the proposed VIO model. Then it is integrated with the policy network for adaptive visual modality selection and serves as the full modality baseline. First, models with four different penalty factors, 1×10^{-5}, 2×10^{-5}, 3×10^{-5} and 4×10^{-5} are tested and compared with the full modality baseline. For a fair comparison, all models are trained with the same optimizer and hyperparameters including the number of training epochs and learning rate. In Table 3.3, the translation and rotation RMSE, average usage rate of visual encoder, and average model GFLOPs of each model are reported. It is observed that both visual encoder usage and GFLOPs decrease as penalty factor λ increases. In the meantime, as visual encoder usage (and GFLOPs) drops, the translation RMSE becomes monotonically worse. However, this does not happen to the rotation RMSE, which indicates that visual features do not always contribute to improving rotation estimation accuracy. The model with $\lambda = 2 \times 10^{-5}$

Method	Seq. 05			Seq. 07			Seq. 10			Average		
	t_{rel}	r_{rel}	Usage	t_{rel}	r_{rel}	Usage	t_{rel}	r_{rel}	Usage	t_{rel}	r_{rel}	Usage
Full Modality	**2.17**	**0.74**	100%	**1.98**	**0.59**	100%	3.31	0.74	100%	2.49	**0.69**	100%
$\lambda = 1 \times 10^{-5}$	2.28	0.81	53.9%	2.71	0.78	59.4%	2.84	**0.60**	59.2%	2.61	0.73	57.5%
$\lambda = 2 \times 10^{-5}$	2.33	0.97	25.5%	2.20	0.95	30.5%	2.81	0.85	28.0%	**2.45**	0.92	28.0%
$\lambda = 3 \times 10^{-5}$	2.69	1.02	14.0%	3.11	0.62	16.3%	**2.58**	0.99	14.9%	2.79	0.88	15.1%
$\lambda = 4 \times 10^{-5}$	3.18	1.00	**6.9%**	4.16	0.77	**8.5%**	3.50	0.95	**7.1%**	3.61	0.91	**7.5%**

Table 3.4: The relative translation and rotation error, and visual encoder usage of the full modality model as well as visual modality selection models with different penalty factors λ on KITTI sequences 05, 07, and 10. All models use the searched visual encoder with low latency target. The last column shows the averaged results on all three test sequences.

achieves 72.0% reduction on visual modality usage at the cost of a relatively small loss in translation RMSE while improving the rotation RMSE. The relative translation error t_{rel}, relative rotation error r_{rel}, and visual encoder usage on each test sequence of 05, 07 and 10 as well as the average across all three test sequences are summarized in Table 3.4. Similarly, the proposed visual modality selection technique achieves comparable accuracy to the full modality baseline with $\lambda = 1 \times 10^{-5}$ and even better results with $\lambda = 2 \times 10^{-5}$ which reduces 72.0% of the visual modality usage. Models with more aggressive penalty factors ($\lambda = 3 \times 10^{-5}$ and $\lambda = 4 \times 10^{-5}$) experience mild performance degradation even though more computation is saved. Figure 3.5 also plots the trajectories of the visual modality selection model with $\lambda = 2 \times 10^{-5}$ evaluated on KITTI test sequences 05, 07 and 10.

In Figure 3.7, the visual interpretation of the learned policy with $\lambda = 2 \times 10^{-5}$ on KITTI sequence 10 is presented. The left heat map shows the local visual encoder usage rate with color coding, where darker (lighter) colors represent lower (higher) usages. The local usage is calculated by averaging the decision within a local window of 31 frames. The right plot shows the speed of the vehicle at each time step where darker colors represent lower speed. An obvious correlation is observed between the

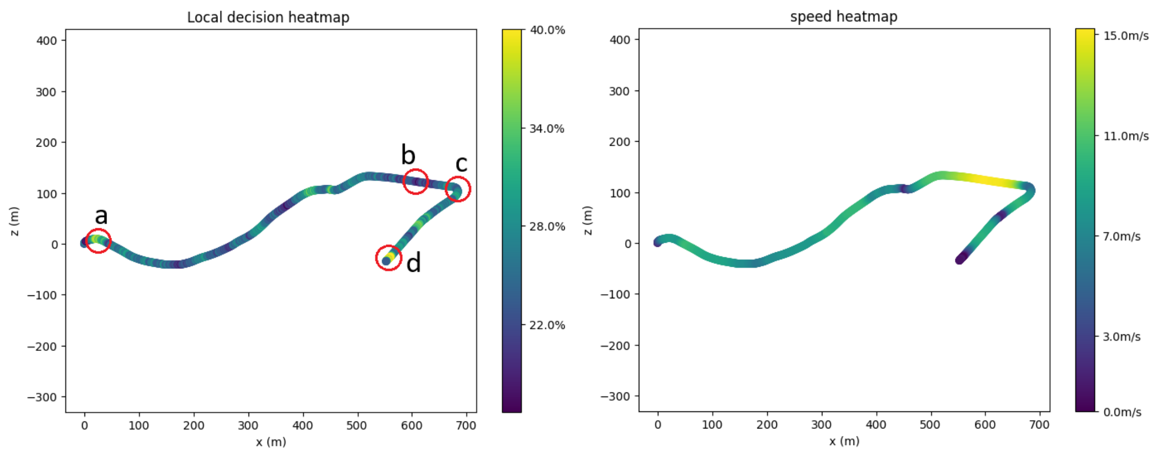


Figure 3.7: Visual interpretation of the learned policy on KTTI sequence 10 with $\lambda = 2 \times 10^{-5}$. The left plot shows the local visual encoder usage rate at each time step calculated by averaging the decision within a local window of 31 frames. The vehicle speed heat map is shown on the right. Four short segments from the path (red circles) are selected to show the policy network's behavior. Segment c shows a low speed with turning scenario, segment a and d show low speed scenarios whereas segment b is a high speed straight movement scenario. The policy network tends to activate the visual encoder more frequently when the vehicle is moving slowly (segments a and d), and decrease the usage of the visual encoder when the vehicle is moving fast (segment b) or making turns (segment c).

visual modality usage and the vehicle speed as well as the turns. Four short segments from the path (marked by red circles in Figure 3.7) are selected to provide more insights into the behavior of the policy network. When the vehicle is moving fast (segment b) or making a sharp turn (segment c), the visual modality is used less frequently. When the vehicle is moving slowly (segments a and d), the visual encoder is activated more frequently.

One possible explanation for this behavior is based on the property of IMU. IMU is capable of estimating the turning angle accurately through a simple first-order integration because it can directly measure the angular velocity, while visual feature based estimation methods can only do the estimation indirectly. However, IMU only measures the acceleration which is the second-order differential of translation. Hence it cannot provide a good estimation of translation without a qualified initialization

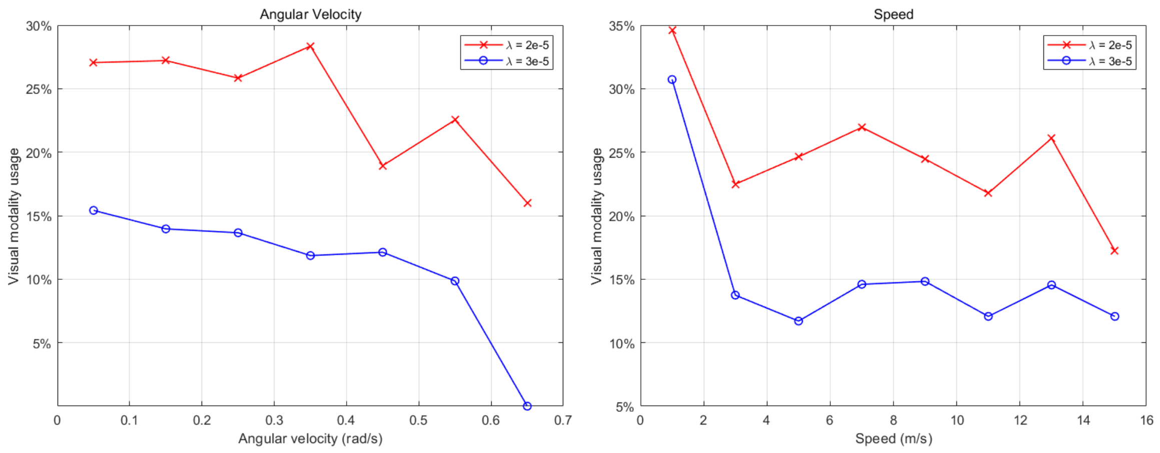


Figure 3.8: The average usage rate of the visual modality for different angular velocities (left) and speeds (right) with two different penalty factors $\lambda = 2 \times 10^{-5}$ and $\lambda = 3 \times 10^{-5}$ over all KITTI test sequences (05, 07, 10). The learned policy tends to use less visual modality with higher angular velocity and higher speed.

of the velocity. Besides, IMU readings are noisier when the vehicle is moving slowly, resulting in inaccurate translation estimation. Thus, when the vehicle is moving slowly, larger translation errors are expected if only IMU is used. And the policy network enables the visual modality more frequently to reduce the error.

To show the general trend, Figure 3.8 plots the visual modality usage versus angular velocity and speed on all test sequences with $\lambda = 2 \times 10^{-5}$ and $\lambda = 3 \times 10^{-5}$. The left plot shows the average visual encoder usage for intervals $[0, 0.1)$, $[0.1, 0.2)$, $\ldots$, $[0.6, 0.7)$ rad$/s$ of the angular velocity, and the right plot shows the averaged visual encoder usage for intervals $[0, 2)$, $[2, 4)$, $\ldots$, $[14, 16)$ m/s of the vehicle speed. It is observed that the usage is closely related to both the angular velocity and the speed. In general, the visual encoder usage tends to decrease with higher angular velocity and higher speed, although there can be occasional spots where this observation does not necessarily hold due to the stochastic nature of the proposed adaptive selection technique.

Method	Seq. 05		Seq. 07		Seq. 10		Avg.		Usage	GFLOPs
	t_{rel}(%)	r_{rel}(°)	t_{rel}(%)	r_{rel}(°)	t_{rel}(%)	r_{rel}(°)	t_{rel}(%)	r_{rel}(°)		
Zou et al. [164] *	2.63	**0.50**	6.43	2.10	5.81	1.80	4.96	1.47	100%	14.19
Beyond Tracking [152] *	2.59	1.20	3.07	1.80	3.94	1.70	3.20	1.57	100%	7.747
GFS-VO [151] *	3.27	1.60	3.37	2.20	6.32	2.30	4.32	2.03	100%	7.747
SelectFusion soft fusion [18] †	4.44	1.69	2.95	1.32	3.41	1.41	3.60	1.47	100%	7.747
SelectFusion hard fusion [18] †	4.11	1.49	3.44	1.86	1.51	0.91	3.02	1.42	100%	7.747
DeepVIO [47] †	2.86	2.32	2.71	1.66	**0.85**	1.03	**2.14**	1.67	100%	14.03
ATVIO [80] †	4.93	2.40	3.78	2.59	5.71	2.96	4.81	2.65	100%	1.790
Full model baseline †	3.03	1.22	2.55	0.97	2.49	**0.53**	2.69	0.91	100%	7.747
NAS searched model †	**2.17**	0.74	**1.98**	**0.59**	3.31	0.74	2.49	**0.69**	100%	0.186
NAS + modality selection †	2.33	0.97	2.20	0.95	2.81	0.85	2.45	0.92	**28.0%**	**0.052**

*: Visual odometry (VO), †: Visual-inertial odometry (VIO)

Table 3.5: Comparison with state-of-the-art deep learning based VO/VIO systems on KITTI testing sequences 05, 07 and 10. The evaluation metrics are: relative translation error (t_{rel}), relative rotation error (r_{rel}), visual encoder usage, and visual encoder GFLOPs.

3.5.3 Comparison to Other Learning Based VO/VIO

The searched model targeting low latency and the searched model with adaptive visual modality selection ($\lambda = 2 \times 10^{-5}$) are compared to the state-of-the-art learning based VO/VIO models. The relative translation and rotation errors on KITTI testing sequences 05, 07 and 10, the visual encoders usage, and the GFLOPs on the visual encoders are reported in Table 3.5. Among all the models, DeepVIO [47] and [164] are self-supervised models, and GFS-VO [151], Beyond Tracking [152], ATVIO [80], and SelectFusion [18] are supervised models. All self-supervised methods are trained on sequences 00 – 08 and tested on 09 and 10. Among supervised methods, GFS-VO [151] and Beyond Tracking [152] are trained on sequences 00, 02, 08 and 09. The rest methods use the same training set as mentioned in Section 3.4.1. DeepVIO [47] uses FlowNetC [32] as its visual encoder, which is much more complicated than FlowNetS [32] used by SelectFusion [18], Zou et al. [164], Beyond Tracking [152], GFS-VO [151], and us. Zou et al. [164] also employ the depth estimation network proposed in [43] to aid the self-supervised learning, which brings extra computation to the system.

ATVIO [80] utilizes the VONAS-A as its visual encoder, which is obtained through the NAS strategy proposed in [15].

It is noticeable that although the main goal of this work is not necessarily maximizing the odometry accuracy, the proposed method still achieves better or comparable performance compared to state-of-the-art methods. The searched model archives comparable or even lower t_{rel} and r_{rel} with far fewer FLOPs, which is only 2.4% of the full model of FlowNetS and 10.4% of the VONAS-A. The optimal model that combines both searched efficient visual encoder and adaptive visual modality selection achieves competitive performance compared to previous VO/VIO methods with only 28.0% visual encoder usage (i.e., it is only active for 28% of the total images), bringing the average visual encoder FLOPs down to 0.052G, which is only 3.2% of the previous best model. For the entire VIO system, it saves up to 99.1% of the computation using both techniques compared to the full model. This is so far the smallest deep learning base VIO system practically deployable on various energy-constrained mobile platforms.

3.6 Summary

This work proposes two techniques to obtain an energy-efficient deep learning based VIO system. First, a one-shot NAS approach is proposed to search for the most efficient visual encoder with lower complexity. The search space is carefully designed to align with typical deep VIO systems, and this work targets on searching for models with low FLOPs and low latency without sacrificing performance. The searched efficient model targeting low latency brings down the system complexity to 2.6% of the baseline model without performance degradation. The second technique reduces the system computation overhead and power consumption by opportunistically disabling the visual modality on the fly when visual information is not critical for maintaining pose estimation accuracy. To learn the selection strategy, a decision module is introduced and end-to-end trained with the Gumbel-Softmax trick. This approach can disable the visual modality up to 72.0% of the time without obvious performance degradation. And the learned policy is interpretable and shows scenario-dependent adaptive behaviors. The optimal model that combines both techniques can save up to 99.1% computation, and run in real-time on a single laptop CPU core. The proposed strategies are model-agnostic and can be easily adapted to other deep VIO systems.

CHAPTER IV

An End-to-End Deep Learning Framework for Multiple Audio Source Separation and Localization

4.1 Introduction

Sound source localization is a problem to localize acoustic sources in space using the captured audio signal from a microphone or microphone array. Typically, the direction of arrival (DOA) or angle of arrival (AOA) information of each source is estimated (Figure 4.1). In the case when multiple sound sources are present, such as the *cocktail party problem*, a good source separation strategy is the premise for precise DOA/AOA estimation. Methods that combine both accurate source separation and localization can enable a wide range of practical applications including autonomous robot navigation, virtual reality (VR) headsets, and enhanced audio surveillance [28] in industrial facilities and critical environments.

With the rapid development of deep neural networks (DNNs), and their great success in various computer vision and natural language processing (NLP) tasks, learning based sound source localization system draws more attention in the research community. Compared to the traditional signal processing based source localization algorithms, learning based approaches usually achieve better performance and can be

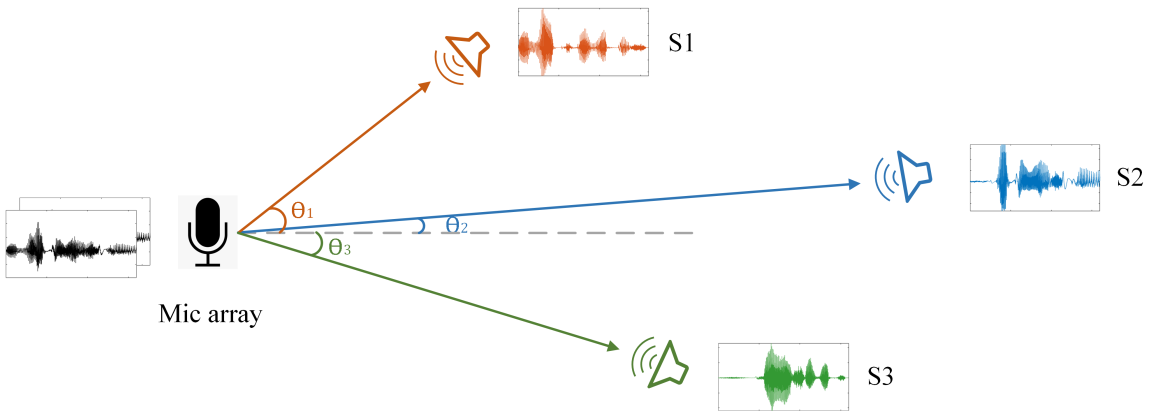


Figure 4.1: Localizing multiple audio sources (S_1, S_2 and S_3) by estimating their DOA or AOA information.

easily deployed on different platforms. Moreover, learning based approaches have no strong assumptions on the signal, noise and environment and hence are more robust to distortions and challenging scenarios, while signal processing based algorithms require algorithm-specific insights to deal with such situations.

This work proposes an efficient end-to-end deep learning framework for source separation and localization [23]. In this approach, separated source signals and their time difference of arrival (TDOA) information between microphones are jointly estimated, then the DOA of each source is estimated using the TDOA information. Although the proposed framework adopts a prior network for source separation, its performance is improved by introducing a companion TDOA estimation network and jointly training them with a new framework where one network assists the other to optimize a similarity loss between the reconstructed and original mixed signals. This new framework ensures the separated sources are realistic as measured by a discriminator and also sufficient to reproduce the original mixture when combined with the estimated TDOA. The proposed framework introduces a new multi-network structure to perform DOA estimation with superior/similar performance compared to the state-of-the-art methods while producing interpretable intermediate information such as separated sources and TDOA information on microphones. Compared to baseline cases where individual networks are trained in isolation, the proposed joint-training scheme achieves superior performance for each task of source separation as well as TDOA estimation because one network assists another network to reproduce realistic reconstructed mixtures during the localization process.

4.2 Related Work

In the early stage, source localization problems were mainly solved by analytical approaches [106, 122, 135] and their robustness is rather limited for practical applications. Recently, deep learning based methods [1, 53, 111, 123, 138] have shown superior performance over the traditional analytical approaches. Existing DNN based source localization methods typically take mixed signals as the input and produce DOA as the output for end-to-end model training and inference without any interpretable intermediate information. However, for the multi-source localization problem, some intermediate results such as the separated source signals and the TDOA information between microphones can be explicitly obtained and utilized to evaluate the loss function to improve the overall end-to-end model performance. Motivated by such observation, the proposed approach takes advantage of such intermediate information for more accurate multi-source localization.

It has been shown that the audio source separation problem can be efficiently solved through deep learning based approaches even with a single channel (microphone) mixture. State-of-the-art performance has been realized in different separation tasks such as speech separation [20, 121, 132], universal sound separation [133], and music source separation [117]. For the multi-source localization problem, first separating each source from the mixture is a natural intermediate step to attain higher

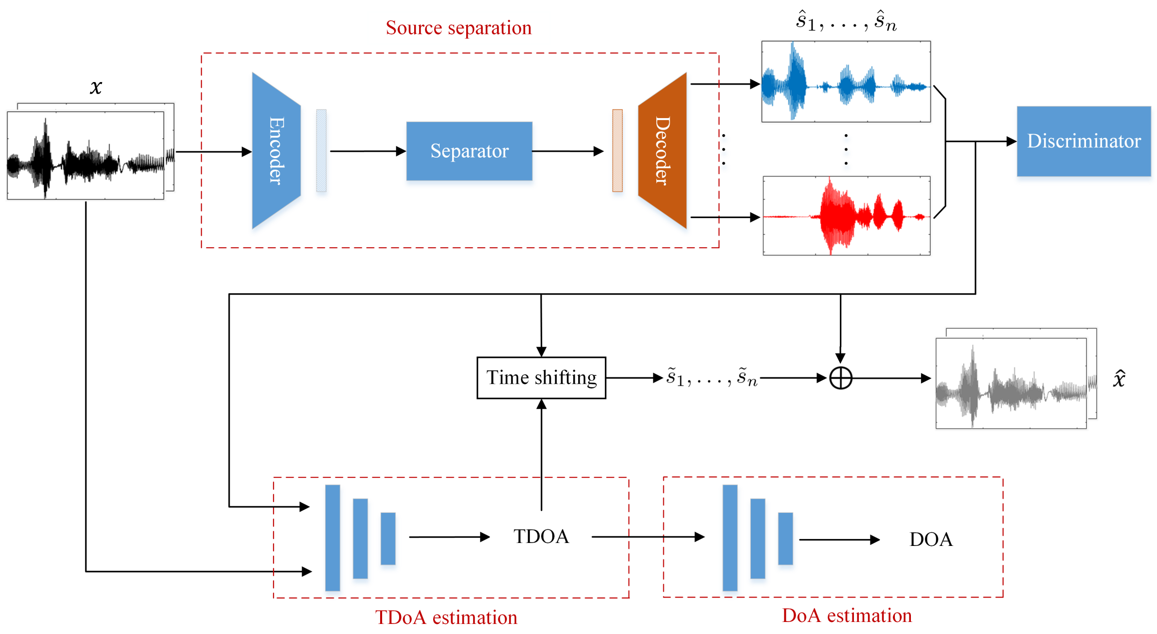


Figure 4.2: The proposed framework diagram. Source separation network estimates source signals $\hat{s}_1, \dots, \hat{s}_n$ from multi-channel mixture x. TDOA information is estimated from x and $\hat{s}_1, \dots, \hat{s}_n$ by TDOA estimation network. $\hat{x}$ is reconstructed by adding $\hat{s}_1, \dots, \hat{s}_n$ and time-shifted source signals $\tilde{s}_1, \dots, \tilde{s}_n$. The reconstructed mixture $\hat{x}$ and discriminator are only used during the training.

source localization accuracy. Hence, an existing state-of-the-art source separation model is adapted and integrated into the proposed framework with TDOA and DOA estimation networks. They are jointly trained with a novel training method to improve both source separation quality and localization accuracy. Besides, the proposed framework is compatible with different deep learning based source separation models as long as the separation is performed in the latent domain.

4.3 Method

Figure 4.2 shows the overall datapath of the proposed scheme for joint source separation, TDOA estimation, and DOA estimation. The proposed model consists of three parts; the source separation network, the TDOA estimation network, and the DOA estimation network. TDOA information is estimated from the multi-channel mixture, then sent to the DOA estimation network for localizing each source. The multi-channel mixture is reconstructed using the separated source signals and the estimated TDOA between microphones, and the similarity loss between the reconstructed mixture $\hat{x}$ and the original mixture x is evaluated for joint optimization of source separation and TDOA estimation. In the meantime, a discriminator is added to improve the quality of the separated source signals.

4.3.1 Source Separation

The source separation network extracts each audio source from the multi-channel mixture. Since various deep learning based audio source separation and speech separation models were previously investigated in the literature, some of the best models [132, 20] are adapted as the source separation network of the proposed framework. As most of the separation models are designed for single channel (microphone) audio source separation, models that perform separation in the latent domains are chosen, where encoder and decoder dimensions can be easily extended to multi-channel (microphone array) inputs and different numbers of sources. The separated audio quality is typically measured by the scale-invariant signal to noise ratio (SI-SNR) [73], defined by

$$\text{SI-SNR}(s, \hat{s}) = 10 \log_{10} \frac{\|\alpha s\|^2}{\|\alpha s - \hat{s}\|^2}, \tag{4.1}$$

where $\alpha = \hat{s}^T s / \|s\|^2$ is a scalar, s is the target source signal and $\hat{s}$ is the estimated source signal. The source separation network is trained to minimize the negative permutation-invariant SI-SNR [155], defined as

$$\mathcal{L}_{\text{sep}}(s^*, \hat{s}) = -\text{SI-SNR}(s^*, \hat{s}) = -10 \log_{10} \frac{\|\alpha s^*\|^2}{\|\alpha s^* - \hat{s}\|^2}, \tag{4.2}$$

where s^* denotes the permutation of the sources that maximizes the SI-SNR, and $\alpha = \hat{s}^T s^* / \|s\|^2$.

4.3.2 TDOA Estimation

This work proposes to simultaneously discriminate and localize N sound sources using the TDOA information estimated with an array of K microphones. The TDOA ΔT_{ij} between a pair of microphones i and j regarding a certain source s is defined as

$$\Delta T_{ij} = \frac{f_s}{c}(\|l_s - l_i\| - \|l_s - l_j\|), \tag{4.3}$$

where l_s is the location (coordinate) of the source, l_i and l_j are the locations of microphone i and j, f_s is the audio signal sampling frequency, c is the speed of sound, and $\|\cdot\|$ denotes the Euclidean norm. There are total $K(K-1)/2$ different TDOAs but only $K-1$ of them are independent. So, only ΔT_{1j} values are chosen for the DOA estimation.

Instead of estimating TDOA by identifying the peak of the cross-correlation of two signals, this work proposes to use a TDOA estimation neural network, which can be easily integrated and jointly trained with the source separation network to build an end-to-end system. Each source signal received at microphone 1 is treated as the reference (non-shifted version) during the source separation stage. The TDOA estimation network uses the estimated reference signal together with the mixtures received at other microphones to estimate the TDOA between microphone pairs regarding the same source. Since the TDOA is discretized due to audio signal sampling and its maximum is limited by the spatial configuration of the microphone array, TDOA estimation is treated as a classification problem where each class represents a possible TDOA in terms of the sample index.

At this point, the source separation network and TDOA estimation network can be independently trained with their own loss functions. For the initial training of the TDOA estimation network, only the original source signals are used. After this preliminary independent training, the source separation network and TDOA estimation network are combined for joint end-to-end training to improve both the separation quality and the TDOA estimation accuracy. The joint training stage involves a new similarity loss (Equation 4.4) between the original mixture x and the reconstructed mixture $\hat{x}$ obtained by applying the estimated TDOA information to the separated source signals. In Equation 4.4, K is the total number of channels (microphones), and $\langle \cdot, \cdot \rangle$ denotes the inner product.

$$\mathcal{L}_{\mathrm{sm}}(x, \hat{x}) = -\frac{1}{K} \sum_{i=1}^{K} \langle x_i^T, \hat{x}_i \rangle. \tag{4.4}$$

This loss function is designed to ensure the reconstructed multi-channel mixture $\hat{x}$ is as close as possible to the original one x. The main issue of applying this similarity loss to network training is the non-differentiable TDOA time-shifting operation for reconstructing the mixture signals. This issue is mitigated by treating the softmax of the output vector y_j from the TDOA estimation network as the channel impulse response, and convolving it with the estimated signal $\hat{s}_j$ to obtain a time-shifted version $\tilde{s}_j$ of the same source j. Then the reconstructed mixture $\hat{x}_i$ of channel i can be obtained by

$$\hat{x}_i = \sum_{j=1}^{N} \tilde{s}_j = \sum_{j=1}^{N} \mathrm{softmax}(y_j) * \hat{s}_j, \tag{4.5}$$

where N is the number of sources, and $*$ denotes the convolution operation. This technique allows end-to-end joint training of the source separation network and TDOA estimation network through back-propagation.

Inspired by the success of generative adversarial networks (GANs) [45], a discriminator network is adopted in the proposed framework to distinguish the estimated/separated source signals (fake samples) from the original source signals (real samples). The discriminator is only applied during the training phase to help on improving the source separation quality. In the proposed framework, the source separation network is treated as the 'generator' in generative adversarial training, and is adversarially trained with the discriminator.

4.3.3 Training Loss Function

For joint training of the source separation network and TDOA estimation network, the total loss function consists of the separation loss, the TDOA estimation loss, the reconstruction loss, and the subjective discriminator loss, defined as

$$\mathcal{L} = \mathcal{L}_{\text{sep}}(s^*, \hat{s}) + \mathcal{L}_{\text{TDOA}} + \alpha \cdot \mathcal{L}_{\text{sm}}(x, \hat{x}) + \beta \cdot \mathbb{E}_{\hat{s}}(\log(1 - D(\hat{s}))), \tag{4.6}$$

where separation loss $\mathcal{L}_{\text{sep}}(s^*, \hat{s})$ is Equation 4.2, $\mathcal{L}_{\text{TDOA}}$ is the cross-entropy loss for TDOA classification, similarity loss $\mathcal{L}_{\text{sm}}$ is Equation 4.4, $D(\hat{s})$ is the discriminator output (probability that estimated source signal $\hat{s}$ is real), and α and β are weights for balancing the loss terms.

4.3.4 DOA Estimation

The DOA estimation network is connected to the TDOA estimation network, taking the estimated TDOA information as the input to estimate the azimuth angle of each source regarding the microphone array. A simple multi-layer perception model is sufficient for solving this regression problem. It is trained separately with the ground-truth TDOA and the corresponding azimuth angles, then it is inserted into the system for the final DOA inference.

4.4 Experimental Setup

4.4.1 System Setup and Datasets

In all the experiments, speech from different speakers is used as multiple sources with a sampling rate of 16 kHz. The microphone array contains $K = 4$ microphones

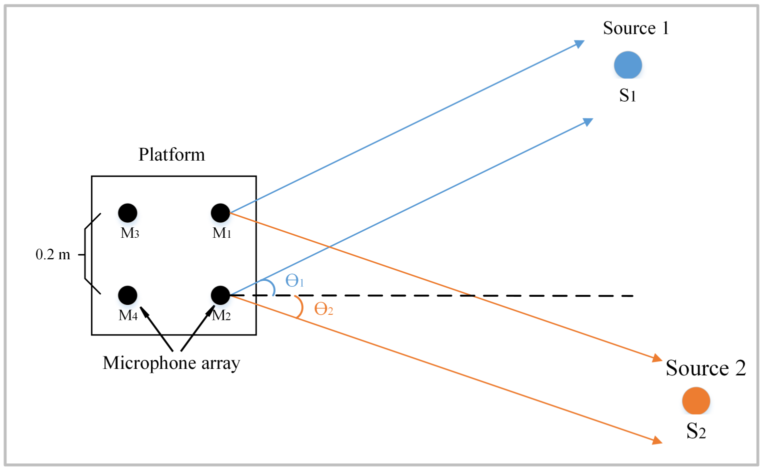


Figure 4.3: The system setup for estimating the DOA (θ_1 and θ_2) of multiple sources (S_1 and S_2) using a microphone array of 4 microphones placed in a square with 0.2-meter side length.

placed in a square shape with 0.2-meter separation per dimension. The distance between sources and the microphone array is restricted in the range of 1 to 3 meters, and all sources are placed with an azimuth angle from 0 to 180°. Since the goal is to estimate accurate azimuth angles, the microphone array and sources are restricted in a 2D plane. The system setup is demonstrated in Figure 4.3.

Publicly available datasets for acoustic source localization and tracking, for example, the LOCATA challenge [82], have limitations such as a small amount of training data, preset number of sources, and restricted microphone array configurations. Hence, synthesized training and testing datasets are generated using LibriSpeech [94], a large-scale dataset with a corpus of read English speech from hundreds of distinctive speakers. The training dataset is generated from the `train-clean-100` set, and speech mixtures are created by randomly mixing speech utterances from different speakers, similar to LibriMix [27]. To simulate microphone array outputs, multi-channel mixtures are created by mixing speech signals using the data augmentation process introduced by Tzinis et al. [131]. The distance and azimuth angle of each source are randomly selected. Then, individual speaker sources are TDOA-shifted and added to create the mixture received at each microphone. The testing dataset is generated from `test-clean` set after removing idle periods greater than 0.5 seconds. Datasets with $N = 3$ and 4 sources are generated for training and evaluation. The audio length is set to 2 seconds in all experiments.

DOA estimation network is separately trained with a synthesized training dataset containing 0.36 million samples. For each sample, source distances are randomly selected from 1 to 3 meters and azimuth angles are randomly selected from 0 to 180°, then the TDOA information is calculated for the corresponding angles.

4.4.2 Model Architecture and Training Details

Two state-of-the-art speech separation models, SuDoRM-RF [132] and DPTNet [20] are adapted to serve as the source separation network. These two models perform separation in the latent domain, thus it is straightforward to adjust the encoder and decoder latent dimensions to accommodate $K = 4$ channel inputs and different numbers of sources ($N = 3$ or 4). In all the experiments, the SuDoRM-RF model uses 16 U-ConvBlocks and ReLU as the mask activation function, and the DPTNet model is shortened by reducing the number of transformer blocks (IntraTransformer and InterTransformer) to 2. The optimizers and other hyperparameters are set to be the same as in the original papers. The separation network is pretrained before jointly training with the TDOA estimation network.

The TDOA estimation network is a 6-layer CNN with four 1D convolutional layers followed by two fully connected layers. In all the experiments, the maximum time shift is less than 20 sample indices, thus the TDOA estimation network has 41 classes (with positive and negative TDOA) output in total. To speed up the joint training process, it is pre-trained with clean speech and time-shifted mixtures from

Separator	N	Sep.	Sep. + Recon.	Sep + Recon. + Disc.
SuDoRM -RF	3	16.75	17.79	18.64
	4	13.06	14.25	14.57
DPTNet	3	14.69	16.92	17.37
	4	10.53	11.77	11.86

Table 4.1: Separation quality evaluation by SI-SNRi (dB) of the proposed framework. Sep., Recon. and Disc. represent separator, mixture reconstruction and discriminator, respectively.

the synthesized training dataset before the joint training with the source separation network. The discriminator is a CNN with four 1D convolutional layers followed by one fully connected layer, and it is trained with the binary cross-entropy loss and Adam optimizer. Noisy inputs are sent to the discriminator to stabilize its training procedure, as introduced by Arjovsky and Bottou [5]. The source separation network and TDOA estimation network are jointly trained for 200 epochs before adding the discriminator for alternated adversarial training. The scalar weights are set to $\alpha = 1$ and $\beta = 0.01$ in the loss function (Equation 4.6). The DOA estimation network is a multi-layer perceptron network with five fully connected layers. The entire framework is implemented and trained using Pytorch [97].

4.5 Experimental Results

The separation quality is evaluated by scale-invariant signal to noise ratio improvement (SI-SNRi) in dB, which is the gain of SI-SNR on the separated signal over the mixture signal. SuDoRM-RF and DPTNet source separators are tested with $N = 3$ and 4 sources, respectively. SI-SNRi values are reported in three cases: training separator solely, training separator with mixture reconstruction, and training separator with mixture reconstruction and discriminator. The evaluation results on source sep-

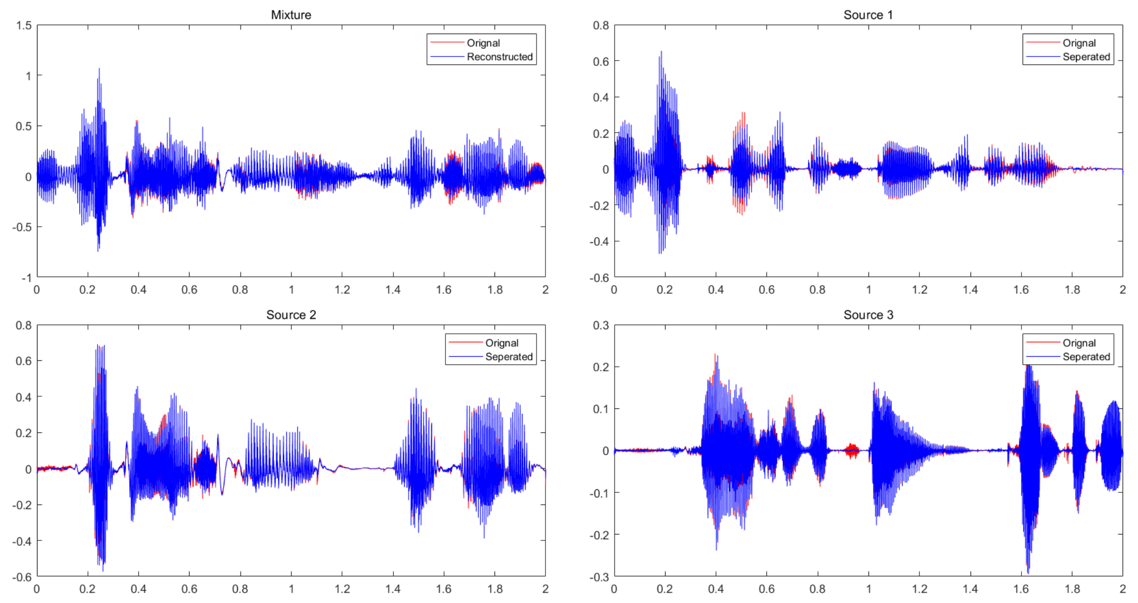


Figure 4.4: The source separation and mixture reconstruction results of one test sample using the proposed framework. Top left: The original (red) and reconstructed (blue) mixtures received at microphone 1. Top right, bottom left and bottom right: Original (red) and separated (blue) audio signals of source 1, 2 and 3, respectively.

aration quality are summarized in Table 4.1. The separation quality is improved by 1.3 – 2.7 dB with the proposed reconstruction structure and discriminator. Figure 4.4 shows the sound waves of the source separation and mixture reconstruction of one test sample using the proposed framework.

Source separation and TDOA estimation are jointly optimized to reduce distortion between the reconstructed mixture and the original mixture. The joint training with the proposed reconstruction loss improves the separation quality. SI-SNRi of separated sources further enhances as the discriminator loss is combined with the reconstruction loss to guide the separated signals to be more realistic speech from a single speaker. It is worth noting that the proposed framework is generalizable to other separation network structures. And it can be further improved with better source separation models in the future.

Separator	N	Without Recon. & Disc.			With Recon. & Disc.		
		E_{TDOA} (ms)	E_{DOA} (°)	R_{DOA}	E_{TDOA} (ms)	E_{DOA} (°)	R_{DOA}
SuDoRM-RF	3	22.4	2.16	93.9%	21.5	2.10	94.5%
	4	28.2	3.25	87.6%	24.4	2.70	91.4%
DPTNet	3	28.8	3.00	89.8%	24.1	2.46	92.7%
	4	44.4	5.38	81.9%	35.6	3.96	86.6%

Table 4.2: MAE of TDOA (E_{TDOA} in millisecond) and DOA (E_{DOA} in degree) estimation, and localization recall (R_{DOA} in percentage) of the proposed framework.

Table 4.2 summarizes the evaluation results on TDOA and DOA estimation of the proposed framework. Mean absolute error (MAE) on TDOA and DOA estimation are reported. Besides, the localization recall on DOA estimation is also reported, where the DOA output is considered true positive only if it is under a threshold of 5° absolute error. Compared to the baseline results without the mixture reconstruction and discriminator, the proposed framework achieves lower TDOA and DOA estimation errors, as well as higher localization recall thanks to the improved separation quality and TDOA estimation accuracy.

Finally, the proposed framework is compared with state-of-the-art DOA estimation algorithms, namely SMESLP [123], I-IDIR-UCA [122] and CHB [128] in Table 4.3. The same recording sequence `seq37-3p-0001` from the AV16.3 corpus [71] with 3 speakers is used for evaluation. For this audio corpus, idle periods are removed and the signal is remixed based on the system microphone array setting since the original dataset uses a circular array with 8 microphones. Unlike SMESLP [123], the proposed system contains only $K = 4$ microphones instead of 8. Results for SMESLP, I-IDIR-UCA and CHB are reported by Sundar et al. [123]. The proposed method with 4 microphones outperforms all three methods on DOA estimation in

Methods	Proposed	SMESLP*	I-IDIR-UCA	CHB
Number of microphones	4	8	8	8
MAE	1.67°	2.05°	-	-
RMSE	3.01°	2.33°	4.1°	2.98°
Non-anomalous frames	98.1%	100%	60%	-

* SMESLP uses part of the testing sequence for fine-tuning and validation. The proposed method uses exclusive sequences for training and validation.

Table 4.3: MAE and RMSE of DOA estimation and percentage of non-anomalous frames of the proposed framework and SMESLP [123], I-IDIR-UCA [122], CHB [128] (reported by Sundar et al. [123]).

terms of mean absolute error (MAE) but gets slightly worse root mean square error (RMSE) compared to MSESLP and CHB using 8 microphones. The percentage of non-anomalous frames is comparable to SMESLP and better than I-IDIR-UCA and CHB. Overall, the proposed framework achieves better or comparable performance with fewer number of microphones. In addition, The proposed scheme produces interpretable intermediate outputs such as separated sources and the TDOA information between microphones, which are not available from other approaches.

4.6 Summary

This work presents an efficient end-to-end deep learning framework for accurate source separation and localization in multi-source environments. By joint training of the source separation network and TDOA estimation network with mixture reconstruction and a discriminator network, the source separation quality as well as the TDOA estimation accuracy are improved. The experimental results confirm that the joint training scheme brings up to 2.7 dB SI-SNR improvement on source separation, as well as higher TDOA and DOA estimation accuracy versus the baseline. Compared to the state-of-the-art source localization systems, the proposed framework achieves superior/similar performance while producing interpretable intermediate information such as separated sources and TDOA information between microphones. The framework is generalizable to other source separation models, and the source separation quality and localization accuracy can be further improved with better separation models in the future.

CHAPTER V

HTNN: Deep Learning in Heterogeneous Transform Domains with Sparse-Orthogonal Weights

5.1 Introduction

Recent deep convolutional neural networks (CNNs) report superb performance in various computer vision tasks such as image classification and object detection. Although the accuracy of CNNs evolved drastically, their computation complexity has also grown super-linearly. The LeNet [74] for handwritten digit classification (MNIST) only requires 0.3 million (M) operations while the winners of the ImageNet classification challenge (ILSVRC) [104] need 0.7 giga (G) (AlexNet [68]) and even 13.5 giga (G) operations (VGG [113]) [125]. It is a significant challenge to deploy such large scale CNNs for real-time applications on energy-constrained Internet-of-Things (IoT) platforms that cannot afford powerful and energy-intensive GPUs. Since the biggest portion of computation comes from convolutional layers, implementing convolution kernels in an efficient way has become a main premise of the successful adoption of large scale CNNs on power- and cost-constrained mobile devices.

Many efficient strategies have been proposed to reduce the complexity of CNNs. One main direction is to replace convolution with simpler operations such as element-wise multiplication using Winograd's minimal filtering algorithm [72] or discrete Fourier transform (DFT) [86]. Another popular direction is to compress the CNN models by weight pruning and quantization to reduce the computation and memory requirements [49, 50, 144, 159]. However, it is not straightforward to apply both techniques at the same time because the sparse spatial weights obtained by pruning are no longer sparse after the transformations required by the first technique. Liu et al. [81] demonstrate the possibility of combining both techniques by applying kernel weight pruning in the Winograd transform domain.

In this work, a new non-convolution based framework is proposed to take advantage of both techniques [22]. Unlike previous work, it is not limited to convolution that uses Winograd transform or DFT. Instead, this work explores the possibility to train deep neural networks (DNNs) in heterogeneous transform domains where convolution is replaced by element-wise multiplication which is (unlike Winograd) no longer equivalent to spatial convolution. To further reduce the computation overhead, this work proposes to use binary-valued fast linear transforms such as the discrete Walsh-Hadamard transform (WHT) and its pseudo-random permuted variations. Such transforms have $O(N \log N)$ complexity and only require additions and subtractions (without multiplication).

The proposed networks are trained with sparse-orthogonal kernels in heterogeneous transform domains. In this approach, two or more kernels in different transform domains can share a hardware multiplier without conflict as the positions of non-zero weights are strictly orthogonal to each other. Thus, the proposed approach is more hardware-friendly compared to conventional weight pruning strategies as it allows parallelized computation of multiple sparse kernels in DNN hardware accelerators with simple multiplexers. Finally, a canonical-signed-digit (CSD) representation

[54] based novel bit-sparse non-uniform quantization is proposed and demonstrated to reduce the density of non-zero digits in quantized weights (i.e., with a sparse non-zero digit constraint) and compute each multiplication with additions/subtractions.

The main contributions are summarized as follows:

- Introduction of HTNN: A new class of heterogeneous transform-domain DNNs to substitute CNNs and their convolution operations with transform-domain element-wise multiplications.

- Feasibility demonstration and evaluation of sparse-orthogonal weights in heterogeneous transform domains that allow hardware multiplier sharing among different kernels for conflict-free parallel execution in hardware accelerations.

- Application of new CSD based non-uniform quantization with a sparse non-zero digit numbering system to reduce the computation complexity by replacing each multiplication with a single addition or subtraction.

- Quantifying the complexity reduction from combined techniques compared to the equivalent sparse CNNs, exhibiting 4.9 – 6.8× gain for the identical DNN accuracy.

5.2 Related Work

5.2.1 Compute Convolutions in Transform Domains

Prior works [86, 136, 139, 143] have demonstrated that convolution in CNNs can be computed with fewer number of operations using DFT or fast Fourier transform (FFT), even for small convolutional kernels (e.g., 3×3). This complexity reduction comes from the duality between convolution in the spatial domain and multiplication in the frequency domain. However, the DFT/FFT transform introduces significant overhead because of complex-numbered operations given real activation input. Therefore, discrete cosine transform (DCT) is often selected as an alternative method [139, 143] to avoid any complex computations. Meanwhile, a different transform based method using Winograd's minimal filtering algorithm [146] was applied to CNNs by Lavin et al. [72] for the first time. It outperforms FFT/DCT based method, especially for small convolution kernels. FFT and Winograd based convolution algorithms are included in the state-of-the-art deep learning library such as NVIDIA cuDNN [25] to improve the computation efficiency of convolutions. Unlike the aforementioned previous work, this work attempts to find different transform domains where element-wise multiplication is no longer equivalent to convolution while DNNs maintain the same accuracy with significantly reduced complexity.

5.2.2 Neural Network Compression

Han et al. [50] are pioneers in CNN weight compression. They present a strategy to iteratively prune less important weights with relatively small magnitudes and to perform retraining to maintain accuracy. Their scheme can achieve an impressive weight pruning ratio of $> 10\times$ for fully connected layers and $\approx 3\times$ for convolutional layers in AlexNet [68] with almost no accuracy degradation. Later, a k-mean clustering based quantization technique was proposed and combined with weight pruning to achieve a higher compression rate [49]. That method has been extended and generalized in multiple directions afterward. For example, Wen et al. [144] propose a structured sparsity learning method to regularize the structures of CNNs (e.g., filter shapes). Zhang et al. [159] formulate network weight pruning as a non-convex optimization problem and adopt the alternating direction method of multipliers (ADMM) framework [11] to solve it. However, these works are still based on spatial convolution and the complexity reduction is limited by the overhead of implementing sparse convolution both in software and hardware [48, 96, 161]. Liu et al. [81] are the first to apply CNN compression techniques to the Winograd transform domain. However, the authors only apply the spatial pruning strategy proposed by Han et al. [50] and the gain is limited. To overcome limitations in those previous approaches, this work introduces a new hardware-efficient sparse structure in heterogeneous transform domains where hardware multipliers are shared for parallel execution of multiple kernels without conflict.

5.3 HTNN: Heterogeneous Transform-Domain DNN

Various kinds of discrete transforms are available in the field of digital signal processing. Many are linear transforms defined by a matrix multiplication in discrete domains. Each transform possesses distinctive properties and some allow fast transform algorithms with complexity of $O(N \log N)$ instead of $O(N^2)$ for the size-N linear transform defined with an $N \times N$ matrix.

It is noticed that a convolutional layer in CNN is a special form of a fully connected layer with weight sharing and pruning in the weight matrix [74]. There exist other (non-convolution) forms of the linear layer that can be described with computationally efficient transforms and are still well-trained with back-propagation algorithms. With a goal of minimizing computational complexity, this work proposes to use Walsh-Hadamard transform (WHT) and its variants (i.e., pseudo-random permuted WHT matrix) that only involve binary elements (± 1). WHT and its permuted versions only require additions, subtractions, and vector permutations instead of multiplications. Using multiple heterogeneous transforms within a DNN enables network training with sparse-orthogonal kernels to further reduce the complexity (Section 5.4).

In this section, WHT and its properties that drew the attention are introduced first. Then the approach to apply WHT to HTNNs quantifying the complexity reduction is discussed.

5.3.1 Walsh-Hadamard Transform and Permuted Variants

Walsh-Hadamard transform (WHT) is a generalized class of Fourier transform [70]. It is built out of size-2 DFT and is in fact equivalent to a multidimensional DFT of size 2^m [70]. Consider a vector $f(x)$ with size $N = 2^m$ and element index $x \in \{0, 1, ...N-1\}$. Using $b_i(x)$ to represent the i^{th} bit (from least significant bit) of x in the binary representation, the 1-dimensional WHT of $f(x)$ is given by:

$$H(u) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} f(x) (-1)^{\sum_{i=0}^{m-1} b_i(x) b_{m-1-i}(u)}, \tag{5.1}$$

where $u \in \{0, 1, ...N-1\}$. The 1D WHT is a linear transform with the transform matrix $\mathbf{H}$ of size $2^m \times 2^m$ that can be calculated recursively. Define the 1×1 WHT by the identity $\mathbf{H}_0 = \mathbf{1}$, then $\mathbf{H}_m$ for $m > 0$ can be obtained by:

$$\mathbf{H}_m = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{H}_{m-1} & \mathbf{H}_{m-1} \\ \mathbf{H}_{m-1} & -\mathbf{H}_{m-1} \end{pmatrix} = \mathbf{H}_1 \otimes \mathbf{H}_{m-1}, \tag{5.2}$$

where $\otimes$ denotes Kronecker product. Omitting the normalization factor $1/\sqrt{2}$, $\mathbf{H}$ only contains $+1$ and -1. Thus WHT is implementable without any multiplication or division, and it can be computed using a fast algorithm [37] to reduce the number of additions/subtractions from N^2 to $N \log N$. The 2D WHT (matrix version) is a straightforward extension of 1D WHT [98] and can be calculated by the following equation given matrix input $f(x, y)$ of size $N \times N$ ($2^m \times 2^m$):

$$H(u, v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) (-1)^z, \tag{5.3}$$

where

$$z = \sum_{i=1}^{m-1} (b_i(x) b_{m-1-i}(u) + b_i(y) b_{m-1-i}(v)).$$

Since the 2D WHT is separable and symmetric, it can be implemented as a sequence of two (row and column-wise) 1D WHT transforms in a fashion similar to that of the 2D DFT/FFT.

A new heterogeneous transform $\mathbf{H_P} = \mathbf{PH}$ is introduced by permuting WHT matrix $\mathbf{H}$ with a permutation matrix $\mathbf{P}$. A transform by $\mathbf{H_P}$ can be performed efficiently by first applying fast WHT and then permuting the result.

5.3.2 Neural Network in Transform Domains

To calculate the output of a single linear layer in a transform domain, ordinary schemes first apply the transform to both the input feature map and the filter kernel, then apply the inverse transform after element-wise multiplications of the transformed input map and transformed kernel. Given the input feature map $\mathbf{x}$, spatial kernel $\mathbf{w}$, binary-valued (± 1) transform matrix $\mathbf{H}$, and binary-valued inverse transform $\mathbf{H}^{-1}$, the output $\mathbf{Y}$ can be computed using the formula:

$$\mathbf{Y} = \mathbf{H}^{-\mathrm{T}}\Big[\big[\mathbf{H}^{\mathrm{T}}\mathbf{x}\mathbf{H}\big] \odot \big[\mathbf{H}^{\mathrm{T}}\mathbf{w}\mathbf{H}\big]\Big]\mathbf{H}^{-1}, \tag{5.4}$$

where $\odot$ denotes the element-wise multiplication. Note that using sparse $\mathbf{w}$ does not result in sparse element-wise multiplications in the transform domain.

This work proposes to train the network kernels directly in the transform domains by back-propagation given the transformed inputs. In this way, kernels are defined in the transform domains and there is no need to apply transforms on the spatial domain kernels. However, applying a transform to the entire activation feature map significantly increases the number of weights since the transform-domain kernel size must match the transformed activation size. To avoid this, the activation is divided into small overlapping patches and transforms are applied to each patch. Patches are overlapped to learn inter-patch dependency although it leads to more multiplications.

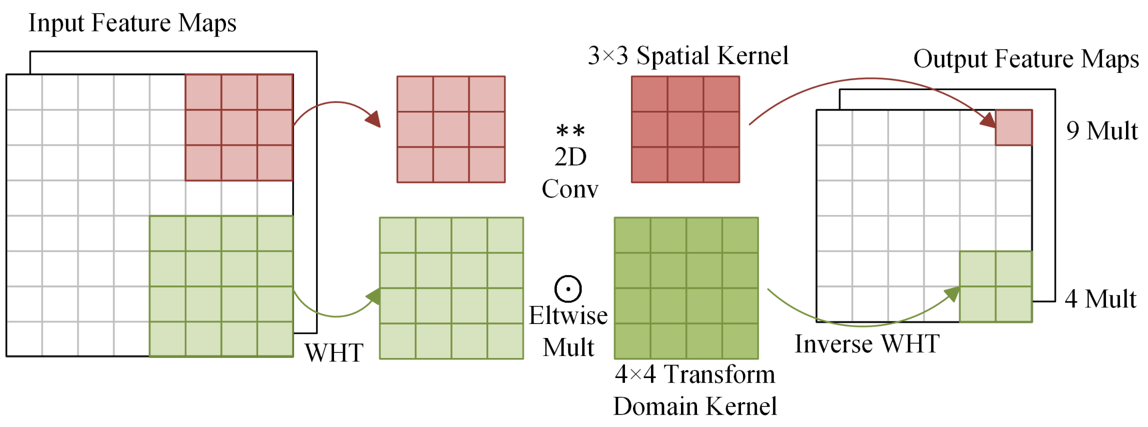


Figure 5.1: Comparison between a convolutional layer and WHT-domain linear layer. Red: Datapath of convolutional layer, needs 9 multiplications to compute 1 output. Green: Datapath of WHT linear layer, only needs 4 multiplications to compute 1 output.

A proper kernel size (i.e., transform size) needs to be chosen carefully to balance storage and computation requirements.

Although the proposed approach is generalizable to replace convolution kernels of any size, the goal is to replace all 3×3 convolutional layers in CNNs with WHT and permuted-WHT linear layers. These small convolution kernels are most commonly used and take a large portion of the overall CNN computations. To avoid a large increase in the number of transform-domain weights, the proposed approach operates based on a 4×4 patch extracted with stride of 2×2 from the $h \times w$ activation feature map and 4×4 (permuted) WHT is applied. The output patch size is 2×2 as it is obtained by taking the central 2×2 block after applying the inverse (permuted) WHT. For the inverse transform, the calculation of the output patch can be simplified by applying a 4×2 matrix $\mathbf{A}$, which consists of the middle two columns of the inverse (permuted) WHT matrix. Denoting the input patch by $\mathbf{x}$ and the 4×4 transform-domain kernel by $\mathbf{K}$, the output patch $\mathbf{D}$ of 4×4 WHT layer can be obtained by:

$$\mathbf{D} = \mathbf{A}^{\mathrm{T}}\Big[\big[\mathbf{H_P}^{\mathrm{T}}\mathbf{x}\mathbf{H_P}\big] \odot \mathbf{K}\Big]\mathbf{A}, \tag{5.5}$$

Since WHT can be efficiently computed with just additions/subtractions, the multiplications needed to compute one output are reduced from 9 to 4 in this approach. Finally, all the output patches are assembled into an $h \times w$ feature map for the next layer. A graphical illustration is given in Figure 5.1.

5.4 Learning for Hardware-Efficient Structure

Compared to an equivalent CNN, HTNN manages to reduce the number of multiplications by 2.25× with WHT-domain linear layers. But there is still significant redundancy in the proposed HTNN models, and the computation can be further reduced by eliminating the redundant connections. Inspired by Zhang's work [159], this problem can be treated as a non-convex optimization problem with combinatorial constraints specifying the sparsity requirements. Before going into detail on the proposed strategies, this section first defines the sparsity learning problem and briefly explains how it can be solved by ADMM.

Learning sparse weights in an N-layer DNN can be expressed as the following optimization problem:

$$\begin{aligned} &\min_{\{\mathbf{W}_i\}} f\left(\{\mathbf{W}_i\}\right), \\ &\text{subject to } \mathbf{W}_i \in \mathbf{S}_i, i = 1, \dots, N, \end{aligned} \tag{5.6}$$

where $f(\cdot)$ denotes the total loss function of DNN, $\mathbf{W}_i$ denotes the weights of i_{th} layer, and $\mathbf{S}_i$ is the desired sparse pattern set for i_{th} layer. Since $\mathbf{S}_1, \dots, \mathbf{S}_N$ are non-convex sets, it is difficult to solve this optimization problem directly. By introducing auxiliary variables $\mathbf{Z}_i$, this problem is equivalent to:

$$\begin{aligned} &\min_{\{\mathbf{W}_i\}} f\left(\{\mathbf{W}_i\}\right) + \sum_{i=1}^{N} g\left(\mathbf{Z}_i\right), \\ &\text{subject to } \mathbf{W}_i = \mathbf{Z}_i, \end{aligned} \tag{5.7}$$

where

$$g(\mathbf{Z}_i) = \begin{cases} 0 & \text{if } \mathbf{Z}_i \in \mathbf{S}_i \\ +\infty & \text{otherwise} \end{cases}.$$

ADMM [159] solves this non-convex problem (Equation 5.7) by first decomposing it into two sub-problems (Equation 5.8 and 5.9), and then alternatively solving one using the solution of the other in an iterative fashion. The first sub-problem is:

$$\min_{\{\mathbf{W}_i\}} f(\{\mathbf{W}_i\}) + \sum_{i=1}^{N} \frac{\rho}{2} \left\| \mathbf{W}_i - \mathbf{Z}_i^k + \mathbf{U}_i^k \right\|_F^2, \tag{5.8}$$

where $\mathbf{W}_i$ is the only optimization variable, scalar ρ is the penalty parameter, and $\mathbf{U}_i$ is a scaled dual variable updated by $\mathbf{U}_i^{k+1} = \mathbf{U}_i^k + \mathbf{W}_i^{k+1} - \mathbf{Z}_i^{k+1}$ in each iteration. This problem can be solved by stochastic gradient descent. The second sub-problem is:

$$\min_{\{\mathbf{Z}_i\}} \sum_{i=1}^{N} g(\mathbf{Z}_i) + \sum_{i=1}^{N} \frac{\rho}{2} \left\| \mathbf{W}_i^{k+1} - \mathbf{Z}_i + \mathbf{U}_i^k \right\|_F^2. \tag{5.9}$$

As $g(\cdot)$ is an indicator function, the analytical solution of the second subproblem can be obtained by:

$$\mathbf{Z}_i^{k+1} = \Pi_{\mathbf{S}_i}\left(\mathbf{W}_i^{k+1} + \mathbf{U}_i^k\right), \tag{5.10}$$

where $\Pi_{\mathbf{S}_i}(\cdot)$ denotes the Euclidean projection onto the set $\mathbf{S}_i$, i.e., applying the desired sparse pattern to $\mathbf{W}_i^{k+1} + \mathbf{U}_i^k$ with the smallest Euclidean distance for projection.

5.4.1 Structured Pruning for Sparse-Orthogonal Kernels in Heterogeneous Domains

Existing CNN weight pruning methods can be broadly separated into two categories: random position pruning [49, 50, 159] and structured pruning [144]. Random position pruning allows weights on any position to be pruned while structured pruning imposes a specific pattern such as an entire channel or kernel filter pruning. Random

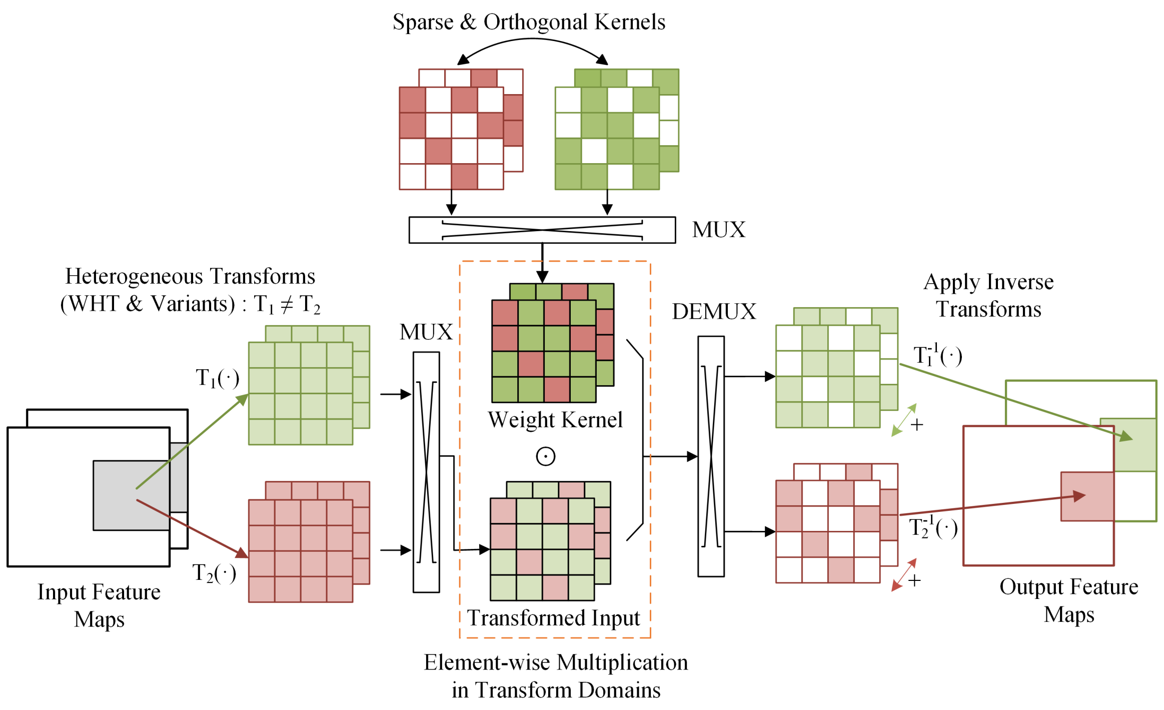


Figure 5.2: Heterogeneous-transform neural network (HTNN) linear layer using sparse-orthogonal kernels in heterogeneous transform domains.

position pruning methods can typically achieve a relatively high pruning rate but there exists significant overhead to compute sparse (randomly patterned) convolutions in software or hardware accelerators [48, 96, 161]. On the opposite, structured pruning methods can enhance computation efficiency but the achievable pruning rate is significantly lower.

This work searches and learns heterogeneous transforms that are grouped with orthogonal (i.e., non-overlapping non-zero positions) sparse weights to allow efficient parallelized computation of multiple sparse kernels in DNN hardware accelerators. The proposed approach is depicted in Figure 5.2. Without loss of generality, it is assumed that the input is multiple channels of 2D activation feature maps. Each input channel is processed in a patch-by-patch manner with overlaps between adjacent patches, then transformed by WHT (**H**) and pseudo-random permuted WHTs

$(\mathbf{H}_{\mathbf{P_1}}, \mathbf{H}_{\mathbf{P_2}}, ...)$. The transformed inputs from different transforms are element-wise multiplied with kernel weights in corresponding transform domains. Results obtained from multiple input channels are accumulated together to form an output channel. Finally inverse transforms are applied and results for different output channels are concatenated to form the final output feature maps.

To reduce the number of weights and computations in this procedure, An elaborate sparsity constraint $\mathbf{S}_i$ is imposed on kernel weights of the i_{th} layer. That is, the non-zero positions of the weights in kernels belonging to different transform domains are sparse and strictly non-overlapping (i.e., orthogonal) as shown in Figure 5.2. This sparse-orthogonality constraint allows an efficient hardware accelerator architecture where an array of multipliers performs parallel element-wise multiplications concurrently serving multiple channels with simple multiplexer (MUX) / de-multiplexer (DEMUX) logic at the input/output of the multiplier array as shown in Figure 5.2. Sparse convolution [48, 96, 161] does not have the same merit as the proposed strategy.

To preserve the network accuracy while achieving a high pruning rate, heterogeneous transforms with different permutations $(\mathbf{H}_{\mathbf{P_1}}, \mathbf{H}_{\mathbf{P_2}}, ...)$ are used deliberately, thus the position of important features in one transform domain is less likely to overlap with the ones in other transform domains. In this way, kernels associated with different transforms can (be trained to) have minimal impact on the other kernels when the sparse-orthogonality constraint is imposed. To find the preferred kernel grouping, the transform-domain layers without a sparse-orthogonality constraint or grouping are trained first. Then cross-correlations between kernel weights are calculated and groups that have low cross-correlation are selected as they tend to have less overlapping on critical weight positions.

5.4.2 Structured Sparse Digit Quantization with Canonical-Signed-Digit Representation

A new structured sparsity constraint for the number representation is proposed to further reduce the computation complexity of HTNN executed on hardware accelerators. Weight quantization is an efficient way to reduce computation complexity and weight storage. Previous approaches [49, 75] mainly focus on reducing the number of bits (i.e., precision) and also the number of possible values of quantized weights. A quantization strategy with K-mean clustering in [49] reduces the storage requirement by using non-uniform quantization while the computation is performed using a conventional floating point multiplier. Leng et al. [75] apply a uniform quantization strategy with ADMM that assumes conventional fixed/floating point multipliers. Here, this work proposes a canonical-signed-digit (CSD) [54] numbering system with a new structured sparsity constraint on digits of weight values. This approach replaces multiplications with much simpler additions/subtractions.

CSD representation [54] is a special way of encoding a value using ternary $\{1, -1, 0\}$ digits in which the number of non-zero digits is minimized. For example, an integer 30 requires 4 non-zero digits (bits) in the conventional binary representation (011110) while it only needs 2 non-zero digits in the CSD form of $(1, 0, 0, 0, -1, 0)$ since $30 = 32 - 2 = 2^5 - 2^1$ holds. The number of non-zero digits in CSD can vary depending on the number but it is always no more than what conventional binary representation needs. Since multiplying a number x by a power of 2 can be obtained by bit-shifting x (with much lower complexity than multiplication), fewer number of non-zero digits in the multiplicand translates to fewer shift and addition operations. To increase the benefit of this CSD representation, an additional sparse digit constraint is imposed to limit the number of non-zero digits to be strictly less than or equal to a predefined parameter k. Using $k = 2$ and 8-bit (digit) weights, a fixed-point weight w has the form of:

$$\begin{aligned} & w = c \cdot 2^a + d \cdot 2^b, \\ & a, b \in \{0, 1, ..., 7\} \quad \& \quad c, d \in \{1, -1, 0\}. \end{aligned} \tag{5.11}$$

Consequently, fixed-point multiplication $x \times w$ can be obtained by $((cx) \ll a) + ((dx) \ll b)$ where $x \ll a$ denotes bit-shifting x by a bits. Note that in this example, multiplication is replaced by a single addition/subtraction of bit-shifted versions of x because $c, d \in \{1, -1, 0\}$. The complexity reduction factor of this technique with $k = 2$ compared to conventional fixed point multiplication is determined by the relative complexity of a multiplier vs. adder & shifting. For hardware multipliers and adders, it is estimated that a $\geq$ 6-bit fixed point multiplier has at least 5$\times$ higher complexity (energy consumption) compared to an adder with the same precision based on Horowitz [59] and circuit synthesis results. The estimated energy for multiplication / addition is 0.2 / 0.03, and 1.1 / 0.05 pJ for 8-, and 16-bit operations, respectively, in a 40nm CMOS process. Note that there is no memory overhead to store CSD numbers because they can be stored in the two's complement form using only Q-bits. After reading weights from the memory, a simple decoder logic [54] can on-the-fly convert two's complement weights back to k-sparse CSD format. It is also possible to store a k-sparse CSD number by (entropy) encoding the tuple (a, b, c, d) of the expression (Equation 5.11).

Imposing k-digit sparsity on the CSD representation requires DNN training with non-uniform quantization. In the ADMM based back-propagation, all possible k-digit sparse CSD numbers are enumerated first and then unquantized numbers are projected to the nearest valid k-digit sparse CSD values during ADMM optimization (Equation 5.7). Thus, k-digit sparse CSD quantization is jointly applied with the

Algorithm 1 HTNN Linear Layer

```
Input: $n \times n$ input feature maps with $c_i$ input channels, $4 \times 4$ weight kernels with $c_o$ output channels and $n_t$ transforms.
for $pp = 1$ to $\frac{n}{2} \times \frac{n}{2}$ do
  for $k = 1$ to $\frac{c_o}{n_t}$ do
    for $t = 1$ to $n_t$ do
      Reset $acc_t$ $(4 \times 4)$ to 0.
    end for
    for $j = 1$ to $c_i$ do
      Load the $4 \times 4$ kernel $(j, k)$.
      for $t = 1$ to $n_t$ do
        if $k == 1$ then
          Apply transform $t$ to the $4 \times 4$ input patch of channel $j$ located at $pp$.
          Store transformed patch.
        else
          Load stored transformed $4 \times 4$ input patch of channel $j$ located at $pp$.
        end if
      end for
      $m \leftarrow$ MUX $n_t$ transformed patches of channel $j$ at $pp$ using the sparsity pattern for kernel $(j, k)$.
      $p \leftarrow$ Element-wise multiplication between $m$ and kernel$(j, k)$.
      for $t = 1$ to $n_t$ do
        $d_t \leftarrow$ DEMUX $p$ using sparsity pattern for kernel $(j, k)$.
        $acc_t \leftarrow acc_t + d_t$.
      end for
    end for
    for $t = 1$ to $n_t$ do
      Obtain the $2 \times 2$ output feature map patch O$(k, t)$ by applying inverse transform $t^{-1}$ to $acc_t$.
    end for
  end for
end for
```

learning of sparse-orthogonal weights described in Section 5.4.1.

5.5 HTNN Complexity Analysis

Algorithm 1 summarizes all computation steps involved in the HTNN linear layer. The overall datapath of the algorithm is depicted in Figure 5.3. c_i, c_o, n, and n_t denote the number of input channels, number of output channels, feature map size

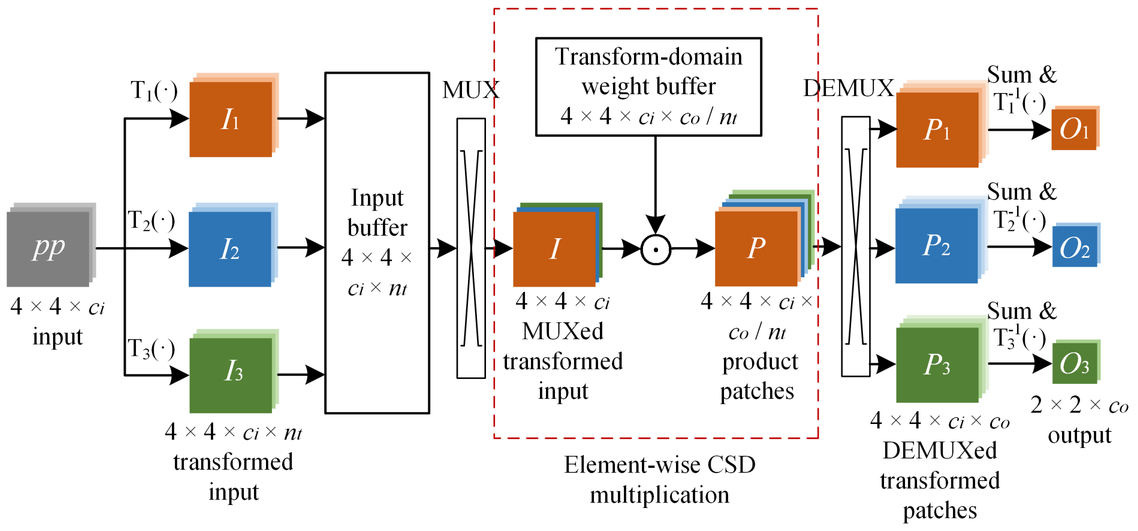

T1(·)
T2(·)
T3(·)
pp
4 × 4 × ci input
I1
I2
I3
4 × 4 × ci × nt transformed input
Input buffer 4 × 4 × ci × nt
MUX
Transform-domain weight buffer 4 × 4 × ci × co / nt
I
P
4 × 4 × ci MUXed transformed input
4 × 4 × ci × co / nt product patches
Element-wise CSD multiplication
DEMUX
P1
P2
P3
4 × 4 × ci × co DEMUXed transformed patches
Sum & T1^-1(·)
Sum & T2^-1(·)
Sum & T3^-1(·)
O1
O2
O3
2 × 2 × co output

Figure 5.3: Datapath of a single HTNN linear layer outer loop iteration with $n_t = 3$ transforms.

per dimension, and number of heterogeneous transforms employed in HTNN, respectively. The $c_i \times c_o$ orthogonal-sparse kernels of size 4×4 are trained offline and then merged/MUXed to $c_i \times c_o/n_t$ 'dense' kernels stored in the transform-domain weight buffer shown in Figure 5.3. For each input patch of size 4×4, the total number of computations required on transforms is n_t, not c_o, while $n_t \ll c_o$ typically holds. Since the transformed input patch is used c_o times, the overhead of input transform is negligible when $n_t \ll c_o$ holds. n_t sparse-orthogonal weight kernels and corresponding transformed input patches are effectively processed and merged at the same time before element-wise multiplications. The output is then DEMUXed to n_t output channels, which are accumulated across associated input channel indices. Finally, the accumulated outputs are transformed to 2×2 output patches. This process repeats for all the remaining patch locations. Note that the inverse transforms need to be applied c_o times per patch, regardless of c_i or n_t. As the number of kernels ($c_i \times c_o$) increases, the overhead of the inverse transform diminishes.

Table 5.1 lists the number of additions and multiplications involved in 4×4

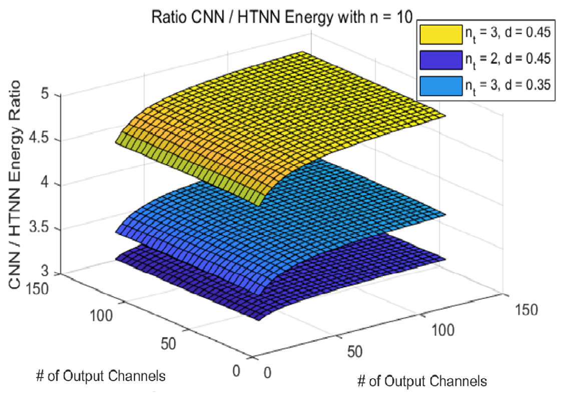
Operation type	Number of operations
HTNN WHT add	$8 \times 8 \times n_t \times c_i \times \frac{n^2}{4}$
HTNN CSD mult	$4 \times 4 \times c_i \times \frac{c_o}{n_t} \times \frac{n^2}{4}$
HTNN accum. add	$4 \times 4 \times (c_i - 1) \times \frac{c_o}{n_t} \times \frac{n^2}{4}$
HTNN 1-WHT add	$6 \times 6 \times c_o \times \frac{n^2}{4}$
Sparse CNN mult	$3 \times 3 \times c_i \times c_o \times n^2 \times d$
Sparse CNN add	$(3 \times 3 \times c_i - 1) \times c_o \times n^2 \times d$

c_o, c_i: output and input channel
n: feature map size
n_t: number of heterogeneous transforms
d: weight density of sparse convolution

Table 5.1: Number of operations performed in a single 4×4 HTNN linear layer and 3×3 sparse convolutional layer.

HTNN linear layer and 3×3 sparse convolutional layer with non-zero weight density of d. The average density of HTNN kernels is $1/n_t$. A 4×4 2D fast WHT requires $8 \times 8 = 64$ additions/subtractions. And the inverse transform to produce each 2×2 patch requires $6 \times 6 = 36$ additions/subtractions.

To compare the estimated energy consumption of the HTNN layer and sparse CNN layer, it is assumed that all the activations and weights are quantized to 8 bits. The sparse CNN layer employs 8-bit uniform quantization, thus involving 8-bit fixed-point multiplications. The HTNN layer uses 8-bit CSD multiplication with $k = 2$ digit sparsity, and each 8-bit CSD multiplication is replaced by a 16-bit addition since it involves adding two 16-bit values (bit-shifted from 8-bit values). Accumulations of outputs in both the CNN layer and HTNN layer are performed with 16-bit additions. Estimated energy consumption of the aforementioned fixed point additions and multiplications are obtained from circuit synthesis and post-APR SPICE simulation results in a 40 nm CMOS process. It is observed that sparse CSD multiplication has $\approx 2\times$ lower energy consumption compared to conventional fixed point multiplication (0.181



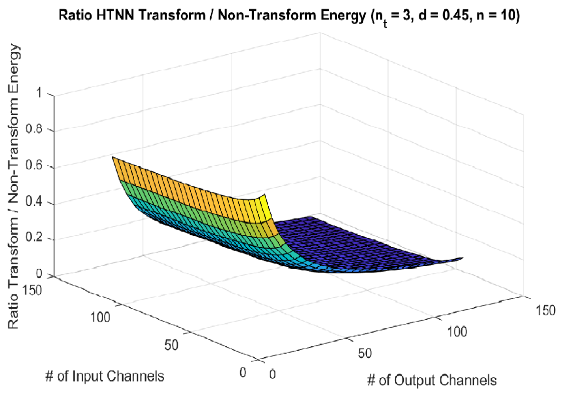


Figure 5.4: Left: HTNN layer vs. sparse CNN layer energy ratio with $n = 10$. Right: Transform vs. non-transform energy ratio in HTNN layer, $d = 0.45$, $n_t = 3$, and $n = 10$

pJ for CSD multiplication vs. 0.353 pJ for regular fixed point multiplication).

The left plot of Figure 5.4 shows the estimated energy ratio (assuming ideal HTNN and sparse CNN hardware accelerators) between the HTNN layer and sparse CNN layer with fixed input feature map size $n = 10$ for different c_i, c_o, n_t, and d. The energy efficiency gain of the HTNN layer over the sparse CNN layer is more evident (about 3.5 – 5×) when they involve a large number of input and/or output channels (c_i and/or c_o). It is worth noting that this analysis is based on ideal hardware accelerator assumption that only accounts for the number of operations related to non-zero weights in the sparse CNN, and ignores the substantial overhead of managing sparse convolutions [48, 96, 161]. The advantage of HTNN that allows a simple MUX/DEMUX structure with less hardware overhead to efficiently merge sparse-orthogonal kernels is not captured in Figure 5.4. The right plot of Figure 5.4 shows the estimated energy ratio between transforms (WHT and inverse WHT) and the other operations (element-wise multiplication and accumulation) in HTNN layer. The transform overhead becomes insignificant as c_i and c_o increase because a single transform is shared among multiple kernels.

5.6 Experiments

5.6.1 Models and Datasets

The proposed HTNN framework is tested with several famous CNN architectures on different datasets. To ensure that most convolutional layers are replaced by transform-domain linear layers, CNNs that heavily use 3×3 convolution kernels are chosen. In HTNN versions, all 3×3 convolutional layers are replaced by 4×4 HTNN linear layers. The architectures evaluated are: ResNet-20 [52], ResNet-18 [52], a lightweight VGGNet VGG-nagadomi [92], and a general convolution-pooling model ConvPool-CNN-C [116]. For dataset, ResNet-20 and VGG-nagadomi are tested on CIFAR-10 [67], ConvPool-CNN-C is tested on CIFAR-100 [67], and ResNet-18 is tested on ImageNet [104]. To replace all 3×3 convolutional layers with 4×4 transform-domain linear layers, the original ResNet-20 and ResNet-18 architectures are modified to use stride-1 convolutional layers followed by 2×2 max-pooling instead of stride-2 convolution. For ResNet-18, the last 2×2 max-pooling layer is also removed to keep the output spatial size of the last residual module even instead of odd (14×14 instead of 7×7). This ensures that 4×4 HTNN linear layer can still be applied to replace the last 3×3 convolutional layers. All experiments are implemented with the Pytorch [97] framework.

5.6.2 Evaluation of DNNs with Heterogeneous Transform Domains

First, three models are trained for ResNet-20, VGG-nagadomi and ConvPool-CNN-C architectures: conventional CNN, HTNN with a single WHT transform, and HTNN with multiple (permuted) WHT transforms. Since the first WHT-domain layer is most sensitive to weight density, only two heterogeneous transforms ($n_t = 2$) are applied to it. For the other layers, two heterogeneous transforms ($n_t = 2$) are

Model		ResNet-20			VGG-nagadomi			
		(sparse) CNN	1-WHT	**3-WHT**	(sparse) CNN	1-WHT	**2-WHT**	(sparse) Wino
Dense network	Accuracy	91.71%	91.73%	**91.90%**	93.27%	93.16%	**93.01%**	93.43%
	# of mult	47.6M	21.2M	**21.2M**	228M	101M	**101M**	101M
	# of add	47.4M	25.3M	**31.3M**	228M	106M	**108M**	103M
Learning kernel sparsity	Accuracy	91.46%	91.16%	**91.55%**	93.12%	92.42%	**93.06%**	93.33%
	Density	45.0%	35.2%	**35.2%**	45.0%	50.0%	**50.0%**	40%
	# of mult	21.4M	7.5M	**7.5M**	103M	50.5M	**50.5M**	40.4M
	# of add	21.3M	12.1M	**18.1M**	103M	55.2M	**58.2M**	43.2M
Learning kernel sparsity with quantized weights	Accuracy	91.44%	-	**91.56%**	93.08%	-	**93.01%**	n/a
	Density	45.0%	-	**24.5%**	45.0%	-	**34.9%**	40%
	Mult	21.4M	-	**0**	103M	-	**0**	40.4M
	CSD mult	0	-	**5.2M**	0	-	**35.2M**	0
	8-bit add	0	-	**11.0M**	0	-	**8.0M**	0
	16-bit add	21.3M	-	**5.0M**	103M	-	**35.0M**	43.2M
	Energy	9.05μJ	-	**1.45μJ**	43.57μJ	-	**8.93μJ**	17.29μJ

Model		ConvPool-CNN-C			
		(sparse) CNN	1-WHT	**3-WHT**	(sparse) Wino
Dense network	Accuracy	71.05%	71.09%	**71.14%**	69.75%
	# of mult	406M	180M	**180M**	180M
	# of add	406M	188M	**199M**	184M
Learning kernel sparsity	Accuracy	70.62%	70.12%	**70.71%**	69.65%
	Density	35.1%	33.4%	**33.4%**	60%
	# of mult	143M	60.1M	**60.1M**	108M
	# of add	143M	69.2M	**79.9M**	113M
Learning kernel sparsity with quantized weights	Accuracy	70.55%	-	**70.51%**	n/a
	Density	34.8%	-	**19.0%**	60%
	Mult	143M	-	**0**	108M
	CSD mult	0	-	**34.2M**	0
	8-bit add	0	-	**20.2M**	0
	16-bit add	143M	-	**33.9M**	113M
	Energy	60.49μJ	-	**8.85μJ**	46.03μJ

Model		ResNet-18		
		(sparse) CNN	**3-WHT**	(sparse) Wino
Dense network	Accuracy	69.8%	**71.2%**	66.8%
	# of mult	1676M	**1285M**	1285M
	# of add	1674M	**1374M**	1320M
Learning kernel sparsity	Accuracy	69.6%	**70.8%**	66.6%
	Density	50%	**33.7%**	35%
	# of mult	838M	**433M**	450M
	# of add	837M	**527M**	490M
Learning kernel sparsity with quantized weights	Accuracy	69.2%	**69.8%**	n/a
	Density	50%	**17.7%**	35%
	Mult	838M	**0**	450M
	CSD mult	0	**227M**	0
	8-bit add	0	**97M**	0
	16-bit add	837M	**226M**	490M
	Energy	354.4μJ	**58.3μJ**	193.2μJ

Table 5.2: Top-1 accuracy and energy estimation for CNN models, proposed HTNN models, and sparse-Winograd models [81]. Top: ResNet-20 and VGG-nagadomi on CIFAR-10. Middle: ConvPool-CNN-C on CIFAR-100. Bottom: ResNet-18 on ImageNet.

used for VGG-nagadomi whereas three transforms are used ($n_t = 3$) for ResNet-20 and ConvPool-CNN-C. For ResNet-18 architecture, two models are trained: the conventional CNN and HTNN with multiple (permuted) WHT transforms. The first convolutional layer of ResNet-18 is kept since it uses 7×7 kernels. All the rest 3×3 convolutional layers are replaced by 4×4 HTNN layers with three heterogeneous transforms ($n_t = 3$). n_t is empirically selected according to CNN pruning results with deep compression [49].

The top-1 accuracy on the test datasets is shown in Table 5.2. The proposed HTNN models can achieve similar or even better accuracy compared to spatial domain CNNs of all four network architectures. To compare computational complexity, the total number of multiplications and additions are calculated on all 3×3 convolutional layers and 4×4 HTNN layers including the overhead of transforms. The total number of multiplication and addition operations are reported in Table 5.2.

5.6.3 Learning and Training for Hardware-Efficient Structures

For learning the sparse-orthogonal kernels, the proposed method is applied to all WHT-domain HTNN models. For sparse CNN baselines, the deep compression strategy described in [49] is applied. Sparse-Winograd pruning results on VGG-nagadomi, ConvPool-CNN-C and ResNet-18 models from [81] are also added for comparison. All results are summarized in Table 5.2.

There is around 0.8% accuracy drop from the single-WHT models while almost no accuracy drop from the multi-WHT models after applying the proposed method. It confirms that using multiple heterogeneous transforms is beneficial to successfully learn the desired sparse-orthogonal kernel structures avoiding collisions on important non-zero weight positions. Besides, the proposed strategy on HTNN models achieves comparable compression ratios with CNN deep compression pruning. As shown in the

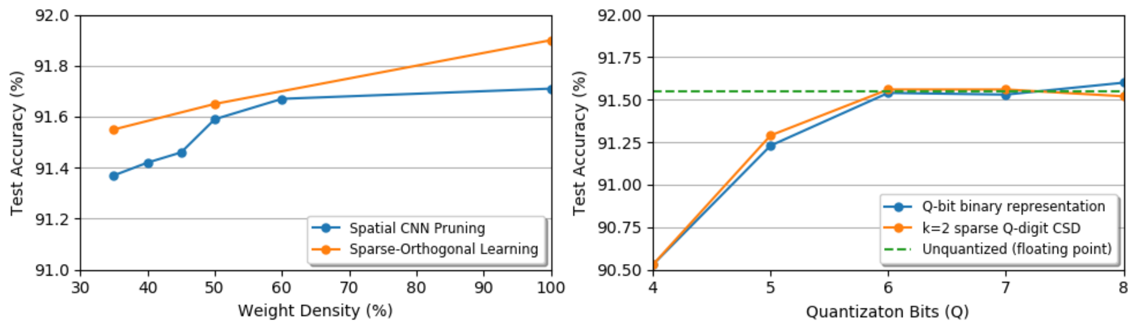


Figure 5.5: Hardware-efficient training for ResNet-20. Left: Test accuracy as a function of weight density for CNN pruning vs. sparse-orthogonal learning for HTNN. Right: HTNN accuracy as a function of quantization bits for Q-bit binary representation and $k = 2$ sparse Q-digit CSD.

left plot of Figure 5.5, the 4×4 HTNN layers achieve 35% weight density (not exactly 1/3 because the first layer uses $n_t = 2$) for the 3-WHT ResNet-20 model while spatial CNN pruning achieves 45% density on 3×3 convolutional layers of CNN ResNet-20 with similar test accuracy.

All four HTNN models are tested with $k = 2$ sparse CSD quantization. It does not incur big accuracy loss compared to learned sparse kernel models with $Q = 6$. The right plot in Figure 5.5 shows the test accuracy vs. quantization bits for both Q-bit binary representation and $k = 2$ sparse CSD representation (also with Q digits). Both quantization methods can achieve the same accuracy where the $k = 2$ sparse CSD quantization offers a factor of 4× complexity reduction for Q = 6. it is observed that plenty of near-zero weights are converted to zeros during the CSD quantization process, thus significantly lowering the non-zero weight density (Table 5.2). However, weight densities remain almost the same for sparse CNN models after quantization since weights with smaller magnitudes are already set to zero during the pruning phase.

To compare the final computation complexity of the CNN models and the transform-domain HTNN models, 8-bit uniform quantization is applied to the spatial CNNs pruned with deep compression, and $k = 2$ sparse CSD (Q = 6) quantization to the HTNN models with sparse-orthogonal kernels. All activations are uniformly quantized to 8 bits for HTNN and CNN. The estimated energies for 8- & 16-bit addition, 8-bit multiplication and 8-bit CSD multiplication are 0.014, 0.070, 0.353 and 0.181 pJ, respectively, in a 40nm CMOS process (from post-APR SPICE simulations). For each model, the total energy is the summation of the energies of all operations estimated using the method described in Section 5.5. For 3×3 convolutional layers, HTNNs (including transform overhead) can achieve $4.9 - 6.8\times$ complexity (energy) reduction compared to quantized sparse spatial CNN models with almost no accuracy drop on the test datasets. In this comparison, the benefit from simplified hardware to handle sparse-orthogonal kernels in HTNN is not included although the overhead of implementing sparse convolution is known to be substantial [48, 96, 161] for sparse CNN. Table 5.2 shows the energy estimation of sparse-Winograd VGG-nagadomi, ConvPool-CNN-C and ResNet-18 models with 8-bit uniform quantized weights. Liu et al. [81] did not provide quantization results and it was not shown whether the sparse-Winograd model can preserve original accuracy with 8-bit quantization. The estimated energy consumption of these sparse-Winograd models yields $1.9 - 5.2\times$ higher energy compared to the HTNN models employing $k = 2, Q = 6$ CSD quantization.

5.7 Summary

This work presents a hardware-efficient neural network, HTNN, that uses heterogeneous transform domains with sparse-orthogonal weights replacing convolutions with element-wise multiplications. Feature maps are transformed into heterogeneous transform domains via fast WHT and its variants, where spatial convolutions are replaced by efficient element-wise multiplications. An effective strategy is proposed to learn hardware-friendly structures with sparse-orthogonal kernels. Furthermore, structured sparse-digit quantization with CSD representation is applied to the sparse-orthogonal weights to substitute multiplications with additions/subtractions. Experimental results strongly demonstrate that neural networks can be trained in heterogeneous transform domains and computation complexity is significantly reduced with learned sparse-orthogonal kernels and structured sparse CSD quantization. Up to 6.8$\times$ complexity reduction is achievable without loss of network accuracy compared to CNNs using sparse convolution with optimally pruned weights. It is expected to have a higher complexity reduction with bigger/more transforms. More aggressive computation reduction could be achieved by using variable kernel sharing rates for different layers while maintaining overall model accuracy.

CHAPTER VI

Conclusion and Future Work

6.1 Conclusion

This dissertation focuses on low-power localization systems, especially hardware-efficient deep learning based localization systems. Deep neural networks are powerful tools that achieve tremendous success in various computer vision and machine learning tasks, yet the complexity of modern deep learning models is enormous and these models rely on powerful computing platforms such as GPUs and tensor processing units (TPUs) for successful deployment. Implementing such big deep learning models on energy-constrained mobile IoT platforms is a big challenge. This dissertation explores four different yet effective ways to balance system performance and power consumption. That is reducing the complexity of deep learning models, adaptive sensor modality selection and fusion, turning to low-power sensors, and system hardware-software co-design for specialized accelerators.

In Chapter II, a real-time visual SLAM system with low power consumption is investigated through hardware-software co-design. This SLAM system is specially optimized for hardware-efficient VLSI implementation while maintaining reasonable performance. Compared to state-of-the-art SLAM systems, the proposed system reduces its overall power consumption and on-chip memory usage to only 200 micro watts and 13 MB. With much lower power consumption and memory usage, the proposed system is capable for mobile IoT applications, and it is already ported on a low-power SLAM ASIC accelerator.

In Chapter III, an energy-efficient deep learning based VIO system with excellent accuracy is obtained by reducing the neural network complexity and adaptive sensor modality selection and fusion. A one-shot neural architecture search approach is proposed to search for the most efficient visual encoder with low complexity and latency without sacrificing the system performance. Besides, the system computation overhead and power consumption are reduced by opportunistically disabling the visual modality on the fly when visual information is not critical for maintaining pose estimation accuracy. Combining both techniques, the optimal model can save up to 99.1% computation, and run in real-time on a single laptop CPU core. The proposed strategies are also model-agnostic and can be easily adapted to other deep VIO systems.

In Chapter IV, an efficient end-to-end deep learning framework for accurate source separation and localization in multi-source environments is presented. The system only relies on a passive low-power microphone sensor array to collect mixed audio signals from surrounding sound sources and estimate the DOA information of each source. The separated signals of the sources and their location information are jointly optimized through self-supervision and adversarial machine learning techniques. The proposed learning framework successfully improves the source separation quality as well as the localization accuracy. Compared to existing methods, it achieves superior/similar performance while producing interpretable intermediate information such as separated sources and TDOA information between microphones, and it is generalizable to other source separation models.

In Chapter V, a hardware-efficient heterogeneous transform-domain deep neural network (HTNN) is introduced. HTNN uses heterogeneous transform domains with hardware-friendly sparse-orthogonal weights to replace convolutions with element-wise multiplications. Structured sparse-digit quantization with CSD representation is applied to the sparse-orthogonal weights to substitute multiplications with additions/subtractions. It is demonstrated that neural networks can be trained in heterogeneous transform domains, and computation complexity is significantly reduced with learned sparse-orthogonal kernels and structured sparse CSD quantization without performance degradation. More aggressive computation reduction could be achieved by using bigger/more transforms and variable kernel sharing rates for different layers without harming the overall accuracy.

6.2 Future Work

There are several directions to explore in the development of low-power hardware-efficient localization systems. With the development of new network architectures and new types of layers, reducing the complexity of neural networks is a continuous research topic. The performance of existing approaches in the literature, such as neural network compression and transformed domain neural networks, is still far from perfect. It is interesting to extend the existing algorithms to optimize DNNs for energy-constrained mobile applications, especially low-power localization systems. For neural architecture search based methods, it is possible to extend the search space to include different types of layers, number of quantization bits, and operation bits, in addition to the CNN kernel sizes, number of channels, and depths. For transform-domain neural networks, applying heterogeneous transforms of different types, numbers or sizes is an exciting direction to explore.

Another interesting direction is to apply adaptive sensor modality selection and fusion on other tasks, such as 3D object detection. 3D object detection is a hot topic in computer vision and one of the core bases for path planning and collision avoidance in autonomous driving systems. State-of-the-art deep learning based 3D object detection systems typically employ multiple sensors and involve massive computation, making them unsuitable for energy-constrained mobile platforms such as unmanned aerial vehicles (UAVs) and small mobile robots. An interesting direction is to extend the proposed adaptive sensor modality selection and fusion technique to different sensor modalities, such as lidar and camera in a typical 3D object detection system. By reducing the usage of the energy-intensive lidar sensor, overall system power consumption can be reduced, enabling the deployment of such systems on mobile platforms.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Sharath Adavanne, Archontis Politis, Joonas Nikunen, and Tuomas Virtanen. Sound event localization and detection of overlapping sources using convolutional recurrent neural networks. *IEEE Journal of Selected Topics in Signal Processing*, 13(1):34–48, 2018.

[2] Yasin Almalioglu, Mehmet Turan, Alp Eren Sari, Muhamad Risqi U Saputra, Pedro PB de Gusmão, Andrew Markham, and Niki Trigoni. Selfvio: Self-supervised deep monocular visual-inertial odometry and depth estimation. *arXiv preprint arXiv:1911.09968*, 2019.

[3] Adrien Angeli, David Filliat, Stéphane Doncieux, and Jean-Arcady Meyer. Fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions on Robotics*, 24(5):1027–1037, 2008.

[4] Mohammad OA Aqel, Mohammad H Marhaban, M Iqbal Saripan, and Napsiah Bt Ismail. Review of visual odometry: types, approaches, challenges, and applications. *SpringerPlus*, 5(1):1–26, 2016.

[5] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *International Conference on Learning Representations (ICLR)*, 2017.

[6] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006.

[7] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016.

[8] Dieter Balemans, Wim Casteels, Simon Vanneste, Jens de Hoog, Siegfried Mercelis, and Peter Hellinckx. Resource efficient sensor fusion by knowledge-based network pruning. *Internet of Things*, 11:100231, 2020.

[9] Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *Bmvc*, volume 1, page 3, 2016.

[10] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for efficient inference. In *International Conference on Machine Learning (ICML)*, pages 527–536. PMLR, 2017.

[11] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[12] Matthew Brown, Gang Hua, and Simon Winder. Discriminative learning of local image descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):43–57, 2010.

[13] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[14] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

[15] Xing Cai, Lanqing Zhang, Chengyuan Li, Ge Li, and Thomas H Li. Vonas: Network design in visual odometry using neural architecture search. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 727–735, 2020.

[16] Víctor Campos, Brendan Jou, Xavier Giró-i Nieto, Jordi Torres, and Shih-Fu Chang. Skip rnn: Learning to skip state updates in recurrent neural networks. *arXiv preprint arXiv:1708.06834*, 2017.

[17] Jose A Castellanos, José MM Montiel, José Neira, and Juan D Tardós. The spmap: A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 15(5):948–952, 1999.

[18] Changhao Chen, Stefano Rosa, Chris Xiaoxuan Lu, Niki Trigoni, and Andrew Markham. Selectfusion: A generic framework to selectively learn multisensory fusion. *arXiv preprint arXiv:1912.13077*, 2019.

[19] Changhao Chen, Stefano Rosa, Yishu Miao, Chris Xiaoxuan Lu, Wei Wu, Andrew Markham, and Niki Trigoni. Selective sensor fusion for neural visual-inertial odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10542–10551, 2019.

[20] Jingjing Chen, Qirong Mao, and Dong Liu. Dual-path transformer network: Direct context-aware modeling for end-to-end monaural speech separation. *arXiv preprint arXiv:2007.13975*, 2020.

[21] Minghao Chen, Jianlong Fu, and Haibin Ling. One-shot neural ensemble architecture search by diversity-guided search space shrinking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16530–16539, 2021.

[22] Yu Chen, Bowen Liu, Pierre Abillama, and Hun-Seok Kim. Htnn: deep learning in heterogeneous transform domains with sparse-orthogonal weights. In *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 1–6. IEEE, 2021.

[23] Yu Chen, Bowen Liu, Zijian Zhang, and Hun-Seok Kim. An end-to-end deep learning framework for multiple audio source separation and localization. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 736–740. IEEE, 2022.

[24] Yu Chen, Mingyu Yang, and Hun-Seok Kim. Search for efficient deep visual-inertial odometry through neural architecture search. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.

[25] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.

[26] Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, and Niki Trigoni. Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[27] Joris Cosentino, Manuel Pariente, Samuele Cornell, Antoine Deleforge, and Emmanuel Vincent. Librimix: An open-source dataset for generalizable speech separation. *arXiv preprint arXiv:2005.11262*, 2020.

[28] Marco Crocco, Marco Cristani, Andrea Trucco, and Vittorio Murino. Audio surveillance: A systematic review. *ACM Computing Surveys (CSUR)*, 48(4):1–46, 2016.

[29] Igor Cvišić, Josip Ćesić, Ivan Marković, and Ivan Petrović. Soft-slam: Computationally efficient stereo visual simultaneous localization and mapping for autonomous unmanned aerial vehicles. *Journal of Field Robotics*, 35(4):578–595, 2018.

[30] Giovanni De Michell and Rajesh K Gupta. Hardware/software co-design. *Proceedings of the IEEE*, 85(3):349–365, 1997.

[31] Nicolas de Palézieux, Tobias Nägeli, and Otmar Hilliges. Duo-vio: Fast, light-weight, stereo inertial odometry. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2237–2242. IEEE, 2016.

[32] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766, 2015.

[33] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine*, 13(2):99–110, 2006.

[34] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[35] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625, 2017.

[36] Jakob Engel, Jörg Stückler, and Daniel Cremers. Large-scale direct slam with stereo cameras. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1935–1942. IEEE, 2015.

[37] Bernard J. Fino and V. Ralph Algazi. Unified matrix treatment of the fast walsh-hadamard transform. *IEEE Transactions on Computers*, 25(11):1142–1146, 1976.

[38] Philipp Fischer, Alexey Dosovitskiy, and Thomas Brox. Descriptor matching with convolutional neural networks: a comparison to sift. *arXiv preprint arXiv:1405.5769*, 2014.

[39] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22. IEEE, 2014.

[40] Xiang Gao, Rui Wang, Nikolaus Demmel, and Daniel Cremers. Ldso: Direct sparse odometry with loop closure. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2198–2204. IEEE, 2018.

[41] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361. IEEE, 2012.

[42] Peter W Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84, 1990.

[43] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3828–3838, 2019.

[44] Vineet Gokhale, Gerardo Moyers Barrera, and R Venkatesha Prasad. Feel: fast, energy-efficient localization for autonomous indoor vehicles. In *IEEE International Conference on Communications*, pages 1–6. IEEE, 2021.

[45] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems (NeurIPS)*, 27, 2014.

[46] Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

[47] Liming Han, Yimin Lin, Guoguang Du, and Shiguo Lian. Deepvio: Self-supervised deep learning of monocular visual inertial odometry using 3d geometric constraints. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6906–6913. IEEE, 2019.

[48] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: Efficient inference engine on compressed deep neural network. *ACM SIGARCH Computer Architecture News*, 44(3):243–254, 2016.

[49] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations (ICLR)*, 2016.

[50] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in Neural Information Processing Systems (NeurIPS)*, 28, 2015.

[51] Christian Hansen, Casper Hansen, Stephen Alstrup, Jakob Grue Simonsen, and Christina Lioma. Neural speed reading with structural-jump-lstm. *arXiv preprint arXiv:1904.00761*, 2019.

[52] 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.

[53] Weipeng He, Petr Motlicek, and Jean-Marc Odobez. Deep neural networks for multiple speaker detection and localization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 74–79. IEEE, 2018.

[54] Reid M Hewlitt and ES Swartzlantler. Canonical signed digit representation for fir digital filters. In *IEEE Workshop on Signal Processing Systems (SIPS)*, pages 416–426. IEEE, 2000.

[55] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Proceedings of the IEEE Conference on*

Computer Vision and Pattern Recognition (CVPR), volume 2, pages 807–814. IEEE, 2005.

[56] Steven A Holmes, Georg Klein, and David W Murray. An o (n^2) square root unscented kalman filter for visual simultaneous localization and mapping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(7):1251–1263, 2008.

[57] Euntae Hong and Jongwoo Lim. Visual inertial odometry using coupled nonlinear optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6879–6885. IEEE, 2017.

[58] Injoon Hong, Gyeonghoon Kim, Youchang Kim, Donghyun Kim, Byeong-Gyu Nam, and Hoi-Jun Yoo. A 27 mw reconfigurable marker-less logarithmic camera pose estimation engine for mobile augmented reality processor. *IEEE Journal of Solid-State Circuits*, 50(11):2513–2523, 2015.

[59] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14. IEEE, 2014.

[60] Weizhe Hua, Yuan Zhou, Christopher M De Sa, Zhiru Zhang, and G Edward Suh. Channel gating neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[61] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens Van Der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017.

[62] Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in Statistics*, pages 492–518. Springer, 1992.

[63] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[64] J Kin, M Gupta, and WH Mangione Smith. Application specific integrated circuits. 1997.

[65] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[66] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234. IEEE, 2007.

[67] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[68] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 25, 2012.

[69] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g 2 o: A general framework for graph optimization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3607–3613. IEEE, 2011.

[70] Henry O. Kunz. On the equivalence between one-dimensional discrete walsh-hadamard and multidimensional discrete fourier transforms. *IEEE Transactions on Computers*, 28(03):267–268, 1979.

[71] Guillaume Lathoud, Jean-Marc Odobez, and Daniel Gatica-Perez. Av16. 3: An audio-visual corpus for speaker localization and tracking. In *International Workshop on Machine Learning for Multimodal Interaction*, pages 182–195. Springer, 2004.

[72] Andrew Lavin and Scott Gray. Fast algorithms for convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4013–4021, 2016.

[73] Jonathan Le Roux, Scott Wisdom, Hakan Erdogan, and John R Hershey. Sdr–half-baked or well done? In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 626–630. IEEE, 2019.

[74] 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.

[75] Cong Leng, Zesheng Dou, Hao Li, Shenghuo Zhu, and Rong Jin. Extremely low bit neural network: Squeeze the last bit out with admm. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[76] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.

[77] Mingyang Li and Anastasios I Mourikis. High-precision, consistent ekf-based visual-inertial odometry. *The International Journal of Robotics Research*, 32(6):690–711, 2013.

[78] Ziyun Li, Yu Chen, Luyao Gong, Lu Liu, Dennis Sylvester, David Blaauw, and Hun-Seok Kim. An 879gops 243mw 80fps vga fully visual cnn-slam processor for wide-range autonomous exploration. In *IEEE International Solid-State Circuits Conference (ISSCC)*, pages 134–136. IEEE, 2019.

[79] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 82–92, 2019.

[80] Li Liu, Ge Li, and Thomas H Li. Atvio: Attention guided visual-inertial odometry. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4125–4129. IEEE, 2021.

[81] Xingyu Liu, Jeff Pool, Song Han, and William J Dally. Efficient sparse-winograd convolutional neural networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[82] Heinrich W Löllmann, Christine Evers, Alexander Schmidt, Heinrich Mellmann, Hendrik Barfuss, Patrick A Naylor, and Walter Kellermann. The locata challenge data corpus for acoustic source localization and tracking. In *IEEE 10th Sensor Array and Multichannel Signal Processing Workshop (SAM)*, pages 410–414. IEEE, 2018.

[83] Manolis IA Lourakis and Antonis A Argyros. Sba: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software (TOMS)*, 36(1):1–30, 2009.

[84] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1150–1157. IEEE, 1999.

[85] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[86] Michael Mathieu, Mikael Henaff, and Yann LeCun. Fast training of convolutional networks through ffts. In *International Conference on Learning Representations (ICLR)*, 2014.

[87] Yue Meng, Chung-Ching Lin, Rameswar Panda, Prasanna Sattigeri, Leonid Karlinsky, Aude Oliva, Kate Saenko, and Rogerio Feris. Ar-net: Adaptive frame resolution for efficient action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 86–104. Springer, 2020.

[88] Yue Meng, Rameswar Panda, Chung-Ching Lin, Prasanna Sattigeri, Leonid Karlinsky, Kate Saenko, Aude Oliva, and Rogerio Feris. Adafuse: Adaptive temporal fusion network for efficient action recognition. *arXiv preprint arXiv:2102.05775*, 2021.

[89] Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning. *The Journal of Machine Learning Research*, 21(1):5183–5244, 2020.

[90] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[91] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[92] Nagadomi. Code for kaggle-cifar10 competition, 5th place. https://github.com/nagadomi/kaggle-cifar10-torch7, 2014.

[93] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–I. Ieee, 2004.

[94] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[95] Rameswar Panda, Chun-Fu Richard Chen, Quanfu Fan, Ximeng Sun, Kate Saenko, Aude Oliva, and Rogerio Feris. Adamml: Adaptive multi-modal learning for efficient video recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7576–7585, 2021.

[96] Angshuman Parashar, Minsoo Rhu, Anurag Mukkara, Antonio Puglielli, Rangharajan Venkatesan, Brucek Khailany, Joel Emer, Stephen W Keckler, and William J Dally. Scnn: An accelerator for compressed-sparse convolutional neural networks. *ACM SIGARCH Computer Architecture News*, 45(2):27–40, 2017.

[97] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[98] William K Pratt, Julius Kane, and Harry C Andrews. Hadamard transform image coding. *Proceedings of the IEEE*, 57(1):58–68, 1969.

[99] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[100] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4780–4789, 2019.

[101] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML)*, pages 1278–1286. PMLR, 2014.

[102] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 430–443. Springer, 2006.

[103] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571. Ieee, 2011.

[104] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015.

[105] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018.

[106] Hiroshi Sawada, Ryo Mukai, Shoko Araki, and S Malcino. Multiple source localization using independent component analysis. In *IEEE Antennas and Propagation Society International Symposium*, volume 4, pages 81–84. IEEE, 2005.

[107] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE robotics & automation magazine*, 18(4):80–92, 2011.

[108] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016.

[109] Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. Neural speed reading via skim-rnn. *arXiv preprint arXiv:1711.02085*, 2017.

[110] E Jared Shamwell, Sarah Leung, and William D Nothwang. Vision-aided absolute trajectory estimation using an unsupervised deep network with online error correction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2524–2531. IEEE, 2018.

[111] Kazuki Shimada, Yuichiro Koyama, Naoya Takahashi, Shusuke Takahashi, and Yuki Mitsufuji. Accdoa: Activity-coupled cartesian direction of arrival representation for sound event localization and detection. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 915–919. IEEE, 2021.

[112] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 118–126, 2015.

[113] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[114] Vishwanath A Sindagi, Yin Zhou, and Oncel Tuzel. Mvx-net: Multimodal voxelnet for 3d object detection. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 7276–7282. IEEE, 2019.

[115] Randall C Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The international Journal of Robotics Research*, 5(4):56–68, 1986.

[116] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. In *International Conference on Learning Representations workshop (ICLR)*, 2015.

[117] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-u-net: A multi-scale neural network for end-to-end audio source separation. *arXiv preprint arXiv:1806.03185*, 2018.

[118] Hauke Strasdat, José MM Montiel, and Andrew J Davison. Visual slam: why filter? *Image and Vision Computing*, 30(2):65–77, 2012.

[119] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 573–580. IEEE, 2012.

[120] Yu-Chi Su, Keng-Yen Huang, Tse-Wei Chen, Yi-Min Tsai, Shao-Yi Chien, and Liang-Gee Chen. A 52 mw full hd 160-degree object viewpoint recognition soc with visual vocabulary processor for wearable vision applications. *IEEE Journal of Solid-State Circuits*, 47(4):797–809, 2012.

[121] Cem Subakan, Mirco Ravanelli, Samuele Cornell, Mirko Bronzi, and Jianyuan Zhong. Attention is all you need in speech separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 21–25. IEEE, 2021.

[122] Harshavardhan Sundar, Thippur V Sreenivas, and Chandra Sekhar Seelamantula. Tdoa-based multiple acoustic source localization without association ambiguity. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(11):1976–1990, 2018.

[123] Harshavardhan Sundar, Weiran Wang, Ming Sun, and Chao Wang. Raw waveform based end-to-end deep convolutional network for spatial localization of multiple acoustic sources. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4642–4646. IEEE, 2020.

[124] Niko Sunderhauf, Sven Lange, and Peter Protzel. Using the unscented kalman filter in mono-slam with inverse depth parametrization for autonomous airship control. In *IEEE International Workshop on Safety, Security and Rescue Robotics*, pages 1–6. IEEE, 2007.

[125] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

[126] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, pages 6105–6114. PMLR, 2019.

[127] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *International Conference on Pattern Recognition (ICPR)*, pages 2464–2469. IEEE, 2016.

[128] Ana M Torres, Maximo Cobos, Basilio Pueo, and Jose J Lopez. Robust acoustic source localization based on modal beamforming and time–frequency processing using circular microphone arrays. *The Journal of the Acoustical Society of America*, 132(3):1511–1520, 2012.

[129] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *International Workshop on Vision Algorithms*, pages 298–372. Springer, 1999.

[130] Spyros G Tzafestas. *Introduction to mobile robot control.* Elsevier, 2013.

[131] Efthymios Tzinis, Shrikant Venkataramani, Zhepei Wang, Cem Subakan, and Paris Smaragdis. Two-step sound source separation: Training on learned latent targets. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 31–35. IEEE, 2020.

[132] Efthymios Tzinis, Zhepei Wang, and Paris Smaragdis. Sudo rm-rf: Efficient networks for universal audio source separation. In *IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2020.

[133] Efthymios Tzinis, Scott Wisdom, John R Hershey, Aren Jansen, and Daniel PW Ellis. Improving universal sound separation using sound classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 96–100. IEEE, 2020.

[134] Shimon Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 203(1153):405–426, 1979.

[135] J-M Valin, François Michaud, Jean Rouat, and Dominic Létourneau. Robust sound source localization using a microphone array on a mobile robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 1228–1233. IEEE, 2003.

[136] Nicolas Vasilache, Jeff Johnson, Michael Mathieu, Soumith Chintala, Serkan Piantino, and Yann LeCun. Fast convolutional nets with fbfft: A gpu performance evaluation. *arXiv preprint arXiv:1412.7580*, 2014.

[137] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–18, 2018.

[138] Juan Manuel Vera-Diaz, Daniel Pizarro, and Javier Macias-Guarasa. Towards end-to-end acoustic localization using deep learning: From audio signals to source position coordinates. *Sensors*, 18(10):3418, 2018.

[139] Vinay Verma, Nikita Agarwal, and Nitin Khanna. Dct-domain deep convolutional neural networks for multiple jpeg compression classification. *Signal Processing: Image Communication*, 67:22–33, 2018.

[140] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050. IEEE, 2017.

[141] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 409–424, 2018.

[142] Yulin Wang, Zhaoxi Chen, Haojun Jiang, Shiji Song, Yizeng Han, and Gao Huang. Adaptive focus for efficient video recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 16249–16258, 2021.

[143] Yunhe Wang, Chang Xu, Shan You, Dacheng Tao, and Chao Xu. Cnnpack: Packing convolutional neural networks in the frequency domain. *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016.

[144] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016.

[145] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

[146] Shmuel Winograd. *Arithmetic complexity of computations*, volume 33. Siam, 1980.

[147] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8817–8826, 2018.

[148] Zuxuan Wu, Caiming Xiong, Chih-Yao Ma, Richard Socher, and Larry S Davis. Adaframe: Adaptive frame selection for fast video recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1278–1287, 2019.

[149] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1492–1500, 2017.

[150] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018.

[151] Fei Xue, Qiuyuan Wang, Xin Wang, Wei Dong, Junqiu Wang, and Hongbin Zha. Guided feature selection for deep visual odometry. In *Asian Conference on Computer Vision*, pages 293–308. Springer, 2018.

[152] Fei Xue, Xin Wang, Shunkai Li, Qiuyuan Wang, Junqiu Wang, and Hongbin Zha. Beyond tracking: Selecting memory and refining poses for deep visual odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8575–8583, 2019.

[153] Mingyu Yang, Yu Chen, and Hun-Seok Kim. Efficient deep visual and inertial odometry with adaptive visual modality selection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 233–250. Springer, 2022.

[154] Tien-Ju Yang, Yi-Lun Liao, and Vivienne Sze. Netadaptv2: Efficient neural architecture search with fast super-network training and architecture optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2402–2411, 2021.

[155] Dong Yu, Morten Kolbæk, Zheng-Hua Tan, and Jesper Jensen. Permutation invariant training of deep models for speaker-independent multi-talker speech separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 241–245. IEEE, 2017.

[156] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models.

In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 702–717. Springer, 2020.

[157] Zhihang Yuan, Bingzhe Wu, Guangyu Sun, Zheng Liang, Shiwan Zhao, and Weichen Bi. S2dnas: Transforming static cnn model for dynamic inference via neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 175–192. Springer, 2020.

[158] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020.

[159] Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang. A systematic dnn weight pruning framework using alternating direction method of multipliers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 184–199, 2018.

[160] Zhengdong Zhang, Amr AbdulZahir Suleiman, Luca Carlone, Vivienne Sze, and Sertac Karaman. Visual-inertial odometry on chip: An algorithm-and-hardware co-design approach. *Robotics: Science and Systems*, 2017.

[161] Xuda Zhou, Zidong Du, Qi Guo, Shaoli Liu, Chengsi Liu, Chao Wang, Xuehai Zhou, Ling Li, Tianshi Chen, and Yunji Chen. Cambricon-s: Addressing irregularity in sparse neural networks through a cooperative software/hardware approach. In *51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 15–28. IEEE, 2018.

[162] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[163] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8697–8710, 2018.

[164] Yuliang Zou, Pan Ji, Quoc-Huy Tran, Jia-Bin Huang, and Manmohan Chandraker. Learning monocular visual odometry via self-supervised long-term modeling. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 710–727. Springer, 2020.