WIP: Distributed inference for human pose estimation using mmWave Wi-Fi

Wouter Lemoine*, Nabeel Nisar Bhat*, Jakob Struye*, Andrey Belogaev*,
Jesus Omar Lacruz†, Joerg Widmer†, Jeroen Famaey*
*University of Antwerp - imec, Antwerp, Belgium. Email: {firstname}.{lastname}@uantwerpen.be
†IMDEA Networks Institute, Madrid, Spain. Email: {firstname}.{lastname}@imdea.org

Abstract—**Joint Communication and Sensing (JCAS) is expected to play a critical role in next-generation wireless networks such as 6G. For complex sensing tasks, such as 3D pose estimation for virtual reality (VR) applications, accurate channel impulse response (CIR) or I/Q samples as well as processing using a neural network is required. Due to the higher bandwidth and antenna array sizes of future wireless networks, it is expected that offloading this data to a remote server for processing would require data rates in the order of 100s of Megabits per second, which is an unreasonable amount of overhead. Therefore it is necessary to preprocess the sensing data locally, and reduce the raw data to useful intermediary features, to mimimize the sensing data transmission overhead, especially when using multiple sensing devices. This paper proposes a method leveraging split inference to distribute neural networks across multiple devices, which achieves high accuracy while addressing the sensing data transfer bottleneck. We evaluate the performance of the proposed method in a VR gaming scenario, where mmWave Wi-Fi signals are used for 3D pose estimation. We show that split inference allows for reducing the communication overhead by three orders of magnitude compared to the centralised approach, while only losing 10% of accuracy. These results pave the way for future work, exploring highly distributed multi-static JCAS as a practical and efficient method of sensing.**

I. Introduction

Joint Communication and Sensing (JCAS) is envisioned to be an important part of next-generation wireless networks such as 6G [1]. It integrates wireless communication and sensing functionalities into a single system allowing efficient use of the Radio Frequency (RF) spectrum. When a Neural Network (NN) is used for processing, it is commonly referred as data-driven or AI-based JCAS. Using JCAS has many advantages, including hardware re-use and time sharing between communication and sensing tasks [1].

JCAS can make use of many different types of signal metadata such as Received Signal Strength Indicator (RSSI) patterns, Channel State Information (CSI), Channel Impulse Response (CIR), or raw I/Q samples. For certain sensing scenarios, it is sufficient to use RSSI patterns, but for more complex tasks such as markerless 3D pose estimation, it is beneficial to make use of CSI, CIR or raw I/Q samples with high granularity. The latter three methods typically generate tremendous amounts of sensing data and would impose a great overhead to transmit this data to a central server, where the sensing result is desired. It is therefore preferred to perform part of the signal processing on the sensing device, to reduce the raw sensing data to a compressed set of useful intermediary features.

Next, Intuitively, using multiple sensing devices can give more accurate results over a single device. However, multiple devices will only exacerbate the sensing data overhead, increasing it linearly with the number of devices, furthering the need to do on-device signal processing. To provide a single pose estimation result on a central server, a method is needed to combine the results of multiple devices.

In this paper, we propose a method leveraging split inference to achieve better performance over single-device methods, while achieving comparable results to a completely centralised approach and keeping the sensing data transfer overhead to a reasonable amount. Using a NN on each device, we can preprocess the data locally, then use a fusing algorithm on a central server to combine results from multiple devices. Finding the optimal split between preprocessing on a device and the postprocessing on a centralised server is a non-trivial optimisation problem and is considered future work beyond the scope of this Work-In-Progress (WIP) paper.

Previously, similar but less complex tasks, such as human position tracking and activity recognition, have been studied using similar methods [2–4], showing that such tasks are feasible using CSI and CIR sensing data. Others have also studied the problem of skeleton-based pose estimation using multiple devices [5, 6] but did not consider distributing the NN over the devices. This sets our work apart as we do consider distributing the NN to decrease the sensing data transfer bottleneck. Other works have also shown that distributing a NN has beneficial effect on the data transfer bottleneck [7], but did not apply it to human pose estimation.

II. Methodology & Architectures

This section describes the steps performed to gather the training and testing data, considered NN-based architectures and how the NNs were trained and tested.

A. Data collection

The input data used for training comes from a data collection campaign where three people were tracked playing an interactive Virtual Reality (VR) game in an office environment. The users were constantly monitored, however, due

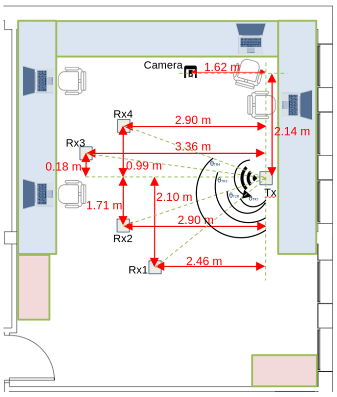


Fig. 1: Top-down view of the data collection setup

to hardware constraints only 105 seconds of sensing data in bursts of 7 seconds is available per person. The data collection setup consists of four devices (denoted Rx in the figure), a transmitter (denoted Tx) and a Kinect camera, placed according to the scheme shown in Figure 1. The receivers and transmitter were controlled by an Xilinx UltraScale Field Programmable Gate Array (FPGA), and the signals were sent and received using mmWave Sivers EVK06002 development kits using 60 GHz up/down converters with a carrier frequency of 60.48 GHz, more information about the testbed can be found in [8]. The Kinect Camera was used to record the ground truth, which consists of 25 body joints measured as (X, Y, Z) coordinates. This generated ≈43 GB of sensing data for testing and training.

A device produces CIR samples at a rate of ≈2775 samples per second in a (12, 128) shape of Float32 values, where 12 is the number of training fields and 128 is the number of range bins. 67 such sub-samples are buffered before being passed to the NN, leading to an input sample shape of (67, 12, 128). The number 67 is chosen such that the CIR sample rate matches the Kinect sample rate of 30 samples per second with interpolation to ≈41.5 samples per second as closely as possible. Next, depending on the number of used devices, denoted n_{sd}, we arrive at a sample shape of (n_{sd}, 67, 12, 128), which will be passed to the NN.

B. NN-based architectures

In this section, we describe the three architectures that have been trained to solve the problem of pose estimation using Wi-Fi CIR data. We use the local NN on a single device and Centralised NN with shared data from multiple devices architectures as baselines for our proposed solution.

The general NN used is shared between architectures. The NN deployed on the devices is an adaption of ResNet18 [9] with n_{sd} input features and outputs features of dimension 75 (25 body joints times 3 coordinates per joint).

We consider the following three architectures:

1) Baseline 1: Local NN on single device: In this architecture, we use the data from only a single device, as shown in Figure 2a. The single device has an embedded NN that directly calculates the human pose estimation. This baseline has the intuitive limiting factor of having only one viewpoint on the room. Using this as a baseline, we later show the benefits of using multiple devices.

2) Baseline 2: Centralised NN with shared data from multiple devices: On the other end, a completely centralised network is used, as shown in Figure 2b. Here, the devices transmit their full CIR data to the central server and all calculations are performed there. Intuitively, this architecture would achieve the best performance, as all of the sensing data is used. However, it requires all raw CIR samples to be transmitted from the distributed devices to the central server, which generates a large communication overhead. The overhead will be estimated in Section III.

3) Proposed solution: Local NN with centralised fusion: This approach is our proposed solution that tries to balance performance and computation/communication overhead. As shown in Figure 2c, a central server that receives the outputs of the local NNs and performs a fusion algorithm to calculate a joint result. This introduces less communication overhead than the centralised NN at the cost of some computational overhead on the devices, which will be proven in Section III and also a small reduction in pose estimation accuracy.

Generally, the fusion algorithm can be any type of function that takes n_{sd} sets of output values from local NNs on multiple devices and outputs a single set of output values. In this early work, a simple averaging function over the output coordinates of each local NN is used. This has empirically shown to have good performance while keeping computation overhead to the bare minimum. We note that more complex fusion algorithms, for example using a NN, would allow the local NN to output different shapes of intermediate data, potentially leading to increased performance. This more complex fusion algorithm may lead to increased performance.

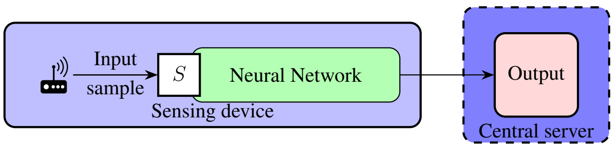
C. Training

As our work focuses on the overhead when distributing NNs, the training happens in an offline and centralised manner. This separates our work from Federated Learning. The loss function used for training the NN was Mean Squared Error (MSE). The data was split into typical 75/2.5/22.5% training/validation/testing sets.

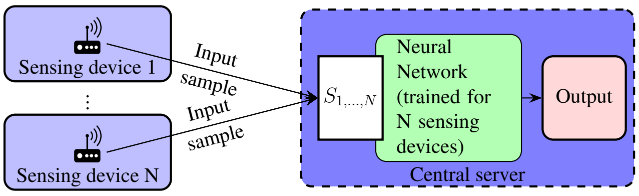
By performing hyperparameter sweeps the NNs were each tuned for optimal performance. For all architectures, 0.0001 for learning rate and 0.000001 for weight decay were empirically the best values. For the local NNs a dropout of 0.2 yielded best performance, whereas for the centralised NN a dropout of 0.35 was deemed optimal.

III. Computation and Communication overhead

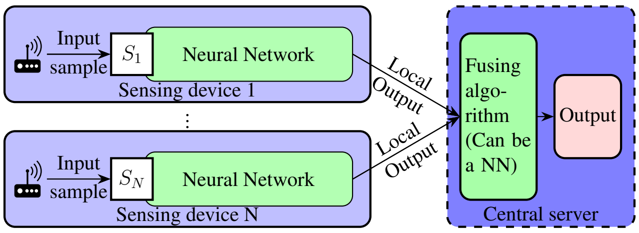
In this section, we describe a mathematical model for the communication and computation overhead for the three approaches. When deciding where to place computation relative to a data source, there's a natural trade-off to consider. Moving more computation away from the data source allows



(a) Local NN on single sensing device



(b) Centralized NN with shared data from multiple sensing devices



(c) Local NN with centralised fusion

Fig. 2: Network architectures

the use of more powerful hardware but requires more data transfer. In contrast, placing computation close to the data source avoids the need for data transfer, though it typically involves using less powerful hardware.

A. Computation Overhead

To calculate the computational overhead, a notion of computation cost of a NN is necessary. Commonly, when using NN, the number of Multiply-Accumulate (MAC) operations is used. A MAC operation is equal to 2 Floating Point Operations (FLOPS), which are a standard metric of computing cost. This work uses ResNet18 as the base model, which comprises of Conv2d, BatchNorm2d and Linear layers, requiring MAC operations. For each of these layer types there is a general formula that can be used to calculate the number of MAC operations.

1) For Conv2d this formula is: $O_{width} \cdot O_{height} \cdot O_{channels} \cdot kernelsize$ with O being the output shape.

2) For Batchnorm2d, the number of MAC operations is equal to the number of input features.

3) For linear layers the number of MAC operations corresponds to the formula: $features_{in} \cdot features_{out}$

Using the corresponding formulas, we calculate the total number of MAC operations for each case. The local NN on single device requires $4.19 \cdot 10^9$ MAC operations. Next, the centralised NN with shared data from multiple, in this case 4, devices requires $4.43 \cdot 10^9$ MAC operations, however these can be disregarded as these operations are run on the central server, for which it can be assumed that it has a significantly larger amount of processing power. The small increase in MAC operations is due to the input layer scaling linearly with the number of devices. Lastly, the local NN with centralised fusion will require n_{sd} times the number of MAC operations for a single device. With $n_{sd} = 4$ this equals $16.76 \cdot 10^9$ total MAC operations. However, this would be split amongst n_{sd} devices. Furthermore, the fusion algorithm has negligible calculation overhead.

B. Communication Overhead

When calculating communication overhead, we again consider the three cases:

1) Local NN on single device: in this architecture there is only a single device, the result gets directly communicated to a centralised server. The data that needs to be transmitted is the final result of 25 (X, Y, Z) coordinates consisting of 32 bit Floating Point (FP) numbers. Such a packet contains 300 B of data. The sample rate equals ≈41.5 samples per second and is equal to the rate of transmission of the result. Multiplying these numbers leads to a data rate of 99.6 kbps.

2) Centralised NN with shared data from multiple devices: this architecture makes use of multiple devices, since the network is centralised all raw data needs to be streamed to a centralised server. The server can then aggregate the data and process it with the NN. The data that needs to be transmitted is of the shape (67, 12, 128) per device as defined in Section II-A. Such a sample also consists of 32 bit FP numbers and is therefore 411 648 B of data per sample. The transmission rate is equal to that of 1) leading to a total data rate of ≈136.7 Mbps per device. With $n_{sd} = 4$, we require a total data rate of ≈546.7 Mbps.

3) Local NN with centralised fusion: this scenario is similar to 1), but since there are multiple devices, the total data rate is equal to n_{sd} times the data rate of a single device (99.6 kbps). With $n_{sd} = 4$, we have a total data rate of only 398.4 kbps. When comparing 1) and 3), we note that the total required data rate increases linearly with the amount of devices n_{sd}. Comparing 2) and 3), we show a 1372-fold decrease in required data rate when the number of devices in both scenarios are equal.

Note that we do not consider the extra overhead that comes with headers, as this depends on the specific protocol used to communicate sensing data.

C. Evaluation

To evaluate our architectures, we used the testing data created from the initial dataset. This simulates the environment in which the NN-based architectures would be deployed, if they were deployed on physical systems. For the single device baseline, we test the NN on all 4 positions separately and pick the best performing position.

IV. Results and Discussion

This section describes the results we gained after training and testing the three architectures. We used $n_{sd} = 4$ devices

TABLE I: Comparison of loss and communication and computation overhead between architectures

Model	Loss (MSE)	Communication Overhead (Mbps) (total)	Computation Overhead (MACs)
Single sensing device Baseline	0.0076	0.0996	$4.19 \cdot 10^9$ (on device)
Centralised Baseline	0.0060	546.7	$4.43 \cdot 10^9$ (on server)
Local NN with centralised fusion	0.0066	0.3984	$16.76 \cdot 10^9$ (on device)

to test these architectures. From testing we show that the centralised baseline has the best performance but, due to communication overhead, this approach is infeasible in a real-world deployment. Our solution lowers the loss by 13% over the single device baseline, while keeping the communication and computation overhead reasonable.

Table I summarizes the MSE of testing with the related communication and computation overhead for each approach. The results show that the local NN with centralised fusion achieves better performance than the single device baseline, however some performance is lost compared to the centralised benchmark. This happens due to information loss caused by processing the data locally and only combining the final results, compared to the centralised NN where the NN can use all information from all devices when calculating the result. On the other hand, the centralised benchmark requires an unworkable data rate, whereas our proposed solution requires little bandwidth in comparison.

V. Future Work

This paper is an exploratory initial work. As of now there are two main limitations: 1) the accuracy of our approach is 10% lower than the fully centralized approach 2) the current approach doesn't account for computational resource limitations on the sensing nodes.

To work out these limitations we consider many avenues for future work. First, consider increasing the complexity of the fusion algorithm. At the moment, only a simple averaging function has been used, which has shown to work well but it potentially can be improved. Second, investigate splitting the NN in an input and output sub-network, where the input sub-networks are run on the devices and the output sub-network runs on the centralised server. This network would be built in such a way that it finds the optimal trade-off between maximal performance, minimal communication and computational overhead on the sensing nodes. Third, use complexity reduction techniques for NN to further reduce computational overhead on the devices [10]. Fourth, increase the number of devices to see how well it scales with higher numbers of devices. Fifth, compression methods can be utilised to further reduce communication overhead.

The first avenue has potential to resolve limitation 1) and the third avenue could effectively deal with limitation 2). The second avenue could provide an answer to both 1) and 2). Ideally all avenues can be combined for optimal performance while accounting for the computing and communication resource constraints at the network edge.

VI. Conclusion

We have shown the strength of distributing a NN to remote devices to decrease the necessary data rate by three orders of magnitude while only impacting performance by 10%. This provides a strong foundation for further research in distributing NNs onto devices to decrease network load to solve complex JCAS tasks.

Acknowledgements

Nabeel Bhat is funded by the Fund for Scientific Research Flanders (FWO) under grant agreement number 1SH5X24N. Part of this research was funded by the FWO WaveVR project (Grant number: G034322N)

References

[1] Thorsten Wild, Volker Braun, and Harish Viswanathan. "Joint Design of Communication and Sensing for Beyond 5G and 6G Systems". In: *IEEE Access* 9 (2021), pp. 30845–30857.

[2] Mohamed Hany Mahmoud et al. "OpenPose-Inspired Reduced-Complexity CSI-Based Wi-Fi Indoor Localization". In: *IEEE Commun. Lett.* 28.9 (2024), pp. 2066–2070.

[3] Hongfei Xue et al. "DeepMV: Multi-View Deep Learning for Device-Free Human Activity Recognition". In: *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4.1 (2020), 34:1–34:26.

[4] Yongsen Ma et al. "SignFi: Sign Language Recognition Using WiFi". In: *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2.1 (2018), 23:1–23:21.

[5] Yili Ren et al. "GoPose: 3D Human Pose Estimation Using WiFi". In: *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6.2 (2022), 69:1–69:25.

[6] Yunjiao Zhou et al. "AdaPose: Towards Cross-Site Device-Free Human Pose Estimation With Commodity WiFi". In: *IEEE Internet of Things Journal* (2024), pp. 1–1.

[7] Qing Xue et al. "A Survey of Beam Management for mmWave and THz Communications Towards 6G". In: *IEEE Commun. Surv. Tutorials* 26.3 (2024), pp. 1520–1559.

[8] Jacopo Pegoraro et al. *DISC: a dataset for integrated sensing and communication in mmWave systems.* Nov. 2022.

[9] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition.* 2016, pp. 770–778.

[10] Youssef Abadade et al. "A Comprehensive Survey on TinyML". In: *IEEE Access* 11 (2023), pp. 96892–96922.