Can Calibration Improve Sample Prioritization?

Ganesh Tata*
University of Alberta
gtata@ualberta.ca

Gautham Krishna Gudur*
Global AI Accelerator, Ericsson
gautham.krishna.gudur@ericsson.com

Gopinath Chennupati
Amazon Alexa
cgnath.dr@gmail.com

Mohammad Emtiyaz Khan
RIKEN Center for AI Project
emtiyaz.khan@riken.jp

Abstract

Calibration can reduce overconfident predictions of deep neural networks, but *can calibration also accelerate training?* In this paper, we show that it can when used to prioritize some examples for performing subset selection. We study the effect of popular calibration techniques in selecting better subsets of samples during training (also called sample prioritization) and observe that calibration can improve the quality of subsets, reduce the number of examples per epoch (by at least 70%), and can thereby speed up the overall training process. We further study the effect of using calibrated pre-trained models coupled with calibration during training to guide sample prioritization, which again seems to improve the quality of samples selected.

1 Introduction

Calibration is a widely used technique in machine learning to reduce overconfidence in predictions. Modern deep neural networks are known to be overconfident classifiers or predictors, and calibrated networks provide trustworthy and reliable confidence estimates [1]. Hence, finding new calibration techniques and improving them has been an active area of research [1, 6, 10, 11].

In this paper, we ask *if calibration aids in accelerating training by using sample prioritization*, i.e., we select training samples based on calibrated predictions to better steer the training performance. We explore different calibration techniques and focus on selecting a subset with the most informative samples during each epoch. We observe that *calibration performed during training plays a crucial role in choosing the most informative subsets*, which in turn accelerates neural network training. We then investigate the effect of an external pre-trained model which is well-calibrated (with larger capacity) on the sample selection process during training.

Our contributions are as follows,

We provide an in-depth study analyzing the effect of various calibration techniques on sample prioritization during training. We also consider pre-trained calibrated *target* models and observe their effect on sample prioritization along with calibration during training. We benchmark our findings on widely used CIFAR-10 and CIFAR-100 datasets and observe the improved quality of the chosen subsets across different subset sizes, which ensures faster deep neural network training.

*Both authors contributed equally to this work.

Has it Trained Yet? Workshop at the Conference on Neural Information Processing Systems (NeurIPS 2022).

2 Background

2.1 Problem Statement

We formulate the problem in the paper as follows. A calibration technique C is performed during training at each epoch, and a sample prioritization function a is then used to select the most informative samples for training each subsequent epoch. We use *Expected Calibration Error (ECE)* for model calibration [10], which measures the absolute difference between the model's accuracy and its confidence.

The paper discusses how a calibration technique C, when coupled with a sample prioritization function a, affects the performance (accuracy and calibration error (ECE)) of the model. In addition, we also observe if this phenomenon can aid in faster and more efficient training. We hypothesize a closer relationship between calibration and sample prioritization during training, wherein the calibrated model probabilities at each epoch are used by a sample prioritization criterion to select the most informative samples for training each subsequent epoch.

2.2 Calibration

Calibration is a technique that curbs overconfident predictions in deep neural networks, wherein the predicted (softmax) probabilities reflect true probabilities of correctness (better confidence estimates) [1]. In this paper, we consider various prominently used calibration techniques which are performed during training.

Label Smoothing implicitly calibrates a model by discouraging overconfident prediction probabilities during training [9]. The one-hot encoded ground truth labels (y_k) are smoothened using a parameter α, that is $y_k^{LS} = y_k(1-\alpha) + \alpha/K$, where K is the number of classes. These smoothened targets y_k^{LS} and predicted outputs p_k are then used to minimize the cross-entropy loss.

Mixup is a data augmentation method [14] which is shown to output well-calibrated predictive scores [13], and is again performed during training.

$$\bar{x} = \lambda x_i + (1-\lambda)x_j$$
$$\bar{y} = \lambda y_i + (1-\lambda)y_j$$

where x_i and x_j are two input data points that are randomly sampled, and y_i and y_j are their respective one-hot encoded labels. Here, $\lambda \sim \text{Beta}(\alpha, \alpha)$ with $\lambda \in [0, 1]$.

Focal Loss is an alternative loss function to cross-entropy which yields calibrated probabilities by minimizing a regularized KL divergence between the predicted and target distributions [8].

$$L_{Focal} = -(1-p)^{\gamma} log p$$

where p is the probability assigned by the model to the ground-truth correct class, and γ is a hyperparameter. When compared with cross-entropy, Focal Loss has an added factor that encourages the samples predicted with correct classes to have lower probabilities. This enables the predicted distribution to have higher entropy, thereby helping avoid overconfident predictions.

2.3 Sample Prioritization

Sample prioritization is the process of selecting important samples during different stages of training to accelerate the training process of a deep neural network without compromising on performance. In this paper, we perform sample prioritization during training using *Max Entropy*, which is a de facto uncertainty sampling technique to select the most efficient samples at each epoch.

Max Entropy selects the most informative samples (top-k) that maximize the predictive entropy [12].

$$\mathbb{H}[y|x, D_{train}] := -\sum_c p(y=c|x, D_{train}) \log p(y=c|x, D_{train})$$

2.4 Pre-trained Calibrated Target models

Pre-trained models have been widely used in literature to obtain comprehensive sample representations before training a downstream task [5]. We use a *pre-trained calibrated model with larger capacity*

which we call the *target* model [3] and use the Max Entropy estimates obtained from this target model at each epoch to select the samples, thereby guiding the corresponding epochs of the model during training. Further, we call the model which is being trained as the *current* model. In this paper, we perform sample prioritization with the target model in addition to calibrating the current model.

Table 1: Test Accuracies (%) and ECEs (%) across various calibration techniques and subset sizes with Resnet-34 as *current* model for both datasets.

Dataset	Calibration	100%		30%		20%		10%	
		Accuracy	ECE	Accuracy	ECE	Accuracy	ECE	Accuracy	ECE
CIFAR-10	**No Calibration** Cross-Entropy (Baseline)	94.1	4.1	93.6	5.33	**93.86**	4.01	**93.23**	5.2
	Label Smoothing 0.03/0.05/0.05/0.03	94	1.84	91.74	3.17	91.48	3.56	91.72	2.71
	Mixup 0.1/0.3/0.2/0.15	**95.1**	2.1	**94.39**	2.67	93.35	2.59	93.17	1.78
	Focal Loss 1/3/3/3	94.69	**1.71**	93.19	**1.2**	92.6	**1.25**	92.25	**1.42**
CIFAR-100	**No Calibration** Cross-Entropy (Baseline)	77.48	5.42	73.13	10.77	71.54	13.16	**69.65**	14.47
	Label Smoothing 0.03/0.03/0.03/0.09	77.05	4.88	72.21	3.45	70.93	5.75	68.63	5.67
	Mixup 0.15/0.15/0.15/0.35	**78.68**	3.59	**73.57**	**1.49**	**72.02**	**2.4**	69.1	**1.16**
	Focal Loss 1/3/3/5	78.59	**3.57**	71.86	1.67	70.61	3.25	65.81	1.82

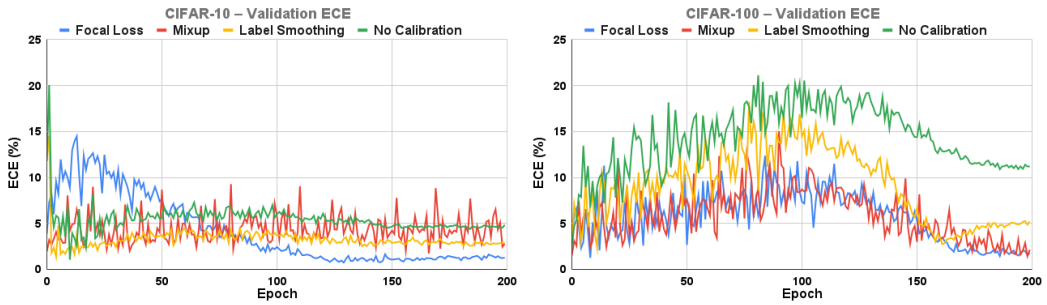
3 Experiments and Results

We perform our experiments on CIFAR-10 and CIFAR-100 [4] datasets with the setting mentioned in Section 2.1, and we use Resnet-34 [2] as the *current* model. For both datasets, the initial training set consisting of 50,000 samples is split into 90% training data and 10% validation data, while the test set contains 10,000 samples. In the sample prioritization setting, we start with 10 warm-up epochs in which all samples are selected during training (no subset selection), following which we select $n\%$ of total training samples in each epoch using the Max Entropy criterion. We choose different settings of subset sizes for each epoch with n – {10, 20, 30}.

We use the Stochastic Gradient Descent optimizer with initial learning rates of 0.01 and 0.1 for CIFAR-10 and CIFAR-100 respectively, trained for 200 epochs with a cosine annealing [7] scheduler, weight decay of $5e^{-4}$ and momentum of 0.9 for both datasets. The models are trained using a V100 GPU. We consider classification accuracies and ECEs across different calibration techniques with their respective parameter sweeps as follows: *Label Smoothing* (α) – {0.01, 0.03, 0.05, 0.07, 0.09}, *Mixup* (α) – {0.1, 0.15, 0.2, 0.25, 0.3, 0.35} and *Focal Loss* (γ) – {1, 2, 3, 4, 5}. As baselines, we consider uncalibrated models with standard cross-entropy loss. For the *target* experiments, we choose Resnet-50 models [2] trained with Mixup (with $\alpha = 0.3$ for CIFAR-10, and $\alpha = 0.25$ for CIFAR-100) as our target models after performing parameter sweeps across all calibration techniques.

3.1 Discussion on Results

Table 1 shows the test ECEs and accuracies for the current model across different calibration techniques and subset sizes with Max Entropy criterion. We can observe that *all calibration techniques have lower test ECEs than their respective uncalibrated models across all subset sizes* for both datasets. This demonstrates that *performing calibration during training improves sample prioritization*. Moreover, these results indicate that there are no significant trade-offs between model accuracy and model confidence (ECE) when calibration is performed with sample prioritization. Figures 1a and 1b also illustrate lower validation ECEs across training epochs for calibrated current models when compared to their uncalibrated counterparts, with 30% subset size as an instance. In particular, we observe that test accuracies for Mixup across all subset sizes are consistently comparable and often higher than their respective uncalibrated models.



(a) CIFAR-10 (b) CIFAR-100

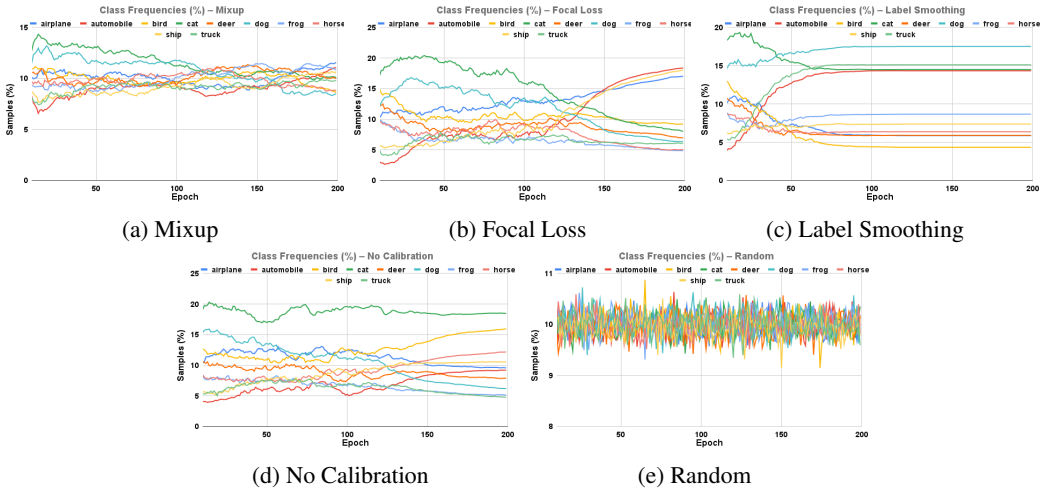
Figure 1: Validation ECEs (%) for both datasets with 30% subset size for *current* model.

Table 2: Test Accuracies (%) and ECEs (%) across various calibration techniques and subset sizes with Resnet-34 as *current* model for both datasets, and Resnet-50 (Mixup) as *target* model.

Dataset	Calibration	100%		30%		20%		10%	
		Accuracy	ECE	Accuracy	ECE	Accuracy	ECE	Accuracy	ECE
CIFAR-10	**No Calibration** Cross-Entropy (Baseline)	94.1	4.1	93.95	4.04	93.43	4.9	93.16	4.11
	Label Smoothing 0.03/0.05/0.05/0.03	94	1.84	93.62	2.93	93.3	3.32	**93.27**	1.9
	Mixup 0.1/0.3/0.15/0.15	**95.1**	2.1	**94.7**	2.88	**93.79**	2.73	93.22	2.16
	Focal Loss 1/2/2/1	94.69	**1.71**	93.15	**1.06**	92.65	**1.58**	92.84	**1.89**
CIFAR-100	**No Calibration** Cross-Entropy (Baseline)	77.48	5.42	75.38	9.36	75.04	9.39	71.07	9.27
	Label Smoothing 0.03/0.03/0.03/0.09	77.05	4.88	**76.06**	2.28	**75.27**	2.67	**72.59**	1.63
	Mixup 0.15/0.2/0.15/0.15	**78.68**	3.59	75.62	**0.86**	74.78	**1.43**	70.32	**0.86**
	Focal Loss 1/2/3/2	78.59	**3.57**	74.89	2.37	73.73	**1.43**	70.89	1.51

As expected, we observe from Table 1 that the test accuracies reduce when the subset size becomes smaller. Moreover, Mixup consistently has higher test accuracies and low test ECEs across different subset sizes compared to other calibration techniques for both datasets. This could be attributed to Mixup being a data transformation/augmentation technique, thereby performing well even in the low-data regime. Further, Figure 2 shows that training with Mixup leads to a relatively balanced representation of classes in the chosen subsets. In contrast, Label Smoothing and Focal Loss are loss-based calibration techniques with no explicit transformation performed on the underlying training data. Here, we note that Focal Loss can be more effective when coupled with a post hoc calibration technique like temperature scaling [8]. However, this setting is not applicable in our paper since we explicitly focus on calibration techniques during training. In addition, Mixup and Focal Loss are known to outperform Label Smoothing [13, 8].

Effectiveness of Target: Table 2 exhibits the results when a pre-trained calibrated target model (Resnet-50 with Mixup) is used for guiding the current model while performing sample prioritization with calibration during training. Interestingly, a well-calibrated target model can boost the performance on both datasets for under-performing calibration techniques (like Label Smoothing) performed only on the current model without the target's influence. However, for calibration techniques that are already performing well (like Mixup and Focal Loss), there is no significant loss in performance on CIFAR-10, while there is a considerable improvement in performance in general on CIFAR-100. This can be clearly observed when comparing Table 2 with Table 1. We assume that the performance of a current model trained with 100% data is similar for all experiments performed with and without a target model for each respective dataset.



(a) Mixup (b) Focal Loss (c) Label Smoothing

(d) No Calibration (e) Random

Figure 2: Class Distribution across epochs for all calibration techniques with Max Entropy (a)-(d) and Random Sampling (e) with 30% subset size for *current* model on CIFAR-10.

Class Distribution: Figures 2a - 2d show the class distribution across training epochs for different calibration techniques with sample prioritization performed using Max Entropy. For comparison, we also consider Random sampling as a baseline (Figure 2e). As discussed above, performing Mixup (Figure 2a) during training leads to a relatively balanced representation of classes in each epoch. In contrast, Label Smoothing and Focal Loss (Figures 2b and 2c) do not exhibit a balanced representation of classes in the chosen subsets. The class representations are imbalanced for the uncalibrated setting as well (Figure 2d).

4 Conclusion

In this paper, we investigate whether existing calibration techniques improve sample prioritization during training. We empirically show that a deep neural network calibrated during training selects better subsets of samples than an uncalibrated model. We also demonstrate the effectiveness of pre-trained calibrated target models in guiding sample prioritization during training, thereby boosting calibration performance. Finally, since calibration aids in sample prioritization by improving the quality of subsets, it ensures faster neural network training.

References

[1] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[3] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *International Conference on Machine Learning*, pages 2712–2721, 2019.

[4] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Master's thesis, University of Toronto*, 2009.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

[6] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In *International Conference on Machine Learning*, pages 2796–2804, 2018.

[7] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

[8] Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip Torr, and Puneet Dokania. Calibrating deep neural networks using focal loss. *Advances in Neural Information Processing Systems*, 33:15288–15299, 2020.

[9] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? *Advances in neural information processing systems*, 32, 2019.

[10] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[11] Jeremy Nixon, Michael W Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. In *CVPR Workshops*, 2019.

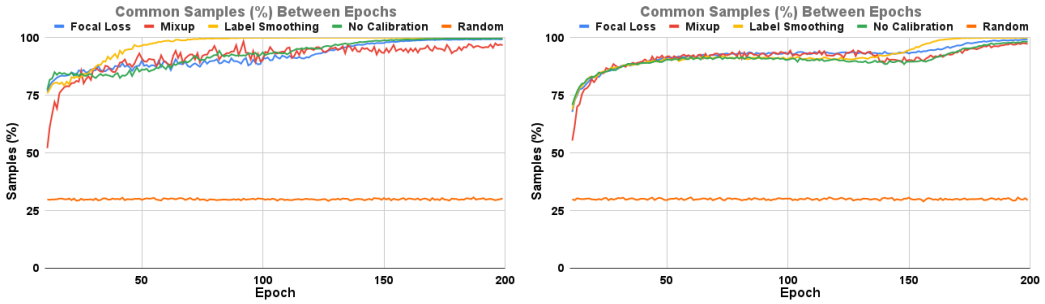
[12] Claude Elwood Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[13] Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[14] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *International Conference on Learning Representations*, 2018.

A Common Samples between Epochs

Figures 3a and 3b show the percentage of common samples between consecutive epochs for CIFAR-10 and CIFAR-100 respectively across various calibration techniques, with Max Entropy and Random Sampling as the sample prioritization criteria with 30% subset size. In all settings, sample selection with Max Entropy leads to a significantly higher percentage of common samples between consecutive epochs throughout training, as opposed to random sampling which results in very few common samples. Although only a small percentage of samples change across epochs, sample prioritization with Max Entropy coupled with any calibration technique yields good classification performance.

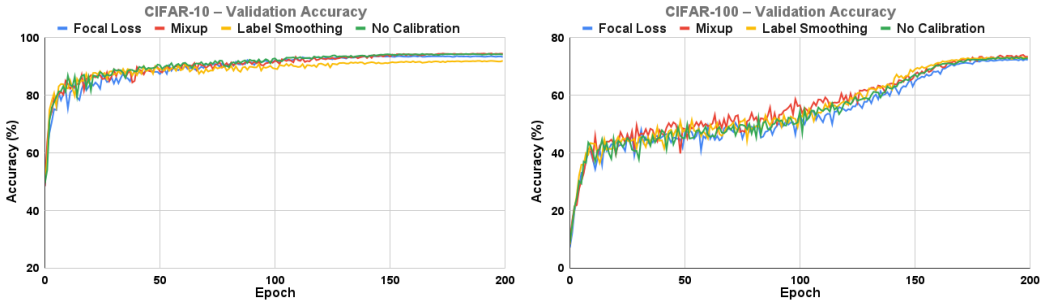


(a) CIFAR-10 (b) CIFAR-100

Figure 3: Common Samples (%) between epochs with 30% subset size for *current* model.

B Validation Accuracies

We report the validation accuracies of CIFAR-10 and CIFAR-100 with 30% subset size for the current model across all calibration techniques with Max Entropy as the sample prioritization criterion in Figures 4a and 4b.

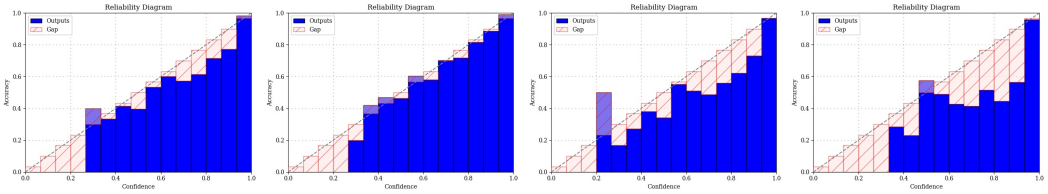


(a) CIFAR-10 (b) CIFAR-100

Figure 4: Validation Accuracies (%) for both datasets with 30% subset size for *current* model.

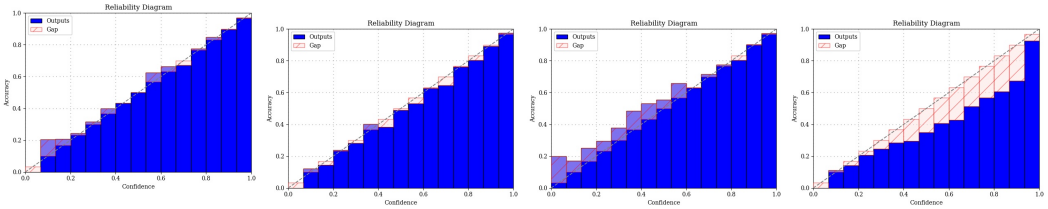
C Reliability Diagrams

We report the reliability diagrams of CIFAR-10 and CIFAR-100 with 30% subset size for the current model across all calibration techniques with Max Entropy as the sample prioritization criterion in Figure 5 and 6.



(a) Mixup (b) Focal Loss (c) Label Smoothing (d) No Calibration

Figure 5: Reliability diagrams for CIFAR-10 with 30% subset size for *current* model.



(a) Mixup (b) Focal Loss (c) Label Smoothing (d) No Calibration

Figure 6: Reliability diagrams for CIFAR-100 with 30% subset size for *current* model.

Data-Efficient Automatic Model Selection in Unsupervised Anomaly Detection

Gautham Krishna Gudur, Raaghul R, Adithya K, Shrihari Vasudevan
Global AI Accelerator, Ericsson
{gautham.krishna.gudur, r.raaghul, shrihari.vasudevan}@ericsson.com

***Abstract*—Anomaly Detection is a widely used technique in machine learning that identifies context-specific outliers. Most real-world anomaly detection applications are unsupervised, owing to the bottleneck of obtaining labeled data for a given context. In this paper, we solve two important problems pertaining to unsupervised anomaly detection. First, we identify only the most informative subsets of data points and obtain ground truths from the domain expert (oracle); second, we perform efficient model selection using a Bayesian Inference framework and recommend the top-k models to be fine-tuned prior to deployment. To this end, we exploit multiple existing and novel acquisition functions, and successfully demonstrate the effectiveness of the proposed framework using a weighted Ranking Score (η) to accurately rank the top-k models. Our empirical results show a significant reduction in data points acquired (with at least 60% reduction) while not compromising on the efficiency of the top-k models chosen, with both uniform and non-uniform priors over models.**

***Index Terms*—Unsupervised Anomaly Detection, Model Selection, Bayesian Inference, Subset Selection**

I. Introduction

Anomaly Detection (AD) is the process of identifying unexpected or unforeseen events in data sets of various kinds. Anomalies could also be considered unlikely events with respect to a deterministic threshold, assuming the existence of a distribution of various events. In the past few years, machine learning has led to major breakthroughs in various areas related to automation and digitization tasks, and AD plays an instrumental role in such tasks.

There have been multiple AD frameworks with a rich literature of supervised, unsupervised, and semi-supervised algorithms [1], [2]. In general, an anomaly is a domain/context-specific definition, which evolves over time, depending on changes in data distributions, geographical constraints, business contexts, and many more. This makes defining anomalies in any domain a cumbersome task and requires extensive human expertise and domain knowledge to be inculcated in the current frameworks. Currently, domain experts rely on their expertise to decide what is or what is not an anomaly. Moreover, most real-world AD systems across a myriad of application contexts have to rely on unsupervised anomaly detection for various reasons like labeling costs from humans and other sources, data privacy issues, and so on. A few representative exemplars of such real-world AD contexts include identifying anomalies in resource utilization in a telecommunication network, identifying operational network issues using Quality of Experience (QoE), Quality of Service (QoS), and other factors, identifying abnormal medical conditions amongst patients, extremities in climate change, and many more. With such AD systems in place, preventive and corrective measures could be taken proactively.

In addition to detecting anomalies, *choosing the appropriate algorithms for the given data in an efficient manner is hard, particularly in an unsupervised setting*. Solution developers predominantly rely on experimentation and/or trial-and-error on the whole data to decide the best approaches for their use cases. The problem at hand is thus two-fold – first, *efficiently choosing the right subset of informative data points to be identified as anomalies*; second, *recommending the best AD algorithms from this subset of data*.

The main contributions of this paper are,

- We propose a *model selection framework for unsupervised anomaly detection* using *Bayesian Inference*, and propose a novel *ranking criterion* for selecting the best models.
- We address the labeled-data scarcity problem in unsupervised AD via *subset selection* wherein, labels for a small fraction of most-informative data points are acquired from an oracle (human), by exploiting multiple *existing and novel acquisition functions*.
- We benchmark our proposed model selection framework using various standard datasets to showcase its effectiveness in unsupervised AD settings with *both uniform and non-uniform priors over models*, which operate in synchronicity with human-augmented user feedback.

We demonstrate our framework as a viable automation approach to develop unsupervised AD solutions for real-world applications with minimal supervision.

II. Related Work

Conventionally, most anomaly detection/outlier detection works have rigorously focused on improving existing anomaly detection algorithms or proposing new interesting ones [1]. Such works deal with multiple types of anomalies, which can broadly be classified as point, contextual and collective anomalies [3].

1) Point anomalies: Data instances are considered anomalous with respect to the rest of the data.

2) Contextual anomalies: Data instances are considered anomalous in a specific context, like a month or day within the context of a week or a location.

3) Collective anomalies: Collection of continuous data instances is considered anomalous with respect to the entire dataset.

This necessitates an anomaly detection system with appropriate approaches to cater to all such anomaly types. Moreover, most anomaly detection systems in real-time are inherently unlabeled, thereby making them unsupervised in nature. These challenges make identifying all such anomalies a tedious process and require data labeling from experts well-versed in their respective domains.

However, algorithms selection and/or recommendation in an unsupervised AD setting has been relatively unexplored. Model selection in AD systems conventionally requires careful and rigorous selection over all possible sets of relevant algorithms. Moreover, finding a common metric for evaluation and effective comparison of these algorithms is hard, due to their unsupervised nature and requires the context of the problem. Hence, a robust framework that can be incorporated into the existing anomaly detection systems (consisting of diverse AD algorithms) becomes necessary to select/recommend the best algorithms for the given data.

We note some existing work on model selection for one-class models [4], [5], however they are limited only to a specific type of model class. METAOD [6] proposes an effective way to select the best approaches in unsupervised anomaly detection, however, it relies on training an offline meta-learner on various meta-train datasets, with hand-picked meta-features, which are computationally expensive to create. Similarly, [7] proposes using a pre-trained model selector and pre-trained parameter estimator for unsupervised anomaly detection which is again cumbersome to learn.

The conventional ways of model selection for any machine learning task, in our case, unsupervised anomaly detection, involve selecting the best hyperparameters from an exhaustive initial range of multiple hyperparameter values using Grid Search or similar mechanisms, with a hold-out validation set [8], [9]. However, such mechanisms are again search algorithms, which end up taking massive amounts of time, and require a thorough knowledge of the hyperparameters to choose from.

Conventional works on active learning typically involve machine learning classification algorithms, and a few interesting applications [10]–[12]. There are also multiple active learning/subset selection works that leverage the use of an oracle (user feedback), which are particularly useful in unsupervised anomaly detection settings, where there are predominantly no labels in real-time applications. This helps alleviate massive labeling efforts for domain experts. Interesting works on finding useful anomalies using active learning, and treating them as a rare category [13], active learning for AD on environmental data [14] are noted. Deep active learning for unsupervised AD is explored in [15], however, none of the above works are in the context of efficient model selection.

Hence, a unified framework for human-augmented and data-efficient model selection in any given anomaly detection system, particularly unsupervised AD, makes it more convenient for the end-user to identify the appropriate best-fit algorithms for the problem at hand.

III. Our Approach

The following are the steps in our proposed unsupervised anomaly detection model selection system.

Algorithm 1: Our Proposed Framework

Input: Train Dataset $\mathcal{D}_{train}$, Total unsupervised anomaly detection models M, Total Bayesian Inference iterations I, Acquisition Function AF, Subset Dataset $\mathcal{D}_{subset}$
Output: Top-k models chosen K, Ranking Score η

Initialize Categorical distribution (likelihood) over M models
Initialize Dirichlet Priors $p_i \sim Dir(\alpha_i), i = 1, .., M$ with concentrations α_i
Obtain probabilities from all M unsupervised AD models with $\mathcal{D}_{train}\{x\}$
Subset selection on $\mathcal{D}_{train}\{x\}$ using AF to obtain $\mathcal{D}_{subset}\{x\}$
Present $\mathcal{D}_{subset}\{x\}$ to oracle to obtain labeled subset $\mathcal{D}_{subset}\{x, y\}$
Choose corresponding best model for $\mathcal{D}_{subset}\{x, y\}$
for $i = 1$ **to** I **do**
  Update model posterior $p_i|\alpha_i, c_i \sim Dir(\alpha_i + c_i), i = 1, .., M$ based on best model for $\mathcal{D}^i_{subset}\{x, y\}$
end for
Select top-k models (K) based on the model posterior
for $k = 1$ **to** K **do**
  Calculate F1-score, Accuracy, Average Precision Score, AUC ROC, η of model k
end for
Return top-k models with best hyperparameters using Grid Search

A. Bayesian Inference Framework for Model Selection

To perform model selection over unsupervised anomaly detection approaches, we propose a Bayesian inference framework using Exact Inference (EI), Stochastic Variational Inference (SVI), or Markov Chain Monte Carlo (MCMC), for modeling posterior probabilities [16]. We do not consider MCMC in this scenario since the time complexity is extremely high, in comparison to SVI and EI.

Given a Categorical likelihood distribution over all AD models, if the prior distribution of the Categorical likelihood distribution is Dirichlet, then the corresponding posterior distribution is also Dirichlet since the Dirichlet distribution is a conjugate prior for the Categorical/Multinomial distribution [17]. The samples of the Dirichlet posterior would give us probabilities of the Categorical distribution over all AD models. Hence, it is straightforward and sufficient to use Exact Inference, but we also perform SVI to benchmark

its performance relative to EI. If the domain necessitates alternative prior distributions, EI may not be feasible and SVI and MCMC could be explored as options for non-conjugate-priors.

1) Discussion on Dirichlet Priors: There are multiple ways to incorporate apriori beliefs in the form of priors for the data in consideration. We propose defining priors over anomaly detection approaches based on a taxonomy of,

- Types of anomalies like point, contextual and collective.
- The given type/distribution of data (tree-based, density-based, etc.).
- By adding domain knowledge on the approaches (along with other meta-data like priors over features), that the user believes will perform well on the given data.

In the scenario where priors in the form of taxonomies are unavailable, we initialize Dirichlet priors over the set of models in consideration to be *uniform*, which is typically the default setting.

The acquisition functions during subset selection (discussed in Section III-B) – where the user provides feedback on anomalous behavior for a chosen subset of data, also enhances the priors over the AD approaches for the next iteration. Incorporating such priors may improve convergence rate (faster convergence), thereby also potentially requiring fewer iterations to obtain effective posteriors.

B. Acquisition Functions for Subset Selection

Acquisition functions are used in subset selection to choose the most informative set of data points to be queried from the overall data D_{train}. We examine and propose the following acquisition functions,

a) Boundary: This acquisition function selects points that are close to the boundary threshold for each model, and are considered the most uncertain.

$$abs(p_{ij} - threshold)$$

b) Max Disagreement: Selects data points wherein each model's disagreement against consensus probabilities (mean probabilities across models) is the largest for some learners.

c) Boundary Max Disagreement: Combines Boundary and Max Disagreement acquisition functions, wherein it first selects the data points closest to the boundary threshold, and then selects the points with Max disagreement.

d) Max Entropy: This acquisition function chooses data points that maximize the predictive entropy.

$$-\sum_{c} p(y = c|x, D_{train}) \log p(y = c|x, D_{train})$$

e) Variance Entropy: This acquisition function selects data points where the probability distribution across various models has the highest variance.

$$\sigma^2 = \frac{\sum_{j=1}^{M} (p_{ij} - \mu)^2}{M}$$

f) Random: This acquisition function chooses data points uniformly at random.

We benchmark the aforementioned acquisition functions with 100% data, which is denoted by *All*.

C. Ranking Score (η)

The ranking score, defined by η, is a position-weighted aggregated similarity metric between the top-k model recommendations of a given subset, and the top-k model recommendations of the entire 100% dataset, i.e., with and without subset selection. The score depends on the presence of a model in the top-k models, as well as its position.

$$\eta = 1/k * \sum_{r=1}^{k} r_{subset}/(r_{subset} + abs(r_{full} - r_{subset}))$$

where r_{subset} is the rank of a model for the given acquisition function, while r_{full} is the rank of a model with the entire dataset. η indicates the effectiveness of the top-k model recommendations for different acquisition functions with respect to the entire 100% data.

IV. Experiments and Results

We train and evaluate our experiments on five different anomaly detection datasets as observed in Table I. These datasets, from DAMI[1] [18], are often used in the unsupervised anomaly detection benchmarking literature.

Dataset	Instances	Attributes	Outliers (%)
Waveform	3443	21	2.9
Annthyroid	7129	21	7.49
Pima	768	8	34.9
Wilt	4819	5	5.33
PageBlocks	5393	10	9.46

TABLE I: Characteristics of the Datasets

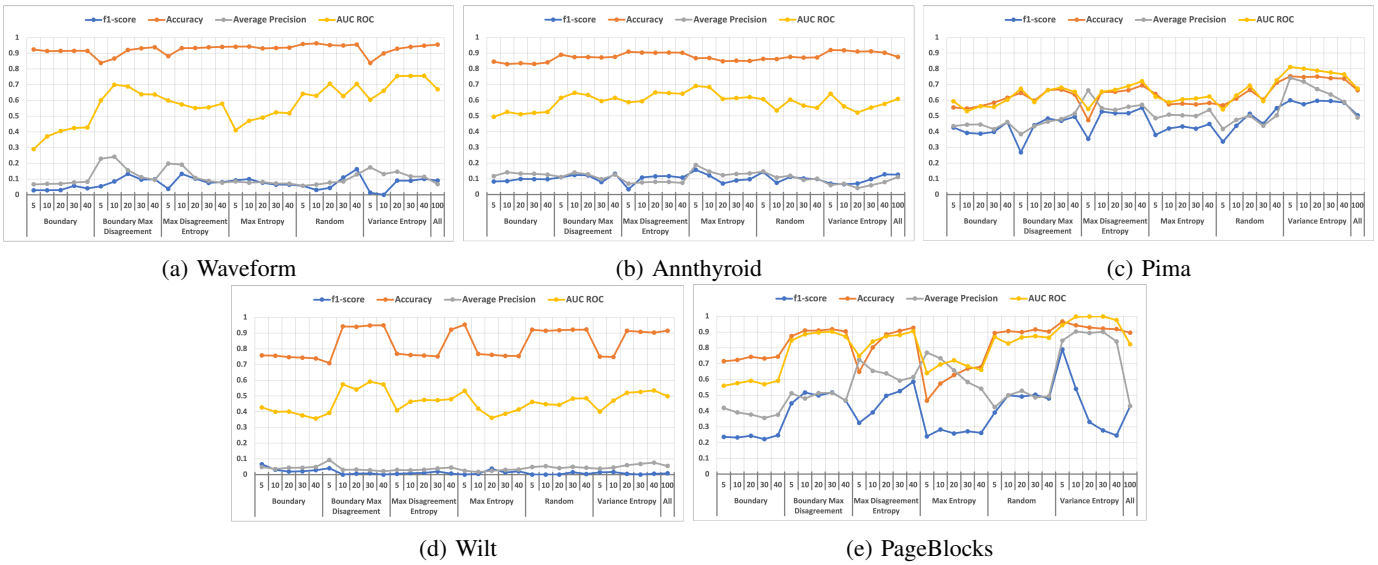
The subset selection experiments are performed with combinations of five different percentages – 5%, 10%, 20%, 30%, 40%, and six acquisition functions – Boundary, Max Disagreement Entropy Boundary, Max Disagreement, Max Entropy, Variance Entropy, Random, as discussed in Section III-B. The experiments are executed for each combination of subset size (%) and acquisition function. In addition, we perform these experiments with full (100%) data for benchmarking.

To evaluate our proposed model selection framework, we experiment with nine commonly used unsupervised anomaly detection algorithms in literature [6], [18], [19], starting with hyperparameters sampled at random. For consistency, we use the PyOD package [2] for implementing all models. The initial unsupervised AD algorithms used are, **(1)** COF **(2)** IsoForest **(3)** CBLOF **(4)** LOF **(5)** OCSVM **(6)** KNN **(7)** HBOS **(8)** ABOD **(9)** LODA. We choose the top-5 models from this initial set of nine AD models.

[1] https://www.dbs.ifi.lmu.de/research/outlier-evaluation/DAMI

	Accuracy (%)		f1-score		Avg Precision		AUC ROC		Time (in sec)	
Dataset	**EI**	**SVI**	**EI**	**SVI**	**EI**	**SVI**	**EI**	**SVI**	**EI**	**SVI**
Waveform	92.395	92.365	0.072	0.062	0.111	0.106	0.577	0.564	0.322	1092.113
Annthyrroid	87.621	87.526	0.1	0.102	0.108	0.103	0.591	0.59	0.294	995.956
Pima	63.513	62.762	0.467	0.462	0.52	0.506	0.652	0.661	0.181	614.822
Wilt	83.375	83.265	0.013	0.011	0.041	0.039	0.467	0.458	0.184	630.476
PageBlocks	81.864	81.766	0.394	0.388	0.591	0.588	0.805	0.798	0.217	746.831

TABLE II: Evaluation Metrics for two Bayesian Inference techniques (Exact Inference (EI) and Stochastic Variational Inference (SVI)) for all datasets, each averaged across all Acquisition Functions and subset sizes, along with corresponding times taken.



(a) Waveform (b) Annthyroid (c) Pima

(d) Wilt (e) PageBlocks

Fig. 1: Acquisition Functions (with corresponding subset sizes (in %) and 100% data) vs Evaluation Metrics for all datasets.

The Bayesian Inference model selection framework, as discussed in Section III-A, uses Exact Inference (EI) and Stochastic Variational Inference (SVI) to select the top-5 AD models across different subset sizes (%) and acquisition functions. Here, we perform our experiments with both uniform and non-uniform Dirichlet priors over all AD models. We use the Pyro package [20] for our Bayesian Inference experiments. Further, we perform hyperparameter tuning for the top-5 models selected with maximum Average Precision Score [18] as the criterion to choose the best hyperparameters. Our experiments are performed on an Intel Core i9 @ 2.3 GHz, with 16 GB memory.

A. Discussion on Results

Table II presents the various evaluation criteria like F1-score, Accuracy, Average Precision Score, AUC ROC for two Bayesian inference model selection techniques – Exact Inference (EI) and Stochastic Variational Inference (SVI), averaged across different acquisition functions and subset sizes for all datasets. These metrics are widely used for evaluating unsupervised anomaly detection models in literature [18]. We can clearly observe that EI and SVI have comparable performance across different evaluation criteria, however, the time taken for SVI is exponentially higher (at least 3000x seconds higher for each dataset) than EI. Hence, we report only the EI results in the forthcoming experiments.

From Figure 1, we can observe the evaluation criteria across all acquisition functions and subset sizes for all datasets with EI, including 100% (All) data. The baseline metrics obtained in our experiments are similar for all datasets from [18]. We primarily focus on F1-score and Average Precision score, while accuracy is mostly not emphasized since AD datasets are highly imbalanced.

Figure 1 shows that the average precision of Variance Entropy decreases across subset sizes for datasets with higher anomalies (like Pima), and increases or is at least consistent for datasets with lower anomalies in general. We also observe that Boundary Max Disagreement effectively converges towards optimal F1-scores and optimal average precision scores, and has comparable efficiencies to 100% (All) data for each dataset, as the subset size increases across all datasets. An interesting observation from Figure 1 is that for an unsupervised setting, random acquisition can perform as well as other acquisition functions.

We then filter the best corresponding subset size and report the top-k recommended models across all acquisition functions, along with the entire data (100% with no acquisition) in Table III. Here, we observe that the Ranking Score (η) (as

Dataset with Best Subset Size	Acquisition Criterion	Ranking (top-5)					Ranking Score (η)
		1	2	3	4	5	
Waveform (100% data)	No acquisition	LOF	LODA	OCSVM	HBOS	ABOD	–
Waveform with 40% subset	Boundary	LOF	LODA	OCSVM	HBOS	ABOD	**1.0**
	Boundary Max Disagreement	LOF	LODA	OCSVM	HBOS	ABOD	**1.0**
	Max Disagreement Entropy	LOF	LODA	OCSVM	HBOS	ABOD	**1.0**
	Max Entropy	LOF	LODA	OCSVM	HBOS	ABOD	**1.0**
	Random	LOF	LODA	OCSVM	HBOS	ABOD	**1.0**
	Variance Entropy	LOF	LODA	OCSVM	HBOS	ABOD	**1.0**
Annthyroid (100% data)	No acquisition	LOF	OCSVM	ABOD	KNN	HBOS	–
Annthyroid with 30% subset	Boundary	LOF	ABOD	OCSVM	KNN	HBOS	0.883
	Boundary Max Disagreement	LOF	OCSVM	ABOD	KNN	HBOS	**1.0**
	Max Disagreement Entropy	LOF	OCSVM	ABOD	HBOS	KNN	0.926
	Max Entropy	LOF	ABOD	KNN	OCSVM	HBOS	0.816
	Random	LOF	OCSVM	ABOD	KNN	HBOS	**1.0**
	Variance Entropy	LOF	OCSVM	HBOS	KNN	ABOD	0.863
Pima (100% data)	No acquisition	ABOD	LOF	LODA	OCSVM	KNN	–
Pima with 40% subset	Boundary	ABOD	LOF	OCSVM	LODA	CBLOF	0.801
	Boundary Max Disagreement	ABOD	LOF	LODA	OCSVM	CBLOF	**0.89**
	Max Disagreement Entropy	LOF	ABOD	OCSVM	LODA	KNN	0.743
	Max Entropy	ABOD	LOF	LODA	OCSVM	KNN	**1.0**
	Random	LOF	ABOD	LODA	OCSVM	CBLOF	0.733
	Variance Entropy	LOF	ABOD	OCSVM	LODA	KNN	0.743
Wilt (100% data)	No acquisition	KNN	OCSVM	HBOS	ABOD	LOF	–
Wilt with 20% subset	Boundary	OCSVM	KNN	LOF	ABOD	HBOS	0.696
	Boundary Max Disagreement	KNN	OCSVM	HBOS	ABOD	LOF	**1.0**
	Max Disagreement Entropy	OCSVM	KNN	HBOS	LOF	ABOD	0.76
	Max Entropy	OCSVM	KNN	LOF	HBOS	ABOD	0.678
	Random	KNN	OCSVM	HBOS	ABOD	LOF	**1.0**
	Variance Entropy	OCSVM	HBOS	KNN	LOF	ABOD	0.678
PageBlocks (100% data)	No acquisition	OCSVM	LOF	HBOS	KNN	ABOD	–
PageBlocks with 10% subset	Boundary	LOF	ABOD	OCSVM	HBOS	CBLOF	0.551
	Boundary Max Disagreement	OCSVM	LOF	HBOS	KNN	ABOD	**1.0**
	Max Disagreement Entropy	LOF	ABOD	OCSVM	IsoForest	KNN	0.539
	Max Entropy	ABOD	OCSVM	LOF	KNN	HBOS	0.662
	Random	OCSVM	LOF	HBOS	KNN	ABOD	**1.0**
	Variance Entropy	LOF	KNN	OCSVM	IsoForest	HBOS	0.535

TABLE III: Acquisition Functions vs top-5 recommended Anomaly Detection models along with their respective Ranking Scores for all datasets with their best corresponding subset sizes, set against 100% data with no acquisition criteria.

observed in Section III-C) is again mostly high for Boundary Max Disagreement, which implies consistent performance in identifying top-k models along with random.

We also showcase the posterior probabilities obtained with EI when initialized with uniform and non-uniform Dirichlet priors in Figure 2, with the Boundary Max Disagreement criterion and the corresponding best subset sizes from Table III. The non-uniform priors over the initial nine models are sampled at random wherein, we observe that the prior probabilities of at least three models take up over 75% of the total probabilities across all datasets, simulating a scenario as discussed in Section III-A1. From Figure 2, we can infer that the posterior probabilities obtained when initialized with non-uniform (random) model priors could be inherently biased, and also perform almost similarly to posterior probabilities when initialized with uniform priors across all datasets. This indicates the effectiveness of our proposed framework of coupling subset selection with Bayesian inference. The top-k corresponding recommended models with uniform and non-uniform priors are also shown in Table IV, which show that even with custom non-uniform Dirichlet priors over models, our framework efficiently recommends the top-k models.

V. Conclusion

In this paper, we present three important contributions pertaining to unsupervised anomaly detection. First, we identify the most important subsets of data points by systematically analyzing various existing and novel acquisition functions.

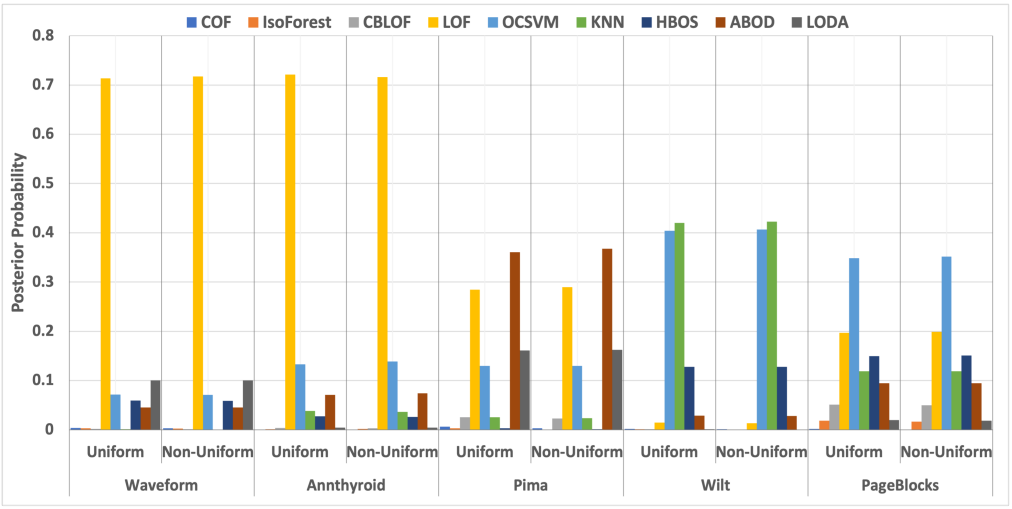


Fig. 2: Posterior Probabilities initialized with uniform and non-uniform Dirichlet Priors with the best performing Boundary Max Disagreement Acquisition Function and best corresponding subset size for each dataset as reported in Table III.

Dataset with Best Subset Size	Dirichlet Prior	Ranking (top-5)				
		1	2	3	4	5
Waveform with 40% subset	Uniform	LOF	LODA	OCSVM	HBOS	ABOD
	Non-Uniform	LOF	LODA	OCSVM	HBOS	ABOD
Annthyroid with 30% subset	Uniform	LOF	OCSVM	ABOD	KNN	HBOS
	Non-Uniform	LOF	OCSVM	ABOD	KNN	HBOS
Pima with 40% subset	Uniform	ABOD	LOF	LODA	OCSVM	KNN
	Non-Uniform	ABOD	LOF	LODA	OCSVM	KNN
Wilt with 20% subset	Uniform	KNN	OCSVM	HBOS	ABOD	LOF
	Non-Uniform	KNN	OCSVM	HBOS	ABOD	LOF
PageBlocks with 10% subset	Uniform	OCSVM	LOF	HBOS	KNN	ABOD
	Non-Uniform	OCSVM	LOF	HBOS	KNN	ABOD

TABLE IV: Top-5 recommended AD algorithms from uniform and non-uniform Dirichlet Priors for all datasets with Boundary Max Disagreement Acquisition Function and best corresponding subset size for each dataset.

Second, we successfully showcase the effectiveness of our unified data-efficient Bayesian Inference model selection framework, demonstrated by extensive benchmarking of evaluation criteria. Finally, we also formulate a Ranking Score (η) to rank our top-k models selected using Bayesian inference which operates in settings with both uniform and non-uniform priors over models, thereby enabling end-users to easily use the selected/recommended models. This work will enable data-efficient model selection for unsupervised AD.

REFERENCES

[1] C. C. Aggarwal, "An introduction to outlier analysis," in *Outlier analysis*. Springer, 2017, pp. 1–34.

[2] Y. Zhao, Z. Nasrullah, and Z. Li, "Pyod: A python toolbox for scalable outlier detection," *Journal of Machine Learning Research*, vol. 20, no. 96, pp. 1–7, 2019.

[3] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[4] E. Burnaev, P. Erofeev, and D. Smolyakov, "Model selection for anomaly detection," in *Eighth International Conference on Machine Vision (ICMV)*, vol. 9875. SPIE, 2015, pp. 445–450.

[5] H. Deng and R. Xu, "Model selection for anomaly detection in wireless ad hoc networks," in *2007 IEEE Symposium on Computational Intelligence and Data Mining*. IEEE, 2007, pp. 540–546.

[6] Y. Zhao, R. Rossi, and L. Akoglu, "Automatic unsupervised outlier model selection," in *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[7] Y. Ying, J. Duan, C. Wang, Y. Wang, C. Huang, and B. Xu, "Automated model selection for time-series anomaly detection," *arXiv preprint arXiv:2009.04395*, 2020.

[8] S. Raschka, "Model evaluation, model selection, and algorithm selection in machine learning," *arXiv preprint arXiv:1811.12808*, 2018.

[9] J. Soenen, E. Van Wolputte, L. Perini, V. Vercruyssen, W. Meert, J. Davis, and H. Blockeel, "The effect of hyperparameter tuning on the comparative evaluation of unsupervised anomaly detection methods," in *Proceedings of the KDD'21 Workshop on Outlier Detection and Description*, 2021, pp. 1–9.

[10] B. Settles, "Active learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2012.

[11] A. Ragav and G. K. Gudur, "Bayesian active learning for wearable stress and affect detection," *arXiv preprint arXiv:2012.02702*, 2020.

[12] G. K. Gudur, P. Sundaramoorthy, and V. Umaashankar, "Activeharnet: Towards on-device deep bayesian active learning for human activity recognition," in *The 3rd International Workshop on Deep Learning for Mobile Systems and Applications*, 2019, pp. 7–12.

[13] D. Pelleg and A. Moore, "Active learning for anomaly and rare-category detection," *Advances in Neural Information Processing Systems*, vol. 17, 2004.

[14] S. Russo, M. Lürig, W. Hao, B. Matthews, and K. Villez, "Active learning for anomaly detection in environmental data," *Environmental Modelling & Software*, vol. 134, 2020.

[15] T. Pimentel, M. Monteiro, A. Veloso, and N. Ziviani, "Deep active learning for anomaly detection," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[16] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *Journal of Machine Learning Research*, vol. 14, pp. 1303–1347, 2013.

[17] T. S. Ferguson, "A bayesian analysis of some nonparametric problems," *The Annals of Statistics*, pp. 209–230, 1973.

[18] G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle, "On the evaluation of unsupervised outlier detection: Measures, datasets, and an empirical study," *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 891–927, 2016.

[19] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao, "Adbench: Anomaly detection benchmark," in *Neural Information Processing Systems (NeurIPS)*, 2022.

[20] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman, "Pyro: Deep universal probabilistic programming," *Journal of Machine Learning Research*, vol. 20, pp. 973–978, 2019.

INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Zero-Shot Federated Learning with New Classes for Audio Classification

Gautham Krishna Gudur[1], *Satheesh Kumar Perepu*[2]

[1]Global AI Accelerator, Ericsson, Chennai, India
[2]Ericsson Research, Chennai, India
gautham.krishna.gudur@ericsson.com, perepu.satheesh.kumar@ericsson.com

Abstract

Federated learning is an effective way of extracting insights from different user devices while preserving the privacy of users. However, new classes with completely unseen data distributions can stream across any device in a federated learning setting, whose data cannot be accessed by the global server or other users. To this end, we propose a unified zero-shot framework to handle these aforementioned challenges during federated learning. We simulate two scenarios here – 1) when the new class labels are not reported by the user, the traditional FL setting is used; 2) when new class labels are reported by the user, we synthesize *Anonymized Data Impressions* by calculating class similarity matrices corresponding to each device's new classes followed by unsupervised clustering to distinguish between new classes across different users. Moreover, our proposed framework can also handle statistical heterogeneities in both labels and models across the participating users. We empirically evaluate our framework on-device across different communication rounds (FL iterations) with new classes in both local and global updates, along with heterogeneous labels and models, on two widely used audio classification applications – keyword spotting and urban sound classification, and observe an average deterministic accuracy increase of ∼4.041% and ∼4.258% respectively.

Index Terms: keyword spotting, urban sound classification, federated learning, new class identification, zero-shot learning, on-device learning

1. Introduction

Deep learning for audio classification is a broad research area with applications like Keyword Spotting (KWS), urban sound identification, etc. KWS is an important application for detecting keywords of importance to specific users, which could be used as voice commands to on-device personal assistants such as Amazon's Alexa, Apple's Siri, etc. [1]. Urban environment sound classification is another interesting application particularly in context-aware computing, urban informatics [2]. The emergence of deep neural networks have conveniently alleviated problems of creating shallow (hand-picked) features to achieve state-of-the-art performance in such acoustic classification tasks [3, 4]. With the recent compute capabilities vested in resource-constrained devices, there is a huge research focus on audio classification using on-device deep learning [5, 6].

Such applications require characterization of insights across numerous user devices for personalization, and collaborative on-device deep learning becomes necessary. Federated Learning (FL) is a decentralized method of training neural networks by securely sharing model updates with a server without the need to transfer sensitive local user data [7, 8]. On-device federated learning has been an active area of research addressing challenges on secure communication protocols, optimization, privacy-preserving networks, etc. [9, 10]. However, handling new/unseen classes in local devices and training them in an FL setting for the global model to possess characteristics of the new classes is a challenging task, since data transfer from local device to server and vice versa is not feasible. Moreover, the new class information of one user is not known among the other users as well, hence the new classes could be similar or different between the users. In addition, there are multiple statistical heterogeneities like model heterogeneities (ability of end-users to architect their own local models), label heterogeneities and non-IIDness across various communication rounds/FL iterations (disparate data and label distributions across devices).

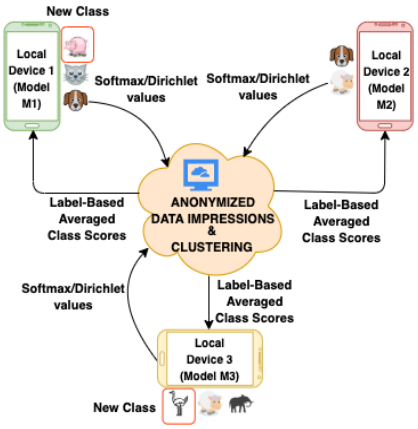


Figure 1: *Architecture of the proposed Federated Learning framework with new classes streaming in across different users.*

One way of handling model heterogeneities and independence in a federated learning setting is by using knowledge distillation [11] with a common student model architecture on each local device [12]. Label and model heterogeneities are handled in an inertial Human Activity Recognition scenario in [13]. Federated learning for keyword spotting, and new class learning and identification in various speech recognition settings are addressed in [14, 15, 16]. A new augmentation technique to reduce false reject rates is proposed in [17], and it addresses algorithmic constraints in FL-KWS training to label examples with no visibility. However, the scope of our proposed work is different in the nature that it primarily addresses identification and similarity detection of new labels in a zero-shot manner when heterogeneous label and model distributions exist across various users and FL iterations. To the best of our knowledge, we are the first to discuss new label identification in FL settings with statistical heterogeneities for audio classification.

Our scientific contributions are: **(1)** A framework with zero-shot learning mechanism by synthesizing *Anonymized Data Impressions* from class similarity matrices to identify new classes for keyword spotting and urban sound detection in

http://dx.doi.org/10.21437/Interspeech.2021-2264

on-device FL settings. **(2)** Provide two scenarios for label acquisition – when class labels are reported by user, and when class labels are not, and propose unsupervised clustering to identify and differentiate between newly reported classes. **(3)** Handling statistical heterogeneities such as heterogeneous distributions in labels, data and models across devices and FL iterations.

2. Our Approach

In this section, we discuss the problem formulation of new classes in FL, and our proposed framework (Algorithm 1) to handle the same. The overall architecture is given in Figure 1.

2.1. Problem Formulation

We assume the following scenario in federated learning. Suppose there are M nodes (devices) in the FL network, holding private local data $\mathcal{D}_i = \{x_{i,j}, y_{i,j}\}$ where i is the FL iteration and j is the user index. Each node consists of public data $\mathcal{D}_0 = \{x_0, y_0\}$. The public data is assumed to be present across the global and all local users as discussed in [12] to handle the various statistical (model) heterogeneities which is a common phenomena in FL. The overall label-set of public dataset is $Y = \{y_0\}$, which represent its unique labels. We re-purpose this public dataset as test set and do not expose it to local models during FL training iterations, but expose only during testing for consistency. Our work's main contribution is to propose a framework to identify new labels across different users without transferring private data in FL setting. We also assume each user can stream data with new labels at any iteration which does not belong to public label-set Y, i.e. $\mathbf{y}_{i,j} \notin Y$. In other words, the global user has no idea of these new labels.

2.2. Anonymized Data Impressions

The main challenge/objective is to detect similar labels across different users in FL heterogeneous settings without the knowledge of local user data. This necessitates us to construct anonymized data without transferring raw sensitive data, and identify new class similarities on the anonymized data. We motivate our framework from the creation of Data Impressions (DI) using zero-shot learning as proposed in [18] to compute *Anonymized Data Impressions*. Let us assume a model $\mathcal{M}$ with input $\mathbf{X}$ and output $\mathbf{y}$, where $\mathbf{X} \in \mathcal{R}^{M \times N}$ is the set of features and $\mathbf{y} \in \mathcal{R}^{M}$. Now, the anonymized feature set $\bar{\mathbf{X}}$, which has same properties of $\mathbf{X}$, can be synthesized in two steps:

(a) Sample Softmax Values: The first step is to sample the softmax values from the Dirichlet distribution [19]. The Class Similarity Matrix (*CSM*) is created which contains important information on how similar the classes are to each other. If the classes are similar, we expect the softmax values are concentrated over these labels. *CSM* is obtained by considering the weights of the model's last layer. Typically, any classification model has the final layer as fully-connected layer with a softmax non-linearity. If the classes are similar, we find similar weights between connections of the penultimate layer to the nodes of the classes [18]. The Class Similarity Matrix is constructed as,

$$C(i,j) = \frac{\mathbf{w}_i^T \mathbf{w}_j}{||\mathbf{w}_i|| ||\mathbf{w}_j||} \tag{1}$$

where $\mathbf{w}_i$ is the vector of weights connecting the previous layer nodes to the class node i. $\mathbf{C} \in \mathcal{R}^{\mathcal{K} \times \mathcal{K}}$ is the Class Similarity Matrix for K classes. We then sample the softmax values as,

$$\text{Softmax} = Dir(K, C) \tag{2}$$

Algorithm 1 Our Proposed Framework

Input: Public Dataset $\mathcal{D}_0\{x_0, y_0\}$, Private Datasets $\mathcal{D}_m^i$, Total users M, Total iterations I, LabelSet l_m for each user, Overall Public LabelSet Y,
Output: Trained Model scores f_G^I
Initialize $f_G^0 = \mathbf{0}$ (Global Model Scores)
for $i = 1$ **to** I **do**
  for $m = 1$ **to** M **do**
    Build: Model $\mathcal{D}_m^i$ and predict $f_{\mathcal{D}_m^i}(x_0)$
    Local Update:
    Choice 1: New classes are not reported
    $f_{\mathcal{D}_m^i}(x_0) = f_G^I(x_0^{l_m}) + \alpha f_{\mathcal{D}_m^i}(x_0)$, where $f_G^I(x_0^{l_m})$ are global scores of l_m with m^{th} user, $\alpha = \frac{len(\mathcal{D}_m^i)}{len(\mathcal{D}_0)}$
    Choice 2: New classes are reported
    Train a new model with $\mathcal{D}_0$ and $\mathcal{D}_m^i$ (new data) together, and send weights of the last layer ($\mathbf{W}_m^i$) to global user.
  end for
  Global Update:
  Choice 1: No user reports new classes
  Update label wise
  $f_G^{i+1} = \sum_{m=1}^{M} \beta_m f_{\mathcal{D}_m^i}(x_0)$, where
  $\beta = \begin{cases} 1 & \text{If labels are unique} \\ \text{acc}(f_{\mathcal{D}_m^{i+1}}(x_0)) & \text{if labels are not unique} \end{cases}$
  where $\text{acc}(f_{\mathcal{D}_m^{i+1}}(x_0))$ is the accuracy metric, defined by the ratio of correctly classified samples to total samples for a given local model.
  Choice 2: Any user reports new classes
  Create *Data Impressions (DI)* for each user m with weights $\mathbf{W}_m^i$ (Section 2.2). Average *DI* of all users with new classes, $\mathbf{X}^i = \sum_{m \in M_{S_k}} \mathbf{X}_m^i$, where M_{S_k} is set of users with new label k.
  Perform *k-medoids clustering* on $\mathbf{X}^i$ across M_{S_k}. Number of clusters = Number of new labels (l_{new}).
  Update public dataset with new DI ($\mathbf{X}^i$), $\mathcal{D}_{new} = \mathcal{D}_0 \bigcup \mathbf{X}^i$, add l_{new} to l_m and Y.
end for

where C is concentration parameter which controls the spread of softmax values over class labels.

(b) Creating Anonymized Data Impressions: Let $\mathbf{Y}^k = [\mathbf{y}_1^k, \mathbf{y}_2^k, \cdots, \mathbf{y}_N^k] \in \mathcal{R}^{K \times N}$ be the N softmax vectors corresponding to class k, sampled from Dirichlet distribution from previous step. Once we obtain the softmax values, we compute the synthesized data features (Data Impressions) by solving the following optimization problem using model $\mathcal{M}$ and sampled softmax values $\mathbf{Y}^k$

$$\bar{\mathbf{x}} = \arg\min_{\mathbf{x}} L_{CE}(\mathbf{y}_i^k, \mathcal{M}(\mathbf{x})) \tag{3}$$

To solve this optimization problem, we initialize the input $\mathbf{x}$ to be random input and iterate until cross-entropy loss (L_{CE}) minimization. This process is repeated for all K categories. In this way, anonymized data impressions are created for each class without the visibility of original input data. We use the TensorFlow framework [20] for all our experiments.

Table 1: *Model Architectures (filters in each layer), Labels and Audio frames per FL iteration across user devices for both datasets. Note the disparate model architectures and labels across users.*

	User 1	User 2	User 3	Global User
Architecture	2-Layer CNN (16, 32) Softmax Activation	3-Layer CNN (16, 16, 32) ReLU Activation	3-Layer Depth-Separable CNN (16, 16, 32) ReLU Activation	–
Keywords	{Yes, No, Up, Down}	{Up, Down, Left, Right}	{Left, Right, On, Off}	{Yes, No, Up, Down, Left, Right, Left, Right, On, Off}
Keyword Frames per iteration	{200-300, 200-300, 200-300, 200-300}	{200-300, 200-300, 200-300, 200-300}	{200-300, 200-300, 200-300, 200-300}	{300*8} = 2400
Sounds	{air conditioner, car horn, children playing}	{children playing, dog bark, drilling }	{drilling, engine idling, gun shot, jackhammer}	{air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer}
Sound Frames per iteration	{40-50, 40-50, 40-50}	{40-50, 40-50, 40-50}	{40-50, 40-50, 40-50, 40-50}	{50*8} = 400

2.3. Proposed Framework

There are three steps in our proposed framework (Algorithm 1).

(a) Build: Each local user creates their own model with their local private data for a specific iteration.

(b) Local Update: In this step, if new classes are not reported, we perform simple weighted α-update [21], where α governs the contributions of new and old models across FL iterations shown in Algorithm 1 Choice 1. If new classes are reported, we train the new class data along with public dataset, and send the new model weights to global user (Choice 2).

(c) Global update: In this step, if no user reports new classes, we perform label-based averaging using the parameter β, which governs contributions of overlapping labels using corresponding test accuracies (Choice 1). If user reports new classes, we create *Anonymized Data Impressions (DI)* for new classes followed by unsupervised clustering using k-medoids with motivations from [22] (Choice 2).

Typically, statistical heterogeneities are widely observed in practical FL settings, hence Choice 1 handles heterogeneities in local and global update steps [13], while Choice 2 handles new classes in our proposed framework.

3. Experiments and Results

We simulate our experiments using *Raspberry Pi 2* as our user device with **Google Speech Commands (GKWS)** [23] and **UrbanSound8K (US8K)** [2] datasets across 10 FL iterations/communication rounds using our proposed framework. In GKWS, we choose the keywords: Yes, No, Up, Down, Left, Right, On, Off, Stop and Go, and perform regular Mel-frequency Cepstral Coefficients (MFCC) extraction as performed in [1], with sampling frequency of 14400 HZ. The MFCC data is divided into 20 windows and each window is of size 50 ms. US8K, an environmental sound dataset, consists of 10 classes of sound events: air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren and street music. We perform similar preprocessing as performed in GKWS for US8K as well.

Public Dataset: We create a Public Dataset (D_0) with 2400 audio frames for GKWS (8 keywords with 300 frames each), and 400 audio frames for US8K (8 sounds with 50 frames each) as shown in Table 1. D_0 is visible to both global and local users in each FL iteration, and is updated with data synthesized for unseen/new classes only – Anonymized Data Impressions.

We initially consider eight labels with an initial Public Dataset in both datasets before streaming new classes (Table 1). We simulate two scenarios for testing just our zero-shot framework – 1) new classes only (homogeneous) with limited users and FL iterations (3 users and 10 FL iterations) for effective analysis of results, 2) new classes with statistical heterogeneities in both labels and models as performed in [13], (10 users and 30 FL iterations). This exhibits near-real-time model heterogeneities as shown in Table 2, and effective convergence.

New Classes: We introduce two new/unseen labels {Stop, Go} for GKWS and {Siren, Street music} for US8K across four FL iterations and two users. In the homogeneous case, for GKWS, we induce 400 samples each with Stop class in iteration 4 for both User 1 and User 2, and 500 samples each with Stop in User 1 iteration 8 and Go in User 2 iteration 8. Similarly, we induce 50 samples each with Siren class in iteration 4 for both User 1 and User 2, and 50 samples each with Siren in User 1 iteration 8 and Street music in User 2 iteration 8. This is the FL scenario with new classes without any heterogeneities. We also discuss similar FL scenarios with statistical heterogeneities.

Table 2: *Heterogeneities in model architectures and new classes changing across FL user iterations for both datasets.*

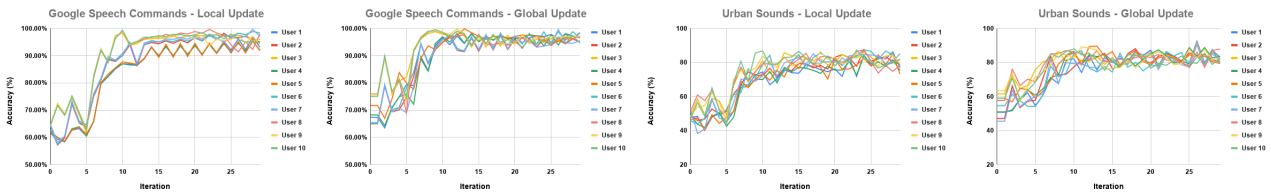
Iteration	New Model	New Class
User 1 Iteration 6	3-Layer ANN (16, 16, 32) ReLU Activation	-
User 1 Iteration 8	1-Layer CNN (16) Softmax Activation	-
User 2 Iteration 4, 6	3-Layer CNN (16, 16, 32) Softmax activation	Stop/Siren
User 3 Iteration 5	4-Layer CNN (8, 16, 16, 32) Softmax activation	-
User 4 Iteration 3, 7	-	Go/Street Music
User 6 Iteration 3, 5	-	Stop/Siren
User 9 Iteration 4	-	Stop/Siren

(a) Label Heterogeneities: In every FL iteration, we consider a random number of audio frames generated between 200-300 samples per label for GKWS, while 40-50 samples per label for US8K. We split these labels across three users such that labels can either be unique or overlapping across users. We also simulate non-IIDness across FL iterations with disparities in both labels and distributions in data (*statistical heterogeneities*).

(b) Model Heterogeneities: We consider the three model architectures as shown in Table 1 motivated from [1, 24], and also change model architectures, filters and activation functions across FL iterations in addition to label heterogeneities with new classes (Table 2). The user iterations are chosen at random.

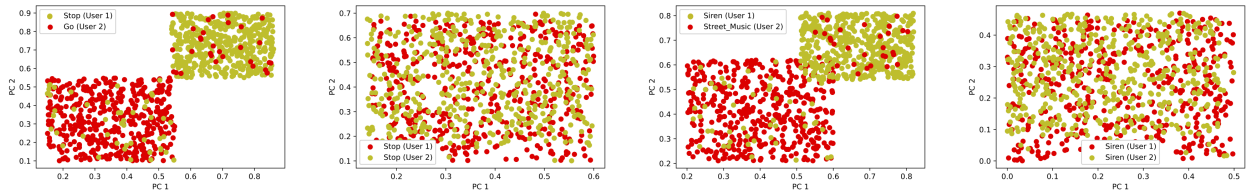
3.1. Discussion on Results

From Table 3, we can observe that there is an accuracy increase in the FL scenario with just new classes (without heterogeneities) in corresponding global updates for all three users than their respective local update accuracies for both datasets in spite of new classes streaming in. The average local-global



(a) (b) (c) (d)

Figure 2: *Local-Global update accuracies (%) across 10 users and 30 FL iterations with new classes and heterogeneities. (a) & (b) show local and global update accuracies respectively for GKWS, while (c) & (d) show these for US8K.*



(a) (b) (c) (d)

Figure 3: *PCA visualizations (with 2 dimensions) after k-medoids clustering with new classes (could be same or different across user devices). (a) & (b) show PCA with different and same classes respectively for GKWS, while (c) & (d) show these for US8K.*

Table 3: *Local-global update accuracies (%) across 3 users and 10 FL iterations, with new classes and without heterogeneities.*

	GKWS			US8K		
User	**Local**	**Global**	**Increase**	**Local**	**Global**	**Increase**
User 1	89.684	93.166	3.482	76.526	80.214	3.688
User 2	91.888	95.28	3.391	75.272	77.944	2.672
User 3	91.517	94.727	3.211	77.61	81.838	4.228
Average	**91.03**	**94.391**	**3.361**	**76.469**	**80**	**3.529**

Table 4: *Final accuracies (%) for both datasets across 10 users and 30 FL iterations, with new classes and heterogeneities.*

Update	**GKWS**	**US8K**
Local	92.5	78.24
Global	96.541	82.498
Increase	**4.041**	**4.258**

accuracy increase across all 10 FL iterations and 3 users is ∼3.361% and ∼3.529% for GKWS and US8K respectively. Similarly, we can also observe that with our proposed framework, the final global accuracies (with convergence after all FL iterations) even with new classes and heterogeneities are 96.541% and 82.498% (Table 4), which are much higher than their respective local update accuracies. The corresponding local-global update accuracies across 30 FL iterations and 10 users are shown in Figure 2. The class similarity matrix with different classes for GKWS is showcased in Figure 4. We can also infer that the clusters effectively formed after performing k-medoids clustering on the new data impressions are equal to the number of new classes, which are visualized using Principal Component Analysis (PCA) in two-dimensions as shown in Figure 3. The new classes can either be different or same across user devices, and these classes are appropriately mapped to the respective end-user devices. The new labels are then finally added back to the overall label set, while the corresponding averaged data impressions are added to the public dataset.

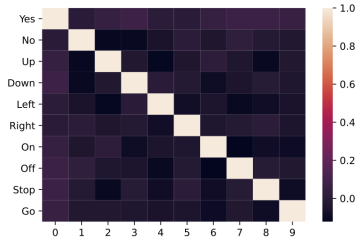


Figure 4: *Class Similarity Matrix for GKWS.*

3.2. On-Device Performance

Raspberry Pi 2 (900MHz quad-core ARM Cortex-A7 CPU with 1GB RAM) is used for evaluating our proposed FL framework as it has similar hardware and software specifications to predominant contemporary IoT/mobile devices. The computation times are identical for both datasets due to similar preprocessing. The training time per epoch for an FL iteration is ∼1.2 sec, while the inference time for one audio sample frame is ∼11 ms. The sizes of the models used are also 520 kB, 350 kB, 270 kB respectively for user architectures mentioned in Table 1.

4. Conclusions

This paper presents a novel framework for handling new labels in a federated learning setting. We propose a zero-shot learning framework by synthesizing Anonymized Data Impressions from Class Similarity matrices to learn new classes across different user devices. We also account for heterogeneities in labels and models across different FL communication rounds, and systematically analyze the results for two widely used audio classification applications – keyword spotting and urban sound classification. We further demonstrate the effectiveness and scalability of our proposed FL framework by simulating our experiments on-device using a Raspberry Pi 2.

5. References

[1] Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello edge: Keyword spotting on microcontrollers," *arXiv preprint arXiv:1711.07128*, 2017.

[2] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *Proceedings of the 22nd ACM International Conference on Multimedia*, 2014, pp. 1041–1044.

[3] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[4] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "Cnn architectures for large-scale audio classification," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 131–135.

[5] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 4087–4091.

[6] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Interspeech*, 2015, pp. 1478–1482.

[7] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," in *Proceedings of Machine Learning and Systems*, vol. 1, 2019, pp. 374–388.

[8] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, vol. 54. PMLR, 2017, pp. 1273–1282.

[9] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, pp. 50–60, 2020.

[10] D. Guliani, F. Beaufays, and G. Motta, "Training speech recognition models with federated learning: A quality/cost framework," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 3080–3084.

[11] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[12] D. Li and J. Wang, "Fedmd: Heterogenous federated learning via model distillation," *arXiv preprint arXiv:1910.03581*, 2019.

[13] G. K. Gudur and S. K. Perepu, "Resource-constrained federated learning with heterogeneous labels and models for human activity recognition," in *Deep Learning for Human Activity Recognition*, vol. 1370, 2021, pp. 57–69.

[14] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, and J. Dureau, "Federated learning for keyword spotting," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6341–6345.

[15] H. Taitelbaum, G. Chechik, and J. Goldberger, "Network adaptation strategies for learning new classes without forgetting the original ones," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3637–3641.

[16] H. Taitelbaum, E. Ben-Reuven, and J. Goldberger, "Adding new classes without access to the original training data with applications to language identification," in *Proc. Interspeech 2018*, 2018, pp. 1808–1812.

[17] A. Hard, K. Partridge, C. Nguyen, N. Subrahmanya, A. Shah, P. Zhu, I. L. Moreno, and R. Mathews, "Training keyword spotting models on non-iid data with federated learning," in *Proc. Interspeech 2020*, 2020, pp. 4343–4347.

[18] G. K. Nayak, K. R. Mopuri, V. Shaj, V. B. Radhakrishnan, and A. Chakraborty, "Zero-shot knowledge distillation in deep networks," in *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 2019, pp. 4743–4751.

[19] T. Minka, "Estimating a dirichlet distribution," 2000.

[20] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.

[21] G. K. Gudur, B. S. Balaji, and S. K. Perepu, "Resource-constrained federated learning with heterogeneous labels and models," *arXiv preprint arXiv:2011.03206*, 2020.

[22] Z. Shuyang, T. Heittola, and T. Virtanen, "Active learning for sound event classification by clustering unlabeled data," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 751–755.

[23] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[24] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1251–1258.

HETEROGENEOUS ZERO-SHOT FEDERATED LEARNING WITH NEW CLASSES FOR AUDIO CLASSIFICATION

Gautham Krishna Gudur
Global AI Accelerator, Ericsson
gautham.krishna.gudur@ericsson.com

Satheesh Kumar Perepu
Ericsson Research
perepu.satheesh.kumar@ericsson.com

ABSTRACT

Federated learning is an effective way of extracting insights from different user devices while preserving the privacy of users. However, new classes with completely unseen data distributions can stream across any device in a federated learning setting, whose data cannot be accessed by the global server or other users. To this end, we propose a unified zero-shot framework to handle these aforementioned challenges during federated learning. We simulate two scenarios here – 1) when the new class labels are not reported by the user, the traditional FL setting is used; 2) when new class labels are reported by the user, we synthesize *Anonymized Data Impressions* by calculating class similarity matrices corresponding to each device's new classes followed by unsupervised clustering to distinguish between new classes across different users. Moreover, our proposed framework can also handle statistical heterogeneities in both labels and models across the participating users. We empirically evaluate our framework on-device across different communication rounds (FL iterations) with new classes in both local and global updates, along with heterogeneous labels and models, on two widely used audio classification applications – keyword spotting and urban sound classification, and observe an average deterministic accuracy increase of $\sim$4.041% and $\sim$4.258% respectively.

1 INTRODUCTION

Deep learning for audio classification is a broad research area with applications like Keyword Spotting (KWS), urban sound identification, etc. KWS is an important application for detecting keywords of importance to specific users, which could be used as voice commands to on-device personal assistants such as Amazon's Alexa, Apple's Siri, etc. (Zhang et al., 2017). Urban environment sound classification is another interesting application particularly in context-aware computing, urban informatics (Salamon et al., 2014). The emergence of deep neural networks have conveniently alleviated problems of creating shallow (hand-picked) features and have achieved state-of-the-art performance in such speech classification tasks (Hinton et al., 2012). With the recent compute capabilities vested in resource-constrained devices, there is a huge research focus on audio classification using on-device deep learning (Chen et al., 2014; Sainath & Parada, 2015).

Such applications require characterization of insights across numerous user devices for personalization, and collaborative on-device deep learning becomes necessary. Federated Learning (FL) is a decentralized method of training neural networks by just securely sharing model updates with a server without the need to transfer sensitive local user data (Bonawitz et al., 2019; McMahan et al., 2017). On-device federated learning has been an active area of research addressing challenges on secure communication protocols, optimization, privacy preserving networks, etc. (Li et al., 2020). However, handling new/unseen classes in local devices and training them in an FL setting for the global model to possess characteristics of the new class is a challenging task, since data transfer from local device to server and vice versa is not feasible. Moreover, the new class information of one user is not known among the other users as well, hence the new classes could be similar or different between the users. In addition, there are multiple statistical heterogeneities like model heterogeneities (ability of end-users to architect their own local models), label heterogeneities and non-IIDness across various communication rounds/FL iterations (disparate data and label distributions across devices).

One way of handling model heterogeneities and independence in a federated learning setting is by using knowledge distillation (Hinton et al., 2015) with a common student model architecture on each local device (Li & Wang, 2019). Label and model heterogeneities are handled in an inertial Human Activity Recognition scenario in (Gudur & Perepu, 2021). Federated learning for keyword spotting (Leroy et al., 2019), and new class learning and identification in various speech recognition settings are addressed in (Taitelbaum et al., 2019; 2018). The paper (Hard et al., 2020) proposes a new augmentation technique to reduce false reject rates and addresses algorithmic constraints in FL-KWS training to label examples with no visibility. However, the scope of our proposed work is different in the nature that it primarily addresses new label identification and similarity detection in a zero-shot manner when heterogeneous label and model distributions exist across various FL iterations and users. To the best of our knowledge, none of the papers discuss new label identification in FL settings with statistical heterogeneities for audio classification.

Scientific contributions: **(1)** A framework with zero-shot learning mechanism by synthesizing *Anonymized Data Impressions* from class similarity matrices to identify new classes for keyword spotting and urban sound detection in on-device FL settings. **(2)** Provide two scenarios for label acquisition – when class label is reported by user, and when class label is not, and propose unsupervised clustering to identify/ differentiate newly reported classes. **(3)** Handling statistical heterogeneities such as heterogeneous distributions in labels/data/models across devices and FL iterations.

2 Our Approach

In this section, we discuss about the problem formulation of new classes and heterogeneities in FL, and our proposed framework (Algorithm 1). The overall architecture is given in Appendix A.

2.1 Problem Formulation:

We assume the following scenario in federated learning. Suppose there are M nodes (devices) in the FL network, holding data with distinct private local data $\mathcal{D}_i = \{x_{i,j}, y_{i,j}\}$ where i is FL iteration and j is the user index. Each node consists of public data $\mathcal{D}_0 = \{x_0, y_0\}$. The public data is assumed to be present across the global and all local users as discussed in (Li & Wang, 2019) to handle the various statistical (model) heterogeneities which is a common phenomena in FL. The overall label-set of public dataset is $Y = \{y_0\}$, which are the unique labels of overall label-set. We re-purpose this public dataset as test set and do not expose it to local models during FL training iterations, but expose only during testing for consistency. Our work's main contribution is to propose a framework to identify new labels across different users without transferring private data in FL setting. We also assume each user can stream data with new labels at any iteration which does not belong to public label-set Y, i.e. $\mathbf{y}_{i,j} \notin Y$. In other words, the global user has no idea of these new labels.

2.2 Anonymized Data Impressions

The main challenge/objective is to identify new classes across different users in FL heterogeneous settings without the knowledge of local user data. This necessitates us to construct anonymized data without transferring raw sensitive data, and identify new class similarities on the anonymized data. We motivate our framework from the creation of Data Impressions (DI) using zero-shot learning as proposed in (Nayak et al., 2019) to compute *Anonymized Data Impressions*. Assume a model $\mathcal{M}$ with input $\mathbf{X}$ and output $\mathbf{y}$, where $\mathbf{X} \in \mathcal{R}^{M \times N}$ is the set of features and $\mathbf{y} \in \mathcal{R}^{M}$. Now, the anonymized feature set $\tilde{\mathbf{X}}$ which has same properties of $\mathbf{X}$ can be synthesized in two steps:

(a) Sample Softmax Values: The first step is to sample the softmax values from the Dirichlet distribution (Minka, 2000). The Class Similarity Matrix (*CSM*) is created which contains important information on how similar the classes are to each other. If the classes are similar, we expect the softmax values are concentrated over these labels. *CSM* is obtained by considering the weights of the model's last layer. Typically, any classification model has the final layer as fully-connected layer with a softmax non-linearity. If the classes are similar, we find similar weights between connections of penultimate layer to the nodes of the classes (Nayak et al., 2019). The Class Similarity Matrix is

Algorithm 1 Our Proposed Framework

Input: Public Dataset $\mathcal{D}_0\{x_0, y_0\}$, Private Datasets $\mathcal{D}_m^i$, Total users M, Total iterations I, LabelSet l_m for each user, Overall Public LabelSet Y
Output: Trained Model scores f_G^I
Initialize $f_G^0 = \mathbf{0}$ (Global Model Scores)
for $i = 1$ **to** I **do**
  for $m = 1$ **to** M **do**
    Build: Model $\mathcal{D}_m^i$ and predict $f_{\mathcal{D}_m^i}(x_0)$
    Local Update:
    Choice 1: New classes are not reported
    $f_{\mathcal{D}_m^i}(x_0) = f_G^I(x_0^{l_m}) + \alpha f_{\mathcal{D}_m^i}(x_0)$, where $f_G^I(x_0^{l_m})$ are global scores of l_m with m^{th} user, $\alpha = \frac{len(\mathcal{D}_m^i)}{len(\mathcal{D}_0)}$
    Choice 2: New classes are reported
    Train a new model with $\mathcal{D}_0$ and $\mathcal{D}_m^i$ (new data) together, and send weights of the last layer ($\mathbf{W}_m^i$) to global user.
  end for
  Global Update:
  Choice 1: No user reports new classes
  Update label wise
  $f_G^{i+1} = \sum_{m=1}^{M} \beta_m f_{\mathcal{D}_m^i}(x_0)$, where
  $\beta = \begin{cases} 1 & \text{If labels are unique} \\ \text{acc}(f_{\mathcal{D}_m^{i+1}}(x_0)) & \text{if labels are not unique} \end{cases}$
  where $\text{acc}(f_{\mathcal{D}_m^{i+1}}(x_0))$ is the accuracy metric, defined by the ratio of correctly classified samples to total samples for a given local model.
  Choice 2: Any user reports new classes
  Create *Data Impressions (DI)* for each user m with weights $\mathbf{W}_m^i$ (Section 2.2). Average *DI* of all users with new classes, $\mathbf{X}^i = \sum_{m \in M_{S_k}} \mathbf{X}_m^i$, where M_{S_k} is set of users with new label k.
  Perform *k-medoids clustering* on $\mathbf{X}^i$ across M_{S_k}. Number of clusters = Number of new labels (l_{new}).
  Update public dataset with new DI ($\mathbf{X}^i$), $\mathcal{D}_{new} = \mathcal{D}_0 \bigcup \mathbf{X}^i$, add l_{new} to l_m and Y.
end for

constructed as,

$$\mathbf{C}(i,j) = \frac{\mathbf{w}_i^T \mathbf{w}_j}{||\mathbf{w}_i|| ||\mathbf{w}_j||} \tag{1}$$

where $\mathbf{w}_i$ is the vector of weights connecting the previous layer nodes to the class node i. $\mathbf{C} \in \mathcal{R}^{\mathcal{K} \times \mathcal{K}}$ is the Class Similarity Matrix for K classes. We then sample the softmax values as,

$$\text{Softmax} = Dir(K, C) \tag{2}$$

where C is concentration parameter which controls the spread of softmax values over class labels.

(b) Creating Anonymized Data Impressions: Let $\mathbf{Y}^k = [\mathbf{y}_1^k, \mathbf{y}_2^k, \cdots, \mathbf{y}_N^k] \in \mathcal{R}^{K \times N}$ be the N softmax vectors corresponding to class k, sampled from Dirichlet distribution from previous step. Once we obtain the softmax values, we compute the synthesized data features (Data Impressions) by solving the following optimization problem using model $\mathcal{M}$ and sampled softmax values $\mathbf{Y}^k$,

$$\bar{\mathbf{x}} = \arg\min_{\mathbf{x}} L_{CE}(\mathbf{y}_i^k, \mathcal{M}(\mathbf{x})) \tag{3}$$

To solve this optimization problem, we initialize the input $\mathbf{x}$ to be random input and iterate until cross-entropy loss (L_{CE}) minimization. This process is repeated for all K categories. In this way, anonymized data impressions are created for each class without the visibility of original input data. We use the TensorFlow framework (Abadi et al., 2016) for all our experiments.

Table 1: Model Architectures (filters in each layer), Labels and Audio frames per FL iteration across user devices for both datasets. Note the disparate model architectures and labels across users.

	User 1	**User 2**	**User 3**	**Global User**
Architecture	2-Layer CNN (16, 32) Softmax Activation	3-Layer CNN (16, 16, 32) ReLU Activation	3-Layer Depth-Separable CNN (16, 16, 32) ReLU Activation	–
Keywords	{Yes, No, Up, Down}	{Up, Down, Left, Right}	{Left, Right, On, Off}	{Yes, No, Up, Down, Left, Right, Left, Right, On, Off}
Keyword Frames per iteration	{200-300, 200-300, 200-300, 200-300}	{200-300, 200-300, 200-300, 200-300}	{200-300, 200-300, 200-300, 200-300}	{300*8} = 2400
Sounds	{air conditioner, car horn, children playing}	{children playing, dog bark, drilling}	{drilling, engine idling, gun shot, jackhammer}	{air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer}
Sound Frames per iteration	{40-50, 40-50, 40-50}	{40-50, 40-50, 40-50}	{40-50, 40-50, 40-50, 40-50}	{50*8} = 400

2.3 Proposed Framework

There are three steps in our proposed FL framework (Algorithm 1).

(a) Build: Each local user creates their own model with their local private data for a specific iteration.

(b) Local Update: In this step, if new classes are not reported, we perform simple weighted α-update (Gudur et al., 2020), where α governs the contributions of new and old models across FL iterations. If new classes are reported, we train the new class data along with public dataset, and send the new model weights to global user.

(c) Global update: In this step, if no user reports new classes, we perform label-based averaging using the parameter β, which governs weightage of overlapping labels using corresponding test accuracies. If user reports new classes, we create *Anonymized Data Impressions (DI)* for new classes followed by unsupervised clustering using k-medoids with motivations from (Shuyang et al., 2017) (Algorithm 1 Choice 2).

Typically, statistical heterogeneities are widely observed in practical FL settings, hence Choice 1 handles heterogeneities in local and global update steps (Gudur & Perepu, 2021), while Choice 2 handles new classes in our proposed framework.

3 Experiments and Results

We simulate our experiments using *Raspberry Pi 2* as our user device with **Google Speech Commands (GKWS)** (Warden, 2018) and **UrbanSound8K (US8K)** (Salamon et al., 2014) datasets (Appendix B) across different FL iterations/commumnication rounds using our proposed framework.

Public Dataset: We create a Public Dataset (D_0) with 2400 audio frames for GKWS (8 keywords with 300 each), and 400 audio frames for US8K (8 sounds with 50 sounds) as shown in Figure 1. D_0 is visible to both global and local users in each FL iteration, and is updated with data synthesized for unseen/new classes only –Anonymized Data Impressions.

We initially consider eight labels with the initial Public Dataset in both datasets before streaming new classes (Table 1). We simulate two scenarios for testing our zero-shot framework - 1) new classes only (homogeneous) with limited users and FL iterations (3 users and 10 iterations) for effective analysis of results, 2) new classes with statistical heterogeneities in both labels and models as performed in Gudur & Perepu (2021), with more users and FL iterations (10 users and 30 iterations) for effective convergence. This exhibits near-real-time statistical heterogeneities as shown in Appendix C Table 3.

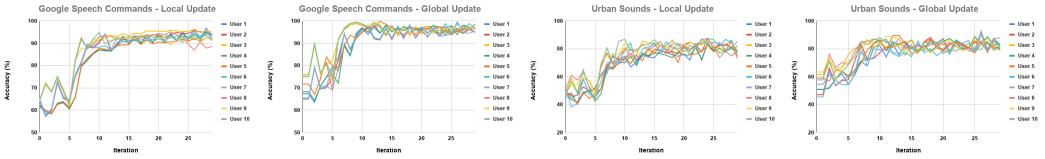
New Classes: We introduce two new/unseen labels {Stop, Go} for GKWS and {Siren, Street music} for US8K across four FL iterations and two users. In the homogeneous case, for GKWS, we induce 400 samples each with Stop class in iteration 4 for both User 1 and User 2, and 500 samples each with Stop in User 1 iteration 8 and Go class in User 2 iteration 8. Similarly, we induce 50 samples each with Siren class in iteration 4 for both User 1 and User 2, and 50 samples each with Siren in User 1 iteration 8 and Street music in User 2 iteration 8. This is the FL scenario with new classes without any heterogeneities. We also discuss similar FL scenarios with statistical heterogeneities.

(a) Label Heterogeneities: In every FL iteration, we also consider a random number of audio frames generated between 200-300 samples per label for GKWS, while 40-50 samples per label for US8K. We split these labels across three users such that labels can either be unique or overlapping

across users. We also simulate non-IIDness across FL iterations with disparities in both labels and distributions in data (*statistical heterogeneities*).

(b) Model Heterogeneities: We consider the three model architectures as shown in Table 1 motivated from (Zhang et al., 2017; Chollet, 2017), and also change model architectures, filters and activation functions over FL iterations in addition to label heterogeneities with new classes (Appendix C Table 3). The FL user iterations for such heterogeneities were chosen at random.

3.1 Discussion on Results



(a) GKWS - Local Update (b) GKWS - Global Update (c) US8K - Local Update (d) US8K - Global Update

Figure 1: Local-Global update accuracies across 10 users and 30 FL iterations for both datasets with new classes and heterogeneities.

Table 2: Final local-global update accuracies (%) with new classes across users and FL iterations.

(a) 3 users, 10 FL iterations without heterogeneities.

	GKWS			US8K		
User	**Local**	**Global**	**Increase**	**Local**	**Global**	**Increase**
User 1	89.684	93.166	3.482	76.526	80.214	3.688
User 2	91.888	95.28	3.391	75.272	77.944	2.672
User 3	91.517	94.727	3.211	77.61	81.838	4.228
Average	**91.03**	**94.391**	**3.361**	**76.469**	**80**	**3.529**

(b) 10 users, 30 FL iterations with heterogeneities.

Update	**GKWS**	**US8K**
Local	92.5	78.24
Global	96.541	82.498
Increase	**4.041**	**4.258**

From Table 2a, we can observe that there is an accuracy increase in FL scenario with just new classes (without heterogeneities) in corresponding global updates for all three users than the respective local update accuracies for both datasets in spite of new classes streaming in. The average local-global accuracy increase across all 10 FL iterations and 3 users is $\sim$3.361% and $\sim$3.529% respectively for GKWS and US8K. Similarly, we can also observe that with our proposed framework, the final global accuracies (with convergence after all FL iterations) even with new classes and heterogeneities are 96.541% and 82.498% (Table 2b) which are much higher than their respective local update accuracies. The corresponding local-global update accuracies across 30 iterations are shown in Figure 1. The class similarity matrix of different classes for GKWS is showcased in Appendix D Figure 3, which elucidates the misclassifications. We can also infer that the clusters effectively formed with k-medoids are equal to number of new classes, which are visualized using Principal Component Analysis (PCA) in two-dimensions. The new classes can either be different or same across user devices (Appendix E Figure 4), and these classes are correctly mapped to the respective end-user devices. The new labels are then finally added to the overall label set while the corresponding averaged data impressions are added to the public dataset. Further, Raspberry Pi 2 performance metrics are observed in Appendix F, showcasing the effectiveness of our proposed FL framework.

4 Conclusion

This paper presents a novel framework for handling new labels in a federated learning setting. We propose a zero-shot learning framework by synthesizing Anonymized Data Impressions from Class Similarity matrices to learn new classes across different user devices. We also account for heterogeneities in labels and models across different communication rounds, and systematically analyze the results for two widely used audio classification applications – keyword spotting and urban sound classification. We further demonstrate the effectiveness and scalability of our proposed FL framework by simulating our experiments on-device using a Raspberry Pi 2.

REFERENCES

Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283, 2016.

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecny, Stefano Mazzocchi, H Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. In *SysML 2019*, 2019.

Guoguo Chen, Carolina Parada, and Georg Heigold. Small-footprint keyword spotting using deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4087–4091. IEEE, 2014.

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

Gautham Krishna Gudur and Satheesh Kumar Perepu. Resource-constrained federated learning with heterogeneous labels and models for human activity recognition. In *Deep Learning for Human Activity Recognition*, pp. 57–69. Springer Singapore, 2021.

Gautham Krishna Gudur, Bala Shyamala Balaji, and Satheesh K Perepu. Resource-constrained federated learning with heterogeneous labels and models. *arXiv preprint arXiv:2011.03206*, 2020.

Andrew Hard, Kurt Partridge, Cameron Nguyen, Niranjan Subrahmanya, Aishanee Shah, Pai Zhu, Ignacio Lopez Moreno, and Rajiv Mathews. Training keyword spotting models on non-iid data with federated learning. In *Proc. Interspeech*, pp. 4343–4347, 2020.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, pp. 82–97, 2012.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

David Leroy, Alice Coucke, Thibaut Lavril, Thibault Gisselbrecht, and Joseph Dureau. Federated learning for keyword spotting. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6341–6345, 2019.

Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37:50–60, 2020.

H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54, pp. 1273–1282, 2017.

Thomas Minka. Estimating a dirichlet distribution, 2000.

Gaurav Kumar Nayak, Konda Reddy Mopuri, Vaisakh Shaj, Venkatesh Babu Radhakrishnan, and Anirban Chakraborty. Zero-shot knowledge distillation in deep networks. In *International Conference on Machine Learning*, pp. 4743–4751, 2019.

Tara N Sainath and Carolina Parada. Convolutional neural networks for small-footprint keyword spotting. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. A dataset and taxonomy for urban sound research. In *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 1041–1044, 2014.

Zhao Shuyang, Toni Heittola, and Tuomas Virtanen. Active learning for sound event classification by clustering unlabeled data. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 751–755, 2017.

Hagai Taitelbaum, Ehud Ben-Reuven, and Jacob Goldberger. Adding new classes without access to the original training data with applications to language identification. In *INTERSPEECH*, pp. 1808–1812, 2018.

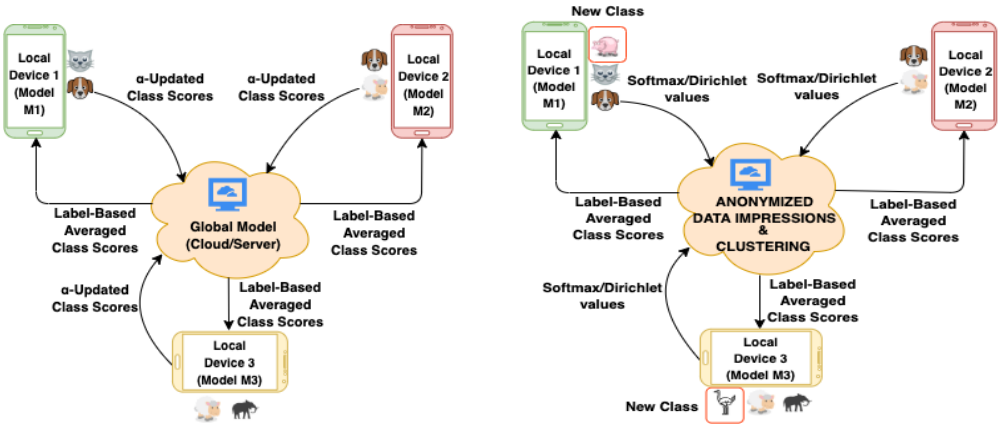
Hagai Taitelbaum, Gal Chechik, and Jacob Goldberger. Network adaptation strategies for learning new classes without forgetting the original ones. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3637–3641, 2019.

Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

Yundong Zhang, Naveen Suda, Liangzhen Lai, and Vikas Chandra. Hello edge: Keyword spotting on microcontrollers. *arXiv preprint arXiv:1711.07128*, 2017.

A Appendix: Overall Architecture

The overall architecture of our proposed framework of new classes identification in a zero-shot manner in FL settings with heterogeneities, along with existing FL scenarios with only heterogeneities (Gudur et al., 2020) is elucidated in Figure 2.



(a) FL with existing classes

(b) FL with new classes

Figure 2: Overall Architecture of FL with existing classes and proposed framework with new classes. Each local device consists of heterogeneous sets of labels and models, and they interact with the global model (cloud/server). When new labels stream in the devices, our proposed zero-shot FL framework in Figure 2b is triggered, else the conventional FL setting in Figure 2a is triggered. The updated consensus is finally distributed across local models and the process continues.

B Appendix: Datasets and Dataset Preprocessing

Google Speech Commands (GKWS) Warden (2018) consists of audio clips of one second and one keyword each by thousands of different people. We choose the keywords: Yes, No, Up, Down, Left, Right, On, Off, Stop and Go, and perform regular Mel-frequency Cepstral Coefficients (MFCC) extraction as performed in (Zhang et al., 2017), with sampling frequency of 14400 HZ. The MFCC data is divided into 20 windows and each window is of size 50 ms.

UrbanSound8K (US8K) (Salamon et al., 2014), an environmental sound dataset, consists of 10 classes of sound events: air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren and street music. All the sounds in the dataset are urban field-recordings. We perform similar preprocessing using MFCC as previously performed in GKWS for US8K as well.

C Appendix: Model Heterogeneities across Iterations

The model and label heterogeneities across and within different FL iterations are observed in Table 3. Changing model architectures, filters and activation functions over FL iterations exhibit near-real-time model heterogeneities. In addition, we also add label heterogeneities with new classes across different FL user iterations.

Table 3: Details of heterogeneities - model architectures (filters) and new classes changing across FL iterations and users for both datasets.

Iteration	New Model	New Class
User 1 Iteration 6	3-Layer ANN (16, 16, 32) ReLU Activation	-
User 1 Iteration 8	1-Layer CNN (16) Softmax Activation	-
User 2 Iteration 4, 6	3-Layer CNN (16, 16, 32) Softmax activation	Stop/Siren
User 3 Iteration 5	4-Layer CNN (8, 16, 16, 32) Softmax activation	-
User 4 Iteration 3, 7	-	Go/Street Music
User 6 Iteration 5, 3	-	Stop/Siren
User 9 Iteration 4	-	Stop/Siren

D Appendix: Class Similarity Matrix

The Class Similarity Matrix calculated from Section 2.2 for the 10 classes of Google Speech Commands Dataset (GKWS) is showcased in Figure 3.

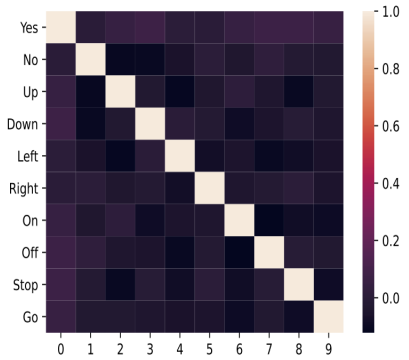
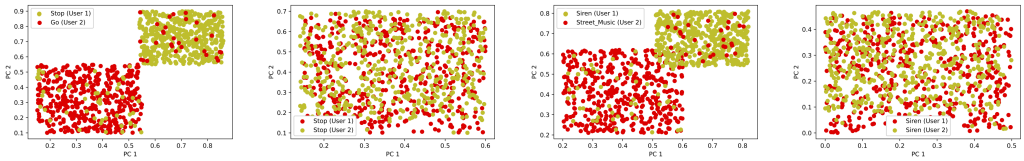


Figure 3: Class Similarity Matrix for Google Speech Commands Dataset.

E APPENDIX: USER REPORTS ON CLUSTERING OF NEW CLASSES

k-medoids unsupervised clustering is performed, and the PCA results (with 2 dimensions) of using new classes which are different across user devices, and also same across user devices are observed in Figure 4. The new classes considered in our experiments are {Stop and Go} for Google Speech Commands dataset, and {Siren and Street music} for UrbanSound8K dataset. The number of clusters returned are the new classes which are correctly mapped to respective end-user devices, and the new labels are added to the overall label set and corresponding data impressions are added to the public dataset. This process is repeated for creating further anonymized data impressions.



(a) GKWS - Different Class (b) GKWS - Same Class (c) US8K - Different Class (d) US8K - Same Class

Figure 4: PCA (with 2 dimensions) of k-medoids unsupervised clustering with new classes, with same and different classes for both datasets.

F APPENDIX: ON-DEVICE PERFORMANCE

Raspberry Pi 2 (900MHz quad-core ARM Cortex-A7 CPU with 1GB RAM) is used for evaluating our proposed FL framework as it has similar hardware and software (HW/SW) specifications to predominant contemporary IoT/mobile devices. The computation times are identical for both datasets due to similar preprocessing. The size of the models used are also 520 kB, 350 kB, 270 kB respectively for user architectures mentioned in Table 1.

Table 4: Computation Times with Raspberry Pi 2

Process	**Time**
Training time per epoch in an FL iteration (i)	$\sim$1.2 sec
Inference time	$\sim$11 ms

Federated Learning with Heterogeneous Labels and Models for Mobile Activity Monitoring

Gautham Krishna Gudur
Global AI Accelerator, Ericsson
gautham.krishna.gudur@ericsson.com

Satheesh Kumar Perepu
Ericsson Research
perepu.satheesh.kumar@ericsson.com

Abstract

Various health-care applications such as assisted living, fall detection etc., require modeling of user behavior through Human Activity Recognition (HAR). Such applications demand characterization of insights from multiple resource-constrained user devices using machine learning techniques for effective personalized activity monitoring. On-device Federated Learning proves to be an effective approach for distributed and collaborative machine learning. However, there are a variety of challenges in addressing statistical (non-IID data) and model heterogeneities across users. In addition, in this paper, we explore a new challenge of interest – to handle *heterogeneities in labels (activities)* across users during federated learning. To this end, we propose a framework for federated label-based aggregation, which leverages overlapping information gain across activities using *Model Distillation Update*. We also propose that federated transfer of model scores is sufficient rather than model weight transfer from device to server. Empirical evaluation with the Heterogeneity Human Activity Recognition (HHAR) dataset (with four activities for effective elucidation of results) on Raspberry Pi 2 indicates an average deterministic accuracy increase of at least $\sim$11.01%, thus demonstrating the on-device capabilities of our proposed framework.

1 Introduction

Human Activity Recognition (HAR) is a technique of significant importance for modeling user behavior in applications like pervasive health monitoring, fitness tracking, fall detection, etc. With the ubiquitous proliferation of personalized sensor-based IoT devices, collaborative and distributed learning is now more possible than ever to help best utilize the behavioral information learnt from multiple users. With the advent of Federated Learning (FL) [1], we can effectively train a centralized model with a federation of users without compromising on sensitive data of users by combining local Stochastic Gradient Descent (SGD) of each local (client) model and aggregating weights on a server, instead of conventionally transferring sensitive data to cloud (*Federated Averaging* algorithm) [18].

On-device federated learning deals with various forms of heterogeneities like device, system, statistical heterogeneities, etc. [15]. Particularly, statistical heterogeneities have gained much research visibility predominantly due to the *non-IID* (non-independent and identically distributed) nature of data from distinct distributions, leading to challenges in personalized federation of devices. An important step in this direction is the ability of end-users to have the choice of architecting their own models, rather than using the pre-defined architectures mandated by the global model. FedMD [14] leverages knowledge distillation [9] to address this, wherein the local models distill their respective knowledge into a *student model* which has a common model architecture. However, as much independence and heterogeneity in architecting the users' own models is ensured in their work, *they do not guarantee heterogeneity and independence in labels across users*.

Machine Learning for Mobile Health Workshop at NeurIPS 2020, Vancouver, Canada.

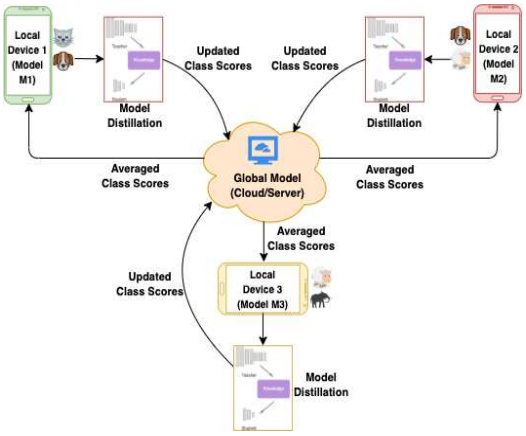


Figure 1: Overall Architecture with *Local Model Distillation*. Each mobile device can consist of disparate set of local labels and models, and they interact with the global model (cloud/server). Each local model is first distilled to a student model, the respective class scores are then aggregated in the global model, and the updated consensus is again distributed across local models.

Many such scenarios with heterogeneous labels and models typically exist in federated IoT settings, such as behaviour/health monitoring, activity tracking, etc. Few works address handling new labels in typical machine learning scenarios, however, to the best of our knowledge, there is no work which addresses this important problem of *label and model heterogeneities* in non-IID federated learning scenarios. More related work on FL, HAR, etc. can be found in Appendix A.

Scientific contributions: **(1)** Enabling end-users to characterize their own preferred local architectures in a federated learning scenario for HAR, so that effective transfer learning and federated aggregation happens between global and local models. **(2)** A framework to allow flexible heterogeneous selection of activities by showcasing scenarios with and without label overlap across different user devices, thereby leveraging the information learnt across devices pertaining to those overlapped activities. **(3)** Empirical demonstration of the framework's ability to handle real-world disparate data/label distributions (non-IID) on simple mobile and wearable devices with a public HAR dataset.

2 Our Approach

Problem Formulation: We assume the following scenario in federated learning. There are multiple local devices which can characterize different model architectures based on end users. We hypothesize that the incoming data to different devices also consist of heterogeneities in activities, with either unique or overlapping activities. We also have a public dataset with the label set consisting of all activities – this can be accessed by any device anytime, and acts as an initial template of the data and labels that can stream through, over different iterations. We re-purpose this public dataset as the test set also, so that consistency is maintained while testing. To make FL iterations independent from the public dataset, we do not expose the public dataset to the local models during learning (training).

Proposed Framework: Our proposed framework to handle heterogeneous labels and models in a federated learning setting is presented in Algorithm 1. There are three important steps here:

- **(a)** **Build**: In this step, we build the model on the incoming data we have in each local user, i.e., local private data for a specific iteration. The users can choose their own model architecture which suits best for the data present in that iteration.
- **(b)** **Local Update (Model Distillation Update)**: In this step, the local models trained on local private data, and distilled to a student model (with public data) with corresponding labels. Distillation acts like a summarization of information captured from previous FL iterations. The global averaged consensus are distributed to the local models before a local update.
- **(c)** **Global update**: In this step, *only the scores of each local model are sent to the server* rather than traditional transfer of weights. The model scores are then averaged with parameter β, where β governs the weightage given to overlapping labels across users using test accuracies of corresponding labels on public data. This module gives the global update scores.

Algorithm 1 Our Proposed Framework

Input: Public Dataset $\mathcal{D}_0\{x_0, y_0\}$, Private Datasets $\mathcal{D}_m^i$, Total users M, Total iterations I, LabelSet l_m for each user
Output: Trained Model scores f_G^I
Initialize $f_G^0 = \mathbf{0}$ (Global Model Scores)
for $i = 1$ **to** I **do**
  for $m = 1$ **to** M **do**
    Build: Model $\mathcal{D}_m^i$ and predict $f_{\mathcal{D}_m^i}(x_0)$
    Local Update (Model Distillation):
    Build a distilled model only on respective local model labels with global averaged probabilities on public dataset D_0. Now, update the model with the new data $\mathcal{D}_m^i$ arriving in this iteration.
  end for
  Global Update: Update label wise
  $f_G^{i+1} = \sum_{m=1}^{M} \beta_m f_{\mathcal{D}_m^i}(x_0)$, where
  $\beta = \begin{cases} 1 & \text{If labels are unique} \\ \text{acc}(f_{\mathcal{D}_m^{i+1}}(x_0)) & \text{if labels are not unique} \end{cases}$
  where $\text{acc}(f_{\mathcal{D}_m^{i+1}}(x_0))$ is the accuracy function of the given model, and is defined by the ratio of correctly classified samples to the total samples for the given local model
end for

3 Experiments and Results

Table 1: Model Architectures (filters in each layer), Labels and Activity Windows per federated learning iteration across user devices. Note the disparate model architectures and labels across users.

	User_1	**User_2**	**User_3**	**Global_User**
Architecture	2-Layer CNN (16, 32) Softmax Activation	3-Layer CNN (16, 16, 32) ReLU Activation	3-Layer ANN (16, 16, 32) ReLU Activation	–
Activities	{Sit, Walk}	{Walk, Stand}	{Stand, StairsUp}	{Sit, Walk, Stand, StairsUp}
Activity Windows per iteration	{2000, 2000} = 4000	{2000, 2000} = 4000	{2000, 2000} = 4000	{2000, 2000, 2000, 2000} = 8000

We simulate a Federated Learning scenario with multiple iterations of incremental data across users' devices. We use the ***Heterogeneity Human Activity Recognition dataset*** [22] for our experiment. More details regarding the dataset and data preprocessing are discussed in Appendix B.

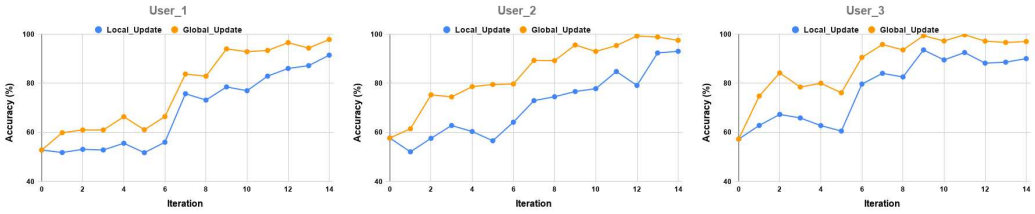
Label Heterogeneities: In our experiment, we consider four activities – {*Sit, Walk, Stand, StairsUp*} across three user devices from the dataset (Table 1). The activities in each local user can either be unique (present only in that single user) or overlapping across users (present in more one user). We split the four activities into three pairs of two activities each for each user, for convenience of showcasing the advantage of overlapping activities. We also create a non-IID environment across different federated learning iterations wherein, the activity data are split with disparities in both the aforementioned labels and distributions in data (*Statistical Heterogeneities*).

Model Heterogeneities: We choose three different model architectures for the three different local users (Table 1). We also use a simple two-layer ANN model with (8, 16) filters as the *distilled student architecture*. To truly showcase near-real-time heterogeneity and model independence, we induce a change in the model architectures across various FL iterations as shown in Appendix C Table 3.

We create a Public Dataset (D_0) with 8000 activity windows, and 2000 activity windows corresponding to each activity. Also, we consider 2000 activity windows per label per iteration in a user (Table 1). In total, we run 15 federated learning iterations in this whole experiment, with each iteration running with early stopping (with a maximum 5 epochs). We track the loss using categorical cross-entropy loss function for classification, and use the Adam optimizer [11] to optimize the classification loss. We simulate all our experiments – both federated learning and inference on a *Raspberry Pi 2*.

3.1 Discussion on Results

In Table 2, *Local Update* and *Global Update* signify the accuracies of each local updated model and the corresponding global updated model (after i^{th} iteration) on Public Dataset respectively. We



(a) User 1 (b) User 2 (c) User 3

Figure 2: Iterations vs *Local Update* and *Global Update* Accuracies across all three users.

can clearly observe that the global updates (governed by β), are higher for all three users than the accuracies of their respective local updates. The average accuracy increase across all iterations from local to global updates is deterministically $\sim$11.01%.

Table 2: Average Accuracies (%) of Local and Global Updates, and their respective Accuracy increase.

	Local Update	**Global Update**	**Accuracy Increase**
User_1	68.38	77.61	9.23
User_2	70.82	84.4	**13.58**
User_3	77.68	87.9	10.22
Average	**72.293**	**83.303**	**11.01**

The significant contribution of overlap in activities towards increase in global accuracies (β-weighted global update information gain) is vividly visible in User 2 (Figure 2b – {*Walk, Stand*}), where, in spite of an accuracy dip in local update at iterations 5 and 12, the global updates at those iterations do not spike down. This can be primarily attributed to the robustness of overlapping label information gain from User 1 (*Walk*) and User 3 (*Stand*) on User 2. On the contrary, for User 3 (Figure 2c), when a dip in local update accuracies are observed at iterations 5 and 8, the global update accuracies in those respective iterations also spike down in a similar fashion. Similar trends in local and global accuracies can also be observed in User 1 (Figure 2a). This also accounts for the highest accuracy increase from local to global updates in User 2 over the rest (Table 2). The final overall global model accuracies averaged across all users after each iteration are also elucidated in Appendix E Figure 3.

3.2 On-Device Performance

Raspberry Pi 2 is used for evaluating our proposed framework as it has similar hardware and software (HW/SW) specifications to predominant contemporary IoT/mobile devices. The computation times taken for execution of our proposed framework are reported in Appendix D Table 4. The distillation mechanism accounts for higher computation time overheads on edge/mobile devices, which depends on the temperature parameter (default set at 1) and the distilled student model architecture chosen.

4 Conclusion

This paper presents a framework for flexibly handling heterogeneous labels and model architectures in federated learning for Human Activity Recognition. We propose a framework with model distillation in local models, and leverage the effectiveness of global model updates with label based averaging (weighted β-update) to obtain higher efficiencies. Moreover, overlapping activities across user devices are found to make our framework robust, and also aid in effective accuracy increase. We also experiment by sending only model scores rather than model weights from user device to server, which reduces latency and memory overheads multifold. We empirically showcase the successful feasibility of our framework on-device, for federated learning across different iterations. We expect a good amount of research focus hereon in handling statistical, model and label based heterogeneities for HAR and other pervasive mobile health tasks.

References

[1] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecny, Stefano Mazzocchi, H Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. In *SysML 2019*, 2019.

[2] Youngjae Chang, Akhil Mathur, Anton Isopoussu, Junehwa Song, and Fahim Kawsar. A systematic study of unsupervised domain adaptation for robust human-activity recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(1):1–30, 2020.

[3] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

[4] Siwei Feng and Marco F Duarte. Few-shot learning-based human activity recognition. *Expert Systems with Applications*, 138:112782, 2019.

[5] Gautham Krishna Gudur, Bala Shyamala Balaji, and Satheesh K Perepu. Resource-constrained federated learning with heterogeneous labels and models. *arXiv preprint arXiv:2011.03206*, 2020.

[6] Gautham Krishna Gudur and Satheesh K Perepu. Resource-constrained federated learning with heterogeneous labels and models for human activity recognition. In *Deep Learning for Human Activity Recognition Workshop, IJCAI*, 2020 [Accepted].

[7] Gautham Krishna Gudur, Prahalathan Sundaramoorthy, and Venkatesh Umaashankar. Activeharnet: Towards on-device deep bayesian active learning for human activity recognition. In *The 3rd International Workshop on Deep Learning for Mobile Systems and Applications*, pages 7–12, 2019.

[8] Nils Y. Hammerla, Shane Halloran, and Thomas Plötz. Deep, convolutional, and recurrent models for human activity recognition using wearables. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 1533–1540. AAAI Press, 2016.

[9] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[10] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[13] Nicholas D. Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, and Fahim Kawsar. An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices. In *Proceedings of the 2015 International Workshop on Internet of Things Towards Applications*, IoT-App '15, pages 7–12, 2015.

[14] Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.

[15] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37:50–60, 2020.

[16] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems 2020*, pages 429–450, 2020.

[17] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 2020.

[18] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54, pages 1273–1282, 2017.

[19] Venet Osmani, Sasitharan Balasubramaniam, and Dmitri Botvich. Human activity recognition in pervasive health-care: Supporting efficient remote collaboration. *Journal of Network and Computer Applications*, 31:628–655, 2008.

[20] Arun Kishore Ramakrishnan, Nayyab Zia Naqvi, Davy Preuveneers, and Yolande Berbers. Federated mobile activity recognition using a smart service adapter for cloud offloading. In *Human Centric Technology and Service in Smart Space*, pages 173–180. Springer, 2012.

[21] Konstantin Sozinov, Vladimir Vlassov, and Sarunas Girdzijauskas. Human activity recognition using federated learning. In *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, pages 1103–1111. IEEE, 2018.

[22] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. Smart devices are different: Assessing and mitigating mobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, SenSys '15, pages 127–140, 2015.

[23] Prahalathan Sundaramoorthy, Gautham Krishna Gudur, Manav Rajiv Moorthy, R. Nidhi Bhandari, and Vineeth Vijayaraghavan. Harnet: Towards on-device incremental learning using deep ensembles on constrained devices. In *Proceedings of the 2nd International Workshop on Embedded and Mobile Deep Learning*, EMDL'18, pages 31–36, 2018.

[24] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek 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*, WWW '17, pages 351–360, 2017.

[25] Shuochao Yao, Yiran Zhao, Huajie Shao, Chao Zhang, Aston Zhang, Shaohan Hu, Dongxin Liu, Shengzhong Liu, Lu Su, and Tarek Abdelzaher. Sensegan: Enabling deep learning for internet of things with a semi-supervised framework. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(3):1–21, 2018.

[26] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

A Appendix: Related Work

Deep learning for HAR, particularly inertial/sensor-based HAR for improving pervasive healthcare has been an active area of research [8, 19]. Particularly, mobile and wearable based deep learning techniques for HAR have proven to be an extremely fruitful area of research with neural network models being able to efficiently run on such resource-constrained devices [24, 13, 23]. Many other challenges in deep learning for HAR tasks have been explored like handling unlabeled data using semi-supervised mechanisms [25, 7], domain adaptation [2], few-shot learning [4] and many more.

Federated Learning has contributed vividly in enabling distributed and collective machine learning across various such devices. Federated learning and differentially private machine learning have, or soon will emerge to become the de facto mechanisms for dealing with sensitive data, data protected by Intellectual Property rights, GDPR, etc. [1]. Federated Learning was first introduced in [18], and new challenges and open problems to be solved [15] and multiple advancements [10] have been proposed and addressed in many interesting recent works.

Multiple device and system heterogeneities making them optimization problems are addressed [16]. Personalized federated learning closely deals with optimizing the degree of personalization and contribution from various clients, thereby enabling effective aggregation as discussed in [3]. Federated learning on the edge with disparate data distributions – non-IID data, and creating a small subset of data globally shared between all devices is discussed in [26].

Particularly for Federated Learning in IoT and pervasive (mobile/wearable/edge) devices, important problems and research directions on mobile and edge networks are addressed in this survey [17], while federated optimization for on-device applications is discussed in [12]. Federated Learning for HAR is addressed in [21] which deals with activity sensing with a smart service adapter, while [20] compares between centralized and federated learning approaches.

FedMD [14], which we believe to be our most closest work, deals with heterogeneities in model architectures, and addresses this problem using transfer learning and knowledge distillation [9], and also uses an initial public dataset across all labels (which can be accessed by any device during federated learning). Current federated learning approaches predominantly handle same labels across all users and do not provide the flexibility to handle unique labels. However, in many practical applications, having unique labels or overlapping labels for each local client/model is a very viable scenario owing to their dependencies and constraints on specific regions, demographics, privacy constrains, etc. A version of the proposed work is discussed for vision tasks in [5] and for HAR tasks in [6].

B Appendix: Dataset and Dataset Preprocessing

The Heterogeneity Human Activity Recognition dataset [22] consists of inertial data from four different mobile phones across nine users performing six daily activities: Biking, Sitting, Standing, Walking, Stairs-Up, Stairs-Down in heterogeneous conditions.

Data Preprocessing: In this experiment, we perform similar preprocessing techniques as stated in [23]. As discussed in [23], we use the mobile phone accelerometer data only and not gyroscope, due to the reduction in data size without substantial accuracy decrease. We initially segment the triaxial accelerometer data into two-second non-overlapping windows and then perform *Decimation* to downsample (normalize) all activity windows to the least sampling frequency (50 Hz). Following this, *Discrete Wavelet Transform (DWT)* is performed for obtaining temporal and frequency information and we use Approximation coefficients only, all together is stated to have a substantial decrease in data size with much loss in information.

C Appendix: Model Heterogeneities across Iterations

The model heterogeneities across and within different iterations are observed in Table 3.

Table 3: Details of Model Architectures (filters in each layer) changed across federated learning iterations and users for both datasets.

Iteration	New Model Architecture
User_1 Iteration_10	3-Layer ANN (16, 16, 32) ReLU Activation
User_1 Iteration_14	1-Layer CNN (16) Softmax Activation
User_2 Iteration_6	3-Layer CNN (16, 16, 32) Softmax activation
User_3 Iteration_5	4-Layer CNN (8, 16, 16, 32) Softmax activation

D Appendix: On-Device Performance

The computation times for on-device performance on Raspberry Pi 2 for our proposed framework are observed in Table 4.

Table 4: Computation Time taken for Execution for HHAR dataset

Process	Computation Time
Training time per epoch in an FL iteration (i)	$\sim$1.7 sec
Inference time	$\sim$15 ms
Discrete Wavelet Transform	$\sim$0.45 ms
Decimation	$\sim$4.6 ms

E Appendix: Overall Global Average Accuracies

The final overall global model accuracies averaged across all users in each iteration is observed in Figure 3.

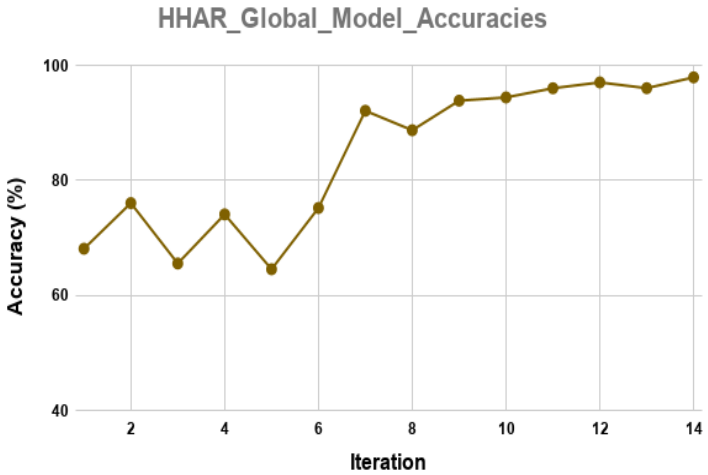


Figure 3: Iterations vs Final Global Average Accuracies (%) with *Local Model Distillation*.

Bayesian Active Learning for Wearable Stress and Affect Detection

Abhijith Ragav*
Solarillion Foundation
abhijithragav@ieee.org

Gautham Krishna Gudur*
Global AI Accelerator, Ericsson
gautham.krishna.gudur@ericsson.com

Abstract

In the recent past, psychological stress has been increasingly observed in humans, and early detection is crucial to prevent health risks. Stress detection using on-device deep learning algorithms has been on the rise owing to advancements in pervasive computing. However, an important challenge that needs to be addressed is handling unlabeled data in real-time via suitable ground truthing techniques (like Active Learning), which should help establish affective states (labels) while also selecting only the most informative data points to query from an oracle. In this paper, we propose a framework with capabilities to represent model uncertainties through approximations in Bayesian Neural Networks using Monte-Carlo (MC) Dropout. This is combined with suitable acquisition functions for active learning. Empirical results on a popular stress and affect detection dataset experimented on a Raspberry Pi 2 indicate that our proposed framework achieves a considerable efficiency boost during inference, with a substantially low number of acquired pool points during active learning across various acquisition functions. Variation Ratios achieves an accuracy of 90.38% which is comparable to the maximum test accuracy achieved while training on about 40% lesser data.

1 Introduction

Psychological stress is encountered by humans in response to physical, mental or emotional challenges presented to them by the environment. Physiological signals can be easily measured using wearable devices and are good indicators of stress, making them suitable for continuous stress monitoring. Deep learning architectures have recently performed well for such tasks, due to their automatic feature extraction capabilities, in contrast to conventional machine learning models which mandate domain knowledge to craft shallow heuristic features [11], [2]. There has been a special interest in bringing deep learning to wearable devices to provide real-time stress and affect monitoring, whilst preserving sensitive user data [13].

Typically, we assume that the physiological user data is inherently labeled, however it is almost practically never the case. One of the least explored areas involving deep learning for such tasks is *Active Learning* – a technique which gives a model, the ability to learn from real-world unlabeled data by querying an oracle (user). By using *Bayesian Neural Networks (BNNs)*, which integrates *Monte Carlo Dropout* with traditional neural networks [4], it is possible to estimate predictive uncertainties. This is coupled with active learning *acquisition functions* for querying the most uncertain data points from the oracle [5]. However, these works have not been considerably discussed in on-device mobile health scenarios, particularly for stress and affect detection. More related work on stress and affect detection, BNNs and active learning can be found in Appendix A.

Scientific contributions: **(1)** A study of Bayesian deep neural networks for real-time stress and affect detection. **(2)** Leveraging the benefits of *Bayesian Active Learning* to model uncertainties, and

*Both authors contributed equally to this work.

Machine Learning for Mobile Health Workshop at NeurIPS 2020, Vancouver, Canada.

exploiting several acquisition functions on-device to instantaneously acquire ground truths (affective states) on-the-fly, thereby substantially reducing the labeling load on oracle.

2 Our Approach

In this section, we discuss in detail our proposed system and approach to perform Active Learning with physiological data.

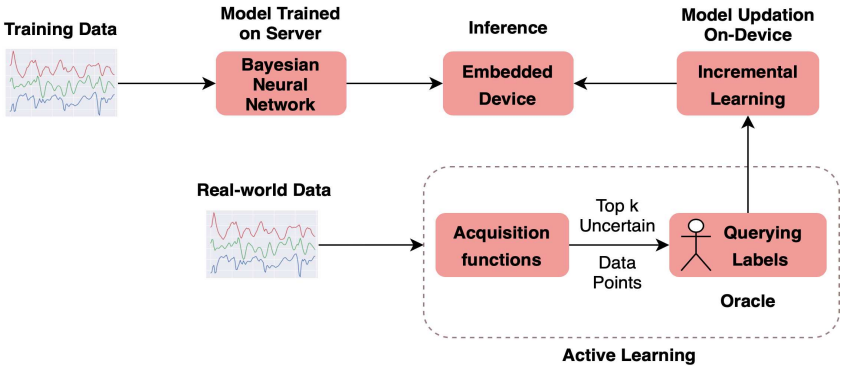


Figure 1: Proposed Architecture

2.1 Background on Modeling Uncertainties

Bayesian Neural Networks (BNNs) offer a probabilistic interpretation to deep learning models by incorporating Gaussian prior (probability distributions) – $p(\omega)$ over our model parameters (weights – ω), thereby modeling output uncertainties. The likelihood model for a classification setting with c classes and x input points is given by,

$$p(y = c|x, \omega) = softmax(f^{\omega}(x))$$

where $f^{\omega}(x)$ is the model output. However, the posterior distribution of BNNs are not easily tractable, hence it becomes computationally intensive for training and inference.

Gal et al. propose that, *Dropout* – a stochastic regularization technique [17], can also perform approximate inference over a deep Gaussian process [4], and thereby learn to model posterior uncertainties without high computational complexities. This is equivalent to performing Variational Inference (VI), where the posterior distribution is approximated by finding another distribution $q^*_\theta(\omega)$, parameterized by θ, within a family of simplified tractable distributions, while minimizing the Kullback-Leibler (KL) divergence between $q^*_\theta(\omega)$ and the true model posterior $p(\omega|D_{train})$.

During inference, we can estimate the mean and variance of the BNN's output by applying dropout before every fully-connected layer during train and test time for multiple stochastic passes (T). This is equivalent to obtaining predictions and uncertainty estimates respectively from the approximate posterior output of the neural network, thereby making the Bayesian NN *non-deterministic* [4]. The predictive distribution for a new data point input x^* can be obtained by,

$$p(y^*|x^*, D_{train}) = \int p(y^*|x^*, \omega)p(\omega|D_{train})d\omega$$

where $p(\omega|D_{train}) = q^*_\theta(\omega)$, and $q^*_\theta(\omega)$ is the dropout distribution approximated using VI. Dropout, being a light-weight operation in most existing NN architectures, enables easier and faster approximation of posterior uncertainties.

2.2 Model Architecture

In our model, 1D-Convolutional (Conv1D) layers are used for effective temporal extraction of features from physiological data. We use a four-layer Conv1D network with 4, 8, 16 and 32 filters, kernel size 3, and a max-pooling layer of size 2 between each Conv1D layer. This is followed by two

Fully-Connected (FC) layers with 32 and 16 neurons each and ReLU activations. An MC-dropout layer with a probability of 0.3 is applied, followed by a softmax layer to obtain the probability scores. The categorical-cross entropy loss of the model is minimized using Adam optimizer.

In order make our model a Bayesian NN so as to obtain uncertainty estimates, we introduce a standard Gaussian prior on the set of our model parameters. Also, to perform approximation inference in our model, we perform dropout at train and test-time as discussed in Section 2.1 to sample from the approximate posterior using multiple stochastic forward passes [4]. After experimenting with multiple dropout iterations (forward passes – *T*), an optimal *T*=10 is utilized in this paper to determine uncertainties. Effectively, our uncertainty-aware model now is a *Bayesian ensembled Convolutional Neural Network (B-CNN)* which can model uncertainties and be used with existing acquisition functions for AL.

2.3 Acquisition functions for Active Learning

As stated in [5], given a classification model M, pool data D_{pool} obtained from real-world, and inputs $x \in D_{pool}$, an acquisition function $a(x, M)$ is a function of x that the active learning system uses to infer the next query point:

$$x^* = argmax_{x \in D_{pool}} a(x, M).$$

Acquisition functions are used in active learning scenarios for approximations in Bayesian CNNs, thereby arriving at the most efficient set of data points to query from D_{pool}. We examine the following acquisition functions to determine the most suitable function for on-device computation:

Max Entropy: Pool points are chosen that maximize the predictive entropy [15].

$$\mathbb{H}[y|x, D_{train}] := -\sum_c p(y = c|x, D_{train}) \log p(y = c|x, D_{train})$$

Bayesian Active Learning by Disagreement (BALD): In BALD, pool points are chosen that maximize the mutual information between predictions and model posterior [8]. The points that maximize the acquisition function are the points that the model finds uncertain on average, and information about model parameters are maximized under the posterior that disagree the most about the outcome.

$$\mathbb{I}[y, \omega|x, D_{train}] = \mathbb{H}[y|x, D_{train}] - E_{p(\omega|D_{train})}\big[\mathbb{H}[y|x, \omega]\big]$$

where $\mathbb{H}[y|x, \omega]$ is the entropy of y, given model weights ω.

Variation Ratios (VR): The LC (Least Confident) method for uncertainty based pool sampling is performed in VR [3].

$$variation - ratio[x] := 1 - \max_y p(y|x, D_{train})$$

Random Sampling: This acquisition function is equivalent to selecting a point from a pool of data points uniformly at random.

3 Experiments and Results

***SWELL-KW Dataset* [10]:** The dataset consists of stress data from 25 participants recorded when they were performing knowledge work tasks such as writing reports, making presentations, reading emails, and searching for information. Stress was induced in the participants by stressors such as time-pressure and email interruptions, and their physiological modalities (Heart Rate and Skin Conductance) were recorded. Based on the participants' self-reports, 3 mental conditions were analyzed – Neutral (N), Interruption (I) and Time-Pressure (T). We report results for the N vs I&T classification task.

The train samples are split in random into D_{train} and D_{pool} points, (30-70 ratio) as an approximation of real-world incoming data, while the unseen D_{test} is used for evaluation purposes only in our experiment. To evaluate our Bayesian CNN framework, we initially pre-train our model for a maximum of 10 epochs on a server to establish the baseline efficiencies on the SWELL dataset, and stock the model in the embedded system. To approximate our posterior and obtain predictive uncertainties, we test our BNN model over *T*=10 stochastic iterations, and average our predictions to calculate our final efficiencies. The average initial baseline accuracy without any active learning incorporated is observed to be 79.12%.

3.1 Bayesian Active Learning

In order to handle ground truth labeling and incorporate model weight updation on incoming test user data, we experiment our proposed framework by deploying the system on a Raspberry Pi 2. The number of acquisition windows used for active learning from D_{pool} can be governed by the **acquisition adaptation factor** $\eta \in [0, 1]$. The incremental model updation is simulated for a maximum of 10 epochs for every acquisition iteration, owing to its non-convergence in loss thereafter.

We analyze various AL acquisition functions mentioned in section 2.3, and observe that Variation Ratios (VR) acquisition function performs the best, while Random Sampling has the least classification accuracy as expected. In VR, only 60% (η=0.6) of the acquisition windows from D_{pool} are required to achieve a test accuracy of 90.38% from a baseline accuracy of 79.12%. This is very close to the maximum test accuracy of 91.92% achieved with η=1.0 (all D_{pool} windows). Note that, η = 0.0 gives the efficiency of the pre-trained model without any data points acquired during incremental learning.

Table 1: *Bayesian-CNN* with *SWELL* – Acquisition Windows (η) vs Accuracy (%).

η	**Max Entropy**	**BALD**	**Variation Ratios**	**Random Sampling**
0.0	79.12	79.12	79.12	79.12
0.2	82.66	81.21	83.11	81.91
0.4	86.43	86.58	88.29	86.76
0.6	89.40	90.22	90.38	88.19
0.8	89.98	90.63	90.82	88.95
1.0	91.92	91.92	91.92	91.92

3.2 On-Device Performance

Raspberry Pi 2 is used for evaluating our proposed active learning framework as it has similar hardware and software specifications to predominant contemporary IoT devices. We observe an average time of about 0.5 sec is utilized for each stochastic forward pass (T) with dropout for acquisition of top η windows. Since we perform T=10 dropout iterations in our experiment, we observe that an average of about 5 seconds are needed for querying the most uncertain data points.

Table 2: Computation Time Taken for Execution per window

Process	**Time**
Inference time	9 ms
Time taken per epoch	0.6 sec

The model size for SWELL is approximately 115 kB, which is substantially small compared to conventional deep learning models. For real-time deployment feasibility, it is practical to have a threshold (upper limit) on the number of D_{pool} collected in a single acquisition iteration. This can be quantified by either number of windows (w_a) or time taken (in seconds).

4 Conclusion

This paper presents empirical contributions with emphasis on Bayesian Active Learning for efficient stress and affect detection. First, we benchmark our efficiencies on the SWELL dataset using a Bayesian-CNN model with stochastic dropout, which incorporates information on predictive uncertainties. Second, we perform active learning on-device using the uncertainty approximations by systematically analyzing various acquisition functions for extracting the most informative data points to be queried by the oracle. This can further be extended to other behavior monitoring tasks in mobile and pervasive healthcare. We also believe this work would trigger more research in under explored areas like handling unlabeled data, and active learning for mobile health scenarios.

References

[1] Sourav Bhattacharya, Petteri Nurmi, Nils Hammerla, and Thomas Plötz. Using unlabeled data in a sparse-coding framework for human activity recognition. *Pervasive and Mobile Computing*, pages 242–262, 2014.

[2] Flavio Di Martino and Franca Delmastro. High-resolution physiological stress prediction models based on ensemble learning and recurrent neural networks. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2020.

[3] Linton C Freeman. *Elementary Applied Statistics: For Students in Behavioral Science*. John Wiley & Sons, 1965.

[4] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1050–1059, 2016.

[5] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 1183–1192, 2017.

[6] Gautham Krishna Gudur, Prahalathan Sundaramoorthy, and Venkatesh Umaashankar. Active-harnet: Towards on-device deep bayesian active learning for human activity recognition. In *The 3rd International Workshop on Deep Learning for Mobile Systems and Applications*, EMDL '19, page 7–12, 2019.

[7] H. M. S. Hossain, N. Roy, and M. A. A. H. Khan. Active learning enabled activity recognition. In *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–9, 2016.

[8] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

[9] Fitri Indra Indikawati and Sri Winiarti. Stress detection from multimodal wearable sensor data. In *IOP Conference Series: Materials Science and Engineering*, page 012028. IOP Publishing, 2020.

[10] Saskia Koldijk, Maya Sappelli, Suzan Verberne, Mark A Neerincx, and Wessel Kraaij. The swell knowledge work dataset for stress and user modeling research. In *Proceedings of the 16th international conference on multimodal interaction*, pages 291–298, 2014.

[11] Dušan Marković, Dejan Vujičić, Dijana Stojić, Željko Jovanović, Uroš Pešović, and Siniša Ranđić. Monitoring system based on iot sensor data with complex event processing and artificial neural networks for patients stress detection. In *18th International Symposium INFOTEH-JAHORINA (INFOTEH)*, pages 1–6. IEEE, 2019.

[12] Rajdeep Kumar Nath, Himanshu Thapliyal, Allison Caban-Holt, and Saraju P Mohanty. Machine learning based solutions for real-time stress monitoring. *IEEE Consumer Electronics Magazine*, 2020.

[13] Abhijith Ragav, Nanda Harishankar Krishna, Naveen Narayanan, Kevin Thelly, and Vineeth Vijayaraghavan. Scalable deep learning for stress and affect detection on resource-constrained devices. In *18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1585–1592. IEEE, 2019.

[14] Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2012.

[15] Claude Elwood Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[16] Yanyao Shen, Hyokun Yun, Zachary Lipton, Yakov Kronrod, and Animashree Anandkumar. Deep active learning for named entity recognition. *Proceedings of the 2nd Workshop on Representation Learning for NLP*, 2017.

[17] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

A Appendix: Related Work

Technological advancements in pervasive and ubiquitous healthcare have drastically improved the quality of human life, and hence has been an actively explored research area. Detecting stress in real-time is crucial for continuous monitoring of mental heath. Deep learning using sensor data has been an evolving domain for computational behavior analysis and healthcare research. Efficient deep learning for stress and affect detection has become the need of the hour, and such approaches have been widely used with promising empirical results by effectively capturing the most discriminative features [12], [9]. However, these works do not address the important problem of concept drift, which must be handled by techniques like Active Learning (AL) to combat changes in data distribution over time.

Conventional AL literature [14] mostly handle low-dimensional data for uncertainty estimations, but do not generalize to deep neural networks as the data is inherently high-dimensional [5]. Deep active learning using uncertainty representations – which presently are the state-of-the-art AL techniques for high-dimensional data, have had very sparse literature. With the advent of Bayesian approaches to deep learning, Gal et al. [5] proposed Bayesian Active Learning for image classification tasks, and is proven to learn from small amounts of unseen data, while Shen et al. [16] incorporate similar techniques for NLP sequence tagging. However, these techniques are predominantly not discussed for sensor and time-series data.

Incorporating AL for obtaining ground truth in mobile sensing systems has been addressed in a few previous works. Most of them have been addressed with Human Activity Recognition (HAR) tasks [1], [6], [7]. Although these works seem to achieve impressive results, the feasibility of on-device learning in mobile health stress and affect detection scenarios with unlabeled data still have not been explored.

Resource-Constrained Federated Learning with Heterogeneous Labels and Models for Human Activity Recognition

Gautham Krishna Gudur[1] and Satheesh Kumar Perepu[2]

[1] Global AI Accelerator, Ericsson
gautham.krishna.gudur@ericsson.com
[2] Ericsson Research
perepu.satheesh.kumar@ericsson.com

Abstract. One of the most significant applications in pervasive computing for modeling user behavior is Human Activity Recognition (HAR). Such applications necessitate us to characterize insights from multiple resource-constrained user devices using machine learning techniques for effective personalized activity monitoring. On-device Federated Learning proves to be an extremely viable option for distributed and collaborative machine learning in such scenarios, and is an active area of research. However, there are a variety of challenges in addressing statistical (non-IID data) and model heterogeneities across users. In addition, in this paper, we explore a new challenge of interest – to handle *heterogeneities in labels (activities)* across users during federated learning. To this end, we propose a framework with two different versions for federated label-based aggregation, which leverage overlapping information gain across activities – one using *Model Distillation Update*, and the other using *Weighted α-update*. Empirical evaluation on the Heterogeneity Human Activity Recognition (HHAR) dataset (with four activities for effective elucidation of results) indicates an average deterministic accuracy increase of at least $\sim$11.01% with the model distillation update strategy and $\sim$9.16% with the weighted α-update strategy. We demonstrate the on-device capabilities of our proposed framework by using Raspberry Pi 2, a single-board computing platform.

Keywords: Human Activity Recognition · On-Device Deep Learning · Federated Learning · Heterogeneous Labels · Heterogeneous Models · Knowledge Distillation.

1 Introduction

Contemporary machine learning, particularly deep learning has led to major breakthroughs in various domains, such as vision, speech, Internet of Things (IoT), etc. Particularly, on-device deep learning has spiked up a huge interest in the research community owing to their automatic feature extraction mechanisms and the compute capabilities vested in resource-constrained mobile and wearable

devices. Sensors embedded in such IoT devices have a vast amount of incoming data which have massive potential to leverage such on-device machine learning techniques on-the-fly to transform them into meaningful information coupled with supervised, unsupervised and/or other learning mechanisms. Human Activity Recognition (HAR) in such personalized IoT devices is a technique of significant importance for our community as it plays a key role in modeling user behavior across a variety of applications like pervasive health monitoring, fitness tracking, fall detection, etc. With the ubiquitous proliferation of such personalized IoT devices, collaborative and distributed learning is now more possible than ever to help best utilize the behavioral information learnt from multiple devices.

However, such collaborative data sharing across devices might always not be feasible owing to privacy concerns from multiple participants. Users might not have any interest in sending their private data to a remote server/cloud, particularly in areas like healthcare. With the advent of *Federated Learning (FL)* [17, 1], it is now possible to effectively train a global/centralized model without compromising on sensitive data of various users by enabling the transfer of model weights and updates from local devices to the cloud, instead of conventionally transferring the sensitive data to the cloud. A server has the role of coordinating between models, however most of the work is not performed by a central entity anymore, but by a federation of clients/devices. The *Federated Averaging (FedAvg)* algorithm was first proposed by McMahan et al. in [17] which combines local Stochastic Gradient Descent (SGD) of each client (local device) with a server that aggregates the model weights. Federated learning has been an active and challenging area of research in solving problems pertaining to secure communication protocols, optimization, privacy preserving networks, etc. [14].

Federated Learning deals with various forms of heterogeneities like device, system, statistical heterogeneities, etc. [14]. Particularly in Federated Learning with IoT scenarios, statistical heterogeneities have gained much visibility as a research problem predominantly owing to the non-IID (non-independent and identically distributed) nature of the vast amounts of streaming real-world data incoming from distinct distributions across devices. This leads to challenges in personalized federation of devices, and necessitates us to address various heterogeneities in data and learning processes for effective model aggregation.

An important step in this direction is the ability of end-users to have the choice of architecting their own models, rather than being constrained by the pre-defined architectures mandated by the global model. One effective way to circumvent this problem is by leveraging the concept of knowledge distillation [8], wherein the disparate local models distill their respective knowledge into various *student models* which have a common model architecture, thereby effectively incorporating model independence and heterogeneity. This was proposed by Li et al. in FedMD [13]. However, as much independence and heterogeneity in architecting the users' own models is ensured in their work, *they do not guarantee heterogeneity and independence in labels across users.*

Many such scenarios with heterogeneous labels and models exist in federated IoT settings, such as behaviour/health monitoring, activity tracking, keyword spotting, next-word prediction, etc. Few works address handling new labels in typical machine learning scenarios, however, to the best of our knowledge, there is no work which addresses this important problem of *label and model heterogeneities* in non-IID federated learning scenarios.

The main scientific contributions in this work are as follows:

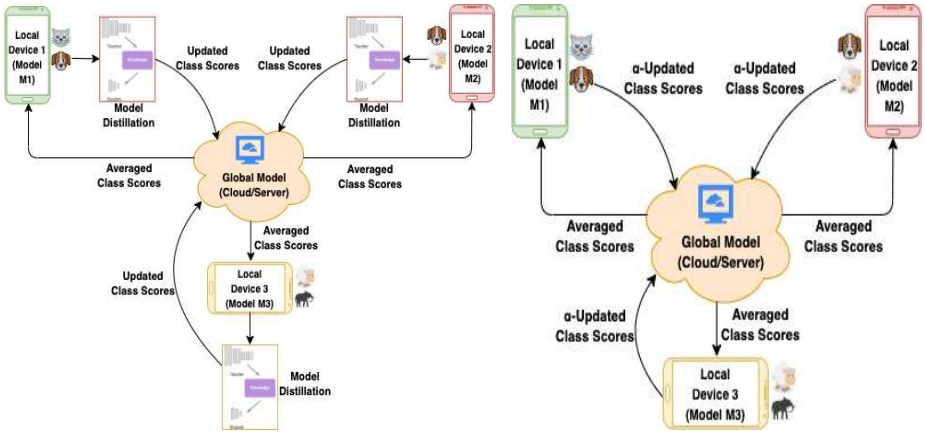
- Enabling end-users to build and characterize their own preferred local architectures in a federated learning scenario for HAR, so that effective transfer learning and federated aggregation happens between global and local models.
- A framework with two different versions to allow flexible heterogeneous selection of activity labels by showcasing scenarios with and without overlap across different user devices, thereby leveraging the information learnt across devices pertaining to those overlapped activities.
- Empirical demonstration of the framework's ability to handle real-world disparate data/label distributions (non-IID) on-device independent of users on a public HAR dataset, capable of running on simple mobile and wearable devices.

2 Related Work

Deep learning for HAR, particularly inertial/sensor-based HAR measured from devices like accelerometer, gyroscope, etc. for improving pervasive healthcare has been an active area of research [7, 18]. Particularly, mobile- and wearable-based deep learning techniques for HAR have proven to be an extremely fruitful area of research with neural network models being able to efficiently run on such resource-constrained devices [23, 12, 22]. Few other challenges with deep learning for HAR have been explored like handling unlabeled data using semi-supervised and active learning mechanisms [24, 6], domain adaptation [2], few-shot learning [4], and many more.

Federated Learning has contributed vividly in enabling distributed and collective machine learning across various such devices. Federated learning and differentially private machine learning have, or soon will emerge to become the de facto mechanisms for dealing with sensitive data, data protected by Intellectual Property rights, GDPR, etc. [1]. Federated Learning was first introduced in [17], and new challenges and open problems to be solved [14] and multiple advancements [9] have been proposed and addressed in many interesting recent works.

Multiple device and system heterogeneities making them optimization problems are addressed in [15]. Personalized federated learning closely deals with optimizing the degree of personalization and contribution from various clients, thereby enabling effective aggregation as discussed in [3]. Federated learning on the edge with disparate data distributions – non-IID data, and creating a small subset of data globally shared between all devices is discussed in [25].



(a) *Model Distillation Update* (b) *Weighted α-update*

Fig. 1: Overall Architecture with both proposed versions. Each device consists of disparate sets of local labels and models, and they interact with the global model (cloud/server). The models in each local device are first updated using one of the two strategies, the respective class scores are then aggregated in the global model, and the updated consensus is again distributed across local models.

Particularly for Federated Learning in IoT and pervasive (mobile or wearable) devices, important problems and research directions on mobile and edge networks are addressed in this survey [16], while federated optimization for on-device applications is discussed in [11]. Federated Learning for HAR is addressed in [20] which deals with activity sensing with a smart service adapter, while [19] compares between centralized and federated learning approaches.

FedMD [13], which we believe to be our most closest work, deals with heterogeneities in model architectures, and addresses this problem using transfer learning and knowledge distillation [8], and also uses an initial public dataset across all labels (which can be accessed by any device during federated learning). Current federated learning approaches predominantly handle same labels across all the users and do not provide the flexibility to handle unique labels. However, in many practical applications, having unique labels for each local client/model is a very viable scenario owing to their dependencies and constraints on specific regions, demographics, privacy constraints, etc. A version of the proposed work is discussed for vision tasks in [5]. However, to the best of our knowledge, none of the works take into account label and model heterogeneities in the context of HAR.

The rest of the paper is organized as follows. Section 3.1 discusses the problem formulation of handling heterogeneous labels and models in on-device federated learning scenarios, and Section 3.2 presents the overall proposed framework and the methods used to address these challenges. Systematic experimentation and evaluation of the framework across different users, devices, iterations, models,

activities in a federated learning setting is showcased in Section 4, while also proving feasibility of the same on resource-constrained devices (Section 4.2). Finally, Section 5 concludes the paper.

3 Our Approach

In this section, we discuss in detail about the problem formulation of heterogeneity in labels and models, and our proposed framework to handle the same (showcased in Figures 1a and 1b).

3.1 Problem Formulation

We assume the following scenario in federated learning. There are multiple local devices which can characterize different model architectures based on the end users. We hypothesize that the incoming data to different devices also consist of heterogeneities in labels, with either unique or overlapping labels. We also have a public dataset with the label set consisting of all labels – this can be accessed by any device anytime, and acts as an initial template of the data and labels that can stream through, over different iterations. We re-purpose this public dataset as the test set also, so that consistency is maintained while testing. To make FL iterations independent from the public dataset, we do not expose the public dataset during learning (training) to the local models. The research problem here is to create a unified framework to handle heterogeneous labels, models and data distributions (non-IID nature) in a federated learning setting.

3.2 Proposed Framework

Our proposed framework to handle heterogeneous labels and models in a federated learning setting is presented in Algorithm 1. There are three important steps in our proposed method.

1. **Build**: In this step, we build the model on the incoming data we have in each local user, i.e., local private data for the specific iteration. The users can choose their own model architecture which suits best for the data present in that iteration.
2. **Local Update**: In this step, we update the averaged global model scores (on public data) for the i^{th} iteration on the local private data. For the first iteration, we do not have any global scores and we initialize the scores to be zero in this case. For the rest of iterations, we have global averaged scores which we can use to update the local model scores according to Algorithm 1. We propose two versions in the local update.
   (a) ***Model Distillation Update***, where the local model is distilled based only on labels corresponding to the local user. Distillation acts a summarization of the information captured from the older models in different FL iterations.

Algorithm 1 Our Proposed Framework (with two version choices)

Input: Public Dataset $\mathcal{D}_0\{x_0, y_0\}$, Private Datasets $\mathcal{D}_m^i$, Total users M, Total iterations I, LabelSet l_m for each user
Output: Trained Model scores f_G^I
Initialize $f_G^0 = \mathbf{0}$ (Global Model Scores)
for $i = 1$ **to** I **do**
  for $m = 1$ **to** M **do**
    Build: Model $\mathcal{D}_m^i$ and predict $f_{\mathcal{D}_m^i}(x_0)$
    Local Update:
    Choice 1 – Model Distillation Update:
    Build a distilled model on only labels corresponding to local user's model with global averaged probabilities on public dataset D_0. Now, update the model with the new data $\mathcal{D}_m^i$ arriving in this iteration.
    Choice 2 – Weighted α-update:
    $f_{\mathcal{D}_m^i}(x_0) = f_G^I(x_0^{l_m}) + \alpha f_{\mathcal{D}_m^i}(x_0)$, where $f_G^I(x_0^{l_m})$ are the global scores of only the set of labels l_m with the m^{th} user, $\alpha = \frac{len(\mathcal{D}_m^i)}{len(\mathcal{D}_0)}$
  end for
  Global Update: Update label wise
  $f_G^{i+1} = \sum_{m=1}^{M} \beta_m f_{\mathcal{D}_m^i}(x_0)$, where
  $\beta = \begin{cases} 1 & \text{If labels are unique} \\ \text{acc}(f_{\mathcal{D}_m^{i+1}}(x_0)) & \text{if labels are not unique} \end{cases}$
  where $\text{acc}(f_{\mathcal{D}_m^{i+1}}(x_0))$ is the accuracy function of the given model, and is defined by the ratio of correctly classified samples to the total samples for the given local model
end for

(b) ***Weighted α-update***, where α is the ratio between the size of current private dataset and the size of public dataset. This parameter governs the contributions of the new and the old models across different FL iterations.

3. **Global update**: In this step, we first train the local model on the respective private datasets for that FL iteration. Further, we evaluate (test) this trained model on the public data, thereby obtaining the model scores on public data. We then perform such label-based averaging across all the users using the β parameter, where β governs the weightage given to unique and overlapping labels across users using test accuracies of the corresponding labels on public data (as given in Algorithm 1). This module gives the global averaged scores.

4 Experiments and Results

We simulate a federated learning scenario with multiple iterations of small chunks of incremental data incoming (details in Table 1), across three different users to test our approach, and assume that the activities arrive in real-time in

Table 1: Model Architectures (filters/units in each layer), Labels (Activities) and Number of Activity Windows per federated learning iteration across user devices. Note the disparate model architectures and labels across users.

	User_1	**User_2**	**User_3**	**Global_User**
Architecture	2-Layer CNN (16, 32) Softmax Activation	3-Layer CNN (16, 16, 32) ReLU Activation	3-Layer ANN (16, 16, 32) ReLU Activation	–
Activities	{Sit, Walk}	{Walk, Stand}	{Stand, StairsUp}	{Sit, Walk, Stand, StairsUp}
Activity Windows per iteration	{2000, 2000} = 4000	{2000, 2000} = 4000	{2000, 2000} = 4000	{2000, 2000, 2000, 2000} = 8000

the users' devices. We use the *Heterogeneity Human Activity Recognition dataset* [21], which consists of inertial data from four different mobile phones across nine users performing six daily activities: Biking, Sitting, Standing, Walking, Stairs-Up, Stairs-Down in heterogeneous conditions.

Data Preprocessing: In this experiment, we perform similar preprocessing techniques as stated in [22]. As discussed, we use the mobile phone accelerometer data only and not gyroscope, due to the reduction in data size without substantial accuracy decrease. We initially segment the triaxial accelerometer data into two-second non-overlapping windows and then perform *Decimation* to downsample (normalize) all activity windows to the least sampling frequency (50 Hz). Following this, Discrete Wavelet Transform (DWT) is performed for obtaining temporal and frequency information and we use Approximation coefficients only, all together is stated to have a substantial decrease in data size.

Now, we discuss the settings for label and model heterogeneities in our experiment.

Label Heterogeneities: In our experiment, we consider only four activities – {*Sit, Walk, Stand, StairsUp*} from the dataset as shown in Table 1. Also, we include the number of activity windows considered per user per iteration (2000 activity windows per iteration). The activities in each local user can either be unique (present only in that single user) or overlapping across users (present in more one user). We split the four activities into three pairs of two activities each, for convenience of showcasing the advantage of overlapping activities in experimentation. We also create a non-IID environment across different federated learning iterations wherein, the activity data across different iterations are split with disparities in both the aforementioned labels and distributions in data (*Statistical Heterogeneities*).

Model Heterogeneities: We choose three different model architectures (CNNs and ANNs) for the three different local users. This is clearly elucidated in Table 1. We also use a simple two-layer ANN model with (8, 16) filters as the *distilled student architecture*. To truly showcase near-real-time heterogeneity and model

Table 2: Details of Model Architectures (filters/units in each layers) changed across federated learning iterations and users.

Iteration	**New Model Architecture**
User_1 Iteration_10	3-Layer ANN (16, 16, 32) ReLU Activation
User_1 Iteration_14	1-Layer CNN (16) Softmax Activation
User_2 Iteration_6	3-Layer CNN (16, 16, 32) Softmax activation
User_3 Iteration_5	4-Layer CNN (8, 16, 16, 32) Softmax activation

independence, we induce a change in the model architectures across and within various FL iterations as shown in Table 2.

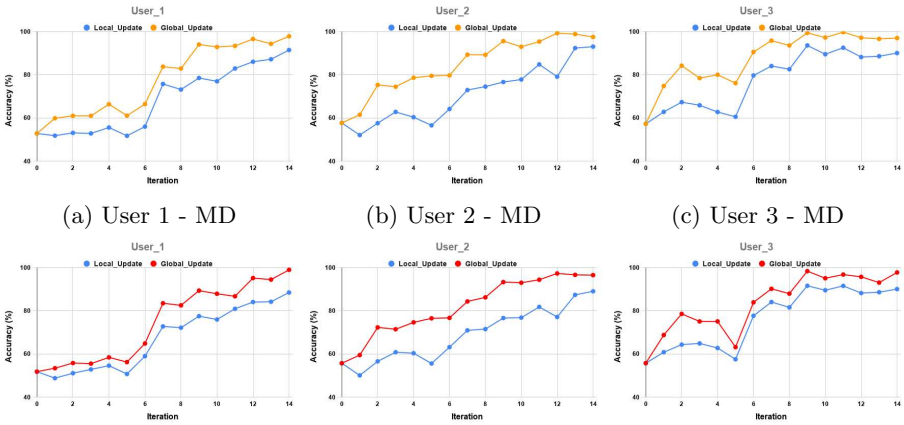
Initially, we divide the activity windows across the three different users according to the four activity labels. We create a Public Dataset (D_0) with 8000 activity windows, with 2000 activity windows corresponding to each activity. Next, we sample 2000 activity windows in every iteration for each label of a user (as shown in Table 1). In total, we ran 15 federated learning iterations in this whole experiment, with each iteration running with early stopping (with a maximum 5 epochs). We track the loss using categorical cross-entropy loss function for multi-class classification, and use the Adam optimizer [10] to optimize the classification loss. We simulate all our experiments – both federated learning and inference on a *Raspberry Pi 2*.

4.1 Discussion on Results

Table 3: Average Accuracies (%) of Local and Global Updates, and their respective Accuracy increase with *Model Distillation Update* and *Weighted α-update*.

	Model Distillation			**Weighted α-update**		
	Local_Update	**Global_Update**	**Increase**	**Local_Update**	**Global_Update**	**Increase**
User_1	68.38	77.61	9.23	66.98	74.29	7.31
User_2	70.82	84.4	13.58	68.88	81.9	13.02
User_3	77.68	87.9	10.22	76.57	83.7	7.13
Average	**72.293**	**83.303**	**11.01**	**70.81**	**79.963**	**9.153**

Figure 2 represents the results across all three users for both proposed versions of our framework on the HHAR dataset. Also, from Table 3, we can clearly



(a) User 1 - MD (b) User 2 - MD (c) User 3 - MD

(d) User 1 - Weighted-α (e) User 2 - Weighted-α (f) User 3 - Weighted-α

Fig. 2: Iterations vs Accuracy across all three users with *Model Distillation Update* (MD) and *Weighted α-update*. *Local_Update* signifies the accuracy of each local updated model (after i^{th} iteration) on Public Dataset. *Global_Update* signifies the accuracy of the corresponding global updated model (averaged across all the users after i^{th} iteration) on Public Dataset.

observe that the global updates – which represent the accuracy of the global updated model (and averaged across all users' labels in the ith iteration governed by β), are higher for all three users than the accuracies of their respective local updates. For instance, from Figures 2a and 2d, we can infer that the corresponding accuracies of labels {*Sit, Walk*} (User 1 labels) after global updates in each iteration are deterministically higher than their respective local updates by an average of ∼9.23% and ∼7.31% across all iterations with model distillation and α-update versions respectively. Similarly for User 2 labels consisting of {*Walk, Stand*}, we observe an average accuracy increase of ∼13.58% and ∼13.02% respectively from local updates to the global updates, while for User 3 labels consisting of {*Stand, StairsUp*}, we observe an average increase of ∼10.22% and ∼7.13% respectively from local updates to global updates in model distillation and α-update versions.

We would like to particularly point out that the overlap in activities significantly contributes to highest increase in accuracies, since information gain (weighted global update) happens only for overlapping labels. This is vividly visible in User 2 (Figure 2b and Figure 2e), whose labels are {*Walk, Stand*}), where, in spite of an accuracy dip in local update at FL iterations 5 and 12, the global update at those iterations do not take a spike down which can be primarily attributed to the information gain from overlapping activity labels between User 1 and User 3 (in this case, *Walk* and *Stand* respectively), thereby showcasing the robustness of overlapping label information gain in User 2. On the contrary, when we observe User 3 (Figure 2c and Figure 2f), in spite of the accuracies of

global updates being inherently better than local updates, when a dip in accuracies of local updates are observed at iterations 5 and 8, the accuracies of global updates at those iterations also spike down in a similar fashion. Similar trends of local and global accuracy trends like those observed in User 3 can also be observed in User 1 (Figure 2a and Figure 2d). This clearly shows that when there are lesser overlapping activity labels (User 1 and User 3), the global model does not learn the activities' characteristics much, while the global updates are more robust in spite of spikes and dips in local updates with such overlapping labels (User 2), thereby leading to higher average increase in accuracies (as observed in Table 3).

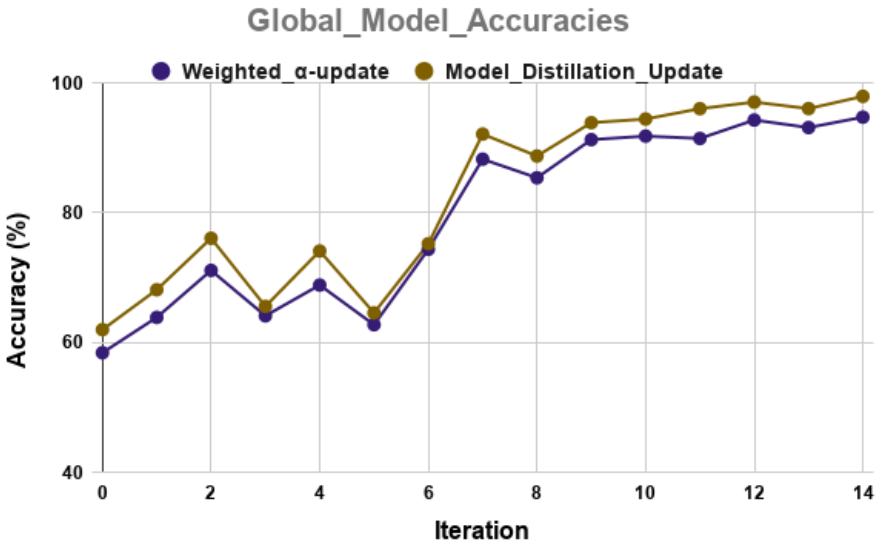


Fig. 3: Iterations vs Final Global Average Accuracies (%) with *Model Distillation Update* and *Weighted α-update*

Overall average deterministic (not relative) increase in accuracies of ∼11.01% and ∼9.153% are observed respectively with the model distillation and α-update versions on the HHAR dataset, which are calculated from the global model updates (Table 3). The overall global model accuracies averaged across all users after each iteration (which is different from global update accuracies after each iteration observed in Figure 2) are also elucidated in Figure 3. We can observe that the distillation version performs better than the α-update version with a ∼3.21% deterministic accuracy increase. With our current framework, *communication (transfer) of just the model scores* of respective activity labels between clients (local devices) and the central cloud is performed, without necessitat-

ing transfer of the entire model weights, which significantly reduces latency and memory overheads.

4.2 On-Device Performance

We observe the on-device performance of our proposed framework by experimenting on a Raspberry Pi 2. We choose this single-board computing platform since it has similar hardware and software (HW/SW) specifications with that of predominant contemporary IoT/mobile devices. The computation times taken for execution of on-device federated learning and inference are reported in Table 4. This clearly shows the feasibility of our proposed system on embedded devices. Also, the distillation mechanism accounts for higher computation overheads in time on edge/mobile devices, and depend on the temperature parameters (default set at 1) and the distilled student model architecture chosen. The end-user can typically make the trade-off of choosing the local distillation version or the α-update version depending on their compute capabilities and accuracy requirements.

Table 4: Time taken for Execution

Process	**Computation Time**
Training time per epoch in an FL iteration (i)	$\sim$1.8 sec
Inference time	$\sim$16 ms
Discrete Wavelet Transform	$\sim$0.45 ms
Decimation	$\sim$4.6 ms

5 Conclusion

This paper presents a unified framework for flexibly handling heterogeneous labels and model architectures in federated learning for Human Activity Recognition (HAR). By leveraging transfer learning along with simple scenario changes in the federated learning setting, we propose a framework with two versions – *Model Distillation Update* and *Weighted α-update* aggregation in local models, and we are able to leverage the effectiveness of global model updates with activity label based averaging across all devices and obtain higher efficiencies. Moreover, overlapping activities are found to make our framework robust, and also helps in effective accuracy increase. We also experiment by sending only model scores rather than model weights from user device to server, which reduces latency and memory overheads multifold. We empirically showcase the successful feasibility of our framework on-device, for federated learning/training across different iterations on the widely used HHAR dataset. We expect a good amount of research focus hereon in handling statistical, model and label based heterogeneities for HAR and other pervasive sensing tasks.

References

1. Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konecny, J., Mazzocchi, S., McMahan, H.B., Overveldt, T.V., Petrou, D., Ramage, D., Roselander, J.: Towards federated learning at scale: System design. In: SysML 2019 (2019)
2. Chang, Y., Mathur, A., Isopoussu, A., Song, J., Kawsar, F.: A systematic study of unsupervised domain adaptation for robust human-activity recognition. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies **4**(1), 1–30 (2020)
3. Deng, Y., Kamani, M.M., Mahdavi, M.: Adaptive personalized federated learning. arXiv preprint arXiv:2003.13461 (2020)
4. Feng, S., Duarte, M.F.: Few-shot learning-based human activity recognition. Expert Systems with Applications **138**, 112782 (2019)
5. Gudur, G.K., Balaji, B.S., Perepu, S.K.: Resource-constrained federated learning with heterogeneous labels and models. arXiv preprint arXiv:2011.03206 (2020)
6. Gudur, G.K., Sundaramoorthy, P., Umaashankar, V.: Activeharnet: Towards on-device deep bayesian active learning for human activity recognition. In: The 3rd International Workshop on Deep Learning for Mobile Systems and Applications. pp. 7–12 (2019)
7. Hammerla, N.Y., Halloran, S., Plötz, T.: Deep, convolutional, and recurrent models for human activity recognition using wearables. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence. pp. 1533–1540. IJCAI'16, AAAI Press (2016)
8. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: NIPS Deep Learning and Representation Learning Workshop (2015)
9. Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al.: Advances and open problems in federated learning. arXiv preprint arXiv:1912.04977 (2019)
10. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
11. Konečnỳ, J., McMahan, H.B., Ramage, D., Richtárik, P.: Federated optimization: Distributed machine learning for on-device intelligence. arXiv preprint arXiv:1610.02527 (2016)
12. Lane, N.D., Bhattacharya, S., Georgiev, P., Forlivesi, C., Kawsar, F.: An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices. In: Proceedings of the 2015 International Workshop on Internet of Things Towards Applications. pp. 7–12. IoT-App '15 (2015)
13. Li, D., Wang, J.: Fedmd: Heterogenous federated learning via model distillation. arXiv preprint arXiv:1910.03581 (2019)
14. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: Challenges, methods, and future directions. IEEE Signal Processing Magazine **37**, 50–60 (2020)
15. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. Proceedings of Machine Learning and Systems 2020 pp. 429–450 (2020)
16. Lim, W.Y.B., Luong, N.C., Hoang, D.T., Jiao, Y., Liang, Y.C., Yang, Q., Niyato, D., Miao, C.: Federated learning in mobile edge networks: A comprehensive survey. IEEE Communications Surveys & Tutorials (2020)
17. McMahan, H.B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.y.: Communication-efficient learning of deep networks from decentralized data. In:

Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. vol. 54, pp. 1273–1282 (2017)
18. Osmani, V., Balasubramaniam, S., Botvich, D.: Human activity recognition in pervasive health-care: Supporting efficient remote collaboration. Journal of Network and Computer Applications **31**, 628–655 (2008)
19. Ramakrishnan, A.K., Naqvi, N.Z., Preuveneers, D., Berbers, Y.: Federated mobile activity recognition using a smart service adapter for cloud offloading. In: Human Centric Technology and Service in Smart Space, pp. 173–180. Springer (2012)
20. Sozinov, K., Vlassov, V., Girdzijauskas, S.: Human activity recognition using federated learning. In: 2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom). pp. 1103–1111. IEEE (2018)
21. Stisen, A., Blunck, H., Bhattacharya, S., Prentow, T.S., Kjærgaard, M.B., Dey, A., Sonne, T., Jensen, M.M.: Smart devices are different: Assessing and mitigating mobile sensing heterogeneities for activity recognition. In: Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems. pp. 127–140. SenSys '15 (2015)
22. Sundaramoorthy, P., Gudur, G.K., Moorthy, M.R., Bhandari, R.N., Vijayaraghavan, V.: Harnet: Towards on-device incremental learning using deep ensembles on constrained devices. In: Proceedings of the 2nd International Workshop on Embedded and Mobile Deep Learning. pp. 31–36. EMDL'18 (2018)
23. Yao, S., Hu, S., Zhao, Y., Zhang, A., Abdelzaher, T.: Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In: Proceedings of the 26th International Conference on World Wide Web. pp. 351–360. WWW '17 (2017)
24. Yao, S., Zhao, Y., Shao, H., Zhang, C., Zhang, A., Hu, S., Liu, D., Liu, S., Su, L., Abdelzaher, T.: Sensegan: Enabling deep learning for internet of things with a semi-supervised framework. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies **2**(3), 1–21 (2018)
25. Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., Chandra, V.: Federated learning with non-iid data. arXiv preprint arXiv:1806.00582 (2018)

ActiveHARNet: Towards On-Device Deep Bayesian Active Learning for Human Activity Recognition

Gautham Krishna Gudur
Global AI Accelerator, Ericsson
gautham.krishna.gudur@ericsson.com

Prahalathan Sundaramoorthy
University of Southern California
prahalat@usc.edu

Venkatesh Umaashankar
Ericsson Research
venkatesh.u@ericsson.com

ABSTRACT

Various health-care applications such as assisted living, fall detection etc., require modeling of user behavior through Human Activity Recognition (HAR). HAR using mobile- and wearable-based deep learning algorithms have been on the rise owing to the advancements in pervasive computing. However, there are two other challenges that need to be addressed: first, the deep learning model should support on-device incremental training (model updation) from real-time incoming data points to learn user behavior over time, while also being resource-friendly; second, a suitable ground truthing technique (like Active Learning) should help establish labels on-the-fly while also selecting only the most informative data points to query from an oracle. Hence, in this paper, we propose *ActiveHARNet*, a resource-efficient deep ensembled model which supports on-device Incremental Learning and inference, with capabilities to represent model uncertainties through approximations in Bayesian Neural Networks using dropout. This is combined with suitable acquisition functions for active learning. Empirical results on two publicly available wrist-worn HAR and fall detection datasets indicate that *ActiveHARNet* achieves considerable efficiency boost during inference across different users, with a substantially low number of acquired pool points (at least 60% reduction) during incremental learning on both datasets experimented with various acquisition functions, thus demonstrating deployment and Incremental Learning feasibility.

CCS CONCEPTS

• **Computing methodologies** → **Active learning settings**; **Neural networks**; • **Human-centered computing** → *Ubiquitous and mobile computing theory, concepts and paradigms.*

KEYWORDS

Human Activity Recognition; Fall Detection; Bayesian Active Learning; On-Device Deep Learning; Incremental Learning

ACM Reference Format:
Gautham Krishna Gudur, Prahalathan Sundaramoorthy, and Venkatesh Umaashankar. 2019. ActiveHARNet: Towards On-Device Deep Bayesian Active Learning for Human Activity Recognition. In *The 3rd International Workshop on Deep Learning for Mobile Systems and Applications (EMDL'19), June 21, 2019, Seoul, Republic of Korea.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3325413.3329790

EMDL'19, June 21, 2019, Seoul, Republic of Korea

ACM ISBN 978-1-4503-6771-4/19/06…$15.00
https://doi.org/10.1145/3325413.3329790

1 INTRODUCTION

Human Activity Recognition (HAR) is an important technique to model user behavior for performing various health-care applications such as fall detection, fitness tracking, health monitoring, etc. The significant escalation in usage of mobile and wearable devices has opened up multiple venues for sensor-based HAR research with various machine learning algorithms. Until recently, machine learning and deep learning algorithms for HAR have been restricted to cloud/server and Graphics Processing Units (GPUs) for obtaining good performance. However, this paradigm has begun to shift with increasing compute capabilities vested in latest smartwatches and mobile phones. Especially, on-device machine learning for monitoring physical activities has been on the rise as an alternative to server-based computes owing to communication and latency overheads [20].

A special interest in bringing deep learning to mobile and wearable devices (incorporating deep learning on the edge) has been an active area of research owing to its automatic feature extraction capabilities, in contrast to conventional machine learning models which mandate domain knowledge to craft shallow heuristic features. One of the unexplored areas involving deep learning for such HAR tasks is *Active Learning* - a technique which gives a model, the ability to learn from real-world unlabeled data by querying an oracle [6]. The integration of Bayesian techniques with deep learning provide us a convenient way to represent model uncertainties by linking *Bayesian Neural Networks (BNNs)* with Gaussian processes using *Dropout* [4] (Section 3.1). These are effectively combined with contemporary deep active learning *acquisition functions* for querying the most uncertain data points from the oracle (Section 3.3).

However, these works have not been considerably discussed in resource-constrained (on-device) HAR scenarios; particularly, during *Incremental Learning* - where a small portion of unseen user data is utilized to update the model, thereby adapting to the new user's characteristics [19]. In this paper, we investigate uncertainty-based deep active learning strategies in HAR and fall detection scenarios for wearable-devices. The main scientific contributions of this paper include:

- A study of a sensor-based light-weight Bayesian deep learning model across various users on wrist-worn heterogeneous HAR and Fall Detection datasets.
- Leveraging the benefits of *Bayesian Active Learning* to model uncertainties, and exploiting several acquisition functions to instantaneously acquire ground truths on-the-fly, thereby substantially reducing the labeling load on oracle.

- Enabling Incremental Learning to facilitate continuous model updation on-device from incoming real-world data independent of users (*User Adaptability*), in turn eliminating the need to retrain the model from scratch.

The rest of the paper is organized as follows. Section 2 presents the related work in the area of deep learning and active learning for HAR. Section 3 discusses about our approach to model uncertainties, the Bayesian *HARNet* model architecture, and the acquisition functions used for querying the oracle. The baseline evaluations for the model in an user-independent scenario on two different datasets are elucidated in Section 4. This is followed by systematic evaluation of the same with the proposed *ActiveHARNet* architecture in resource-constrained incremental active learning scenarios in Section 5.

2 RELATED WORK

Technological advancements in pervasive and ubiquitous healthcare have drastically improved the quality of human life, and hence has been an actively explored research area [13], [10]. Sensor-based deep learning has been an evolving domain for computational behavior analysis and health-care research. Mobile/wearable deep learning for HAR and fall detection [12] among elders have become the need of the hour for patient monitoring; they have been widely used with promising empirical results by effectively capturing the most discriminative features using convolutional [8] and recurrent neural networks, Restricted Boltzmann Machine [1] and other ensembled models [19], [21]. However, these works assume that the incoming streams of data points are labeled in real-time, thereby necessitating ground truthing techniques like active learning to handle unlabeled data.

Conventional Active Learning (AL) literature [14] mostly handle low-dimensional data for uncertainty estimations, but do not generalize to deep neural networks as the data is inherently high-dimensional [5]. Deep active learning using uncertainty representations - which presently are the state-of-the-art AL techniques for high-dimensional data, have had very sparse literature. With the advent of Bayesian approaches to deep learning, Gal et al. [5] proposed Bayesian Active Learning for image classification tasks, and is proven to learn from small amounts of unseen data, while Shen et al. [16] incorporate similar techniques for NLP sequence tagging. However, these techniques are predominantly not discussed for sensor and time-series data.

Incorporating AL for obtaining ground truth in mobile sensing systems has been addressed in few previous works discussed as follows. Hossain et al. [6] incorporate a dynamic k-means clustering approach with AL in HAR tasks. However, this work was before the ubiquitousness of deep learning algorithms, thereby making the model dependent on heuristic hand-picked features. Lasecki et al. [9] discuss about real-time crowd sourcing on-demand to recognize activities using Hidden Markov Models (HMMs) on videos, but not on inertial data. Moreover, deep learning models have vastly outperformed HMMs in video classification setting. Bhattacharya et al. [2] propose a compact and sparse coding framework to reduce the amount of ground truth annotation using unsupervised learning. Although these works seem to achieve impressive results, the feasibility of on-device Incremental Learning (model updation) scenarios with unlabeled data still seems debatable.

In this paper, we propose ***ActiveHARNet***: a unified and novel deep Incremental Active Learning framework for Human Activity Recognition and Fall Detection tasks for ground truthing on resource-efficient platforms, by modeling uncertainty estimates on deep neural networks using Bayesian approximations, thereby efficiently adapting to new user behavior.

3 OUR APPROACH

In this section, we discuss in detail about our proposed *ActiveHARNet* pipeline/architecture (showcased in Figure 1), and our approach to perform Incremental Active Learning.

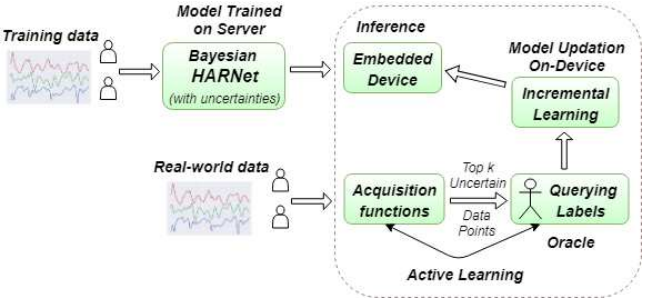


Figure 1: *ActiveHARNet* Architecture

3.1 Background on Modeling Uncertainties

Bayesian Neural Networks (BNNs) offer a probabilistic interpretation to deep learning models by incorporating Gaussian prior (probability distributions) - $p(\omega)$ over our model parameters (weights - ω), thereby modeling output uncertainties. The likelihood model for a classification setting with c classes and x input points is given by,

$$p(y = c|x, \omega) = softmax(f^{\omega}(x))$$

where $f^{\omega}(x)$ is the model output. However, the posterior distribution of BNNs are not easily tractable, hence it becomes computationally intensive for training and inference.

Gal et al. propose that, *Dropout* - a stochastic regularization technique [17], can also perform approximate inference over a deep Gaussian process [4], thereby learn the model posterior uncertainties without high computational complexities. This is equivalent to performing Variational Inference (VI), where the posterior distribution is approximated by finding another distribution $q^*_\theta(\omega)$, parameterized by θ, within a family of simplified tractable distributions, while minimizing the Kullback-Leibler (KL) divergence between $q^*_\theta(\omega)$ and the true model posterior $p(\omega|D_{train})$.

During inference, we can estimate the mean and variance of the BNN's output by applying dropout before every fully-connected layer during train and test time for multiple stochastic passes (T). This is equivalent to obtaining predictions and uncertainty estimates respectively from the approximate posterior output of the neural network, thereby making the Bayesian NN *non-deterministic* [4]. The predictive distribution for a new data point input x^* can

be obtained by,

$$p(y^*|x^*, D_{train}) = \int p(y^*|x^*, \omega)p(\omega|D_{train})d\omega$$

where $p(\omega|D_{train}) = q^*_\theta(\omega)$, and $q^*_\theta(\omega)$ is the dropout distribution approximated using VI. Dropout, being a light-weight operation in most existing NN architectures, enables easier and faster approximation of posterior uncertainties.

3.2 Model Architecture

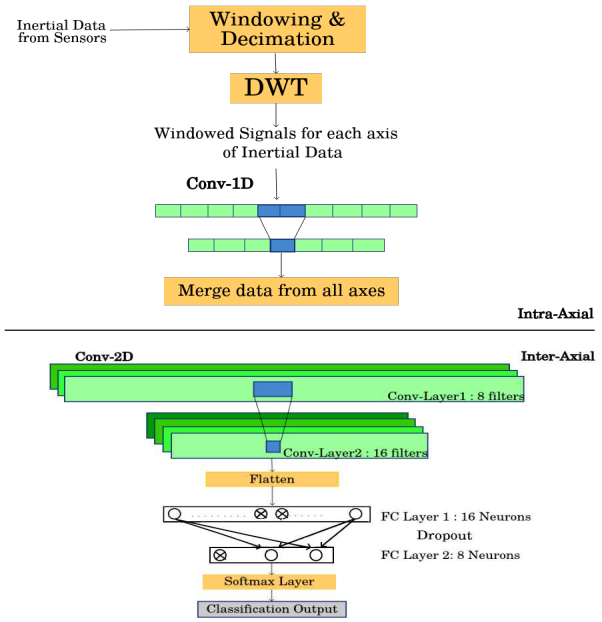


Figure 2: *HARNet Architecture*

To extract discriminative features from inertial data, a combination of local features and spatial interactions between the three axes of the accelerometer can be exploited, using a combination of convolutional 1D and 2D layers.

Intra-Axial dependencies are captured using a two-layer stacked convolutional 1D network, with 8 and 16 filters each and kernel size 2. Batch normalization is performed followed by a max-pooling layer of size 2.

Inter-axial dependencies from the concatenated intra-axial features are captured using a two-layer stacked 2-D CNN network comprising of 8 and 16 filters each with receptive field size 3x3, followed by batch normalization and a max pooling layer of size 3x2.

This is followed by two Fully-Connected (FC) layers with 16 and 8 neurons each, with weight regularization (L2-regularizer with a weight decay constant), and ReLU activations. A dropout layer with a probability of 0.3 is applied, followed by a softmax layer to get the probability estimates (scores). The categorical-cross entropy loss of the model is minimized using Adam optimizer with a learning rate of $2e^{-4}$ and implemented using the TensorFlow framework. We choose this *HARNet* architecture [19] with extensive parametric optimization, as it is a state-of-the-art architecture for heterogeneous HAR tasks by taking into account the efficiency, model size and inference times, with extremely less parameters (~31,000 parameters) compared to contemporary deep learning architectures.

In order make *HARNet* (Figure 2) a Bayesian NN so as to obtain uncertainty estimates, we introduce a standard Gaussian prior on the set of our model parameters. Also, to perform approximation inference in our model, we perform dropout at train and test-time as discussed in Section 3.1 to sample from the approximate posterior using multiple stochastic forward passes (*MC-dropout*) [4]. After experimenting with multiple dropout iterations (forward passes - T), an optimal T=10 in utilized this paper to determine uncertainties. Effectively, *HARNet is a Bayesian ensembled Convolutional Neural Network (B-CNN)* which can model uncertainties, which can be used with existing acquisition functions for AL.

3.3 Acquisition functions for Active Learning

As stated in [5], given a classification model M, pool data D_{pool} obtained from real-world, and inputs $x \in D_{pool}$, an acquisition function $a(x, M)$ is a function of x that the active learning system uses to infer the next query point:

$$x^* = argmax_{x \in D_{pool}} a(x, M).$$

Acquisition functions are used in active learning scenarios for approximations in Bayesian CNNs, thereby arriving at the most efficient set of data points to query from D_{pool}. We examine the following acquisition functions to determine the most suitable function for on-device computation:

3.3.1 Max Entropy. Pool points are chosen that maximize the predictive entropy [15].

$$\mathbb{H}[y|x, D_{train}] := -\sum_c p(y = c|x, D_{train}) \log p(y = c|x, D_{train})$$

3.3.2 Bayesian Active Learning by Disagreement (BALD). In BALD, pool points are chosen that maximize the mutual information between predictions and model posterior [7]. The points that maximize the acquisition function are the points that the model finds uncertain on average, and information about model parameters are maximized under the posterior that disagree the most about the outcome.

$$\mathbb{I}[y, \omega|x, D_{train}] = \mathbb{H}[y|x, D_{train}] - E_{p(\omega|D_{train})}\big[\mathbb{H}[y|x, \omega]\big]$$

where $\mathbb{H}[y|x, \omega]$ is the entropy of y, given model weights ω.

3.3.3 Variation Ratios (VR). The LC (Least Confident) method for uncertainty based pool sampling is performed in VR [3].

$$variation - ratio[x] := 1 - \max_y p(y|x, D_{train})$$

3.3.4 Random Sampling. This acquisition function is equivalent to selecting a point from a pool of data points uniformly at random.

4 BASELINE EVALUATION

To evaluate our Bayesian *ActiveHARNet* framework on an embedded platform, we experiment and analyze the results on two wrist-worn public datasets, which were performed across multiple users in real-world. Datasets with multiple users were rigorously selected to exhibit the capabilities of *ActiveHARNet* like incremental active learning and user adaptability. This is essentially the pre-training phase where the model is stocked in the embedded system. To approximate our posterior with predictive uncertainties, we test our BNN model over T=10 stochastic iterations, and average our predictions to calculate our final efficiencies on both datasets. Also, each model is trained for a maximum of 50 epochs to establish the baseline efficiencies, as we observe that loss saturates extensively and does not converge after the same.

4.1 Heterogeneous Human Activity Recognition (HHAR) Smartwatch Dataset

HHAR dataset, proposed by Allan et al. [18], contains 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.

Data Preprocessing. As performed in [19], we first segment the raw inertial accelerometer data into two-second non-overlapping windows. To handle disparity in sampling frequencies across devices, we perform *Decimation* - a down-sampling technique on all windows, to the least sampling frequency (100 Hz - Samsung Galaxy Gear) to obtain uniform distribution in data. Hence, the size of each window (w_a) is 200. Further, to obtain temporal and frequency information, we perform *Discrete Wavelet Transform (DWT)* and take only the Approximate coefficients. Note that, performing such operations compress the size of the sensor data by more than ~50%. Also, we utilize only accelerometer data and not gyroscope, since the former reduces the size of the dataset by half without compromising much on accuracy.

Initially, we benchmark our baseline accuracies on the server using Bayesian *HARNet* with the *Leave-One-User-Out (LOOCV)* strategy. The test user samples are split in random into D_{test} and D_{pool} points, with D_{pool} slightly higher than D_{test} (70-30 ratio) as an approximation of real-world incoming data, while the unseen D_{test} is always used for evaluation purposes only in our experiment for both datasets. The exact number of D_{pool} and D_{test} differs with various users, and is subjective in real-time. The average accuracy using *LOOCV* is observed to be ~61%. Also, from Figure 3, we can infer that the model performs the best on user 'd' data with ~84% classification accuracy, while the classification accuracies of user 'i' and user 'g' are the least with ~25% and ~36% respectively. These disparate changes in accuracies can be attributed to the unique execution style of activities by the users.

4.2 Notch Wrist-worn Fall Detection Dataset

This dataset uses an off-the-shelf *Notch* sensor as performed in [11] (only the wrist-worn accelerometer data is used). The dataset is collected by seven volunteers across various age groups performing simulated falls and activities (activities are termed as not-falls).

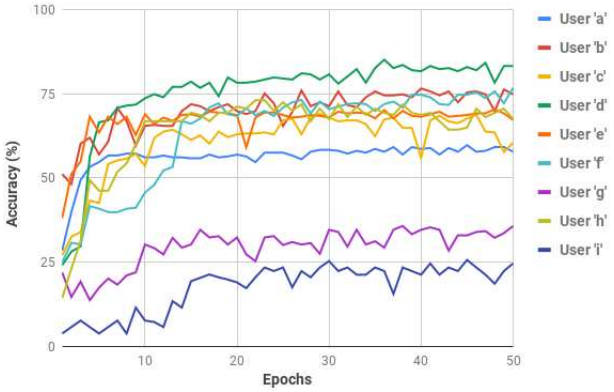


Figure 3: Baseline Accuracy (%) vs Epochs on *HHAR* for Users 'a' through 'i'

Data Preprocessing. We segment the raw inertial data of each activity into non-overlapping windows with the given standardized sampling frequency of 31.25 Hz as in [11], hence there is no decimation performed here. Further, similar to the *HHAR* dataset preprocessing, we perform *Discrete Wavelet Transform (DWT)* to obtain temporal and frequency information of the sensor data and take only the Approximate coefficients. Note that performing these operations compress the size of the sensor data by more than ~50%.

From Table 1, we can observe the f1-scores and accuracies with *LOOCV* strategy on the Bayesian *HARNet*. f1-score would be a better estimate for handling the imbalance in falls and activities, since fall is the rare case event in this binary classification setting. A similar randomized D_{test} and D_{pool} split (70-30 ratio) is performed, and the average f1-score on D_{test} using Bayesian *HARNet* is found to be 0.927, which is substantially higher than [11], whose best-performing deep learning model's f1-score is calculated to be 0.837 from its precision and recall scores.

Table 1: Baseline f1-scores and Accuracies on *Notch*

	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

5 INCREMENTAL ACTIVE LEARNING

In order to handle ground truth labeling and incorporate model weight updation on incoming test user data, we experiment incremental active training with *LOOCV* on both datasets by deploying the system on a Raspberry Pi 2. We choose this single-board computing platform, since it has similar hardware and software specifications with predominant contemporary wearable devices. The number of acquisition windows used for incremental active training from D_{pool} can be governed by the *acquisition adaptation factor* $\eta \in [0, 1]$. The incremental model updation was simulated for a maximum of 10 epochs, owing to its non-convergence in loss thereafter.

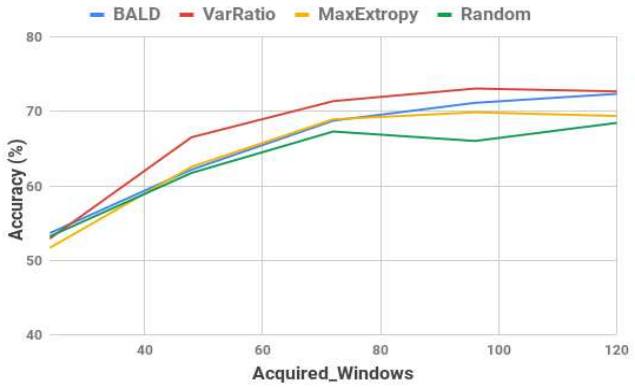


Figure 4: *ActiveHARNet* on *HHAR* for User 'i' across acquisition functions - Accuracy (%) vs Acquisition Windows

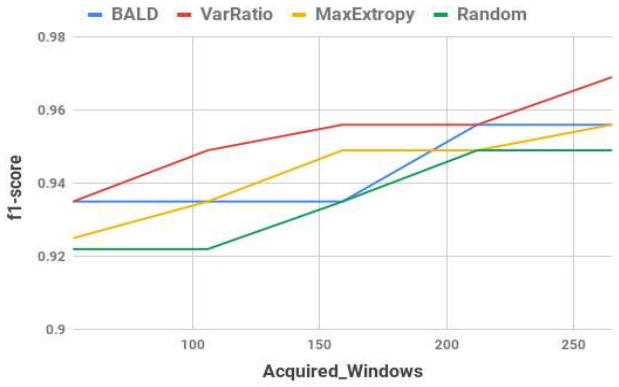


Figure 5: *ActiveHARNet* on *Notch* for User 5 across acquisition functions - f1-scores vs Acquisition Windows

5.1 ActiveHARNet on HHAR dataset

We analyze various AL acquisition functions mentioned in section 3.3 for all users, particularly for the worst performing user 'i' (least classification accuracy), and we can infer from Figure 4 that Variation Ratios (VR) acquisition function performs the best, while Random Sampling has the least classification accuracy as expected. In VR, only 62 (η=0.5 or 50%) acquisition windows from the total D_{pool} (123 windows) were required for user 'i' to achieve a test accuracy of ~70% from a baseline accuracy of ~25%, which is a substantial increase of ~45% in test accuracy. With η=1.0 (all D_{pool} windows), a maximum of ~73% is achieved.

We test *ActiveHARNet* using VR across all users, and observe that the average baseline accuracy (η=0) increases from 61% to a maximum of ~86% (η=1) (Table 2). Also, very few acquisition windows (η=0.4, accuracy=83.05%) from incoming D_{pool} are found to be sufficient for achieving competitive efficiencies (85.87%) as η=1.0. Note that, η=0.0 gives the efficiency of the pre-trained model without any data points acquired during incremental learning.

Table 2: *ActiveHARNet* on *HHAR* with Variation Ratios for all users - Acquisition Windows (η) vs Accuracy (%)

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

5.2 ActiveHARNet on Notch dataset

We analyze AL acquisition functions for all users as performed in Section 5.1, and showcase the f1-scores of user 5 (least performing) in Figure 5. Variation Ratios (VR) has the highest classification efficiency again, requiring only 150 acquisition windows (η=0.4) for a competitive f1-score of 0.956 from the total D_{pool} of 265 windows (η=1.0), whose f1-score is 0.969. Random acquisition performs with the least f1-score again.

Also, from Table 3, VR across all users yield average f1-scores of 0.943 and 0.948 for η=0.4 and η=0.6 respectively, from a baseline 0.928 (η=0.0), thus scaling well to new users with substantially less data points.

Table 3: *ActiveHARNet* on *Notch* with Variation Ratios for all users - Acquisition Windows (η) vs f1-scores

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

5.3 On-Device Incremental Learning

Raspberry Pi 2 is used for evaluating incremental active learning, and an average time of ~1.4 sec is utilized for each stochastic forward pass (T) with dropout for acquisition of top η windows. Since we perform T=10 dropout iterations in our experiment, we observe that an average of ~14 seconds are needed for querying most uncertain data points. Variation Ratios was found to be converging slightly faster with better test accuracies than BALD and MaxEntropy acquisition functions on both datasets, while Random Sampling converged relatively slower. Also, there is substantial increase in relative HHAR efficiencies during incremental active learning than that of Notch. This can be attributed to the nature of the dataset, and the different ways people perform activities. Notch, being a fall detection problem with two classes, has a higher probability of better recognition efficiencies, than a multi-class HHAR problem.

Table 4: Computation Time Taken for Execution per window

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

The model size for HHAR was ∼ 315 kB, while for Notch, it was found to be ∼180 kB, which is substantially small compared to conventional deep learning models. For real-time deployment feasibility, it is practical to have a threshold (upper limit) on the number of D_{pool} collected at a single point of time. This can be quantified by either number of windows (w_a) or time taken (in seconds). We propose time as a benchmark (for instance, 15 minutes since start of an activity cycle; the setting could be personalized based on user), so that the oracle efficiently remembers the recently performed activities when queried for ground truth. Many such acquisition iterations, in turn, model updations would ideally happen during real-time, and our experiments showcased the practical feasibility of one such acquisition iteration with ∼14 seconds. The end-user can also have a trade-off between efficiency and model updation time, and this is proportional to the number of data points to be queried by the oracle.

6 CONCLUSION

This paper presents three new empirical contributions with emphasis on Bayesian Active Learning for embedded/wearable deep learning on HAR and fall detection scenarios. First, we benchmark our efficiencies for both datasets using the Bayesian *HARNet* model, which can incorporate uncertainties. Second, we systematically analyze various acquisition functions for active learning and exploit Bayesian Neural Networks with stochastic dropout for extracting the most informative data points to be queried by the oracle, using uncertainty approximations. Third, we propose *ActiveHARNet* - a resource-friendly unified framework which facilitates on-device Incremental Learning (model updation) for seamless physical activity monitoring and fall detection over-time, and can further be extended to other behavior monitoring tasks in pervasive healthcare.

REFERENCES

[1] Bhattacharya, S., and Lane, N. D. From smart to deep: Robust activity recognition on smartwatches using deep learning. In *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)* (2016), IEEE, pp. 1–6.

[2] Bhattacharya, S., Nurmi, P., Hammerla, N., and Plötz, T. Using unlabeled data in a sparse-coding framework for human activity recognition. *Pervasive and Mobile Computing* (2014), 242–262.

[3] Freeman, L. C. *Elementary Applied Statistics: For Students in Behavioral Science*. John Wiley & Sons, 1965.

[4] Gal, Y., and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48* (2016), ICML'16, pp. 1050–1059.

[5] Gal, Y., Islam, R., and Ghahramani, Z. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (2017), ICML'17, pp. 1183–1192.

[6] Hossain, H. M. S., Roy, N., and Khan, M. A. A. H. Active learning enabled activity recognition. In *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)* (2016), pp. 1–9.

[7] Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745* (2011).

[8] Lane, N. D., Bhattacharya, S., Georgiev, P., Forlivesi, C., and Kawsar, F. An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices. In *Proceedings of the 2015 International Workshop on Internet of Things Towards Applications* (2015), IoT-App '15, ACM, pp. 7–12.

[9] Lasecki, W. S., Song, Y. C., Kautz, H., and Bigham, J. P. Real-time crowd labeling for deployable activity recognition. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work* (2013), CSCW '13, ACM, pp. 1203–1212.

[10] Liu, X., Liu, L., Simske, S. J., and Liu, J. Human daily activity recognition for healthcare using wearable and visual sensing data. In *IEEE International Conference on Healthcare Informatics (ICHI)* (2016), IEEE, pp. 24–31.

[11] Mauldin, T. R., Canby, M. E., Metsis, V., Ngu, A. H. H., and Rivera, C. C. Smartfall: A smartwatch-based fall detection system using deep learning. *Sensors 18* (2018).

[12] Ojetola, O., Gaura, E. I., and Brusey, J. Fall detection with wearable sensors–safe (smart fall detection). In *Seventh International Conference on Intelligent Environments* (2011), pp. 318–321.

[13] Osmani, V., Balasubramaniam, S., and Botvich, D. Human activity recognition in pervasive health-care: Supporting efficient remote collaboration. *Journal of Network and Computer Applications 31*, 4 (2008), 628–655.

[14] Settles, B. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* (2012).

[15] Shannon, C. E. A mathematical theory of communication. *Bell system technical journal 27*, 3 (1948), 379–423.

[16] Shen, Y., Yun, H., Lipton, Z., Kronrod, Y., and Anandkumar, A. Deep active learning for named entity recognition. *Proceedings of the 2nd Workshop on Representation Learning for NLP* (2017).

[17] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research 15* (2014), 1929–1958.

[18] Stisen, A., Blunck, H., Bhattacharya, S., Prentow, T. S., Kjærgaard, M. B., Dey, A., Sonne, T., and Jensen, M. M. Smart devices are different: Assessing and mitigating mobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems* (2015), SenSys '15, ACM, pp. 127–140.

[19] Sundaramoorthy, P., Gudur, G. K., Moorthy, M. R., Bhandari, R. N., and Vijayaraghavan, V. Harnet: Towards on-device incremental learning using deep ensembles on constrained devices. In *Proceedings of the 2nd International Workshop on Embedded and Mobile Deep Learning* (2018), EMDL'18, ACM, pp. 31–36.

[20] Wang, J., Chen, Y., Hao, S., Peng, X., and Hu, L. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters* (2019), 3–11.

[21] Yao, S., Hu, S., Zhao, Y., Zhang, A., and Abdelzaher, T. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In *Proceedings of the 26th International Conference on World Wide Web* (2017), WWW '17, pp. 351–360.

HARNet: Towards On-Device Incremental Learning using Deep Ensembles on Constrained Devices

Prahalathan Sundaramoorthy
Solarillion Foundation
Chennai, India
prahalath27@gmail.com

Gautham Krishna Gudur
Solarillion Foundation
Chennai, India
gauthamkrishna.gudur@gmail.com

Manav Rajiv Moorthy
SSN College of Engineering
Chennai, India
manav15057@cse.ssn.edu.in

R Nidhi Bhandari
SSN College of Engineering
Chennai, India
nidhi@cse.ssn.edu.in

Vineeth Vijayaraghavan
Solarillion Foundation
Chennai, India
vineethv@ieee.org

ABSTRACT

Recent advancements in the domain of pervasive computing have seen the incorporation of sensor-based Deep Learning algorithms in Human Activity Recognition (HAR). Contemporary Deep Learning models are engineered to alleviate the difficulties posed by conventional Machine Learning algorithms which require extensive domain knowledge to obtain heuristic hand-crafted features. Upon training and deployment of these Deep Learning models on ubiquitous mobile/embedded devices, it must be ensured that the model adheres to their computation and memory limitations, in addition to addressing the various mobile- and user-based heterogeneities prevalent in actuality. To handle this, we propose HARNet - a resource-efficient and computationally viable network to enable on-line Incremental Learning and User Adaptability as a mitigation technique for anomalous user behavior in HAR. Heterogeneity Activity Recognition Dataset was used to evaluate HARNet and other proposed variants by utilizing acceleration data acquired from diverse mobile platforms across three different modes from a practical application perspective. We perform Decimation as a Down-sampling technique for generalizing sampling frequencies across mobile devices, and Discrete Wavelet Transform for preserving information across frequency and time. Systematic evaluation of HARNet on User Adaptability yields an increase in accuracy by ~35% by leveraging the model's capability to extract discriminative features across activities in heterogeneous environments.

CCS CONCEPTS

• **Human-centered computing → Ubiquitous and mobile devices**; **Empirical studies in ubiquitous and mobile computing**; • **Computing methodologies → Neural networks**; *Ensemble methods*;

KEYWORDS

Activity Recognition; Deep Learning; Incremental Learning

EMDL'18, June 15, 2018, Munich, Germany

ACM ISBN 978-1-4503-5844-6/18/06…$15.00
https://doi.org/10.1145/3212725.3212728

ACM Reference Format:
Prahalathan Sundaramoorthy, Gautham Krishna Gudur, Manav Rajiv Moorthy, R Nidhi Bhandari, and Vineeth Vijayaraghavan. 2018. HARNet: Towards On-Device Incremental Learning using Deep Ensembles on Constrained Devices. In *EMDL'18: 2nd International Workshop on Embedded and Mobile Deep Learning, June 15, 2018, Munich, Germany.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3212725.3212728

1 INTRODUCTION

The ubiquitous proliferation of low-cost mobile devices with embedded sensors has spawned a growing research in extracting contextual information from sensor data, particularly for HAR, owing to its applications in healthcare, physical activity monitoring, fitness tracking, behavioral analysis, etc. [3][16]. The contemporary evolution of Machine Learning has provided a convenient way for exploiting these raw sensor data to abstract meaningful information. However, employing Machine Learning algorithms generally requires extensive domain knowledge for feature engineering, which is often limited by the competency of the human designing the model.

Recently, the emergence of sophisticated Deep Learning techniques has greatly alleviated the problem of crafting shallow features that have questionable generalizability. Powerful Deep Learning methods aid in automatic extraction of discriminative features by exploring hidden correlations within and between data, thereby capturing intricate details that are crucial for achieving high-level classification efficacy and robustness. However, learning complex features generally involves training models that require extensive resources, thereby making them unfriendly for real-world deployment on lightweight wearables. Furthermore, the device- and user-related diversities, such as different sensor types, device orientations, varied user-behavior, CPU loads etc., oftentimes hamper the real-world performance of the model [17]. This brings about a growing necessity for developing resource-friendly and robust HAR systems that leverage mutual interaction between the model and data, optimized to achieve state-of-the-art accuracies.

In this paper, we intend to address the following two prominent challenges in HAR.

On-device Incremental Learning

Most deep learning HAR systems are often trained on remote servers off-line or via cloud. To facilitate *User Adaptability* through *Incremental Learning* - a technique to enhance the performance of these

models by catering to each user independently, the raw inertial data needs to be transmitted to the servers from the device. However, communication between the server and device is often compromised due to latency issues (Round Trip Time taken between the server and device), and overheads in synchronization of data. One possible approach to achieve User Adaptability is to ensure training can be performed on the resource-constrained mobile/embedded systems, provided that the model is optimized.

Heterogeneity

When a HAR system is tested on multiple smartphones in real-world, the performance across various users is generally sub-optimal when compared to its simulated environment. This is due to the presence of various mobile-sensing heterogeneities prevalent during deployment. These heterogeneities predominantly include varying sampling rates, sampling rate instability due to different OS types, CPU load conditions and varied user characteristics among others [17].

In this paper, we focus on systematic minimization of resources to develop a generic HAR model in heterogeneous conditions that can be effectively trained and deployed on a Mobile/Embedded platform, whilst achieving on-par accuracies compared to state-of-the-art recognition models.

2 RELATED WORK

Modeling Deep Learning architectures for HAR has been an extensive area of research. Many researchers predominantly use Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Restricted Boltzmann Machines (RBMs) and Deep Belief Networks (DBNs) or a combination of these to build their recognition models [18].

Jiang et al. [8] effectively used Deep CNN to learn local features across dimensions from a synthesized activity image. Although this approach yields appreciable results, creating and analyzing an activity image is laborious and memory consuming, which might not be well-suited for deployment on mobile and embedded platforms. Ronao et al. [15] successfully demonstrated the usage of two-dimensional CNN for efficient classification of activities. However, the model was tested on a dataset that contains data from a single smartphone, thereby failing to showcase its generalizing capabilities across devices. Ravi et al. [14] performed temporal convolutions on Short-Time Fourier Transform (STFT) spectrogram of the input signals. Though the system was deployed on low-cost wearable devices, its capability to perform well for unseen users is not pronounced.

To learn hierarchial features, Guan et al. [4] proposed an ensemble of LSTM learners for building a robust recognition system. RBMs and multi-layer RBMs [13] have also been used to capture local and multi-modal interactions in HAR. Ordóñez et al. [12] and Hammerla et al. [5] exploit their own convolutional and recurrent network architectures, but fail to illustrate the performance of the same under real-world heterogeneous environments. Implementing hybrid models using a combination of CNNs and RNNs has been proposed by Yao et al. [19] in DeepSense, which fuses data from multiple sensor modalities while also incorporating temporal relations. Although these works achieve impressive results in terms of accuracy and classification time, the feasibility of incremental learning seems to be debatable.

The rest of the paper is organized as follows: Section 3 and Section 4 discuss the dataset and various preprocessing steps to achieve dataset reduction for reducing memory overhead in devices. We elucidate our proposed model in Section 5, followed by systematic evaluation and demonstration under heterogeneous and resource-constrained scenarios in Sections 6 and 7.

3 DATASET

We utilize the Heterogeneity dataset (D_H) proposed by Allan et al. [17] to design our model. D_H consists of inertial values recorded from accelerometer and gyroscope present in eight smartphones across nine users performing six daily activities (Table 1). To ensure uniformity, each activity was performed for five minutes by all users across all phones. The real-world sensing diversities are reflected by data from different phones operating with varying Sampling Frequencies (F_S, ranging from 50-200 Hz) in D_H.

Table 1: Heterogeneity Dataset (D_H) characterized by their respective attributes

Activities	Devices	F_S	Users
['Biking', 'Sitting', 'Standing', 'Walking', 'StairsUp', 'StairsDown']	Nexus 4 Samsung S3 Samsung S3 Mini Samsung S+	200 150 100 50	[a, b, c, d, e, f, g, h, i]

4 DATASET PREPROCESSING

In order to handle the varying sampling frequencies of different devices and to obtain a rich yet sparse representation of the signal components, we perform the following preprocessing steps.

4.1 Windowing and Decimation

We initially segment raw inertial data into non-overlapping two-second activity windows (w_a). These segmented chunks of data have non-uniform lengths due to varying sampling frequencies of the devices (F_s). This disparity may impede the performance of the model, particularly when F_s of smartphones are not identical during training and testing. To handle this issue, *Up-sampling* or *Down-sampling* of data can be performed to ensure that each window w_a has a fixed size. Up-sampling is likely to induce noise in the data and increase memory requirements [17]. Hence, a better approach would be to down-sample the signals to a common sampling frequency, as the characteristics of the input signals are likely to be retained, while resulting in data size reduction.

The authors perform *Decimation* - a technique that down-samples a signal, by applying an 8^{th} order Chebyshev type-I filter without any phase shift. In D_H, we choose the lowest F_s - 50 Hz as the common sampling frequency for down-sampling. Decimation is performed on all signals from mobile devices whose F_s is greater than 50 Hz, thereby ensuring consistency in size of each window w_a. Decimation results in data reduction upto 75% for phones with highest F_s - 200Hz.

4.2 Discrete Wavelet Transform

A better representation of the raw inertial signals can be obtained by capturing both temporal and frequency information, which retain locally well-defined temporal characteristics in the frequency domain [11].

DWT convolves the incoming signal *x(n)* with a wavelet ψ by using multiple filter banks to achieve decomposition into high- and low-frequency components. These components are represented by *Detail* (C_D) and *Approximation* (C_A) coefficients. By discarding C_D and utilizing C_A, we get a smoothened version of *x(n)* [2]. This process yields a sparse representation of the signal, thereby compressing the size of the data (~50%).

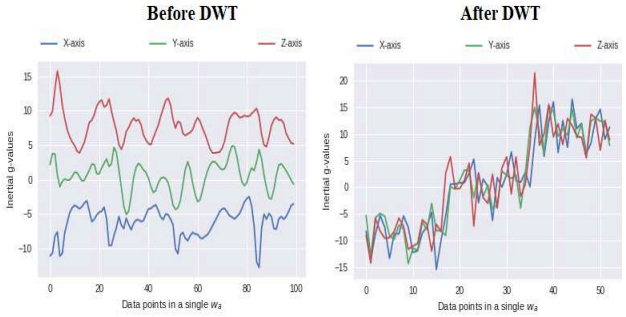


Figure 1: Inertial accelerometer signals of the three axes before and after DWT

From Figure 1, we can visually interpret the enhanced correlation between inertial *g*-values of the three axes after performing DWT by utilizing the temporal and frequency information captured.

5 MODEL

HAR in real-time requires identification of discriminative sets of features for effective classification. Deep Learning facilitates automatic extraction of such distinctive features, that otherwise do not generalize across datasets. Comprehensive analysis of correlations across various axes is essential to learn such features efficiently. Hence, in this paper, we study various architectures to extract the intra-axial and inter-axial feature dependencies.

INTRA-AXIAL DEPENDENCIES

Each activity window w_a is constituted of $\{w_a^X, w_a^Y, w_a^Z\}$ denoting the input vectors across axes X, Y and Z respectively. These vectors are represented by the frequency sub-bands of C_A. To extract the local information within each vector in $\{w_a^X, w_a^Y, w_a^Z\}$, we systematically evaluate the following intra-axial variants as shown in Figure 2.

Conv-1D. Convolutional Neural Networks (CNNs) are used as powerful feature learning tools in many Deep Learning classification tasks. Each convolutional kernel analyzes and extracts local characteristics within each input axis. We combine the learned features of all axes for achieving distinctive representation of the incoming sensor signal for classification.

The intra-axial CNN takes an input vector at the first layer L_1^C from $\{w_a^X, w_a^Y, w_a^Z\}$. Each layer L_n^C provides a feature-map $f_{L_n^C}$ as

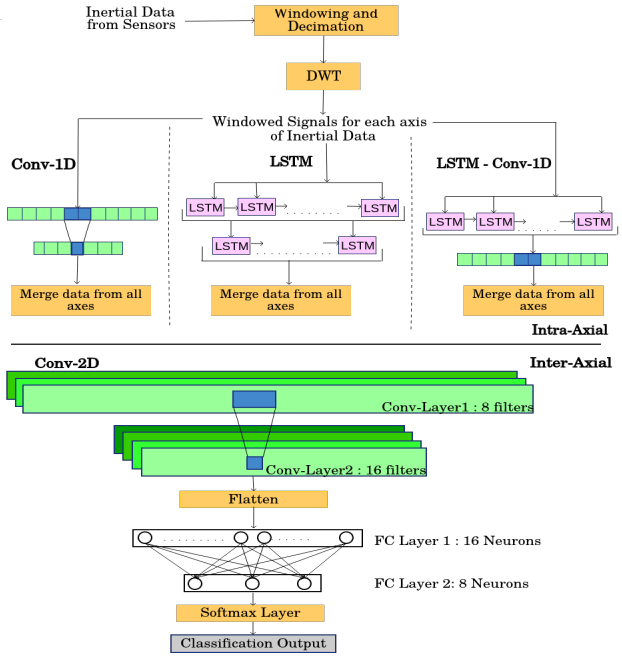


Figure 2: Model Architecture Variants

the input for every subsequent layer L_{n+1}^C in the network. Each feature map is obtained using convolutional filters applied throughout the feature map $f_{L_n^C}$ and is provided as input to the hidden layer L_{n+1}^C.

In this variant, we utilize a two-layer stacked convolutional network with a receptive field size (kernel size) of 2. To regularize each mini-batch and reduce the internal covariate shift, a Batch Normalization layer is also used in each stack [7]. Each batch-norm layer is followed by a Max-pooling layer of pool size 2x2.

LSTM. The Long Short-Term Memory (LSTM) units are one of the extended variants for vanilla recurrent networks. LSTMs have proven to be successful in capturing pattern information in time-series data, as they have the potential to model dynamic temporal behavior [6].

Recurrent units of the first layer utilize the input vectors $\{w_a^X, w_a^Y, w_a^Z\}$ to learn the local temporal characteristics. The input of the each following hidden recurrent layer L_n^R is of the form $d = \bigcup_{i=0}^{N} d_i$, where N is the number of units of the previous recurrent layer L_{n-1}^R. These sequences will be modeled for each timestep d_t by remembering the states for the previous d_{t-1} timesteps, $\forall t \in [0, N]$.

We use a two-layer stacked LSTM network in this variant, comprising of 32 and 20 output cells each. A Hyperbolic Tangent (*tanh*) activation function is used for the same.

LSTM → Conv-1D. By using LSTMs for modeling complex temporal relations and 1-D convolutions for extracting the most salient features from these functions pertaining to each axis, we could obtain a well-learned representation of the input signal.

Table 2: Total parameters, Accuracies, F1-Scores and Time taken for Classification of a single window for D_H across all models in mode M_U

Model	Params	Accuracy	F1-Score	Time (in ms)
HAR-CNet	31,806	95.68	0.9619	**10.9**
HAR-LNet	29,910	95.42	0.9573	850.2
HAR-LCNet	40,094	**96.79**	**0.9651**	68.9

This proposed framework utilizes a combination of layers from both LSTM and Conv-1D. Inputs $\{w_a^X, w_a^Y, w_a^Z\}$ are initially fed into a recurrent network from which a modeled sequence $f_{L_n^R}$ is obtained, which is further used to generate feature maps using 1-D convolutions. The Convolutional layer L_n^C is stacked over the final recurrent layer L_n^R, and takes the input $f_{L_n^R}$ to provide a feature-extracted vector $f_{L_n^C}$ per axis.

In this variant, we propose a one-dimensional convolutional layer comprising of 8 filters and a kernel size of 2 over an LSTM layer similar to the aforementioned variant, with a Batch Normalization regularizer and a 2x2 pooling layer.

INTER-AXIAL DEPENDENCIES

A two-dimensional CNN can effectively learn distinctive characteristics across spatial dimensions [10]. We aim to capture the interactions between data from the three axes, using convolutional layers.

The outputs of the intra-axial models for all three input vectors $\{w_a^X, w_a^Y, w_a^Z\}$ are concatenated to form a feature matrix F, which provides a sophisticated representation from which inter-axial dependencies can be easily correlated.

In this paper, we propose an inter-axial model - a two-layer stacked 2-D CNN with convolutional layers comprising of 8 and 16 filters each and a receptive field of size 3x3. Each convolutional layer is followed by a Batch Normalization and a Pooling layer of size (3x2). This stacked network is followed by two Fully-Connected (FC) layers constituting 16 and 8 neurons each with Rectified Linear Unit (ReLU) activations. The Dropout regularization technique is applied after each Fully-Connected layer with a probability of 0.25. Negative log-likelihood (Softmax) probability estimations are used for classification of activities.

The intra-axial patterns and inter-axial interactions together will enable extensive analysis and modeling of activities. Using deep ensembles of the intra-axial variants with the inter-axial model provide an all-encompassing and rich representation of the input signals.

In this work, we thus propose the following *HARNet* variants:

•***HAR-CNet*** : $\{Conv\text{-}1D \rightarrow Conv\text{-}2D\}$

•***HAR-LNet*** : $\{LSTM \rightarrow Conv\text{-}2D\}$

•***HAR-LCNet*** : $\{LSTM \rightarrow Conv\text{-}1D \rightarrow Conv\text{-}2D\}$

Parametric optimization of convolutional filters and kernel size, recurrent cells and FC neurons drastically reduces the memory and time complexities. Significant reduction of such parameters in each layer enables efficient memory management on a resource-constrained platform. We recursively prune the model parameters to systematically arrive at the optimal proposed variants, while not compromising on recognition accuracy. Introducing Dropout between FC layers further reduces the number of parameters, thereby enabling successful on-device training and deployment on constrained devices.

We formalize our ensembled deep model using the TensorFlow module [1]. The network is trained with a learning rate of $2e^{-4}$ to minimize the *categorical cross-entropy loss* (ρ) as shown below.

$$\rho = -\sum_{k=1}^{K} y_{i,k} \log(x_{i,k}) \tag{1}$$

where x are the predictions, y are the target values, i denotes the data points from w_a across K classes. This loss ρ is optimized during back-propagation for each mini-batch using the Adam optimizer [9].

6 EXPERIMENTS AND RESULTS

The evaluation modes of HAR algorithms are crucial in quantifying its extensibility and generalizability during real-world deployment. We evaluate the performance of our proposed models across the three modes stated below.

•*Mixed User Mode* (M_U)

This is one of the most commonly used evaluation modes in HAR. In this mode, the whole dataset is split into stratified samples of 80% train and 20% test data.

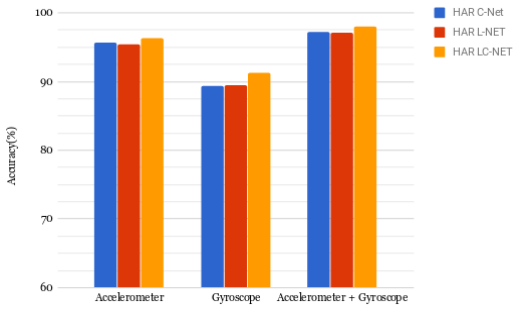


Figure 3: Sensor Minimization: A Comparative Analysis of Accuracies for Accelerometer and Gyroscope

Sensor Minimization: We analyze the classification accuracies of our model variants using data from different combinations of both accelerometer and gyroscope. From Fig. 3, we observe that the accuracies obtained when using data from both accelerometer and gyroscope do not significantly exceed those obtained by using just the accelerometer data (~ 1.5%). We thus perform sensor minimization to address the challenge of On-Device Incremental Learning by forgoing data from gyroscope, which substantially reduces memory requirements by 50% and computational cost for each input vector. We hence use the data from accelerometer alone for further analysis of our models.

The results for mode M_U on dataset D_H are showcased in Table 2. We observe that HAR-LCNet outperforms the other two variants in terms of accuracy and F1-score, as it exploits a combination of recurrent and convolutional networks. Each recurrent unit preserves the observed patterns in accelerometer data over time across each axis by utilizing a common weight matrix W, which encapsulates the diversity in instances of the same activity. Using convolutional filters over these modeled sequences then provides a rich feature-set from which the model can learn effectively.

	'Stand'	'Sit'	'Walk'	'Stairsup'	'Stairsdown'	'Bike'
'Stand'	99.28	0.72	0	0	0	0
'Sit'	0.12	99.88	0	0	0	0
'Walk'	0	0	90.19	3.76	6.05	0
'Stairsup'	0	0	4.48	87.75	6.92	0.85
'Stairsdown'	0	0	4.16	5.27	90.57	0
'Bike'	0.73	0	0	0.73	0.48	98.06

Figure 4: Confusion Matrix for *HARNet* in Mode M_U

Upon comparing HAR-LCNet with the next-best performing model HAR-CNet, we observe that HAR-CNet is ~7x faster than HAR-LCNet in terms of inference time per sample, with a ~1% difference in accuracy and F1-score. Considering the resource-constrained nature of mobile/embedded platforms, we narrow down to HAR-CNet as our final ensembled deep framework - ***HARNet***, which gives high accuracy with least classification time. The confusion matrix of *HARNet* is shown in Figure 4. Majority of the misclassification occurs between the classes : 'StairsUp', 'StairsDown' and 'Walking', which can be attributed due to the lack of orientation details of the smartphone, that is traditionally captured by the gyroscope.

•*Device Independent Mode* (D_I)

This mode aims at evaluating the model's performance across various devices, thereby reflecting its capability to deal with various mobile-sensing heterogeneities when deployed in a real-world scenario. A stratified k-fold cross validation technique is employed for a *Leave-One-Device-Out* approach. The average accuracy and F1-score obtained are 89.5% and 0.887 respectively. Figure 5 illustrates the accuracy of our model across devices, thereby showcasing its generalizing capabilities.

•*User Independent Mode* (U_I)

In this mode, we attempt to classify activities performed by a previously unseen user through the *Leave-One-User-Out* approach. This mode hence provides a strong measurement of generalizability of the model across diverse users. A similar cross validation technique as mentioned above is used. Testing in this mode yields an F1-score of 0.80 using the accelerometer data alone which is higher than the F1-scores presented in [20] and [17], which use data from both accelerometer and gyroscope.

We analyze the relation between the number of epochs and classification accuracies for two specific users: 'b' and 'c', for whom the best and least accuracy are observed. We can infer that user 'b'

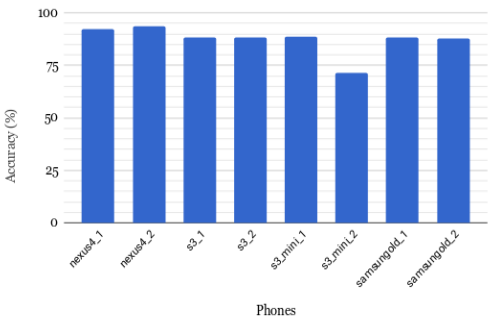


Figure 5: Mode D_I: Comparative Analysis of Accuracies across various devices

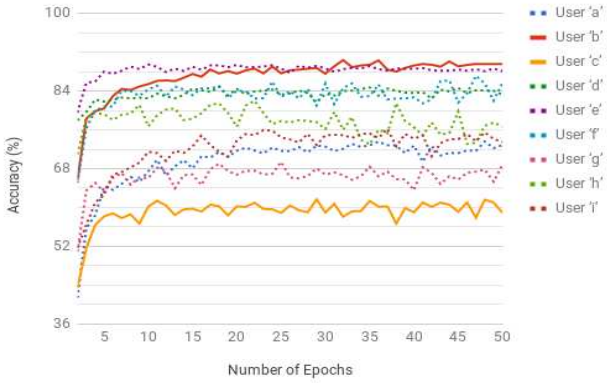


Figure 6: Mode U_I: Comparative Analysis of Accuracies vs number of Epochs for Users 'a' through 'i'

performs activities similar to the general trend as high accuracy is observed for the same. However, user 'c' achieves least accuracy which can be attributed to the user's unique physical build, posture and execution of activities. It is evident from Figure 6 that even though the number of epochs is increased during training phase, the model does not yield better accuracies for user 'c'. To enhance the efficiencies of such users, the model should adapt to the user's unique behavioral pattern. We hence perform *Incremental Learning* by using a portion of the data from the unseen test user to update the weights of the previously trained model, thereby adapting to even the least-performing users.

7 ON-LINE INCREMENTAL LEARNING

We experiment user-based Incremental Learning using *HARNet* for the users 'b' and 'c' by deploying the system on a Raspberry Pi 3 Model B. The model is initially trained in mode U_I, and the trained weights and parameters are stocked on Raspberry Pi. The portion of the unseen user data included for Incremental Learning is governed by the adaption factor λ. We first assign λ=0.25 for both users and observe the accuracy change. As shown in Figure 7, the accuracies

of the users increased after performing Incremental Learning, particularly for user 'c', where there is a substantial increase in accuracy of ~35%.

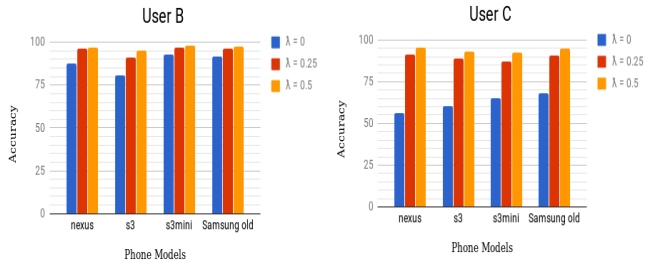


Figure 7: Incremental Learning for User with Best and Least Accuracy

When there is an influx of a stream of data (λ increases) for a particular user, the model adapts itself well to the user's behavioral pattern, thus leading to higher accuracies. Table 3 illustrates the time taken for preprocessing and testing phases per activity window on the Raspberry Pi. The user-based incremental learning on Raspberry Pi takes 3 seconds per epoch. It is evident that inference time per activity window w_a is attributed to the size of the model (~0.5 MB). Furthermore, the time taken for preprocessing and testing together ensures the computational viability of the proposed methodology on embedded and mobile platforms.

Table 3: Time taken for Execution per activity window (w_a)

Process	Computational Time
Inference time	17 ms
Discrete Wavelet Transform	0.5 ms
Decimation	4.8 ms

8 CONCLUSION

In this paper, we proposed *HARNet* - a Deep Learning framework with capabilities to handle various mobile-sensing and user-based heterogeneities while being resource-friendly on low-cost embedded and mobile platforms. By systematically optimizing the data preprocessing and model design phases, we were able to achieve remarkable accuracies using *HARNet*, which has a size of ~0.5 MB. Thus, the authors were able to perform Incremental Learning on Raspberry Pi 3 to facilitate User Adaptability, which proves beneficial for anomalous users. Notably, an increase in accuracy of ~35% was achieved signifying the feasibility of *HARNet* on embedded and mobile devices.

9 ACKNOWLEDGEMENT

The authors would like to thank Solarillion Foundation for its support and funding of the research work carried out.

REFERENCES

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

[2] Borovykh, A., Bohte, S., and Oosterlee, C. W. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691* (2017).

[3] Buttussi, F., and Chittaro, L. Mopet: A context-aware and user-adaptive wearable system for fitness training. *Artif. Intell. Med. 42*, 2 (Feb. 2008), 153–163.

[4] Guan, Y., and Plötz, T. Ensembles of deep lstm learners for activity recognition using wearables. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 1*, 2 (June 2017), 11:1–11:28.

[5] Hammerla, N. Y., Halloran, S., and Plötz, T. Deep, convolutional, and recurrent models for human activity recognition using wearables. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (2016), IJCAI'16, AAAI Press, pp. 1533–1540.

[6] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural Comput. 9*, 8 (Nov. 1997), 1735–1780.

[7] Ioffe, S., and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37* (2015), ICML'15, JMLR.org, pp. 448–456.

[8] Jiang, W., and Yin, Z. Human activity recognition using wearable sensors by deep convolutional neural networks. In *Proceedings of the 23rd ACM International Conference on Multimedia* (New York, NY, USA, 2015), MM '15, ACM, pp. 1307–1310.

[9] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[10] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1* (USA, 2012), NIPS'12, Curran Associates Inc., pp. 1097–1105.

[11] Najafi, B., Aminian, K., Paraschiv-Ionescu, A., Loew, F., Bula, C. J., and Robert, P. Ambulatory system for human motion analysis using a kinematic sensor: monitoring of daily physical activity in the elderly. *IEEE Transactions on Biomedical Engineering 50*, 6 (June 2003), 711–723.

[12] Ordóñez, F. J., and Roggen, D. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors 16*, 1 (2016).

[13] Radu, V., Lane, N. D., Bhattacharya, S., Mascolo, C., Marina, M. K., and Kawsar, F. 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* (New York, NY, USA, 2016), UbiComp '16, ACM, pp. 185–188.

[14] Ravi, D., Wong, C., Lo, B., and Yang, G. Z. Deep learning for human activity recognition: A resource efficient implementation on low-power devices. In *2016 IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks (BSN)* (June 2016), pp. 71–76.

[15] Ronao, C. A., and Cho, S.-B. Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Syst. Appl. 59*, C (Oct. 2016), 235–244.

[16] Shoaib, M., Bosch, S., Incel, O. D., Scholten, H., and Havinga, P. J. M. Fusion of smartphone motion sensors for physical activity recognition. *Sensors 14*, 6 (2014), 10146–10176.

[17] Stisen, A., Blunck, H., Bhattacharya, S., Prentow, T. S., Kjærgaard, M. B., Dey, A., Sonne, T., and Jensen, M. M. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems* (New York, NY, USA, 2015), SenSys '15, ACM, pp. 127–140.

[18] Wang, J., Chen, Y., Hao, S., Peng, X., and Hu, L. Deep learning for sensor-based activity recognition: A survey. *arXiv preprint arXiv:1707.03502* (2017).

[19] Yao, S., Hu, S., Zhao, Y., Zhang, A., and Abdelzaher, T. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In *Proceedings of the 26th International Conference on World Wide Web* (Republic and Canton of Geneva, Switzerland, 2017), WWW '17, International World Wide Web Conferences Steering Committee, pp. 351–360.

[20] Yao, S., Zhao, Y., Shao, H., Zhang, A., Zhang, C., Li, S., and Abdelzaher, T. Rdeepsense: Reliable deep mobile computing models with uncertainty estimations. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 1*, 4 (Jan. 2018), 173:1–173:26.

A Vision-based Deep On-Device Intelligent Bus Stop Recognition System

Gautham Krishna Gudur[*]
Global AI Accelerator
Ericsson
gautham.krishna.gudur@ericsson.com

Ateendra Ramesh[*]
Computer Science and Engineering
SUNY at Buffalo
ateendra@buffalo.edu

Srinivasan R
Dept. of Information Technology
SSN College of Engineering
srinivasanr@ssn.edu.in

ABSTRACT

Intelligent public transportation systems are the cornerstone to any smart city, given the advancements made in the field of self-driving autonomous vehicles – particularly for autonomous buses, where it becomes really difficult to systematize a way to identify the arrival of a bus stop on-the-fly for the bus to appropriately halt and notify its passengers. This paper proposes an automatic and intelligent bus stop recognition system built on computer vision techniques, deployed on a low-cost single-board computing platform with minimal human supervision. The on-device recognition engine aims to extract the features of a bus stop and its surrounding environment, which eliminates the need for a conventional Global Positioning System (GPS) look-up, thereby alleviating network latency and accuracy issues. The dataset proposed in this paper consists of images of 11 different bus stops taken at different locations in Chennai, India during day and night. The core engine consists of a convolutional neural network (CNN) of size ~260 kB that is computationally lightweight for training and inference. In order to automatically scale and adapt to the dynamic landscape of bus stops over time, incremental learning (model updation) techniques were explored on-device from real-time incoming data points. Real-time incoming streams of images are unlabeled, hence suitable ground truthing strategies (like Active Learning), should help establish labels on-the-fly. Light-weight Bayesian Active Learning strategies using Bayesian Neural Networks using dropout (capable of representing model uncertainties) enable selection of the most informative images to query from an oracle. Intelligent rendering of the inference module by iteratively looking for better images on either sides of the bus stop environment propels the system towards human-like behavior. The proposed work can be integrated seamlessly into the widespread existing vision-based self-driving autonomous vehicles.

CCS CONCEPTS

• **Human-centered computing** → **Ubiquitous computing**; • **Applied computing** → **Transportation**; • **Computing methodologies** → *Neural networks*; *Online learning settings*; *Active learning settings.*

KEYWORDS

Bus Stop Recognition, On-Device Deep Learning, Computer Vision, Bayesian Active Learning, Incremental Learning

ACM Reference Format:
Gautham Krishna Gudur, Ateendra Ramesh, and Srinivasan R. 2019. A Vision-based Deep On-Device Intelligent Bus Stop Recognition System. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and the 2019 International Symposium on Wearable Computers (UbiComp/ISWC '19 Adjunct), September 9–13, 2019, London, United Kingdom.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3341162.3349323

1 INTRODUCTION

Buses are a widespread mode of transportation that are used by the majority of the public. Recently, according to APTA 2019 [1], over 47% of people use buses as their preferred public transport mode in the United States, while 76% of buses have automatic bus stop announcements. In a country like India, conventionally the conductor of the bus intimates (whistles) when a bus stop arrives and announces its location aloud, while the driver halts the bus. This scenario still exists in most buses in India, however, with increase in automatic announcing mechanisms like pre-defined queues which have been widely followed – wherein the sequence of bus stops are stocked initially, this seems to be an easier alternative to the conductor's manual announcement of bus stops.

With the recent advancements and innovations in self-driving autonomous vehicles (buses) which are mostly based on state-of-the-art computer vision techniques, it becomes

[*]A part of this work was also done when affiliated with SSN College of Engineering.

UbiComp/ISWC '19 Adjunct, September 9–13, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6869-8/19/09…$15.00
https://doi.org/10.1145/3341162.3349323

an increasing necessity to not only know the sequence of bus stops, but intelligently perceive where the bus stop exactly is, in order for the bus to halt at the right place. Many current bus stops in India, particularly rural and suburban bus stops predominantly do not have bounded or localized spaces/lanes. The surroundings of the bus stops also dynamically change and evolve over time. Moreover, bus routes are periodically revised and bus stops also change based on demand and traffic patterns. This calls for an intelligent transit system to not only identify bus stops automatically, but to incrementally adapt on-device to the new dynamic surroundings of the bus stops on-the-fly with minimal human intervention, and intimate passengers about the same. To reduce the overhead of capturing images during the whole route during inference (classification of bus stop), we propose that the images are captured only when speed of the bus is below a certain ideal threshold, only after which the inference engine would capture and classify an image. The real-time speed can be acquired from the speedometer of the bus and the ideal minimum threshold (10 km/hr for instance) is subject to locality and traffic conditions.

Global Positioning System (GPS) look-up can be used bus stops identification, however latency issues in the network, along with accuracy and privacy concerns make GPS look-up disadvantageous. Also, it might be hard to localize, identify and halt the vehicle right in front of the bus stop using GPS. Thus, in order to address and alleviate the aforementioned concerns, we propose a novel on-device vision-based solution. Given the significant developments in the field of vision-based deep learning in self-driving modes of transport [2], our system could seamlessly integrate with such systems with minimal processing power. This enables on-device feasibility and remote recognition in real-time.

Furthermore, given that the working of this paper focuses on real-time recognition of bus stops, real world unlabeled data necessitates the requirement of ground truth for various bus stop images. Hence, the authors utilize Bayesian Active Learning strategies by leveraging recent advancements of *Bayesian Neural Networks* to conveniently model uncertainties, making it easier to combine various acquisition functions to learn unlabeled data efficiently, thereby reducing the load of querying the oracle. Incremental Active Learning mechanisms employed on-device help overcome the dynamic nature of real-time bus stop data, and also enable scalability of bus stops. The proposed mechanism would in turn learn the most informative acquired images from uncertainty estimations, with minimal human intervention.

The motivation for building the inference system is to simulate the thinking of the human brain, i.e., a way that imitates or mimics a person's reaction when inquired about a particular bus stop. Typically, the individual would look to their left and right and then make a decision, and when the person is unsure of the bus stop, he/she would look for further frames in order to become more confident. This idea has been encompassed here, wherein the system would not classify a bus stop if not fully confident with only two images, which further augments its performance and intelligent decision making capabilities, in turn making it robust and efficient. The key contributions of this paper include:

- Proposing a dataset of bus stop images acquired from cameras placed atop a bus in few select locations in the city of Chennai, India, and a light-weight model for the same to perform on-device inference.
- Incorporating Incremental Learning capabilities to enable scalability, and dynamically adapt to evolving surroundings across various bus stops on-device. Data augmentation techniques to handle class imbalance for new classes are also studied.
- A study of *Bayesian Active Learning* using Bayesian Neural Networks to model uncertainties, by examining several acquisition functions to acquire labels on-the-fly and substantially reducing labeling load on oracle.
- An intelligent inference engine which mimics human-like thinking.

2 RELATED WORK

Rapid developments have been made in field of image recognition in tandem with intelligent and autonomous vehicles. However, in the scope of bus stop recognition, very few systems employ such learning models that are extremely capable of learning the dynamically evolving surroundings of bus stop over time.

The system proposed by Pan et al. [13] focuses on an image based (HOG algorithm with SVM), however this work is used for identification of a bus and its number, rather than the bus stop. The authors from [16] and [9] proposed intelligent bus stop identification systems using trajectories from GPS traces of bus routes in smartphones with impressive efficiencies. However, the use of GPS in systems increase the network latency and communication overheads.

In [5], the authors propose a system that recognizes painted patterns/messages on the road using a Kalman filter and pattern matching, however this system is computationally expensive. Moreover, road signs might not be a dependable means of pinpointing a bus stop in most places. Most previous works assume that the incoming bus stop data are labeled previously, which puts forth the requirement to label real-time data on-the-fly. Active Learning (AL) techniques effectively identify the most informative data points and query the oracle (user) for ground truth.

Traditional algorithms in AL [14], [4] are statistically proven for low-dimensional data, but do not generalize

(a) 7H - Mogappair (b) Thirumangalam (c) Gurunath (d) Collector Nagar (e) Pari Salai (Night)

Figure 1: Examples of Bus Stops

across deep neural networks, which are inherently high-dimensional. Approximations using uncertainty sampling has been an active area of research for many years now, with efficient Bayesian AL strategies [8] and [12] recently proposed. These proposed works primarily deal with image data, and help in identifying the most uncertain images to be queried by the oracle. Moreover, the authors from [10] propose Incremental Active Learning for wearable on-device scenarios, and our proposed work also takes motivation from the same.

The rest of the paper is organized as follows. We propose our dataset in Section 3 and our Bayesian model architecture in Section 4. Section 5 discusses about the various acquisition functions used for querying the oracle during Bayesian Active Learning. The baseline efficiencies for the model with existing and new classes with data augmentation are elucidated in Section 6. This is followed by systematic evaluation in resource-constrained Incremental Active Learning scenarios in Section 7. A novel intelligent inference mechanism is presented in Section 8, and Section 9 concludes the paper.

3 DATASET

The dataset primarily consists of images of 8 bus stops from Chennai, India, of which five are public bus stops and three are taken inside SSN College of Engineering (SSNCE) over different days. These images were acquired using two 5 MP cameras placed in opposite directions on buses during the day. In addition to these eight bus stops, three bus stops were captured during night. We propose using two different classifiers for day and night separately, for which a light sensor can be used to switch between daylight and night-time (including dark and overcast times).

For each bus stop during the day, 60 images were acquired for each side of the bus (left & right), thus, resulting in a total of 960 images. The classification model was trained with 720 images stratified across each bus stop, and the rest 240 were used for testing the same. The same amount of images per bus stop on either sides were acquired for the night. Figure 1 illustrates examples of different bus stops in the dataset. We focus on the results acquired during the day, and the same methodology and model can be scaled during night.

Furthermore, the images were resized to $32 \times 32 \times 3$ and normalized (divided RGB pixel values by 255 for easier model convergence), in order to facilitate hardware-friendly computation with lesser memory overhead.

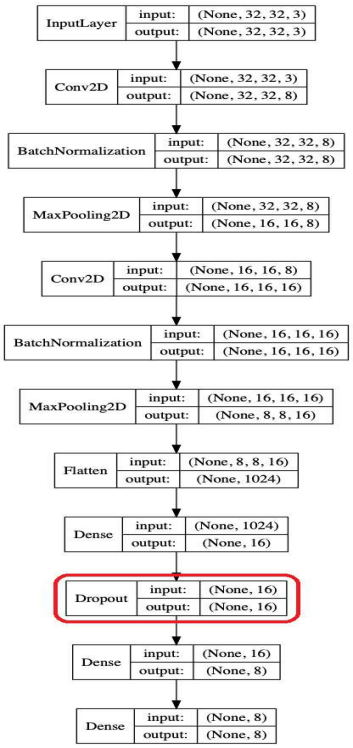


Figure 2: *Bayesian Convolutional Neural Network (CNN)* Architecture which can model uncertainties

4 MODEL ARCHITECTURE

Bus-Stop Recognition in real-time requires identification of discriminative features of the acquired bus stop scenes for efficient classification. In this paper, we utilize Convolutional Neural Networks (CNNs) which are powerful mechanisms for distinctive spatial representations and offer automatic, effective feature learning capabilities using a series of Convolutional and Pooling layers. The convolutional layers take in the images in three-dimensions and perform convolutional operations to the input sequence with various kernels (receptive fields) and desired amount of filters (feature maps).

The model consists of two stacked two-layered convolutional network comprising of 8 and 16 filters each, and receptive field size of 2x2. Each convolutional layer is followed by a Batch Normalization and a Max Pooling layer of size 2x2 each. This is followed by two Fully-Connected (FC) layers with 16 and 8 neurons with weight regularization (L2-regularizer with a weight decay constant), and ReLU activation functions. The *Dropout* regularization technique [17] is used between the FC layers with a probability of 0.3, and finally a Softmax layer is used for calculating the negative log-likelihood probability estimates. The categorical-cross entropy loss of the model is minimized using Adam optimizer, with a learning rate of 10^{-3}, and implemented using the TensorFlow framework.

Dropout is used at both train and test times between FC layers for multiple stochastic forward passes (*T=10* in this paper), to sample the approximate posterior as stated in [7]. Since the weights in *Bayesian (Convolutional) Neural Networks* are probability distributions (Gaussian priors) instead of point estimates – equivalent to performing dropout for T iterations, they can efficiently model uncertainty estimates, which can be used with existing deep acquisition functions for Active Learning. The model architecture can be observed from Figure 2.

5 ACQUISITION FUNCTIONS FOR ACTIVE LEARNING

Given a model M, real-time pool data D_{pool}, and inputs $x \in D_{pool}$, [8] states that an acquisition function $a(x, M)$ is a function of x that the Active Learning system uses for inference of the next query point:

$$x^* = argmax_{x \in D_{pool}} a(x, M).$$

Acquisition functions are used in AL to quantify uncertainties and arrive at the most efficient set of data points to query from D_{pool}. We examine the following acquisition functions, whose detailed results are observed in Section 7:

Bayesian Active Learning by Disagreement (BALD). Information about model parameters are maximized under the posterior that disagree the most about the outcome [11].

$$\mathbb{I}[y, \omega | x, D_{train}] = \mathbb{H}[y|x, D_{train}] - E_{p(\omega|D_{train})}\big[\mathbb{H}[y|x, \omega]\big]$$

where $\mathbb{H}[y|x, \omega]$ is the entropy of y, given model weights ω.

Variation Ratios. Utilizes the Least Confident method for uncertainty based pool sampling [6].

$$variation - ratio[x] := 1 - \max_y p(y|x, D_{train})$$

Max Entropy. Predictive entropy is maximized by appropriately chosen pool points. [15].

$$\mathbb{H}[y|x, D_{train}] := -\sum_c p(y = c|x, D_{train}) \log p(y = c|x, D_{train})$$

Random Sampling. Equivalent to choosing a random point from a uniform pool distribution.

6 BASELINE EXPERIMENTS AND RESULTS

Initially, only 7 bus stops are taken into consideration and called existing classes, while the eighth class (SOMCA bus stop) is treated as the new unseen bus stop for illustrating scalability.

Existing Classes

The experiments are performed with 630 train images and 210 test images stratified across all seven classes. This is generally a one-time training from scratch on server for establishing the baseline accuracies, and it can be observed from Figure 3 that the training and testing modules give high accuracies of ~97% and ~96% respectively.

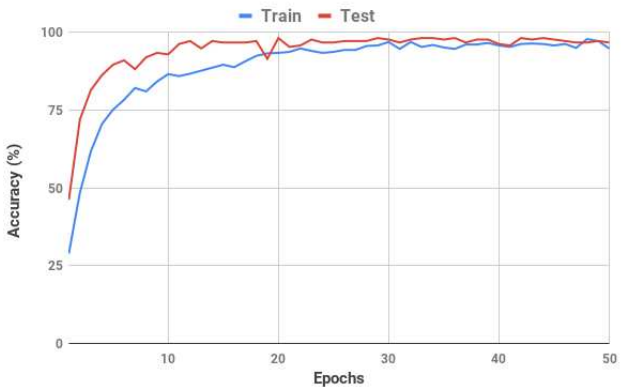


Figure 3: Baseline Train, Test Accuracies vs Epochs

Data Augmentation for New Classes

When a new bus stop is being added to the route, it becomes a necessity for the model to scale and handle the incoming real-time data of the new class. The acquired number of images from the new bus stop is predominantly lesser than that of existing classes which might result in a class imbalance. Hence, image augmentation techniques are applied in order to ensure stratified training across all classes. Techniques like zoom, shear and rotation by small fractions are utilized for generating new images [3], which almost resemble the images acquired from a real-time camera.

Initially, we assume only 4 data points were collected from either side of the new bus stop. From Figure 4, it can be seen that the accuracy of the model with just the 8 data points, retrained from scratch gives an accuracy of 86.25%. However, the model with new bus stop data augmented to sufficiently higher images such that no class imbalance exists, achieves an accuracy of 96.7% which is a ~10.5% increase in accuracy.

This shows the effectiveness of data augmentation strategies for new classes, thereby ensuring scalability of bus stops.

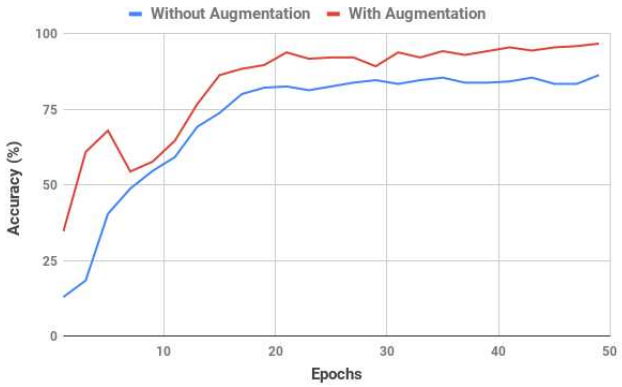


Figure 4: With and Without Data Augmentation for New Class; Test Accuracies vs Epochs

7 INCREMENTAL ACTIVE LEARNING

In order to reduce oracle's load in labeling the incoming real-time bus stop images, we perform Bayesian AL using various acquisition functions as mentioned in Section 5. The system is deployed on a Raspberry Pi 2 in real-time, and the stocked model weights are updated on-device in an incremental manner. In the event of there being a change in scene in an existing bus stop environment, incremental training will emphasize on learning the most recent and salient features of that bus stop. Moreover, when a new bus stop is introduced, the existing model with the newly learned weights are updated.

Incremental Active Learning on Existing Classes

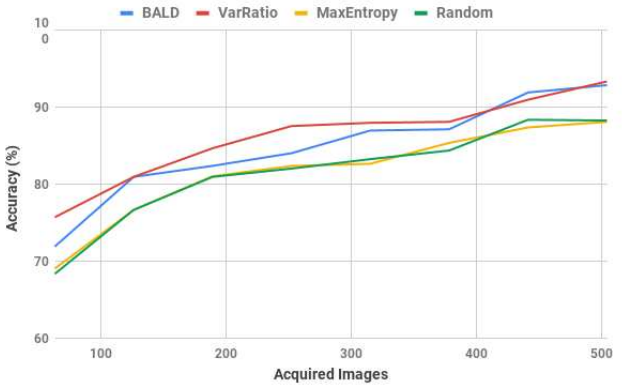


Figure 5: Acquired Images vs Accuracy on existing classes

The training data with existing 7 classes are split into pool (D_{pool}) and train (D_{train}) (80-20 ratio) for simulating the Bayesian Incremental Active Learning framework, as an approximation for real-world data. D_{test} is used for evaluation purposes only. The initial accuracy with just 20% of train data is observed to be 64.28%. After Incremental Active Learning, various acquisition functions are utilized for evaluation of the most informative queries. From Figure 5, we can infer that Variation Ratios performs the best, achieving ~88% with just less than 250 data points (less than 50% of total D_{pool}), which is a good trade-off between accuracy and images actively acquired. Random Sampling has the least incremental accuracy as expected.

Incremental Active Learning on Augmented Classes

A similar training mechanism (80-20% – D_{pool}-D_{train} split) as that of Section 7 is followed for existing and augmented classes together as well, with D_{test} used for evaluation. We can infer from Figure 6 with an effective trade-off between accuracy and acquired images that Variation Ratios again performs the best again, with a classification accuracy of ~90% with just ~180 images (~37% of total D_{pool}).

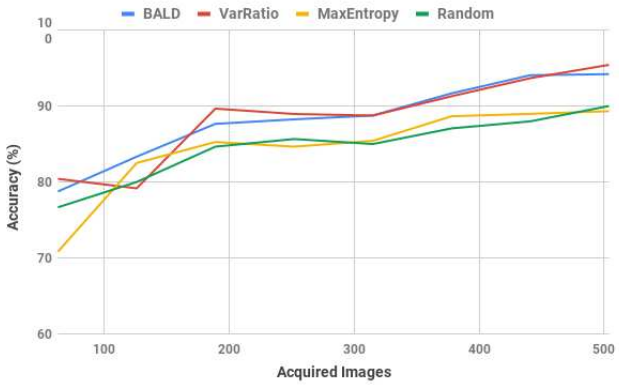


Figure 6: Acquired Images vs Accuracy on augmented classes

After the first acquisition iteration/experiment, the further incremental updates would typically require substantially fewer number of actively queried data points from AL to achieve on-par base classification accuracies of ~96%.

8 INTELLIGENT INFERENCE

Inference Methodology. The authors propose an intelligent inference approach that is built on human intuition. In this approach, the model classifies multiple iterative bus stop images acquired on demand as opposed to conventional classification. Let the number of images captured and classified during inference be n, which is initially set to 2 and capped at 10 ($2 < n < 10$). In order to facilitate intelligent decision making, we propose a *confidence factor* α – which is the ratio of mode of predicted classes to n. Just like a human brain, the system acquires images and classifies them iteratively

until it can assure a confidence of at least α. Typically, the threshold for α is set to a majority among the classified ($\alpha >$ 0.5 for 2 images, $\alpha \geq 0.67$ for 3 images, and so on). In order to simulate real-time testing, a classifier is trained on the aforementioned 720 images and tested iteratively on examples at random from the test data.

The results showcased in Section 7 is a conventional way of testing with stratified splits during Incremental Active Learning. On the contrary, the accuracy of the proposed intelligent inference mechanism would steer the model towards near-100% accuracy, while the bus stop is deemed misclassified only when $n > 10$. However, when simulated across numerous trials, the performance of the model did not falter with the maximum value of n reaching 5.

Inference Engine. The entire deep convolutional network model is sized 266 kB, thus making it ideal for effective on-device training and inference. Dropout at inference time is also used owing to the Bayesian nature of the model for active querying using uncertainty-based acquisition functions. The Incremental Active Learning module can be customized depending on the locality and bus usage characteristics, like periodic per trip update, per day/night update, etc. The various metrics and Inference times on Raspberry Pi 2 are observed in Table 1. Also, many such incremental updates would happen in real-time, and we can observe that for a total of *T=10* dropout iterations, ~12 seconds were used for querying most uncertain data points during one such acquisition iteration.

Table 1: Time taken for Execution

Process	Computational Time
Inference time	11 ms
Incremental Learning per epoch	~1.7ms
Dropout iteration	~1.2ms

9 CONCLUSION AND FUTURE WORK

This paper proposes an intelligent on-device Bus Stop recognition engine which can accurately classify bus stop images in real-time, thereby eliminating the need for GPS network and latency overheads. By systematically optimizing upon different Bayesian Incremental Active Learning methodologies involving multiple acquisition functions, the proposed system adapts to the dynamically evolving nature of bus stops using periodic updates. Variation Ratios acquisition function is observed to perform the best during Active Learning, furthermore, data augmentation strategies are introduced for new bus stops to ensure scalability. Hence, a resource-friendly unified framework for Bus Stop Recognition which facilitates seamless integration with existing self-driving vehicles with image/video recognition capabilities is proposed in this paper. In future, we aim to create a unified (master) model across all buses traveling in the same route with periodic incremental updates from every bus, which enables information sharing across buses, thereby making bus stop recognition truly ubiquitous.

REFERENCES

[1] American public transportation association (apta) 2019 fact book. https://www.apta.com/wp-content/uploads/APTA_Fact-Book-2019_FINAL.pdf, 2019. Online.

[2] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* (2016).

[3] CireÅŞan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation 22*, 12 (2010), 3207âĂŞ3220.

[4] Dasgupta, S., Kalai, A. T., and Monteleoni, C. Aanalysis of perceptron-based active learning. In *International Conference on Computational Learning Theory* (2005), Springer, pp. 249–263.

[5] Franke, U., and Ismail, A. Recognition of bus stops through computer vision. In *IV2003 Intelligent Vehicles Symposium. Proceedings* (2003), IEEE, pp. 650–655.

[6] Freeman, L. C. *Elementary applied statistics: for students in behavioral science.* John Wiley & Sons, 1965.

[7] Gal, Y., and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48* (2016), ICML'16, pp. 1050–1059.

[8] Gal, Y., Islam, R., and Ghahramani, Z. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (2017), ICML'17, pp. 1183–1192.

[9] Garg, N., Ramadurai, G., and Ranu, S. Mining bus stops from raw gps data of bus trajectories. In *10th International Conference on Communication Systems Networks (COMSNETS)* (2018), IEEE, pp. 583–588.

[10] Gudur, G. K., Sundaramoorthy, P., and Umaashankar, V. Activeharnet: Towards on-device deep bayesian active learning for human activity recognition. In *The 3rd International Workshop on Deep Learning for Mobile Systems and Applications* (2019), EMDL '19, ACM, pp. 7–12.

[11] Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745* (2011).

[12] Kendall, A., and Gal, Y. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems* (2017), pp. 5574–5584.

[13] Pan, H., Yi, C., and Tian, Y. A primary travelling assistant system of bus detection and recognition for visually impaired people. In *International Conference on Multimedia and Expo Workshops (ICMEW)* (2013), IEEE, pp. 1–6.

[14] Settles, B. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning 6*, 1 (2012), 1–114.

[15] Shannon, C. E. A mathematical theory of communication. *Bell system technical journal 27*, 3 (1948), 379–423.

[16] Srinivasan, K., and Kalpakis, K. Intelligent bus stop identification using smartphone sensors. In *14th International Conference on Machine Learning and Applications (ICMLA)* (2015), IEEE, pp. 954–959.

[17] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research 15* (2014), 1929–1958.

A Dynamically Adaptive Movie Occupancy Forecasting System with Feature Optimization

Sundararaman Venkataramani
Computer Science
SSN College of Engineering
sundararaman15129@cse.ssn.edu.in

Ateendra Ramesh
Computer Science and Engineering
University at Buffalo
ateendra@buffalo.edu

Sharan Sundar S
Computer Science
SSN College of Engineering
sharansundar15096@cse.ssn.edu.in

Aashish Kumar Jain
Computer Science and Engineering
University at Buffalo
ajain28@buffalo.edu

Gautham Krishna Gudur
Global AI Accelerator
Ericsson
gautham.krishna.gudur@ericsson.com

Vineeth Vijayaraghavan
Director - Research and Outreach
Solarillion Foundation
vineethv@ieee.org

***Abstract*—Demand Forecasting is a primary revenue management strategy in any business model, particularly in the highly volatile entertainment/movie industry wherein, inaccurate forecasting may lead to loss in revenue, improper workforce allocation and food wastage or shortage. Predominant challenges in Occupancy Forecasting might involve complexities in modeling external factors – particularly in Indian multiplexes with multilingual movies, high degrees of uncertainty in crowd-behavior, seasonality drifts, influence of socio-economic events and weather conditions. In this paper, we investigate the problem of movie occupancy forecasting, a significant step in the decision-making process of movie scheduling and resource management, by leveraging the historical transactions performed in a multiplex consisting of eight screens with an average footfall of over 5500 on holidays and over 3500 on nonholidays every day. To effectively capture crowd behavior and predict the occupancy, we engineer and benchmark behavioral features by structuring recent historical transaction data spanning over five years from one of the top Indian movie multiplex chains, and propose various deep learning and conventional machine learning models. We also propose and optimize on a novel feature called *Sale Velocity* to incorporate the dynamic crowd behavior in movies. The performance of these models are benchmarked in real-time using *Mean Absolute Percentage Error (MAPE)*, and found to be highly promising while substantially outperforming a domain expert's predictions.**

***Keywords*—Movie Occupancy Forecasting, Feature Engineering, Machine Learning, Predictive Modeling, Time-Series Forecasting**

I. Introduction

In commercial cinema industry, occupancy prediction has prime importance in organizing and decision making. Foretelling demand allows management personnel to plan appropriately on issues as workforce allocation, supplies, financial budgeting, pricing and inventory. According to the Ficci-KPMG Media and Entertainment Industry Report [1] from 2017, by the end of 2016, there were around 6,000 single screens and around 2,500 multiplex screens in India. Multiplexes have been adding screens at the rate of 8-9% annually over the past few years, in 2016, multiplexes together added approximately 200 screens across the country and trends indicate that the industry is likely to continue to grow at a similar pace, adding 150-200 screens a year. Considering the enormous number of movie-goers and the amount of resources spent, there arises a need for optimizing the expenditure. Occupancy forecasting aims to minimize costs and maximize revenue, thus offering both public and private benefits.

Traditionally, revenue management is associated with statistical techniques which can predict, sometimes with good accuracy, occupancy rates and demand. However, some of the techniques require important statistical skills and lengthy procedures, particularly for Movie Occupancy Forecasting – which requires multiple Key Point Indicators (KPIs) and trend/behavior analysis to be applied in order for them to function accurately. Eliashberg et al. [2] talk about conditional forecasting and demand driven scheduling with traditional statistical approaches. Machine learning models for forecasting are relatively contemporary [3], and utilize learning algorithms to build predictive models for forecasting in the cinema industry – which is more fruitful, considering the amount of advances in the branches of big data and data science. These models can be easily used and deployed by personnel who do not possess much domain expertise or advanced training in statistics. Such models can be encapsulated into relatively inexpensive applications or computed on the cloud, thus providing cost-effective forecasts.

Forecasting is the most crucial component in pricing and budgeting [4], [5]. To develop optimal pricing strategies and minimize resources being spent, we explore and engineer features to capture user-behaviour and propose machine learning models for predictive forecasting, that predict using these learned features. Finally, feature tuning is applied to fine-tune the model according to the business model's requirements and use cases.

The key contributions of this paper are as follows:

- Proposing a historical transactional time-series dataset over five recent years for a top Indian movie multiplex chain.

2375-9259/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDMW.2019.00118

- Benchmarking new features exclusively for Movie Occupancy Forecasting by rigorous empirical experimentation and feature engineering.
- A study of various conventional machine learning and tree-based ensemble models, and state-of-the-art deep learning models including Recurrent Neural Networks (with Long Short Term Memory - LSTM).
- Proposing and optimizing on a novel *sale velocity* feature to capture the dynamic crowd behavior and demand over time.

The rest of the paper is organized as follows. We describe the transactional dataset and the ground truth utilized in Section II. Benchmarking and proposing various feature for Movie Occupancy Forecasting, the reasons behind using them and their importance, i.e, Feature Extraction and Engineering are discussed in Section III. The various machine learning, deep learning models used for time-series forecasting are presented in Section IV. The various metrics pertaining to movie occupancy forecasting are introduced and performance of the models are validated and elucidated in Section V. Section VI proposes a new dynamic feature to capture crowd behavior, and Section VII concludes the paper.

II. Dataset

The data is developed and housed by one of the renowned multiplex chains in India and is a collection of movie ticket booking transactions of a single multiplex sprawling over a period of five years from 2013 to 2017. It encompasses about a million transactions carried out in the following two ways - Offline (in-person ticket counters) and Online (with the corporation's website and mobile application). Each transaction comprises of a unique transaction ID, a movie ID, number of tickets booked, time of booking and other relevant metadata pertaining to the show and transaction. Historical transactional data of the company are stored data dumps (databases) from which the required data was extracted using SQL queries

TABLE I
Screen Capacity across 8 Screens

Screen	Capacity
S1	310
S2	310
S3	110
S4	110
S5	110
S6	131
S7	120
S8	120

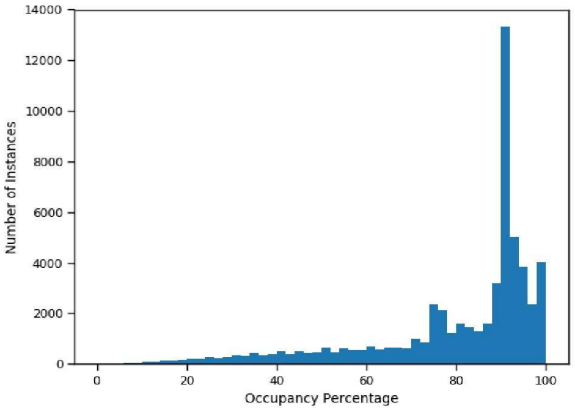
.

The multiplex spans across eight screens with different seating capacities as shown in Table I, operating with an average footfall of ∼5500 on holidays and ∼3500 on nonholidays, with each screen having at least four and/or five movies screened per day on holidays and nonholidays respectively.

Ground Truth (Percentage Tickets Sold - P_{sold})

P_{sold} represents the percentage of tickets sold or the final occupancy percentage of a show in the multiplex which is used as the ground truth for training and validating our forecasting model. Figure 1 shows the histogram of P_{sold} for all the screens from 2013 to 2017.

Fig. 1. Histogram of Occupancy percentages for shows from 2013 - 2017



Forecasting Requirements

A movie being played on a screen at a unique time is referred to as a show in this paper henceforth. The multiplex requires that the prediction of all shows for a day d_i should be a day-ahead i.e., predictions are made by time t_{pred} on the previous day d_{i-1}. Hence, for forecasting the occupancy of a given show, all the transactions occurring in [t_0, t_{pred}) are utilized, with t_0 denoting the time of the first booking made for the show.

III. Feature Engineering and Extraction

Feature engineering is the process of transforming raw data into features that enables the model to identify discriminant characteristics of the data, which results in improved model accuracies [6]. The performance of machine learning models is strongly dependent on the feature engineering phase. Therefore, the data pre-processing and transforming pipelines play a crucial role in deploying machine learning algorithms and is typically domain-specific involving considerable human expertise [7].

To represent a show, transactions leading up to the show cannot be directly fed into any machine learning model as they do not provide a unified representation of a show across characteristics like screen capacity, movie's prior performance, crowd behavior, etc. Hence, we develop features that best represent the shows of a day by leveraging empirical observations in the data, while taking the aforementioned characteristics of the domain into account.

The following relevant features for a movie occupancy forecasting paradigm are benchmarked and finalized by discussing with a domain expert, empirical analysis of the data as well

TABLE II
USEFUL NOTATIONS AND THEIR DESCRIPTION

Notation	Feature/Variable	Description
S_c	Screen Capacity	Total number of seats available i.e., total tickets available for sale
t_{sch}	Scheduled Screening Time	Time at which the show has been scheduled
t_0	Booking Start Time	Time at which the bookings for the show start
t_{pred}	Prediction Time	Time at which the occupancy has to be predicted
P_{held}	Percentage Tickets Held	Tickets unavailable for sale to the general public as percentage of S_c
P_{sold}	Percentage Tickets Sold	Tickets sold for a given show as a percentage of the S_c at t_{sch}
P_{avail}	Percentage Tickets Available	Tickets still available for sale at t_{pred}
P_{pred}	Predicted Ticket Sale Percentage	Tickets predicted to be sold as a percentage of S_c
n_t	Transaction Count	Total number of transactions made for the show until t_{pred}
T_{avg}	Average Tickets Sold per transaction	Mean tickets sold every transaction
v_x	Sale Velocity	Factor that accounts into screen capacity and time to sell $x\%$ tickets
s_x	Slot	Time Slot x of the day at which show is scheduled to be screened
t_{left}	Time Left To Show	Time left in hours for the screening of the show
d_r	Days Since Release	Days elapsed since the release of a show

as performance on the cross-validation data. The features and notations for the same can be observed in Table II.

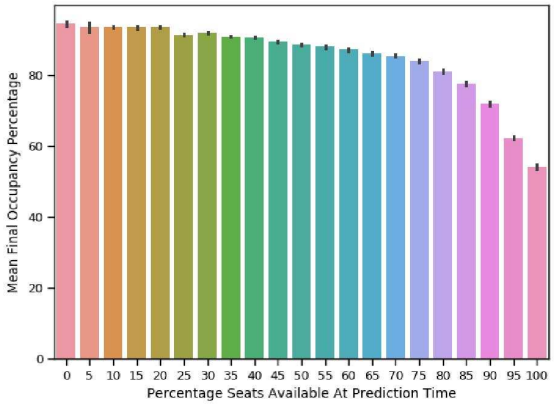
A. Percentage Tickets Held (P_{held})

While initiating ticket sales for a show, a small percentage of tickets denoted by P_{held} is reserved for the multiplex employees and/or its partners, and not open for sale to the general public. We observed that P_{held} directly influences the number of tickets sold, with a positive correlation.

B. Percentage Tickets Available (P_{avail})

The percentage seats available for sale at the time of prediction (t_{pred}) is denoted by P_{avail}. Figure 2 shows that availability of seats is inversely proportional to the total tickets sold (P_{sold}).

Fig. 2. Mean P_{sold} vs P_{avail}



C. Screen Capacity (S_c)

The capacity of a screen is denoted by S_c and their numeric values can be observed in Table I. We infer that for a show with higher S_c, it gets harder to achieve similar percentage of tickets sold (P_{sold}) than a show with lower S_c, since the remaining tickets to be sold are numerically higher.

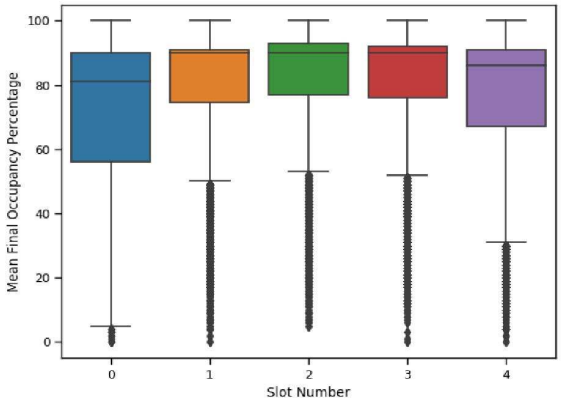
D. Slots (s)

The scheduled time of a show (t_{sch}) is observed to correlate with P_{sold}, since similar crowd-behavior across different days at particular times is observed. Hence, the general operating hours of the multiplex during which shows are played are discretized into five slots as shown in Table III, after experimenting with different number of slots. From the box plot in Figure 3, it can be inferred that shows being screened during s_0 (8 am - 11 AM) are erratic, with a high interquartile range (IQR) for P_{sold} which makes it difficult to model, while slots $s_{1\text{-}3}$ exhibited a more consistent turnout.

TABLE III
START AND END TIMES FOR DIFFERENT SLOTS

Slot	Start Time - End Time
s_0	08:00 - 11:00
s_1	11:00 - 14:00
s_2	14:00 - 17:00
s_3	17:00 - 20:00
s_4	20:00 - 01:00

Fig. 3. Box Plot - Relationship between s and Mean P_{sold}



E. Time Left to Show (t_{left})

Time remaining for the show to be screened at the time of prediction (t_{pred}), denoted by t_{left}, plays a vital role in tickets that will be sold in [t_{pred}, t_{sch}] (also referred to as *Walk-In Crowd* in the later sections of the paper). Figure 4 clearly shows that tickets sold after prediction (new tickets) are dependent on the time to show (t_{left}).

Fig. 4. ($P_{sold} - P_{AtPred}$) vs t_{left}

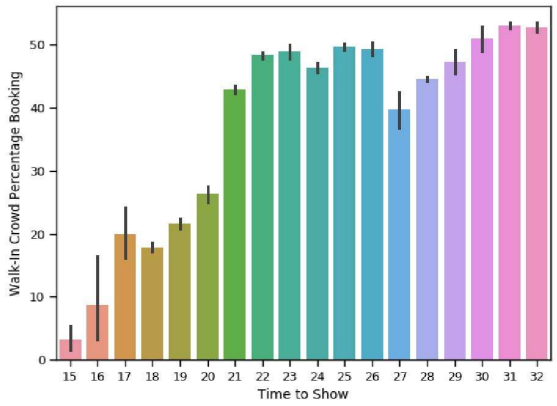
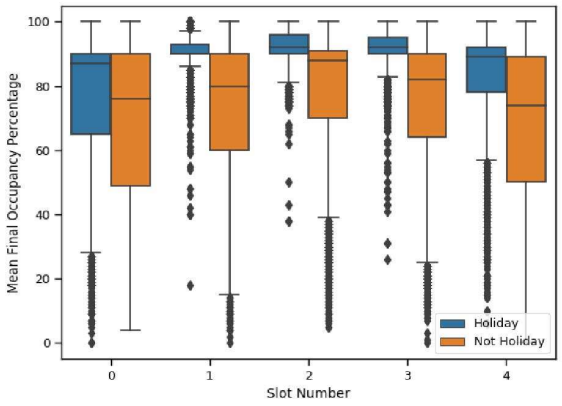


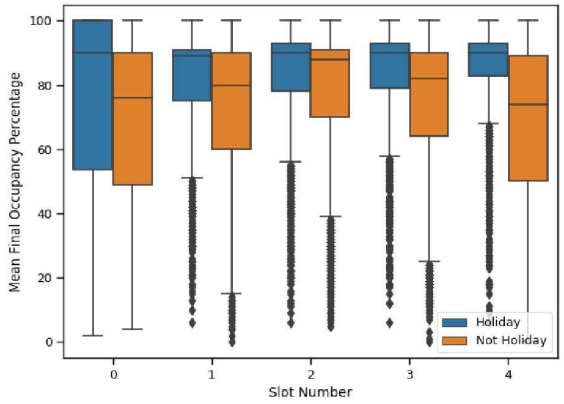
Fig. 5. Box Plot - Relationship between Show day holiday (h_s), Slot (s) and Mean P_{sold}



F. Holiday Factors (h_s and h_n)

The screening of a show in proximity of a holiday is vital in predicting P_{sold} and to model the same. Two binary features are added, namely - Show Day Holiday (h_s) and Next Day Holiday (h_n), where $h_s = 1$ when the show screening day is a holiday, while $h_n = 1$ when the day after the show's screening day is a holiday. These features are used as opposed to adding a 'day of the week' feature to account for holidays during the week. The relationship between h_s and h_n with Mean P_{sold} for shows in different slots (s) have been portrayed in Figures 5 and 6 respectively.

Fig. 6. Box Plot - Relationship between next day Holiday (h_n), Slot (s) and Mean P_{sold}

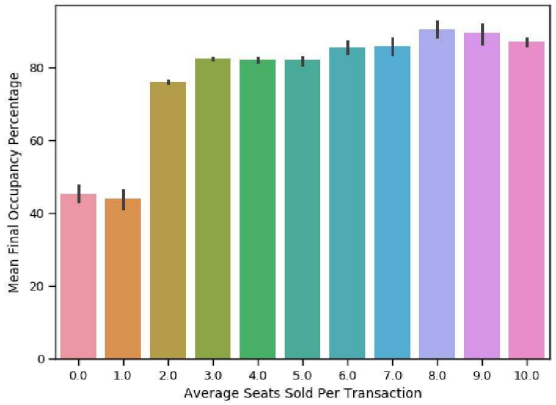


G. Average ticket sold per transaction (T_{avg})

Average number of seats or tickets sold per transaction of a show in [t_0, t_{pred}) is denoted by T_{avg} and is calculated as shown in Equation 1. It is a good indicator of the number of tickets that are likely to be sold in a single transaction that will happen in [t_{pred}, t_{sch}]. The T_{avg} is rounded off to the nearest integer. T_{avg} was observed to correlate with P_{sold} as shown in Figure 7.

$$T_{avg} = \left\lceil \frac{S_c - P_{AtPred}}{n_t} \right\rceil \tag{1}$$

Fig. 7. Box Plot - Correlation between T_{avg} and Mean P_{sold}



H. Days Since Release (d_r)

The feature *Days Since Release* (d_r) is the number of days since a particular movie was first shown and is capped at an empirically chosen maximum value of 200 days, with respect to Indian movie scenarios. We posit that d_r in tandem with the *History-related features* from Section III-I would capture a movie's occupancy pattern over time.

I. History-related Features

History-related features take into account the movie's performance prior to the predictions made for the given movie. Therefore, the most recent k shows were chosen for the same movie to extract historical features which are imperative in understanding the performance of a movie over time, which in turn effects the effectiveness of the forecasting model. These features include the last k historical percentage of tickets sold (P_{sold}), their mean (μ_h), median (m) and standard deviation (σ_h). The optimal value of k for this dataset is found to be 7 after rigorous experimentation on cross-validation data. If there are less than k screenings for the movie for which prediction is to be made, the missing values are set to -1 and are not included in the calculation of μ_h and σ_h.

IV. Models Architectures

We experiment across various ensemble tree-based, deep learning and ensemble models like variants of Gradient Boosting, Extremely Randomized Trees, variants of neural networks and LSTMs, and we describe only the following models which have performed the best. We utilize Grid Search for initial parametric optimization and arrive at the initial set of parameters, followed by introducing a new feature capturing crowd-behavior which is discussed in Section VI.

Conventional Movie Forecasting

Traditionally, the movie occupancy of the multiplex has been manually forecasted day-ahead by observing historical data using the domain expert's knowledge on the performance of the movies being played, their social media and public reception in tandem with reviews from film critiques. They make a prediction for an entire day as opposed to predicting for each show.

A. Extremely Randomized Trees (Extra Trees)

Conventionally, tree-based ensemble models have been effective in time-series paradigms. In this paper, we utilize an Extremely Randomized Trees (Extra Trees) Regressor model as proposed in [8], for the task of predicting P_{sold}. It also has an edge over Random Forests in terms of training and testing speeds. We utilize 100 estimators with minimum sample split and minimum leafs as 5, as the hyper parameters for the model.

B. Deep Neural Networks

Artificial Neural networks are a class of machine learning algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates [9]. A Deep Neural Network (DNN) contains layers of interconnected perceptrons, which is similar performing to multiple linear regressions. Neural nets are widely used because of their ability to generalize and respond to unexpected inputs/patterns. DNNs have the ability to learn and model non-linear and complex relationships and are also known as universal function approximators. The model consists of three dense layers with 128, 64, 32 neurons respectively, each with Leaky-ReLU as the activation function. Dropout with 0.2 probability between each Fully-Connected layer is introduced to reduce over-fitting [11], with the Huber loss function and RMSProp optimizer.

C. Branched LSTMs

Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Networks (RNNs) capable of learning order dependencies in sequence prediction problems [10]. LSTMs have an edge over conventional feed-forward neural networks and RNNs in time-series forecasting problems, owing to their property of selectively remembering patterns over time, and their ability to learn required context to make predictions than being pre-defined.

In [10], it is also established with a variety of experiments that LSTMs can efficiently learn patterns sequential data and/or temporally sequential data. The predictions of n^{th} sample in a sequence of test samples can be influenced by an input present many time steps before. Gating mechanisms are responsible for preserving and releasing the long term dependencies in the network.

In our case, *Show History* is a time sequence feature which pertains to previous occupancies of the show. To include this feature effectively, the model takes in two separate inputs – show history and other features. We combine LSTMs for previous seven *Show History* data points, and Linear Dense layers for the other features resulting in a Branched-LSTM. The show history consists of the previous 7 history points and is directly fed into an LSTM branch. The LSTM branch consists of 3 layers with 128, 64 and 32 units respectively with a recurrent dropout of probability 0.3. The other features are fed into branch of 2 dense layers. The outputs from the LSTM branch and the Dense branch are concatenated with the other features and passed on to the Dense block. The Dense block consists of 3 layers with 128, 64, 32 units respectively. Activation function used is Leaky-ReLU with alpha value of 0.3, with the Huber loss function and RMSProp optimizer, similar to the DNN architecture observed in Section IV-B.

V. Metrics and Results

A. Metrics

In-order to validate the efficiency of the models mentioned in Section IV, we propose two metrics built on *Mean Absolute Percentage Error (MAPE)*, namely – ***Show-wise Error*** and ***Day-wise Error*** denoted by δ_S and δ_D respectively. δ_S for a show is computed as shown in Equation 2.

$$\delta_S = |P_{pred} - P_{sold}| \tag{2}$$

where, P_{pred} denotes the predicted sold ticket percentage.

We also compute Day-wise forecast to compare the model's performance against the corporation's current model mentioned in Section IV, with the model's day-wise error δ_D for a day D with shows S computed using Equation 3.

$$\delta_D = \frac{\left| \sum_{s \in S} P_{pred_s} S_{c_s} - \sum_{s \in S} P_{sold_s} S_{c_s} \right|}{\sum_{s \in S} P_{sold_s} S_{c_s}} * 100 \quad (3)$$

We also use *Mean Absolute Error (MAE)* defined as the average of the absolute differences in the actual turnout $P_{sold_s} S_{cs}$ and the predicted turnout $P_{pred_s} S_{cs}$ for all shows S of day D where $s \in S$. We utilize MAE for day wise predictions.

We also use E_{10}, the percentage of shows with MAPE < 10 for comparing the results along with $\mathbf{R^2}$score that is used to measure the performance of the regression models.

B. Data Split

The multiplex data is split into train, validation and test sets with shows from 2013-2016 used as the training set, first half of 2017 used for cross-validation and the second half of 2017 used for testing.

C. Show-wise and Day-wise Results

The show-wise & day-wise MAPE, MAE, and $\mathbf{R^2}$-score for the models discussed in Section IV have been shown in Tables IV and V respectively. It can be observed across all models that the Day-wise MAPE was quantifiably lower than the Show-wise MAPE which can be attributed to the fact that prediction errors from different shows of a day cancelled out each other. Furthermore, Extra-trees and B-LSTM models performed similarly with respect to the three metrics for both Show-wise and Day-wise predictions whereas, the same metrics calculated for DNN were sub-par in comparison.

Models	MAPE	$\mathbf{E_{10}}$	$\mathbf{R^2}$ Score
Extra Trees	7.70	73.74	0.92
DNN	9.02	67.91	0.89
B-LSTM	7.76	75.15	0.91

TABLE IV
SHOW WISE RESULTS

Models	MAPE	$\mathbf{E_{10}}$	MAE
Extra Trees	5.19	86.58	180.58
DNN	7.62	75.5	266.18
B-LSTM	5.54	84.46	200.37

TABLE V
DAY WISE RESULTS

VI. Capturing Crowd-Behavior

A. Sale Velocity (v_x)

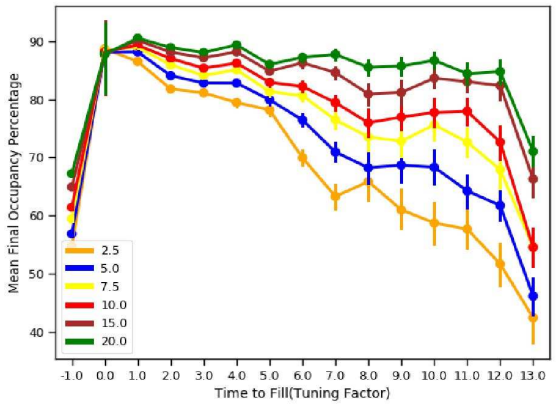
The model used only static features and does not take the demand for the show tickets into account, which can dynamically change over the course of the booking period thereby, leading to large δ_S as static features do not capture the changing demand.

We therefore use Sale velocity, denoted by v_x, to capture the rate at which the tickets are sold for a show. v_x indicates the ticket demand for the show and is calculated for different ticket sale percentages as a feature to capture the demand for the show. v_x can be defined as the ratio between $x\%$ of Screen Capacity (S_{cx}) and the difference between the time taken when $x\%$ of tickets are sold (t_x) and booking start time (t_0). The velocity of sale of $x\%$ tickets is calculated as shown in Equation 4.

$$v_x = \frac{S_{cx}}{t_x - t_0} \quad (4)$$

A screen with a seats may have higher velocity than a screen with b seats, given $a < b$, even when the same number of tickets have been sold. It can be inferred that S_c is directly proportional to the time taken to fill $x\%$ seats, hence the reason for utilizing S_c in the computation of v_x. If $x\%$ of tickets have not been sold at prediction time (t_{pred}), the sale velocity is set to -1. Figure 8 shows the correlation between mean percentage of final tickets sold and time to fill (t_{left}) for the aforementioned ticket-sale percentages.

Fig. 8. Line plot showing the correlation between Mean P_{sold} and t_{left} given v_x for different x (shown in the legend)



We can also observe that P_{sold} increases with increase in v_x for a given time. For instance, if a show has sold 20% of S_c in the same time as another show to sell 5% of S_c, then the former indeed must have a higher demand and thereby, higher occupancy. Hence, v_x is the dynamic feature which covers most ground when it comes to capturing crowd-behaviour.

Models	MAPE	$\mathbf{E_{10}}$	$\mathbf{R^2}$ Score
Extra Trees	7.71	73.9	0.92
ANN	8.52	71.65	0.90
B-LSTM	7.75	74.87	0.91

TABLE VI
SHOW WISE RESULTS WITH v_x

We incorporate Sale Velocity (v_x) as a feature with values of x as $[2.5, 5, 7.5, 10, 15, 20]$. From the results shown in Table VI & VII in comparison with Tables IV & V, it can be inferred that using v_x leads to a reduction in MAE for the day-wise predictions. However, the sale velocity is not available for all

Models	MAPE	E_{10}	MAE
Extra Trees	5.14	87.58	179.23
DNN	5.04	86.58	187.63
B-LSTM	4.97	85.91	180.33

TABLE VII
DAY WISE RESULTS WITH v_x

the shows as $x\%$ of the tickets might not be sold for all values of x. We observed that the percentage of shows with error less than 10% increases with increase in x and this is can be used to indicate the error or prediction which might help to model the uncertainty.

VII. CONCLUSION

This paper discusses the problem of Movie Occupancy Forecasting in the present entertainment industry, and introduces new benchmarked features which can be effectively used to efficiently forecast the occupancy of a movie multiplex. Furthermore, the features engineered for this work are benchmarked and validated on tree-based ensemble machine learning models, and state-of-the-art deep learning architectures. The customer/crowd behavior over time was taken into account and a new *Sale Velocity* feature is proposed and optimized for maximizing the performance of the model. The aforementioned domain expert in Section IV performed with an MAPE of 8.04 which is outperformed by the models proposed. We believe our work can pave the way and act as a benchmark for effective Demand Forecasting, particularly – Movie Occupancy Forecasting systems.

VIII. ACKNOWLEDGMENT

The authors would like to acknowledge Solarillion Foundation for its support and funding of the research work carried out.

IX. FUTURE WORK

During real time prediction, the models discussed in the paper might become stale and subject to concept drift. One such mechanism to overcome this would be to supplement these models with incremental learning mechanisms that would ensure they stay up to date with the most recent and relevant trends. Also, movie occupancy predictions can be used in conjunction with other systems that address use cases such as dynamically pricing movie tickets and food sales forecasting [12].

REFERENCES

[1] *FICCI Media and Entertainment Report 2017*, KPMG India.
[2] Yue, X. and Liu, J., 2006. Demand forecast sharing in a dual-channel supply chain. European Journal of Operational Research, 174(1), pp.646–667.
[3] Caicedo, William and Payares, Fabin, *A Machine Learning Model for Occupancy Rates and Demand Forecasting in the Hospitality Industry*, 15th Ibero-American Conference on AI, San Jos, Costa Ric.,(2016), pp. 201–211.
[4] Lee, A.O., *Airline Reservations Forecasting: Probabilistic and Statistical Models of the Booking Process.*,Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA (1990).
[5] Weatherford, L.R., Kimes, S.E., *A Comparison of Forecasting Methods for Hotel Revenue Management*, International Journal of Forecasting 19(3), pp.401415 (2003).
[6] Jeff Heaton , *An Empirical Analysis of Feature Engineering for Predictive Modeling*, CoRR,abs/1701.07852, (2017).
[7] Yoshua Bengio and Aaron C. Courville and Pascal Vincent, *Unsupervised Feature Learning and Deep Learning: A Review and New Perspectives*, CoRR, abs/1206.5538, (2012).
[8] Geurts, Pierre and Ernst, Damien and Wehenkel, Louis , *Extremely Randomized Trees*, Machine Learning, (2006), pp.3–42.
[9] Schmidhuber, J., *Deep Learning in Neural Networks: An overview*, Neural Networks,Volume 61, January 01, 2015, pp.85–117.
[10] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory", Neural Computation, vol. 9, no. 8, pp.1735-1780, 1997.
[11] Srivastava, Nitish and Hinton, Geoffrey and Krizhevsky, Alex and Sutskever, Ilya and Salakhutdinov, Ruslan, *Dropout: A Simple Way to prevent Neural Networks from Overfitting*, The Journal of Machine Learning Research, (2014), pp.1929–1958.
[12] Adithya Ganesan V., Divi S., Moudhgalya N.B., Sriharsha U., Vijayaraghavan V. (2020) Forecasting Food Sales in a Multiplex Using Dynamic Artificial Neural Networks. In: Arai K., Kapoor S. (eds) Advances in Computer Vision. CVC 2019. Advances in Intelligent Systems and Computing, vol 944. Springer, Cham

Resource-Constrained Federated Learning with Heterogeneous Labels and Models

Gautham Krishna Gudur
Global AI Accelerator, Ericsson
gautham.krishna.gudur@ericsson.com

Bala Shyamala Balaji
Indian Institute of Technology, Madras
balashyamala@gmail.com

Perepu Satheesh Kumar
Ericsson Research
perepu.satheesh.kumar@ericsson.com

ABSTRACT

Various IoT applications demand resource-constrained machine learning mechanisms for different applications such as pervasive healthcare, activity monitoring, speech recognition, real-time computer vision, etc. This necessitates us to leverage information from multiple devices with few communication overheads. Federated Learning proves to be an extremely viable option for distributed and collaborative machine learning. Particularly, on-device federated learning is an active area of research, however, there are a variety of challenges in addressing statistical (non-IID data) and model heterogeneities. In addition, in this paper we explore a new challenge of interest – to handle *label heterogeneities* in federated learning. To this end, we propose a framework with simple α-weighted federated aggregation of scores which leverages overlapping information gain across labels, while saving bandwidth costs in the process. Empirical evaluation on Animals-10 dataset (with 4 labels for effective elucidation of results) indicates an average deterministic accuracy increase of at least ~16.7%. We also demonstrate the on-device capabilities of our proposed framework by experimenting with federated learning and inference across different iterations on a Raspberry Pi 2, a single-board computing platform.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; *Distributed computing methodologies.*

KEYWORDS

On-Device Federated Learning, Heterogeneous Labels, Heterogeneous Models, Transfer Learning

ACM Reference Format:
Gautham Krishna Gudur, Bala Shyamala Balaji, and Perepu Satheesh Kumar. 2020. Resource-Constrained Federated Learning with Heterogeneous Labels and Models. In *The 3rd International Workshop on Artificial Intelligence of Things (AIoT'20) at KDD 2020: The 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 22-27, 2020, San Diego, CA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

KDD'20 Workshops: AIoT'20, August 22-27, 2020, San Diego, CA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Contemporary machine learning, particularly deep learning has led to major breakthroughs in various domains, such as computer vision, natural language processing, speech recognition, Internet of Things (IoT), etc. Particularly, on-device machine learning has spiked up a huge interest in the research community owing to the compute capabilities vested in resource-constrained devices like mobile and wearable devices. Sensor data from various IoT devices have a vast amount of incoming data which have massive potential to leverage such on-device machine learning techniques on-the-fly to transform them into meaningful information coupled with supervised, unsupervised and/or other learning mechanisms. With the ubiquitous proliferation of such personalized IoT devices, collaborative and distributed learning is now possible more than ever to help best utilize the information learnt from multiple devices.

However, such collaborative data sharing across devices might always not be feasible owing to privacy concerns from multiple participants. Moreover, users might not prefer nor have any interest in sending their sensitive data to a remote server/cloud, particularly in fields like health-care, defense, telecommunication, etc. With the advent of *Federated Learning (FL)* [17], [2], it is now possible to effectively train a global/centralized model without compromising on sensitive data of various users by enabling the transfer of model weights and updates from local devices to the cloud, instead of conventionally transferring the sensitive data to the cloud. A server has the role of coordinating between models, however most of the work is not performed by a central entity anymore but by a federation of users. The *Federated Averaging (FedAvg)* algorithm [17] was proposed by McMahan et al. which aggregates the model parameters of each client (local device) by combining local Stochastic Gradient Descent (SGD) through a server. Federated learning has been an active and challenging area of research in solving problems pertaining to secure communication protocols, device and statistical heterogeneities, and privacy preserving networks [12].

Federated Learning deals with various forms of heterogeneities like device, system, statistical heterogeneities, etc. [12], [19]. Particularly, FL in IoT and edge computing scenarios, statistical heterogeneities have gained much visibility as a research problem predominantly owing to the non-IID (non-independent and identically distributed) nature of the vast amounts of streaming real-world data incoming from distinct distributions across devices. This leads to challenges in personalized federation of devices, and necessitates us to address various heterogeneities in data and learning processes for effective model aggregation.

One important step in this direction is the ability of end-users to have the choice of architecting their own models, rather than being constrained by the pre-defined architecture mandated by the global

model for model aggregation. One effective way to circumvent this problem is by leveraging the concept of knowledge distillation [5], wherein the disparate local models distill their respective knowledge into various *student models* which has a common model architecture, thereby effectively incorporating model independence and heterogeneity. This was proposed by Li et al. in FedMD [11]. However, as much independence and heterogeneity in architecting the users' own models is ensured in their work, they do not guarantee *heterogeneity and independence in labels across devices.*

For a concrete example, a central intelligent system/service acts as the global cloud/server with multiple telecommunication operators being the clients in a typical federated learning scenario. However, most operators (clients) typically choose to have their own machine learning model architectures, and also have their own labels which are governed by few standardized and also localized company bodies. This necessitates seamless interaction of the local clients with the global cloud/server independent of model architectures and labels. Many such scenarios and settings with such heterogeneous labels and models exist in federated IoT/on-device settings, such as behaviour/health monitoring, activity tracking, keyword spotting, next-word prediction, etc. Few works address handling new labels in typical machine learning scenarios, however, to the best of our knowledge, there is no work which addresses this important problem of *label and model heterogeneities* in non-IID federated learning scenarios.

The main scientific contributions in this work are as follows:

- Enabling end-users to build and characterize their own preferred local architectures in a federated learning scenario, so that effective transfer learning happens between the global and local models.
- A framework to allow flexible heterogeneous selection of labels by showcasing scenarios with and without overlap across different user devices, thereby leveraging the information learnt across devices pertaining to those overlapped classes.
- Empirical demonstration of the framework's ability to handle different data distributions (statistical heterogeneities and non-IIDness) from various user devices.
- Demonstrating the feasibility of on-device personalized federated learning from incoming real-world data independent of users, capable of running on simple mobile and wearable devices.

2 RELATED WORK

Federated Learning has contributed vividly in enabling distributed and collective machine learning across various devices. Federated learning and differentially private machine learning have/soon will emerge to become the de facto mechanisms for dealing with sensitive data, and data protected by Intellectual Property rights, GDPR, etc [2]. Federated Learning was first introduced by McMahan et al. in [17], and new challenges and open problems to be solved, aggregation and optimization methods, [12], strategies [10], and multiple advancements [7] have been proposed and addressed in many interesting recent works.

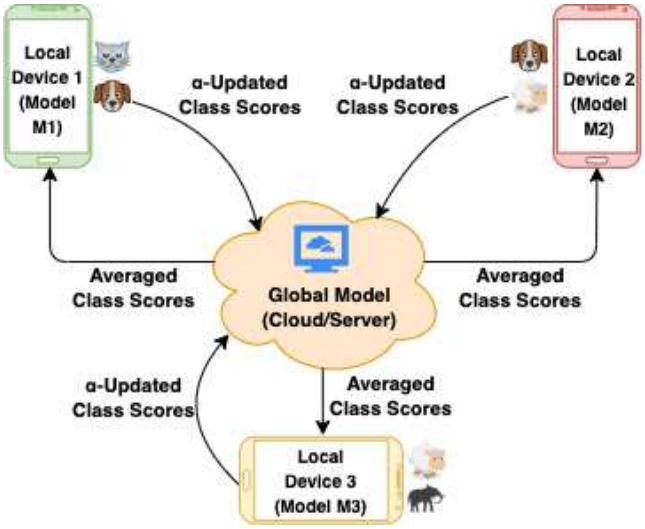


Figure 1: Overall Architecture of the proposed framework with *Weighted Local Update with* α. Each mobile device can consist of disparate set of local labels and models, and they interact with the global model (cloud/server). The models in each local device are first updated using weighted α score, the respective class scores are then aggregated in the global model, and the updated consensus is again distributed across local models.

Particularly for Federated Learning in IoT and pervasive (mobile/wearable/edge) devices, important problems and research directions are addressed FL on mobile and edge networks in this survey [16]. Federated Optimization for on-device applications is discussed in [9].

Multiple device/client and system heterogeneities, including client communication mechanisms and fair resource allocation, inherently making most of them optimization problems, and are addressed in various works [13], [4], [19], [14]. Personalized federated learning closely deals with optimizing the degree of personalization and contribution from various clients, thereby enabling effective aggregation as discussed in [3].

Recent works on convergence of Federated Averaging (FedAvg) algorithm on disparate data distributions – non-IID data, and creating a small subset of data globally shared between all edge devices are proposed in [20], [15] respectively. Mohri et al. propose Agnostic Federated Learning [18], which addresses about handling any target data distribution formed by a mixture of client distributions.

FedMD [11], which we believe to be our most closest work, deals with heterogeneities in model architectures, and addresses this problem using transfer learning and knowledge distillation [5], and also uses an initial public dataset across all labels (which can be accessed by any device during federated learning). On similar grounds, federated distillation and augmentation in non-IID data distributions are used in [6]. Current federated learning approaches predominantly handle same labels across all the users and do not provide the flexibility to handle unique labels. However, in many practical applications, having unique labels for each local client/model is a

Table 1: Model Architectures (filters/units in each layer indicated within parentheses), Labels and Number of Images per federated learning iteration across user devices. Note the disparate model architectures and labels across users.

	User_1	User_2	User_3	Global_User
Model Architecture	2-Layer CNN (16, 32) Softmax Activation	3-Layer CNN (16, 16, 32) ReLU Activation	2-Layer CNN (16, 32) ReLU Activation	–
Labels	{Cat, Dog}	{Dog, Sheep}	{Sheep, Elephant}	{Cat, Dog, Sheep, Elephant}
Images per iteration	{500, 500} = 1000	{500, 500} = 1000	{500, 500} = 1000	{500, 500, 500, 500} = 2000

very viable and common scenario owing to their dependencies and constrains on specific regions, demographics, etc. To the best of our knowledge, none of the works take into account, label and model heterogeneities.

The rest of the paper is organized as follows. Section 3 discusses the problem formulation of handling heterogeneous labels and models in on-device federated learning scenarios (section 3.1), and section 3.2 presents the overall proposed framework and the methods used to address these challenges. Systematic experimentation and evaluation of the framework across different users, devices, iterations, models, labels in a federated learning setting is showcased in section 4, while also proving feasibility of the same on resource-constrained devices (section 4.2). Finally, section 5 concludes the paper.

3 OUR APPROACH

In this section, we discuss in detail about the problem formulation of heterogeneity in labels and models, and our proposed framework to handle the same (showcased in Figure 1).

3.1 Problem Formulation

We assume the following scenario in federated learning. There are multiple local devices which can characterize different model architectures based on the end users. The incoming streaming real-world data in all the devices is non-IID in nature, wherein the distribution and data characteristics differ across devices. We hypothesize that the incoming data to the different devices also consist of heterogeneities in labels, with either unique or overlapped labels. We also have a public dataset with the label set consisting of all labels pre-defined – this can be accessed by any device anytime, and acts as an initial template of the data and labels that can stream through, over different iterations. We repurpose this public dataset as the test set also, so that consistency is maintained while testing. To make different FL iterations independent from the public dataset, we do not involve the public dataset during federated learning (training) in the local models. The research problem here is to create a unified framework to handle heterogeneous models, labels and data distributions (with non-IID nature) in a federated learning setting.

3.2 Proposed Framework

The proposed framework and methods to handle the heterogeneous labels and models in a federated learning setting is presented in Algorithm 1. There are three important steps in the proposed method.

Algorithm 1: Proposed Framework to handle heterogeneous labels and models in Federated Learning

Input - Public Data set $\mathcal{D}_0\{x_0, y_0\}$, Private datasets $\mathcal{D}_m^i$, Total users M, Total iterations I, LabelSet for each user l_m

Output - Trained Model scores f_G^I

Initialize - $f_G^0 = \mathbf{0}$ (Global Model Scores)

for $i = 1$ **to** I **do**

  for $m = 1$ **to** M **do**

    Build: Model $\mathcal{D}_m^i$ and predict $f_{\mathcal{D}_m^i}(x_0)$

    Local Update: $f_{\mathcal{D}_m^i}(x_0) = f_G^I(x_0^{l_m}) + \alpha f_{\mathcal{D}_m^i}(x_0)$, where $f_G^I(x_0^{l_m})$ are the global scores of only the set of labels l_m with the m^{th} user, and $\alpha = \frac{len(\mathcal{D}_m^i)}{len(\mathcal{D}_0)}$

    Global Update: Update label wise,

    $$f_G^{i+1} = \sum_{m=1}^{M} \beta_m f_{\mathcal{D}_m^i}(x_0), \text{ where,}$$

    $$\beta = \begin{cases} 1 & \text{If labels are unique} \\ \text{acc}(f_{\mathcal{D}_m^i}(x_0)) & \text{If labels overlap} \end{cases}$$

  end for

end for

Build: In this step, we build the model on the incoming data we have in each local user, i.e., local private data for the specific iteration. The users can choose their own model architecture which suits best for the data present in the iteration.

Local Update: In this step, we update the averaged global model scores (on public data) for the i^{th} iteration on the local private data. For the first iteration, we do not have any global scores and we initialize the scores to be zero in this case. For the rest of iterations, we have global averaged scores which we can use to update the local model scores according to Algorithm 1. In the local update, we propose a *weighted α-update*, where α is the ratio between the size of current private dataset and the size of public dataset, and governs the contributions of the new and the old models across different federated learning iterations.

Global Update: In this step, we first train the local model on the respective private datasets for that FL iteration. Further, we evaluate (test) this trained model on the public data, thereby obtaining the model scores on public data. We then perform the same operation across all the users and average them using the β parameter, where

β governs the weightage given to overlapping labels across users using test accuracies of the corresponding labels on public data (as given in Algorithm 1). This module gives the global averaged scores.

4 EXPERIMENTS AND RESULTS

We simulate a Federated Learning scenario with multiple iterations of small chunks of incremental data incoming (details in Table 1), across three different users to test our approach, and assume that the images arrive in real-time in the users' devices. We use the *Animals-10 dataset* [1], which consists of ten different animal categories (labels) extracted from Google Images. Now, we discuss the settings for label and model heterogeneities in our experiment.

Label Heterogeneities: In our experiment, we consider only four labels – {*Cat, Dog, Sheep, Elephant*} from the dataset as shown in Table 1. Also, we include the number of images considered per user per iteration (500 images per iteration). The labels in each local user can either be unique (present only in that single user) or overlapped across users (present in more one user). We split the four labels into three pairs of two labels each, for convenience of showcasing the advantage of overlapping labels in experimentation. We also create a non-IID environment across different federated learning iterations wherein, the image data across different iterations are split with disparities in both labels and distributions of data (*Statistical Heterogeneities*).

Table 2: Details of Model Architectures (filters/units in each layer indicated within parentheses) changed across federated learning iterations and users.

Iteration	New Model Architecture
User_1 Iteration_10	3-Layer ANN (16, 16, 32) ReLU Activation
User_1 Iteration_14	1-Layer CNN (16) Softmax Activation
User_2 Iteration_6	3-Layer CNN (16, 16, 32) Softmax activation
User_3 Iteration_5	4-Layer CNN (8, 16, 16, 32) Softmax activation

Model Heterogeneities: We choose three different model architectures for the three different local user device. This is clearly elucidated in Table 1. To truly showcase near-real-time heterogeneity and model independence, we change the model across and within various FL iterations as shown in Table 2.

Initially, we divide the images across different users according to the four labels. Each image is re-scaled to size 128*128. We create a Public Dataset - D_0 with 2000 images, with 500 images corresponding to each label. Next, we sample 500 images in every iterations as per the labels of the user and we use them for our problem (as shown in Table 1). In total, we ran 15 iterations in this whole federated learning experiment, with each iteration running with early stopping (max 5 epochs). We track the loss using categorical cross-entropy loss function for multi-class classification, and use the Adam optimizer [8] to optimize the classification loss. We simulate all our experiments – both federated learning and inference on a *Raspberry Pi 2*.

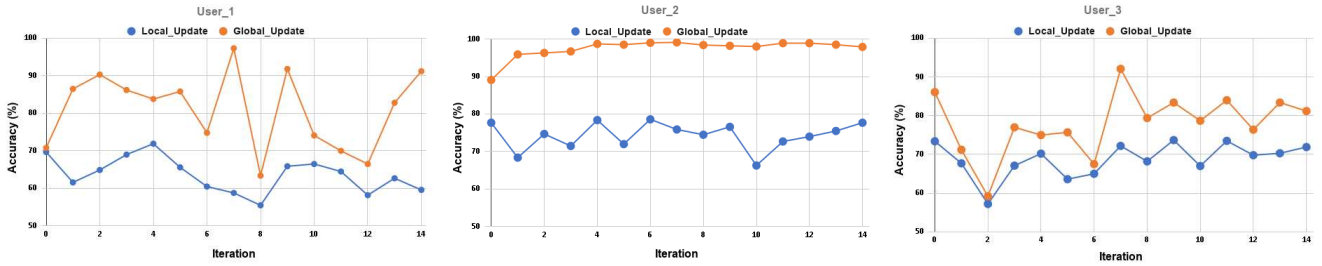
4.1 Discussion on Results

Figure 2 represents the results across all three users on Animals Dataset. Also, from Table 3, we can clearly observe that the global updates – which represent the accuracies of the global updated model (and averaged across all users' labels in the ith iteration governed by β), are higher for all three users than the accuracies of their respective local updates. For instance, from Figure 2a, we can infer that the corresponding accuracies of labels {*Cat, Dog*} (User 1 labels) after global updates in each iteration are deterministically higher than their respective local updates by an average of ~17.4% across all iterations with weighted α-update. Similarly for User 2, labels consisting of {*Dog, Sheep*}, we observe an average accuracy increase of ~23.2% from local updates to the global updates, while for User 3 labels consisting of {*Sheep, Elephant*}, we observe an average increase of ~9.3% from local updates to global updates.

Table 3: Average Accuracies (%) of Local and Global Updates, and their respective Accuracy increase with *Weighted α-update*

	Local_Update	Global_Update	Acc. Increase
User_1	63.66	81.02	17.36
User_2	74.3	97.47	**23.17**
User_3	68.72	78.02	9.3
Average	**68.89**	**85.5**	**16.61**

We would like to particularly point out that the overlap in labels significantly contributed to highest increase in accuracies, owing to the fact that information gain (weighted global update) happens only for overlapping labels. This is vividly visible in User 2 (Figure 2b, whose labels are {*Dog, Sheep*}), where in spite of an accuracy dip in local update at iteration 10, the global update at that iteration does not take a spike down which can be primarily attributed to the information gain from overlapped labels between User 1 and User 3 (in this case, *Dog* and *Sheep* respectively), thereby showcasing the robustness of overlapped label information gain in User 2. On the contrary, when we observe User 1 (Figure 2a), in spite of the accuracies of global updates being inherently better than local updates, when a dip in accuracies of local updates are observed at iteration 8 and 12, the accuracies of global updates at that iteration also spike down in a similar fashion. Similar trends of local and global accuracy trends like those observed in User 1 can also be observed in User 3 (Figure 2c). This clearly shows that when there are lesser overlapped labels (User 1 and User 3), the global model does not learn the label characteristics as much as when there are more overlapped labels – the global updates are more robust in spite of spikes and dips in local updates with overlapped labels (User 2), thereby leading to higher average increase in accuracies (as observed in Table 3).



(a) User_1 (b) User_2 (c) User_3

Figure 2: Iterations vs Accuracies across all three users with Weighted Local Update with α. *Local_Update* signifies the accuracy of each local updated model (after i^{th} iteration) on Public Dataset. *Global_Update* signifies the accuracy of the corresponding global updated model (after i^{th} iteration) on Public Dataset.

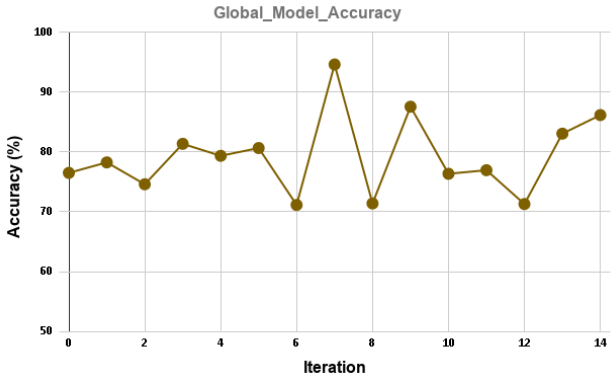


Figure 3: Iterations vs Final Global Average Accuracies (%) with *Weighted α-update*

An overall average increase of ~16.7% deterministic accuracy increase (not relative) is observed with our proposed framework across all the different users, which can be calculated from the global model updates. The overall global model accuracies alone (which is different from global update accuracies observed in Figure 2) after each iteration are also elucidated in Figure 3. We propose that, with our current framework, communication (transfer) of just the model scores of respective labels between clients (local devices) and the central cloud is enough, without necessitating transfer of the whole model weights, which significantly reduces latency and memory overheads. We also point out that we tried local model distillation similar to the work done in FedMD [11] for handling label and model heterogeneities, however, we felt using only score consensus from the model without distillation yielded far lesser on-device computation times.

4.2 On-Device Performance

We observe the on-device performance of our proposed framework, which is experimented on a Raspberry Pi 2. We choose this single-board computing platform since it has similar hardware and software (HW/SW) specifications with that of predominant contemporary IoT/edge/mobile devices. The computation times taken for execution of on-device federated learning and inference are reported in Table 4. This clearly shows the feasibility of our proposed system on embedded devices.

Table 4: Time taken for Execution

Process	Computation Time
Training time per epoch in an FL iteration (i)	~1.8 sec
Inference time	~15 ms

5 CONCLUSION

This paper presents a unified framework for flexibly handling heterogeneous labels and model architectures in federated learning in a non-IID fashion. By leveraging transfer learning along with simple scenario changes in the federated learning setting, we propose a framework with α-update aggregation in local models, and we are able to leverage the effectiveness of global model updates across all devices and obtain higher efficiencies. Moreover, overlapping labels are found to make our framework robust, and also helps in effective accuracy increase. We empirically showcase the successful feasibility of our framework on resource-constrained devices for federated learning/training and inference across different iterations on the Animals-10 dataset. We expect a good amount of research focus hereon in developing statistical, model and label based heterogeneities.

REFERENCES

[1] Alessio, C. *Animals-10 Dataset*, 2019. https://www.kaggle.com/alessiocorrado99/animals10.
[2] Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konecny, J., Mazzocchi, S., McMahan, H. B., Overveldt, T. V., Petrou, D., Ramage, D., and Roselander, J. Towards federated learning at scale: System design. In *SysML 2019* (2019).
[3] Deng, Y., Kamani, M. M., and Mahdavi, M. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461* (2020).
[4] Ghosh, A., Hong, J., Yin, D., and Ramchandran, K. Robust federated learning in a heterogeneous environment. *arXiv preprint arXiv:1906.06629* (2019).
[5] Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop* (2015).
[6] Jeong, E., Oh, S., Kim, H., Park, J., Bennis, M., and Kim, S.-L. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479* (2018).

[7] Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977* (2019).

[8] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[9] Konečnỳ, J., McMahan, H. B., Ramage, D., and Richtárik, P. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527* (2016).

[10] Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning* (2016).

[11] Li, D., and Wang, J. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581* (2019).

[12] Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine 37*, 3 (2020), 50–60.

[13] Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems 2020* (2020), 429–450.

[14] Li, T., Sanjabi, M., Beirami, A., and Smith, V. Fair resource allocation in federated learning. In *8th International Conference on Learning Representations (ICLR)* (2020).

[15] Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. On the convergence of fedavg on non-iid data. In *8th International Conference on Learning Representations (ICLR)* (2020).

[16] Lim, W. Y. B., Luong, N. C., Hoang, D. T., Jiao, Y., Liang, Y.-C., Yang, Q., Niyato, D., and Miao, C. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials* (2020).

[17] McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics* (2017), vol. 54, pp. 1273–1282.

[18] Mohri, M., Sivek, G., and Suresh, A. T. Agnostic federated learning. In *Proceedings of the 36th International Conference on Machine Learning* (2019), vol. 97, pp. 4615–4625.

[19] Nishio, T., and Yonetani, R. Client selection for federated learning with heterogeneous resources in mobile edge. In *IEEE International Conference on Communications (ICC)* (2019), pp. 1–7.

[20] Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., and Chandra, V. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582* (2018).

A Generic Multi-modal Dynamic Gesture Recognition System using Machine Learning

Gautham Krishna G[α], Karthik Subramanian Nathan[β], Yogesh Kumar B[γ], Ankith A Prabhu[δ],
Ajay Kannan[π], Vineeth Vijayaraghavan[ϵ]
Research Assistant[αγ], Undergraduate Student[βδπ], Director - Research & Outreach, Solarillion Foundation, Chennai, India[ϵ]
Solarillion Foundation[αγϵ], College of Engineering, Guindy[βπ], SRM University[δ]
Email: (gautham.krishna[α], nathankarthik[β], yogesh.bkumar[γ], ankithprabhu[δ], ajaykannan[π], vineethv[ϵ])@ieee.org

***Abstract*—Human computer interaction facilitates intelligent communication between humans and computers, in which gesture recognition plays a prominent role. This paper proposes a machine learning system to identify dynamic gestures using tri-axial acceleration data acquired from two public datasets. These datasets, uWave and Sony, were acquired using accelerometers embedded in Wii remotes and smartwatches, respectively. A dynamic gesture signed by the user is characterized by a generic set of features extracted across time and frequency domains. The system was analyzed from an end-user perspective and was modelled to operate in three modes. The modes of operation determine the subsets of data to be used for training and testing the system. From an initial set of seven classifiers, three were chosen to evaluate each dataset across all modes rendering the system towards mode-neutrality and dataset-independence. The proposed system is able to classify gestures performed at varying speeds with minimum preprocessing, making it computationally efficient. Moreover, this system was found to run on a low-cost embedded platform – Raspberry Pi Zero (USD 5), making it economically viable.**

***Keywords*—Gesture recognition; accelerometers; feature extraction; machine learning algorithms**

I. Introduction

Gesture recognition can be defined as the perception of non-verbal communication through an interface that identifies gestures using mathematical, probabilistic and statistical methods. The field of gesture recognition has been experiencing a rapid growth amidst increased interests shown by researchers in the industry. The goal of current research has been the quick and accurate classification of gestures with minimalistic computation, whilst being economically feasible. Gesture recognition can find use in various tasks such as developing aids for the audio-vocally impaired using sign language interpretation, virtual gaming and smart home environments.

Modern gesture recognition systems can be divided into two broad categories - vision based and motion based systems. The vision based system proposed by Chen et al. in [1] uses digital cameras, and that proposed by Biswas et al. in [2] uses infrared cameras to track the movement of the user. For accurate classification of the gestures, these systems require proper lighting, delicate and expensive hardware and computationally intensive algorithms. On the other hand, motion based systems use data acquired from sensors like accelerometer, gyroscope and flex sensor to identify the gestures being performed by the user. Of late, most gesture recognition systems designed for effective interaction utilize accelerometers for cheaper and accurate data collection.

Accelerometer-based hand gesture recognition systems deal with either static or dynamic gestures as mentioned in [3]. Static gestures can be uniquely characterized by identifying their start and end points, while dynamic gestures require the entire data sequence of a gesture sample to be considered. Constructing a dynamic gesture recognition system that is compatible with any user is difficult, as the manner in which the same gesture is performed varies from user-to-user. This variation arises because of the disparate speeds of the dynamic gestures signed by users. To tackle this problem, a common set of features which represent the dynamic nature of the gestures across various users should be selected.

The authors of this paper propose a gesture recognition system using a generic feature set, implemented on two public datasets using accelerometers - uWave and Sony, as shown in [4] and [5], respectively. This system has been trained and tested across various classifiers and modes, giving equal importance to both accuracy and classification time, unlike most conventional systems. This results in a computationally efficient model for the classification of dynamic gestures which is compatible with low-cost systems.

The rest of the paper is organized as follows. Section II presents the related work in the area of Gesture Recognition. Section III states about the problem statement of the paper. Sections IV & V deal with the datasets and pre-processing used in building the model. Section VI showcases the features extracted from the datasets. Section VII discusses about the different modes provided to the end-user. Section VIII describes the model used and the experiments performed. Section IX enumerates the results analyzed in the paper. Section X finally concludes the paper.

II. Related Work

Since the inception of gesture recognition systems, there has been a plethora of research in this domain using accelerometers. Specifically tri-axial accelerometers have been in the spotlight recently owing to their low-cost and low-power requirements in conjunction with their miniature sizes, making them ideal for embedding into the consumer electronic platform. Previous gesture recognition systems have also used sensors such as flex sensors and gyroscope, but they have their own shortcomings. The glove based system proposed by Zimmerman et al. in [6] utilizes flex sensors, which requires intensive calibration on all sensors. The use of the these sensors increases the cost of the system and also makes the system physically cumbersome. These shortcomings make

inertial sensors like accelerometers and gyroscope, a better alternative. In this paper, datasets employing accelerometers were preferred over gyroscopes, as processing the data from gyroscopes results in a higher computational burden.

Contemporary gesture recognition systems employ Dynamic Time Warping (DTW) algorithms for classification of gestures. For each user, Liu et al. [4] employs DTW to compute the look-up table (template) for each gesture, but it is not representative of all users in the dataset, thereby not generalizing for the user-independent paradigm. To achieve a generic look-up table for each gesture that represents multiple users, the proposed system in [7] uses the concept of idealizing a template wherein, the gestures exhibiting the least cost when internal DTW is performed is chosen as the look-up table. Furthermore, DTW is performed again for gesture classification while testing, making the model computationally very expensive to be used in a low-cost embedded platform. The accelerometer-based gesture recognition system proposed in [8] uses continuous Hidden Markov Models (HMMs), but their computational complexity is commensurate with the size of the feature vectors which increase rapidly. In addition, choosing the optimum number of states is difficult in multi-user temporal sequences, thereby increasing the complexity of estimating probability functions.

Moreover, the length of the time series acceleration values of a gesture sample is made equal and quantization is performed in the system proposed in [4]. This makes the data points either lossy or redundant. However, the paper proposed utilizes the data points as provided in the datasets without any windowing or alteration, thereby decreasing the computational costs whilst not compromising on efficiency.

In the system employed in [9], Helmi et al. have selected a set of features that has been implemented on a dataset that contains gestures performed by a single user, making them gesture-dependent and not taking into account the concept of generic features encompassing multiple users. The need for a generic set of features which capture the similarities between gestures, motivated the authors of this paper to implement a feature set that can be applied to any system.

III. Problem Statement

Previous works in haptic-based gesture recognition systems employed computationally expensive algorithms for identification of gestures with instrumented and multifarious sensors, making them reliant on specialized hardware. Moreover, most gesture recognition systems extract features that are model-dependent, and fail to provide the user a choice between accuracy and classification time. Thus a need for a flexible gesture recognition system that identifies dynamic gestures accurately with minimalistic hardware, along with a generic feature set arises.

To overcome this problem, this paper presents a machine-learning based dynamic gesture recognition system that utilizes accelerometers along with a generic set of features that can be implemented across any model with adequate gesture samples. The system provides the end-user the option to choose between accuracy and classification time, thereby giving equal importance to both.

IV. Datasets

TABLE I. Datasets Characterized by their Respective Attributes

Dataset	Users (U)	Gestures (N_G)	Samples per Gesture (S_G)	Days (N_D)	N_{GS}
uWave (D_u)	8	8	10	7	4480
Sony (D_S)	8	20	20	-	3200

For this paper, the authors have chosen two public gesture datasets, viz. uWave dataset (D_u) and Sony dataset (D_S). The datasets were selected owing to their large user campaign of 8 users, with a multitude of gesture samples for a variety of gestures. D_u encompasses an 8-gesture vocabulary with 560 gesture samples per dataset over a period of 7 days, while D_S consists of a collection of 20 gestures with 160 gesture samples each. This shows the diverse nature of the datasets. Both the datasets are characterized by U users signing N_G gestures with S_G samples per gesture, over N_D days with the total number of gesture samples being N_{GS} per dataset, as shown in Table I.

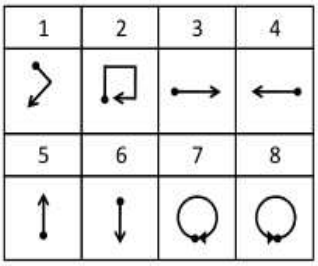


Fig. 1. uWave gestures.

D_u comprises of 3-D accelerations (g-values) that were recorded using a Wii Remote. The start of a gesture is indicated by pressing the 'A' button on the Wii Remote and the end is detected by releasing the button as mentioned in [4]. D_u consists of four gestures which have 1-D motion while the remaining gestures have 2-D motion, as depicted in Fig. 1.

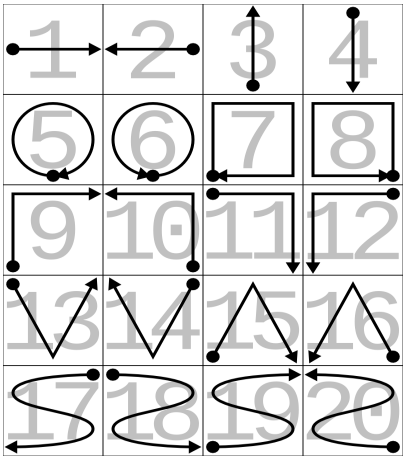


Fig. 2. Sony gestures.

D_S was recorded using a tri-axial accelerometer of a first generation Sony Smartwatch which was worn on the right wrist of the user. Each gesture instance was performed by tapping the smartwatch screen to indicate the start and end of

TABLE II. CONFUSION MATRIX FOR U_M MODE ON D_S; BROWN: GESTURE SIGNED, YELLOW: GESTURE CLASSIFIED, GREEN: CORRECT CLASSIFICATIONS, RED: INCORRECT CLASSIFICATIONS

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	2.5	0	0	97.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	97.5	0	0	0	0	0	0	2.5	0	0	0	0	0	0
8	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	95	5	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	2.5	97.5	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	97.5	2.5	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	97.5	2.5	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	97.5	2.5	0	0	0	0
16	0	0	2.5	0	2.5	0	0	0	0	0	0	0	0	0	2.5	92.5	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100

the gesture. The data recorded from the smartwatch consists of timestamps from different clock sources of an Android device, along with the acceleration (g-values) measured across each axis, as mentioned in [5]. In D_S, there are four gestures which have motions in 1-D while the remaining gestures have motions in 2-D as illustrated in Fig. 2.

V. DATASET PREPROCESSING

To generalize the datasets, we eliminate the dependency on time from the datasets, which consist of timestamps along with the raw g-values. The set of g-values per gesture sample (γ), each with a cardinality of n_γ is given by,

$$\begin{aligned} \gamma = \{g_x^i,\ g_y^i,\ g_z^i\}, \forall\ i \in [1,\ n_\gamma] \\ where,\ g_x^i,\ g_y^i,\ g_z^i\ are \\ g-values\ of\ the\ accelerometer \end{aligned} \tag{1}$$

On account of the dynamic nature and varying speeds at which different gestures are signed by the users, n_γ varies between different gesture samples. From (1), an overall dataset (G) can be defined by,

$$G = \bigcup_{i=1}^{N_{GS}} \gamma^i \tag{2}$$

Equation (2) is representative of both D_S and D_u datasets. No pre-processing other than elimination of timestamps was done to both the datasets as the g-values in γ would be altered, thereby making the datasets lossy.

VI. FEATURE EXTRACTION

A. Feature Characterization

The proposed system uses the features mentioned in Table III, which have been already utilized in previous accelerometer-based studies. From [10], the significance of Minimum, Maximum, Mean, Skew, Kurtosis and Cross correlation have been shown for Activity Recognition, which is a superset of Gesture Recognition. The inter-axial Pearson product-moment (PM) correlation coefficients as a feature has been signified in [11]. The Spectral Energy as a feature has been illustrated in [12]. The aforementioned features were iteratively eliminated in various domains until the efficiency of the model was the highest.

TABLE III. FEATURE CHARACTERIZATION IN TIME AND FREQUENCY DOMAINS

Features\Domain	Time	Frequency	
		FFT	HT
Mean	✓ (T^1)	✗	✓ (H^1)
Skew	✓ (T^2)	✗	✓ (H^2)
Kurtosis	✓ (T^3)	✗	✗
PM correlation coefficients	✓ (T^4)	✗	✗
Cross correlation	✓ (T^5)	✗	✗
Energy	✗	✓ (F^1)	✓ (H^3)
Minimum	✗	✗	✓ (H^4)
Maximum	✗	✗	✓ (H^5)

B. Domain Characterization

The set of extracted features were then applied in both time and frequency domains. The features that were extracted in the time domain are Mean, Skew, Kurtosis, PM correlation coefficients and Cross correlation. These features along each axis of the gesture samples of a dataset (G) in time domain constitute the TF vector, as given in (3).

$$TF = \bigcup_{j=1}^{5} \{T_x^j,\ T_y^j,\ T_z^j\} \tag{3}$$

The time-domain sequences were converted to frequency domain using Fast Fourier Transform (FFT) and only Energy, as mentioned in [12] was used, while the other features from FFT did not provide any significant improvements. The feature vector (FF) along each axis of the gesture samples of a dataset (G) is represented by (4).

$$FF = \{F_x^1,\ F_y^1,\ F_z^1\} \tag{4}$$

TABLE IV. CONFUSION MATRIX FOR U_I MODE ON D_S; BROWN: GESTURE SIGNED, YELLOW: GESTURE CLASSIFIED, GREEN: CORRECT CLASSIFICATIONS, RED: INCORRECT CLASSIFICATIONS

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	95	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	65	15	0	0	5	0	0	0	0	0	15	0	0	0	0	0
6	0	0	0	0	0	95	5	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	90	10	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0
9	5	0	5	0	0	0	0	0	85	5	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	10	90	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	95	5	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	5	95	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	10	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	80	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	95	0	0	5
18	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	5	90	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	70

From [13], it can be seen that the Hilbert Transform (HT) hypothesized as a feature was successful in providing a competitive recognition rate for camera-based handwriting gestures. This novel approach was used in the accelerometer-based gesture recognition model proposed by the authors.

Hilbert transform is a linear transformation used in signal processing, that accepts as input a temporal signal, and produces an analytic signal. The analytic signal consists of a real and an imaginary part as explained in [13], where the negative half of the frequency spectrum is zeroed out. The real part represents the input signal and the imaginary part (y) is representative of the Hilbert transformed signal, which is phase shifted by $\pm 90°$. For an input signal ($x(t)$), the analytic signal ($x_a(t)$) after Hilbert transform is given by (5).

$$\begin{aligned} x_a = F^{-1}(F(x)2U) = x + iy, \\ where,\ F - Fourier\ transform, \\ U - \ Unit\ step\ function, \\ y - Hilbert\ transform\ of\ x. \end{aligned} \tag{5}$$

This approach of using Hilbert Transform for feature extraction is applied across all gesture samples, and the features - Mean, Skew, Minimum, Maximum and Energy are calculated as shown in Table III. The features Mean and Skew, which have been used in TF, and Energy which is calculated in FF are also used as Hilbert features since reusing them here yields a better performance for classifying different gestures. These Hilbert transformed features along each axis on a dataset (G) make up the HF vector, which is given by Equation 6.

$$HF = \bigcup_{j=1}^{5} \{H_x^j,\ H_y^j,\ H_z^j\} \tag{6}$$

The FeatureSet (FS) for a single dataset (G) is formed from (3), (4) and (6) by appending all the feature vectors - TF, FF and HF calculated across a dataset (G), as shown in (7).

$$FS = \bigcup_{j=1}^{N_{GS}} \{TF \cup FF \cup HF\}^j \tag{7}$$

VII. END-USER MODELLING

This paper enables the end-user to select any one of three proposed modes of operation - User Dependent, Mixed User and User Independent. The User Dependent (U_D) mode is an estimator of how well the system performs when the train-test split is between the gestures of a single user. Mixed User (U_M) is representative of the complete set of gestures of all participants. The User Independent (U_I) mode employs a stratified k-fold cross validation technique which corresponds to training on a number of users and testing on the rest.

VIII. EXPERIMENT

The three modes U_D, U_M and U_I, as explained in Section VII, were first trained and tested on an Intel Core i5 CPU @ 2.20 GHz, operating on Ubuntu 16.04. The proposed system was initially implemented using seven classification algorithms that were identified from [14], [15] and [16] – Extremely Randomized Trees (Extra Trees), Random Forests, Gradient Boosting, Bagging, Decision Trees, Naive Bayes and Ridge Classifier.

Upon further analysis, the seven classifiers were found to run on a low-cost Raspberry Pi Zero, and the accuracies and time taken for a gesture sample to be classified are catalogued for the three modes across both datasets, as shown in Table V. From this set of seven classifiers, the Extra Trees (ET), Gradient Boosting (GB) and Ridge Classifier (RC) were chosen, based on the inferences from Section IX-A. Each of the three modes is evaluated using the three classifiers individually, and the efficiencies are noted and analyzed for both the datasets D_u and D_S in Sections IX-B and IX-C, respectively.

IX. RESULTS

A. Evaluation of Classifiers

The classifiers chosen in Section VIII were further analyzed for each dataset based on their computational characteristics.

TABLE V. AVERAGE ACCURACIES (ACC) FOR THE DATASETS D_u AND D_S AND TIME TAKEN (IN SECONDS) FOR CLASSIFICATION OF A SINGLE GESTURE SAMPLE ACROSS ALL MODES; GREEN: HIGHEST EFFICIENCIES IN ALL MODES, YELLOW: LEAST CLASSIFICATION TIMES TAKEN FOR A GESTURE SAMPLE IN ALL MODES

	uWave (D_u)						Sony (D_s)					
Classifier\Mode	User Dependent (U_D)		Mixed User (U_M)		User Independent (U_D)		User Dependent (U_D)		Mixed User (U_M)		User Independent (U_D)	
	Acc	Time	Acc	Time	Acc	Time	Acc	Time	Acc	Time	Acc	Time
Extra Trees	97.76	0.6287	97.85	0.6873	82.49	0.6853	95.88	0.6038	98.63	0.6538	75.1	0.669
Random Forest	97.41	0.6991	95.45	0.73	77.91	0.6995	95.25	0.6887	97.13	0.7041	70.13	0.686
Gradient Boosting	93.75	0.0072	94.38	0.0076	75.64	0.0078	90.5	0.0055	95.5	0.0057	66.41	0.0057
Bagging	92.74	0.1526	94.19	0.1527	76.64	0.1528	93.5	0.1673	93.37	0.1527	62.44	0.1529
Decision Trees	89.55	0.0015	84.11	0.0053	66.73	0.0015	84.13	0.0014	86.25	0.0051	50.9	0.0015
Naive Bayes	91.16	0.0273	71.96	0.0292	64.66	0.0273	91.38	0.0118	65.5	0.0116	54.35	0.0117
Ridge Classifier	97.5	0.0013	83.84	0.0013	74.64	0.0013	94.13	0.0013	77.75	0.0014	61.59	0.0013

It can be inferred that ET yields the highest accuracy in all modes for both datasets D_u and D_S, among all the three chosen classifiers. Apart from being more accurate than Random Forest, ET is also computationally less expensive as stated in [17]. It can also observed that the time taken for classification of a gesture sample is the least in RC. GB, whilst yielding a significantly lower classification time than ET, provides reasonable accuracies and hence was chosen.

Based on the users' preferences, a trade-off can be made between efficiency and classification time of a gesture sample among the three classifiers. To evaluate the performance of this model in all the modes across both datasets, evaluation metrics - cross-validation scores and confusion matrices were taken for the results obtained from the ET classifier, as it has the highest efficiency and an acceptable classification time.

TABLE VI. ACCURACIES OF 8 USERS FOR BOTH DATASETS D_u AND D_S IN USER DEPENDENT MODE (U_D) USING EXTRA TREES CLASSIFIER

Data\User	1	2	3	4	5	6	7	8	Avg
D_u	96.42	93.5	98.57	97.14	99.28	99.28	97.88	100	97.76
D_S	98	92	94	93	98	100	95	97	95.88

B. uWave Dataset Analysis

In the U_D mode on D_u, each user's data was divided into a randomized 75%-25% train-test split across all 8 gestures. The average recognition rate achieved over all 8 users was found to be 97.76%, using ET. The individual users' efficiencies for the same can be observed in Table VI.

The classification model implemented using U_M mode on D_u with a randomized 75%-25% train-test split, provides an accuracy of 97.85%, and its confusion matrix can be showcased in Table VII.

TABLE VII. CONFUSION MATRIX FOR U_M MODE ON D_u; BROWN: GESTURE SIGNED, YELLOW: GESTURE CLASSIFIED, GREEN: CORRECT CLASSIFICATIONS, RED: INCORRECT CLASSIFICATIONS

	1	2	3	4	5	6	7	8
1	99.29	0.71	0	0	0	0	0	0
2	0.71	99.29	0	0	0	0	0	0
3	0.71	0	97.86	1.43	0	0	0	0
4	0	0	2.14	97.86	0	0	0	0
5	0	0	0	0	97.14	2.86	0	0
6	0	0	0	0	0	100	0	0
7	1.43	1.43	0	0	0	0	95	3.57
8	0	0.71	0	0	0	0	2.86	96.43

It was seen that on applying U_I mode on D_u, an average efficiency (average Leave-One-User-Out Cross-Validation score) of 82.49% was observed, while [4] has achieved 75.4%. The confusion matrix for the last user as test, trained upon the first seven users, which yields an accuracy of 92.14% has been illustrated in Table VIII.

TABLE VIII. CONFUSION MATRIX FOR U_I MODE ON D_u; BROWN: GESTURE SIGNED, YELLOW: GESTURE CLASSIFIED, GREEN: CORRECT CLASSIFICATIONS, RED: INCORRECT CLASSIFICATIONS

	1	2	3	4	5	6	7	8
1	95.71	4.29	0	0	0	0	0	0
2	0	100	0	0	0	0	0	0
3	0	0	82.86	17.14	0	0	0	0
4	0	0	0	100	0	0	0	0
5	0	0	0	0	100	0	0	0
6	0	4.29	0	0	0	95.71	0	0
7	0	1.43	0	0	0	0	70	28.57
8	0	1.43	0	0	0	0	5.71	92.86

Table IX shows that RC has the least classification time along with an accuracy of 97.5%, which is marginally lesser than ET, thereby effectively capturing the similarities between gesture samples signed by the same user, i.e, in U_D mode. GB provided accuracies and computational times which are intermediary between ET and RC across all three modes, thereby giving the user a choice to reduce classification time by a significant margin, while having a reasonable accuracy.

TABLE IX. ACCURACIES (ACC) AND TIME TAKEN (IN SECONDS) FOR CLASSIFICATION OF A SINGLE GESTURE SAMPLE FOR D_u USING THE 3 SELECTED CLASSIFIERS ACROSS ALL MODES

Classifier\Mode	U_D		U_M		U_I	
	Acc	Time	Acc	Time	Acc	Time
ET	97.76	0.6287	97.85	0.6873	82.49	0.6853
GB	93.75	0.0072	94.38	0.0076	75.64	0.0078
RC	97.5	0.0013	83.84	0.0013	74.64	0.0013

C. Sony Dataset Analysis

For all eight participants, the average accuracy for D_S in U_D mode was observed to be 95.88% using ET, where each user's data was divided into a 75%-25% train-test split in random, as done in D_u. The efficiencies across all individual users can be observed in Table VI.

TABLE X. ACCURACIES (ACC) AND TIME TAKEN (IN SECONDS) FOR CLASSIFICATION OF A SINGLE GESTURE SAMPLE FOR D_S USING THE 3 SELECTED CLASSIFIERS ACROSS ALL MODES

Classifier\Mode	U_D		U_M		U_I	
	Acc	Time	Acc	Time	Acc	Time
ET	95.88	0.6038	98.63	0.6538	75.1	0.669
GB	90.5	0.0055	95.5	0.0057	66.41	0.0057
RC	94.13	0.0013	77.75	0.0014	61.59	0.0013

In the U_M mode, the proposed model with a 75%-25% train-test split at random on D_S, yields an efficiency of 98.625%. Table II shows the confusion matrix for the U_M mode across all 20 gestures.

TABLE XI. BEST ACCURACIES (ACC) AND LEAST TIMES TAKEN (IN SECONDS) FOR CLASSIFICATION OF A SINGLE GESTURE SAMPLE ACROSS ALL MODES FOR BOTH DATASETS D_u AND D_S

Dataset\Mode	U_D		U_M		U_I	
	Acc (ET)	Time (RC)	Acc (ET)	Time (RC)	Acc (ET)	Time (RC)
D_u	97.76	0.0013	97.85	0.0013	82.49	0.0013
D_S	95.88	0.0013	98.63	0.0014	75.1	0.0013

An average 8-fold cross validation score (average recognition rate across all users) of 75.093% was observed in the U_I mode of D_S. The confusion matrix trained upon the first seven users with the last user as test, which yields an accuracy of 91.75%, is shown in Table IV.

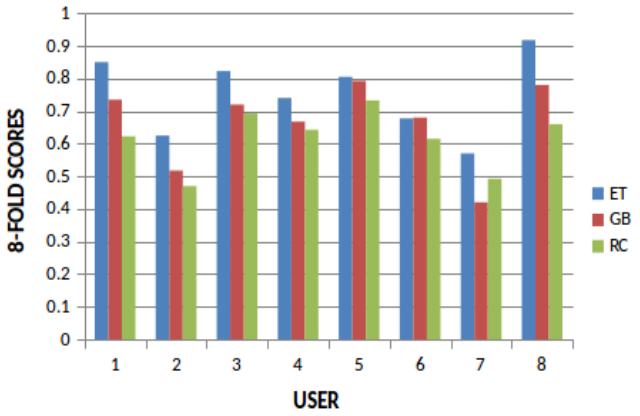


Fig. 3. 8-fold Cross-validation Scores for all Users in D_S.

Fig. 3 shows the behavior of all users as test (Leave-One-User-Out Cross-Validation scores) in the U_I mode on D_S. The large number of misclassifications in the second and seventh user is indicative of both users having not performed the gestures in a manner similar to the rest, thereby reducing the overall efficiency across all classifiers.

Owing to the generic FeatureSet (FS) implemented, the results observed in Table X for D_S are analogous to the results observed in D_u for all three chosen classifiers across all the three modes.

X. CONCLUSION

A Gesture Recognition system with the capability to operate in any of the three modes – User Dependent, Mixed User and User Independent, with a set of generic features was designed, validated and tested in this paper. The proposed system was tested on two public accelerometer-based gesture datasets – uWave and Sony. Table XI showcases the best classifier for each category - Efficiency and Classification Time for a gesture sample, across all three modes of both the datasets. As can be seen in Table XI, Extremely Randomized Trees was observed to perform the best in terms of accuracy, while Ridge Classifier provided the least classification time which was around 500 times faster than ET for a gesture sample, irrespective of the mode or dataset.

The end-users are given the flexibility to choose any combination of modes and classifiers according to their requirements. This system was implemented on a Raspberry Pi Zero priced at 5 USD making it a low-cost alternative.

ACKNOWLEDMENT

The authors would like to thank Solarillion Foundation for its support and funding of the research work carried out.

REFERENCES

[1] Q. Chen, N. D. Georganas and E. M. Petriu, "Real-time Vision-based Hand Gesture Recognition Using Haar-like Features," 2007 IEEE Instrumentation & Measurement Technology Conference IMTC 2007, Warsaw, 2007, pp. 1-6.

[2] K. K. Biswas and S. K. Basu, "Gesture recognition using Microsoft Kinect," The 5th International Conference on Automation, Robotics and Applications, Wellington, 2011, pp. 100-103.

[3] S. Mitra and T. Acharya, "Gesture Recognition: A Survey," in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 37, no. 3, pp. 311-324, May 2007.

[4] J. Liu, Z. Wang, L. Zhong, J. Wickramasuriya and V. Vasudevan, "uWave: Accelerometer-based personalized gesture recognition and its applications," 2009 IEEE International Conference on Pervasive Computing and Communications, Galveston, TX, 2009, pp. 1-9.

[5] SmartWatch Gestures Dataset, Technologies of Vision, Fondazione Bruno Kessler, [Online]
Available: https://tev.fbk.eu/technologies/smartwatch-gestures-dataset

[6] T.G. Zimmerman, J. Lanier, C. Blanchard, S. Bryson, and Y. Harvill, "Hand Gesture Interface Device," Proc SIGCHI/GI Conf. Human Factors in Computing Systems and Graphics Interface, pp 189-192, 1986.

[7] Shah Muhammed Abid Hussain and A. B. M. Harun-ur Rashid, "User independent hand gesture recognition by accelerated DTW," 2012 International Conference on Informatics, Electronics & Vision (ICIEV), Dhaka, 2012, pp. 1033-1037.

[8] T. Pylvninen, "Accelerometer Based Gesture Recognition Using Continuous HMMs.," in IbPRIA (1), 2005, vol. 3522, pp. 639646. 492

[9] N. Helmi and M. Helmi, "Applying a neuro-fuzzy classifier for gesture-based control using a single wrist-mounted accelerometer," 2009 IEEE International Symposium on Computational Intelligence in Robotics and Automation - (CIRA), Daejeon, 2009, pp. 216-221.

[10] K. Altun, B. Barshan, and O. Tunel, "Comparative study on classifying human activities with miniature inertial and magnetic sensors.," Pattern Recognition, vol. 43, no. 10, pp. 36053620, 2010.

[11] L. Jing, Y. Zhou, Z. Cheng, and J. Wang, "A Recognition Method for One-Stroke Finger Gestures Using a MEMS 3D Accelerometer.," IEICE Transactions, vol. 94D, no. 5, pp. 10621072, 2011.

[12] J. Wu, G. Pan, D. Zhang, G. Qi, and S. Li, "Gesture Recognition with a 3-D Accelerometer.," in UIC, 2009, vol. 5585, pp. 2538.

[13] H. Ishida, T. Takahashi, I. Ide, and H. Murase, "A Hilbert warping method for handwriting gesture recognition.," Pattern Recognition, vol. 43, no. 8, pp. 27992806, 2010.

[14] L. Tencer, M. Reznkov, and M. Cheriet, "Evaluation of techniques for signature classification from accelerometer and gyroscope data.," in ICDAR, 2015, pp. 10661070.

[15] W. Aswolinskiy, R.F. Reinhart, and J.J. Steil. "Impact of regularization on the model space for time series classification." In Machine Learning Reports, pages 4956, 2015.

[16] Y. Yang and Y. Yu, "A hand gestures recognition approach combined attribute bagging with symmetrical uncertainty," 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery, Sichuan, 2012, pp. 2551-2554.

[17] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," Machine Learning, vol. 63, no. 1, pp. 342, Jul. 2006.

Label Frequency Transformation for Multi-Label Multi-Class Text Classification

Raghavan A K
Global AI Accelerator, Ericsson / Chennai
k.raghavan.a@ericsson.com

Venkatesh Umaashankar
Ericsson Research / Chennai
venkatesh.u@ericsson.com

Gautham Krishna Gudur
Global AI Accelerator, Ericsson / Chennai
gautham.krishna.gudur@ericsson.com

Abstract

In this paper, we (***Team Raghavan***) describe the system of our submission for GermEval 2019 Task 1 - *Subtask (a)* and *Subtask (b)*, which are multi-label multi-class classification tasks. The goal is to classify short texts describing German books into one or multiple classes, 8 generic categories for *Subtask (a)* and 343 specific categories for *Subtask (b)*. Our system comprises of three stages. **(a)** Transform multi-label multi-class problem into single-label multi-class problem. Build a category model. **(b)** Build a class count model to predict the number of classes a given input belongs to. **(c)** Transform single-label problem into multi-label problem back again by selecting the top-k predictions from the category model, with the optimal k value predicted from the class count model. Our approach utilizes a *Support Vector Classification* model on the extracted vectorized *tf-idf* features by leveraging the *Byte-Pair Token Encoding (BPE)*, and reaches f1-micro scores of **0.857** in the test evaluation phase and **0.878** in post evaluation phase for *Subtask (a)*, while **0.395** in post evaluation phase for *Subtask (b)* of the competition. We have provided our solution code in the following link: https://github.com/oneraghavan/germeval-2019.

1 Introduction

Multi-label Multi-class Hierarchical Classification tasks typically encompass multiple possible (one or more) labels for each instance (not mutually exclusive) across multiple possible classes (two or more) with many levels of hierarchies, and are widely used in domains like text classification (Rousu et al., 2006), image classification (Hsu et al., 2009) and bioinformatics (Barutcuoglu et al., 2006), (Feng et al., 2017). In this paper, we present our submission approach for *Subtask (a)* and *Subtask (b)* in *GermEval 2019 Task 1* (Remus et al., 2019). Here, we convert our task into two sub-problems: first, to predict the category; second, to predict the class frequency corresponding to the multi-label setting. Our approach can be broadly split into three stages.

- Transform the multi-label multi-class problem into a single-label multi-class problem, and build a category model.
- Build a class count predictor model to predict the number of classes that a given input could be categorized.
- Transform single-label problem back into a multi-label problem by selecting the top k predictions from the category model, with the optimal k value predicted from the class count model.

Conventionally, Natural Language Processing (NLP), particularly text classification tasks have been modeled using variants of Support Vector Machines (Joachims, 1998) and Naive Bayes Classifiers (McCallum et al., 1998). With the widespread adoption of deep learning models, there has been considerable increase in efficiencies for such tasks, however they are still considered black box models, and it is extremely hard to interpret them, in contrast to conventional Machine Learning algorithms. Moreover, the time and computational resources required for training deep neural networks are extremely higher than conventional Machine Learning models.

Hence, in our approach, we utilize the traditional ***Support Vector Classification*** modeled using *tf-idf* (term frequency - inverse document frequency) feature vectors combined with class count predictor, and we leverage the *Byte-Pair Encoding (BPE)* compression tokenization mechanism. Experimental results from exploiting

such simple model fusion approaches show that in the post-evaluation phase, f1-micro scores of 0.878 on *Subtask (a)* and 0.395 on textitSubtask (b) could be achieved. The overall modeling pipeline/architecture of our approach can be found in Figure 1.

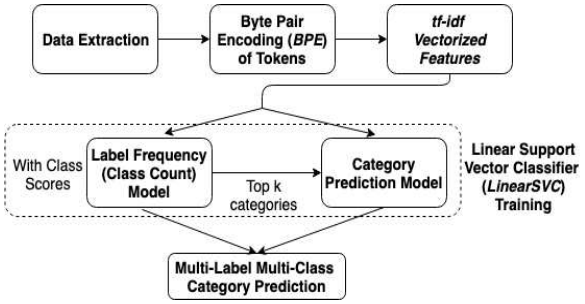


Figure 1: Modeling Pipeline/Architecture for both *Subtask (a)* and *Subtask (b)*

The rest of the paper is organized as follows. The characteristics of the dataset are described in Section 2. In Section 3, the data extraction and pre-processing techniques to obtain our feature vectors are discussed. Section 4 presents our model architecture along with training and aggregation phases. This is followed by systematic evaluation of our model's results and submission in Sections 5 and 6, and we finally conclude the paper.

2 Data Description

The *GermEval 2019 Task 1* dataset (Remus et al., 2019) consists of German books crawled from `randomhouse.de`, with the following attributes – title, description, author name, ISBN and book release date. Apart from date, most of the other features available are in the form of text, where title and description are very short texts. A total of 343 categories are present across three levels of hierarchy with 8, 93 and 242 categories in each level, and multiple labels can be assigned to each book.

The corpus has a very imbalanced label distribution. Figure 2 shows the top-level label distributions, while Figure 3 shows the top 30 label distributions, and it can be vividly inferred that the label categories are highly skewed.

3 Data Extraction and Prepossessing

The corpus was presented as an XML file, with XML tags for each feature. The XML files were parsed using the Python library - *BeautifulSoup*,

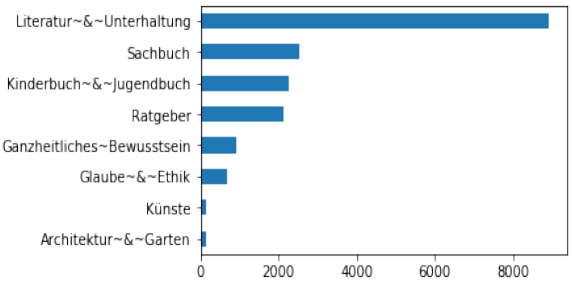


Figure 2: Top-level Label Distribution

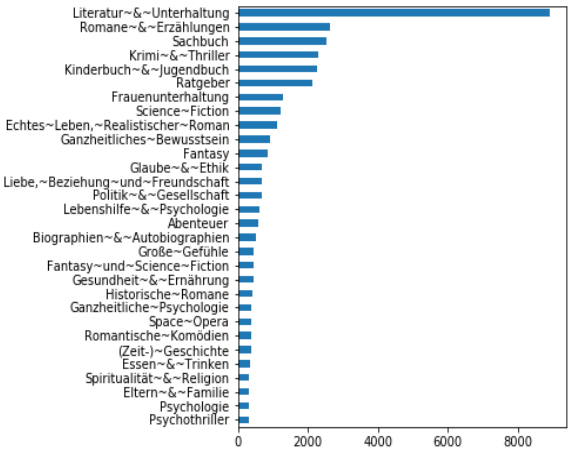


Figure 3: Label Distribution for top 30 classes with hierarchies

and the columns given in the corpus such as title, description, list of authors, date of publication, ISBN were extracted from the XML format into a CSV format. The corpus has hierarchy information in the *hierarchy.txt* file, which was used to validate and prepare the corpus for modelling.

From the extracted data, we initially filter out the stop words and numbers from title and description for every book using the *Natural Language Toolkit (NLTK)* library available in Python. We then utilize ***Byte-Pair Encoding (BPE)*** (Heinzerling and Strube, 2018) based tokenization to create tokens from the titles and descriptions for various texts. BPE tokenization, in this context, identifies the most common consecutive bytes of German words and replaces with a byte that does not occur within the data. For instance, *'Ein Blick hinter die'* might be converted into *'__ein __blick __hinter __die'* after BPE with a vocabulary size of 25,000. We utilize the BPEmb library for the same, which utilizes pre-trained byte-pair embeddings, and require no tokenization, also being much shorter.

The German words are split into multiple smaller sub-words, which would inherently en-

able better representations for each book's Title and Description, and provide more context for the feature vectors which are learned. Subwords in BPEmb also enable effective guessing of unknown/out-of-vocabulary words' meaning based on context. For instance, the suffix -shire in Melfordshire indicates a location.

After tokenizing each title and description of books using *BPE*, we create ***term frequency - inverse document frequency (tf-idf)*** vectors from the tokens. *tf-idf* is a widely used statistical measure in domains of NLP, text mining and information retrieval, which signifies the importance of a word to a document in the whole corpus (Ramos, 2003).

We experimented with various contiguous sequences of n-grams – unigrams, bigrams and trigrams for creating *tf-idf* vectors, and observed that bigram models yielded better results than the rest. Each book can have multiple authors, so we treat the authors' data as a binomial attribute (creating a *Label Binarizer* which returns 1 if the book was authored by that author, else 0) across all authors. We extract the ***year*** from the publication date, and utilized it as a categorical feature. We also extracted the ***Group ID*** and ***Publisher ID*** from the 13-digit ISBN and represented them as categorical features again.

We created two sets of target variables from the extracted data – one with the categories as target labels, and other with the count of categories. Hence, we also created and pipelined two different models – *class count model* and *category score predictor model*. For instance, for a given book instance, if there are k categories, then that training sample has been duplicated k times with each sample having one category. Adding all features, a sparse feature matrix was created. The final feature matrix is of size (17783 x 594931) for both class count model and category score predictor model.

Labels	Counts
Label 1	15549
Label 2	1004
Label 3	70
Label 4	4

Table 1: Class Count Frequencies

The class count model is a classification problem with labels 1, 2, 3 as class frequencies. While there are book instances up-to 4 top-level observed class frequencies, there are only 4 training samples for category 4 which is extremely less (negligible), hence we consider only 3 categories. The class count distribution can observed in Table 1. The categories are predicted based on an 8-class classification approach for top-level categories (*Subtask (a)*), and a 343-class classification mechanism for multi-level (*Subtask (b)*) categories.

4 Model Pipeline

Given the corpus has a heavy class imbalance across all levels, we choose a Support Vector Machine (SVM) based model for the task. SVMs are known to exhibit robustness and perform effectively under class imbalance (Tang et al., 2009).

The ***Linear Support Vector Classifier (LinearSVC)*** in a multi-class classification problem utilizes a one vs rest scheme, wherein the objective is to return the best-fit hyperplane that categorizes the data in an n-dimensional space. Identifying the right hyperplane is primarily dependent on the *margin* (maximized distance between hyperplane and nearest data point), and the loss function governing the margin. We utilize the *squared-hinge loss* for the same, as it is widely used for maximum-margin classification in SVMs that penalizes the violated margins more strongly (quadratically). The *squared-hinge loss*, l for Linear SVM is given by,

$$l(y) = max(0, 1 - t \cdot y)^2$$

where y is the classifier score $y = w.x + b$, w, b are the parameters of the hyperplane, x is the input variable(s) and the intended output t = ± 1.

We created two Linear SVC models, one for predicting the count of classes a book could belong to, and the other for category score predictor. *Linear SVC* was chosen as it uses the *liblinear* framework and scales well with the increase in the number of features (Fan et al., 2008). Moreover, it is computationally faster than most other Linear SVM implementations, and utilizes many other advanced optimization techniques. We utilize the *scikit-learn* Python framework (Pedregosa et al., 2011) to train and test the *LinearSVC* model, which uses liblinear as its default implementation. Also, this implementation offers flexibility in the choice of penalties and loss function parameters, and would inherently scale well to larger samples of data.

In our case, use of *tf-idf* vectors to represent words created a large number of feature vectors. The *tf-idf* vector representation were very sparse owing to the short text representations for each book. Compressed sparse representation (*CSR*) of the feature matrix was further utilized reduce the size of the training set.

4.1 Model Parameters

After extensive parametric optimization for both models – count classifier and category classification using grid search, we arrived the following set of final parameters to yield best efficiencies. The optimal parameters for the LinearSVC model are elucidated in Table 2.

Parameter	Optimal Value
C	1.0
Tolerance for stopping	0.0001
Loss	Squared Hinge Loss
Penalty	L2
Optimization algorithm	Dual
Max Iterations	3000

Table 2: Optimal Parameters for *LinearSVC*

For the top-level classifier, the following class weights are also used to handle class imbalance. We utilized grid search to fine tune the class weights again, which can be observed in Table 3.

Category	Class Weight
Kinderbuch & Jugendbuch	1.8
Ratgeber	3
Sachbuch	2
Glaube & Ethik	2
Künste	6
Architektur & Garten	6
Literatur & Unterhaltung	1
Ganzheitliches Bewusstsein	1

Table 3: Class Weights for Top-level Categories to handle Class Imbalance in *LinearSVC*

The class weights might be conventionally perceived that the categories with lower cardinality might have higher class weights. However, the model prioritizes and takes into account the confusion between various classes than the imbalance in classes alone. For instance, we could observe that a lot of Sachbuch and Glaube get classified as Literatur & Unterhaltung, hence giving more weightage to the latter aids in higher efficiencies.

4.2 Model Fusion

We pipeline the class score predictor and class count predictor models together for effective classification of categories. Since the input features for both models remain the same, we train our model with categories as target variables for the class score predictor model, and with class counts as target variables for class count predictor model.

In the class count model, we also add the scores of output class predictions to the input feature space. This is motivated by the inherent fact that there is a high correlation between the number of categories a book belongs to, and its corresponding class with the highest score. While predicting a book's category, we first get the class scores for all categories from the class score predictor model, and then append those predictions with input data to the class count classifier model. Once the class count predicts number of possible categories k, we find the top k category predictions from the class category predictor (likelihood) model.

In this way, the class imbalanced multi-label problem is split into two simple prediction problems. Also, by splitting the problem into smaller chunks, we are able to train multiple models in parallel, thus reducing the total training time of the system. With the above setup, retraining the entire set of models takes just under 2 minutes.

5 Evaluation

For building an end-to-end classifier system, we build a Classifier Class extending the *scikit-learn* Base estimator API, with the respective fit and predict functions. The constructor parameters passed, are the hyperparameters for the class count predictor and class likelihood predictor models. Inside the constructor, we create two LinearSVC models. In the fit (training) phase, both the predictors are trained with their respective target variables. We utilize a ***K-Fold Cross-validation*** strategy ($K = 4$) and get the class scores to be used in training for class count predictor. Once we have the class scores, we retrain the class predictor with the whole training data again. The class count predictor is then utilized for training along with the class scores appended.

Similarly, in the predict phase, the class prediction model is first used to obtain class prediction scores, and further utilized by class count predictor to get the class count distribution. We select take k highest category predictions from the class

scores.

The micro-f1 scores for *Subtask (a)* and *Subtask (b)* with CV=4 folds can be found in Table 4.

CV Fold	*Subtask (a)*	*Subtask (b)*
Fold 1	0.833	0.384
Fold 2	0.943	0.471
Fold 3	0.950	0.484
Fold 4	0.900	0.397

Table 4: k-fold Cross-Validation (k=4) on *Subtask a* and *Subtask b*

The experimental setup (HW/SW configurations) utilized for our solution are as follows:

(1) Intel® Xeon® Processor E5-2650 v4 30M Cache, 2.20 GHz, 12 Cores, 24 Threads **(2)** 250 GB RAM **(3)** CentOS 7.

6 Submission and Results

With the above setup, the model was able to achieve the results showcased in Table 5 in the test phase.

Phase	*Subtask (a)*	*Subtask (b)*
Validation Phase	0.851	0.4098
Test Phase	0.857	-
Post Evaluation Phase	0.878	0.3947

Table 5: Evaluation Metrics (f1-micro scores) for Test Data on *Subtask a* and *Subtask b*

Team Raghavan achieves rank 4 in *Subtask (a)* during the test phase (f1-score of $\sim$0.86). However, in the post-evaluation phase, we achieve an f1-score of 0.878, which secures us the first position in *Subtask (a)*. The additional 0.02 gain in micro-f1 score during post-evaluation phase finally was achieved by adding ISBN based features – *Group ID* and *Publisher ID*.

7 Conclusion

In this paper, we have successfully demonstrated that traditional approaches like Linear Support Vector Machine Classifier, with a class count predictor model can effectively model Multi-label Multi-class Hierarchical Text Classification of German blurbs – *GermEval Task 1*. The model designed by us was aimed for top-level categories, i.e., *Subtask (a)*, which implements flat classification and doesn't make use of much hierarchical dependencies. That is one primary reason for *Subtask (b)* achieving relatively less efficiencies.

The authors would like to emphasize that conventional machine learning solutions would help in better interpretability, and when pipelined/fused with the right set of techniques, can effectively save a lot computational resources and time.

8 Credits

The authors would like to thank their colleagues at Ericsson Research and Global AI Accelerator (GAIA) at Ericsson during this competition. Also, we would like to thank the *scikit-learn* developers for actively developing and maintaining this awesome tool. Finally, we thank the *GermEval* competition organizers for fostering a friendly and collaborative environment around this dataset and for answering our questions throughout the competition.

References

Zafer Barutcuoglu, Robert E Schapire, and Olga G Troyanskaya. 2006. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.

Shou Feng, Ping Fu, and Wenbin Zheng. 2017. A hierarchical multi-label classification algorithm for gene function prediction. *Algorithms*, 10(4):138.

Benjamin Heinzerling and Michael Strube. 2018. BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA).

Daniel J Hsu, Sham M Kakade, John Langford, and Tong Zhang. 2009. Multi-label prediction via compressed sensing. In *Advances in neural information processing systems*, pages 772–780.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Andrew McCallum, Kamal Nigam, et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, pages 41–48. Citeseer.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Juan Ramos. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the First Instructional Conference on Machine Learning*, volume 242, pages 133–142.

Steffen Remus, Rami Aly, and Chris Biemann. 2019. Germeval-2019 task 1: Shared task on hierarchical classification of blurbs. In *Proceedings of the GermEval 2019 Workshop. Erlangen, Germany*.

Juho Rousu, Craig Saunders, Sandor Szedmak, and John Shawe-Taylor. 2006. Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, 7(Jul):1601–1626.

Y. Tang, Y. Zhang, N. V. Chawla, and S. Krasser. 2009. Svms modeling for highly imbalanced classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(1):281–288.

Heterogeneous Zero-Shot Federated Learning with New Classes for On-Device Audio Classification

Gautham Krishna Gudur, Satheesh Kumar Perepu
Ericsson

Deep learning for audio classification is a broad research area with practical applications like Keyword Spotting (KWS), urban sound identification, etc. With the recent compute capabilities vested in resource-constrained devices, there is a natural research focus on audio classification with deep learning on-device. Particularly, on-device Federated learning (FL) is an effective way of extracting insights from different user devices while preserving the privacy of user data [2]. However, new classes with completely unseen data distributions can stream across any device in an FL setting, whose data cannot be accessed by the global server or other users. Moreover, the new class information of one user is not known among other users as well, hence the new classes could be similar or different between users. In addition, there are multiple statistical heterogeneities like label and model heterogeneities across various communication rounds/FL iterations. To this end, we propose a unified zero-shot FL framework to handle these aforementioned challenges in scenarios when new class labels are reported across users with statistical heterogeneities [1].

In order to identify new classes across different users in FL settings, we construct anonymized data without transferring local sensitive data, and identify new classes on this anonymized data. We motivate our framework from the creation of zero-shot *Anonymized Data Impressions* as proposed in [3]. The anonymized feature set $\bar{\mathbf{x}}$ (which has similar properties to original input data) can be synthesized in two steps:

(a) Sample Softmax Values from the Dirichlet distribution [4]. We control the distribution by the *Class Similarity Matrix* which contains important information on how similar the classes are to each other. If the classes are similar, we find similar weights between connections of penultimate layer to the nodes of the classes [3]. We then sample the softmax values from the Dirichlet distribution (with the corresponding concentration parameter which controls its spread).

(b) Creating Anonymized Data Impressions: Once we obtain the softmax values, we compute the synthesized data features (*Data Impressions,* $\bar{\mathbf{x}}$) by minimizing the cross-entropy loss on the model created from randomly initialized input data ($\mathbf{x}$) and the generated sampled softmax values. In this way, the data impressions are created anonymously for each new class without the visibility of original input data.

There are three steps in our proposed FL framework: *Build*, *Local Update* and *Global Update*. The softmax values are sampled with the class similarity matrix in the local devices (local update step), while the anonymized data impressions for new classes are created followed by unsupervised clustering using k-medoids (in the global update step). We also perform parameterized updates [5, 6] to handle statistical heterogeneities. The statistical heterogeneities typically are disparities in models and label distributions with new classes across and within various user devices and federated learning iterations.

We simulate our experiments using *Raspberry Pi 2* with two publicly available datasets on keyword spotting and urban sound classification. We simulate two scenarios for testing our proposed framework – 1) new classes only (homogeneous) with limited users and FL iterations, 2) new classes with statistical heterogeneities in both labels and models with more users and FL iterations, which exhibits near-real-time statistical heterogeneities. The results show effective increase in the local and global update accuracies for both scenarios. Unsupervised clustering with k-medoids on the resultant data impressions for new classes is performed and visualized using PCA, and these new classes are mapped to the respective end-user devices. The new labels are finally added to the overall label set while the corresponding averaged data impressions are added to the public dataset.

References

[1] Gautham Krishna Gudur et al. (2021). Zero-Shot Federated Learning with New Classes for Audio Classification. In *Proc. Interspeech 2021.*

[2] H Brendan McMahan et al. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS 2017).*

[3] Gaurav Kumar Nayak et al. (2019). Zero-Shot Knowledge Distillation in Deep Networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019).*

[4] Thomas Minka (2000). Estimating a Dirichlet distribution.

[5] Gautham Krishna Gudur et al. (2020). Resource-Constrained Federated Learning with Heterogeneous Labels and Models. In *arXiv preprint arXiv:2011.03206.*

[6] Gautham Krishna Gudur et al. (2020) Resource-Constrained Federated Learning with Heterogeneous Labels and Models for Human Activity Recognition. In *Deep Learning for Human Activity Recognition: Second International Workshop, Held in Conjunction with IJCAI-PRICAI 2020.*

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

Gautham Krishna Gudur[†], Prahalathan Sundaramoorthy[‡], Venkatesh Umaashankar[†]
[†]Ericsson, India, [‡]University of Southern California, USA

The ubiquitous proliferation of low-cost mobile and wearable devices has spawned growing research in extracting contextual information from sensor data, particularly for Human Activity Recognition (HAR) owing to its applications in fall detection, fitness tracking, health monitoring, etc. Contemporary Deep Learning models are engineered to alleviate the difficulties posed by conventional Machine Learning algorithms which require extensive domain knowledge to obtain heuristic hand-crafted features that have questionable generalizability. The pervasive deployment of such models necessitates us to primarily perform inference, and also progress towards minimalistic training on the edge with continuous incoming stream/flow of data from different users. In an on-device *Incremental Learning* [1] setting, to facilitate continuous updation of incoming streams of unseen test user data in real-world across various users (*User Adaptability*), a small portion of data is included to update the weights of a pre-trained stocked deep learning model, thereby eliminating the need to retrain the model from scratch.

However, acquiring labeled data in real-time has always been a major challenge, particularly in HAR wherein, conventional systems assume that the incoming activities performed by users are already labeled, thereby necessitating ground truthing techniques like *Active Learning* (AL). In this study, we propose a unified *Deep Bayesian Active Learning* framework for HAR which supports high-dimensional data, by combining recent advancements in Bayesian deep learning (Bayesian Neural Networks) with AL [2]. This solves the need for labeled data by querying only a handful of data points from the oracle, and also mitigates the problem of representing uncertainty in deep learning without sacrificing either computational complexity or test accuracy. By utilizing the stochastic regularization technique - *Dropout* which is considered as an approximate Bayesian inference in deep Gaussian processes by Yarin et al. [3], we sample from the approximate posterior using multiple stochastic forward passes to arrive at the most uncertain data points from incoming pool/test data. We experiment with multiple AL acquisition functions like BALD (Bayesian Active Learning by Disagreement), Max Entropy, Variation Ratios & Random Sampling across 3 different publicly available mobile- and wearable- based HAR & Fall Detection datasets [4][5]. Preliminary results show considerable increase in accuracies and f1-scores on a Leave-One-User-Out (LOOCV) Incremental Learning setting, with just $\sim$40% of data points on the *HARNet* architecture [1].

Data Programming, being an automated data-labeling paradigm, enables us to provide certain initial heuristics called *Labeling Functions* (LFs) to automate the process of ground truthing [6]. The users can programmatically create labels using weak supervision strategies or crowd-sourced labels, which naturally tend to be noisy and conflicting. The entire labeling process using LFs is then represented as a Generative model, and further tuned using a noise-aware Discriminative model. In our future works, we aim to leverage such data programming techniques to establish ground truths for unknown incoming test data by finding minimalistic patterns that govern LFs and also extracting dependencies between different classes, thereby facilitating label generation on-the-fly. This effectively reduces any oracle involvement during real-time and enables non-experts to easily automate ground-truthing. Data Programming can also be used to label training data from scratch when scarce or no labeled data is available.

References

[1] Prahalathan Sundaramoorthy et al. (2018) HARNet: Towards On-Device Incremental Learning using Deep Ensembles on Constrained Devices. In *Proceedings of the 2nd International Workshop on Embedded and Mobile Deep Learning (EMDL'18)*.

[2] Yarin Gal et al. (2017). Deep Bayesian Active Learning with Image Data. In *Proceedings of the 34th International Conference on Machine Learning (ICML'17)*.

[3] Yarin Gal et al. (2016). Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML'16)*.

[4] Allan Stisen et al. (2015). Smart Devices are Different: Assessing and Mitigating Mobile Sensing Heterogeneities for Activity Recognition. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys '15)*.

[5] Taylor R. Mauldin et al. (2018). SmartFall: A Smartwatch-Based Fall Detection System Using Deep Learning. In *Sensors 2018*.

[6] Alexander J. Ratner et al. (2016). Data Programming: Creating Large Training Sets, Quickly. In *Advances in Neural Information Processing Systems (NIPS 2016)*.

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 87 (2016) 270 – 274

Procedia Computer Science

4th International Conference on Recent Trends in Computer Science & Engineering

Analysis of Routing Protocol for Low-Power and Lossy Networks in IoT Real Time Applications

G. Gautham Krishna[a], G. Krishna[a], Dr. N. Bhalaji[a]

[a]*Department of Information Technology, SSN College of Engineering, Kalavakkam, Chennai 603 110*

Abstract

The wide-scaled sensing by Wireless Sensor Networks (WSN) has impacted several areas in the modern generation. It has offered the ability to measure, observe and understand the various physical factors from our environment. The rapid increase of WSN devices in an actuating-communicating network has led to the evolution of Internet of Things (IoT), where information is shared seamlessly across platforms by blending the sensors and actuators with our environment. These low cost WSN devices provide automation in medical and environmental monitoring. Evaluating the performance of these sensors using RPL enhances their use in real world applications. The realization of these RPL performances from different nodes focuses our study to utilize WSNs in our day-to-day applications. The effective sensor nodes (motes) for the appropriate environmental scenarios are analyzed, and we propose a collective view of the metrics for the same, for enhanced throughput in the given field of usage.

Peer-review under responsibility of the Organizing Committee of ICRTCSE 2016

Keywords: RPL; Wireless Sensor Network (WSN); Internet of Things (IoT); Cooja simulator; Contiki;

1. Introduction

Routing Protocol for Low Power and Lossy Networks (RPL) is an IPv6 routing protocol [1], which specifically for wireless networks is optimized, and designed by IETF over low power and lossy network (ROLL) [2] as a proposed standard. The motes (sensor nodes) in a WSN are capable of processing, gathering sensory information and communicating with other connected nodes in the network. For effective implementation of RPL in real time applications, the analysis of various performance metrics of the respective motes is essential.

In this paper, we have considered a Contiki test-bed to investigate the performance of IPv6 RPL. The performance metrics for two motes, Zolertia Z1 mote and WiSMote are analyzed for various environment based implementations of RPL. Zolertia Z1 mote is a low-power WSN module that serves as a general purpose development platform for WSN developers to test and deploy applications with the best trade-off between time of development and hardware flexibility [3]. WiSMote is a wireless sensor and an actuator module well adapted to WSN applications which has low consumption [4]. By simulation in Cooja (Contiki WSN simulator), we have achieved a pattern of performance metrics to analyze and differentiate the following environments, Smart building and Agriculture.

The rest of the paper is organized as follows. Section 2 deals with a Literature Review. Sections 3 and 4 showcase the Performance Analysis of metrics with the two motes, Zolertia Z1 and WiSMote, respectively. Finally, the paper is concluded in Section 5, followed by References.

Peer-review under responsibility of the Organizing Committee of ICRTCSE 2016
doi:10.1016/j.procs.2016.05.160

2. Literature Review

Low-Power and Lossy Networks (LLNs) comprise mainly of constrained nodes with limiting processing power and volatile energy. Traffic patterns are mostly not point-to-point, but mostly multipoint-to-point, or multipoint-to-multipoint. These lead to typically lower data rates leading to instability [5]. Contiki is an operating system which supports lossless and low-power monitoring of Internet of Things devices, and supports RPL. The topological assignment of nodes is based on multi-hop transmissions and it has been used in environment monitoring, health-care and other smart systems [6].

RPL is a dynamic routing protocol which is aimed at communication from a source node to a sink node with the increased complexity and overhead. There are several instances where a network can achieve only half of the throughput achieved than the corresponding lossless network, and hence packet retransmissions occur. Studies have shown that 50 to 80% of the energy for communication is wasted in overcoming packet collisions and environmental factors [7]. The path selection in RPL uses various factors which compute the best paths with the best routing metrics. RPL ensures advanced monitoring applications in several conditions.

The various validity threats, the internal and the external play a major role in making conclusions. The Simulation of lossy medium is emulated by using the Cooja feature Unit Disk Graph Model (UDGM). The real network of lossyness may not be accurately validated. Nevertheless, the simulation confirms the functionality and behaviour of RPL [8]. The main objective of analysis is to observe the impact of the various RPL parameters on its performance with respect to metrics like Latency, Energy Consumption, etc. These differences in behaviours are observed and they help in determining changes in factors affecting them.

3. Performance Analysis of Zolertia Z1 using RPL

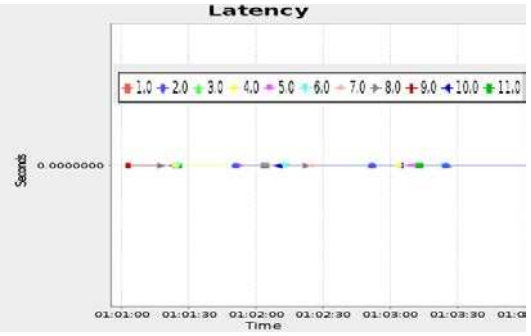


Fig. 3.1(a) - 1 sink, 10 senders (Latency)

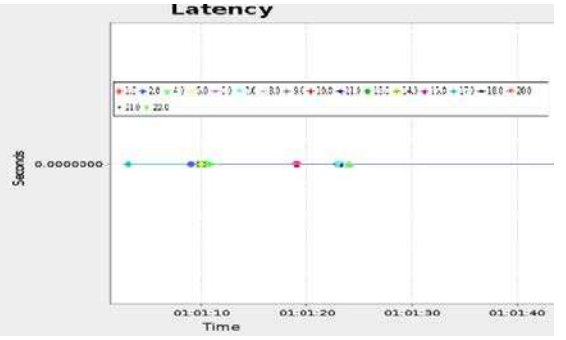
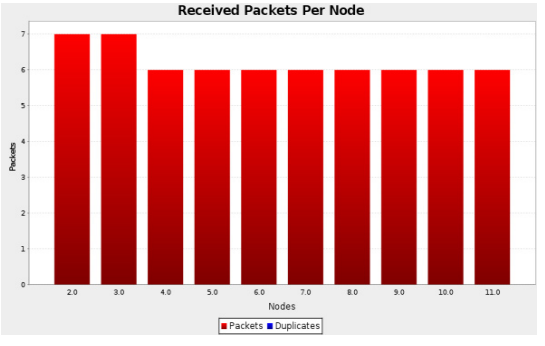


Fig. 3.1(b) - 2 sinks, 20 senders (Latency)



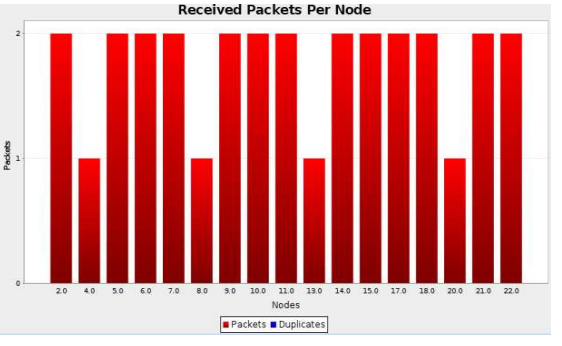


Fig. 3.2(a) - 1 sink, 10 senders (Packets per node)

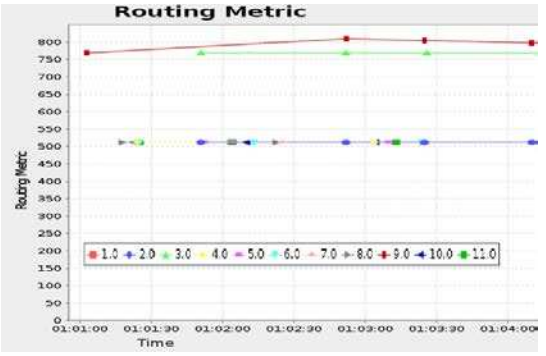


Fig. 3.3(a) - 1 sink, 10 senders (Routing Metric)

Fig. 3.2(b) - 2 sinks, 20 senders (Packets per node)

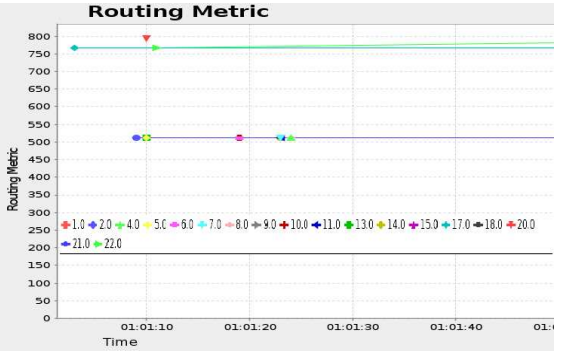


Fig. 3.3(b) - 2 sinks, 20 senders (Routing Metric)

4. Performance Analysis of WiSMote using RPL

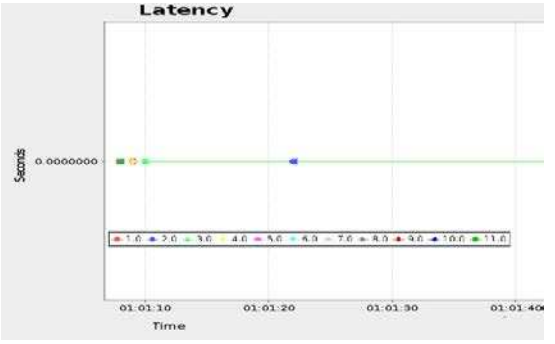


Fig. 4.1(a) - 1 sink, 10 senders (Latency)

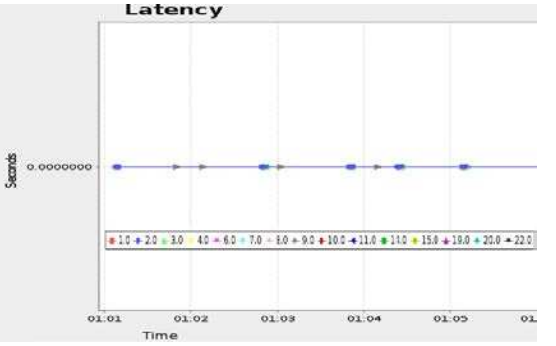


Fig. 4.1(b) - 2 sinks, 20 senders (Latency)

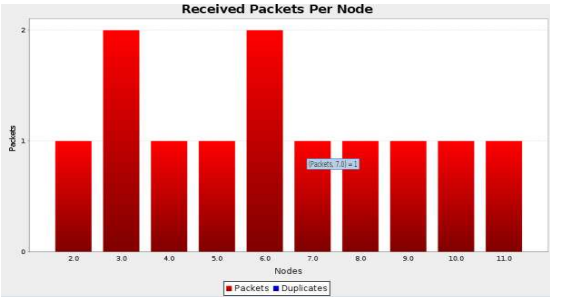


Fig. 4.2(a) - 1 sink, 10 senders (Packets per node)

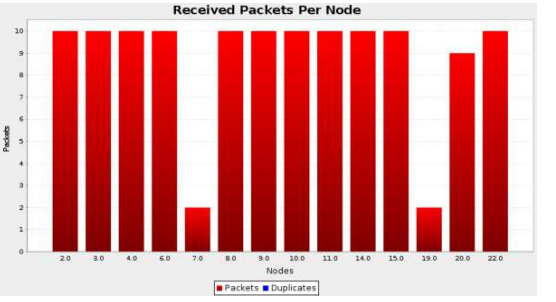


Fig. 4.2(b) - 2 sinks, 20 senders (Packets per node)

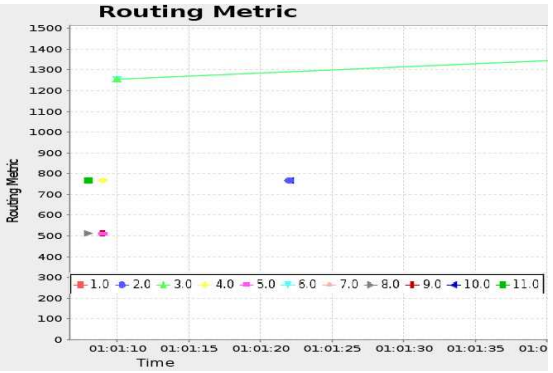


Fig. 3.3(a) - 1 sink, 10 senders (Routing Metric)

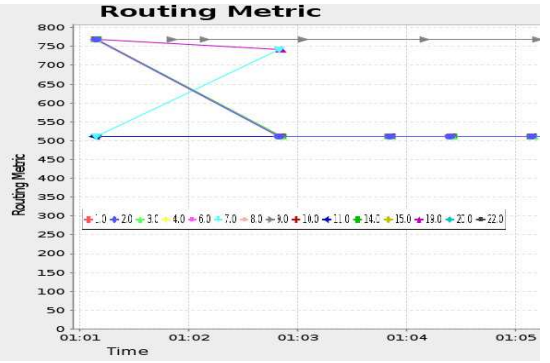


Fig. 3.3(b) - 2 sinks, 20 senders (Routing Metric)

The amount of time it takes, for a packet of data to move across a network connection (latency) has been observed for the 2 motes. Fig 3.1(a) and 3.1(b) show the latency graph for Z1 for two different scenarios (1 sink with 10 senders, and 2 sinks with 20 senders) and Fig 4.1(a) and Fig 4.1(b) show the latency graph for WiSMote for two different scenarios (1 sink with 10 senders, and 2 sinks with 20 senders), it is clearly observed that more packets are transmitted by Z1 mote than WiSMote in a particular time interval (0 - 4 sec).

Fig 3.2(a) and Fig 3.2(b) indicate the number of packets that have been received per node, it is observed that the average number of packets received per node in a 10 sender - 1 sink Zolertia Z1 scenario is 6 whereas, the average number of packets that have been received per node in 20 sender - 2 sink Z1 mote scenario is 2. The previously observed situation gets reversed in WiSMote. Fig 4.2(a) and Fig 4.2(b) show number of packets received per node in a 1 sink - 10 sender WiSMote scenario, it is observed that the average number of packets received is 1, whereas in a 2 sink - 20 sender WiSMote scenario, it is observed that the average number of packets received is 10.

The routing metric for the individual nodes of Zolertia Z1 and WiSMote have been depicted in Fig 3.3(a), Fig 3.3(b) respectively, it is observed that Z1 motes largely travel by 2 different paths which are considered to be optimal for them whereas the network traffic is directed across several paths in WiSMote.

5. Conclusion

The increase in number of devices with communicating–actuating capabilities is popularising Internet of Things (IoT) where blending of sensing and actuation functions takes place in the background and opens up a wide range of possibility for WSN based real-time application. Using RPL for analysis of WSN can enhance their functionality. This paper presents an overview of analysis of sensors using RPL where Zolertia Z1 and WiSMote were considered. The performance of each node of Zolertia Z1 and WiSMote in terms of Latency, Received packets per node, Routing metric has been investigated in detail using RPL. In our future works, we have planned to investigate in detail other network factors such as ETX, Beacon interval and Network hops. Zolertia Z1 is hence more suitable for Agriculture (larger field area), while WiSMote is more suitable for Smart Building (relatively smaller area).

References

[1] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, Marimuthu Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions", Future Generation Computer Systems, Volume 29, Issue 7, 2013, pp.1645–1660.

[2] J. P. Vasseur, R. Kelsey, R. Struik, P. Levis, T. Winter, A. Brandt, J. Hui, K. Pister, T. Clausen, and P. Thubert, "RPL: IPv6 Routing Protocol for Low power and Lossy Networks", Internet Engineering Task Force (IETF), ISSN: 2070-1721

[3] "Zolertia Z1 mote" [Online], http://zolertia.io/z1

[4] Hazrat Ali, "A Performance Evaluation of RPL in Contiki" Master Thesis, School of Computing, Blekinge Institute of Technology

[5] "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks" [Online], https://tools.ietf.org/html/rfc6550

[6] Tsung-Han Lee, Xiang-Shen Xie , Lin-Huang Chang, "RSSI-Based IPv6 Routing Metrics for RPL in Low power and Lossy Networks",

Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC), 5-8 Oct. 2014, San Diego, CA, pp. 1714 – 1719.

[7] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks" Proceedings of the First International Conference on Embedded Networked Sensor Systems (Sensys), Los Angeles, CA, 2003, pp. 1-13.

[8] "WiSMote" [Online], http://www.aragosystems.com/en/wisnet-item/wisnet-wismote-item.html