Deep Learning for the Internet of Battlefield Things

Tarek Abdelzaher, Shuochao Yao
Department of Computer Science
University of Illinois at Urbana, Champaign
Urbana, IL 61801
USA

zaher@illinois.edu, syao9@illinois.edu

ABSTRACT

The paper describes the technical challenges and recent solutions that bring the advantages of deep learning to the world of embedded connected devices in the battlefield, known as the Internet of Battlefield Things. Deep learning has shown significant improvements in fields such as computed vision, speech recognition, and text processing, creating unprecedented opportunities for new interaction modalities with smart devices. However, deep learning systems are resource intensive. In denied or contested battlefield settings, devices may not be able to connect to a back-end cloud for machine intelligence services (the way, Google Home devices do, for example, to interpret voice commands or recognize users). Instead, field devices must rely on their own resources to process deep learning functions. This leads to several research challenges. First, what deep neural network structures can effectively process and fuse sensory input data for diverse sensor fusion applications, beyond vision and speech? Second, how to reduce the resource consumption of trained deep learning models so they can run on resource-limited devices in the field? Third, how can confidence be estimated in results produced by such deep learning devices? Finally, how to minimize the need for labelled data? The paper describes experiences with a system called DeepSense that offers an architecture for deep learning geared for fusing time-series multi-sensor data. It describes means for reducing the resource consumption of the framework by more than an order of magnitude without loss of accuracy. It then presents an approach for estimating confidence in results, offering a tradeoff between resource consumption, fidelity, and training needs. Finally, it describes how to use unlabelled data for training to reduce the need for labor-intensive data labelling. The work directly contributes to the design of AI-empowered Multisensor Fusion Engines that can fit within resource limitations in the field to empower a new generation of situation awareness applications.

1.0 INTRODUCTION

The Internet of Battlefield Things (IoBT) will integrate sensors, actuators, and computing elements to deliver a myriad of defence services that are highly configurable, adaptive, assured, and intelligent. Advances in sensing, computing, and communication technologies create opportunities for automating several defence functions such as data collection and analysis (to meet decision needs), damage assessment, anomaly detection, and logistics management. Machine learning holds the promise that future computing devices, used in military operations, will offer human-like capabilities, such as scene understanding, voice recognition, complex activity detection, and problem diagnostics. Recent advances in deep learning have revolutionized the field through the development of algorithms that train neural networks to perform human-like tasks with a significant degree of accuracy.

Several challenges, however, must be solved before deep learning can empower military multi-sensor fusion systems [5]. For example, what deep neural network structures can effectively process and fuse sensory input

data for diverse fusion applications? How to reduce the resource consumption of deep learning models such that they can be efficiently deployed on resource-constrained devices? How to compute confidence in the correctness of deep learning outputs? Finally, how to minimize the need for labelled data in learning? This paper covers four recent advances in deep learning, developed by the authors, addressing the above problems:

- First, we describe a general architecture for deep neural networks that is geared specifically for multi-sensor data fusion. This architecture is designed to recover spatio-temporal patterns in multi-sensor data by a clever exploitation of different types of neural networks, including (i) fully connected networks (that recover global spatial patterns), (ii) convolutional networks (that recover local spatial patterns), and (iii) recurrent networks (that recover temporal trends).

- Second, we discuss recent advances in neural network compression that allow significant reduction in the memory footprint, execution latency, and total energy consumed by trained neural networks, all without an appreciable reduction in result quality. These advances allow fitting deep neural networks into a very small form factor, so they can live inside devices that can be embedded within a variety of battlefield "things" such as weapons and munitions, thereby producing intelligent behaviour.

- Third, we describe solutions that allow us to compute confidence in results of deep learning systems. Hence, not only will such systems be able to derive insights from input data (e.g., recognize individuals from their sensory signatures), but also compute appropriate confidence estimates in their observations.

- Finally, a big drawback of deep learning systems is the need for labelled data for training. Many military systems will have sensors that collect data. Labelling such data, however, is a very resource intensive process. To alleviate this problem, recent work has done significant advances on training neural networks using primarily *unlabelled* data.

We elaborate on these core problems and their emerging solutions to help lay a foundation for building IoBT systems enriched with effective, efficient, and reliable deep learning models.

2 DEEP LEARNING ARCHITECTURE FOR MULTISENSOR FUSION

One key question in designing a deep learning architecture for multi-sensor fusion is to decide on the type of neural networks to use as well as their interconnection. Briefly, a neural network is composed of several layers of computing nodes. Each node in a given layer is connected to nodes in the previous layer via links of various weights. Individual nodes emulate individual neurons, inspired by the anatomy of the human brain. Each node implements a nonlinear function of inputs to produce its output. Hence, by controlling the weights on the links between nodes one can implement arbitrary nonlinear mappings and transfomations. The input to the first stage of the network is the raw data collected (e.g., individual pixels of an image). The output of the last stage is the result being computed (e.g., the identity of the individual in the picture). The key advance in deep learning literature has been the development of training algorithms that adjust link weights automatically by exposing the neural network to training inputs, coupled with corresponding ground truth output labels. During training, link weights are adjusted in a manner that reduces divergence of network outputs from the ground truth labels over time, essentially learning the complex nonlinear function between inputs and outputs.

Algorithms for several types of deep neural networks were designed. Fully connected neural networks are those where each node in a layer is connected to all nodes in the previous layer. They are good at extracting global spatial patterns. Convolutional neural networks are those where each node in one layer is connected to only a subset of nodes in the previous layer. They can extract complex local features, such as those necessary to

identify handwritten letters on a larger surface, or a small object within a larger image. Finally, recurrent neural networks accept, as input, data from multiple points in time and extract temporal trends.

Arguably, as an alternative to using deep neural networks, one can use laws of physics to derive the needed outputs from sensor data. For example, in a location estimation task, one can double-integrate inputs that comprise accelerometer data to obtain velocity and position. The problem with such approaches is that they are prone to noise. Most estimators make assumptions on the statistical distribution of noise offering accurate results only when such assumptions are satisfied. In the complex battlefield environment, noise is hard to model. It may be non-linear, non-additive, correlated, and biased. Since neural networks are composed of nonlinear elements, they can learn very complex nonlinear relations, allowing the extraction of signals from noise even when the two are intertwined in a complex nonlinear fashion.

Moreover, unlike other machine learning approaches that rely on the design of clever input features (to support the intended estimation or classification tasks), deep learning has the advantage of being able to ingest raw data directly and automatically compose relevant features by adjusting link weights. Hence, less human effort is consumed in feature engineering. In a world dominated by data and computing devices, saving human cognitive bandwidth by employing a machine could be a good trade-off.

A novel framework, called DeepSense [1], was recently proposed as a general-purpose learning framework for sensor fusion systems. It integrates convolutional neural networks (CNN) and recurrent neural networks (RNN) to extract spatio-temporal features of input signals. Sensory data are aligned and divided into time intervals for processing. For each interval, DeepSense first applies an individual CNN to each sensor data stream, encoding relevant local features. A (global) CNN is then applied to the respective outputs to model interactions among multiple sensors for effective sensor fusion. Next, an RNN is applied to extract temporal trends. Multi-sensor fusion systems are generally categorized into two common subgroups: systems that do estimation and systems that do classification (depending on whether the sought results are continuous or categorical, respectively). Hence, at the last stage, either an affine transformation or a softmax output is used by DeepSense, depending on whether the output is an estimation or a classification result.

This architecture solves the general problem of learning how to perform complex multi-sensor fusion tasks for purposes of estimation or classification from time-series data. For estimation-oriented problems, DeepSense learns the physical system, thereby yielding accurate outputs despite noise in sensor data. For classification-oriented problems, the neural network acts as an automatic feature extractor that encodes local, global, and temporal information. DeepSense can be easily customized for a specific IoBT application. The detailed mathematical formulation of the DeepSense learning algorithm can be found in a related paper [1].

An evaluation of DeepSense has shown that it does very well on tasks involving target recognition (e.g., identification of a specific individual from sensor measurements) as well tasks involving activity detection (e.g., separation of walking versus running, regardless of differences across individuals in how the activity is performed). DeepSense also outperformed estimators derived from laws-of-physics, when applied to tasks amenable to a clean theoretical formulation, such as dead reckoning (position estimation) from accelerometer and gyroscope data. Fig. 1 illustrates an example of the quality of position estimation, comparing actual position of a vehicle in motion to that computed from noisy accelerometer and gyroscope data using a deep neural network.

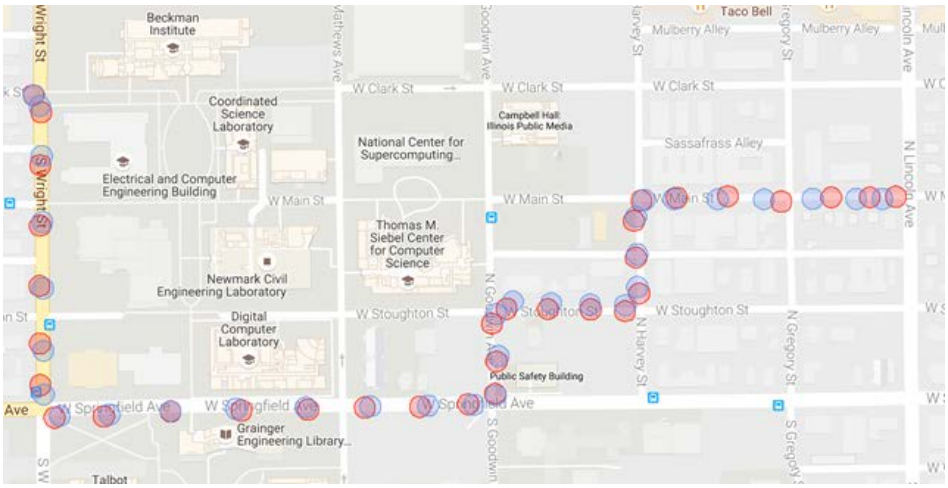

Figure 1: Quality of vehicle position estimation using a deep neural network that processes input from a noisy accelerometer and a gyroscope. The figure compares the actual position of the vehicle (red circles) to the estimated one (blue circles).

The results offer evidence that a general deep learning architecture can beat hand-crafted solutions designed for the individual application spaces. While current work is by no means a consummate proof of generalizability, a main appeal of applying deep learning in IoBT contexts lies in obviating the need for human-intensive per-application customization. More research is needed to substantiate or refute the early evidence and to understand the limits of generalizability of learning models across various systems and applications.

2. COMPRESSING DEEP NEURAL NETWORKS

Once trained, deep neural networks can be used to perform complex estimation, prediction, detection, or identification/classification tasks. Unfortunately, typical networks produced by deep learning techniques are very large. They may include several hundred layers, each composed of possibly thousands of nodes. Clearly, execution of such complex structures will consume a significant amount of resources. Often the observed phenomena evolve over lower-dimensional manifolds. This observation suggests that neural networks produced by deep learning methods can be reduced in size, perhaps without significant loss of accuracy.

Several attempts were made to simplify deep neural networks after they have been trained. One approach is to remove edges that have low weights. The removal produces a sparse matrix (to represent the neural network), where most of the cells are zeros. The sparsity of the matrix allows for reductions in storage and computation time. Unfortunately, these reductions do not scale proportionally to the fraction of zero entries in the sparse matrix. This is because sparse matrix algebra is not as efficient as dense matrix algebra. To illustrate, Table 1 shows the time it takes to multiply a matrix of indicated dimensions by a vector, when only 1% of the elements in the matrix are non-zero (as a fraction of time consumed when all elements are non-zeros). As can be seen, the processing time is significantly above 1%.

Table 1: Normalized processing time of a (m X k) matrix with only 1% non-zero entries, when multiplied by a (k X 1) vector. Observe that the processing time is significantly larger than 1% of the time needed to multiply a matrix with no zero entries. Hence, savings do not scale proportionally to the fraction of zero entries.

m	k	Time(Sparse/Dense)
100	100	51.7%
100	1000	29.1%
1000	100	33.6%
1000	1000	11.7%

An extension of DeepSense, called DeepIoT [6], removes nodes instead of edges to fix the above problem. Removal of entire nodes from the neural network is equivalent to removal of entire rows/columns from the corresponding matrix. This produces a new matrix that is also dense, but that has smaller dimensions. The approach was shown to be significantly more effective at reducing resource consumption.

DeepIoT searches a large space of reduced neural network architectures by dropping nodes according to some dropout probabilities. A thinned network structure is thus generated. The challenge is to set these dropout probabilities in an informed manner to find the optimal slim network structure that preserves the accuracy of sensing application results, while maximally reducing the resource consumption. To do so, DeepIoT exploits redundancies in the learned network structure. From the perspective of model compression, an element that is more redundant should have a higher probability to be dropped. A contribution of DeepIoT lies in exploiting this insight to significantly reduce network size. DeepIoT takes model parameters of each layer as input, learns parameter redundancies, and generates dropout probabilities accordingly, allowing for an efficient search in the space of reduced network architectures for one that minimally compromises quality of results [6]. The system was shown to reduce resource consumption by nearly two orders of magnitude, almost without jeopardizing result quality. Figure 2 shows representative measurements illustrating how output accuracy depends on the resource consumption of the neural network for an image recognition task. As the network is compressed, the inference time and inference energy are decreased (on the horizontal axis). Ultimately, quality is affected (see the vertical axis). The leftmost curve in both figures is for DeepIoT. It shows that it maintains the original quality of an uncompressed system the longest (until resources consumed drop by almost two orders of magnitude).

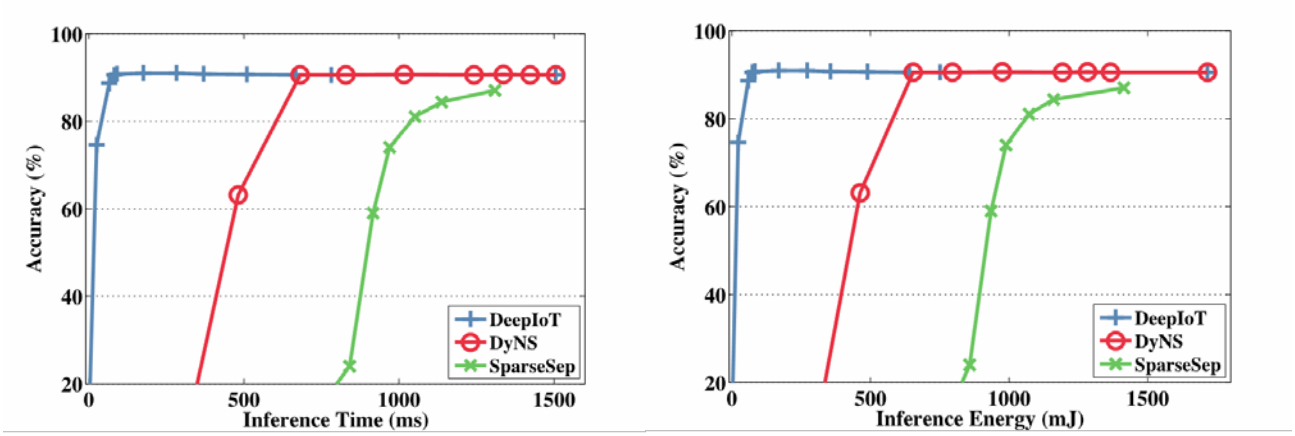

Figure 2: The relation between result accuracy and both inference time (left) and inference energy (right) of a neural network as it is compressed using three different approaches, when performing an image recognition task. DeepIoT shows the best trade-off between accuracy (of image recognition) and resource consumption.

The success of DeepIoT suggests an interesting question. If significantly reduced network structures are so good at representing the learned phenomena, what would happen if one simply used a much smaller neural network during training to begin with (instead of using a large one initially then reducing it after it is trained)? Interestingly, using a smaller network from the start does not lead to the same result. Rather, it reduces quality significantly. An informal intuition lies in the following analogy. Imagine resource consumption of a neural network to be represented by "elevation" in some feature space. It could be that some lower-elevation points that are particularly high quality are completely surrounded by high-elevation ridges in the feature space. These enclosed points are therefore unreachable from the "bottom" without climbing over a high-elevation ridge first to find them. Starting by training a large neural network is equivalent to climbing the high-elevation ridge. Starting from a small neural network does not explore the high elevation points, thus potentially failing at discovering some of the favourable local minima that are enclosed by the higher ridges. Those are discoverable only by training a larger network then compressing it in search of favourable local minima.

The above intuition reflects the fact that training a neural network is a highly nonlinear process that is prone to getting trapped in local minima. Training a lower-dimensional network from the start is likely to converge to a low-quality local minimum. In contrast, training a larger network then compressing it using the DeepIoT framework converges to a better solution.

3. OFFERING ASSURED RESULTS

The next problem concerns assessing the reliability of deep learning models. In particular, how does one offer principled uncertainty estimates that can faithfully reflect the correctness of model predictions? Principled uncertainty estimation is critical when deep learning is used to support applications that require quantified reliability assurances. Recent work focused on two related challenges. First, how to develop methods that provide accurate uncertainty estimates in prediction results obtained from deep learning models? Second, how to develop resource-efficient solutions for the uncertainty estimation problem, such that they can be implemented on resource-limited devices?

A simple, well-calibrated and efficient uncertainty estimation algorithm for the multi-sensor data fusion

problem, called RDeepSense, was recently proposed [3] as an extension of DeepSense [1]. It emits a distribution estimate instead of a point estimate at the output layer. Intuitively speaking, the algorithm models node outputs with random variables and estimates their distribution parameters. Estimation of the mean of the random variable is what traditional learning does. Estimation of the variance, however, is what yields confidence in results. A smaller estimated variance corresponds to a higher confidence in the computed mean.

Interestingly, the estimation of the mean and the estimation of the variance are interrelated. Typically, the estimator jointly determines both by minimizing some error function. The choice of that function has an important effect on estimation accuracy of the two parameters. Specifically, using common error functions, such as the mean square error, was shown underestimate the uncertainty. This is so because such an estimator predicts a very accurate mean value. If the mean value is estimated well, the variance observed around that mean on training data is small and may thus underestimate variance encountered later during testing. In contrast, when using a nonlinear error function, such as the negative log-likelihood, the estimated mean is often biased (because the nonlinearity penalizes erring on one side more than erring on the other, causing the estimated mean to drift towards the heavily penalized side). The biased (i.e., incorrect) mean estimate results in increased measured variance around the mean, leading to an artificially inflated uncertainty estimate.

RDeepSense exploits the above intuition to arrive at an estimate of variance that neither underestimates nor overestimates the true value. Specifically, it uses a weighted sum of the above two error functions (namely, means square error and negative log-likelihood) as the combined loss function. The weights are adjusted (calibrated) such that the underestimation and overestimation roughly cancel out. RDeepSense was shown to generate very good uncertainty estimates that allow defining accurate confidence intervals for outputs of the deep learner. An example is shown in Figure 3.

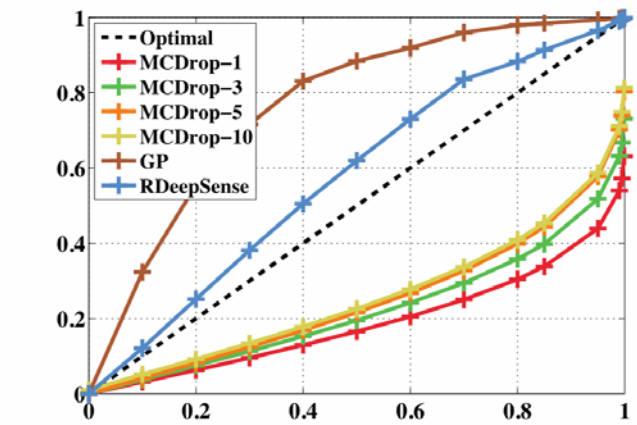

Figure 3: The relation between estimated confidence window (horizontal axis) and percentage of data that falls in that window (vertical axis) for different confidence estimation algorithms. When confidence is properly estimated, x% of the data should fall in an x% confidence window. The RDeepSense curve comes closest to such an identity function (a diagonal line).

Figure 3 shows the relation between the confidence window and the percentage of outputs that fall in that window for different confidence estimation algorithms applied to a representative learning task. The task in question was to estimate commute delays between points in New York City as a function of related variables,

such as start and end locations and time of day. For example, when an 80% confidence window is computed, 80% of the data should fall in that window. Similarly when a 90% confidence window is computed, 90% of the data should fall in that window. In other words, the ideal relation between the confidence window (shown on the horizontal axis) and the percentage of data falling into it (shown on the vertical axis) should be the identity function (a diagonal line). As can be seen from the figure, RDeepSense comes closest to the diagonal line in this experiment. Regarding resource efficiency, since RDeepSense emits a distribution estimate instead of a point estimate at the output layer, it does uncertainty estimation in a single run. Compared with sampling-based and ensemble-based methods [4], RDeepSense has a much reduced execution time and energy consumption.

The ability to compute confidence in deep learning results offers another interesting resource optimization possibility. Namely, one may structure a deep neural network into stages, each consisting of several layers, and compute confidence in (intermediate) results after each stage. Once a high-enough confidence is reported, it becomes possible to skip the execution of the remaining stages. For example, consider a deep neural network whose job is to identify the presence of humans in a landscape. The presence of humans may be easier to identify in some images than others. Consequently, it could be that fewer stages need to be executed for some images to reach an acceptable level of confidence in results. Figure 4 depicts a scheduler that uses the above intuition to schedule multiple inference tasks in a manner that maximizes marginal increase in confidence. In this framework, once a point of diminishing return is reached during the execution of an inference task, the inference task in question can be stopped and results returned even if some stages of the corresponding deep neural network have not yet been executed.

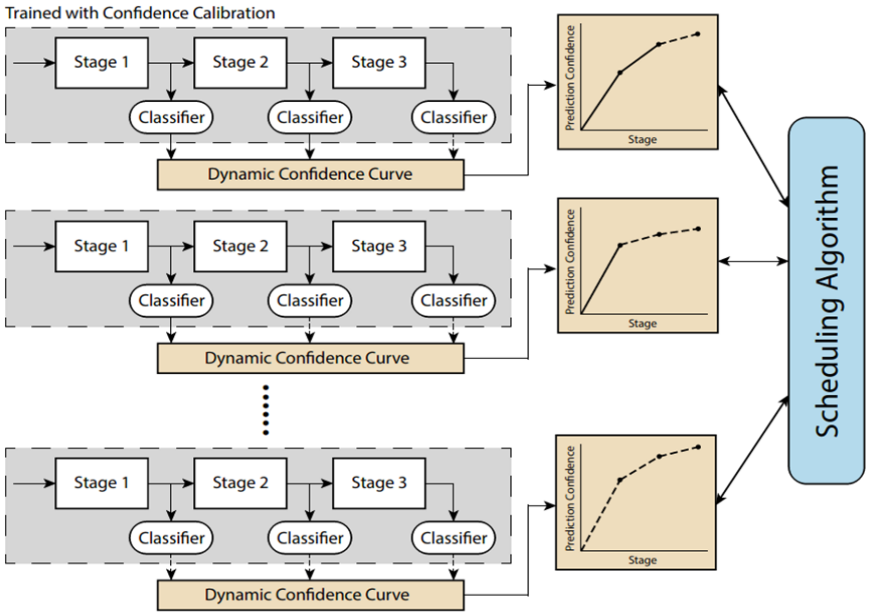

Figure 4: An uncertainty-aware framework for deep neural network execution. Inference tasks are executed until the point of diminishing returns (in confidence in result quality). The rest of the deep neural network is then skipped

4. LEARNING FROM UNLABELED DATA

A general disadvantage of deep learning methods lies in the need for large amounts of labelled data. To learn well from empirical measurements, the neural network must be given a sufficient number of labelled examples from which network parameters are to be estimated. Since the number of parameters is large, so is the required number of labelled examples. This needs for labelling offers a significant practical impediment to the use of deep learning in practical contexts, where labelling cannot be easily done.

Recently, Generative adversarial networks (GAN) have been proposed as a promising deep learning technique for unsupervised and semi-supervised learning [2]. The GAN training strategy is to define a game between two competing networks; a generator network and a discriminator network. The generator network picks arbitrary labels then selects the most appropriate input data for the label from the (unlabelled) input space. The discriminator network receives a mix of samples produced by the generator network (i.e., labels paired with previously unlabelled input measurements), and true (correctly labelled) data. Its goal is to learn to distinguish between the two. An error function for the discriminator computes how successful it is at separating originally labelled data from data labelled by the generator.

The generator is trained to fool the discriminator. As the discriminator learns to separate the two data streams, it shares its model with the generator. The generator uses that model to increase its sophistication in generating realistic-looking samples to fool the discriminator. The GAN training strategy thus leverages the unlabelled data to improve both the generator and discriminator, which results in improving the ability of the system to label data in a manner that is indistinguishable from known ground truth samples, essentially improving the mapping from the input space to the output space taking into account both labelled and unlabelled inputs. Evaluation shows that the semi-supervised strategy greatly improves performance (over using labelled data alone), while reducing the need for larger amounts of labelled data.

The intuition why unlabelled data can improve the performance of learning lies in that it carries information on the structure of the input space. For example, assume we knew that sensors were inserted into lakes to measure their chemical composition. Figure 5-a (left) shows to samples (filled-in black circles) labelled lake A and B, as well as one unlabelled sample (filled-in grey circle). It is hard to guess which lake the unlabelled sample belongs to. Figure 5-b (right), on the other hand, shows the aforementioned samples together with many more unlabelled ones. The new unlabelled samples give insight into the structure of the input space (i.e., the lakes) from which the samples were drawn. It strongly suggests that the unlabelled grey circle belongs to lake A. In fact, most of the data labels can be guessed at that point.

Figure 5: Illustration of the basic insight why unlabelled data helps the learning process. On the left (4-a) only one unlabelled point is given. It is hard to guess the label. On the right (4-b) more unlabelled points are shown. Consequently, labels are easier to guess.

5. CONCLUSIONS

The above discussion presented recent advances in deep learning that customize the computational power of this approach to the needs of multi-sensor data fusion systems operating in potentially resource-scarce environments. The presented results suggest that substantial resource economies are possible by compressing larger neural networks to fit embedded devices without an appreciable quality degradation. Surveyed recent work also suggests that accurate estimates of confidence in results can be attained and that unlabelled data can be used to improve learning quality. Several questions are left for future work. For example, how can different devices share the deep learning burden? In an environment with many computing nodes with different connections to various sensors, how much of the deep neural network architecture should run on which node? Can one exploit sensing redundancies to further reduce the resource consumption of learning algorithms? Are current deep learning methods optimal for signal processing tasks that are typically performed in the frequency domain (as opposed to the time domain)? While one can ingest frequency domain features as deep learning inputs, would additional improvements be gained if the training algorithms were re-designed specifically for frequency domain signals? How susceptible are deep learning systems to adversarial manipulation, such as wilful mislabelling of data or compromised sensor measurements? Does their susceptibility increase when compressed to fit on smaller devices, or does compression actually improve robustness? The argument for the latter would follow the common wisdom that more complex solutions are usually more sensitive to assumptions that underlie their correctness, whereas simpler solutions (while not as performant) are often more robust. Can one add explainability in addition to confidence when presenting deep learning results? These topics are currently an active area of research in investigating advantages of deep learning for multi-sensor fusion systems.

ACKNOWLEDGEMENTS

Research reported in this paper was sponsored in part by the Army Research Laboratory under Cooperative Agreement W911NF-17-2-0196 (ARL IoBT CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

[1] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, Tarek Abdelzaher, "DeepSense: A Unified Deep Learning Framework for Time-Series Mobile Sensing Data Processing," In Proc. *International World Wide Web Conference (WWW)*, Perth, Australia, April 2017.

[2] Shuochao Yao, Yiran Zhao, Huajie Shao, Chao Zhang, Aston Zhang, Shaohan Hu, Dongxin Liu, Shengzhong Liu, Lu Su, Tarek Abdelzaher, "SenseGAN: Enabling Deep Learning for Internet of Things with a Semi-Supervised Framework," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)*, 2018. (Also appears in UbiComp '18, Singapore, October 2018.)

[3] Shuochao Yao, Yiran Zhao, Huajie Shao, Aston Zhang, Chao Zhang, Shen Li, and Tarek Abdelzaher, "RDeepSense: Reliable Deep Mobile Computing Models with Uncertainty Estimation," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)*, 2018. (Also appears in UbiComp '18, Singapore, October 2018.)

[4] Shuochao Yao, Yiran Zhao, Huajie Shao, Chao Zhang, Aston Zhang, Dongxin Liu, Shengzhong Liu, Lu Su and Tarek Abdelzaher, "ApDeepSense: Deep Learning Uncertainty Estimation Without the Pain for IoT Applications," In Proc. *IEEE International Conference on Distibuted Computing Systems (ICDCS)*, Vienna, Austria, July 2018.

[5] Shuochao Yao, Yiran Zhao, Aston Zhang, Shaohan Hu, Huajie Shao, Chao Zhang, Lu Su, Tarek Abdelzaher, "Deep Learning for the Internet of Things," *IEEE Computer*, Vol. 51, No. 5, May 2018.

[6] Shuochao Yao, Yiran Zhao, Aston Zhang, Lu Su, and Tarek Abdelzaher, "DeepIoT: Compressing Deep Neural Network Structures for Sensing Systems with a Compressor-Critic Framework," In Proc. *15th ACM Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Delft, The Netherlands, November 2017.