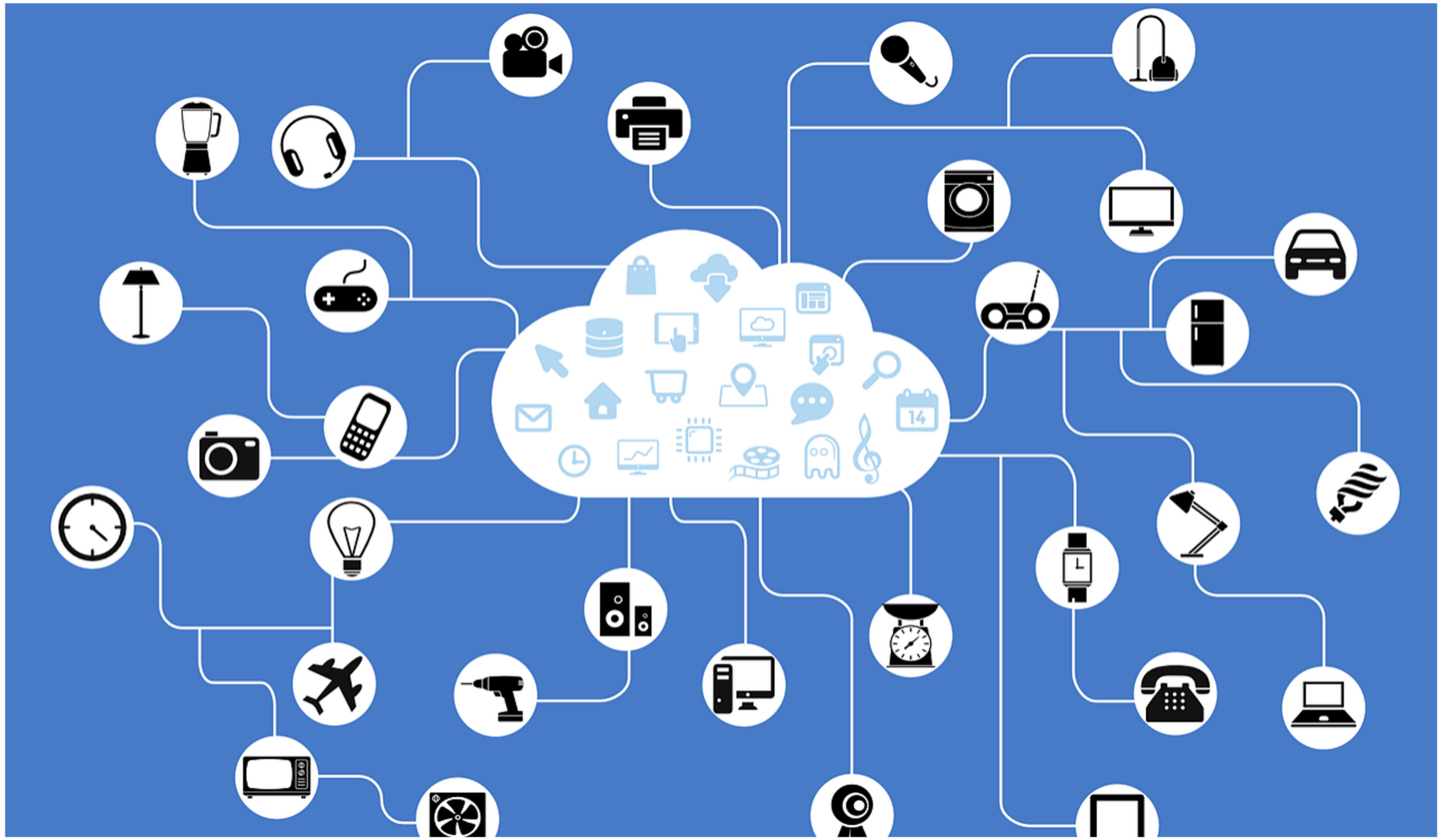
Handling Real-time Unlabeled Data in Activity Recognition On-Device using Deep Bayesian Active Learning and Data Programming

Gautham Krishna Gudur
Ericsson, India

Prahalathan Sundaramoorthy
University of Southern California

Venkatesh Umaashankar
Ericsson Research, India

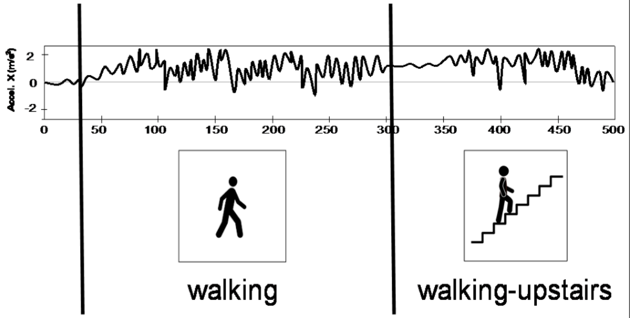


WEARABLE/ MOBI-QUITOUS COMPUTING

- Expansive growth of usage of mobile phones, smartwatches across various users.
- Significant research in the field of ubiquitous & wearable computing.
- Data from sensors embedded in wearables conveniently provide a way to extract contextual, behavioural information of users.

Applications particularly gaining importance in fields such as health-care and fitness tracking are

- **Human Activity Recognition (HAR)**
- **Fall Detection**



DEEP LEARNING FOR HAR

ALLEVIATES THE PROBLEM OF CRAFTING SHALLOW HAND-PICKED FEATURES

AUTOMATICALLY EXTRACTS DISCRIMINATIVE FEATURES

DOES NOT REQUIRE EXTENSIVE DOMAIN KNOWLEDGE

ENHANCES SCALABILITY AND GENERALIZABILITY

PROMINENT CHALLENGES IN ON-DEVICE HAR

1. On-Device Incremental Learning

- Model updation incrementally
- Facilitation of User Adaptability
- Complex deep architectures generally have high computational overheads, hence difficult to update models on-device

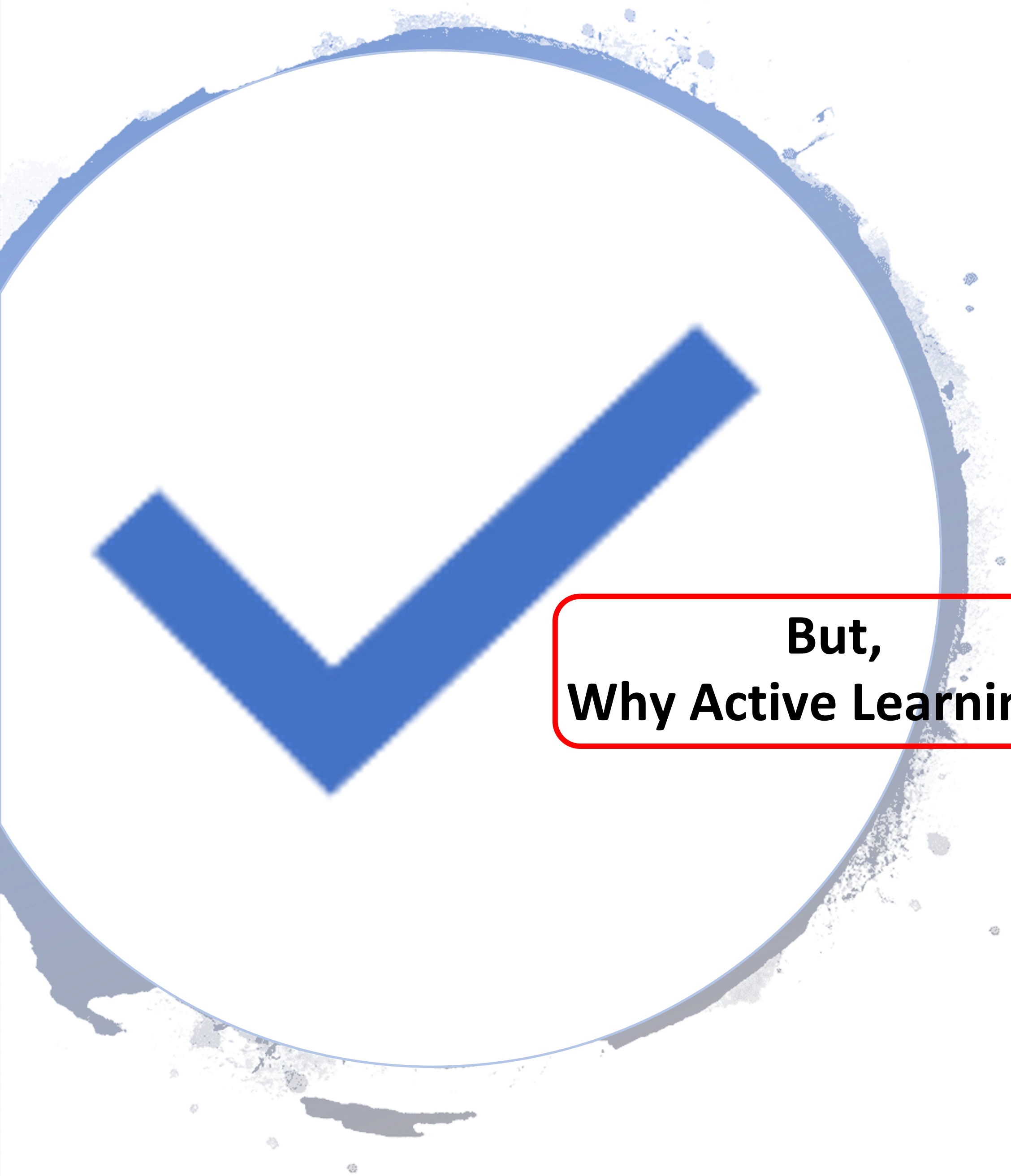
PROMINENT CHALLENGES IN ON-DEVICE HAR

2. Label Acquisition during Incremental Learning

- Real-time acquisition of labels (ground truthing) is hard
- Labelling load on oracle (user) needs to be reduced

GOALS OF OUR PROPOSED SYSTEM

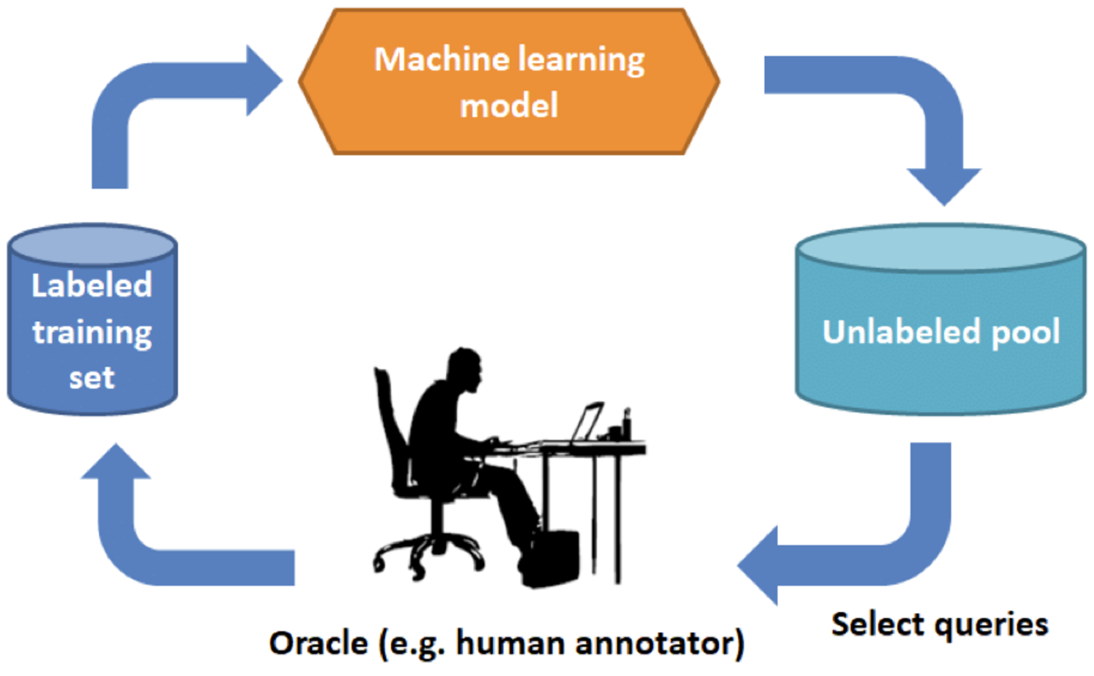
- A generic HAR model which handles **Incremental Learning** on wearables, and is **resource-friendly**
- **Active Learning**, which queries the oracle only necessary (most-informative) labels on-device
- Facilitate **User Adaptability**
- Test the generalizing Incremental Active Learning capabilities together on **HAR** and **Fall Detection** tasks



But,
Why Active Learning?

GOALS OF OUR PROPOSED SYSTEM

- A generic HAR model which handles **Incremental Learning** on wearables, and is **resource-friendly**
- **Active Learning**, which queries the oracle only necessary (most-informative) labels on-device
- Facilitate **User Adaptability**
- Test the generalizing Incremental Active Learning capabilities together on **HAR** and **Fall Detection** tasks



- A big challenge in many applications is obtaining labelled data.
- **Active Learning (AL), over unsupervised techniques, can be used predominantly to substantiate the confidence on the queried data points.**
- Instead of labeling hundreds of activities, an ideal system should query few labels in each activity.

ACTIVE LEARNING

BAYESIAN NEURAL NETS (BNNs)

- Offer a probabilistic interpretation to deep learning models.
- Incorporate Gaussian prior (probability distributions p(ω)) over our model parameters ω.
- Can possess and **model uncertainty information**.

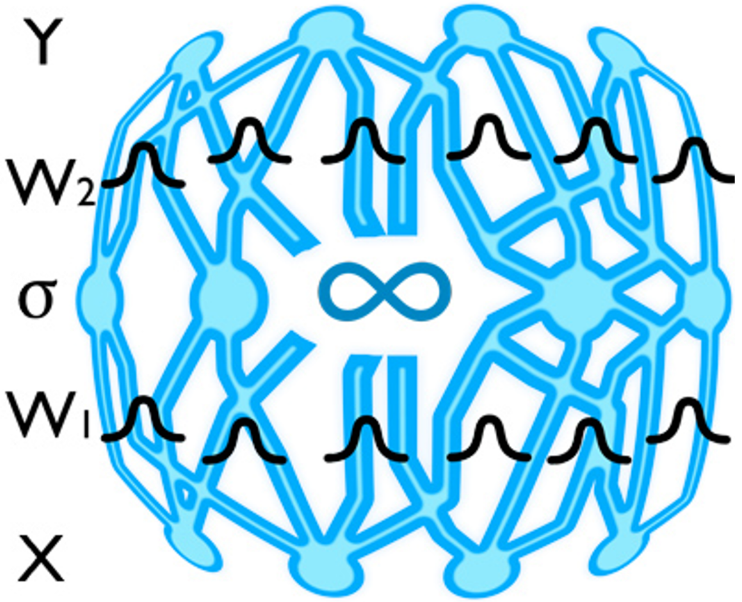


Fig. taken from Prof. Yarin Gal's blog

MODELING UNCERTAINTIES USING DROPOUT

- **Dropout** - a stochastic regularization technique can perform approximate inference over a deep Gaussian process
- Learns the model posterior uncertainties **without high computational complexities** over few stochastic iterations at both train/test times
- Termed Monte-Carlo Dropout **(MC-Dropout)**
- Equivalent to performing Variational Inference
- $p(y* | x*, D_{train}) = \int p(y* | x*, \omega)\ p(\omega | D_{train})\ d\omega$

[Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning, ICML '16]

MODELING UNCERTAINTIES USING DROPOUT

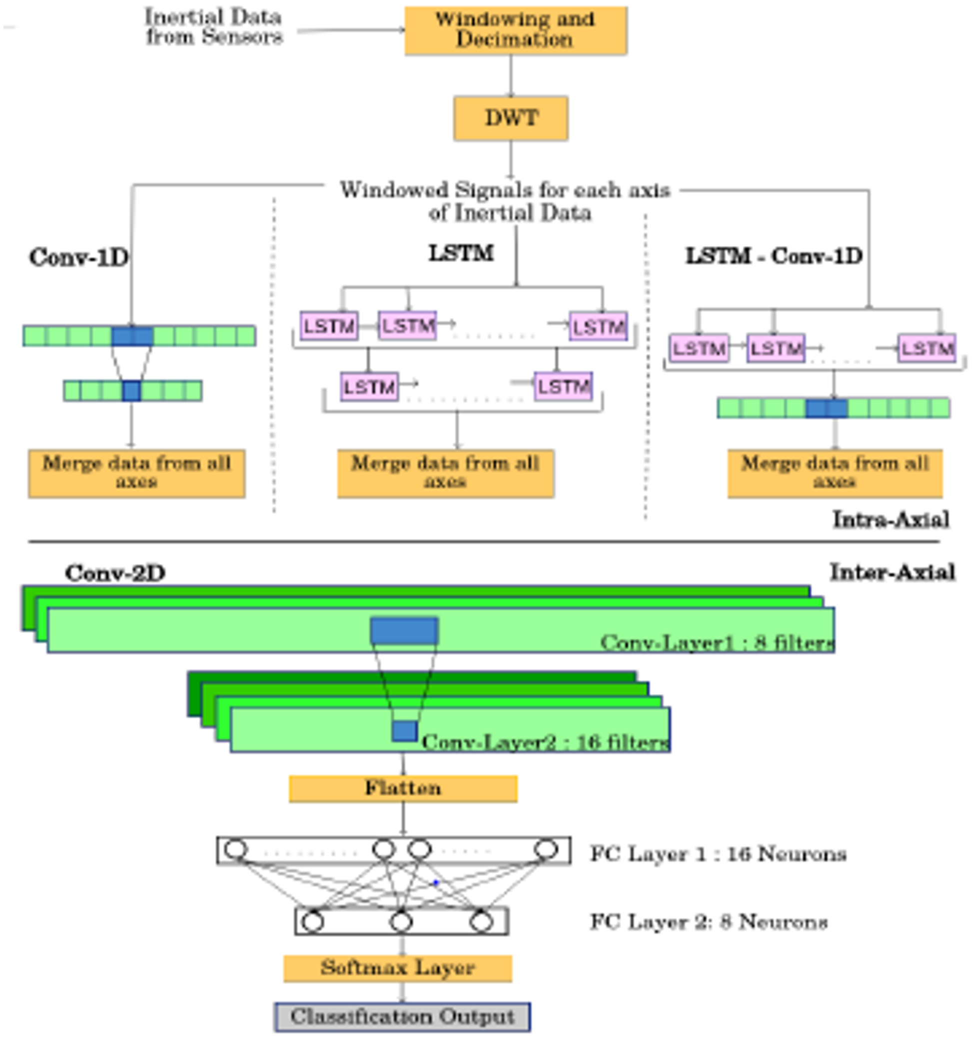
- **Dropout** - a stochastic regularization technique can perform approximate inference over a deep Gaussian process
- Learns the model posterior uncertainties **without high computational complexities** over few stochastic iterations at both train/test times
- Termed Monte-Carlo Dropout **(MC-Dropout)**
- Equivalent to performing Variational Inference
- $p(y* | x*, D_{train}) = \int p(y* | x*, \omega)\, \boxed{p(\omega | D_{train})}\, d\omega$

Posterior

[*Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning, ICML '16*]

HARNet ARCHITECTURE

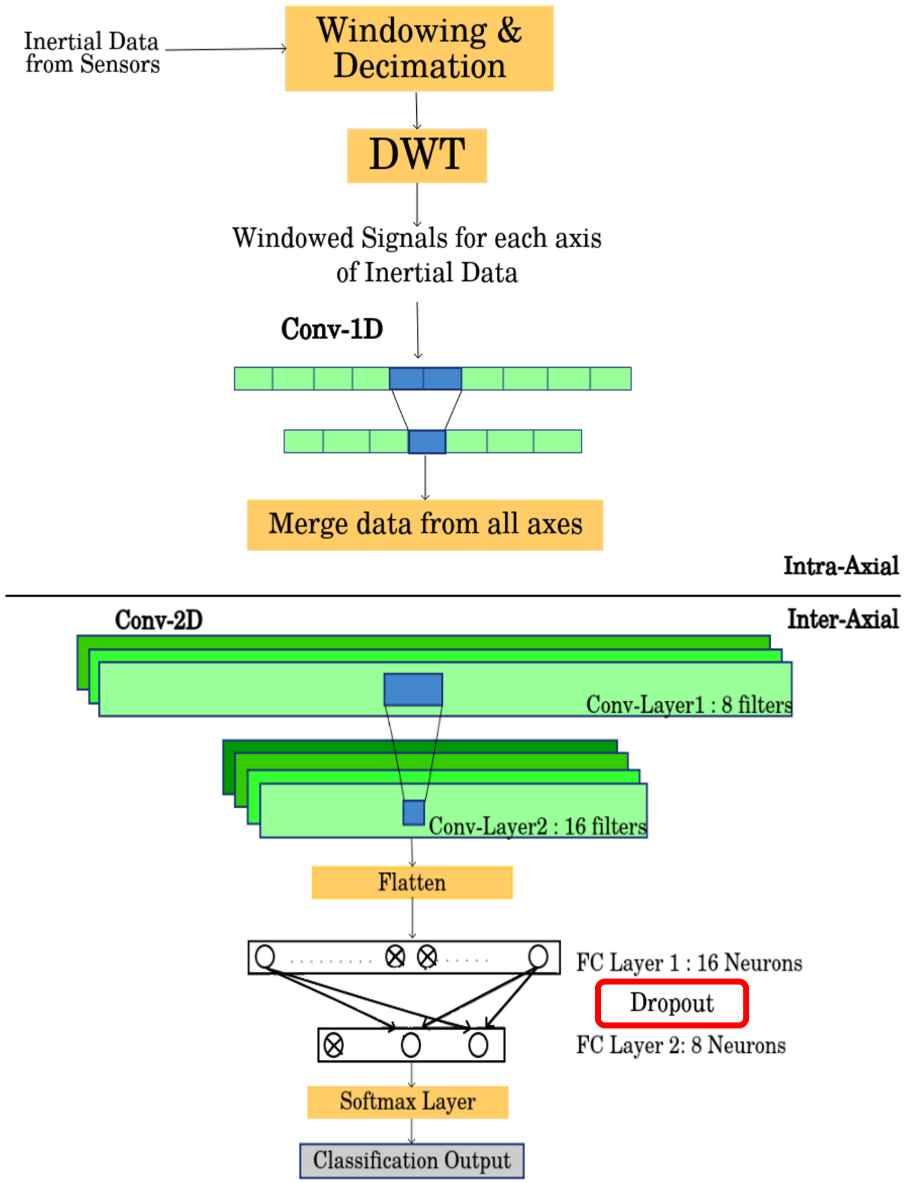
- State-of-the-art architecture for modeling user-independent heterogeneous HAR tasks – three different variants with intra-axial and inter-axial ensemble of convolutional and recurrent layers.
- Intra-axial variants can model correlations and information within each axis of the sensor (accelerometer, gyroscope, etc.)
  - *HAR-CNet* - Two stacked Conv-1Ds
  - *HAR-LNet* - Two stacked LSTMs
  - *HAR-LCNet* - Stacked LSTM and Conv-1D
- Inter-axial variants can model spatial interactions between axes.
  - Two layer stacked Conv-2D
  - Two Fully-Connected layers
- *HAR-CNet* is ~7x faster than *HAR-LCNet* and ~85x faster than *HAR-LNet*, with just ~1% difference in accuracies, hence benchmarked *HAR-CNet*.
- Preprocessing techniques used:
  - Windowing,
  - Decimation (down-sampling to normalize sampling rates across different devices)
  - Discrete Wavelet Transform (DWT)
- Robust across new users (User Adaptability), and resource-efficient with just ~31,000 parameters, supports Incremental Learning.



[HARNet: Towards On-Device Incremental Learning using Deep Ensembles on Constrained Devices, EMDL '18]()

BAYESIAN HARNet ARCHITECTURE

- Utilize *HARNet* architecture, and treat it as a Bayesian Neural Net (with Dropout).
- Intra-Axial and Inter-Axial dependencies exploited using stacked Conv-1D and Conv-2D architectures.
- Pre-processing techniques – Windowing, Decimation (down-sampling) and Discrete Wavelet Transform (DWT).
- Conv-1D to extract characteristics within each axis (X, Y, Z of accelerometer data).
- Conv-2D to capture interactions between data from three axes, thereby learning discriminative features across spatial dimensions.

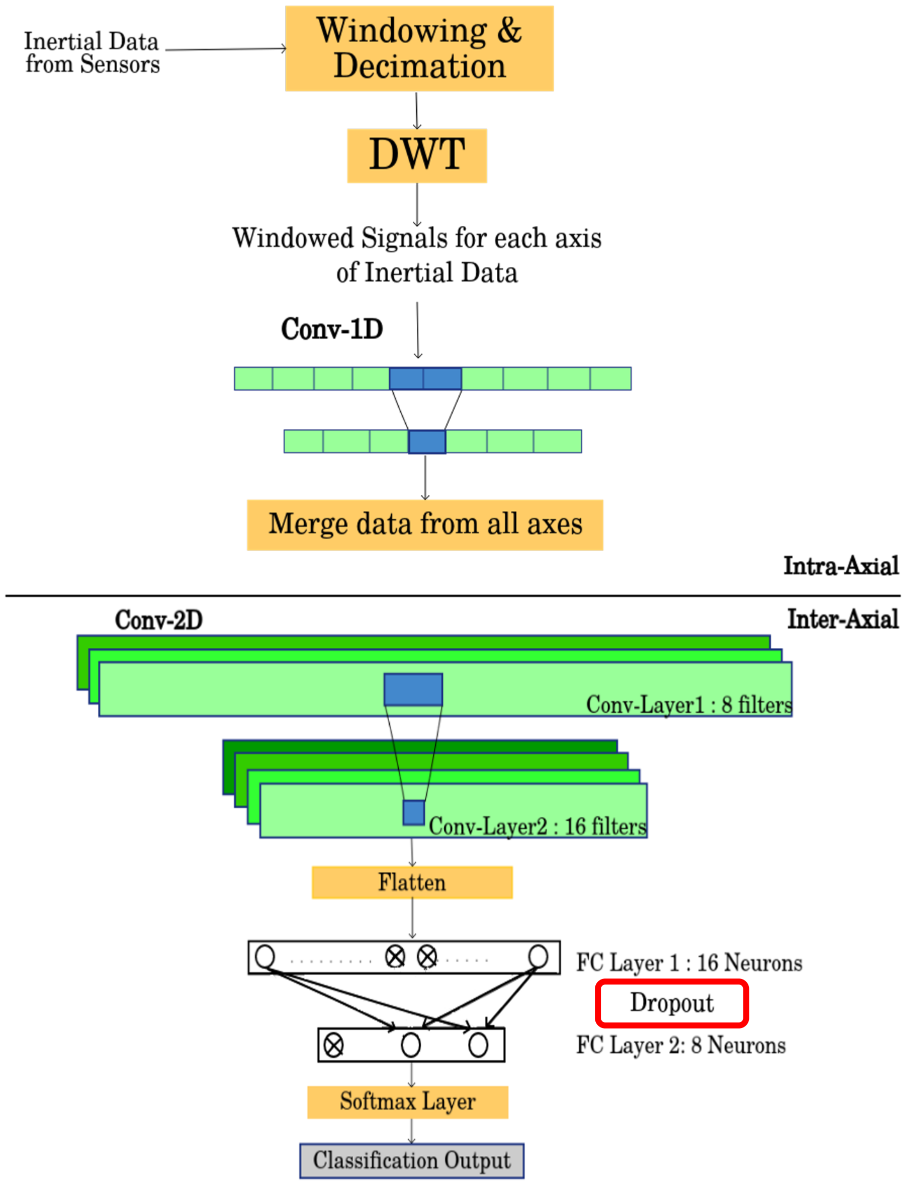


BAYESIAN HARNet ARCHITECTURE

- Two stacked Conv-1D layers with 8 & 16 filters each size 2, BatchNorm, Max-Pool size 2 (Intra-axial)
- Two stacked Conv-2D layers with 8 & 16 filters each size 3x3, BatchNorm, Max-Pool size 3x2 (Inter-axial)
- Two Fully-Connected Layers with 16 & 8 neurons each and ReLU activations.
- Dropout drop probability of 0.3.
- Softmax Layer to estimate probability scores
- Categorical cross-entropy loss with Adam Optimizer

Refer paper for more details:

[*ActiveHARNet: Towards On-Device Deep Bayesian Active Learning for Human Activity Recognition, EMDL '19*]



ACQUISITION FUNCTIONS

- Uncertainty measures from Bayesian *HARNet* need to be quantified
- Arriving at most efficient set of data points (select k from n) to query from D_{pool}

ACQUISITION FUNCTIONS

Given incoming data point x and unknown label y with data D and parameters ω,

- *Max Entropy:* Maximize predictive entropy

  $H[y|x,D] := -\sum_c p(y = c|x,D) \log p(y = c|x,D)$ c

- *BALD (Bayesian Active Learning by Disagreement):* Maximize mutual information between predictions and model posterior

  $I[y,\omega|x,D] = H[y|x,D] - E_{p(\omega|D)} H[y|x,\omega]$

- Maximize *Variation Ratios:*

  variation-ratio[x] := $1 - \max p(y|x,D)$ y

- *Random Acquisitions:* Select data points from pool uniformly at random.

DATASETS USED

Heterogeneous Human Activity Recognition (HHAR) Smartwatch Dataset

Smart Devices are Different: Assessing and Mitigating Mobile Sensing Heterogeneities for Activity Recognition, SenSys ‘15

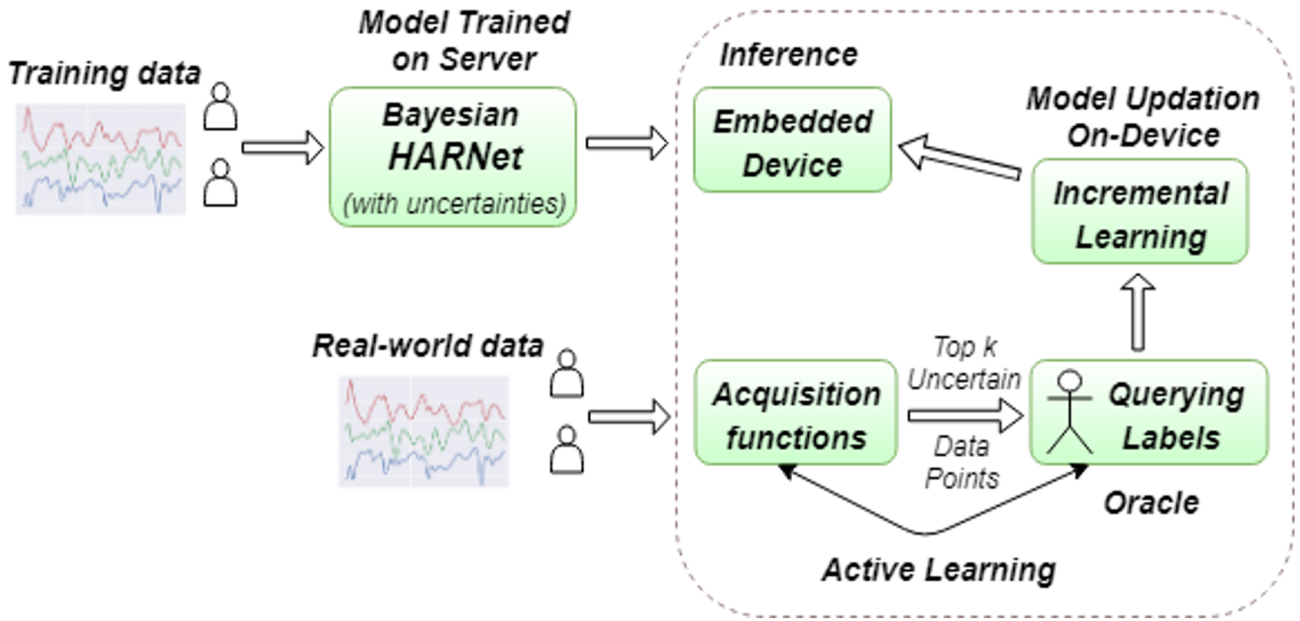
- Utilizing accelerometer data from different wearables - two LG G smartwatches and two Samsung Galaxy Gears across nine users performing six activities: Biking, Sitting, Standing, Walking, Stairs-Up, Stairs-Down in real- time heterogeneous conditions.

Notch Wrist-worn Fall Detection Dataset

Smartfall: A smartwatch-based fall detection system using deep learning, Sensors ’18

- Utilizing wrist-worn accelerometer data from an off-the-shelf Notch sensor by seven volunteers across various age groups performing simulated falls and activities (activities are termed as not-falls) .

ActiveHARNet ARCHITECTURE



[*ActiveHARNet: Towards On-Device Deep Bayesian Active Learning for Human Activity Recognition, EMDL '19*]()

- User-Independent Incremental Active Learning is experimented on Raspberry Pi 2 (similar H/W, S/W with predominant contemporary wearables), with the trained model weights being stocked.
- The number of acquisition pool windows used for incremental active training can be governed by the acquisition adaptation factor η ∈ [0, 1].

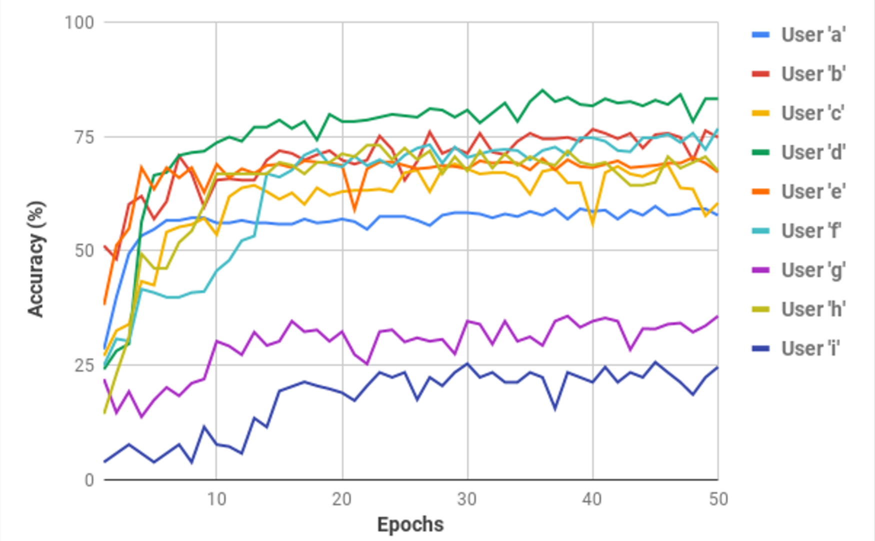
BASELINE EFFICIENCIES using Bayesian *HARNet*

- A stratified k-fold *Leave-User-Out* (testing on previously unseen users) cross validation technique was used for evaluating User Adaptability.
- Unseen user data split into test and pool data, pool treated as real-world data, test is untouched.

HHAR

User ’d’ – 84%; User ’g’ – 36%; User ‘i’ - 25%

Average – 61%



Notch

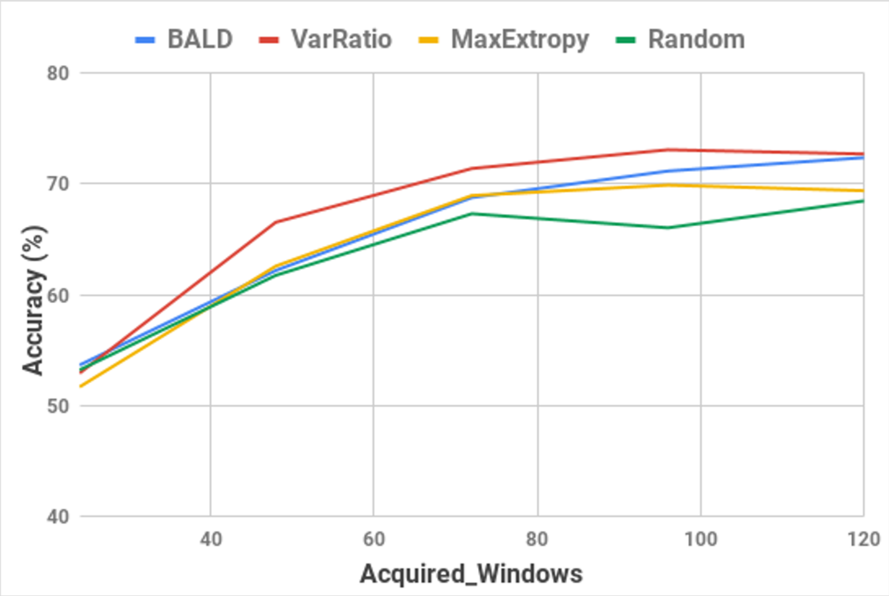
Average f1 – 0.927

f1-score used since fall is a very rare-class

	User 1	User 2	User 3	User 4	User 5	User 6	User 7
f1-score	0.9326	0.9214	0.9357	0.9372	0.9195	0.9229	0.9248
Accuracy	97.02	94.44	94.05	95.36	94.08	94.59	94.65

ActiveHARNet on HHAR

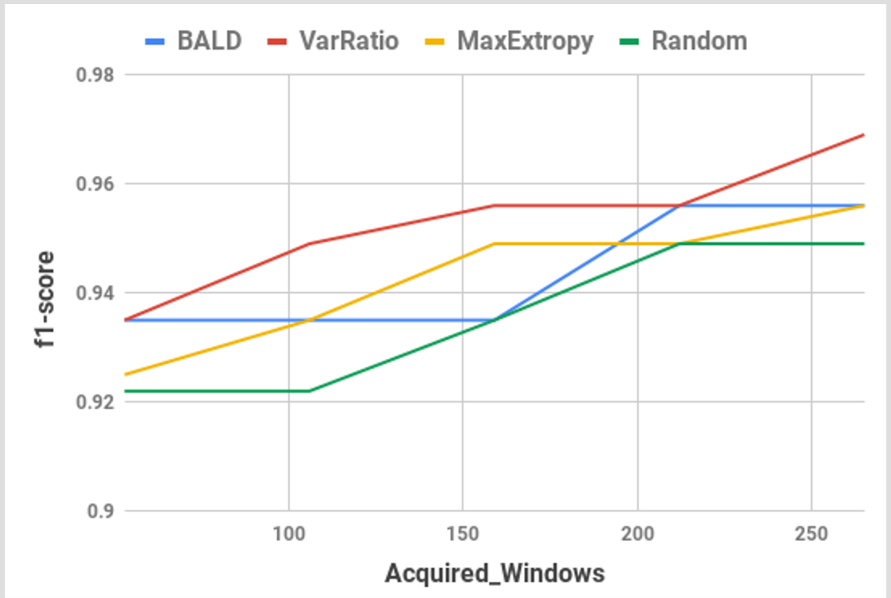
- Variation Ratios (VR) acquisition function performs the best.
- User 'i' (least performing) – accuracy increase from 25% - 70% with just ~60 pool points. ~49% (η=0.49 - 60 pool points) of total 123 data points gives this 45% accuracy increase. With all 123 data points (100% - η=1.0), gives 73% accuracy.
- All users average: 61% (η=0) to 86% (η=1) for VR. η=0.4 gives near-equal 85.87%.



η	User a	User b	User c	User d	User e	User f	User g	User h	User i	Avg.
0.0	57.83	74.86	60.5	83.79	67.25	76.77	35.78	67.5	24.66	61
0.2	83.52	89.76	75.7	91.95	81.53	79.79	73.39	78.75	52.92	78.59
0.4	89.15	91.72	80.85	92.3	85.05	84.57	76.23	81	66.53	83.05
0.6	91.55	92.18	82.26	93.26	87.92	86.96	77.15	83.5	71.38	85.13
0.8	92.64	93.24	82.28	93.56	87.52	88.07	78.58	82.6	73.07	85.73
1.0	92.72	93.16	85.06	93.64	89.95	87.96	76.23	81.375	72.69	85.87

ActiveHARNet on Notch

- Variation Ratios (VR) acquisition function again performs the best here.
- User 5 (least performing) – f1-score increases from ~0.92 - 0.95 with just 150 pool points (η=0.4). With all 265 data points (100% - η=1.0), gives 0.969 f1-score.
- All users average: 0.928 (η=0) to 0.943 (η=0.4) and to 0.948 (η=0.6) for VR.



η	User 1	User 2	User 3	User 4	User 5	User 6	User 7	Avg.
0.0	0.932	0.921	0.936	0.937	0.92	0.923	0.925	0.928
0.2	0.938	0.924	0.945	0.947	0.935	0.932	0.925	0.935
0.4	0.943	0.929	0.961	0.952	0.949	0.932	0.932	0.943
0.6	0.949	0.929	0.965	0.952	0.956	0.945	0.936	0.948
0.8	0.943	0.937	0.968	0.965	0.956	0.953	0.942	0.952
1.0	0.952	0.937	0.965	0.956	0.969	0.945	0.936	0.951

INCREMENTAL ACTIVE LEARNING

Process	HHAR	Notch
Inference time	14 ms	11 ms
Discrete Wavelet Transform	0.5 ms	0.39 ms
Decimation	3.4 ms	–
Time taken per epoch	1.8 sec	1.2 sec

- HHAR takes a model size of 315 kB, Notch takes 180kB.
- T=10 stochastic dropout iterations (1.4 sec per iteration) were used, hence total ~ 14 seconds for Bayesian Active Learning.
- Number of data points collected for each acquisition iteration can be bounded based on **time** or **acquired count (number of data points)** criterion.
- **Time** is proposed as preferred metric, i.e., periodic updates at fixed intervals – since oracle would only be able to remember recent trends in activities.
- Cannot guarantee users to perform activities within given time frame, hence thresholding based on count of data points is not recommended.

WHAT IF WE COULD GENERATE LABELS AUTOMATICALLY FROM SCRATCH USING JUST SIMPLE HEURISTIC KNOWLEDGE?

WHAT IF WE COULD GENERATE LABELS AUTOMATICALLY FROM SCRATCH USING JUST SIMPLE HEURISTIC KNOWLEDGE?

DATA PROGRAMMING

AN AUTOMATIC DATA LABELLING PARADIGM

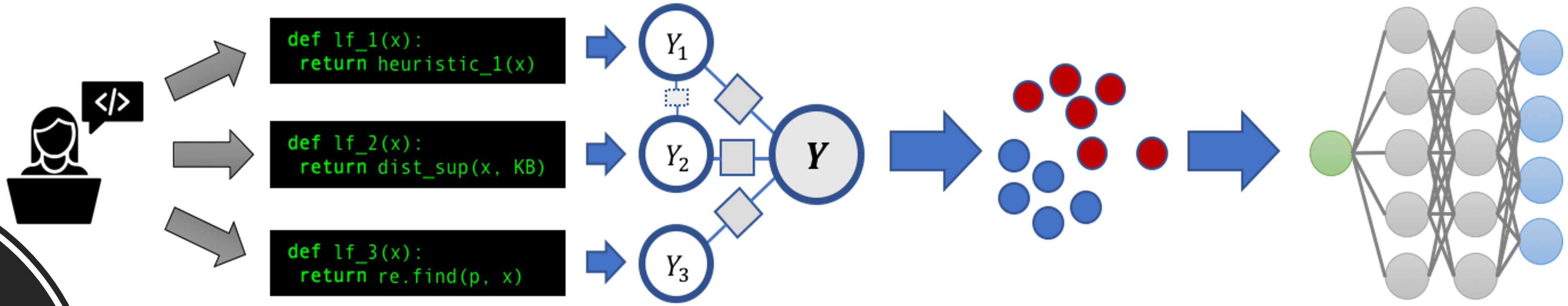


Fig. taken from Snorkel's blog (Hazy Research group, Stanford)

DATA PROGRAMMING

- Labeling both train/test data using simple heuristic/noisy **Labeling Functions (LFs)** programmatically – mostly weakly supervised.
- Might include dependencies between them.
- Can have varying coverage & accuracy, labels present or abstained.
- LFs can be – domain heuristics, crowdsourced data, weak classifiers, even partially-known labels.
- LF Matrix – which is predominantly sparse – is fed into a Generative model fine-tuned by a Discriminative model.

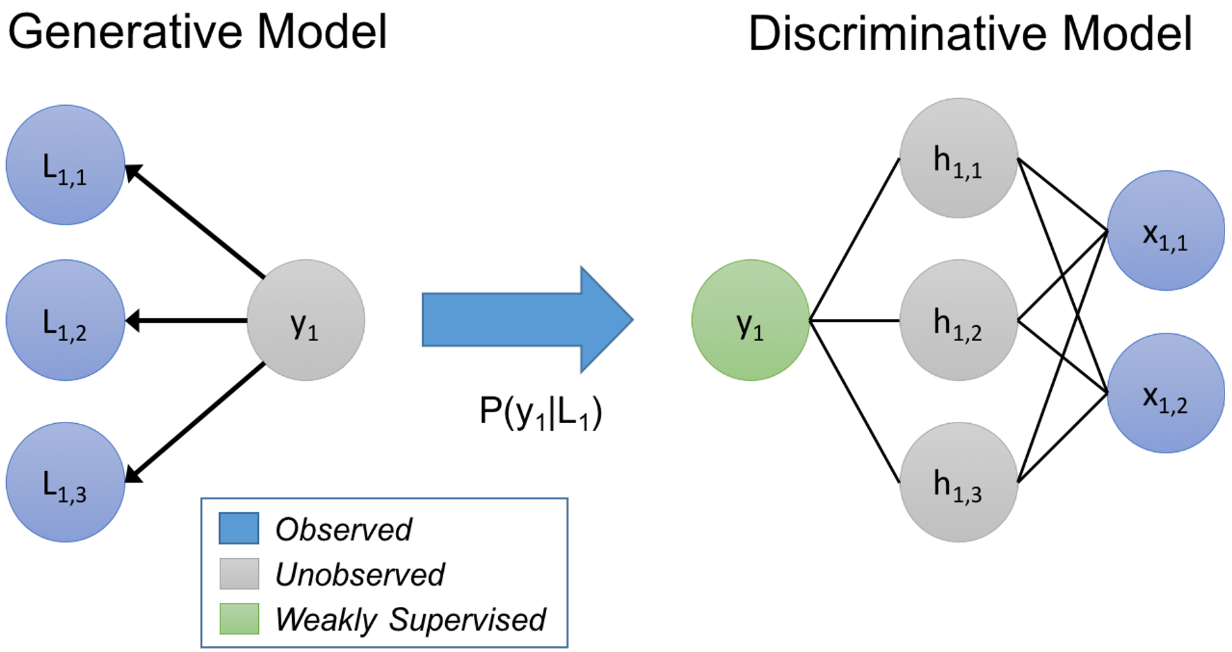


Fig. taken from Snorkel's blog (Hazy Research group, Stanford)

Given – labels generated by LFs (L), data points (x), corresponding unknown true labels (y),

- A **Generative model** P(L, Y) with parameters ω, gives out probabilities over unknown labels.
- The noise-aware **Discriminative model** P(Y|L) on marginals can be used to govern and minimize expected loss.
- As and when dataset is larger, loss tends to get better – in turn labels generated.

Data Programming: Creating Large Training Sets, Quickly, NeurIPS 2016

DATA PROGRAMMING

DATA PROGRAMMING FOR MOBI-QUITOUS SENSING *

* Mobile/Wearable sensing, particularly HAR tasks are predominantly time-series.
* Hard to obtain training labels
* Harder to obtain real-world data on-the-fly – particularly in an Incremental Learning scenario for continuous model updation.
* Challenging to label incoming data on-device using Data Programming, since high computational resources required for modeling the whole pipeline.
* Reduces oracle involvement, enables non-experts to easily automate ground truthing.

* Work in Progress

Contact:

Gautham Krishna Gudur
gauthamkrishna.gudur@gmail.com

Let’s Connect!

THANK YOU!

QUESTIONS?