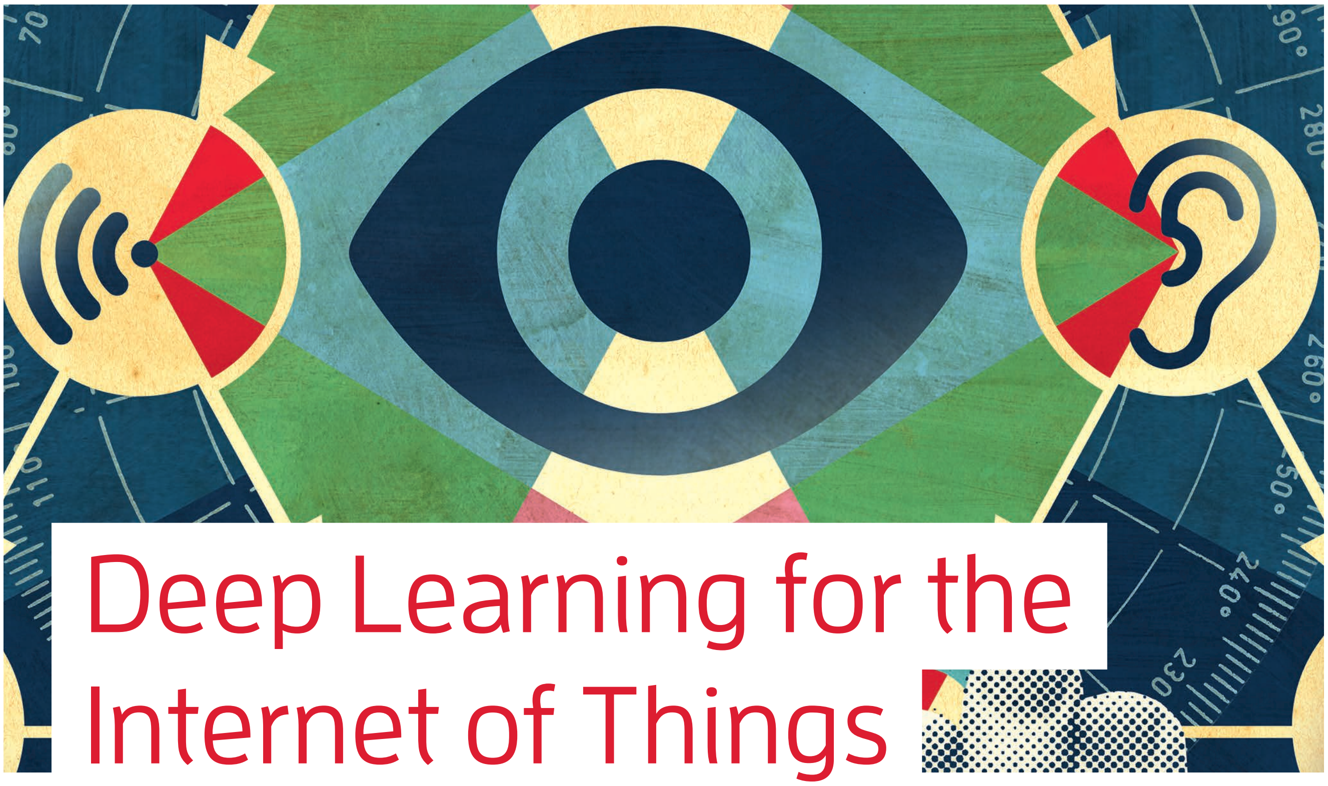

Deep Learning for the Internet of Things

Shuochao Yao and Yiran Zhao, University of Illinois Urbana-Champaign (UIUC)
Aston Zhang, Amazon AI
Shaohan Hu, IBM Thomas J. Watson Research Center
Huajie Shao and Chao Zhang, UIUC
Lu Su, State University of New York, Buffalo
Tarek Abdelzaher, UIUC

How can the advantages of deep learning be brought to the emerging world of embedded IoT devices? The authors discuss several core challenges in embedded and mobile deep learning, as well as recent solutions demonstrating the feasibility of building IoT applications that are powered by effective, efficient, and reliable deep learning models.

The proliferation of internetworked mobile and embedded devices leads to visions of the Internet of Things (IoT), giving rise to a sensor-rich world where physical things in our everyday environment are increasingly enriched with computing, sensing, and communication capabilities. Such capabilities promise to revolutionize the interactions between humans and physical objects.

Indeed, significant research efforts have been spent toward building smarter and more user-friendly applications on mobile and embedded devices and sensors. At the same time, recent advances in deep learning have greatly changed the way that computing devices process human-centric content such as images, video, speech, and audio. Applying deep neural networks to IoT devices could thus bring about a generation of applications capable of performing complex sensing and recognition tasks to support a new realm of interactions between humans and their physical surroundings. This article discusses four key research questions toward the realization of such novel interactions between humans and (deep-) learning-enabled physical things, namely: What deep neural network structures can effectively process and fuse sensory input data for diverse IoT applications? How to reduce the resource consumption of deep learning models such that they can be efficiently deployed on resource-constrained IoT devices? How to compute confidence measurements in the correctness of deep learning predictions for IoT applications? Finally, how to minimize the need for labeled data in learning?

To elaborate on the above challenges, first, observe that IoT applications often depend on collaboration among multiple sensors, which requires designing novel neural network structures for multisensor data fusion. These structures should be able to model complex interactions among multiple sensory inputs over time and effectively encode features of sensory inputs that are pertinent to desired recognition and other tasks. We review a general deep learning framework for this purpose, called DeepSense,[1] that provides a unified yet customizable solution for the learning needs of various IoT applications. It demonstrates that certain combinations of deep neural network topologies are particularly well-suited for learning from sensor data.

Second, IoT devices are usually low-end systems with limited computational, energy, and memory resources. One key impediment in deploying deep neural networks on IoT devices therefore lies in the high resource demand of trained deep neural network models. While existing neural network compression algorithms can effectively reduce the number of model parameters, not all of these models lead to matrix representations that can be efficiently implemented on commodity IoT devices. Recent work describes a particularly effective deep learning compression algorithm, called DeepIoT,[2] that can directly compress the structures of commonly used deep neural networks. The compressed model can be deployed on commodity devices. A large proportion of execution time, energy, and memory can be reduced with little effect on the final prediction accuracy.

Third, reliability assurances are important in cyber-physical and IoT applications. The need for offering such assurances calls for well-calibrated estimation of uncertainty associated with learning results. We present simple methods for generating well-calibrated uncertainty estimates for the predictions computed in deep neural networks, called RDeepSense.[3] It achieves accurate and well-calibrated estimations by changing the objective function to faithfully reflect prediction correctness.

Finally, labeling data for learning purposes is time-consuming. One must teach sensing devices to recognize objects and concepts without the benefit of (many) examples, where ground truth values for such objects and concepts are given. Unsupervised and semisupervised solutions are needed to solve the challenge of learning with limited labeled (and mostly unlabeled) samples, while approaching the performance of learning from fully labeled data.

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

ON DEEP LEARNING MODELS FOR SENSOR DATA

A key research challenge toward the realization of learning-enabled IoT systems lies in the design of deep neural network structures that can effectively estimate outputs of interest from noisy time-series multisensor measurements.[1]

Despite the large variety of embedded and mobile computing tasks in IoT contexts, one can generally categorize them into two common subtypes: estimation tasks and classification tasks, depending on whether prediction results are continuous or categorical, respectively. The question therefore becomes whether or not a general neural network architecture exists that can effectively learn the structure of models needed for estimation and classification tasks from sensor data. Such a general deep learning neural network architecture would, in principle, overcome disadvantages of today's approaches that are based on analytical model simplifications or the use of hand-crafted engineered features.

Traditionally, for estimation-oriented problems such as tracking and localization, sensor inputs are processed based on the physical models of the phenomena involved. Sensors generate measurements of physical quantities such as acceleration and angular velocity. From these measurements, other physical quantities are derived (such as displacement through double integration of acceleration over time). However, measurements of commodity sensors are noisy. The noise in measurements

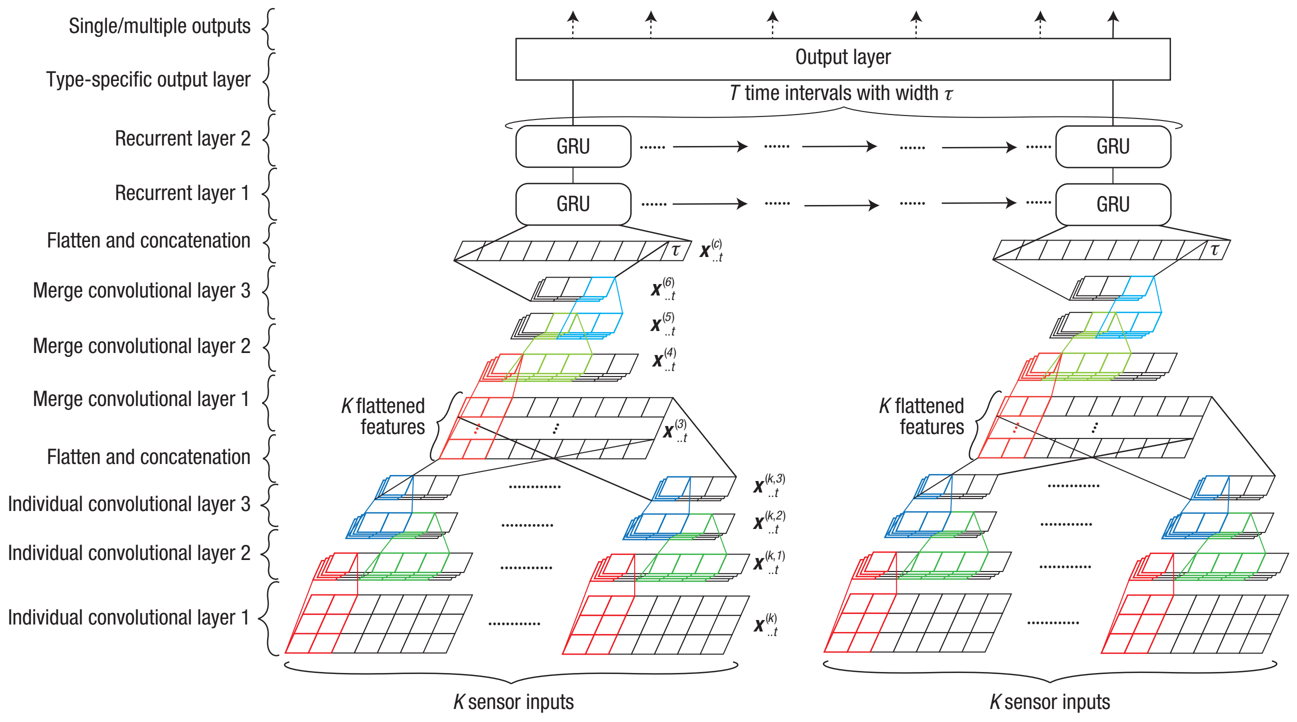

FIGURE 1. Main architecture of the DeepSense framework.

is nonlinear and might be correlated over time, which makes it hard to model. It is therefore challenging to separate signal from noise, leading to estimation errors and bias.

For classification-oriented problems, such as activity and context recognition, a typical approach is to compute appropriate features derived from raw sensor data. These hand-crafted features are then fed into a classifier for training. Designing good hand-crafted features can be time-consuming; it requires extensive experiments to generalize well to diverse settings such as different sensor noise patterns and heterogeneous user behaviors.

A general deep learning framework can effectively address both of the aforementioned challenges by automatically adapting the learned neural network to complex correlated noise patterns while, at the same time, converging on the extraction of maximally robust signal features that are most suited for the task at hand. A recent framework, called DeepSense, demonstrates a case for feasibility of such a general solution.

As shown in Figure 1, DeepSense integrates convolutional neural networks (CNNs) and recurrent neural networks (RNNs). Sensory inputs are aligned and divided into time intervals for processing time-series data. For each interval, DeepSense first applies an individual CNN to each sensor, encoding relevant local features within the sensor's data stream. Then, a (global) CNN is applied on the respective outputs to model interactions among multiple sensors for effective sensor fusion. Next, an RNN is applied to extract temporal patterns. At last, either an affine transformation or a softmax output is used, depending on whether we want to model an estimation or a classification task.

This architecture solves the general problem of learning multisensor fusion tasks for purposes of estimation or classification from time-series data. For estimation-oriented problems, DeepSense learns the physical system and noise models to yield outputs from noisy sensor data directly. The neural network acts as an approximate transfer function. For classification-oriented problems, the neural network acts as an automatic feature extractor encoding local, global, and temporal information.

As a unified model, DeepSense can be easily customized for a specific IoT application. The application designer needs only to decide on the number of sensory inputs, input/output dimensions, and the training objective function. The detailed mathematical formulation of DeepSense can be found in a related article.[1]

Encouraging results were reported on applying DeepSense in two representative sensing tasks: *heterogeneous human activity recognition* (HHAR) and *user identification with biometric motion analysis* (UserID). HHAR is a motion-sensor-based activity recognition task. It is tested on new users who have not appeared in the training set. In contrast, UserID uses motion sensors for user identification from activities such as walking, biking, and climbing stairs.

To understand the contributions of different architectural components, variants of the DeepSense model were introduced by removing some design component(s) from the general architecture. DS-singleGRU simplifies the RNN by replacing its two-layer stacked GRU architecture with a single-layer GRU of a larger dimension, while keeping the number of parameters the same. DS-noIndvConv skips the convolutional subnets for individual sensors, keeping a single CNN that merges data from all sensors in each time window. Finally, DS-noMergeConv skips the global convolutional subnet that merges sensor data. Instead, it flattens the output of each individual convolutional subnet and concatenates them into a single vector as the input to the RNN.

These models (together with the overall DeepSense model) were compared to various custom-designed or hand-crafted baselines for each

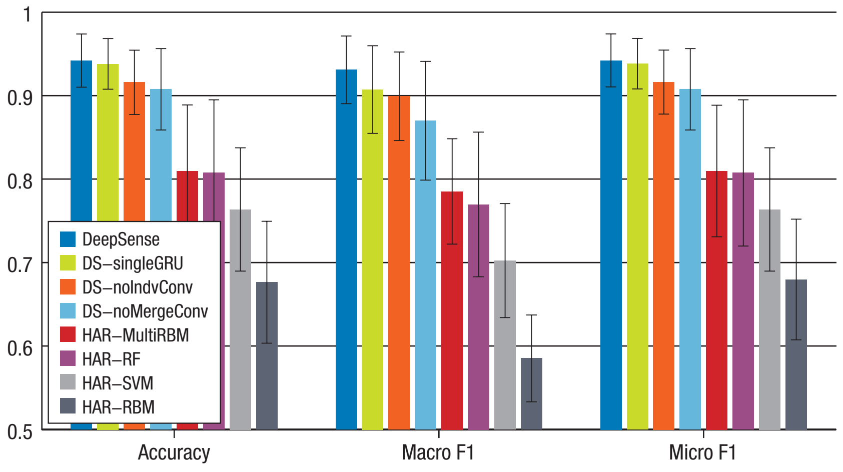

FIGURE 2. Performance metrics of heterogeneous human activity recognition (HHAR) task with the DeepSense framework.

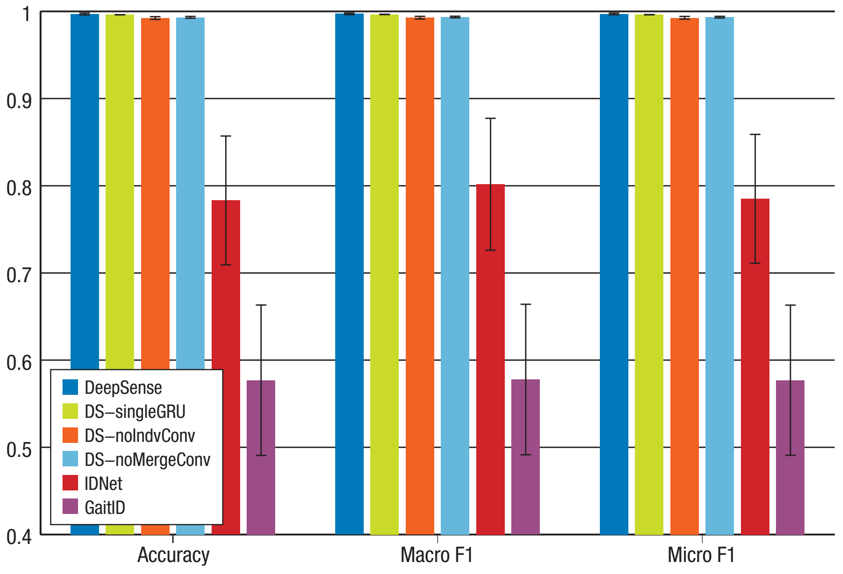

FIGURE 3. Performance metrics of UserID task with the DeepSense framework.

application, including HAR-RF,[4] HAR-SVM,[4] HRA-RBM, and HRA-MultiRBM[5] for activity recognition, and GaitID[6] and IDNet[7] for used identification.

Accuracy results in performing HHAR and UserID tasks are illustrated in Figures 2 and 3, respectively. The DeepSense based algorithms (including DeepSense and its three variants) outperform other baseline algorithms by a large margin (that is, at least 10 percent for HHAR and at least 20 percent for UserID). The results offer anecdotal evidence that a general deep learning architecture can beat hand-crafted solutions designed for the individual application spaces. Although current work is by no means a consummate proof of generalizability, this property (if true) would be very important, because a main appeal of applying deep learning in IoT contexts lies in obviating the need for per-application customization of theoretical derivations and hand-crafted features. More research is needed to substantiate or refute the early evidence and to understand the limits of generalizability of learning models across IoT systems.

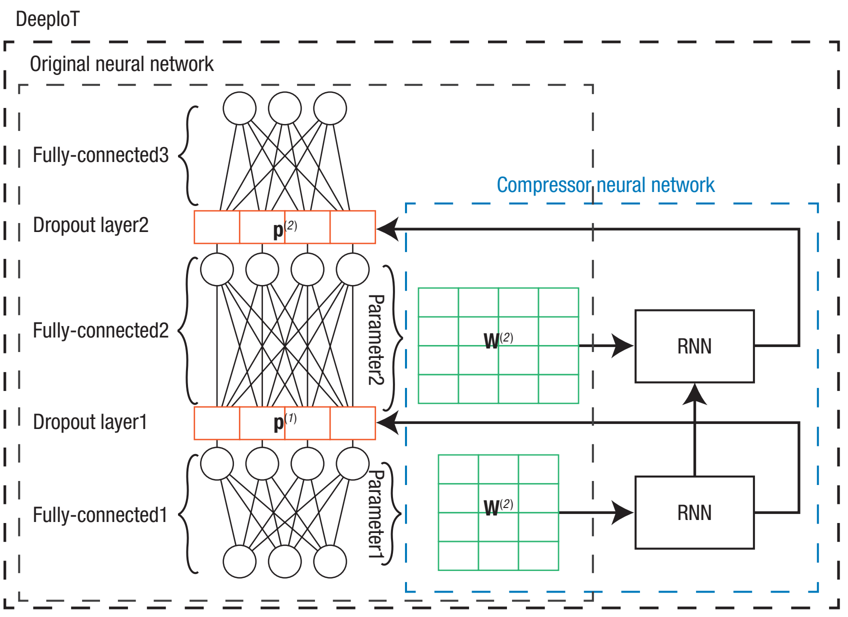

FIGURE 4. Overall DeepIoT system framework. Orange boxes represent dropout operations. Green boxes represent parameters of the original neural network.

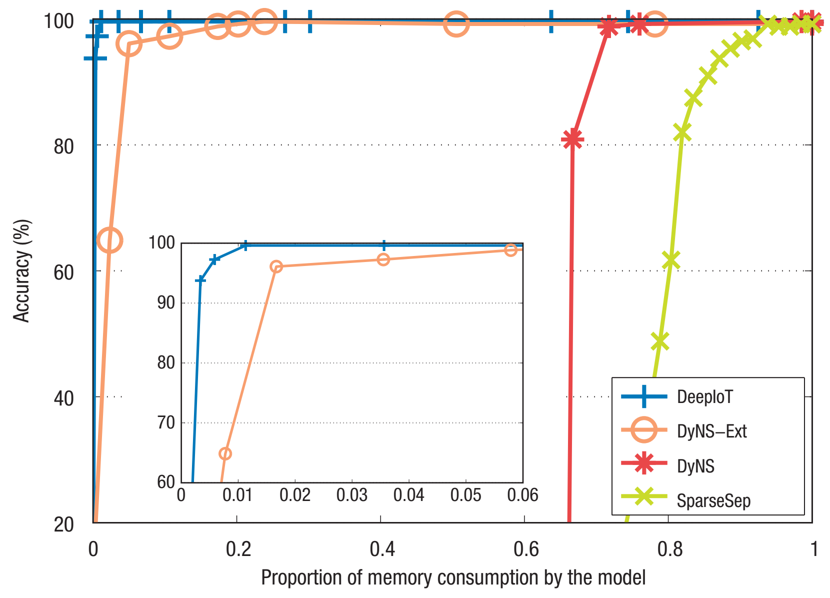

FIGURE 5. The tradeoff between testing accuracy and memory consumption by models.

COMPRESSING NEURAL NETWORK STRUCTURES

Resource constraints of IoT devices remain an important impediment toward deploying deep learning models. A key question is therefore whether it is possible to compress deep neural networks, such as those described in the previous section, to a point where they fit comfortably on low-end embedded devices, enabling real-time "intelligent" interactions with their environment. Can a unified approach compress commonly used deep learning structures, including fully connected, convolutional, and recurrent neural networks, as well as their combinations? To what degree does the resulting compression reduce energy, execution time, and memory needs in practice?[2]

An illustration of such a compression framework, called DeepIoT,[2] is shown in Figure 4. DeepIoT borrows the idea of dropping hidden elements from a widely used deep learning regularization method called dropout. The dropout operation gives each hidden element a dropout probability. During the dropout process, hidden elements can be pruned according to their dropout probabilities. A "thinned" network structure can thus be generated. The challenge is to set these dropout probabilities in an informed manner to generate the optimal slim network structure that preserves the accuracy of sensing applications while maximally reducing their resource consumption. An important purpose of DeepIoT is thus to find the optimal dropout probability for each hidden element in the neural network.

To obtain the optimal dropout probabilities for nodes in the neural network, DeepIoT exploits the network parameters themselves. From the perspective of model compression, an element that is more redundant should have a higher probability of being dropped. A contribution of DeepIoT lies in exploiting a novel compressor neural network to solve this problem. It takes model parameters of each layer as input, learns parameter redundancies, and generates the dropout probabilities accordingly. The compressor neural network is optimized jointly with the original neural network to be compressed in an iterative manner that tries to minimize the loss function of the original IoT application.

Evaluation shows that the DeepIoT compression algorithm is able to greatly reduce the network size, execution time, and energy consumption without hurting the prediction accuracy.[2] We continue to use UserID as the running application examples, and compare compression efficacy to that of several baselines; namely, DyNS,[8] SparseSep,[9] and DyNS-Ext. DyNS is a magnitude-based network pruning algorithm that prunes weights in convolutional kernels and fully connected layers based on their magnitude. SparseSep simplifies the fully connected layer by the sparse coding technique, and compresses the convolutional layer with matrix factorization. DyNS-Ext extends the magnitude-based method used in DyNS to recurrent layers. Just like DeepIoT, DyNS-Ext can be applied to all commonly used deep network modules, including fully connected layers, convolutional layers, and recurrent layers. All models use 32-bit floats without quantization. Experiments are conducted on the Intel Edison platform.

The detailed tradeoff between testing accuracy and memory consumption of the resulting models is illustrated in Figure 5. We compress the original DeepSense neural network with different compression ratios and observe the final testing accuracy. DeepIoT achieves the best tradeoff.

The tradeoff between execution time and testing accuracy is shown in Figure 6. Similarly, the tradeoff between energy consumption and testing accuracy is shown in Figure 7. DeepIoT offers the best reduction in execution time (approximately 80.8 percent) as well as the best reduction in energy consumption (approximately 83.3 percent) without apparent loss in accuracy.

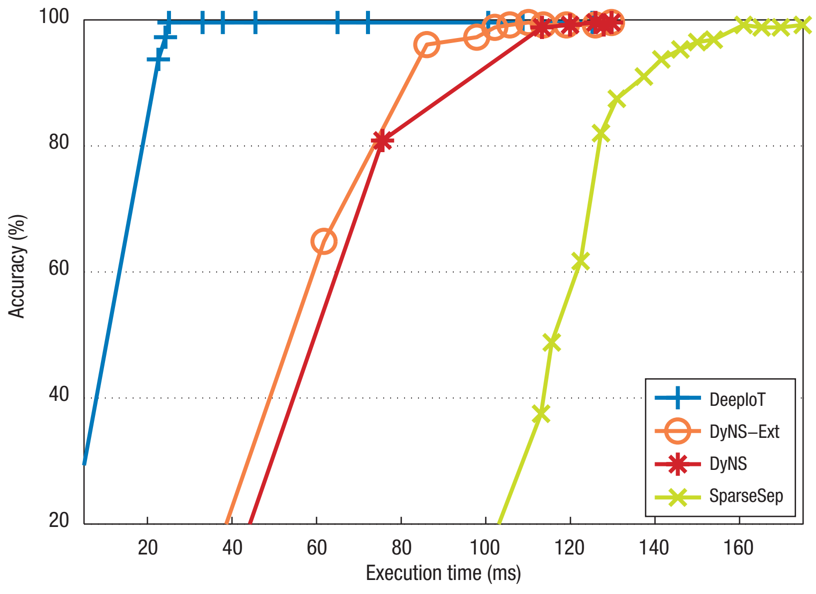

FIGURE 6. The tradeoff between testing accuracy and execution time.

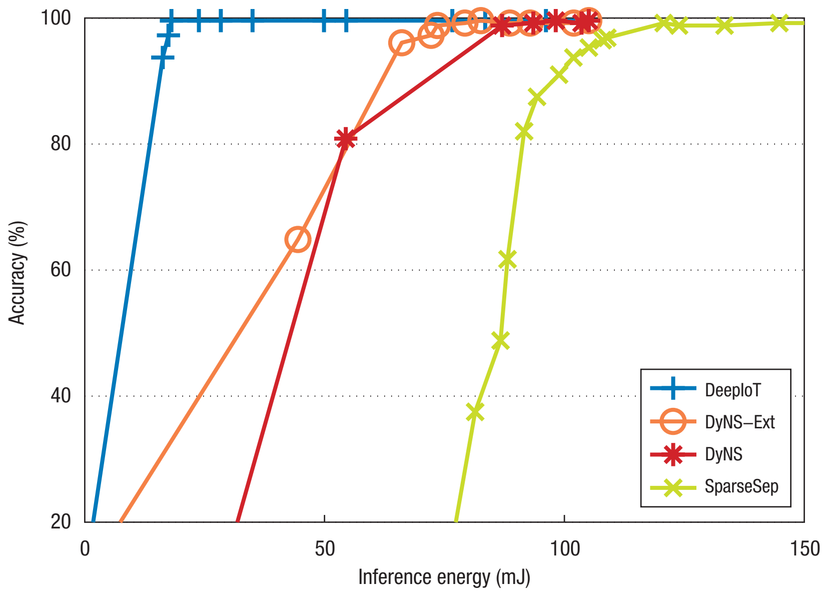

FIGURE 7. The tradeoff between testing accuracy and energy consumption.

The ability of compression algorithms to significantly reduce network size without affecting accuracy suggests that the underlying models of IoT applications are inherently low-dimensional, thus allowing for significant simplifications of learned neural network structures. This is good news in terms of feasibility of implementation on resource-limited hardware, such as the Edison board used on the above evaluation.

ESTIMATING UNCERTAINTY

The next problem concerns the reliability of deep learning models. In particular, how to 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 IoT applications that require quantified reliability assurances.

Recent work focused on two related challenges:

TABLE 1. Mean absolute error (MAE) and negative log-likelihood (NLL) for the **NYCommute** task.

Deep learning algorithm	MAE	NLL
RDeepSense	5.64	7.7
SSP-1	8.15	4.86
SSP-3	7.90	4.67
SSP-5	7.51	4.84
SSP-10	7.03	4.81
MCDrop-3	5.69	19,995.6
MCDrop-5	5.64	1,335.73
MCDrop-10	5.61	640.35
MCDrop-20	5.61	640.35
Gaussian Process	11.84	7.46

- how to develop methods that provide accurate uncertainty estimates in prediction results obtained from deep learning models, and
- how to develop resource-efficient solutions for the uncertainty estimation problem, such that they can be implemented on resource-limited IoT devices.

In this section, we introduce a simple, well-calibrated, and efficient uncertainty estimation algorithm for a multilayer perceptron (MLP) called RDeepSense.[3] RDeepSense enables uncertainty estimation with theoretically proven error bounds for IoT applications.

There are only two steps in computing uncertainty for an arbitrary fully connected neural network. First, insert dropout operations to each fully connected layer. Second, adopt a proper scoring rule as the loss function and emit a distribution estimate instead of a point estimate at the output layer.

Intuitively speaking, the dropout operations convert a traditional (deterministic) neural network with parameters into a random Bayesian neural network model with random variables, which equates a neural network to a statistical model. Proper scoring rules (based on the loss function) then measure the accuracy of probabilistic predictions.

The loss function has a large effect on the final results. Taking a regression problem as an example, using the mean square error as the loss function tends to underestimate the uncertainties. This is so because the training process is focused on predicting an accurate mean value without concerning itself with the variance. At the same time, using negative log-likelihood as the loss function tends to overestimate the uncertainties. The reason is that, during the early phase of training a neural network with log-likelihood loss, it is relatively hard to generate an accurate estimate of the mean. Increasing the value of estimated variance can consistently decrease the negative log-likelihood loss with a high probability. Therefore, the predicted uncertainty tends to favor a larger variance that overestimates the true uncertainty.

RDeepSense applies a tunable function, based on a weighted sum of negative log-likelihood and mean square error, as the loss function. The underestimation effect of mean square error and the overestimation effect of negative log-likelihood are thus balanced by tuning the weighted sum. RDeepSense was shown to generate well-calibrated uncertainty estimates.

Regarding resource efficiency, since RDeepSense emits a distribution estimate instead of a point estimate at the output layer, it can do the uncertainty estimation in a single run. Compared with sampling-based and ensemble-based methods that require running a model k times for k samples, RDeepSense results in much reduced execution time and energy consumption.

We evaluate the accuracy of uncertainty estimation of RDeepSense and related baselines on the *NYCommute* task. *NYCommute* predicts commute times in New York City based on a data set of taxi-cab pick-up/drop-off times and locations.

We compare RDeepSense to three baseline algorithms. They are called MCDrop,[10] SSP,[11] and Gaussian Process (GP). All deep-learning-based algorithms use a four-layer fully connected neural network with 500 hidden dimensions. MCDrop is based on a Monte Carlo dropout. Compared with RDeepSense, the main difference is that MCDrop is not optimized by a proper scoring rule. MCDrop requires running the neural network multiple times to generate samples for uncertainty estimation. We use MCDrop-k to represent MCDrop with k samples. SSP trains the neural network with proper scoring methods. Compared with RDeepSense, the main difference is that SSP uses the ensemble method instead of the dropout operation in each layer. SSP requires training multiple neural networks for the ensemble. We use SSP-k to represent SSP with an ensemble of k individual neural networks. GP is a Gaussian-process-based algorithm. It is used to illustrate the quality of uncertainty estimation generated by a statistical model. In testing, we compute the z% confidence interval based on the predicted mean and variance of each algorithm. We then measure the fraction of the testing data that falls into this confidence interval. For a well-calibrated uncertainty estimation, the fraction of testing data that falls into the confidence interval should be similar to z%.

The comparison result is shown in Table 1. MCDrop-k shows low MAE and high NLL, while SSP-k shows high MAE and low NLL. MCDrop-k tries to minimize the mean square error, while SSP-k tries to minimize the negative log-likelihood. Therefore, MCDrop-k focuses more on the mean of predictive distribution, and SSP-k focuses more on the overall likelihood. RDeepSense combines two objective functions, mean square error and negative log-likelihood, to find a balance between these two.

The calibration curves are illustrated in Figures 8 and 9. Both MCDrop-k and SSP-k fail to generate high-quality uncertainty estimates, either underestimating or overestimating the uncertainty. However, RDeepSense provides uncertainty estimates with good quality, outperforming GP by a significant margin. The results offer a path toward accurate estimation of uncertainty in outputs of deep learning models.

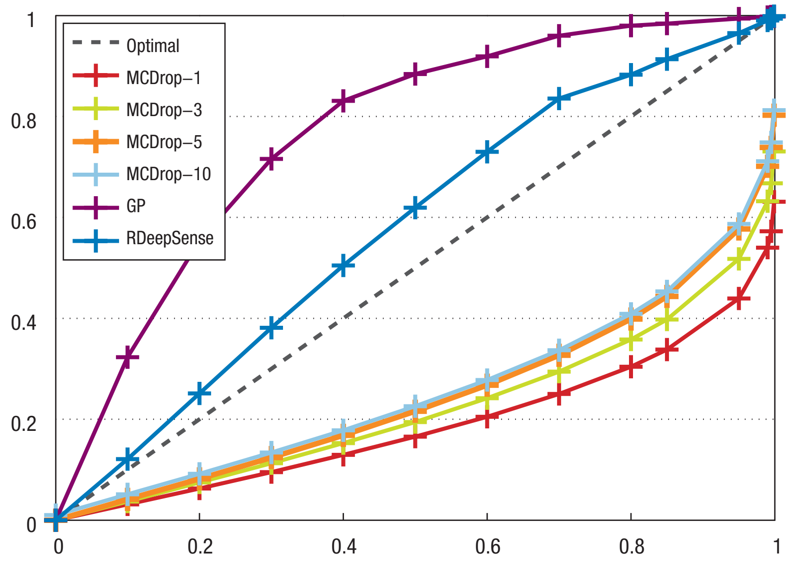

FIGURE 8. The calibration curves of RDeepSense, GP, and MCDrop-k.

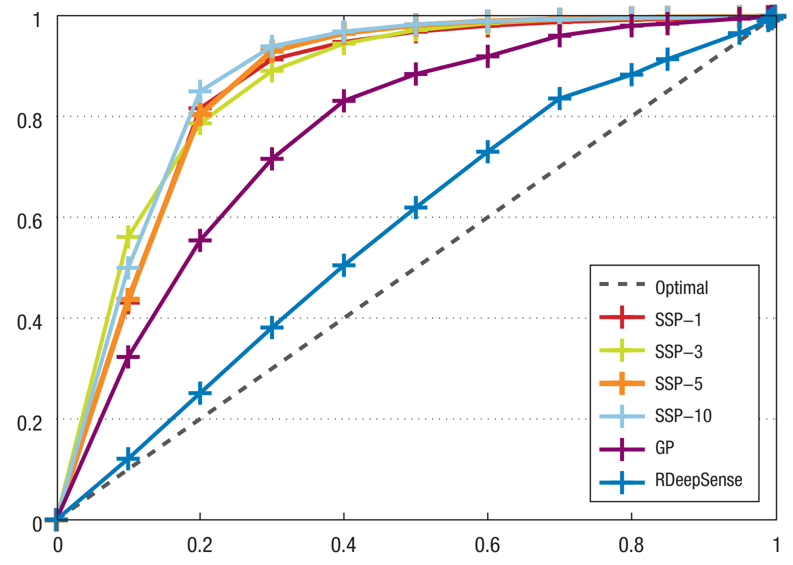

FIGURE 9. The calibration curves of RDeepSense, GP, and SSP-k.

MINIMIZING LABELED DATA

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

Recently, generative adversarial networks (GAN) has been proposed as a promising deep learning technique for unsupervised and semisupervised learning.[12] The GAN training strategy is to define a game between two competing networks. The generator network maps a source of noise to the input space. The discriminator network receives either a generated sample or a true data sample and must distinguish between the two. The generator is trained to fool the discriminator. Here, we define the input probabilistic space as the joint probabilistic distribution of input sensory data and classification label. The GAN training

TABLE 2. Semisupervised training of HHAR with DeepSense framework.

p%	10%	5%	3%	2%	1%
Sense-GAN	94.8%	92.5%	91.4%	90.4%	88.3%
DeepSense	92.0%	89.3%	85.3%	83.6%	79.1%

strategy leverages the unlabeled data to increase the capacity of generator and discriminator networks, which explicitly improve the discriminating ability of classifier in return.

Evaluation shows that the semisupervised strategy, called SenseGAN, greatly reduces the requirements of labeled data. We continue to use HHAR with DeepSense framework[1] as the running application example, where we take *p*% of the overall dataset as labeled data.

As shown in Table 2, the semisupervised training can preserve the classification accuracy with only 10 percent of labeled data by leveraging 90 percent of unlabeled data. However, extensive studies are still needed to explore the possibility of training with fewer number of labeled as well as unlabeled data in IoT context.

We introduced challenges and emerging solutions that suggest feasibility of building effective, efficient, and reliable IoT systems enriched with deep learning techniques. More studies are needed to further verify the applicability of results. Can one build a unified deep learning framework for largely heterogeneous sensory inputs, such as audio signals, Wi-Fi signals, and motion inputs? What are the impact of neural network compression on system performance, such as execution time and energy consumption? Can one extend uncertainty measurements to other deep learning models besides MLPs? How does one learn in highly dynamic environments where it is impossible to collect a large number of data samples? More investigation is needed to address these questions. ■

ACKNOWLEDGMENTS

Research reported in this paper was sponsored in part by NSF under grants CNS 16-18627 and CNS 13-20209 and in part by the Army Research Laboratory under Cooperative Agreements W911NF-09-2-0053 and W911NF-17-2-0196. 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, NSF, 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. S. Yao et al., "DeepSense: A Unified Deep Learning Framework for Time-Series Mobile Sensing Data Processing," *Proc. 26th Int'l Conf. World Wide Web*, 2017, pp. 351–360.
2. S. Yao et al., "DeepIoT: Compressing Deep Neural Network Structures for Sensing Systems with a Compressor-Critic Framework," *Proc. 15th ACM Conf. Embedded Network Sensor System*, 2017; https://arxiv.org/abs/1706.01215.
3. S. Yao et al., "RDeepSense: Reliable Deep Mobile Computing Models with Uncertainty Estimations," *Proc. ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 4, 2018, p. 173.
4. A. Stisen et al., "Smart Devices are Different: Assessing and Mitigating Mobile Sensing Heterogeneities for Activity Recognition," *Proc. 13th ACM Conf. Embedded Network Sensor System* (SenSys 15), 2015, pp. 127–140.
5. V. Radu et al., "Towards Multimodal Deep Learning for Activity Recognition on Mobile Devices," *Proc. ACM Int'l Joint Conf. Pervasive and Ubiquitous Computing: Adjunct* (UbiComp 16), 2016, pp. 185–188.
6. H.M. Thang et al., "Gait Identification Using Accelerometer on Mobile Phone," *Proc. Int'l Conf. Control, Automation and Information Sciences* (ICCAIS 12), 2012, https://doi.org/10.1109/ICCAIS.2012.6466615.
7. M. Gadaleta and M. Rossi, "Idnet: Smartphone-Based Gait Recognition with Convolutional Neural Networks," 2016; https://arxiv.org/abs/1606.03238.
8. Y. Guo, A. Yao, and Y. Chen, "Dynamic Network Surgery for Efficient DNNs," *Proc. 30th Int'l Conf. Neural Information Processing System* (NIPS 16), 2016, pp. 1387–1395.
9. S. Bhattacharya and N.D. Lane "Sparsification and Separation of Deep Learning Layers for Constrained Resource Inference on Wearables," *Proc. 14th ACM Conf. Embedded Network Sensor Systems* (SenSys 16), 2016, pp. 176–189.
10. Y. Gal and Z. Ghahramani, "Dropout

ABOUT THE AUTHORS

SHUOCHAO YAO is a PhD student in the Department of Computer Science at the University of Illinois Urbana-Champaign (UIUC). His research interests include deep learning on Internet of Things (IoT), cyber-physical systems, and crowd and social sensing. Yao received a BS in information engineering from Shanghai Jiao Tong University. Contact him at syao9@illinois.edu.

YIRAN ZHAO is a PhD student in the Department of Computer Science at UIUC. His research interests include cyber-physical systems, machine learning, and IoT applications. Zhao received a BS in information engineering from Shanghai Jiao Tong University. Contact him at zhao97@illinois.edu.

ASTON ZHANG is an applied scientist at Amazon AI. His research focuses is on deep learning. Zhang received a PhD in computer science from UIUC. He previously interned with Google Research, Microsoft Research, Yahoo Labs, UBS, and proprietary trading, and has served in program committees for WWW, KDD, SIGIR, and WSDM. He is a coauthor and coinstructor of the deep learning tutorial with Apache MXNet/Gluon. Contact him at lzhang74@illinois.edu.

SHAOHAN HU is a research staff member at IBM Thomas J. Watson Research Center. His research interests include cyber-physical systems, mobile ubiquitous computing, crowd and social sensing, big data analytics, cloud computing, and quantum computing. Hu received a PhD in computer science from UIUC. Contact him at shaohan.hu@ibm.com.

HUAJIE SHAO is a PhD student in the Department of Computer Science at UIUC. His research interests include data analysis in social networks, applied machine learning, sensor networks, and distributed data centers. Shao received an MS from Zhejiang University. Contact him at hshao5@illinois.edu.

CHAO ZHANG is a PhD student in the Department of Computer Science at UIUC. His research interests include social media analysis, spatiotemporal data mining, text mining, graph mining, and urban computing. Zhang received an MS from Zhejiang University. Contact him at czhang82@illinois.edu.

LU SU is an assistant professor in the Department of Computer Science and Engineering at State University of New York Buffalo. He has also worked at IBM T. J. Watson Research Center and National Center for Supercomputing Applications. Su received a PhD in computer science from UIUC. Su's research interests include the general areas of mobile and crowd sensing systems, Internet of Things, and cyber-physical systems. He is the recipient of an NSF Career Award, University at Buffalo Young Investigator Award, ICCPS 17 Best Paper Award, and the ICDCS 17 Best Student Paper Award. He is a member of ACM and IEEE. Contact him at lusu@buffalo.edu.

TAREK ABDELZAHER is a professor and Willett Faculty Scholar in the Department of Computer Science at UIUC. His research interests include understanding and influencing performance and temporal properties of networked embedded, social and software systems in the face of increasing complexity, distribution, and degree of interaction with an external physical environment. Abdelzaher received a PhD in quality of service adaptation in real-time systems from the University of Michigan. He has authored or coauthored more than 200 refereed publications in real-time computing, distributed systems, sensor networks, and control. He is a member of IEEE and ACM. Contact him at zaher@illinois.edu.

as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," *Proc. 33rd Int'l Conf. Machine Learning* (ICML 16), 2016, pp. 1050–1059.

11. B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles," 2016; https://arxiv.org/abs/1612.01474.
12. I. Goodfellow et al., "Generative Adversarial Nets," 2014; https://arxiv.org/abs/1406.2661.