SenseGAN: Enabling Deep Learning for Internet of Things with a Semi-Supervised Framework

SHUOCHAO YAO, University of Illinois Urbana Champaign
YIRAN ZHAO, University of Illinois Urbana Champaign
HUAJIE SHAO, University of Illinois Urbana Champaign
CHAO ZHANG, University of Illinois Urbana Champaign
ASTON ZHANG, Amazon AI
SHAOHAN HU, IBM Research
DONGXIN LIU, University of Illinois Urbana Champaign
SHENGZHONG LIU, University of Illinois Urbana Champaign
LU SU, State University of New York at Buffalo
TAREK ABDELZAHER, University of Illinois Urbana Champaign

Recent proliferation of Internet of Things (IoT) devices with enhanced computing and sensing capabilities has revolutionized our everyday life. The massive data from these ubiquitous devices motivate the creation of intelligent IoT systems that can collectively learn. However, labelling data for learning purposes is extremely time-consuming, which greatly hinders deployment. In this paper, we describe a semi-supervised deep learning framework, called SenseGAN, that can leverage abundant unlabelled sensing data thereby minimizing the need for labelling effort. SenseGAN jointly trains three components with an adversarial game: (i) a classifier for predicting labels of input sensing data; (ii) a generator for generating sensing data samples based on the input labels; and (iii) a discriminator for differentiating the joint data/label distribution between real samples and partially generated samples from either the classifier or the generator. The classifier and the generator try to generate fake data/labels that can fool the discriminator. The adversarial game among the three components can mutually boost their performance, which helps the classifier learn to predict correct labels with unlabelled data in return. SenseGAN can effectively handle multimodal sensing inputs and easily stabilize the adversarial training process, which helps improve the performance of the classifier. Experiments on three IoT applications demonstrate the substantial improvements in accuracy and F1 score under SenseGAN, compared with supervised counterparts trained only on the labelled portion of the data, as well as other supervised and semi-supervised baselines. For these three applications, SenseGAN requires only 10% of the originally labelled data, to attain nearly the same accuracy as a deep learning classifier trained on the fully labelled dataset.

CCS Concepts: • **Human-centered computing → Ubiquitous and mobile computing theory, concepts and paradigms**; • **Computing methodologies → Machine learning**;

Additional Key Words and Phrases: Internet-of-Things, Deep Learning, Mobile Computing, GAN, Semi-Supervised Learning

Authors' addresses: Shuochao Yao, University of Illinois Urbana Champaign, syao9@illinois.edu; Yiran Zhao, University of Illinois Urbana Champaign; Huajie Shao, University of Illinois Urbana Champaign; Chao Zhang, University of Illinois Urbana Champaign; Aston Zhang, Amazon AI; Shaohan Hu, IBM Research; Dongxin Liu, University of Illinois Urbana Champaign; Shengzhong Liu, University of Illinois Urbana Champaign; Lu Su, State University of New York at Buffalo; Tarek Abdelzaher, University of Illinois Urbana Champaign.

2474-9567/2018/9-ART144 $15.00
https://doi.org/10.1145/3264954

ACM Reference Format:
Shuochao Yao, Yiran Zhao, Huajie Shao, Chao Zhang, Aston Zhang, Shaohan Hu, Dongxin Liu, Shengzhong Liu, Lu Su, and Tarek Abdelzaher. 2018. SenseGAN: Enabling Deep Learning for Internet of Things with a Semi-Supervised Framework. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 3, Article 144 (September 2018), 21 pages. https://doi.org/10.1145/3264954

1 INTRODUCTION

The increasing number of interconnected smart "things" heralds the era of pervasive sensing and ubiquitous computing. Everyday sensing devices have spawned numerous intelligent applications that take advantage of the resulting large volumes of data. Great progress has been made on exploiting such data, including for health and wellbeing [3, 5, 12, 16, 24, 36, 39], behavior and activity recognition [6, 7, 30, 31, 33, 42, 49], tracking and localization [8, 17, 23, 25, 40, 52], crowd and social sensing [43, 44, 50, 51].

At the same time, deep neural networks have greatly changed the way people process human-centric content, such as images, audio, and text [28]. Recently, deep learning architectures have been proposed for IoT devices that perform various complex sensing and recognition tasks [47]. Researchers have made illuminating studies on designing specific structures for IoT tasks [27, 45], applying deep neural networks on diverse sensory modalities [34, 45], providing deep neural networks with well-calibrated uncertainty measurement [46], and compressing the neural network models or structures to fit low-end IoT devices [20, 26, 48].

One key link missing from the picture, however, is the ability to effectively take advantage of massive *unlabelled* sensory data, to reduce the need for learning from labelled samples. Labelling data is always time-consuming. This laborious process has become one key factor that hinders researchers and engineers from applying neural networks to sensing and recognition tasks on IoT devices. IoT applications with a large amount of sensing data therefore call for a *semi*-supervised deep learning framework to solve the challenge of limited labelled data.

In attacking this problem, we propose SenseGAN, a semi-supervised deep learning framework for IoT applications. One core feature of SenseGAN is its capability to leverage unlabelled data for training deep learning networks. SenseGAN can run on resource-constrained IoT devices without additional time or energy consumption compared with its supervised counterpart after training on workstations.

SenseGAN is built upon the idea of Generative Adversarial Networks (GANs) [14, 32], a powerful technique that casts generative modelling as a game between two networks: a generator that produces synthetic data given some noise sources and a discriminator who discriminates the probabilistic distributions between the synthetic and true data [14]. A simple extension of GANs can be used as semi-supervised learning by forcing the discriminator network to output class labels [32].

However, there are three major challenges in applying existing GANs [2, 10, 11, 18, 29, 32] directly as semi-supervised frameworks for diverse IoT applications.

(1) The vanilla extensions of GANs regard the discriminator network as both discriminator and classifier [2, 10, 18, 32], which inevitably requires the discriminator network to possess higher network complexity compared with a normal deep learning classifier. This higher complexity often limits its suitability for resource-constrained IoT devices due to the accompanying high energy consumption and execution time requirements.
(2) The original GANs suffer from training instability [10, 11, 29, 32]. For training a GAN framework with a specifically designed neural network for an arbitrary IoT application, consistently stable training is an important and challenging factor. Noise caused by training instability can mislead the classifier and hurt the final predictive accuracy of the resulting system.
(3) The discriminator and generator in vanilla GANs are not specifically designed for multimodal sensing data [2, 10, 11, 18, 29, 32]. They cannot effectively handle multiple sensing inputs from IoT applications and will cause performance degradation of the classifier.

Therefore, inspired by recent advances of GANs and related studies on deep neural networks [2, 11, 18, 22, 29], we design a novel semi-supervised learning framework, SenseGAN, that directly allows an existing deep learning classifier for an IoT application to leverage unlabelled sensing data. No additional time or energy consumption is required after the training process, compared with the corresponding supervised learning case.

Specifically, we adopt the idea of enabling a discriminator to differentiate the joint data/label distributions between the real data/label samples and the partially generated data/label samples made by either the generator or the classifier. Such design can easily decouple the functionalities of discriminator and classifier into two separate neural networks. For an IoT application, users can design their own neural network structure for classification and replace the classifier in the SenseGAN framework with users' own design for the purpose of semi-supervised learning. The adversarial game among the discriminator, generator, and classifier mutually enhances the performance of all automatically.

In order to stabilize the semi-supervised learning process of SenseGAN, on one hand, we use the Wasserstein metric, also called Earth mover's distance, as the objective function of the discriminator instead of the Jensen-Shannon divergence that may cause numerical instability [2]. On the other hand, we use the Gumbel-Softmax function for categorical representations to eliminate the non-differentiable problem of traditional categorical variables [22]. These two changes greatly improve training stability as well as the final predictive performance. In addition, we also design the specific neural network structures for the discriminator and generator that are suitable for multimodal sensor inputs from IoT applications.

We demonstrate the effectiveness and the efficiency of SenseGAN using the following three representative problems in IoT applications, which illustrate the potential for solving different tasks with the same neural-network based semi-supervised learning framework:

(1) *Heterogeneous human activity recognition:* Human activity recognition itself is a mature problem. Allan *et al.* illustrated that state-of-the-art algorithms do not generalize well across users when a new user is tested who has not appeared in the training set [35]. In this paper, we further test under a more challenging scenario, where most training data is unlabelled.
(2) *User identification with biometric motion analysis:* We use the biometric motion, such as walking, biking, and climbing stairs, for user identification. Similarly, most training data is unlabelled.
(3) *Wi-Fi signal based gesture recognition:* Recently, radio frequency signals have been explored as another emerging resource for sensing on IoT devices. This task uses the Received Signal Strength Indicator (RSSI) of Wi-Fi signals to recognize hand gestures.

Experiments are conducted on the above three IoT tasks with different proportions of labelled and unlabelled data. SenseGAN attains significant gains compared to all baselines (supervised and semi-supervised, deep-learning or non-deep-learning). SenseGAN requires only $30 \sim 200$ samples per category (*i.e.*, 10% of the labelled data) to nearly match the predictive performance of its supervised counterpart trained on a fully labelled dataset. We also conduct experiments on IoT tasks with extremely limited data. SenseGAN can leverage limited data as both labelled and unlabelled data in order to boost the predictive performance.

We validate the main design choices of SenseGAN with ablation study, eliminating individual design components from the framework to create baselines and showing the impact of removing each component. Experiments show that design components for stabilizing training and handling multimodal sensing inputs jointly enhance SenseGAN to better leverage unlabelled data for training. Energy and time consumption are also measured on an IoT computing platform, called Edison [1]. SenseGAN does not need a longer execution time or higher energy consumption compared with its supervised counterpart.

The rest of this paper is organized as follows. Section 2 introduces some preliminary background and related work. We describe the technical details of SenseGAN in Section 3. The evaluation is conducted in Section 4. Finally, we conclude in Section 5.

2 PRELIMINARIES

In this section, we give a preliminary overview of GANs [14], GANs with the Wasserstein metric [2, 18], the Gumbel-Softmax function for categorical representations [22], and deep learning classification models for IoT applications, all of which serve as key design ingredients of our SenseGAN framework.

2.1 Generative Adversarial Networks (GANs)

The idea of GANs is to design a game between two competing networks. The generator network takes the noise vectors as inputs and generates data samples. The discriminator network takes either a generated sample or a real data sample, and distinguishes between the two. The generator is trained to fool the discriminator [14].

The game between generator G and discriminator D can be formulated as the minmax objective:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}[\log(D(\mathbf{x}))] + \mathbb{E}_{\tilde{\mathbf{x}}\sim\mathbb{P}_g}[\log(1 - D(\tilde{\mathbf{x}}))], \tag{1}$$

where $\mathbb{P}_r$ is the real data distribution and $\mathbb{P}_g$ the generated data distribution implicitly defined by $\tilde{\mathbf{x}} = G(\mathbf{z})$, $\mathbf{z} \sim p(\mathbf{z})$ (the input of generator $\mathbf{z}$ is sampled from a simple noise distribution, such as the uniform distribution or a spherical Gaussian distribution).

If the discriminator is optimal, the training objective function (1) amounts to minimizing the Jensen-Shannon (JS) divergence between $\mathbb{P}_r$ and $\mathbb{P}_g$. In practice, a stochastic lower-bound to the JS divergence is minimized.

2.2 Wasserstein GANs

Since training instability has hindered the deployment of GANs on deeper and more complex neural network structures, a great amount of research efforts have been made recently to tackle this problem. Arjovsky et al. [2] argue that Jensen-Shannon divergence are potentially not continuous and thus cannot provide a usable gradient for the generator. They propose Earth mover's distance (also called Wasserstein-1 Distance) as the training objective. Earth mover's distance is the minimum cost of transporting mass in order to transform one distribution into the other distribution, where the cost is mass times transport distance. They have shown that the Earth mover's distance is continuous everywhere, and differentiable almost everywhere, providing the desirable property for GANs training.

Wasserstein GAN (WGAN) is thus proposed, and its objective function is constructed by applying the Kantorovich-Rubinstein duality [38]

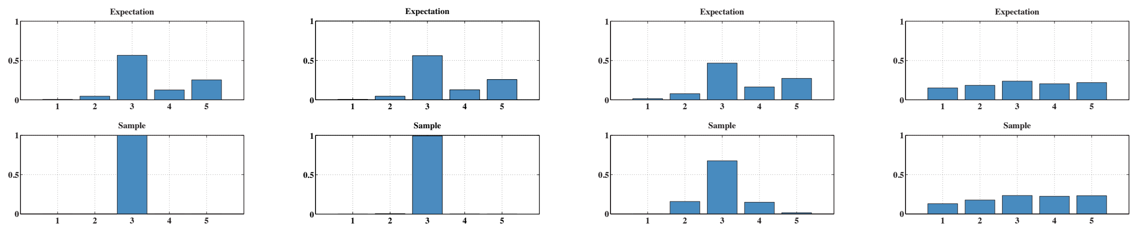
$$\min_G \max_{D\in\mathcal{D}} \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}[D(\mathbf{x})] - \mathbb{E}_{\tilde{\mathbf{x}}\sim\mathbb{P}_g}[D(\tilde{\mathbf{x}})], \tag{2}$$

where $\mathcal{D}$ is the set of 1-Lipschitz functions, $\mathbb{P}_r$ and $\mathbb{P}_g$ are defined the same as they are in Section 2.1. Then given the optimal discriminator, WGAN's training objective function (2) amounts to minimizing the Earth mover's distance between $\mathbb{P}_r$ and $\mathbb{P}_g$.

Then the remaining question is how to enforce the Lipschitz constraint on the discriminator. Arjovsky et al. [2] make the Lipschitz constraint by clipping the weights within a compact space $[-c, c]$. This results in a subset of k-Lipschitz functions for k depending on the space boundary parameter c and the structure of the discriminator.

However Gulrajani et al. [18] claim that weight clipping can still cause optimization difficulties such as capacity underuse and gradients exploding or vanishing. Then an alternative is proposed: a differentiable function is 1-Lipschitz if and only if it has gradients with norm less than or equal to 1 everywhere. Since an exact constraint is not easily tractable, Gulrajani et al. enforce a soft version by making gradient penalty on sampled points. The loss function for the discriminator then becomes:

$$\mathbb{E}_{\tilde{\mathbf{x}}\sim\mathbb{P}_g}[D(\tilde{\mathbf{x}})] - \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}[D(\mathbf{x})] + \lambda \cdot \mathbb{E}_{\hat{\mathbf{x}}\sim\mathbb{P}_{\hat{\mathbf{x}}}}\left[(\|\nabla D(\hat{\mathbf{x}})\|_2 - 1)^2\right], \tag{3}$$



(a) Categorical distribution. (b) Gumbel-Softmax distribution with $\tau = 0.1$. (c) Gumbel-Softmax distribution with $\tau = 1$. (d) Gumbel-Softmax distribution with $\tau = 10$.

Fig. 1. The expectation and sample of Categorical and Gumbel-Softmax distribution.

where gradient penalties are made by taking samples $\hat{\mathbf{x}}$ from straight lines between points in the data distribution $\mathbb{P}_r$ and the generator distribution $\mathbb{P}_g$:

$$\begin{gathered} \epsilon \sim U[0,1], x \sim \mathbb{P}_r, \tilde{x} \sim \mathbb{P}_g, \\ \hat{\mathbf{x}} = \epsilon x + (1-\epsilon)\tilde{x}, \end{gathered} \tag{4}$$

where $U[0,1]$ is the uniform distribution from 0 to 1.

The WGAN with gradient penalty has achieved the state-of-the-art performance on multiple generative tasks, such as images and text generations. To the best of our knowledge, SenseGAN is the first study to adopt WGAN with gradient penalty training strategy into the semi-supervised learning.

2.3 Gumbel Softmax

Discrete variables sampled from categorical distribution is a powerful technique for representing categorical distributions. In our framework, the discriminator in SenseGAN is designed to differentiate the joint data/label distributions between partially generated samples and real samples, which naturally requires taking discrete variables for categorical representation as inputs.

However, neural networks with discrete variables involving sampling from categorical distributions are non-differentiable, making them difficult to train with the backpropagation algorithm. Existing stochastic gradient estimation requires variance reduction techniques to stabilize the training process. In order to alleviate the problem of high-variance gradient estimation in neural networks with discrete variables, a continuous relaxation of the discrete variable is needed.

Recent study on Gumbel-Softmax defines a continuous distribution over the simplex that can approximate samples from a categorical distribution [22], where the theoretical analysis has been made.

We assume categorical samples are encoded as l-dimensional one-hot vectors lying on the corners of the $(l-1)$-dimensional simplex, Δ^{l-1}. $o_1, \cdots, o_l$ are multinomial logits for l categories. We can therefore generate l-dimensional sample vectors y:

$$y_i = \frac{\exp((o_i + g_i)/\tau)}{\sum_{j=1}^{l} \exp((o_j + g_j)/\tau)}, \tag{5}$$

where $g_1, \cdots, g_l$ are i.i.d. samples drawn from Gumbel(0, 1). The Gumbel(0, 1) distribution can be sampled using inverse transform sampling by drawing $u \sim U[0,1]$ and computing $g = -\log(-\log(u))$.

As the softmax temperature τ approaches 0, samples from the Gumbel-Softmax distribution turns into one-hot representations, and its expectation approaches the corresponding categorical distribution from the uniform distribution. The expectation and sample of Gumbel-Softmax distribution with different softmax temperature τ are illustrated in Figure 1. In SenseGAN, the multinomial logits of the classifier go through the Gumbel-Softmax

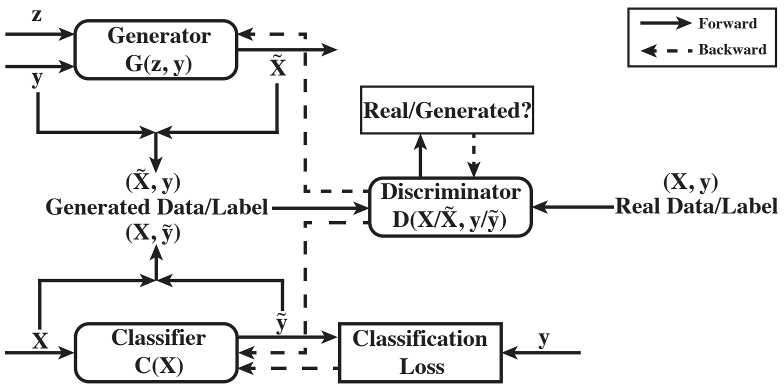


Fig. 2. The illustration of SenseGAN components.

before being fed into the discriminator, which further improves the training stability and the final predictive performance.

2.4 Deep Learning for IoT Applications

Recently, deep learning is emerging as a powerful learning component in IoT applications [47]. These highly capable models are good at making sophisticated mappings between unstructured data such as sensor inputs and target quantities, which can hardly be achieved by traditional machine learning models. Lane et al. [27] build a multilayer perceptron improving the performance of multiple audio sensing tasks. Yao et al. [45] design a unified deep learning framework for IoT tasks that can fuse multiple sensory modalities and extract temporal relationships along sensor inputs. However, to the best of our knowledge, all the previous works focus on the supervised learning scenario, which fails to utilize the abundant unlabelled data in IoT applications.

3 SENSEGAN FRAMEWORK

In this section, we introduce our SenseGAN semi-supervised learning framework for IoT applications. We separate our descriptions into five parts. First, we provide an overview of the SenseGAN framework with its internal interactions among different components. Next, we introduce our designs of generator, discriminator, and classifier in SenseGAN respectively. In the end, we discuss SenseGAN's training process.

3.1 SenseGAN Components Overview

The SenseGAN framework consists of three basic components. We assume the following notations: $\mathbf{X}$ denotes the input sensing data tensor, $\mathbf{y}$ the one-hot categorical representation, and $\mathbf{z}$ the random vector drawn from the random Gaussian distribution. The relationship among these three components is illustrated in Figure 2.

(1) **Generator** $\tilde{\mathbf{X}} \sim G(\mathbf{z}, \mathbf{y})$: The generator takes a random vector and a corresponding one-hot categorical representation as the input, and generates a sensing data tensor $\tilde{\mathbf{X}}$ that "fools" the discriminator into thinking that it is the real sensing data.
(2) **Classifier** $\tilde{\mathbf{y}} \sim C(\mathbf{X})$: The classifier takes a sensing data tensor as the input and generates classification result $\tilde{\mathbf{y}}$ that can "fool" the discriminator into thinking that it is the real data label. If the sensing data tensor happens to be from the limited amount of labelled data, the classification result $\tilde{\mathbf{y}}$ should also fit the supervision of label $\mathbf{y}$.
(3) **Discriminator** $D(\mathbf{X}/\tilde{\mathbf{X}}, \mathbf{y}/\tilde{\mathbf{y}})$: The discriminator takes a pair of sensing data and corresponding one-hot categorical representation as the input. It gives each input pair a score for indicating whether the pair is sampled from the real labelled dataset or partially generated by other components.

The intuition of how our SenseGAN framework is able to leverage unlabelled data to enhance its predictive power is as follows. The discriminator tries to discriminate real data/label samples and the partially generated data/label samples; the generator attempts to generate and recover the real sensing inputs based on the categorical information that can fool the discriminator; and the classifier tries to predict the label of sensing inputs that can both fit the supervision and fool the discriminator. During the adversarial game among the three components under the training process, the resulting three improved components can mutually boost performance. When the training reaches the optimality, the discriminator will have learnt the true joint probability distribution of the input sensing data and their corresponding labels for both the labelled and unlabelled samples. The classifier will have learnt the true conditional probability of labels given the sensing input.

All three components are represented by neural networks. We will discuss their specific structures for dealing with the multimodal sensing inputs in detail in the following subsections. In addition, the structure of the classifier can be task-specific or unified with diverse IoT applications [45]. We therefore will not introduce the detailed structure for the classifier but only discuss its output representation. We treat the classifier as an modular and customizable component for IoT applications when using SenseGAN for semi-supervised learning.

3.2 SenseGAN Generator

The goal here is to generate sensing data tensor by modelling the conditional probability of sensing data given the one-hot label representation. The randomness of generated samples is controlled by the input random vectors.

We denote the generated sensing data tensor as $\tilde{\mathbf{X}} \in \mathbb{R}^{n \times t \times f \times d}$, where n is the number of sensors used in an IoT application, t is the number of time steps, f is the length of frequency domain or the number of time points within a time step, and d is the feature dimension of data on each frequency or time point. We denote the one-hot label representation as $\mathbf{y} \in \mathbb{R}^{l}$, where l is the number of categories of an IoT application. We also denote the input random vector as $\mathbf{z} \in \mathbb{R}^{r}$, where r is the dimension of random vector.

The illustration of SenseGAN generator $G(\mathbf{z}, \mathbf{y})$ is shown in Figure 3. The input of generator $\mathbf{T}_0$ is the concatenated vector of the one-hot label representation $\mathbf{y}$ and the random vector $\mathbf{z}$. $\mathbf{T}_0$ first goes through a fully-connected layer for learning a latent representation $\mathbf{T}_1$, which is then transformed into a matrix form $\mathbf{T}_2$.

Then the generator starts the process of generating sensing data tensor as $\tilde{\mathbf{X}} \in \mathbb{R}^{n \times t \times f \times d}$. We assume that n sensing data possess a joint latent representation, and the generation process is first going through generating along the time/frequency dimensions, t and f, with a shared structure and then going through generating the feature dimension d with individual structures for n sensors.

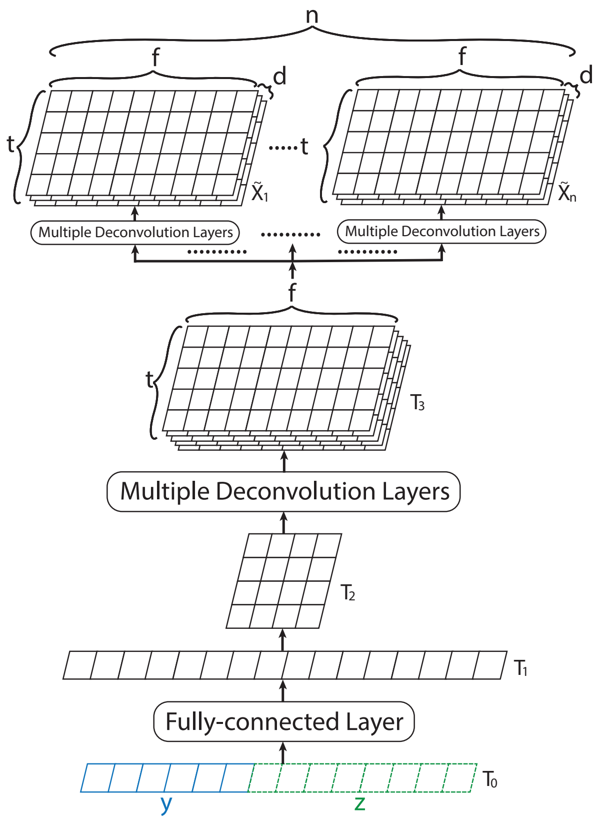
As shown in Figure 3, $\mathbf{T}_2$ goes through multiple deconvolution layers and generates a joint latent representation along the time and frequency dimensions $\mathbf{T}_3 \in \mathbb{R}^{t \times f \times c}$, where c is the number of filters in the deconvolution layer. Then the joint representation $\mathbf{T}_3$ goes through n independent multiple deconvolution layers for generating the sensing data tensor for n sensors, $\tilde{\mathbf{X}} = [\tilde{\mathbf{X}}_1, \cdots, \tilde{\mathbf{X}}_n]$.

The whole generator structure tries to learn the conditional distribution of sensing data given the label representation $P(\mathbf{X}|\mathbf{y})$. It also helps the discriminator to learn the joint data/label distribution $P(\mathbf{X}, \mathbf{y})$ and the classifier to learn the conditional distribution of label given the sensing data $P(\mathbf{y}|\mathbf{X})$ during the adversarial game by leveraging unlabelled data.

3.3 SenseGAN Discriminator

The SenseGAN discriminator aims to differentiate partially generated data/label tuples from the real ones by modelling the joint data/label distribution.

We again denote the sensing data as $\mathbf{X} \in \mathbb{R}^{n \times t \times f \times d}$, and the one-hot label representation as $\mathbf{y} \in \mathbb{R}^{l}$. The illustration of SenseGAN discriminator $D(\mathbf{X}, \mathbf{y})$ with two sensor inputs is shown in Figure 4. The structure for n sensor inputs can be similarly designed. We first need to generate input $\hat{\mathbf{X}}$ containing both data and label information. We simply expand label representation $\mathbf{y}$ into $\mathbf{Y} \in \mathbb{R}^{n \times t \times f \times l}$ with the tiling operation, and then



n
f
d
t
$\tilde{X}_1$
$\tilde{X}_n$
Multiple Deconvolution Layers
Multiple Deconvolution Layers
f
t
T_3
Multiple Deconvolution Layers
T_2
T_1
Fully-connected Layer
T_0
y
z

Fig. 3. The neural network design of SenseGAN generator.

concatenate $\mathbf{X}$ and $\mathbf{Y}$ into the generator input $\hat{\mathbf{X}} \in \mathbb{R}^{n \times t \times f \times (d+l)}$ as shown in Figure 4. These expansion and concatenation operations can help to merge the data and label information and to balance gradient penalty (3) between data and label inputs.

The design of discriminator is to first merge the information from multiple sensors and then extract important relationships along the time and frequency dimensions. Since convolution layers can learn about the long-term dependency by stacking multiple of them [9], SenseGAN uses full-convolution structure (with fully-connected layers at the end) instead of the recurrent neural network for speeding up the training process. The empirical comparison betweem CNN-based and RNN-based discriminator is shown in Section 4.3.

As shown in Figure 4, SenseGAN discriminator first learns a joint representation $\mathbf{T}_1$ from multiple sensors through a convolution layer. Then multiple convolution layers are used to learn the latent representation $\mathbf{T}_2$ along the time and frequency dimension. At last, the discriminator generates the score v through multiple fully-connected layers for determining whether the input $\hat{\mathbf{X}}$ is drawn from real data samples or partially generated by other components.

The discriminator tries to differentiate whether the input is drawn from real samples or is partially generated by learning the joint data and label distribution $p(\mathbf{X}, \mathbf{y})$. The discriminator structure in Figure 4 tries to fit the multimodal sensing inputs that exists in IoT applications better.

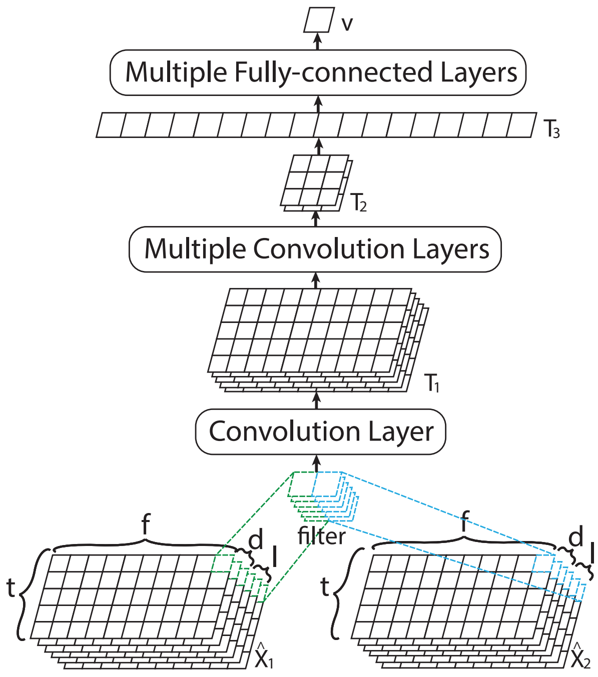


Fig. 4. The neural network design of SenseGAN discriminator with two sensor inputs.

3.4 SenseGAN Classifier

As mentioned previously, the SenseGAN classifier, by design, serves as an application-dependent customizable module, capable of taking the form of any existing neural network classifiers. The classifier can be formulated as $\mathbf{o} = C(\mathbf{X})$, where $\mathbf{X} \in \mathbb{R}^{n \times t \times f \times d}$ is the sensing data input, and $\mathbf{o} \in \mathbb{R}^{l}$ is the multinomial logits for l categories.

However as shown in Figure 2, two other components take categorical representations instead of multinomial logits as inputs. When the sensing data inputs $\mathbf{X}$ have no existing label, the classifier feeds the predictions $\tilde{\mathbf{y}}$ as well as the sensing data $\mathbf{X}$ into the discriminator. On the other hand, if the sensing data inputs $\mathbf{X}$ have existing labels, the classifier feeds the predictions $\tilde{\mathbf{y}}$ into classification loss with true label $\mathbf{y}$ for supervision. We denote these two predictions as $\tilde{\mathbf{y}}_d$ and $\tilde{\mathbf{y}}_l$ for discriminator input and classification loss input respectively.

Softmax function is a common practice for converting multinomial logits into categorical representations, which is also our choice for $\tilde{\mathbf{y}}_l$. However for $\tilde{\mathbf{y}}_d$, in order to prevent discriminator from differentiating real or generated samples by just looking at whether the categorical representations are one-hot representations or categorical distribution, we choose Gumbel-Softmax (5) for converting multinomial logits into categorical representations, which is introduced in Section 2.3. Compared with sampling from categorical distribution, Gumbel-Softmax can eliminate non-differentiable operations and prevent using gradient estimation method with high variance.

3.5 SenseGAN Training

We now discuss the SenseGAN training that involves the aforementioned three components. We denote $\{(\mathbf{X}_l, \mathbf{y}_l)\}$ as the labelled dataset and $\{(\mathbf{X}_u)\}$ as the unlabelled dataset.

The SenseGAN objective function consists of two parts. The first part captures the classifier's intent to minimize the classification loss of labelled data, which can be formulated as:

$$\min_{C} \frac{1}{m_l} \sum_{(\mathbf{X}_l, \mathbf{y}_l)} \ell\Big(\mathbf{y}, C(\mathbf{X})\Big) \tag{6}$$

where $\ell(\cdot, \cdot)$ denotes the cross entropy loss and $C(\cdot)$ the SenseGAN classifier. The second part is the adversarial game among the discriminator, the generator, and the classifier. The discriminator tries to distinguish positive data/label tuples of "real" labelled dataset from negative data/label tuples that are partially generated by the generator or classifier. At the same time, the generator and classifier try to generate data/label tuples that can fool the discriminator. Here we provide the formal definition of negative and positive data/label tuples during the training process.

There are two types of negative data/label tuples. The first type is $(\mathbf{X}_u^{(1)}, C(\mathbf{X}_u^{(1)}))$, where we sample sensing data from the unlabelled dataset and generate one-hot label with the SenseGAN classifier. The second type is $(G(\mathbf{z}, \mathbf{y}_g), \mathbf{y}_g)$, where we randomly generate one-hot label from all possible categories and generate sensing data with the SenseGAN generator.

The positive data/label tuples are $(\mathbf{X}_l, \mathbf{y}_l)$, where we directly sample data/label tuples from the labelled dataset. In order to prevent the discriminator from overfitting the limited labelled dataset by memorizing all of them, we introduce the pseudo positive data/label tuples, $(\mathbf{X}_u^{(2)}, C(\mathbf{X}_u^{(2)}))$ when training the discriminator in practice. The detailed definitions of these negative and positive data/label tuples are summarized in Table 1.

With these definitions, we can formulate the second part of the objective function. We apply the Earth mover's distance to formulate the adversarial game, which is similar to the Wasserstein GAN (2).

$$\begin{gathered}
\max_{D \in \mathcal{D}} \frac{1}{m_l + m_u^{(2)}} \Bigg(\sum_{\mathbf{X}_l, \mathbf{y}_l} D(\mathbf{X}_l, \mathbf{y}_l) + \sum_{\mathbf{X}_u^{(2)}} D\Big(\mathbf{X}_u^{(2)}, C\big(\mathbf{X}_u^{(2)}\big)\Big) \Bigg) - \frac{\alpha}{m_g} \sum_{\mathbf{y}_g} D\Big(G(\mathbf{z}, \mathbf{y}_g), \mathbf{y}_g\Big) - \frac{1-\alpha}{m_u^{(1)}} \sum_{\mathbf{X}_u^{(1)}} D\Big(\mathbf{X}_u^{(1)}, C\big(\mathbf{X}_u^{(1)}\big)\Big), \\
\min_{G, C} - \frac{\alpha}{m_g} \sum_{\mathbf{y}_g} D\Big(G(\mathbf{z}, \mathbf{y}_g), \mathbf{y}_g\Big) - \frac{1-\alpha}{m_u^{(1)}} \sum_{\mathbf{X}_u^{(1)}} D\Big(\mathbf{X}_u^{(1)}, C\big(\mathbf{X}_u^{(1)}\big)\Big),
\end{gathered} \tag{7}$$

where α is a hyper-parameter for balancing two types of negative tuples with default value 0.5; $\mathcal{D}$ is the set of 1-Lipschitz functions; m_l, $m_u^{(2)}$, $m_u^{(1)}$, and m_g are the batch sizes of positive, pseudo-positive, and two negative tuples during training. In SenseGAN, we enforce the Lipschitz constraint by making gradient penalty (3) with λ.

The pseudo positive tuples $(\mathbf{X}_u^{(2)}, C(\mathbf{X}_u^{(2)}))$ also work as a variance reduction method for the negative tuples $(\mathbf{X}_u^{(1)}, C(\mathbf{X}_u^{(1)}))$, which further stabilize the training process [15]. In general, the pseudo positive tuple introduces a biased distribution into the discriminator. However, the bias is controlled by the batch sizes of positive and pseudo positive tuples, m_l and $m_u^{(2)}$. In addition, the classifier can often achieve a reasonably good prediction result quickly, which further reduces the bias. The empirical evaluation of pseudo positive tuple is shown in Section 4.3.

By combining the two objective functions (6) and (7) with hyper-parameter γ, we can summarize our final training process. As shown in Algorithm 1, the training proceeds by iteratively updating the parameters of the discriminator, classifier, and generator. The update of discriminator (Line 3 - Line 8) consists of three parts: increasing the score of positive tuples, reducing the score of negative tuples, and imposing gradient penalty for enforcing the Lipschitz constraint. The update of classifier (Line 9 - Line 12) consists of two parts: reducing the classification loss of labelled data and learning to make prediction that can obtain high score by the discriminator.

The update of generator (Line 13 - Line 16) has only one part: learning to generate sensing data that can obtain high score by the discriminator so that the discriminator can be fooled to believe that the generated sensing

Table 1. Definitions of positive and negative data/label tuples.

Postive Tuples		Pseudo Postive Tuples	
Data sampled from the labelled dataset	$\mathbf{X}_l$	Data sampled from the unlabelled dataset	$\mathbf{X}_u^{(2)}$
Corresponding real label	$\mathbf{y}_l$	Corresponding label generated by the Classifier	$C(\mathbf{X}_u^{(2)})$
Negative Tuples 1		**Negative Tuples 2**	
Data sampled from the unlabelled dataset	$\mathbf{X}_u^{(1)}$	Corresponding data generated by the Generator	$G(\mathbf{z}, \mathbf{y}_g)$
Corresponding label generated by the Classifier	$C(\mathbf{X}_u^{(1)})$	Randomly generated label	$\mathbf{y}_g$

Algorithm 1 SenseGAN training process

```
1:  Initialize: discriminator with parameter $\theta_D$, generator with parameter $\theta_G$, and classifier with parameter $\theta_C$.
2:  for $k$ training iterations do
3:      for $k_d$ iterations do
4:          Sample $(\mathbf{X}_l, \mathbf{y}_l)$, $(\mathbf{X}_u^{(1)})$, $(\mathbf{X}_u^{(2)})$, $(\mathbf{y}_g)$, $\epsilon \sim U[0,1]$, and $\mathbf{z} \sim \mathcal{N}(0,1)$
5:          $\bar{\mathbf{X}} = \epsilon \cdot \mathbf{X}_l + (1-\epsilon) \cdot G(\mathbf{z}, \mathbf{y}_g)$
6:          $\bar{\mathbf{y}} = Gumbel\_softmax(\epsilon \cdot \mathbf{y}_l + (1-\epsilon) \cdot \mathbf{y}_g)$
7:          Update $\theta_D$ by descending along its gradient $\nabla_{\theta_D}\Big[\frac{\alpha}{m_g}\sum_{\mathbf{y}_g} D\big(G(\mathbf{z},\mathbf{y}_g),\mathbf{y}_g\big) + \frac{1-\alpha}{m_u^{(1)}}\sum_{\mathbf{X}_u^{(1)}} D\Big(\mathbf{X}_u^{(1)}, C\big(\mathbf{X}_u^{(1)}\big)\Big) - \frac{1}{m_l+m_u^{(2)}}\Big(\sum_{\mathbf{X}_l,\mathbf{y}_l} D(\mathbf{X}_l,\mathbf{y}_l) + \sum_{\mathbf{X}_u^{(2)}} D\Big(\mathbf{X}_u^{(2)}, C\big(\mathbf{X}_u^{(2)}\big)\Big)\Big) + \lambda(\|\nabla D(\bar{\mathbf{X}},\bar{\mathbf{y}})\|_2 - 1)^2\Big]$
8:      end for
9:      for $k_c$ iterations do
10:         Sample $(\mathbf{X}_l, \mathbf{y}_l)$ and $(\mathbf{X}_i^{(1)})$
11:         Update $\theta_C$ by descending along its gradient $\nabla_{\theta_C}\Big[\frac{\gamma}{m_l}\sum_{(\mathbf{X}_l,\mathbf{y}_l)} CE\big(\mathbf{y}, C(\mathbf{X})\big) - \frac{1-\alpha}{m_u^{(1)}}\sum_{\mathbf{X}_u^{(1)}} D\Big(\mathbf{X}_u^{(1)}, C\big(\mathbf{X}_u^{(1)}\big)\Big)\Big]$
12:     end for
13:     for $k_g$ iterations do
14:         Sample $(\mathbf{y}_g)$ and $\mathbf{z} \sim \mathcal{N}(0,1)$
15:         Update $\theta_G$ by descending along its gradient $\nabla_{\theta_G}\Big[-\frac{\alpha}{m_g}\sum_{\mathbf{y}_g} D\big(G(\mathbf{z},\mathbf{y}_g),\mathbf{y}_g\big)\Big]$
16:     end for
17: end for
```

data is real. This iterative process can gradually improve the performance of all three components by mutually boosting themselves with limited labelled supervision and abundant unlabelled data.

4 EVALUATION

In this section, we test SenseGAN on three IoT applications with several sets of experiments evaluating the effectiveness, the design choices, and the resource consumption of SenseGAN on commodity IoT devices.

4.1 Experiments Overview

For the evaluation, we conduct all our experiments on three different tasks. We next briefly describe these tasks and introduce the training and testing datasets.

(1) *Heterogeneous human activity recognition (HHAR):* In this task, we perform a human activity recognition task with accelerometer and gyroscope measurements. We use the dataset collected by Allan et al. [35]. This dataset contains readings from two motion sensors (accelerometer and gyroscope). Readings were recorded when users executed activities scripted in no specific order, while carrying smartwatches and smartphones.

The dataset contains 9 users, 6 activities (biking, sitting, standing, walking, climbStairup, and climbStairdown), and 6 types of mobile devices. For this task, accelerometer and gyroscope measurements are classifier inputs, and activities are used as labels.
(2) *User identification with biometric motion analysis (UserID):* In this task, we perform user identification with biometric motion analysis. We classify users' identity according to accelerometer and gyroscope measurements. We use the same dataset as in the HHAR task. For this task, accelerometer and gyroscope measurements are classifier inputs, and users' unique IDs are used as labels.
(3) *Wi-Fi signal based gesture recognition (Wisture):* In this task, we perform gesture recognition (swipe, push, and pull) with Received Signal Strength Indicator (RSSI) of Wi-Fi signal. We use the dataset collected by Mohamed et al. [21]. This dataset contains labeled Wi-Fi RSSI measurements corresponding to three hand gestures made near a smartphone under different spatial and data traffic scenarios. The Wi-Fi RSSI measurements are classifier inputs, and gestures are used as labels.

For all the following experiments, the HHAR task performs leave-one-user-out cross-validation. We select 1 out of 9 users as the testing dataset with the left as the training dataset. The UserID and Wisture tasks perform 10-fold cross-validation. Each time we choose 10% data as the testing dataset with the left as the training dataset. Then we further divide the training dataset into labelled and unlabelled dataset according to the specification of each experiment.

For all experiments with SenseGAN, we choose DeepSense [45] as the classifier, which is a state-of-the-art neural network structure designed for IoT applications. The generator and the discriminator usually require more training steps to achieve better performance. We therefore set k_d, k_c, and k_g in Algorithm 1 to be 5, 1, and 7 respectively without fine-tuning, which consistently achieves decent performance for all three tasks.

4.2 Effectiveness

We first illustrate the effectiveness of SenseGAN at leveraging unlabelled data for IoT applications. We evaluate the performance of SenseGAN with different proportions of labelled and unlabelled data, and compare SenseGAN with supervised deep learning algorithms and other traditional supervised/semi-supervised machine learning algorithms, which are believed to have better predictive performance with smaller datasets. The baseline algorithms are as follows:

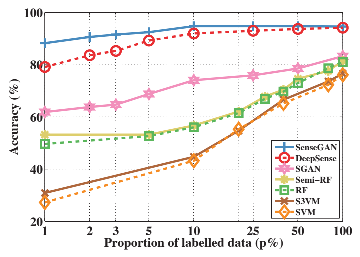
(1) *DeepSense:* A state-of-the-art supervised neural network structure designed for IoT applications [45], also the classifier used in SenseGAN. DeepSense baseline is chosen to show the performance gain of SenseGAN by leveraging unlabelled data. DeepSense takes the same input as the SenseGAN model, which divides the sensing data into equal-length time interval followed by Fourier transformation.
(2) *SGAN:* A general GAN-based semi-supervised deep learning algorithm [32]. The algorithm is not specifically designed for adapting the IoT sensing data. This baseline is chosen to show the importance of design choices in SenseGAN that tailor the GAN training method for IoT applications. SGAN also takes the same input as the SenseGAN model.
(3) *RF:* The random forests supervised classifier. The input of random forests is the concatenation of popular time-domain and frequency domain features from [13] and ECDF features [19].
(4) *SVM:* The support vector machine supervised classifier, with the same feature selection as the RF model.
(5) *Semi-RF:* A semi-supervised learning algorithm that puts a self-training wrapper on the random forest classifier [37]. Semi-RF takes the same input as the RF model.
(6) *S3VM:* Semi-supervised SVMs (S3VMs) with the goal of maximizing the margin of unlabelled data for classification [4]. S3VM takes the same input as the SVM model.

4.2.1 *IoT Applications with Different Proportions of Labelled Data.*

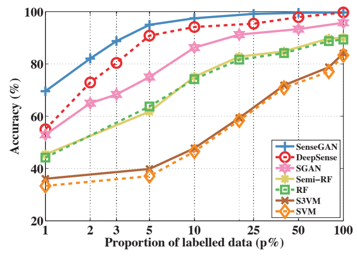
We first conduct a series of experiments on evaluating SenseGAN against baseline algorithms with different proportions of labelled data. The original datasets for tasks are all labelled data samples. During these experiments, we randomly select $p\%$ of training samples as labelled data, and treat the rest training samples as unlabelled data. When $p = 100\%$, SenseGAN stops generating “fake" data. SenseGAN then becomes equivalent to its supervised counterpart, DeepSense. For semi-supervised algorithms, SenseGAN, Semi-RF, and S3VM, all unlabelled data is used for training. For supervised algorithms, DeepSense, RF, and SVM, only labelled data is used for training. Please notice that the testing data never appears in the unlabelled dataset.

The results of three IoT tasks are shown in Figure 5 and 6. Two figures are plotted in log scale along the p−axis, because we are more interested in the cases with limited labelled data. The performance gains between semi-supervised and supervised algorithms can be small when the proportion of labelled data p is large.

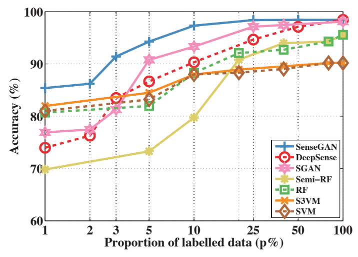
One interesting observation is that two deep learning methods, SenseGAN and DeepSense, perform almost consistently better than the traditional machine learning methods even with tiny proportion of labelled data (with the only exception of DeepSense on the Wisture task). This is mainly attributed to their structures that can effectively handle the multimodal sensing data in three IoT applications. Performance can be further improved by our proposed SenseGAN framework. As shown in Figure 5 and 6, SenseGAN shows a significant improvement on both accuracy and F1 score compared with all baseline algorithms. SenseGAN can achieve within 2% and 0.03 drops on accuracy and F1 score with only 10% of labelled data. In our experiments, 10% of labelled data equals around 200, 130, and 30 labelled data samples per category for the HHAR, UserID, and Wisture tasks respectively, which is easily affordable by human labelling. In addition, the existing GAN-based semi-supervised deep learning method, SGAN, consistently shows an inferior performance compared to SenseGAN in all three tasks. These experimental results empirically show the effectiveness of our SenseGAN design choices in general. More ablation study on verifying the design choices of SenseGAN will be shown in Section 4.3.



(a) The HHAR task.

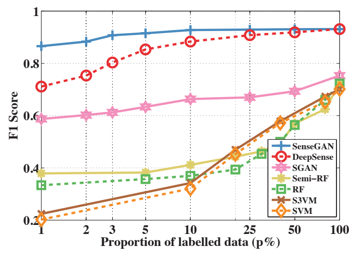


(b) The UserID task.

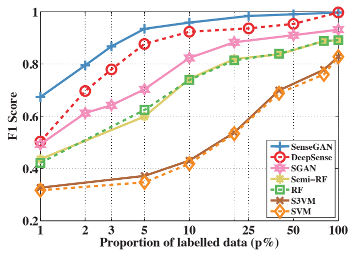


(c) The Wisture task.

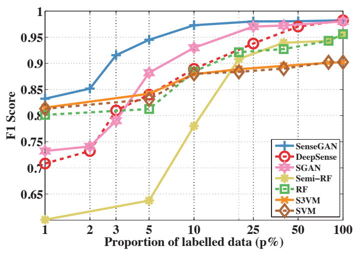
Fig. 5. The accuracy of models with $p\%$ of labelled data for three IoT applications.



(a) The HHAR task.

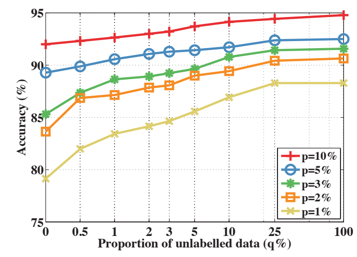


(b) The UserID task.

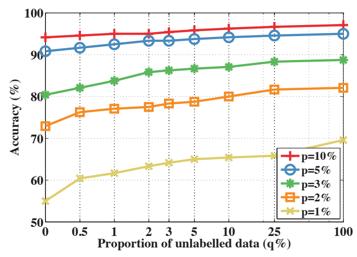


(c) The Wisture task.

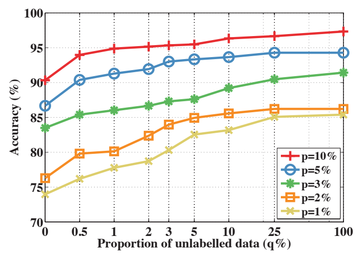
Fig. 6. The F1 Score of models with $p\%$ of labelled data for three IoT applications.



(a) The HHAR task.

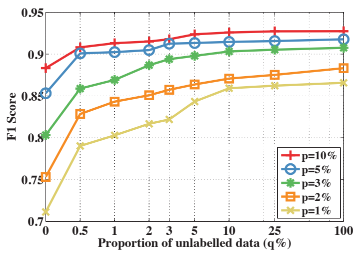


(b) The UserID task.

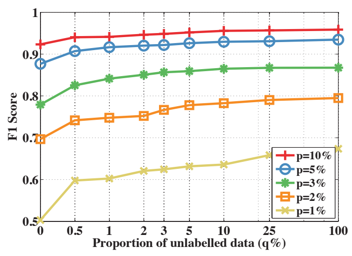


(c) The Wisture task.

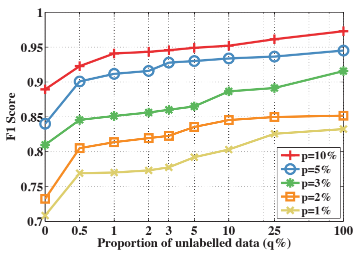
Fig. 7. The accuracy of models with q% of unlabelled data for three IoT applications.



(a) The HHAR task.



(b) The UserID task.



(c) The Wisture task.

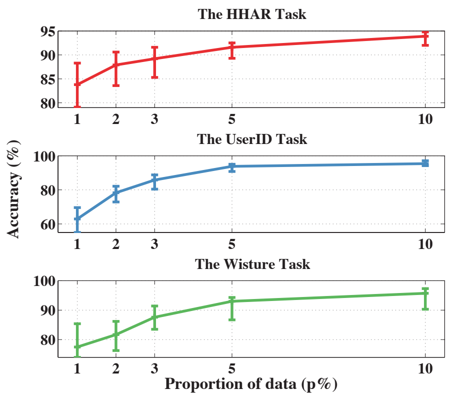
Fig. 8. The F1 Score of models with q% of unlabelled data for three IoT applications.

SenseGAN also attains a great improvement compared with its supervised counterpart, DeepSense. However the other two traditional semi-supervised methods, Semi-RF and S3VM, only achieve tiny improvements compared with their supervised counterparts, RF and SVM. Semi-RF even performs worse than RF on the Wisture task. These observations indicate that SenseGAN can effectively leverage the unlabelled multimodal sensor data for improving the complex IoT recognition tasks.

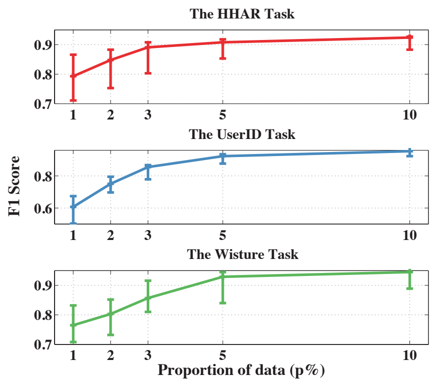
4.2.2 IoT Applications with Different Proportions of Unlabelled Data. We have shown that SenseGAN can utilize labelled data efficiently. It can consistently achieve the best performance with a large margin on three IoT tasks with different proportions of labelled data. However, we have not investigated whether SenseGAN can utilize unlabelled data efficiently. Next, we conduct a series of experiments on performing semi-supervised learning with different proportions of unlabelled data to explore the effect of unlabelled-data sizes on the SenseGAN predictive performance.

For these experiments, we randomly select $p = [1, 2, 3, 5, 10]\%$ of training samples as labelled data, and randomly select $q\%$ of the remaining training samples as unlabelled data, and discard all the rest training samples. We evaluate only SenseGAN here, because Semi-RF and S3VM show little improvement on three IoT tasks even with $q = 100\%$ in Section 4.2.1.

When $q = 0\%$, the model is trained with only labelled data, which is equivalent to the DeepSense model. When $q = 100\%$, all experiments are fully semi-supervised, which are the same as those in Section 4.2.1, because all data samples other than $p\%$ of labelled data are used as unlabelled data for training. As shown in Figure 7 and 8, SenseGAN with $q = 25\%$ can attain almost the same predictive performance compared with $q = 100\%$ on both accuracy and F1 score. Even when the proportion of unlabelled data is tiny, such as $q = 0.5\%$ or 1%, SenseGAN can still achieve decent improvements compared with its supervised counterpart. This indicates that SenseGAN can efficiently use both labelled and unlabelled data to effectively improve the performance of the neural network classifier. Due to the scale of y-axis in Figure 7 and 8, it seems that, when the proportion of labelled



(a) Accuracy.



(b) F1 Score.

Fig. 9. SenseGAN-LaUL with $p\%$ labelled data as unlabelled data.

Table 2. Different design choices with $p = 10\%$ and $q = 100\%$.

(a) Accuracy.

	SenseGAN	SG-simpG	SG-simpD	SG-rnnD	SG-noGumbel	SG-noWGAN	SG-noPseudo	DeepSense
HHAR	**0.948**	0.932	0.926	0.935	0.918	0.920	0.930	0.920
UserID	**0.971**	0.950	0.946	0.949	0.917	0.916	0.917	0.942
Wisture	**0.973**	**0.973**	**0.973**	0.943	0.943	0.953	0.914	0.903

(b) F1 Score.

	SenseGAN	SG-simpG	SG-simpD	SG-rnnD	SG-noGumbel	SG-noWGAN	SG-noPseudo	DeepSense
HHAR	**0.927**	0.915	0.903	0.894	0.882	0.878	0.913	0.883
UserID	**0.959**	0.941	0.939	0.938	0.911	0.914	0.898	0.923
Wisture	**0.973**	**0.973**	**0.973**	0.939	0.939	0.942	0.884	0.889

data p is large, increasing the proportion of unlabelled data q can only make limited improvement. However, the unlabelled data does play an important role in SenseGAN from the perspective of reducing predictive error. Take UserID task with $p = 10\%$ as an example, SenseGAN reduces the error of accuracy from $5.8\%(= 100\% - 94.2\%)$ to $2.9\%(= 100\% - 97.1\%)$ and the error of F1 score from $0.077(= 1 - 0.923)$ to $0.041(= 1 - 0.959)$ by increasing q from 0% to 100%, which greatly reduces the errors by around 50%.

4.2.3 IoT Applications with Labelled Data as Unlabelled Data (LaUL). In a lot of proof-of-concept IoT applications, the total amount of data is limited. Researchers probably want to label all existing samples for training. However, concerns still exist that neural networks may cause performance degradation as shown in Figure 5c and 6c. In this experiment, we try to fully leverage the limited amount of data samples with SenseGAN by using them as both labelled and unlabelled data. Since the classifier is learned from both labelled supervision and joint interactions with the discriminator and generator in SenseGAN. Using labelled data as unlabelled data can enhance the discriminator and generator, and therefore boost the performance of classifier in return.

During the experiments, we randomly select $p\%$ of original training data as labelled data and discard all the rest training samples. These $p\%$ of data is used as both labelled and unlabelled data for training SenseGAN in the manner just described in Algorithm 1.

In Figure 9, we plot the results of training SenseGAN with Labelled data as UnLabelled data, called SenseGAN-LaUL, with $p = [1, 2, 3, 5, 10]\%$, each accompanied with a upper and a lower bound. The upper bound is training

SenseGAN with all left training samples as unlabelled data (*i.e.*, $q = 100\%$ in Section 4.2.2). The lower bound is the DeepSense model trained on only labelled data (*i.e.*, $q = 0\%$ in Section 4.2.2).

SenseGAN-LaUL attains a considerable improvement on all the three IoT tasks compared with the upper and lower bound. SenseGAN-LaUL can almost always achieve more than 50% of the maximum gain, *i.e.*, the middle point between the upper and lower bound, on three tasks with both accuracy and F1 score. Therefore, SenseGAN-LaUL is a good choice for training IoT applications with extremely limited data. Performance agains show that SenseGAN can efficiently utilize labelled and unlabelled data to effectively improve the predictive performance of neural network classifier on IoT tasks.

4.3 Ablation Study for Design Choices

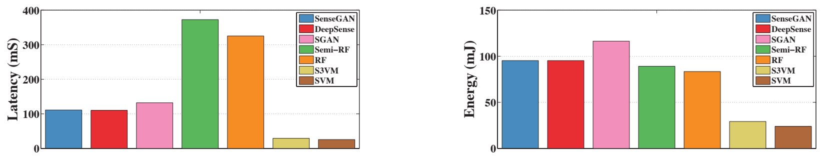
The previous experiments focus on evaluating the performance of SenseGAN for semi-supervised learning on IoT tasks. Recall that we have many design choices within the SenseGAN structure in Section 3. In this subsection, we evaluate these design choices of SenseGAN against baseline models generated by deleting one design component from the SenseGAN model at a time and measuring the impact. This approach results in the following baselines:

(1) *SG-noWGAN*: This model does not train the adversarial game with the Earth mover's distance, defined by Equation (3). Instead it uses the original less stable objective function, defined by Equation (1).
(2) *SG-noGumbel*: This model uses softmax instead of Gumbel-Softmax when feeding the output of classifier into the discriminator, which is discussed in Section 3.4.
(3) *SG-noPseudo*: This model deletes the pseudo positive data/label tuples $(\mathbf{X}_u^{(2)}, C(\mathbf{X}_u^{(2)}))$ in the training objective function (7).
(4) *SG-simpG*: This model uses a simple structure for the generator. As shown in Figure 3, instead of individual deconvolution layers for each sensor, this model uses single but larger deconvolution layers to generate outputs for all sensors.
(5) *SG-simpD*: This model uses a simple structure for the discriminator. As shown in Figure 4, instead of individual convolution layers for each sensor, this model uses single but larger convolution layers to extract the information from multiple sensor inputs altogether.
(6) *SG-rnnD*: Instead of using a full-convolution structure, this model uses the DeepSense structure (with RNN layers) [45] as the discriminator. To best of our knowledge, existing deep learning libraries do not support high-order gradient for RNN-based structure. Therefore, we use weight clipping [2] instead of the gradient penalty [18] for the WGAN training.

The baseline models SG-simpG and SG-simpD are designed such that they each have the same number of parameters as SenseGAN.

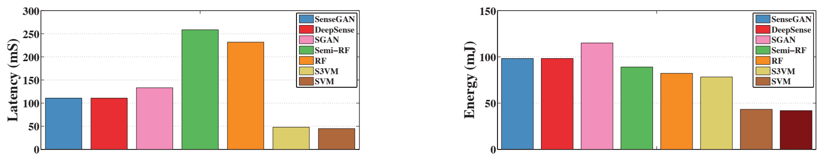
We evaluate all baseline algorithms as well as SenseGAN and DeepSense on the three IoT tasks with an illustrating case that randomly selects $p = 10\%$ of training samples as labelled data and regards all the rest as unlabelled data, *i.e.*, $q = 100\%$. SenseGAN and DeepSense work as the "upper bound" and the "lower bound" respectively. All baseline algorithms should perform better than the supervised counterpart by leveraging unlabelled data. SenseGAN should perform better than all baseline algorithms, if all design choices of SenseGAN are reasonable.

Experiment results are shown in Table 2. As seen, SenseGAN outperforms all baseline algorithms in all three tasks, providing strong empirical supports for our design choices. The Wisture takes only one sensor as input, so the structure of SenseGAN, SG-simpG, and SG-simpD are identical in this task. These three structures therefore have the same predictive performance for the Wisture task. However, SG-noGumbel and SG-noWGAN can easily perform worse than the supervised counterpart, DeepSense. This indicates that training instability can also affect the final performance of classifier by providing noisy supervision during the adversarial training. SG-simpG and SG-simpD also suffer from performance degradation. Therefore, our specifically designed structures for



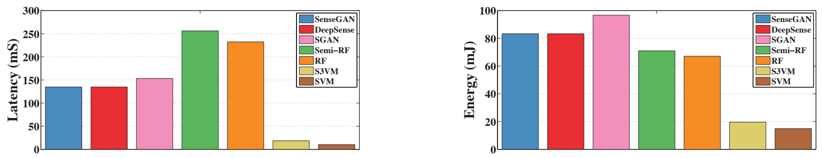
(a) Running time (b) Energy consumption

Fig. 10. Running time and energy consumption of HHAR



(a) Running time (b) Energy consumption

Fig. 11. Running time and energy consumption of UserID



(a) Running time (b) Energy consumption

Fig. 12. Running time and energy consumption of Wisture

handling multimodal sensing inputs are crucial to semi-supervised learning for IoT applications. SG-noPseudo also perform worse than the supervised counterpart. Thus pseudo positive data/label tuples are important to prevent discriminator from overfitting when the number of labelled data is limited. In addition, SenseGAN outperforms SG-rnnD in all three tasks, indicating the instability of RNN-based discriminator.

4.4 Energy and Time Efficiencies

Finally, we evaluate the energy and time efficiencies of SenseGAN when running on an IoT device, Intel Edison. Intel Edison computing platform is powered by the Intel Atom SoC dual-core CPU at 500 MHz and is equipped with 1GB memory and 4GB flash storage. For fairness, all models are run solely on CPU during experiments. Please notice that only the classifier of SenseGAN need to be loaded and executed on the IoT device during inference. The discriminator and generator only help the classifier to leverage unlabelled data during training, and they can be deleted after the training process.

We evaluate SenseGAN and all baseline models used in Section 4.2 trained on 10% of labelled data by measuring their per-inference running time and energy consumption. All the measurements are reported by taking the mean of 500 experiments.

The results on three IoT tasks are illustrated in Figure 10, 11, and 12. SenseGAN and its supervised counterpart, DeepSense, have almost the same running time and energy consumption on three tasks, while other semi-supervised models consume a little more time and energy during inference compared their supervised counterparts. This observation highlights the feature of SenseGAN that can leverage unlabelled sensing data for training without additional time and energy consumption during inference on IoT devices. In addition, random forest based algorithms, Semi-RF and RF, take relatively long time to run. It is because random forest models consist

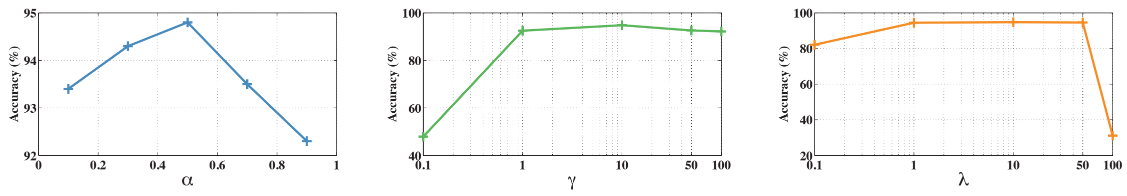


Fig. 13. Accuracy of HHAR with hyperparameters α, γ, and λ.

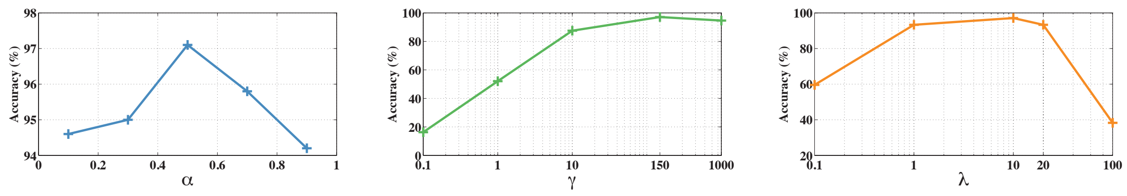


Fig. 14. Accuracy of UserID with hyperparameters α, γ, and λ.

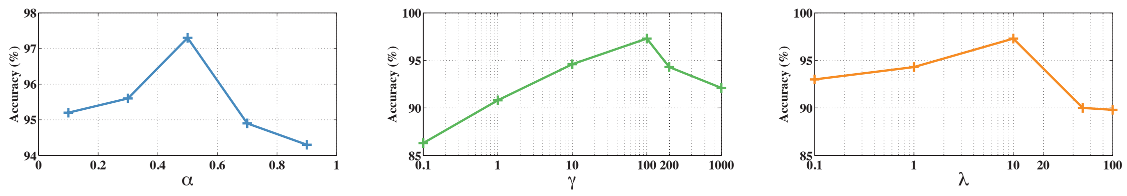


Fig. 15. Accuracy of Wisture with hyperparameters α, γ, and λ.

of long decision paths that contain a large number of condition operations, which slow down the instruction pipelining in the CPU.

4.5 Hyper-parameter Tuning

The training process of SenseGAN is controlled by several hyper-parameters. α controls the tradeoff between two types of negative tuples created by the generator and the classifier. γ controls the tradeoff between loss functions generated by the supervised classification error and the adversarial game among three components. λ controls the strength of gradient penalty. In this subsection, we evaluate the final performance of SenseGAN with different choices of these three hyper-parameters on three IoT tasks.

We still evaluate with an illustrating case that randomly selects $p = 10\%$ of training samples as labelled data and regards all the rest as unlabelled data, i.e., $q = 100\%$. The default value of α is 0.5; λ is 10; and γ is 10, 150, and 100 for HHAR, UserID, and Wisture respectively. During each set of experiments, we only change the value of target hyper-parameter, while keeping all other hyper-parameters as the default values. The tradeoffs between the prediction accuracy of the choices of hyper-parameters are shown in Figure 13, 14, and 15.

For α, it achieves the best performance with 0.5 on all three tasks. Therefore, treating two types of negative tuples equally helps to improve the predictive performance of SenseGAN. Similarly, λ consistently achieves the best performance when equals to 10 in all three tasks. These experiments show that fixed fault values work well for α and λ in practice, so we do not have to tune the value of α and λ for different tasks. However, the values of γ are different when achieving the highest accuracy in three tasks. When γ is small, the classifier cannot obtain enough learning signal from the labelled data. When γ is large, the learning signal from the labelled data can overwhelms the learning signal from the adversarial game. Therefore, we need to fine tune the value of γ to balance these two types of learning signals for different tasks.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we present a semi-supervised deep-learning framework, SenseGAN, for IoT applications. SenseGAN separates the functionalities of discriminator and classifier into two neural networks, designs specific generator and discriminator structures for handling multimodal sensing inputs, and stabilizes and enhances the adversarial training process by WGAN with gradient penalty as well as Gumbel-Softmax for categorical representations. The evaluation empirically shows that SenseGAN can efficiently leverage both labelled and unlabelled data to effectively improve the predictive power of the classifier without additional time and energy consumption during the inference.

Several improvement opportunities remain. First, our architecture for fusing multi-sensor data is not yet optimal. For example, we can use attention-based structures [41] to guide the discriminator and the generator to focus on data segments containing more representative information. Second, SenseGAN focuses on the classification problem in the IoT scenario. However, regression is another common IoT task that needs to be considered. To solve the regression problem with SenseGAN, we can transform the range of continuous target outputs into a set of intervals that can be used as discrete classes. By treating regression problems as classification problems, SenseGAN can be applied in the manner described in this paper. However, we still need further studies to formally solve this problem. Third, the learning process of SenseGAN is computationally intensive. Therefore, more studies are needed to enable online adaptive learning with streaming unlabelled sensing data on low-power IoT devices. Finally, more evaluation is needed in the context of deployed application scenarios to better understand the feasibility of needed (albeit minimal) labeling and the limitations of the approach.

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] Intel edison compute module. http://www.intel.com/content/dam/support/us/en/documents/edison/sb/edison-module_HG_331189.pdf.
[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.
[3] J. S. Bauer, S. Consolvo, B. Greenstein, J. Schooler, E. Wu, N. F. Watson, and J. Kientz. Shuteye: encouraging awareness of healthy sleep recommendations with a mobile, peripheral display. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1401–1410. ACM, 2012.
[4] K. P. Bennett and A. Demiriz. Semi-supervised support vector machines. In *Advances in Neural Information processing systems*, pages 368–374, 1999.
[5] F. R. Bentley, Y.-Y. Chen, and C. Holz. Reducing the stress of coordination: sharing travel time information between contacts on mobile phones. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 967–970. ACM, 2015.
[6] K.-Y. Chen, D. Ashbrook, M. Goel, S.-H. Lee, and S. Patel. Airlink: sharing files between multiple devices using in-air gestures. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 565–569. ACM, 2014.
[7] T. Choudhury, S. Consolvo, B. Harrison, J. Hightower, A. LaMarca, L. LeGrand, A. Rahimi, A. Rea, G. Bordello, B. Hemingway, et al. The mobile sensing platform: An embedded activity recognition system. *IEEE Pervasive Computing*, 7(2), 2008.
[8] J. Chung, M. Donahoe, C. Schmandt, I.-J. Kim, P. Razavai, and M. Wiseman. Indoor location sensing using geo-magnetism. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 141–154. ACM, 2011.
[9] Y. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. In *ICML*, 2017.

[10] E. Denton, S. Gross, and R. Fergus. Semi-supervised learning with context-conditional generative adversarial networks. *arXiv preprint arXiv:1611.06430*, 2016.

[11] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

[12] M. Faurholt-Jepsen, M. Vinberg, M. Frost, S. Debel, E. Margrethe Christensen, J. E. Bardram, and L. V. Kessing. Behavioral activities collected through smartphones and the association with illness activity in bipolar disorder. *International journal of methods in psychiatric research*, 25(4):309–323, 2016.

[13] D. Figo, P. C. Diniz, D. R. Ferreira, and J. M. Cardoso. Preprocessing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing*, 14(7):645–662, 2010.

[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[15] E. Greensmith, P. L. Bartlett, and J. Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530, 2004.

[16] E. Griffiths, T. S. Saponas, and A. Brush. Health chair: implicitly sensing heart and respiratory rate. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 661–671. ACM, 2014.

[17] T. Grosse-Puppendahl, X. Dellangnol, C. Hatzfeld, B. Fu, M. Kupnik, A. Kuijper, M. R. Hastall, J. Scott, and M. Gruteser. Platypus: Indoor localization and identification through sensing of electric potential changes in human bodies. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pages 17–30. ACM, 2016.

[18] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[19] N. Y. Hammerla, R. Kirkham, P. Andras, and T. Ploetz. On preserving statistical characteristics of accelerometry data using their empirical cumulative distribution. In *Proceedings of the 2013 International Symposium on Wearable Computers*, pages 65–68. ACM, 2013.

[20] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[21] M. A. A. Haseeb and R. Parasuraman. Wisture: Rnn-based learning of wireless signals for gesture recognition in unmodified smartphones. *arXiv preprint arXiv:1707.08569*, 2017.

[22] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[23] Y. Jiang, X. Pan, K. Li, Q. Lv, R. P. Dick, M. Hannigan, and L. Shang. Ariel: Automatic wi-fi based room fingerprinting for indoor localization. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 441–450. ACM, 2012.

[24] S. Kaiser, A. Parks, P. Leopard, C. Albright, J. Carlson, M. Goel, D. Nassehi, and E. C. Larson. Design and learnability of vortex whistles for managing chronic lung function via smartphones. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 569–580. ACM, 2016.

[25] C. Koehler, N. Banovic, I. Oakley, J. Mankoff, and A. K. Dey. Indoor-alps: an adaptive indoor location prediction system. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 171–181. ACM, 2014.

[26] N. D. Lane, S. Bhattacharya, A. Mathur, P. Georgiev, C. Forlivesi, and F. Kawsar. Squeezing deep learning into mobile and embedded devices. *IEEE Pervasive Computing*, 16(3):82–88, 2017.

[27] N. D. Lane, P. Georgiev, and L. Qendro. Deepear: robust smartphone audio sensing in unconstrained acoustic environments using deep learning. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 283–294. ACM, 2015.

[28] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[29] C. Li, K. Xu, J. Zhu, and B. Zhang. Triple generative adversarial nets. *arXiv preprint arXiv:1703.02291*, 2017.

[30] A. Mannini, S. S. Intille, M. Rosenberger, A. M. Sabatini, and W. Haskell. Activity recognition using a single accelerometer placed at the wrist or ankle. *Medicine and science in sports and exercise*, 45(11):2193, 2013.

[31] P. Melgarejo, X. Zhang, P. Ramanathan, and D. Chu. Leveraging directional antenna capabilities for fine-grained gesture recognition. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 541–551. ACM, 2014.

[32] A. Odena. Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*, 2016.

[33] Q. Pu, S. Gupta, S. Gollakota, and S. Patel. Whole-home gesture recognition using wireless signals. In *Proceedings of the 19th annual international conference on Mobile computing & networking*, pages 27–38. ACM, 2013.

[34] V. Radu, N. D. Lane, S. Bhattacharya, C. Mascolo, M. K. Marina, and F. Kawsar. Towards multimodal deep learning for activity recognition on mobile devices. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, pages 185–188. ACM, 2016.

[35] A. Stisen, H. Blunck, S. Bhattacharya, T. S. Prentow, M. B. Kjærgaard, A. Dey, T. Sonne, and M. M. Jensen. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 127–140. ACM, 2015.

[36] T. Toscos, K. Connelly, and Y. Rogers. Best intentions: health monitoring technology and children. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 1431–1440. ACM, 2012.

[37] I. Triguero, S. García, and F. Herrera. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information Systems*, 42(2):245–284, 2015.

[38] C. Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[39] E. J. Wang, W. Li, D. Hawkins, T. Gernsheimer, C. Norby-Slycord, and S. N. Patel. Hemaapp: noninvasive blood screening of hemoglobin using smartphone cameras. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 593–604. ACM, 2016.

[40] J. Weppner, B. Bischke, and P. Lukowicz. Monitoring crowd condition in public spaces by tracking mobile consumer devices with wifi interface. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, pages 1363–1371. ACM, 2016.

[41] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[42] L. Yao, F. Nie, Q. Z. Sheng, T. Gu, X. Li, and S. Wang. Learning from less for better: semi-supervised activity recognition via shared structure discovery. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 13–24. ACM, 2016.

[43] S. Yao, M. T. Amin, L. Su, S. Hu, S. Li, S. Wang, Y. Zhao, T. Abdelzaher, L. Kaplan, C. Aggarwal, et al. Recursive ground truth estimator for social data streams. In *Proceedings of the 15th International Conference on Information Processing in Sensor Networks*, page 14. IEEE Press, 2016.

[44] S. Yao, S. Hu, S. Li, Y. Zhao, L. Su, L. Kaplan, A. Yener, and T. Abdelzaher. On source dependency models for reliable social sensing: Algorithms and fundamental error bounds. In *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*, pages 467–476. IEEE, 2016.

[45] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In *Proceedings of the 26th International Conference on World Wide Web*, pages 351–360. International World Wide Web Conferences Steering Committee, 2017.

[46] S. Yao, Y. Zhao, H. Shao, A. Zhang, C. Zhang, S. Li, and T. Abdelzaher. Rdeepsense: Reliable deep mobile computing models with uncertainty estimations. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4):173, 2018.

[47] S. Yao, Y. Zhao, A. Zhang, S. Hu, H. Shao, C. Zhang, L. Su, and T. Abdelzaher. Deep learning for the internet of things. *Computer*, 51(5):32–41, 2018.

[48] S. Yao, Y. Zhao, A. Zhang, L. Su, and T. Abdelzaher. Deepiot: Compressing deep neural network structures for sensing systems with a compressor-critic framework. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. ACM, 2017.

[49] H.-S. Yeo, J. Lee, A. Bianchi, and A. Quigley. Watchmi: pressure touch, twist and pan gesture input on unmodified smartwatches. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 394–399. ACM, 2016.

[50] C. Zhang, L. Liu, D. Lei, Q. Yuan, H. Zhuang, T. Hanratty, and J. Han. Triovecevent: Embedding-based online local event detection in geo-tagged tweet streams. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 595–604. ACM, 2017.

[51] C. Zhang, K. Zhang, Q. Yuan, H. Peng, Y. Zheng, T. Hanratty, S. Wang, and J. Han. Regions, periods, activities: Uncovering urban dynamics via cross-modal representation learning. In *Proceedings of the 26th International Conference on World Wide Web*, pages 361–370. International World Wide Web Conferences Steering Committee, 2017.

[52] Y. Zhang, M. K. Chong, J. Müller, A. Bulling, and H. Gellersen. Eye tracking for public displays in the wild. *Personal and Ubiquitous Computing*, 19(5-6):967–981, 2015.

Received February 2018; revised May 2017; accepted September 2018