IMULet: A Cloudlet for Inertial Tracking

Mohammed Alloulah
Nokia Bell Labs, USA
mohammed.alloulah@nokia-bell-labs.com

Lauri Tuominen
Aalto University, Finland
lauri.a.tuominen@aalto.fi

ABSTRACT

Inertial measurement units (IMUs) afford the problem of localisation unique advantages owing to their independence of costly deployment and calibration efforts. However, IMU models have traditionally suffered from excessive drifts that have limited their appeal and utility. Newer machine learning (ML) approaches can better model and compensate for such inherent drift at the expense of (i) increased computational penalty and (ii) fragility w.r.t. changes in the signal profile that these ML models have been trained on.

In this paper we propose an edge cloud-based inertial tracking architecture that overcomes the above limitations. Our IMU tracking cloudlet is comprised of: (i) an on-device component that compresses inertial signals for wireless transmission, (ii) a cloud-side ML model that tracks the temporal dynamics of inertial signals, and (iii) a cloud-side deep latent space tracking in order to seamlessly manage model adaptation—i.e. to mitigate the fragility of ML over-specialisation. Early evaluation demonstrates the feasibility of our approach and exposes items of future research.

CCS CONCEPTS

• **Computing methodologies** → Learning latent representations; Distributed computing methodologies; • **Networks** → Location based services.

KEYWORDS

Mobile Computing, Indoor Tracking, Deep Learning

ACM Reference Format:
Mohammed Alloulah and Lauri Tuominen. 2021. IMULet: A Cloudlet for Inertial Tracking. In *The 22nd International Workshop on Mobile Computing Systems and Applications (HotMobile '21), February 24–26, 2021, Virtual, United Kingdom.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3446382.3448364

1 INTRODUCTION

An arsenal of localiser modalities has greatly advanced the problem of indoor and outdoor tracking, albeit with various modality-specific pros and cons. Of such modalities, inertial tracking is especially intriguing owing to its infrastructure-less appeal; Inertial tracking indoors does not rely on costly deployment and calibration effort. Particularly, in a multi-modal system—e.g. autonomous robot employing vision-inertial or radio-inertial dual perception configuration—an inertial measurement unit (IMU) can help maintain a level of service under momentary occlusion and/or a low SNR regime because of its relative independence of dynamic environmental conditions. Such unique behaviour is crucial for closing the tracking performance gap (or error long tail) that has hitherto hindered the flourishing of indoor location-based services compared to their outdoor counterparts.

In this paper we set out to instantiate an IMU industrial-grade asset tracker that is *scalable* (i.e. configurable) across many different usage scenarios. A configurable inertial tracker will naturally aid in supplying accurate and reliable localisation, part of a suite of localiser modalities. To this end, we adopt a cloudlet architecture [23] and call our system *IMULet.*[1] IMULet achieves two main design objectives: (i) ultra-low power operation for aggressively extending the battery life of asset tracking tags, and (ii) adaptive model customisation in order to overcome signal heterogeneity and to ensure the inertial tracking model is appropriately matched to a given motion profile and/or device characteristics.

Traditional hand-crafted inertial trackers, such as pedestrian dead reckoning (PDR) methods, are hard to tune [5]. Further, they exhibit excessive error growth owing to their *cumulative integration* errors. Recent advances in ML-based inertial navigation has demonstrated improved ability to capture and compensate for such cumulative integration errors through a sequence-based learning formulation [6]. However, inertial signals are characterised by distribution variabilities e.g. as a result of device heterogeneity or owing to changes in their motion profile—together referred to as domain shift. Such domain shift makes ML models hard to generalise without extensive adaptation [28].

ML inertial trackers can be interpreted as an over-specialised approach to inertial modelling whereby additional data-driven[2] priors are ingested in order to further constrain and enhance estimation. Invariably, performance gains from over-tuning come with susceptibility to changes in these data priors. That is, a model specialised for a source domain with certain signal characteristics (e.g. wheeled robot) will perform adversely on a target domain with differing characteristics (e.g. parcel carried by a pedestrian).

Generative adversarial networks (GANs) have been shown to mitigate against such domain shift; however, at the expense of high-complexity modelling revolving around recurrent neural networks (RNNs) as building blocks [7]. Such high-complexity models operating directly on raw IMU signals are hard to employ in ultra-low power devices [9], as we will discuss in §5. Further, knowing when to specialise a localiser model without explicit laborious testing against groundtruth labels remains an open problem [6].

In addressing the aforementioned challenges, IMULet incorporates novel measures to *deploy* and *adapt* inertial ML models on resource-constrained devices. IMULet strives to simultaneously

[1] pronounced amulet
[2] a view we do not necessarily subscribe to. See [20] for a lively debate on this.

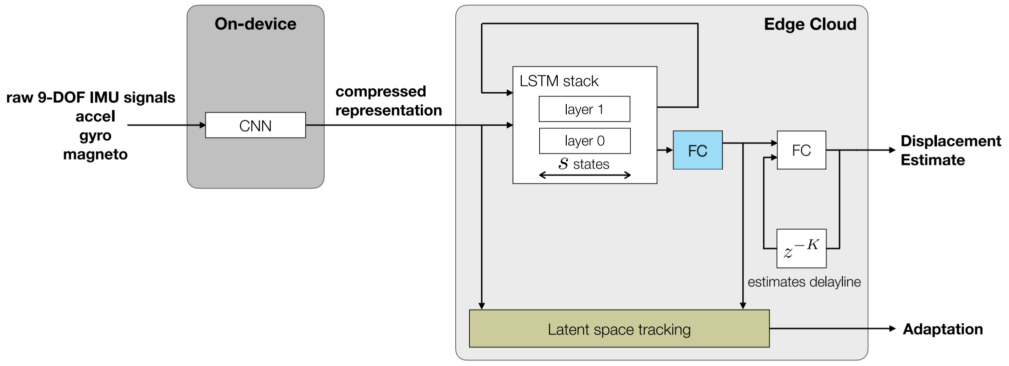


Figure 1: Proposed inertial tracking cloudlet architecture.

maximise the battery life of ultra-low power mobile tags while offering state-of-the-art inertial tracking performance as well as seamless domain adaptability. IMULet achieves this through three key ideas.

(i) Model partitioning. IMULet splits the neural network between the tag and the edge cloud such that the tag executes a relatively small workload. The resultant signal representation after on-device processing significantly reduces the amount of data to be transmitted to the edge cloud compared to raw inertial signals.

(ii) Cloud-side domain alignment. By a careful neural network design which we will detail, IMULet restricts the task of adapting to a new motion profile to take place on the part of the neural network residing in the cloud only. As such, IMULet eliminates or minimises the need for on-device weights update—be it via on-device computations or via uplink+downlink edge cloud communications.

(iii) Latent space-based tracking. IMULet detects an event of anomalous inertial signals that the model has not been trained on by tracking the deep latent space residing in the edge cloud. Upon such anomalous event, the adaptation process of the model is triggered. Tracking the latent space has the advantage of being robust and entirely cloud-based, thereby costing no energy penalty and maximising tag battery life.

The combination of these three ideas allows IMULet to meet the aggressive performance, energy efficiency, and small form factor demands of low-cost industrial asset tracking tags. As such, IMULet enhances the opportunities to scale asset tracking tags up or down in order to address wider use cases and/or industrial market segments.

We proceed next to establish the need to adopt an edge cloud approach for IMU tracking.

2 THE CASE FOR IMU CLOUDLETS

The ML model and latent space tracking are key parts of our proposal as they provide the capabilities for adaptive, infrastructure-free IMU tracking. However, these capabilities alone do not guarantee a feasible real-world solution. In fact, the increased computational complexity of the ML model comes with the risk of introducing compute and/or communication latency and consuming too much power on the envisioned ultra-low power tags. These risks may result in higher error rates, or even system failure.

A cloudlet is a software architecture that mitigates against risks associated with compute-intensive, bandwidth-hungry, and latency-sensitive applications [12, 23]. A cloudlet architecture provides a separation of concerns in order to reduce various delays present in the end-to-end path between system components: from capturing on-device data to the application that consumes this data. For IMULet, a cloudlet architecture will overcome the barriers associated with ultra-low power trackers by allowing them to continue to operate on compute- and energy-constrained hardware irrespective of data acquisition rate and the computational burden of modern ML models. Crucially, to overcome the problem related to latency, a cloudlet ensures that communications between the device and the model occurs as fast as possible in order to push the envelope of IMU tracking performance.

In what follows we describe the impact of latency upon the IMU tracking architecture, citing current industry best practices and based on published latency figures from Microsoft Azure. Compute requirements are then highlighted. Together, latency and compute demonstrate the need and feasibility of an IMU tracking cloudlet architecture.

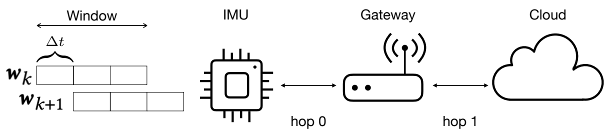
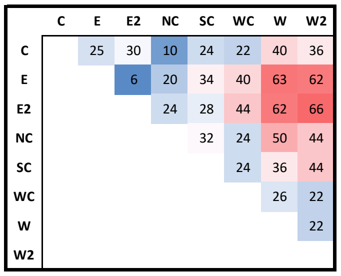


Figure 2: Illustration of IMU signal segmentation with window overlap along with two networking hops, where hop 0 is a wireless transmission between the IMU and a gateway, and hop 1 is a round-trip communication between the gateway and a remote cloud.

Network latency. A cloudlet can significantly reduce the network latency, thereby enhancing IMU tracking rate. To see this, let us consider a low-end IMU operating at a nominal 100Hz sampling frequency i.e. T_s = 10ms sampling period. As shown on the left hand side of Figure 2, a signal overlap of Δt between two consecutive windows $\boldsymbol{w}_k$ and $\boldsymbol{w}_{k+1}$ is utilised in order to enhance displacement estimates. Assuming a moderate $\Delta t = 5T_s$, the hard requirement for the end-to-end (E2E) system delay becomes 50ms. We will define E2E latency as the network round-trip time (RTT) + compute time [12]. In order to drive the discussion, let us also assume an IMU sensor data transmission over WiFi and a back-end cloud hosted



	C	E	E2	NC	SC	WC	W	W2
C		25	30	10	24	22	40	36
E			6	20	34	40	63	62
E2				24	28	44	62	66
NC					32	24	50	44
SC						24	36	44
WC							26	22
W								22
W2								

Figure 3: The monthly average round-trip time (RTT) between Azure regions as measured by ThousandEyes [19]. C, E, E2, NC, SC, WC, W, and W2 are respectively central, east, east 2, north central, south central, west central, west, and west 2.

on Microsoft Azure. The overall network delay now involves two hops: (1) wireless IMU $\longrightarrow$ gateway and (2) RTT gateway $\leftrightarrow$ cloud as depicted in Figure 2. First, the 90th percentile WiFi latency is around 20ms based on empirical evaluation done in [25]. Second, the average Azure RTT between data centres located in various US regions is around 34ms; An excerpt from [19] for the breakdown of US inter-region RTT is reproduced in Figure 3. Careful cloud instantiation may optimise for cloud latency given application geography. As such, we would expect to have enough margin to guarantee meeting the above stated E2E delay requirement of 50ms, given the two-hop networking overhead of a conventional cloud architecture.

Now consider that the 90th percentile WiFi latency is even worse in the lower power 2.4GHz band [25]. Other IoT wireless protocols typically slacken latency further in favour of power optimisation e.g. as in the 900MHz band [1]. Further, higher-end IMU's have even higher sampling rates e.g. 400Hz–2kHz [16]. As such, E2E system delay requirements may become significantly tighter, thus necessitating adopting an edge computing approach over a network-bottlenecked conventional cloud [12, 23]. This is especially the case when the IMU cloudlet is a subsystem that interfaces with a broader multi-modal tracker involving high-rate vision and radar as in [4].

Compute time (ms) – mean/standard deviation (μ/σ)				
Hardware	Intel Core i7		NVIDIA GeForce RTX 2080 Ti	
Batch size	single	batched	single	batched
512	1.62/0.31	51.88/2.49	0.91/0.07	1.07/0.13
256	2.22/0.91	31.29/2.65	0.88/0.08	1.04/0.09
128	2.45/1.45	15.51/2.26	0.89/0.12	1.00/0.10

Table 1: Scalable inferences through GPU batch processing.

Compute scalability. A cloudlet leveraging commodity GPU's can offer vast IMU tracking scalability at minimal compute delay. To see this, we measured the inference time of the overall proposed neural network[3], with and without GPU acceleration. We further repeated the experiments for both single- and batch-processed inferences. Results are shown in Table 1. On a standard laptop Intel processor, single inferences took approx. 1–2ms to compute, while batch processing of multiple measurements scaled poorly owing to the inability of generic CPU architecture to capitalise on parallelism. In contrast, batched inferences were executed on the NVIDIA RTX 2080 Ti GPU over approx. 1ms irrespective of how many measurements were processed concurrently. This suggests that the modestly priced RTX 2080 Ti GPU may be sufficient to service a warehouse with hundreds of high-rate IMU tags comfortably.

[3]whose details we treat in subsequent sections

3 CLOUDLET ARCHITECTURE

We next describe the three design techniques IMULet incorporates in order to realise an adaptive and scalable inertial tracking couldlet.

3.1 Model partitioning

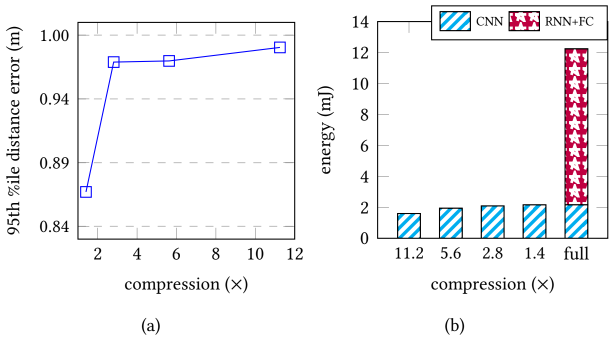


Figure 4: Localisation error as a function of the compression ratio, along with the required on-device compute energy: (a) 95th %ile distance error in metres on logarithmic scale and (b) energy in milli-Joules as measured on an NVIDIA Jetson Nano device using a Keysight N6705C power analyser.

IMULet splits the model between the tag and the cloud such that the tag executes only the fairly light-weight convolutional (CNN) section of the model. The CNN acts as an embedding network that maps the input signals to a low-dimensional space. That is, the asset tracking tag executes a few dimensionality reduction neural layers that compress the amount of data to be transmitted to the cloud compared to raw inertial signals. Such device-cloud model partitioning is illustrated in Figure 1.

In an end-to-end learning fashion, IMULet can choose to arbitrarily compress the raw inertial signals fed to the cloud portion of model consisting of the RNN and the fully connected (FC) blocks. The degree of compression is defined by the strides of the convolutions and the number of channels in the output signal. Figure 4 depicts the achieved localisation error as a function of the compression ratio—defined as the ratio between the sizes of the input and output matrices—along with its correspondingly required on-device compute energy. For instance, as per Figure 4(a), around 12cm 95th %ile localisation error can be traded off for a substantial 11 times reduction in transmitted signal. The compute energy required by on-device compression is around 2mJ irrespective of the compression configuration as shown in Figure 4(b). Out-of-the box and without optimisation, the energy expenditure of the compressive CNN amounts to less than 17% of overall network energy; RNN and FC dominate the full model execution energy. Typically, CNNs can be numerically optimised further in order to yield large energy savings e.g. through quantisation in [13].

There are, however, a number of ways in which design decisions for on-device compute may interact with the cloud portion of the model. For example, the quantisation dynamic range when transmitting over the wireless channel is one important design aspect to consider [17]. In turn, the cloud-side of the model could be rendered less sensitive to such quantisation effects using a variety of ML techniques, the simplest of which is perhaps data augmentation with on-device representation in order to promote model resilience to quantisation. Other aspects include the compression of sparsified signals which may introduce further reconstruction artefacts to the cloud-side. This emerging design landscape belongs to a so-called wireless-ML co-design [17].

3.2 Cloud-side domain alignment

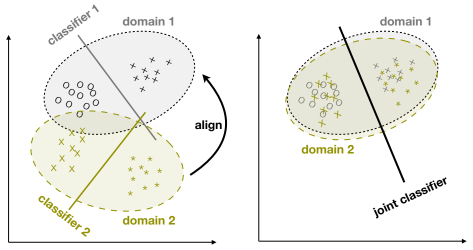


Figure 5: Domain alignment is a training-time intervention through which two or more signal domains can be made to share a joint classifier without compromising their individual performances shown here as binary class separability.

While the CNN block of the neural network provides initial signal compression (i.e. on the tag), the rest of the network architecture including domain adaptation takes place in the cloud. This reduces the need for weight update of the CNN block, and restricts adaptation to the cloud side. That is, neither on-device compute nor uplink+downlink communications to/from the cloud is needed for domain adaptation. To this end, IMULet leverages a domain adaptation technique based on optimal transport (OT) [3]. Critically, IMULet derives a domain alignment loss based on OT for the latent space residing in the cloud portion of the neural network model only. As such, the alignment loss does not affect the compressed representation executed on-device, thereby minimising the need for frequent updates, which translates into substantial energy savings.

Figure 5 illustrates the general concept of domain alignment as discussed in [10]. This flavour of domain adaptation attempts to find a latent space representation that is invariant to domain shift. This is achieved by training the model with labelled data from one or more source domains (possibly mixed with unlabelled data) such that the model is able to generalise to data drawn from any related but potentially shifted target domain. As such, the joint classifier boundaries would result in high accuracy irrespective of changes in input distribution. Note, however, that in contrast to single-source domain adaptation, IMULet deals with a more general, but related problem known as domain generalisation. In domain generalisation, a network is trained *jointly* on multiple source domains—instead of one—in order to arrive at a model that generalises well to unseen target domains [28]. Furthermore, compared to a classifier with a small discrete number of classes, such alignment is intuitively harder to achieve when considering the continuous nature of values position coordinates can assume. That is, in terms of problem-specific complexities, we are concerned with a localisation MSE loss which is harder to re-align compared to say a cross entropy loss for a small finite set of classes. This is further compounded by the feedback[4] as depicted in our model in Figure 1.

Concretely and inspired by [26], IMULet splits the output FC layer into two as shown in Figure 1. On the one hand, [26] suggests that aligning the latent spaces of deeper layers is more effective for domain invariance since deep layers will more fully capture the semantics of a given physical phenomenon. On the other hand, at the final output (i.e. displacement estimate), the latent space representation is too truncated for effective alignment as it no longer carries "soft" inter-domain information. That is, IMULet seeks a deep layer with sufficient high-dimensionality in order to manipulate at training-time for domain invariance. This is achieved by optimising over a loss that incorporates a domain mismatch term as well as an MSE localisation term. For example, the LSTM may output a 1600 dimensional vector that forms an input to a first FC layer[5] whose output is 64 dimensional vector, followed by a second FC layer that transforms into Cartesian coordinates. Here, the 64 dimensional latent space may serve as a good balanced representation for optimising for domain invariance. Mathematically, we can write a combined loss expression as

$$\mathcal{L} = \mathcal{L}_{\text{regression}} + \mathcal{L}_{\text{alignment}} \tag{1}$$

where $\mathcal{L}_{\text{regression}}$ is computed at the output of the 2nd FC whereas $\mathcal{L}_{\text{alignment}}$ is computed at the output of the 1st FC. In practice, the multipliers $\lambda_{\text{regression}}$ and $\lambda_{\text{alignment}}$ are further used as hyperparameters. Note that the alignment loss can be computed either under a supervised configuration, under unsupervised one, or a combination thereof subject to performance and system architecture constrains. Mixing labels and surrogate labels in the regression loss is a standard technique for combating catastrophic forgetting [3]. The loss may also incorporate other terms not shown above e.g. to promote reconstruction, adversarial loss [15], etc. Additionally, the internal learnt representation of the neural network can be Cartesian, polar, or quaternion.

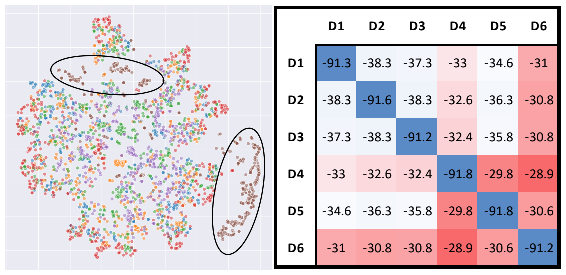
Multi-domain scalability. The OT formulation IMULet employs is domain scalable [3]. Meaning, IMULet alternates between training a joint localiser and aligning for domain invariance. IMULet schedules many domains to partake in the joint training procedure that preserves the multi-domain discriminative information while optimising for multi-domain alignment.

3.3 Latent space-based anomaly detection

This is where enforcing the geometric OT structure on the latent space during training pays dividends. When the model encounters anomalous inertial signals not seen during training, model adaptation is initiated by tracking the deep latent space residing in the cloud. Latent space tracking is robust and entirely cloud-based; thus it has no implications on tag battery life.

Figure 6 shows an example of latent space tracking. A previously adapted model is fed new unseen inertial signals from a wheeled motion profile[6], denoted by D6 in Figure 6(b). Because the model has not seen this motion profile before, its t-SNE latent space embedding is not evenly spaced, as highlighted in Figure 6(a).

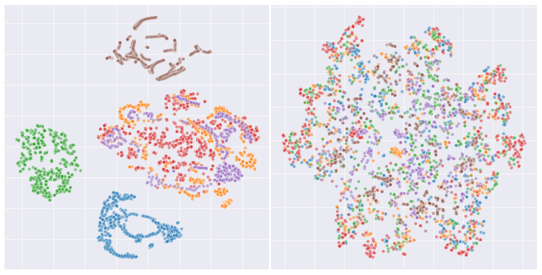
[4]both LSTM states and previous estimated displacement coordinates
[5]highlighted in light blue in Figure 1
[6]i.e. trolley in [8]



	D1	D2	D3	D4	D5	D6
D1	-91.3	-38.3	-37.3	-33	-34.6	-31
D2	-38.3	-91.6	-38.3	-32.6	-36.3	-30.8
D3	-37.3	-38.3	-91.2	-32.4	-35.8	-30.8
D4	-33	-32.6	-32.4	-91.8	-29.8	-28.9
D5	-34.6	-36.3	-35.8	-29.8	-91.8	-30.6
D6	-31	-30.8	-30.8	-28.9	-30.6	-91.2

(a) (b)

Figure 6: (a) t-SNE visualisation of adapted deep latent space showing telltale signatures of unseen motion profile by virtue of uneven latent space utilisation. (b) The deep latent space corresponding to the unseen domain exhibits large pair-wise distances to other seen domains which can be accurately quantified in high dimensions using the Wasserstein distance. Colour scale is in dB.



(a) (b)

Figure 7: Deriving alignment loss for the deep latent space residing on the cloud-side. (a) t-SNE visualization for the on-device compressed representation. (b) t-SNE visualization for the cloud-side deep latent space for which domain adaptation has been performed.

IMULet tracks such effect on the latent space in high dimensions using the *Wasserstein* distance[7]:

$$\mathcal{W}(\mu, \nu) = \underset{\substack{\gamma \in \Gamma(\mu,\nu) \\ \text{s.t. } \gamma\mathbf{1}=\mu,\ \gamma^T\mathbf{1}=\nu}}{\operatorname{argmin}} \quad \langle \boldsymbol{\gamma}, \mathbf{C} \rangle_F \tag{2}$$

where the probabilistic coupling $\boldsymbol{\gamma}$ is between two empirical distributions μ and ν, Γ is the space of the joint probability distributions with marginals μ and ν, $\mathbf{C} \geq 0$ is a cost matrix $\in \mathbb{R}^{n_\mu \times n_\nu}$ representing the pairwise distances, and $\langle \cdot, \cdot \rangle_F$ is the Frobenius dot product.

Figure 6(b) depicts a domain confusion matrix in decibels computed as Wasserstein distance metrics in high dimensions. It is clear that domain D6 is the unadapted for domain since it exhibits the largest pair-wise distances to all other domains as indicated by the red colour intensity.

That is, information gleaned from the latent spaces can be used for detecting anomalous asset use and triggering localisation model adaptation..

4 PRELIMINARY RESULTS

We now discuss preliminary results obtained using the Oxford dataset for deep inertial odometry [8].

4.1 Qualitative

Figure 7 depicts t-SNE visualisations of six different motion profiles i.e. domains for the cloudlet architecture of §3. Specifically, Figure 7(a) illustrates the t-SNE visualisation for the on-device compressed representation, while Figure 7(b) is for the cloud-side deep latent space for which domain alignment is performed. The network is adapted to cope better with tag signals from many different domains. Such adaptation is evident in the fact that the latent space of Figure 7(b) is more evenly utilised and is shared across many inertial signal motion profiles. As such, its tracking performance is likely to be less sensitive to unseen profiles. Without adaptation, the t-SNE of the latent space exhibits some per-profile clustering effects as shown in Figure 7(a).

[7]in discrete form

4.2 Quantitative

The overall tracking performance of IMULet in terms of distance and heading errors is shown in Figure 8. Trajectory tracking was characterised on sequences of inertial data 20 seconds in length, and contained data from six different motion profiles at once [8]. The 95th %ile distance error is under 1m with an average error growth of 0.5m for all 20-second sessions. The heading performance is considerably better than distance with the 95th %ile error under 15° and average error growth bounded below 10°.

The generalisability of the cloud-side deep latent space is illustrated in Figure 9 by means of the localisation performance for an unseen domain. The unseen domain is the challenging trolley scenario from [8]. Specifically, it is evident that OT-based multi-domain adaptation applied to the deep cloud-side latent space results in measurable improvements (especially for the heading performance) compared to an architecturally identical baseline with unadapted model.

5 DISCUSSION

Adaptation trigger. The Wasserstein distance becomes less meaningful in higher dimensions [22]. It is therefore important to balance the need for effective domain alignment with the need for reliable adaptation trigger mechanism. That is, there exists a tension between higher dimentionality for effective domain alignment and lower dimentionality for a robust Wasserstein metric. We have demonstrated that a sweet spot does exist; however, further investigations are needed to fully characterise this interplay.

Multi-domain adaptation. Domain translation is the concept of mapping features from one domain to another. It is widely applied in computer vision (CV) for image-to-image translation tasks. Domain translation can also be used as a domain adaptation technique. This idea has recently been applied to inertial tracking to tackle the problem of mapping between signals from different motion profiles using GANs, with and without paired data [7]. There are two difficulties in adopting such an approach for IMULet. First, as domain translation models rely on having a specific target domain, they may not perform as well on completely unseen domains. Second, LSTM-based GANs operating on raw IMU signals [7] would require either:

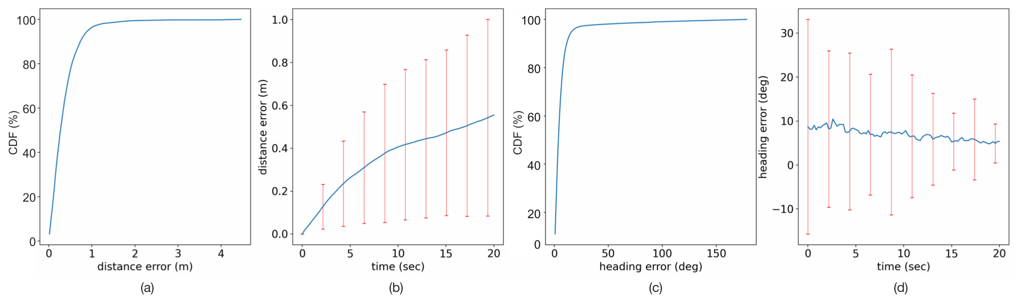


Figure 8: Tracking performance of IMULet. (a) distance error. (b) distance error growth. (c) heading error. (d) heading error growth.

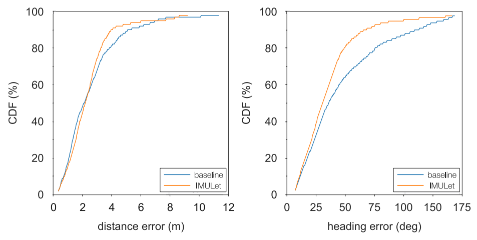


Figure 9: Generalising to new unseen inertial motion profiles is evident through measurable improvements of localisation performance over a naive baseline, especially for heading.

(a) intensive on-device memory and compute resources, or (b) transmitting uncompressed signals to the cloud. Both (a) & (b) we argue are unsuitable for real-world deployments of ultra-low power asset tracking tags. In contrast to domain translation or single-domain adaptation, IMULet extracts domain-invariant features from inertial signals of multiple domains without compromising the discriminative content of the learnt representation. This problem is related to domain generalisation, in which the goal is to decrease a model's sensitivity to domain shifts unknown a priori [2, 21, 28]. Zhao et al. pursue a similar domain generalisability goal using adversarial training [29]; however, IMULet judiciously enforces a geometric OT criterion in order to implement the latent space tracking discussed in §3.3.

Privacy & Security. Tracking the latent space may also afford opportunities to derive subtle but powerful usage analytics of the asset to which a tag is attached e.g. a change in the acceleration patterns of the motor driving a wheeled robot say as a result of defective mechanics. If viable, the notion of latent space tracking may also have privacy and security implications beyond model adaptation which should be investigated.

6 RELATED WORK

ML IMU. Taking an ML approach to IMU indoor tracking, Oxford University researchers propose an RNN-based, sequence-by-sequence location estimator operating on raw inertial signals [6, 7]. However, despite out-performing traditional dead reckoning methods, their architecture is acausal (i.e. incurs latency) and is computationally too expensive for ultra-low power tags. Trading off performance, a lighter-weight alternative was investigated in [9] using a WaveNet as an LSTM substitute. Other works opt to aid conventional state-space Kalman formulations with ML-based parametrisation e.g. [5].

Split & Federated Learning. The idea of splitting the execution of a model between a client and a server [14, 18, 27] is a subbranch of federated learning (FL) [17]. When having a number of pretrained server-side networks, coarse- and fine-grained matching is performed in order to select the best suited pretrained model given client-side hidden represenation in order to preserve privacy [24]. Specifically, in contrast to our cloud-side alignment and latent space tracking, [24] trains autoencoders independently on both the server and client sides and shares the intermediate hidden representation of the data for model selection. Joint wireless-FL in which appropriate communication and compute resources are co-designed in order to meet a certain performance objective in a production system remains an open problem [17].

Optimal Transport. OT has been used historically to find a "least-effort" mapping between two resources. However, its use in domain adaptation is comparatively recent [10]. For image classification, an OT-based loss function was used to align the latent spaces of two domains, thereby achieving competitive domain adaptation on multiple CV benchmarks [3]. We extend the use of OT for domain generalisation and characterise its performance for inertial navigation. OT has theoretical connections to other flavours of distribution divergence metrics such as maximum mean discrepancy (MMD) [11]. The latter we have found to be less effective for inertial navigation.

7 CONCLUSION

In this paper we propose IMULet: a cloudlet architecture for inertial tracking suitable for ultra-low power mobile tags. IMULet has an on-device component and an edge cloud-side component. On-device compression and cloud-side model adaptation mechanism are discussed and shown to afford promising early performance. We plan to scale our evaluation and further illuminate system nuances and their interaction in order to provide a summary of expectations on what can be achieved using IMULet for adaptive, yet practical ML-based inertial navigation.

ACKNOWLEDGMENTS

We would like to thank Dr. Eric Rozner and other anonymous reviewers for their feedback, Anton Isopoussu for discussions on optimal transport, René Mayrhofer for discussions on privacy & security, Zoran Radivojevic and Milad Alizadeh for helping with the energy measurements, and Howard Huang, Swetha Muniraju, and Rachel Burrows for input on manuscript.

REFERENCES

[1] 2017. IEEE Standard for Information technology–Telecommunications and information exchange between systems - Local and metropolitan area networks–Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Sub 1 GHz License Exempt Operation. *IEEE Std 802.11ah-2016 (Amendment to IEEE Std 802.11-2016, as amended by IEEE Std 802.11ai-2016)* (2017), 1–594.

[2] Kei Akuzawa, Yusuke Iwasawa, and Yutaka Matsuo. 2019. Adversarial invariant feature learning with accuracy constraint for domain generalization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 315–331.

[3] Bharath Bhushan Damodaran, Benjamin Kellenberger, Rémi Flamary, Devis Tuia, and Nicolas Courty. 2018. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 447–463.

[4] Mario Bijelic, Tobias Gruber, Fahim Mannan, Florian Kraus, Werner Ritter, Klaus Dietmayer, and Felix Heide. 2020. Seeing Through Fog Without Seeing Fog: Deep Multimodal Sensor Fusion in Unseen Adverse Weather. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11682–11692.

[5] Martin Brossard, Axel Barrau, and Silvère Bonnabel. 2020. AI-IMU dead-reckoning. *IEEE Transactions on Intelligent Vehicles* (2020).

[6] Changhao Chen, Xiaoxuan Lu, Johan Wahlstrom, Andrew Markham, and Niki Trigoni. 2019. Deep neural network based inertial odometry using low-cost inertial measurement units. *IEEE Trans. Mobile Comput.* (2019).

[7] Changhao Chen, Yishu Miao, Chris Xiaoxuan Lu, Linhai Xie, Phil Blunsom, Andrew Markham, and Niki Trigoni. 2019. Motiontransformer: Transferring neural inertial tracking between domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 8009–8016.

[8] Changhao Chen, Peijun Zhao, Chris Xiaoxuan Lu, Wei Wang, Andrew Markham, and Niki Trigoni. 2018. OxIOD: The Dataset for Deep Inertial Odometry. arXiv:1809.07491 [cs.RO]

[9] Changhao Chen, Peijun Zhao, Chris Xiaoxuan Lu, Wei Wang, Andrew Markham, and Niki Trigoni. 2020. Deep-Learning-Based Pedestrian Inertial Navigation: Methods, Data Set, and On-Device Inference. *IEEE Internet of Things Journal* 7, 5 (2020), 4431–4441.

[10] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy. 2017. Optimal Transport for Domain Adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 9 (2017), 1853–1865.

[11] Aude Genevay, Gabriel Peyré, and Marco Cuturi. 2018. Learning generative models with sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*. 1608–1617.

[12] Shilpa George, Thomas Eiszler, Roger Iyengar, Haithem Turki, Ziqiang Feng, Junjue Wang, Padmanabhan Pillai, and Mahadev Satyanarayanan. 2020. OpenRTiST: End-to-End Benchmarking for Edge Computing. *IEEE Pervasive Computing* 19, 4 (2020), 10–18.

[13] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. 2014. Compressing Deep Convolutional Networks using Vector Quantization. arXiv:1412.6115 [cs.CV]

[14] Otkrist Gupta and Ramesh Raskar. 2018. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications* 116 (2018), 1–8.

[15] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. 2018. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*. PMLR, 1989–1998.

[16] https://www.xsens.com. [n.d.]. Xsens inertial sensor modules. https://www.xsens.com/inertial-sensor-modules. Accessed: 04-01-21.

[17] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2019. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977* (2019).

[18] Stefanos Laskaridis, Stylianos I Venieris, Mario Almeida, Ilias Leontiadis, and Nicholas D Lane. 2020. SPINN: synergistic progressive inference of neural networks over device and cloud. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. 1–15.

[19] Microsoft. [n.d.]. Azure network round-trip latency statistics. https://docs.microsoft.com/en-us/azure/networking/azure-network-latency. Accessed: 04-01-21.

[20] MONTREAL.AI. [n.d.]. AI DEBATE 2: Moving AI Forward: An Interdisciplinary Approach. https://montrealartificialintelligence.com/aidebate2/. Accessed: 04-01-21.

[21] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. 2013. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*. 10–18.

[22] Gabriel Peyré and Marco Cuturi. 2020. Computational Optimal Transport. arXiv:1803.00567 [stat.ML]

[23] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. 2009. The case for VM-based cloudlets in mobile computing. *IEEE pervasive Computing* 8, 4 (2009), 14–23.

[24] Vivek Sharma, Praneeth Vepakomma, Tristan Swedish, Ken Chang, Jayashree Kalpathy-Cramer, and Ramesh Raskar. 2019. ExpertMatcher: Automating ML Model Selection for Clients using Hidden Representations. arXiv:1910.03731 [cs.CV]

[25] Kaixin Sui, Mengyu Zhou, Dapeng Liu, Minghua Ma, Dan Pei, Youjian Zhao, Zimu Li, and Thomas Moscibroda. 2016. Characterizing and improving wifi latency in large-scale operational networks. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*. 347–360.

[26] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474* (2014).

[27] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. 2018. Split learning for health: Distributed deep learning without sharing raw patient data. *preprint arXiv:1812.00564* (2018).

[28] Garrett Wilson and Diane J Cook. 2020. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)* 11, 5 (2020), 1–46.

[29] Mingmin Zhao, Shichao Yue, Dina Katabi, Tommi S Jaakkola, and Matt T Bianchi. 2017. Learning sleep stages from radio signals: A conditional adversarial architecture. In *International Conference on Machine Learning*. 4100–4109.