Big Data Challenges for the Internet of Things

Shuochao Yao

Internet of Things (IoT)

Smart Home

Embedded & Mobile Devices

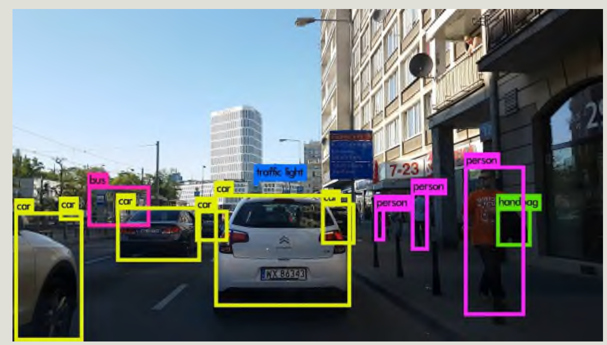
Smart City

Deep Learning

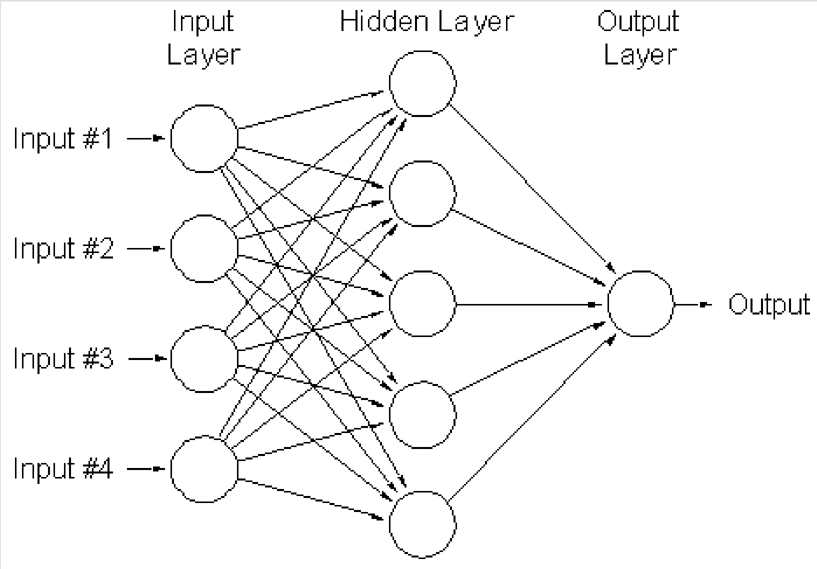
Speech Recognition

Activity Recognition

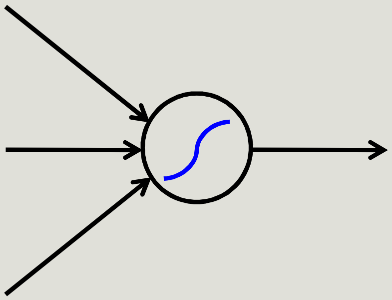
Object Detection



Recap: Fully-connected neural network

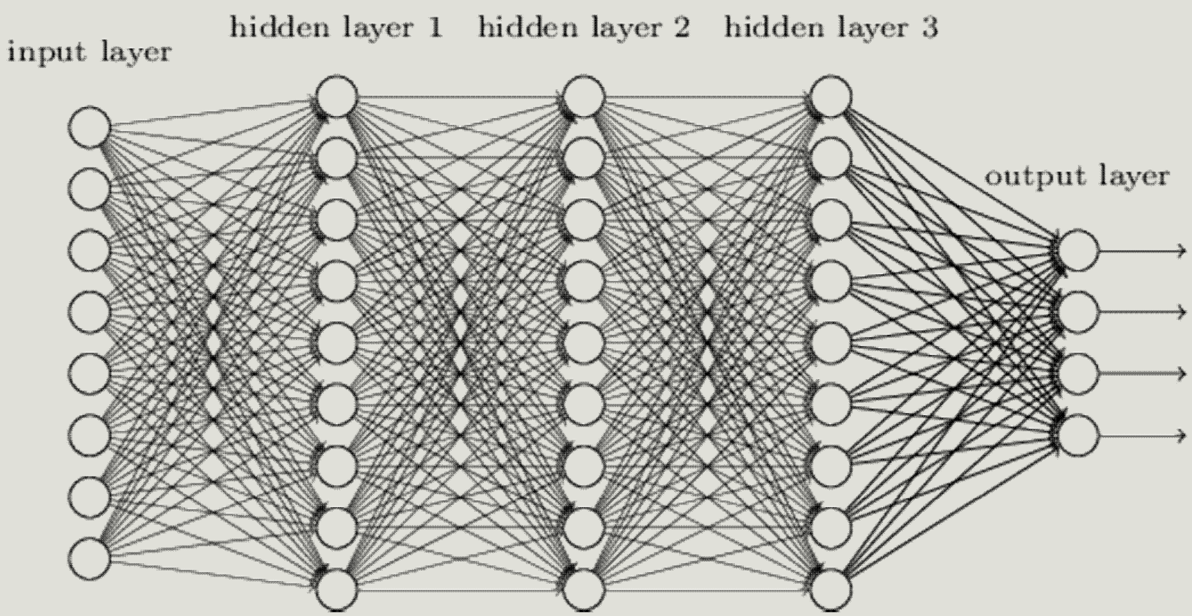


- Can learn nonlinear functions provided each perceptron has a differentiable nonlinearity



Sigmoid: $g(t) = \frac{1}{1+e^{-t}}$

Recap: Fully-connected neural network



From fully connected to convolutional networks

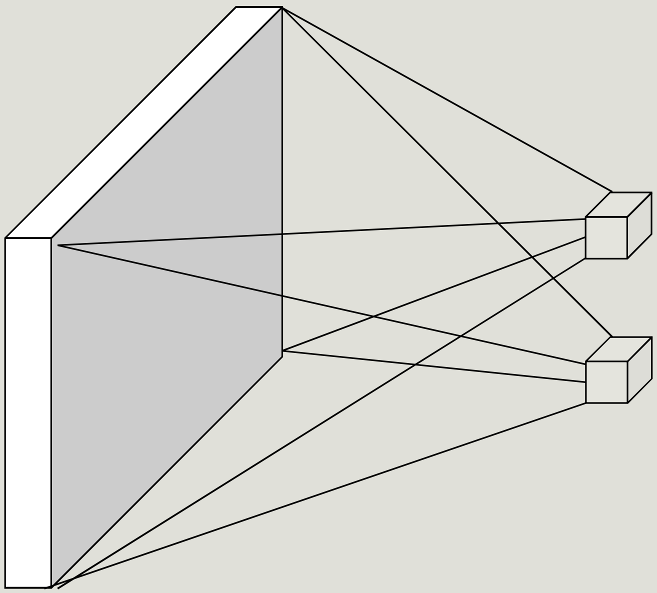
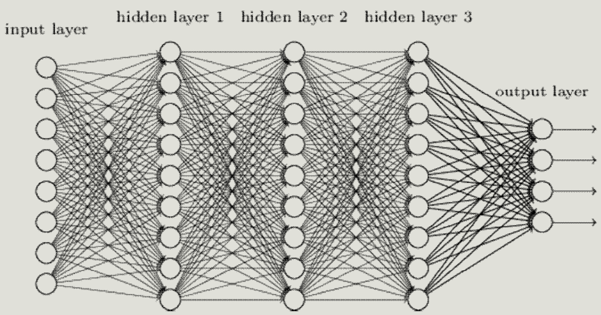
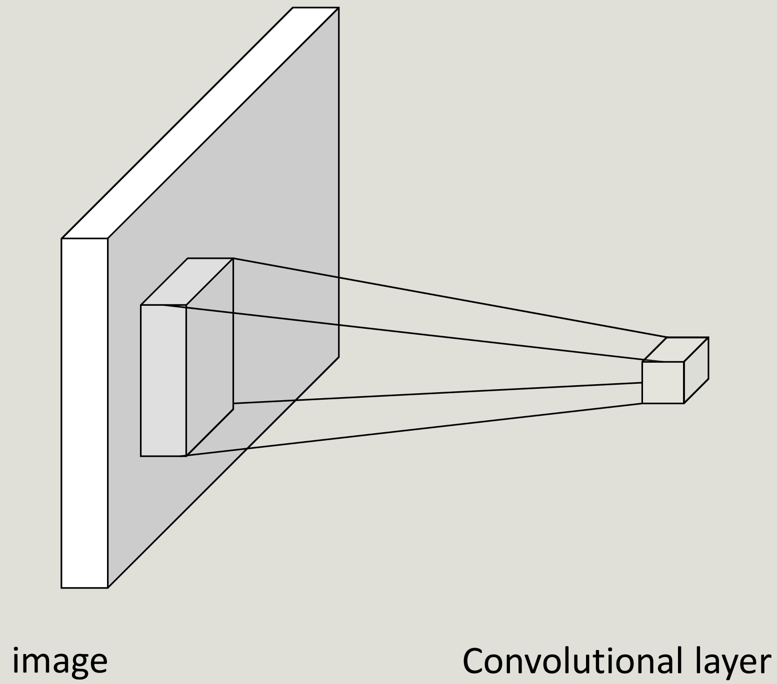


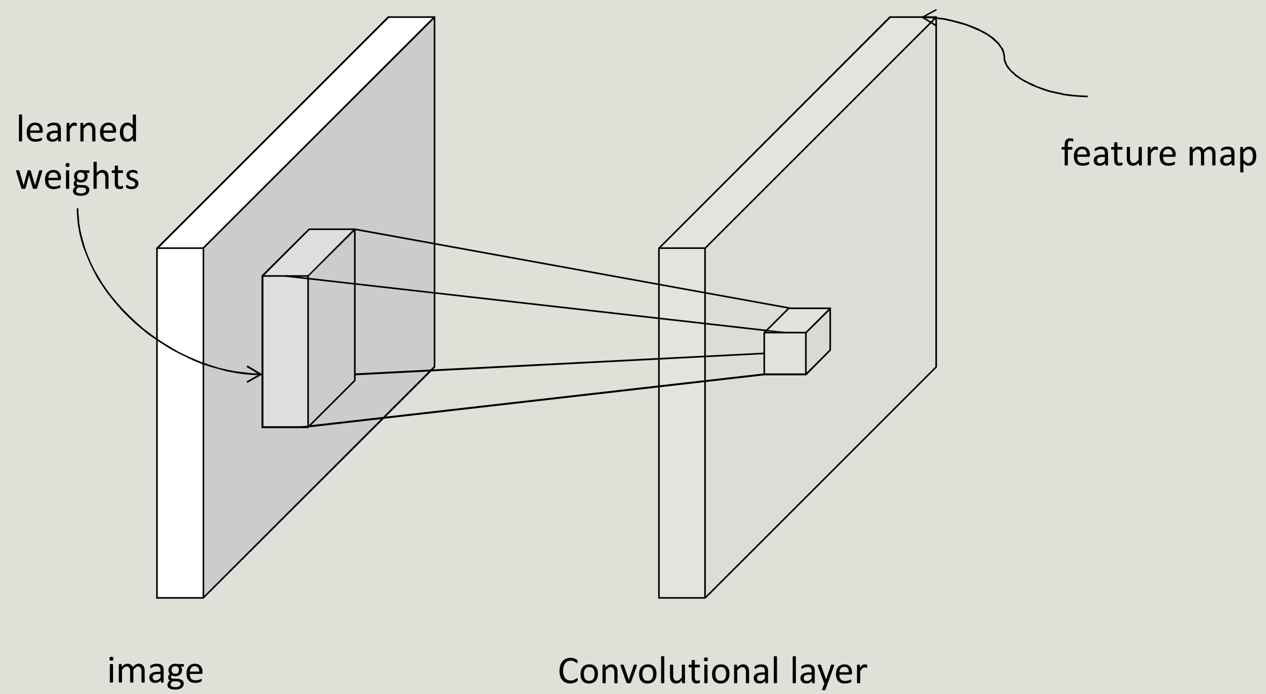
image Fully connected layer



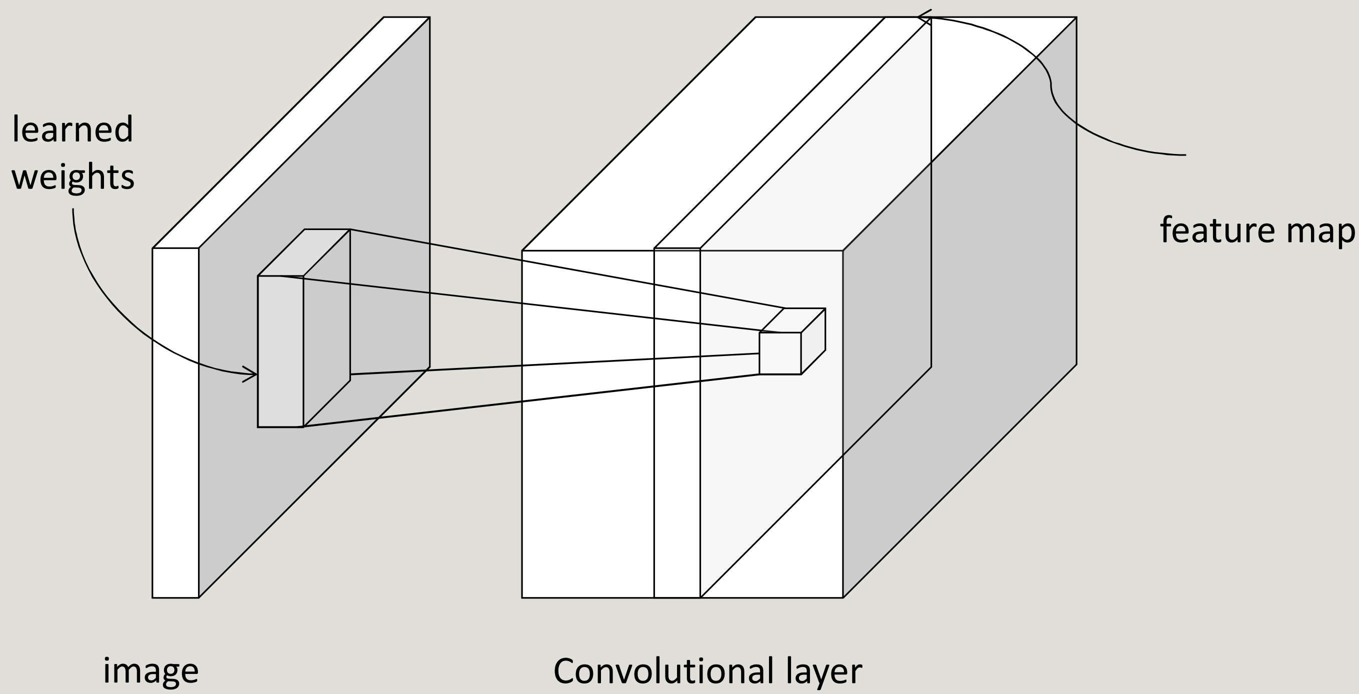
Convolutional neural networks



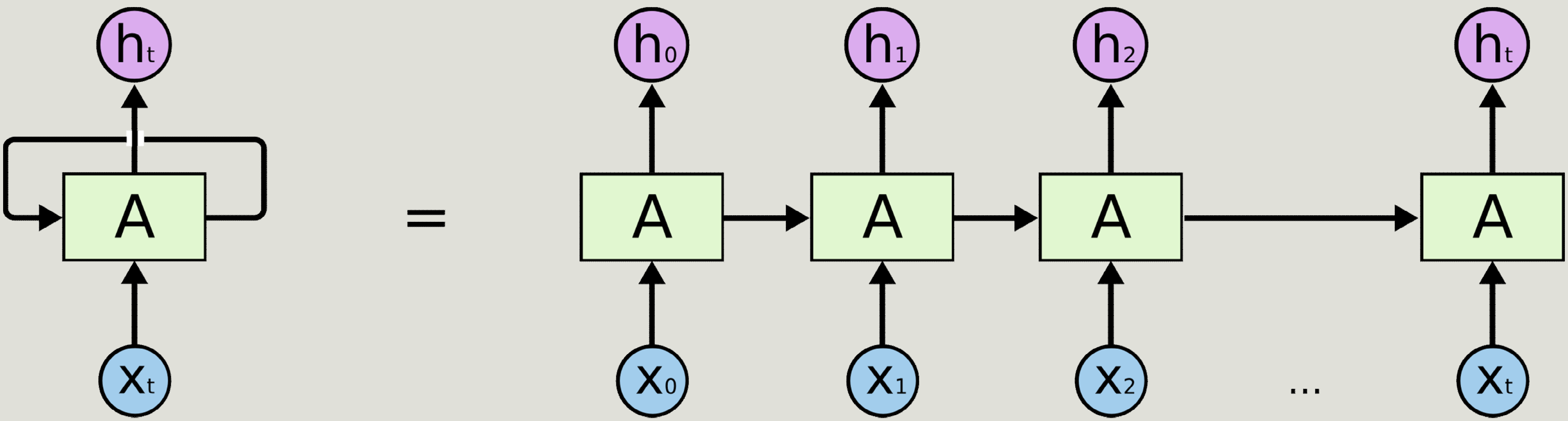
Convolutional neural networks



Convolutional neural networks



Recap: Recurrent neural network



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Challenges

Deep learning for sensor-rich IoT systems.

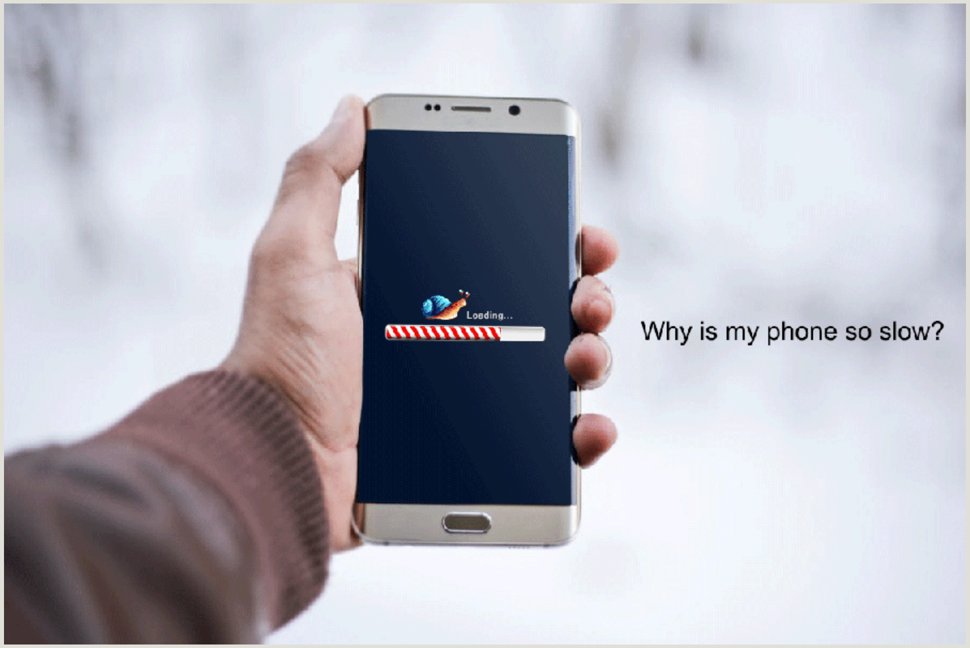
Deep learning for resource-constrained IoT systems.

Deep learning for reliable IoT systems.

Deep learning for label-limited IoT systems.

Sensor-rich IoT systems

Resource-constrained IoT systems



Reliable IoT system

Label-limited IoT system

About 1 million people work as full-time or part-time data labellers

Outline

DeepSense: A unified deep learning framework for time-series mobile sensing data processing. (WWW 2017)

RDeepSense: Reliable Deep Mobile Computing Models with Uncertainty Estimations. (Ubicomp 2018)

DeepIoT: Compressing Deep Neural Network Structures for Sensing Systems with a Compressor-Critic Framework. (SenSys 2017)

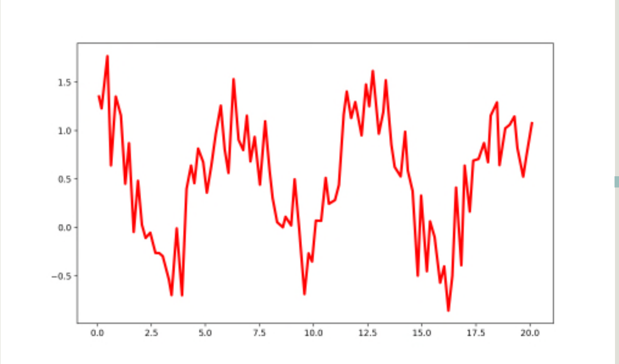
Outline

DeepSense: A unified deep learning framework for time-series mobile sensing data processing. (WWW 2017)

RDeepSense: Reliable Deep Mobile Computing Models with Uncertainty Estimations. (Ubicomp 2018)

DeepIoT: Compressing Deep Neural Network Structures for Sensing Systems with a Compressor-Critic Framework. (SenSys 2017)

Challenges

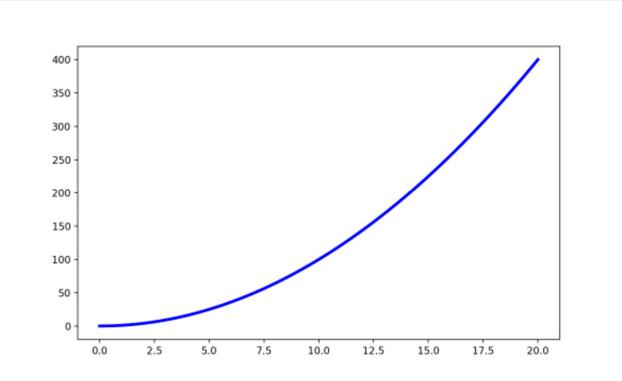


Sensor Input: noisy physical quantity

Noise Model

Physical Rule

$$\frac{\partial^2 d}{\partial^2 t} = a$$

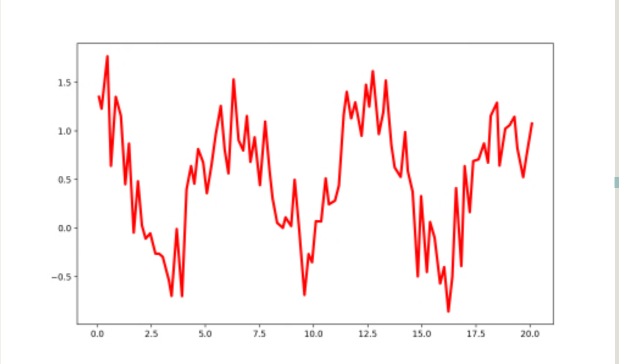


Out: clean target physical quantity

Challenges

Nonlinear
Time-dependent

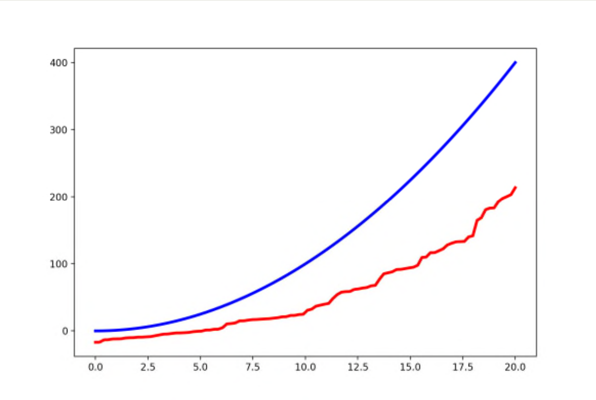
Noise Model



Sensor Input: noisy physical quantity

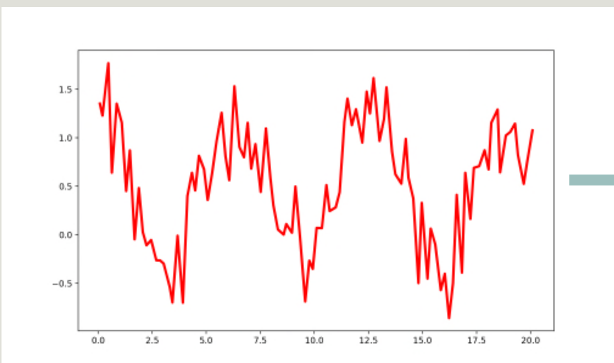
Physical Rule

$$\frac{\partial^2 d}{\partial^2 t} = a$$



Out: noisy target physical quantity

Challenges

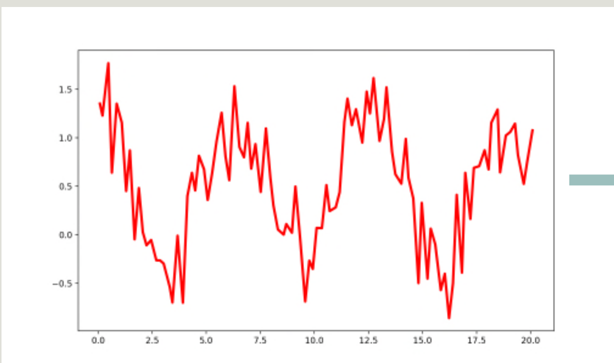


Sensor Input

Hand-crafted features

Classifier

Challenges

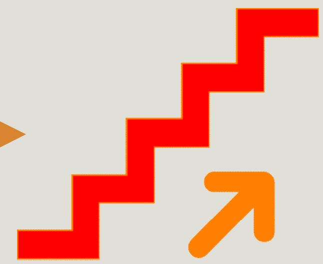


Sensor Input

Hand-crafted features

Time-consuming
Not Robust

Classifier



Hand-crafted features

Table 7 Summary of classification of time-domain techniques regarding computational costs, storage requirements, and precision (double/single/int)

Time-domain metric	Ref(s)	Comp. cost	Storage req.	Precision	Mobile device
Mean	[5, 2 [illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Std. deviation	[15, [illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Median	[2, [illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Range	[11]	[illegible]	[illegible]	[illegible]	[illegible]
Maximum	[4, [illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Minimum	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
RMS	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Integration	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Correlation	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Cross-correl [illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Differences	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Zero-cross [illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
SMA	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
SVM	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
DSVM	[19]	[illegible]	[illegible]	[illegible]	[illegible]

Table 8 Summary of classification of frequency-domain techniques regarding computational costs, storage requirements and precision (double/single/int)

Frequency-domain metric	Ref(s)	Comp. cost	Storage req.	Precision	Mobile device
Energy	[5, 16, 18, 27, 28, 41]	Medium	Low	Double/single	Moderate
Entropy	[5, 16, 18, 28]	High	Low	Double/single	No
Coeff. sum	[62]	Medium	Low	Double/single	Moderate
[illegible] freq.	[21, 22, 23, 24, 37]	Medium	Low	Double/single	Moderate
[illegible]	[illegible]	Low	Low	Double/single	Yes

Table 9 Summary of classification of symbolic string-domain techniques regarding computational costs, storage requirements, and precision (double/single/int)

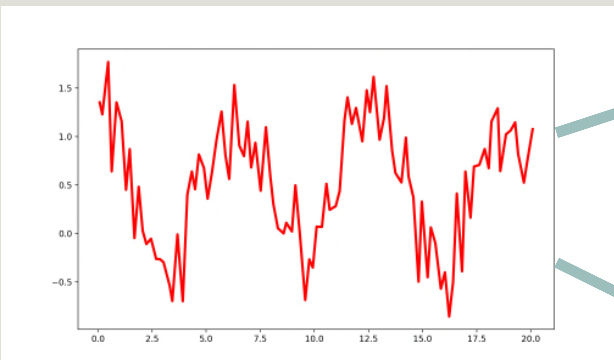
String-domain metric	Ref(s)	Comp. cost	Storage req.	Precision	Mobile device
Minimum distance	[31]	Low	Low	Int	Yes
Levenshtein	[14]	Medium	Medium	Int	Moderate
DTW	[46]	Medium	Medium	Int	Moderate

DeepSense: a Unified Model

A deep learning model that models different types of mobile sensing applications in a unified manner.

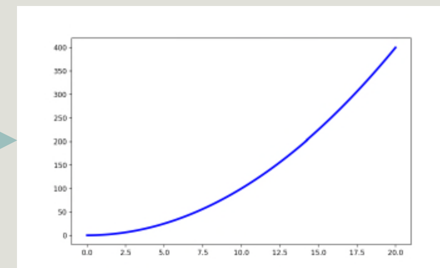
DeepSense: a Unified Model

A learnable complex nonlinear functions:
composition of physical system and noise model



Sensor Input

DeepSense



DeepSense

An automatic feature extractor
and classifier

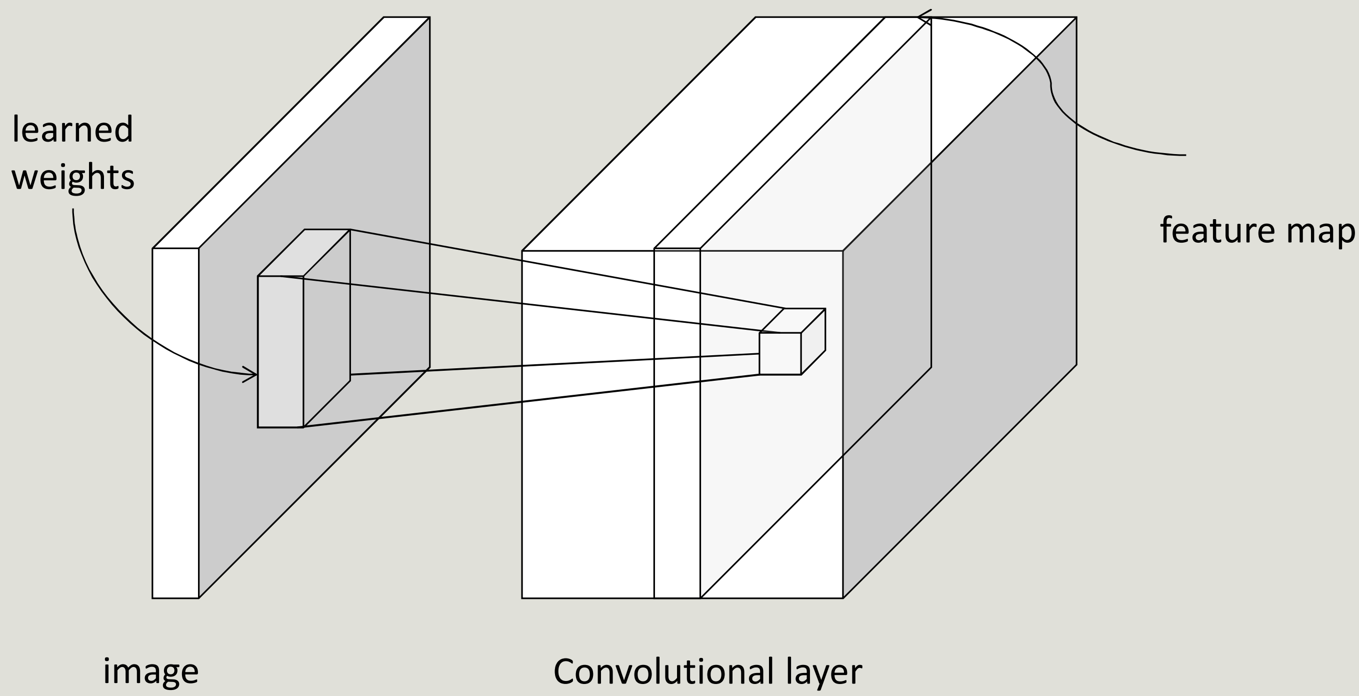
DeepSense: Properties

Target physical quantity
- Multiple sensor inputs (input physical quantities).
- Physical rules involve single quantity.
- Physical rules involve multiple quantities.
- Physical rules involve time.

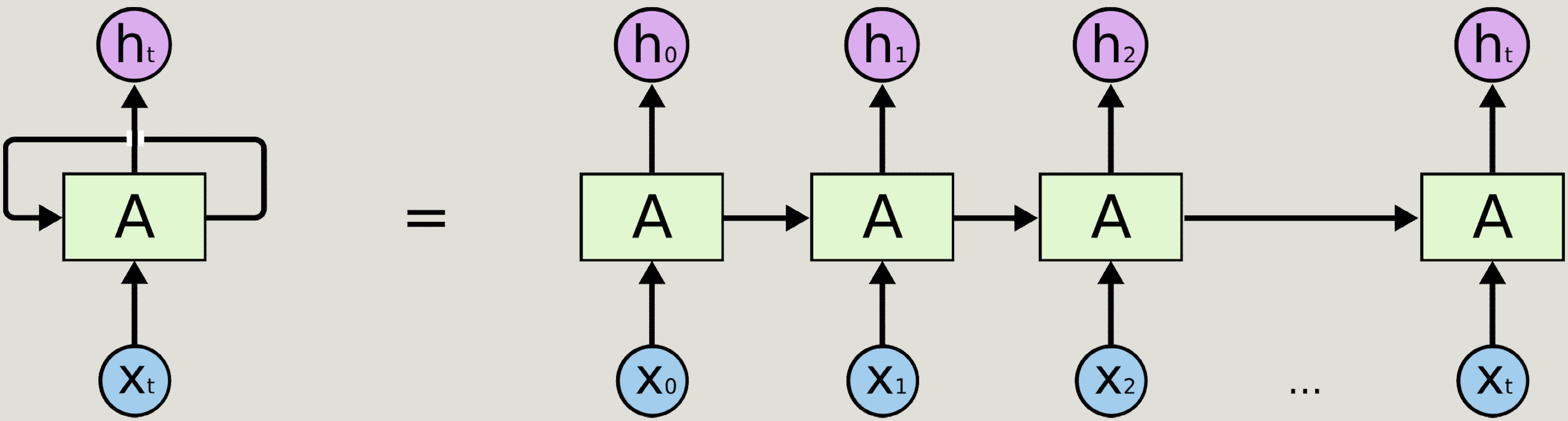
- Target classes
  - Multiple sensor inputs.
  - Local features within each sensor input.
  - Global features that fuse multiple senor inputs.
  - Temporal dependencies.

- DeepSense
  - Interactions with single sensor.
  - Interactions with multiple sensors.
  - Interactions along time.

Recap: Convolutional neural networks

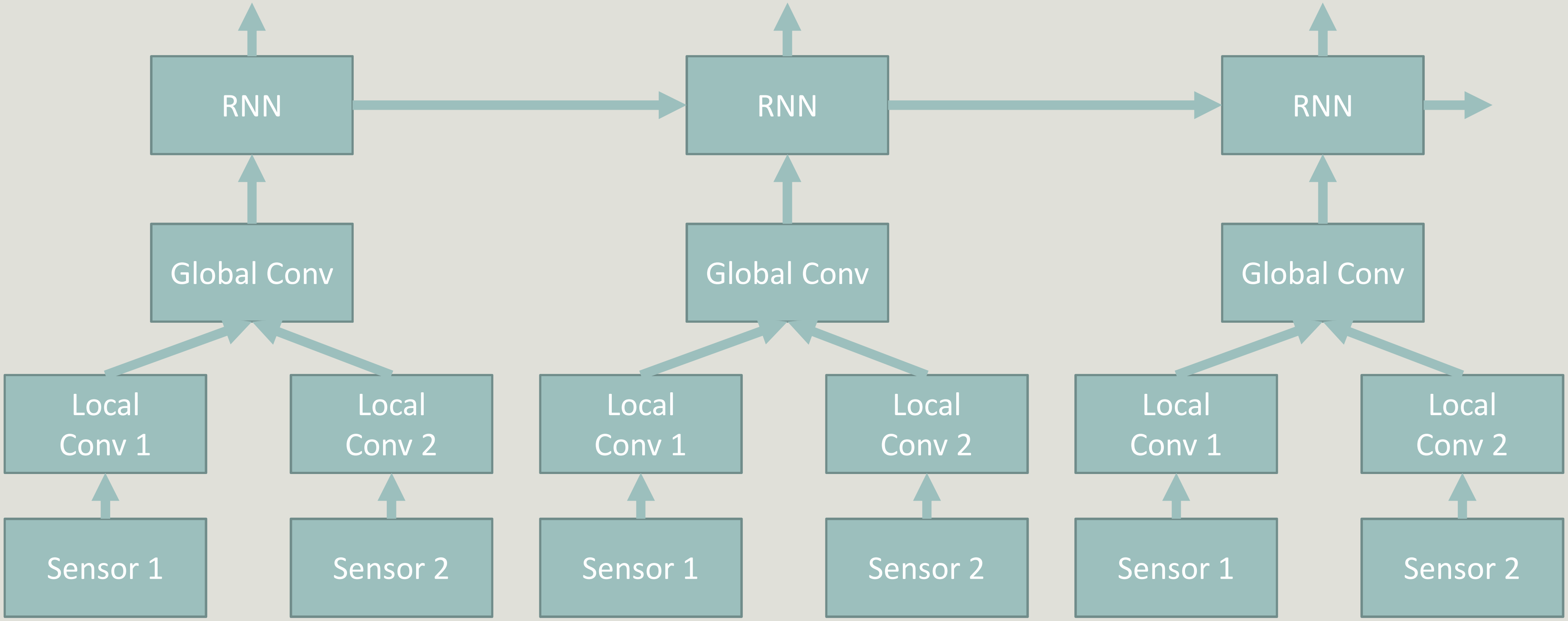


Recap: Recurrent neural network

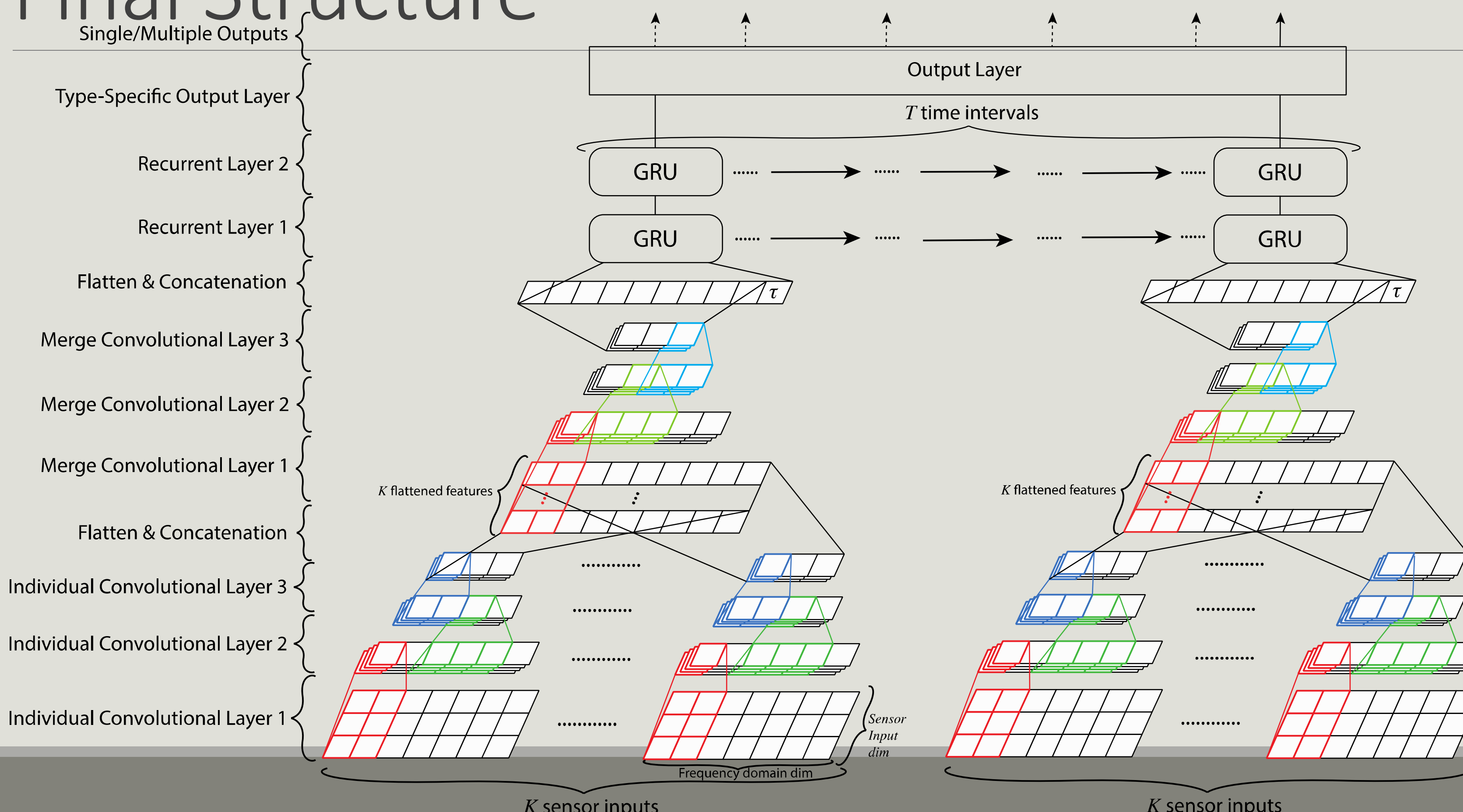


http://colah.github.io/posts/2015-08-Understanding-LSTMs/

DeepSense: Network Structure



Final Structure



DeepSense: Customization

Most Structure is pre-defined with default values.

For a particular mobile sensing task, you need only to define:

- Number of sensor inputs.
- Input/output dimension.
- Regression/classification.

More customization:

- Objective function for training.

Evaluation Tasks

Car tracking with motion sensors (CarTrack)

- Regression Based
- GPS is unavailable in underground road
- Sensing error will be accumulated and there is no additional signal to erase the error
- The capability of DeepSense of ***learning physical rules for noisy sensor data****.*

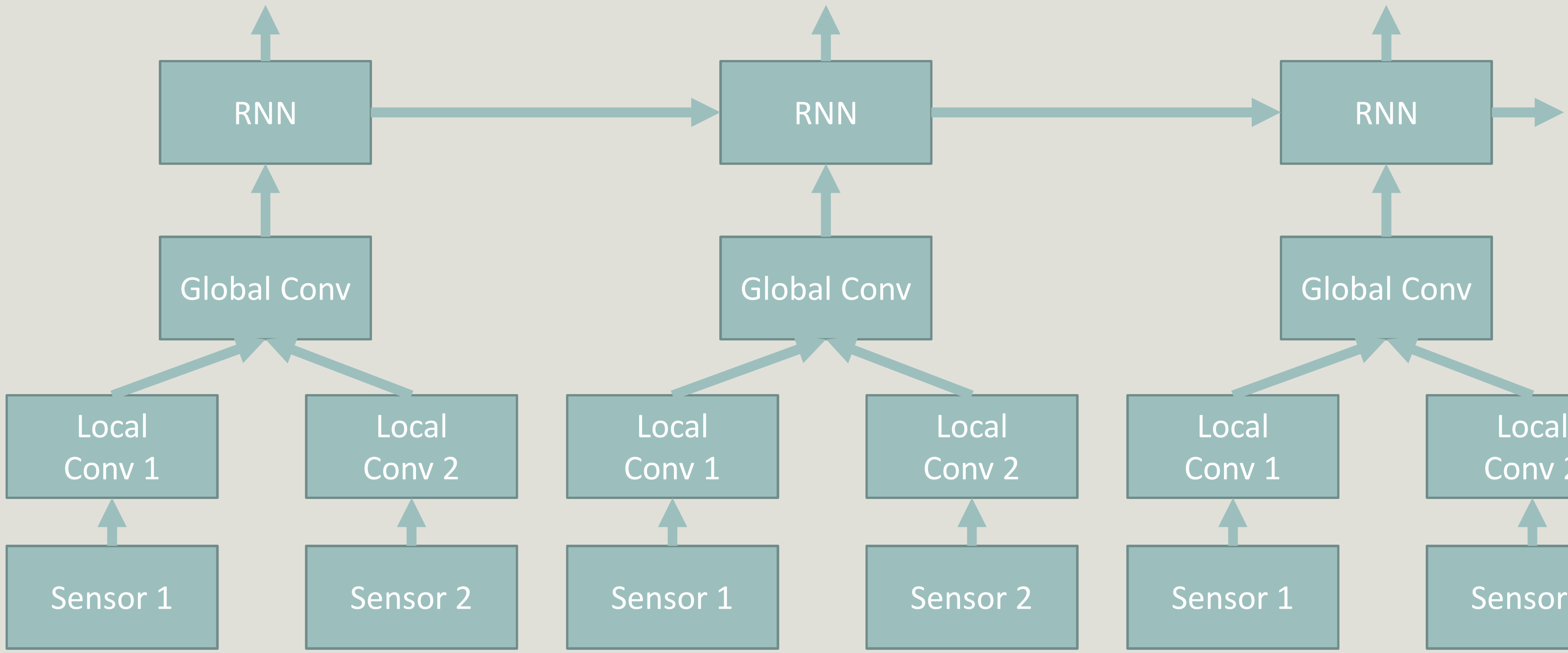
Heterogeneous Human activity recognition (HHAR)

- Classification Based
- State-of-the-art algorithms do not generalize well for a new user who does not appear in the training set
- The capability of DeepSense to ***extract features that generalize well.***

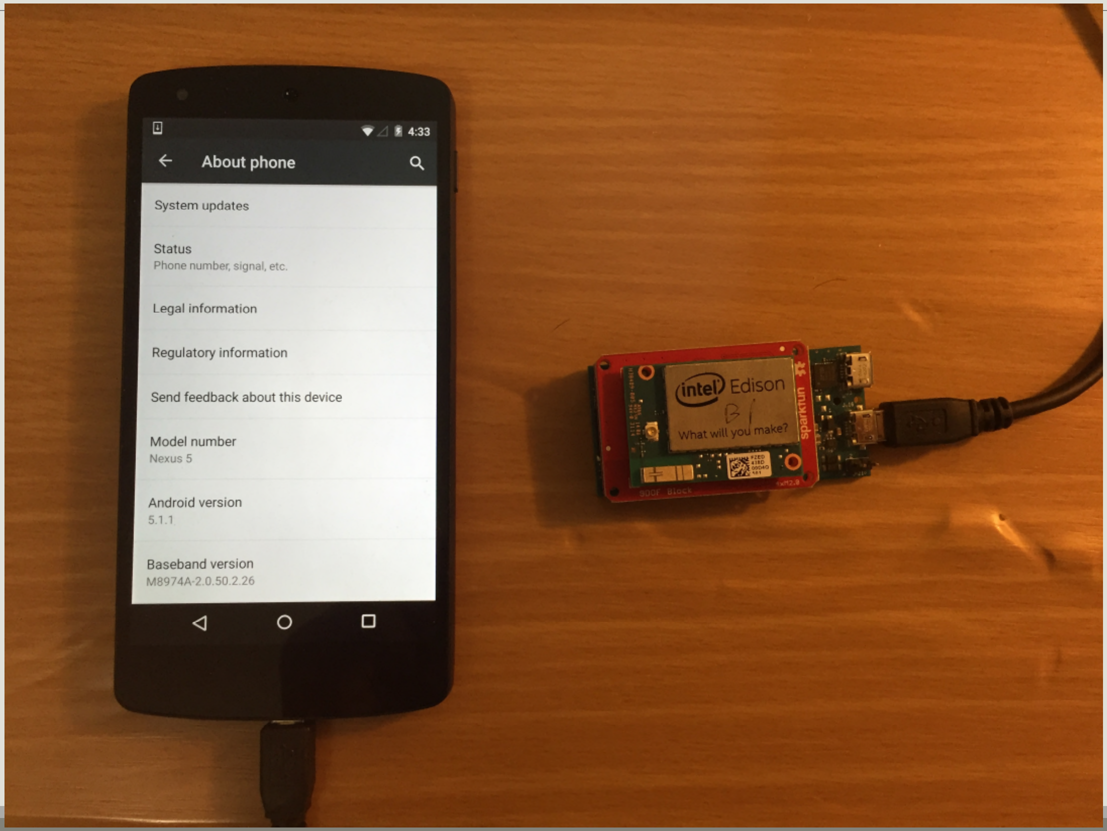
User Identification with motion analysis (UserID)

- Classification Based
- Extend the biometric gait analysis for user identification (walking, biking, stairing up/down, sitting, and standing)
- The capability of DeepSense to ***extract features that differentiate well.***

Evaluation Baselines



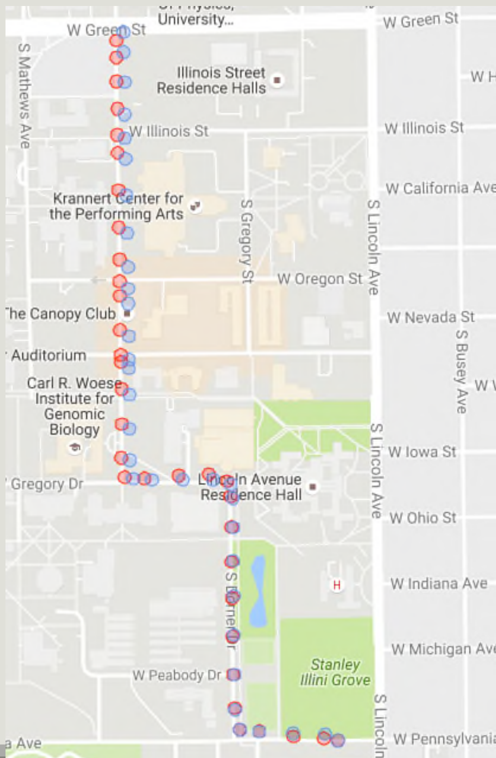
Testing Platform

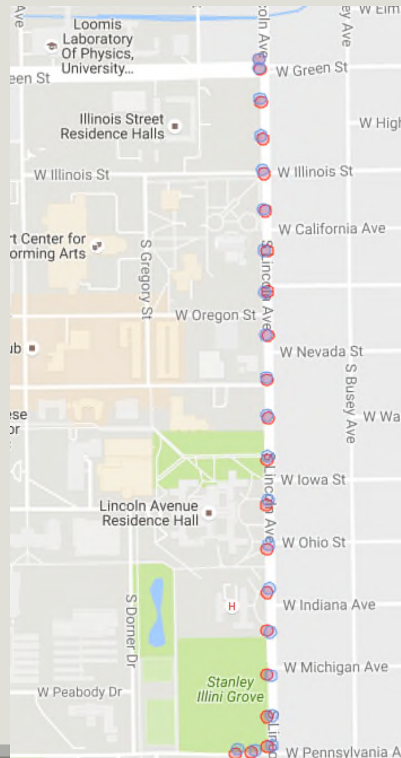


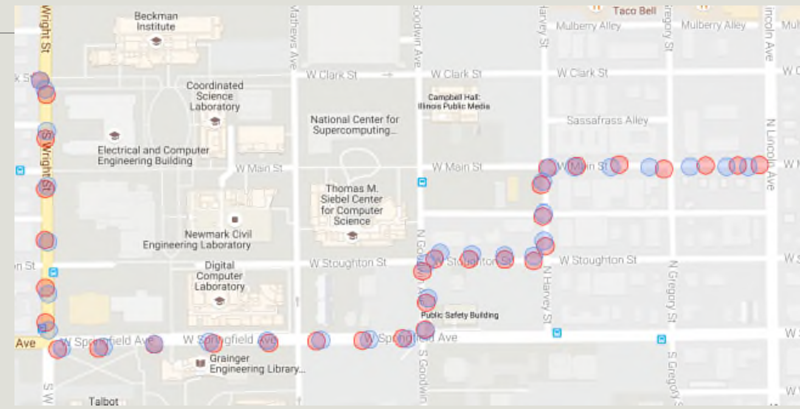
CarTrack: Accuracy

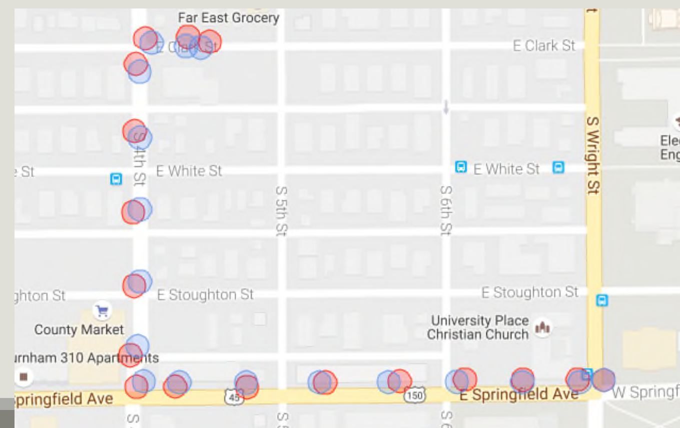
	MAE (meter)	Map-Aided Track
DeepSense	**40.43 ± 5.24**	**93.8**%
DS-SingleGRU	44.97 ± 5.80	90.2%
DS-noIndvConv	52.15 ± 6.24	88.3%
DS-noMergeConv	53.06 ± 6.59	87.5%
Sensor-fusion	606.59 ± 56.57	
eNav (w/o GPS)		6.7%

CarTrack: Examples

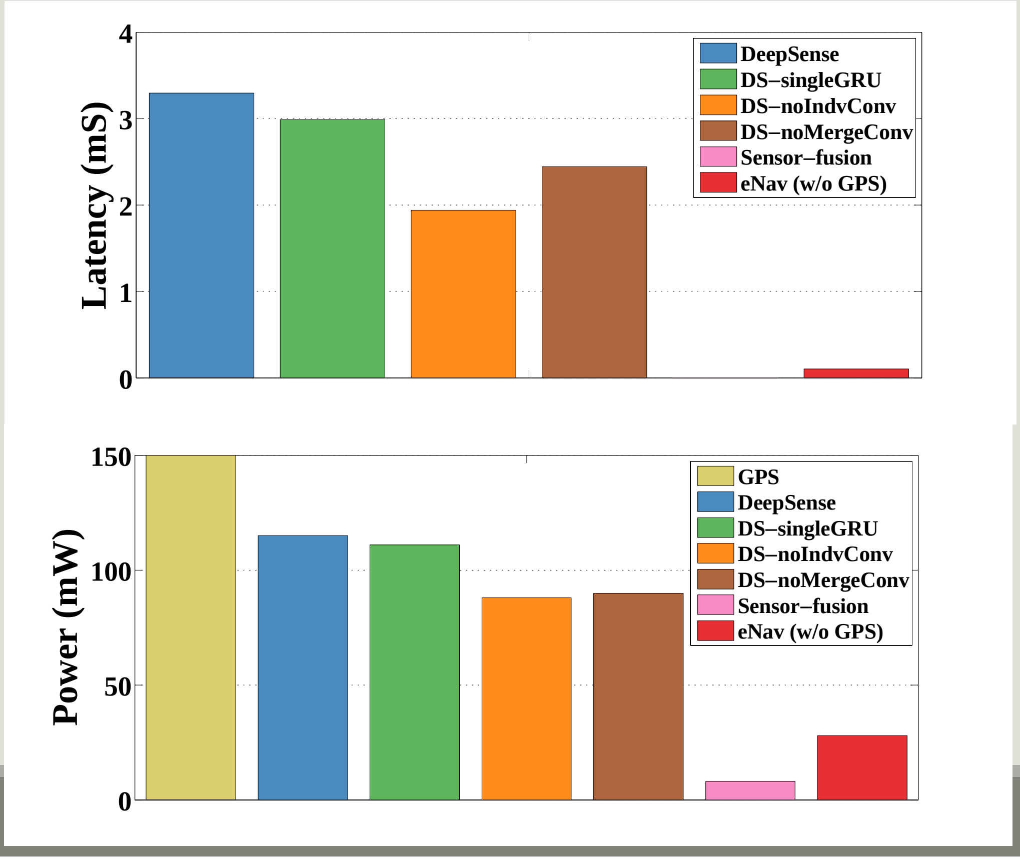




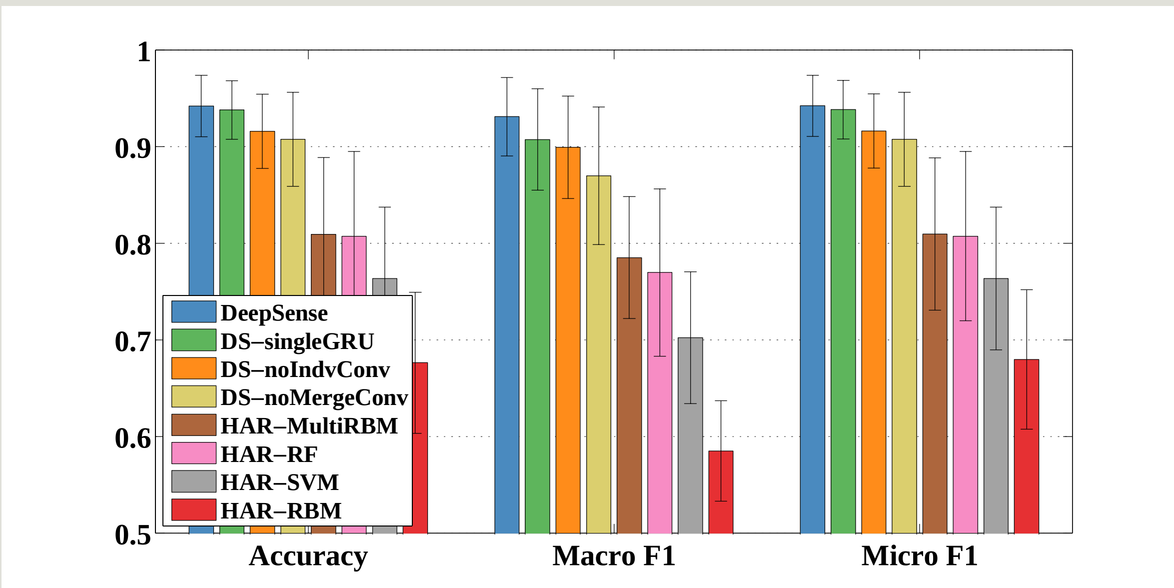


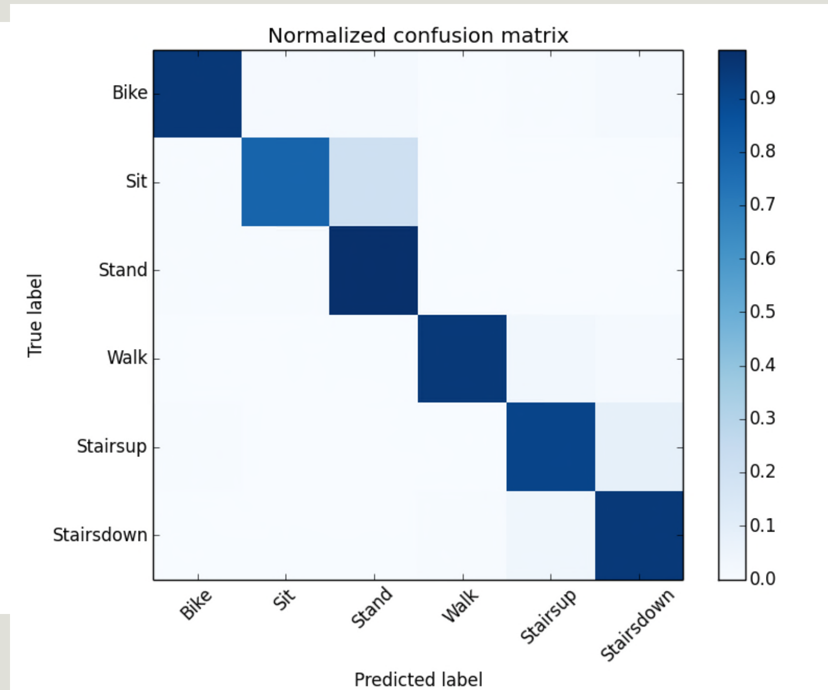


CarTrack: Running time & Energy

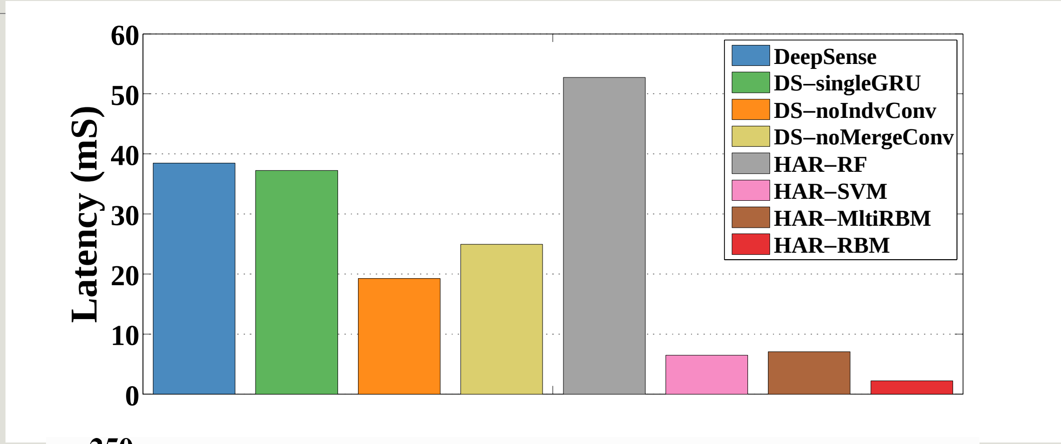


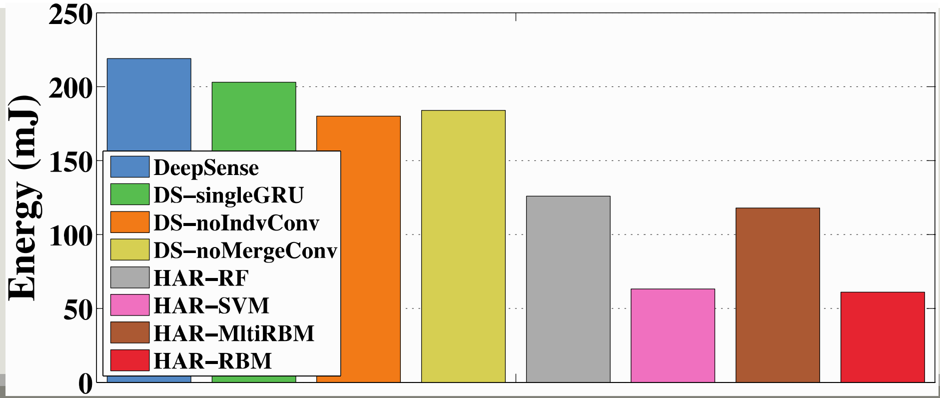
HHAR: Accuracy



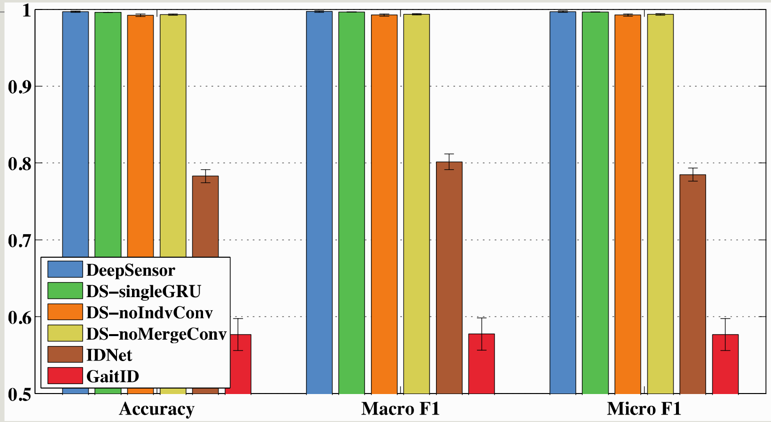


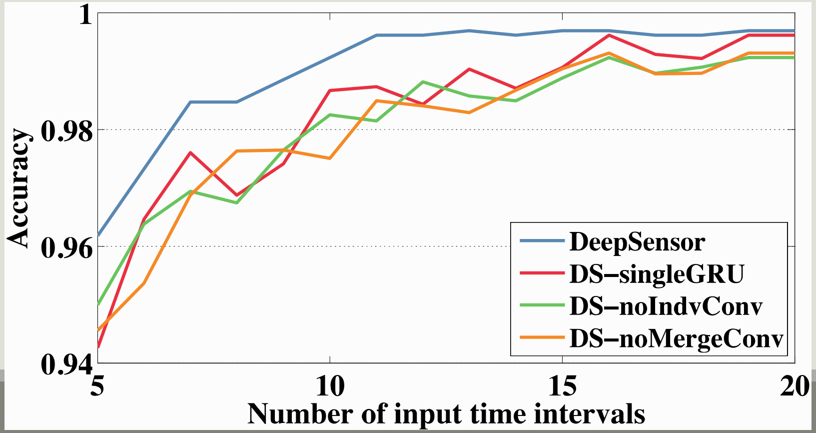
HHAR: Running time & Energy



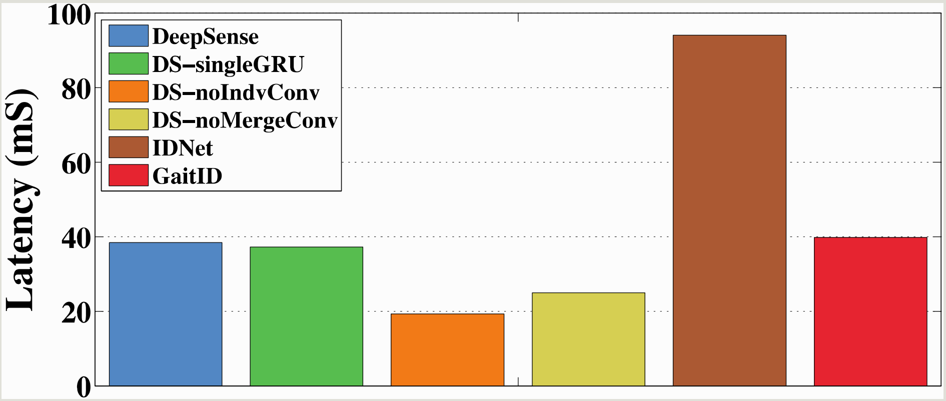


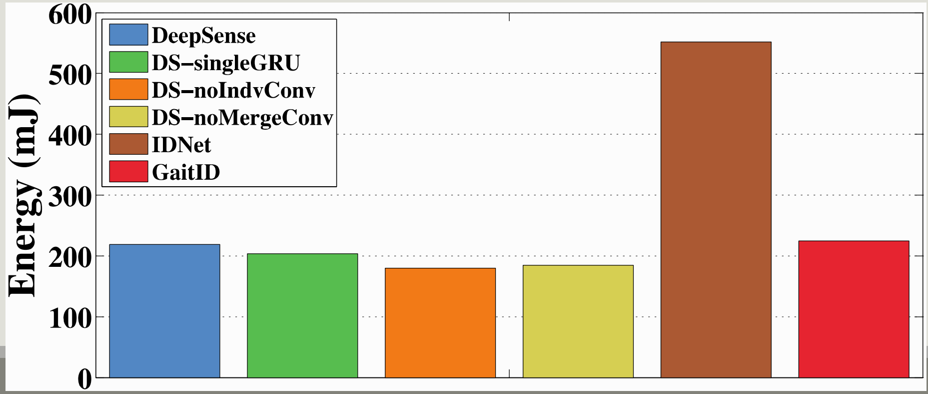
UserID: Accuracy





UserID: Running time & Energy





Code available

https://github.com/yscacaca/DeepSense

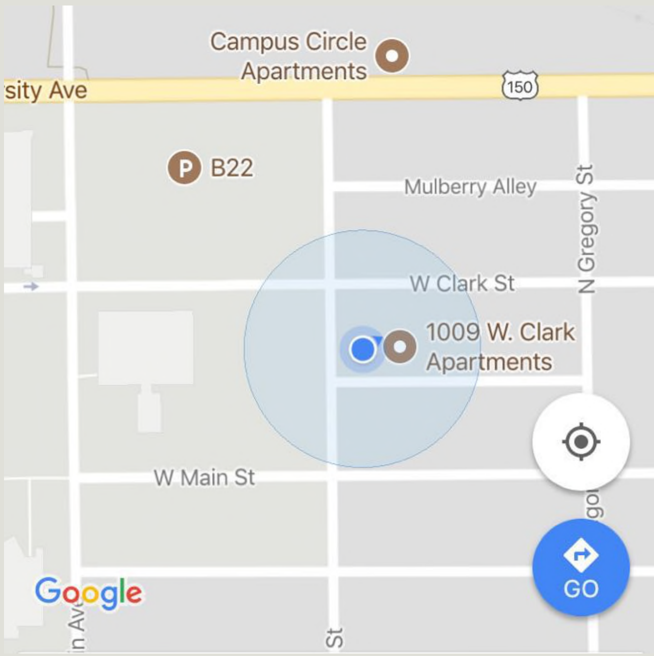
Outline

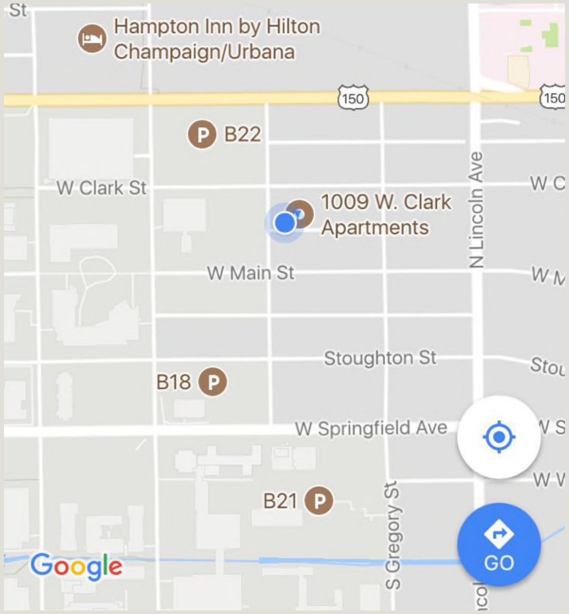
DeepSense: A unified deep learning framework for time-series mobile sensing data processing. (WWW 2017)

RDeepSense: Reliable Deep Mobile Computing Models with Uncertainty Estimations. (Ubicomp 2018)

DeepIoT: Compressing Deep Neural Network Structures for Sensing Systems with a Compressor-Critic Framework. (SenSys 2017)

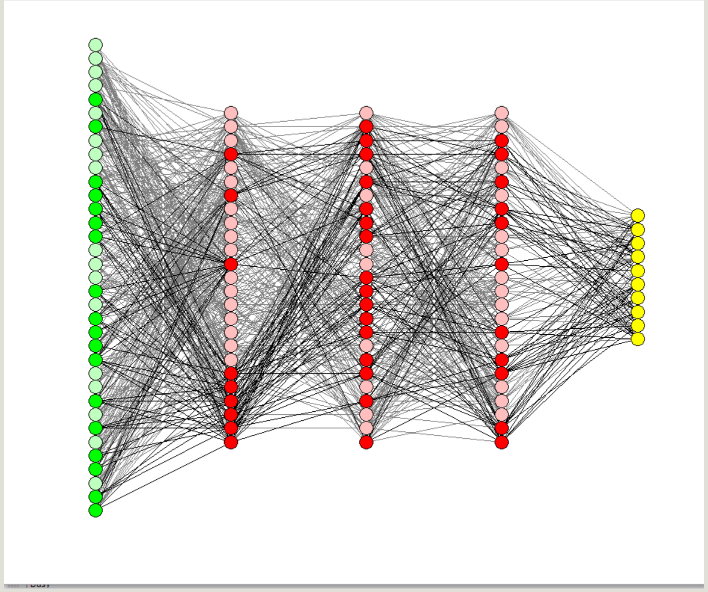
Motivations





Background: Dropout as Bayesian Approximation (MCDrop)

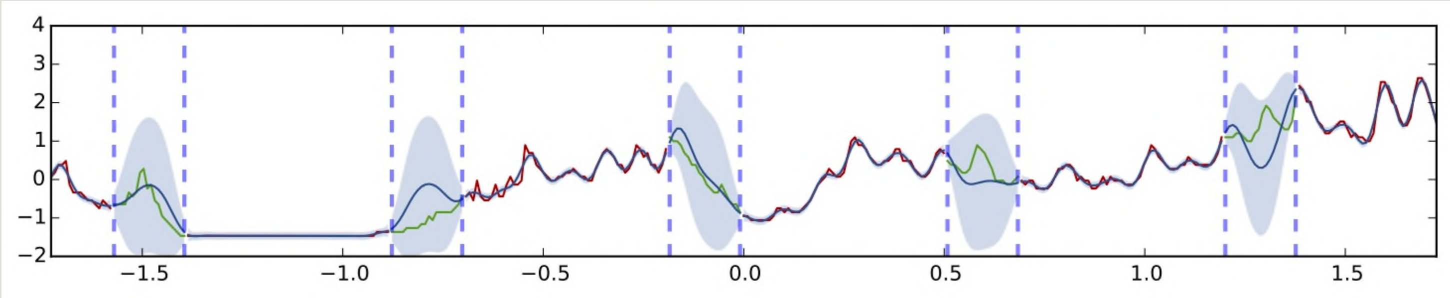
Dropout operation convert a deterministic neural network into a probabilistic neural network



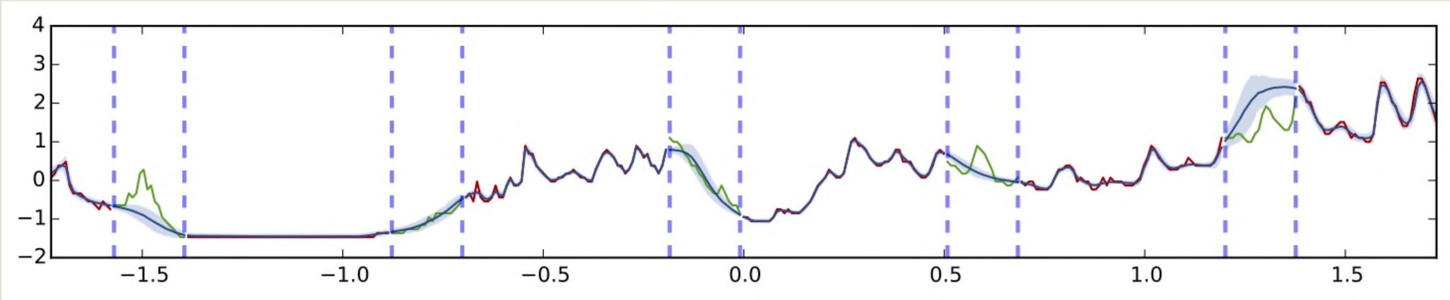
It has been proven that dropout training in deep neural networks is an approximate Bayesian inference in deep Gaussian processes.

Gal, Yarin, and Zoubin Ghahramani. "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning." *international conference on machine learning*. 2016.

Background: Dropout as Bayesian Approximation (Underestimate)



Gaussian process with SE covariance function



Dropout using uncertainty information (5 hidden layers, ReLU non-linearty)

Background: Model Ensemble with Log Likelihood (SSP)

Train neural networks with NLL (Negative Log-Likelihood) instead of MSE (Mean Square Error).

Tend to overestimate.

$$\frac{1}{2}\log \boxed{\sigma^2} + \frac{1}{2\boxed{\sigma^2}}\boxed{(y-\mu)^2}$$

Large at the beginning of training

Enlarging variance at the beginning of training can easily reduce NLL

RDeepSense: Balancing the Bias Variance Tradeoff

Design an objective function that balances the underestimation and overestimation:

$$(1-\alpha)\left(\underbrace{\boxed{\frac{1}{2}\log\sigma^2 + \frac{1}{2\sigma^2}(y-\mu)^2}}_{\text{NLL: overestimate}}\right) + \alpha\underbrace{\boxed{(y-\mu)^2}}_{\text{MSE: underestimate}}$$

Hyper-parameter α balances two effects

Proved that dropout training with this object function was equivalent to a specific deep Gaussian process model.

RDeepSense: Reducing Resource Consumption

Previous works are based on either sampling or ensemble method, which is resource consuming.

Use the test-time dropout operation instead of sampling

$$\widetilde{W_l} = diag(p_l) W_l$$

$$y_l = x_l \widetilde{W_l} + b_l$$

$$x_{l+1} = f(y_l)$$

This is a biased approximation, but works well in evaluations.

RDeepSense: Comparasion

Algorithm	Dropout Training	Proper Scoring Rules	Ensemble method	Obtain predictive uncertainty with single run
RDeepSense	✓	✓	✕	✓
MCDrop	✓	✕(underestimate)	✕	✕
SSP	✕	✓(overestimate)	✓	✕

Evaluation: Hardware

Intel Edison

- Intel Atom SoC dual-core CPU at 500 MHz
- 1GB memory

Run solely on CPU

Evaluation: Dataset

1. BPEst : Monitor cuffless blood pressure through photoplethysmogram from fingertip.

2. NYCommute: Estimate commute time in New York City through the pick-up time and location as well as the drop-off location.

3. GasSen: Estimate real concentration of Ethylene and CO gas mixture from an array of low-end chemical sensors.

Evaluation: Baseline Algorithms

1. MCDrop: This algorithm is based on Monte Carlo dropout. We use MCDrop-k to represent MCDrop with k samples.

2. SSP: Ensemble of multiple neural networks trained with NLL. We use SSP-k to represent SSP by ensemble k individual neural networks.

3. RDeepSense-MC: This algorithm is basically the proposed RDeepSense algorithm. The algorithm uses Monte Carlo sampling instead of our proposed approximation during infernce. We use RDeepSense-MCk to present RDeepSense- MC with k samples .

4. GP: Gaussian process (GP).

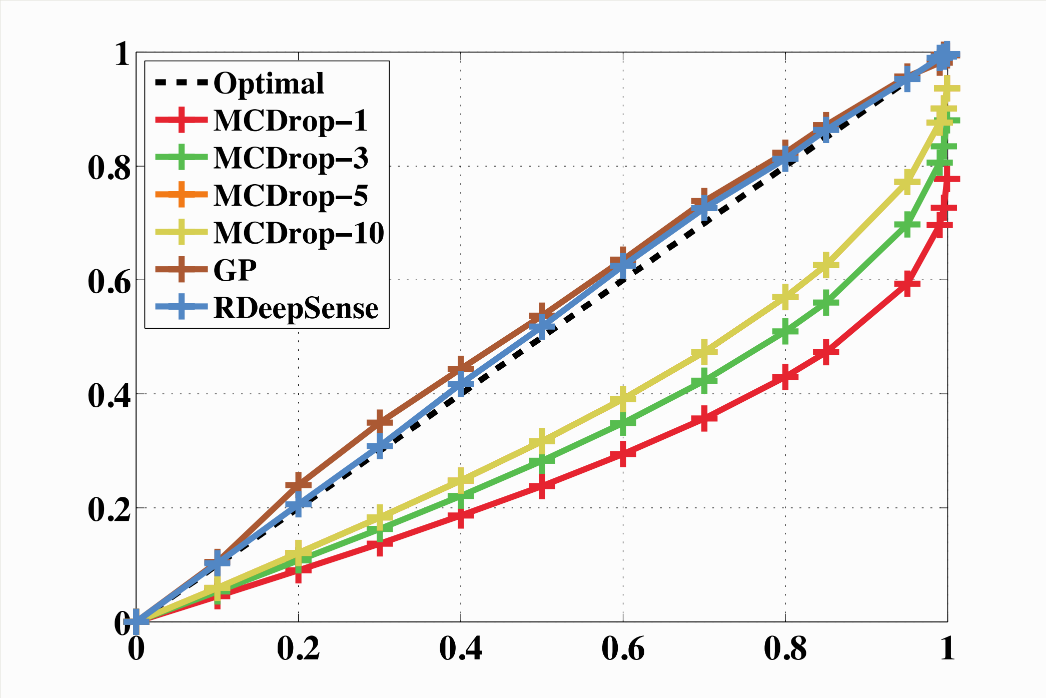
Evaluation: Reliability Diagrams

1. Compute the z% confidence interval for each testing data based on predictive mean and variance of each algorithm.

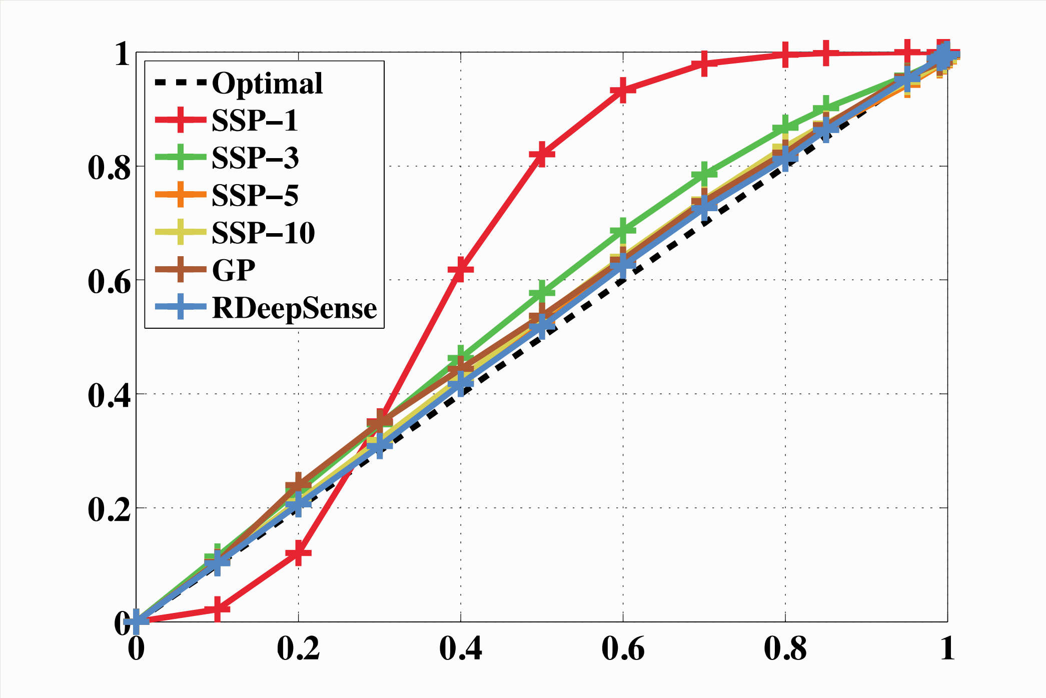
2. Measure the fraction of the testing data that falls into this confidence interval.

3. For a well-calibrated uncertainty estimation, the fraction of testing data that falls into the confidence interval should be similar to z%.

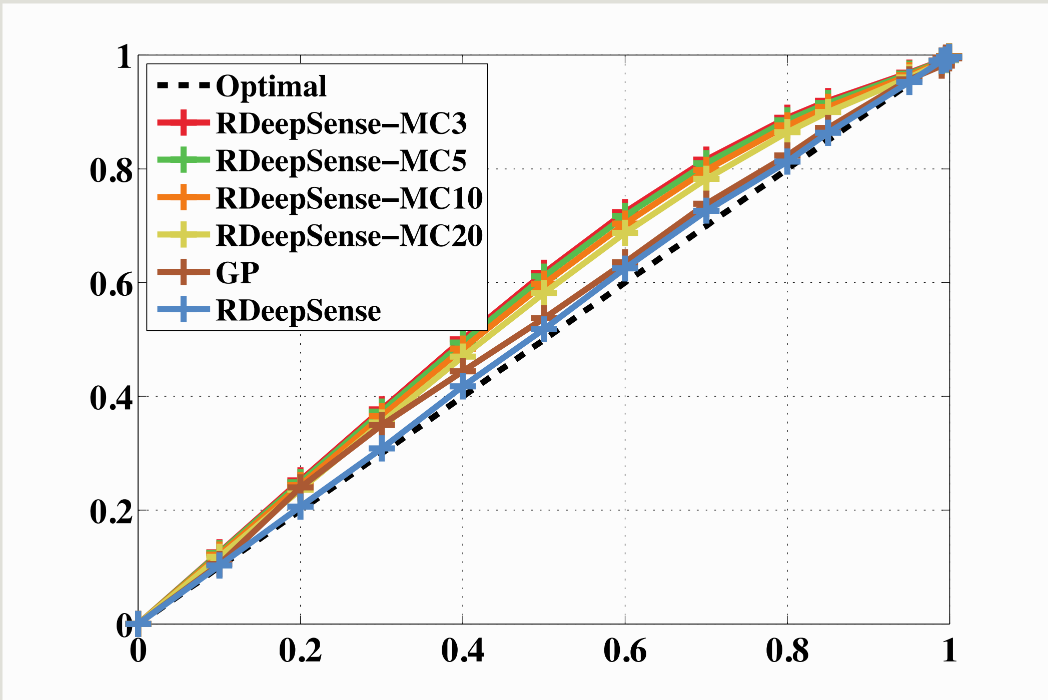
BPEst: MCDrop & RdeepSense



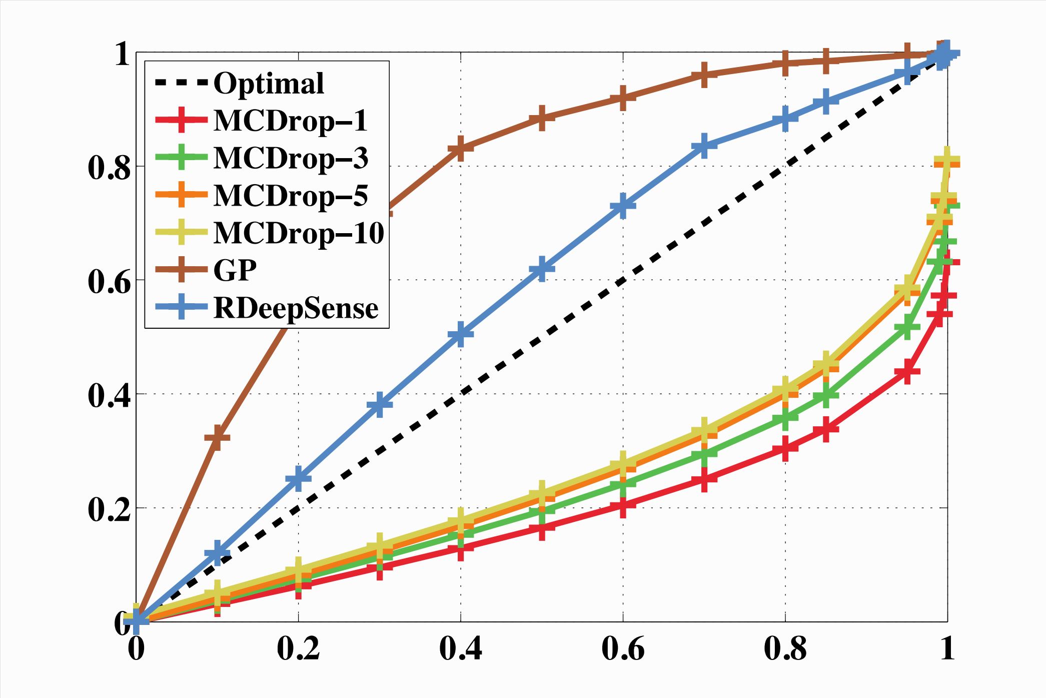
BPEst: SSP & RDeepSense



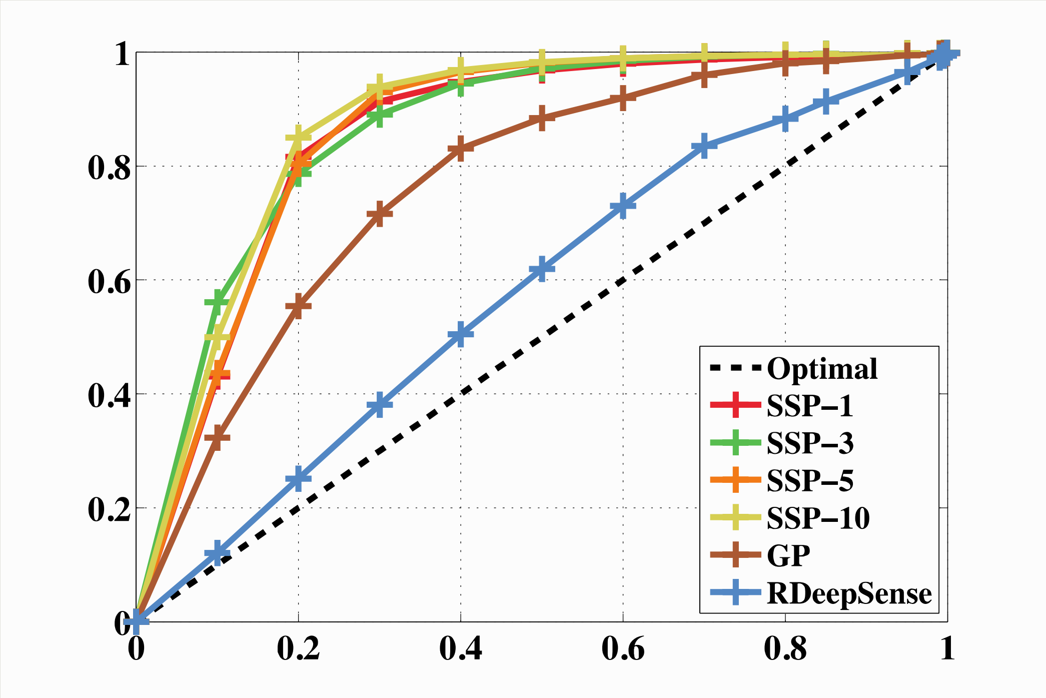
BPEst: RDeepSense



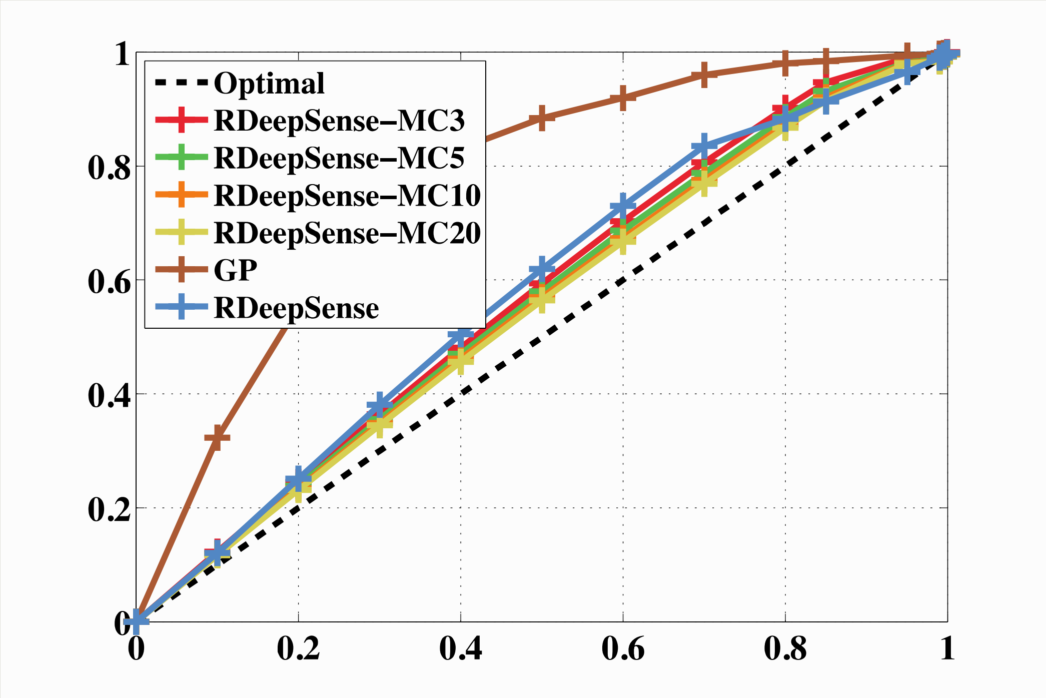
NYCommute: MCDrop & RDeepSense



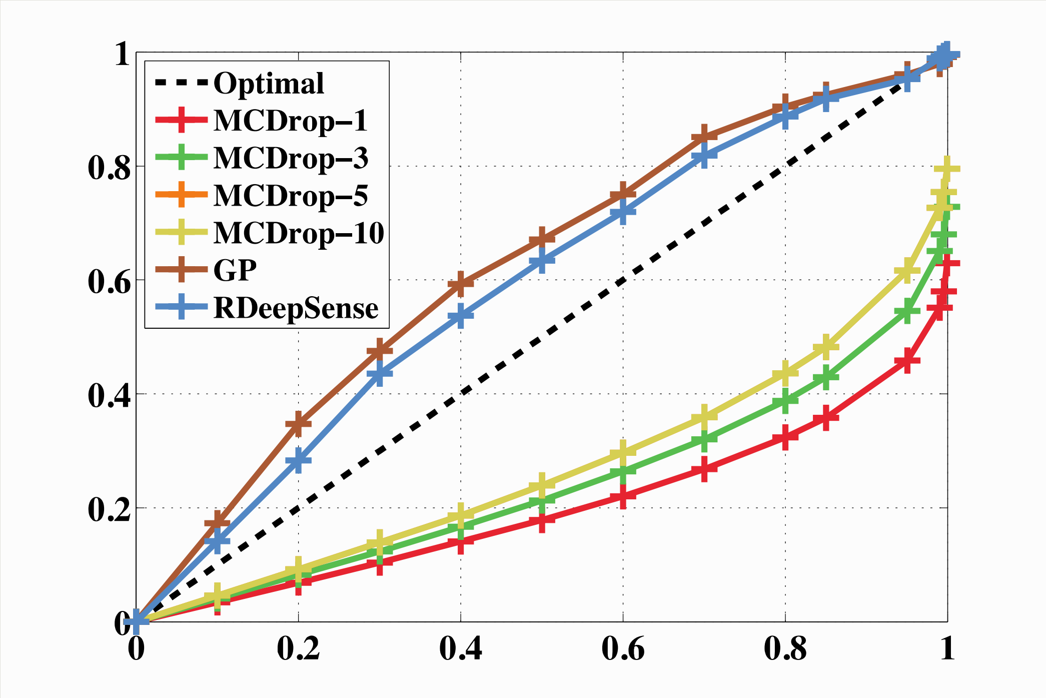
NYCommute: SSP & RDeepSense



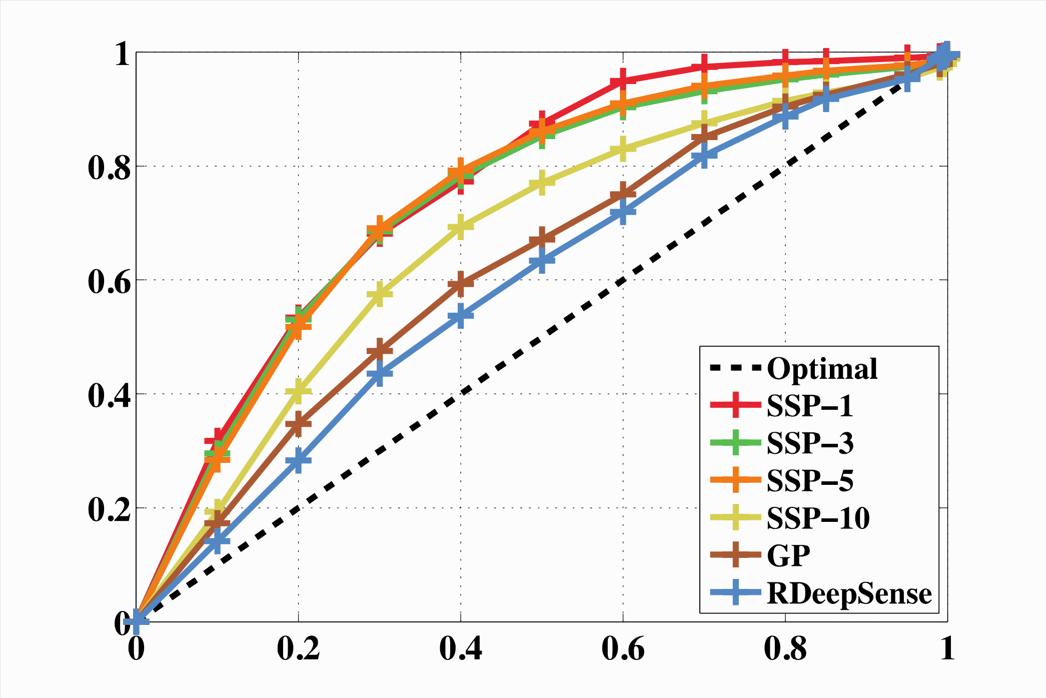
NYCommute: RDeepSense



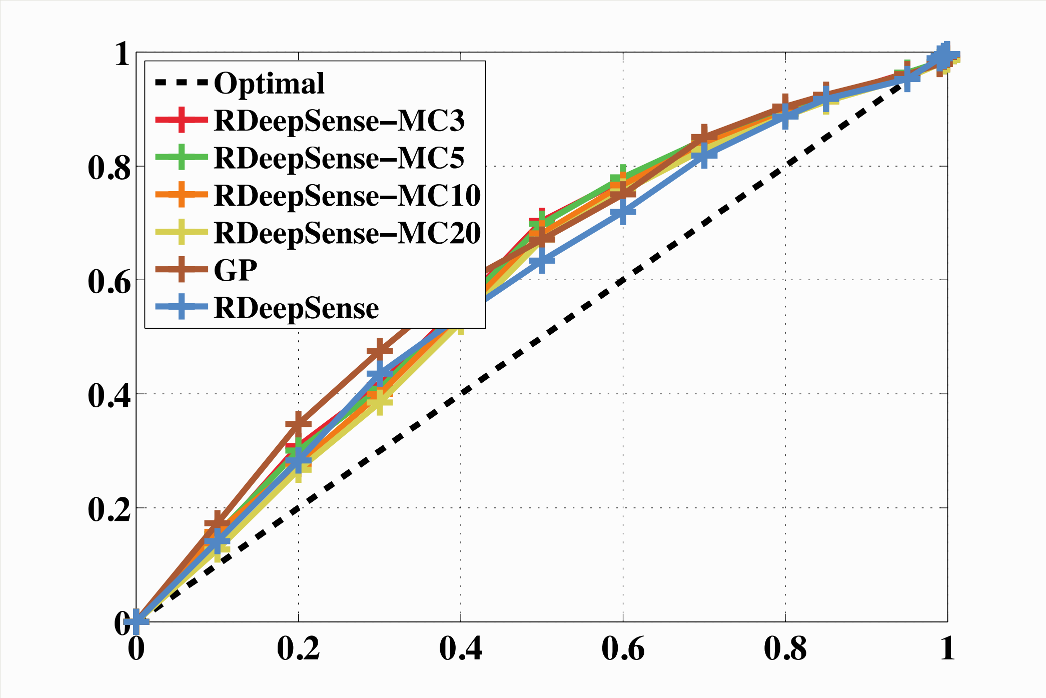
GasSen: MCDrop & RDeepSense



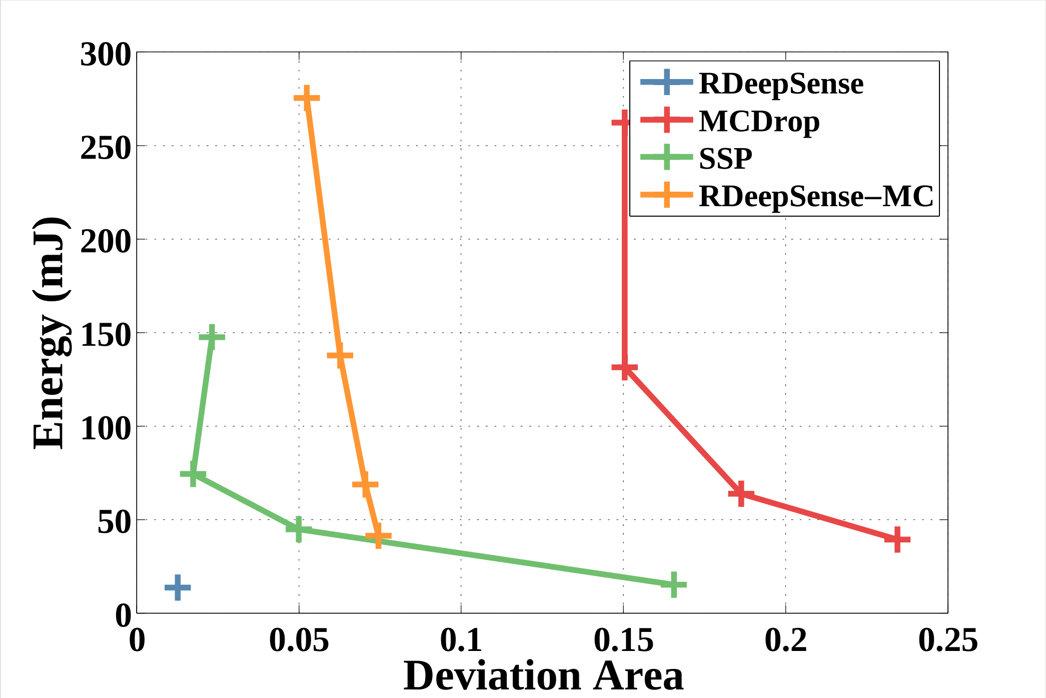
GasSen: SSP & RDeepSense



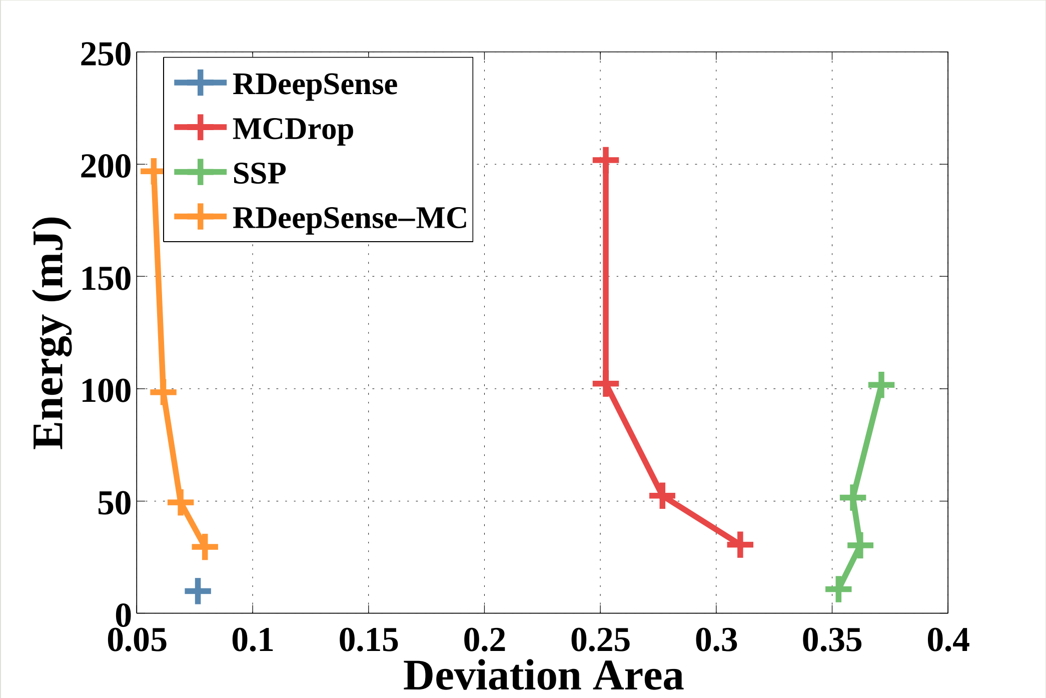
GasSen: RDeepSense



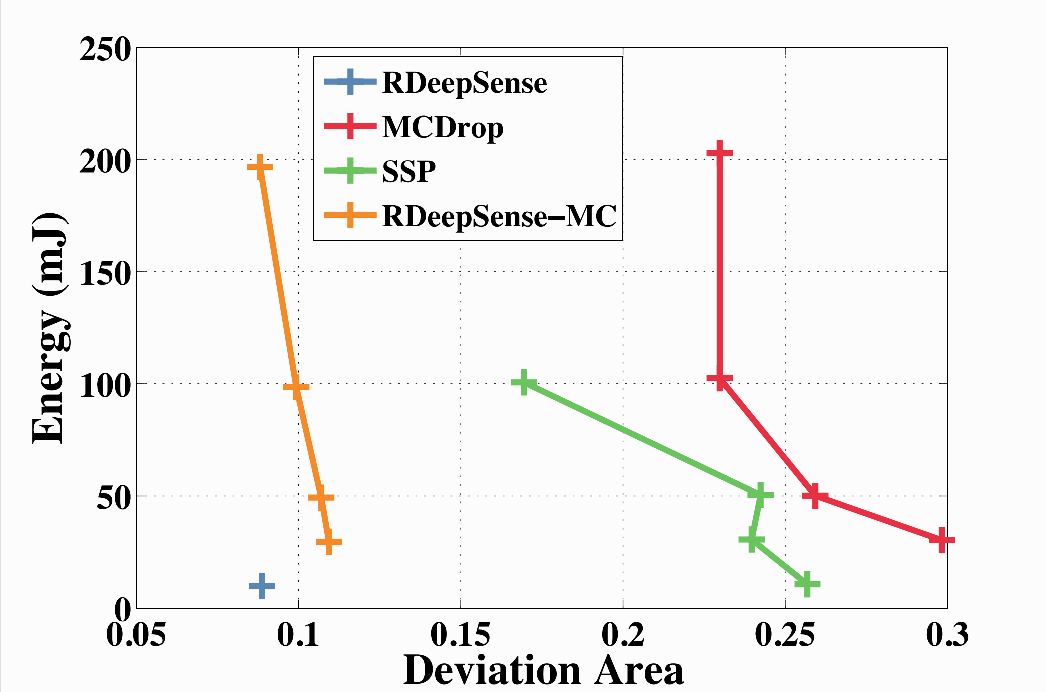
Energy Consumption: BPEst



Energy Consumption: NYCommute



Energy Consumption: GasSen



Effect of hyper-parameter α