CONTEXT-AWARE MIXTURE OF DEEP NEURAL NETWORKS

A Thesis

by

ARASH PAKBIN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2020

Major Subject: Computer Science

ABSTRACT

One of the key challenges activity recognition in wearable computing faces is that activity characteristics may be context-dependent and change under different situations, requiring flexibility and adaptability of the algorithm. We develop α-β network, a context-aware mixture of deep models, to enhance human activity recognition performance. This framework has also shown promising results for opportunistically selecting sensor subsets in noisy environments and/or when the number of sensors used to make predictions is to be minimized.

DEDICATION

This thesis is dedicated to my grandmother Zahra, my grandfather Bahman, my father Mehrdad, my mother Mitra, and my sister Azarnoosh.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Dr. Bobak Mortazavi and Zhangyang (Atlas) Wang of the Department of Computer Science and Engineering and Dr. Xiaoning Qian of the Department of Department of Electrical & Computer Engineering.

Dataset preparation in 4.3.2 was done by Nathan C. Hurley of the Department of Computer Science and Engineering. Figure 4.3 was done by Bobak Jack Mortazavi of the Department of Computer Science and Engineering.

Funding Sources

My research that led to this thesis was funded by National Science Foundation Engineering Research Center for Precise Advanced Technologies and Health Systems for Underserved Populations (PATHS-UP) (Award #1648451).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Tracking individuals throughout their days has shown to be useful in helping maintain/regain their health. For instance, there has been research showing food intake tracking results in a healthier diet [3] and, on the other hand, evidence has been found that exercise tracking improves recovery in heart failure patients [4]. Daily tracking has been made more possible than anytime before through ubiquitous wearable sensors [5].

In today's era, digital information is everywhere. Sensors are built and used in many settings which produce data. Back to our example, Human Activity Recognition (HAR) is concerned with the recognition of actions and goals of humans based on observations made through the sensors. HAR enables us to provide personalized support to users and address their needs based on what they are doing and what their goals are. HAR is the cornerstone of nearly all remote health monitoring systems as information regarding daily activities have proven to be valuable in diagnosing potential and existing health problems. Everyday, the amount of EHR grows in the world as sensors are prevalent and measurements are made. As another example, Electronic Health Record (EHR), is a collection of health data of patients which is electronically stored [6] which can be leveraged to make decision support systems to elevate healthcare. In fact, HAR and EHR can be seen as use cases for Internet of Things (IoT) where there are several sensors which can be leveraged for recognizing objectives which here are the activity being performed and health assessments respectively. IoT plays a major role in prevalence and effectiveness of systems such as HAR and EHR as it provides an ecosystem of "things" which collect and share data [7] which can be leveraged to collect information.

IoT is a rapidly growing field and as everyday passes, the complexity and heterogeneity of data it provides grows [7]. Wearable devices can be seen as an example in this field. Many people use wearable devices such as smartphones as smartwatches and their number is growing as more people start using them with the advent of new technologies and because of the fact that have become more accessible the last few years. Additionally, the number and heterogeneity of sensors used in them is also growing. Everyday, more sensors are added to the wearable devices which add to the heterogeneity and complexity of the the data they are collecting.

However, "with great power, comes great responsibility". As the amount of data grows, making sense of the data and processing it becomes more challenging. On the other hand, IoT devices have shown to be significantly more effective in constrained environments rather than unconstrained ones [8]. In more unconstrained environments, since the number of possibilities are very large compared to constrained environments which is the case for most of the times in labs, additional data sources can be useful as they reduce the number of possibilities and consequently make the prediction task easier [9]. As an example which is already in reflected in their work, knowing the place in which an activity takes place helps in detecting it since not all activities have the same probability of happening in different environments. The mentioned additional data can be used in dividing the heterogenoeus data which is inherent in more complex (and less constrained) environments into smaller data groups which can be modeled differently. This “additional data” can be referred to as “context”.

Context awareness can aid in managing large quantities of data. Context can enable the IoT system to decide about the importance of different information coming from different sources[10]. Context can be defined as “any information that can be used to characterize the situation of a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves”[7]. In other words it makes "choosing" among different information sources possible depending on the "situation" which is defined using contextual information.

To give a specific example, context awareness in IoT has three different features: presentation, execution and tagging [7]. Presentation enables a context-aware IoT system to choose the most relevant information and services to present to the user. Execution enables the system to perform automatically and take actions based on a given situation. Tagging allows a context-aware system to have information about the data collection, such as the sensors in charge of specific parts of the data or the time they were collected [7]. Context awareness allows for extracting new knowledge from previously collected data for higher level understanding of a given situation [11, 12].

Context is any additional information that can be used to improve recognition [13]. It provides a high-level understanding of the data and recognitions of the need to use the data in context-dependent fashion. Essentially, context allows for choosing among different information sources depending on different situations. By imposing context information on any effort to make sense of data, context-aware reasoning benefits a range of decision making tasks in reducing output search spaces. Context-aware approaches have previously been investigated, developing a classification of context as a first step towards a personalized recognition system [14]. Context-awareness has been shown to boost performance in many applications, including activity recognition [15].

However, as Table 1 shows, context definitions in many works, are either too narrow or too abstract and high level. Context is defined based on the usage in each specific domain. This problem will hamper the scalability of context as it needs to be updated often and needs to be adjusted based on the domain. For example, the first entry in Table 1 shows the contexts defined as 7 activities which were narrowly described. Such a definition for context leaves out a great area of blank space where it can not fit into the context. The person may be in an 8th context which has not been defined and the system cannot function properly in that case. Moreover, in case of trying to define all the possible contexts in such a granular way, the context modeling would be too complex. Context could also be defined too broad and general that could not fit many applications. For example it could be “Low Level Location” entry in Table 1. If we only know we are at a certain place, such as home, it may not be helpful in some applications that need to know whether you are in front of a TV, for example, or not. The overall point would be that defining the context too broad or too narrow has the problem that it needs to be redefined, and there is a need for a definition which can address a broader array of problems.

Also, the way context is formulated at the moment is that it is defined first, and then used for grouping/processing the data. Another use case of the context, if defined data driven, would be the opposite. Data clusters could be formed to come up with contexts and that could be used for explaining contextual information in data, rather than using context to group data together.

In this work, we want to define a context ontology which does not need to be redefined for every domain and use case. For it to not be redefined, it needs to be data driven. In that case, the need for domain knowledge would be minimized. In our work, instead of trying to use context to explain data, we try to find contextual information that make data more reasonable. We gain insights from the data, rather than using the data at hand to make sense of the data. Such a context discovery would be totally unsupervised and does not need any information regarding the context from the data. Contexts, after extraction, can be used for clustering and explaining the data. Also, in addition to explaining the data, we still want our context-aware system to show the gains achieved by context awareness. One of the most important gains of context-awareness is performance boost which we expect our model to have.

Defining context before modeling and defining it solely based on the data are two extreme cases. As another objective of our work, we plan to make a balance between the two. We use the contextual information opportunistically to guide our context discovery. The context discovery in this case would be semi-supervised in a sense that the contextual information does not need to be known at all times, and it is used opportunistically, if available, to guide the discovery. This way, we will have a balance between the two extreme cases.

2. LITERATURE REVIEW

2.1 Mixture of Experts (MoE)

In many applications of machine learning, heterogeneous data can be divided into smaller homogeneous groups that can be modeled more accurately. Context-awareness, as an example, plays an important role in improving the performance of activity recognition systems [16, 17]. In Mixture of Experts (MoE) models [1] such group-level clustering and modeling comes as a single training step, rather than splitting data a prior then building models. Elements are clustered based upon their relationship and the next level modeling accuracy. MoEs have successfully served different applications from classification and regression tasks [18, 19] to phenotyping in medical datasets [20]. In [1, 18, 21], authors provided formulations of probabilities of observed given different experts.

As mentioned earlier, in many applications of machine learning, heterogeneous is better modeled through division into smaller homogeneous groups. Mixture of Experts (MoE) [1] performs such group-level modeling inherently. MoE has been proposed to solve nonlinear supervised learning problems. MoE first divides the input space into a set of regions with soft boundaries and uses simpler surfaces for fitting the data in each region. MoEs have successfully served different applications from classification and regression tasks [18][19] to phenotyping [20]. In the conventional MoE, class assignments are completely unobserved and they are chosen in the training process in a way that maximizes the log likelihood in each iteration [22].

The architecture Jordan et al. proposed in [1] is a hierarchical structure of unspecified length (Figure 2.1) but in each layer of hierarchy, there are two types of networks:

- **Gating network**: This network sits at non-terminals and by receiving a vector $\mathbf{x}$ as the input, produces an output on a simplex which is interpreted as a distribution over different expert networks as it is a partition of unity.
- **Expert network**: These networks sit at leaves, accept the same input $\mathbf{x}$ to produce an output

which is the type of output we want our whole framework to output. For example if we are solving a regression problem which is modeling a value based on time **t**, each expert network accepts a time **t** input variable and produces a value.

Therefore, at each layer, for each input **x**, there will be several outputs **y** which need to be combined. The combination is a weighted average based on the Gating network output. In other words, the output vectors of Expert networks get multiplied by gating network outputs and are summed at each terminal to be provided to the upper level in the tree if exists.

The assumptions that MoE makes for modeling the data are interesting and important in understanding it. The assumption is that there are several experts, several sources, each having its own class, that can produce the output given the input. Each of them behave according to a class-conditional distribution of output given input. Even their distribution families is not necessarily the same. However, in practice, they are assumed to be the same in most cases but with different

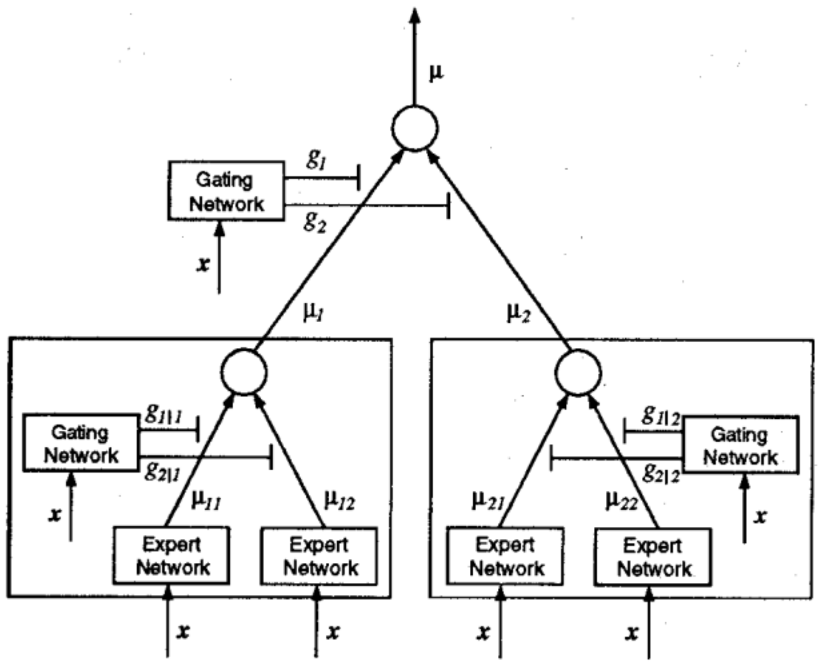


Figure 2.1: The Mixture of Experts (MoE) schematic laid out by Jordan et al. Graph reprinted from [1].

parameters. So given different class-conditional distributions implemented by Expert networks, the Gating network gets the final say in the importance of each of those distributions in the output of the whole layer based on the probability it assigns to it.

However, the expert assignments is not present in the data and the expert assignment process needs to be done in an unsupervised manner. Therefore, the training is done through maximum likelihood. The likelihood function, which has been written as log-likelihood because of computational convenience and stability, can be written as:

$$l(\theta; \mathcal{X}) = \sum_t ln \sum_i {g_i}^{(t)} \sum_j {g_{j|i}}^{(t)} P_{ij}(y^{(t)}) \tag{2.1}$$

However, according to Dempster et al., Eq. 2.1 is intractable and since the experts assignments are "missing"/"hidden", the log-likelihood is "incomplete" [22]. The "complete" data likelihood would have expert assignments. Dempster et al. propose maximizing the log-likelihood in an iterative way where each iteration consists of two steps:

- **E step**: finding the expected value of complete data log-likelihood based om the current estimation of all the parameters
- **M step**: Optimizing the parameters by maximizing the computed expected value

The algorithm explained above is named Expectation-Maximization (EM) and can be shown that converges to a local maximum. Therefore multiple runs with different initialization values help in achieving a good solution.

MoE has gained more popularity recently. We briefly talk about two of the recent works who have gotten their main idea from MoE framework. Shazeer et al. believe that MoE structure can be used to increase model capacity by adding more expert networks but not necessarily model computation if not a lot of them are used for producing the output [23]. They make the gating sparse (not a large number of networks are turned on for each prediction) and in order to ensure that the utilization of experts is balanced, they have come up with a definition of importance for each expert and they enforce that. As another way to expand the MoE idea, Aljundi et al. use the same idea of selector-expert present in the MoE framework. They dedicate a network to each task. For the gating which decides which network to use, they use the reconstruction error of an autoencoder which is dedicated to that network and that decides which one to use.

2.2 Context

Context life cycle is “how sensor data is collected, modeled, and processed, and how knowledge is extracted from the collected data.” [7]. Therefore, specific frameworks need to be developed to text context into account. Context life cycles can be different and can have different phases [11] but they all share common stages [10, 24]:

- Context acquisition: obtaining data from different sources
- Context modeling: using data which makes sense the most to model the whole data
- Context reasoning: processing raw data and extracting information
- Context distribution: distribution of knowledge

Now we delve into more detail of each specific segment of context life cycle:

1. Context Acquisition: In context acquisition, five different factors are taken into consideration [11]: the acquisition process, frequency, responsibility, sensor types, and source. As can be seen, heterogeneity of data comes naturally in context awareness as the difference in data such as in types and frequencies is considered.

2. Context Modelling: Context needs to be presented in a way that helps understand its properties and relationships. Context is domain and feature dependant. Different requirements call for different context modelings. Context modeling can be assessed using different measures, namely [25, 15]: distributed composition, efficient context provisioning, dependencies and relationships, heterogeneity and mobility, imperfection, incompleteness and ambiguity, level of formality, partial validation, reasoning, richness and quality of information, timeliness, and usability of modeling formalisms [25, 15]. Context modeling can be done in different

ways such as multidisciplinary, domain focused, user-centric, and chemistry inspired [24]. Here, we just talk about ontology based context modeling since it has the most similarity and relevance to our work. In ontology based modeling, context is modeled with ontology, and represented with semantic ontology languages. Ontology based modeling can be used for both reasoning and knowledge extraction. Nonetheless, complex representations in ontology languages may make information retrieval hard.

3. Context Reasoning: Context reasoning involves leveraging the higher level knowledge to make sense and reason and using the available context to extract new knowledge [11, 26]. Context reasoning has three main steps [27, 10]:

    - Context preprocessing: In preprocessing, data is first cleaned and then methods such as dimensionality reduction, feature subset induction are used for further processing. Outlier elimination and data imputation are also done in this stage.
    - Sensor fusion: In sensor fusion, as the name implies, data coming from different sensors and modalities are combined. This will increase the dependability, accuracy and reliability which cannot be provided by single sensor data.
    - Inference phase: In order to produce high level context, lower level context is mapped to higher level and its main focus is to recognize new context

1. Supervised learning: Supervised training data consists of labeled data points meaning that each sample drawn from the training distribution has a vector of features along with label(s). A model trained on such a dataset will be used for predicting labels. Supervised learning techniques are abundant. Here, we talk about algorithms that are most relevant to our approach which are: artificial neural networks and ensembles of classifiers.

    (a) Artificial Neural Networks (ANN): ANNs, computing systems which have been inspired biological neural networks, have proven to be a strong machine learning algorithm that have shown superior performance in many domains. ANNs have been used successfully for human activity recognition with wearable sensors [11, 28]. Deep learning is a term used when ANNs get "deep" meaning the number of layers get large. Convolutional Neural Networks (CNN) are a type of neural networks which have convolutional layers used for extracting features from images.

(b) Ensemble Algorithms: In ensemble algorithms, instead of using a single classifier, multiple classifiers are used together for achieving a better performance. The results from single classifiers can be combined in many ways: voting, bagging, and boosting. In voting, each classifier generates an answer and the answer which has the largest number of repetitions is declared as the final answer. In bagging, random subsamples of the qqqtraining set are used for training different models. In boosting, the method is similar to bagging with a difference that the final answer of the classifiers are weighted based on the difficulty of classification.

2. Unsupervised learning: The aim of the unsupervised learning technique is to cluster the unlabeled data. In this technique, the data samples only have features and they do not come labeled so the task would be to group them. In this technique, a large corpus of heterogeneous data is divided into smaller homogeneous subsets. Among all the different techniques in unsupervised learning, we talk clustering. On of the known clustering algorithms is K-means where K points in the data space are chosen as the mean for K overlapping subgroups of data. K-means has been used for context reasoning in [29].

2.3 Uncertainty Quantification

Estimating uncertainty in an AI system is very important in many aspects such as safety [30]. Uncertainty estimation would be of paramount importance in cases where the cost of making a mistake in an AI system is high such as in decision support systems and autonomous vehicles. Malinin et al. give a good description on different types of uncertainty [30]:

- **Model uncertainty** is a measure of the uncertainty in estimating the model parameters given the training data. In other words, it measures how well the data is explained by the model.

This type of uncertainty reduces as the training data size increases.

- **Data uncertainty** is due to inherent complexities in the training data such as class overlap and label noise. This type of uncertainty does not reduce with increasing the training data size.

- **Distributional uncertainty** happens when the distribution from which the training data is drawn is different from that of the testing data.

According to Malinin et al., the uncertainty in prediction is defined on the posterior distribution over class labels after observing the training data $P(w_c|x^*, \mathbf{D})$ where w_c are class labels, x^* is the input vector for which the output is computed, and $\mathbf{D}$ which is the training data. They introduce an interesting breakdown of uncertainty:

$$P(w_c|x^*, \mathbf{D}) = \int\int P(w_c|\mu)p(\mu|x^*, \theta)p(\theta|D)d\mu d\theta \tag{2.2}$$

Where $P(w_c|\mu)$ which is the distribution of class outputs given the true output is the data uncertainty, $p(\mu|x^*, \theta)$ is the distribution of the true output given model parameters and the input and models the distributional uncertainty and $p(\theta|D)$ is the model's uncertainty as it models the uncertainty in estimating the parameter set θ after observing the data. They also introduce measures of uncertainty which is ways of assigning an uncertainty to the distributions and starts from easier cases such as maximum probability and entropy to mutual information and differential entropy.

Softmax layers are popular for the output layer of neural network classifiers. They produce values similar to distributions over outputs but there has been shown that interpreting them as probabilities is not accurate. However, Hendrycks et al. [31] have found meaningful patterns in the statistics of prediction probability of incorrect and out of distribution examples as they tend to be lower than the prediction probability for correct examples.

Backed by the fact that softmax output layers are not ideal for uncertainty quantification based on probabilities, some works have suggested replacing the softmax layer with Dirichlet distribution therefore the deterministic network would output the parameters of that Dirichlet distribution which results in more expressive power of uncertainty [32]

On the other hand, there are other works that first train a model using a softmax output layer and then by making changes they perform UQ. For example Liang et al. perform out of distribution detection through temperature scaling and small perturbations to the input [33] and Lee et al. put class condotional Gaussian distributions on top of a softmax trained network and perform out of distribution detection through Mahalanobis distance [34].

Another interesting and important principle is the maximum entropy principle based on which we find the highest entropy distribution among all that satisfy a condition [35]. In discriminants which are our focus, such conditions are:

$$\int P(\theta,\gamma)[y_t\mathcal{L}(X_t|\theta)-\gamma_t]d\theta d\gamma \geq 0 \tag{2.3}$$

Which describes all the distributions over discriminant parameters where their expected value makes true predictions [36].

3. PROPOSED WORK[*]

3.1 Context-aware Mixture of Deep Neural Networks

In this work, we develop a context-aware mixture of deep neural networks to learn the context from the data. In our design, we dedicate a network to detection of context and several networks which are specific to contexts. We will first give an explanation regarding how the framework is formulated, then we will explain what the network structure looks like and how it is trained. Also, we will explain how the number of contexts would be determined in such an unsupervised manner.

Without loss of generality, we explain the formulas in the context of activity recognition but they can be applied to other domains as well.

In this work we develop a mixture of neural networks where each mixture component is dedicated to one specific context. There are two types of networks in our methodology: Alpha and Beta. Given the sensor data, the Alpha network detects context by generating a probability distribution over all known contexts. Each context has a dedicated Beta network that outputs a probability distribution over different activities. Our activity recognition problem features a latent context variable and can be formulated as:

$$\begin{aligned} \log P(ACTIVITY|X,\theta) &= \sum_{n=1}^{N} \log P(activity_i|x_i,\theta) \\ &= \sum_{n=1}^{N} \log \sum_{c=1}^{N_c} P(activity_i, c_i = c|x_i,\theta) \end{aligned} \tag{3.1}$$

where θdenotes the mixture component parameters, N denotes the number of data samples, and N_c denotes the number of expected clusters (contexts) to which each data point may belong. Our objective is to maximize Eq. 3.1 with respect to θ The log-likelihood has a lower bound:

[*]Reprinted with permission from “Uncertainty Quantification for Deep Context-Aware Mobile Activity Recognition and Unknown Context Discovery” by Huo, Zepeng and PakBin, Arash and Chen, Xiaohan and Hurley, Nathan and Yuan, Ye and Qian, Xiaoning and Wang, Zhangyang and Huang, Shuai and Mortazavi, Bobak, 2020. AISTATS, PMLR 108:3894-3904, Copyright 2020 by PMLR. URL: http://proceedings.mlr.press/v108/huo20a/huo20a.pdf.

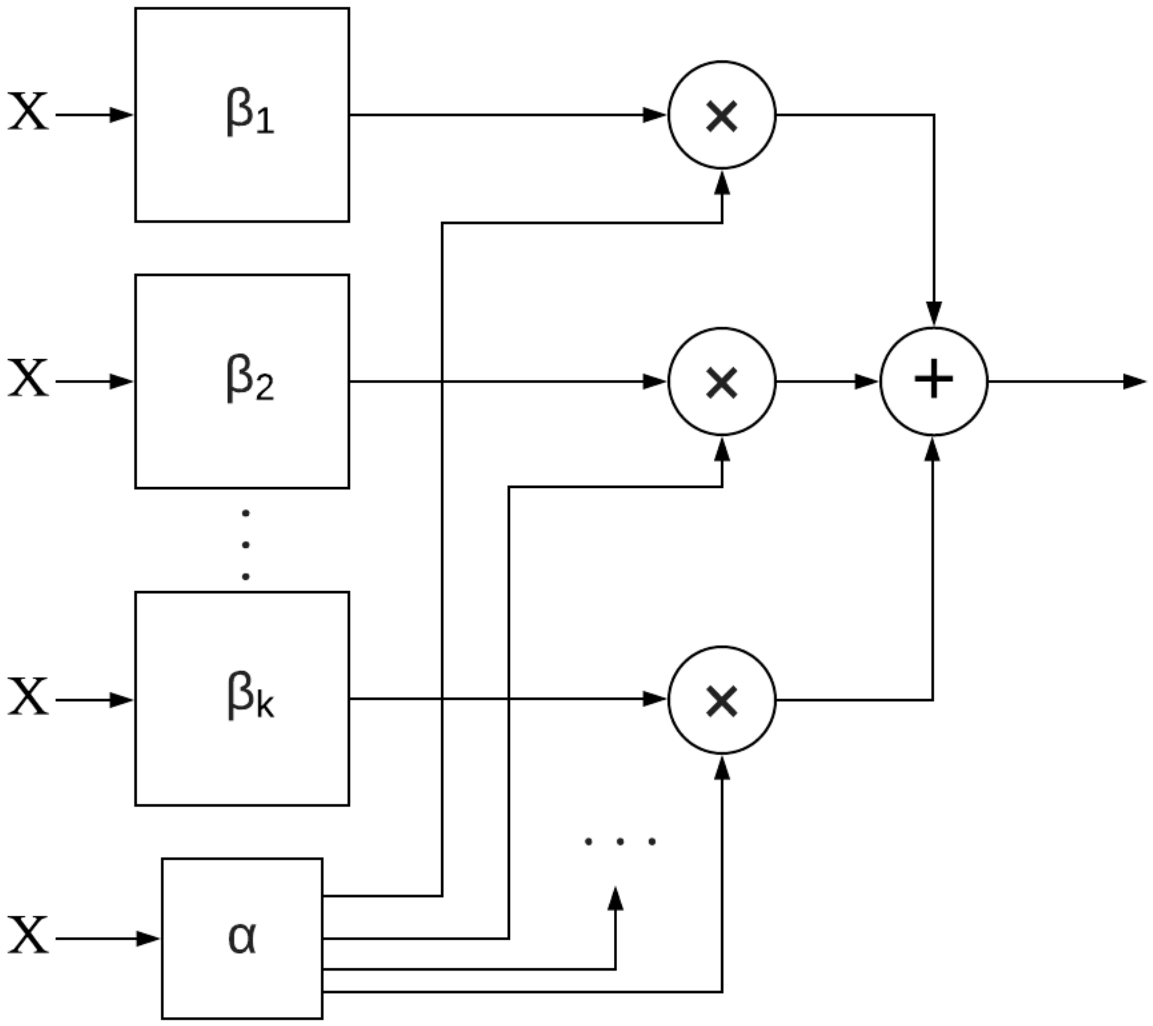


Figure 3.1: The architecture of α-β network. Figure reprinted with permission from Huo et al. AISTATS 2020 with reference number PMLR 108:3894-3904, 2020 [2].

$$
\begin{aligned}
\log P(ACTIVITY|X,\theta) = \\
\sum_{n=1}^{N} \log \sum_{c=1}^{N_c} P(activity_i, c_i = c|x_i, \theta) \\
\geq \sum_{n=1}^{N} \sum_{c=1}^{N_c} q(c_i = c) \log \frac{P(activity_i, c_i = c|x_i, \theta)}{q(c_i = c)} \\
= \sum_{n=1}^{N} \sum_{c=1}^{N_c} q(c_i = c) \log \frac{P(activity_i|c_i = c, x_i, \theta).P(c_i = c)}{q(c_i = c)} \\
= \mathcal{L}(\theta, q)
\end{aligned}
\tag{3.2}
$$

Where $q(.)$ is the distribution over different contexts and $P(activity_i|c_i = c, x_i, \theta)$ is the distribution over activities given the context and the input. $q(.)$ is modeled using the context-detecting Alpha network, and each $P(activity_i|c_i = c, x_i, \theta)$ is modeled using a Beta network (each context has its Beta network). The lower bound in Eq. 3.2 can be maximized using the EM algorithm. It should be noted that no labeled contextual data is used in the training process and the Alpha-Beta network is learning the context in an unsupervised manner.

After training Alpha-Beta network, Alpha network discovers the context given sensor readings and each Beta network detects the output in its specific context. No ground truth of context is used for training the Alpha-Beta network and it only uses activity ground truth for training. The model, while training, will try to put similar data samples together in the same context so their modeling would be easier and would give the best performance in terms of accuracy and F1-score.

This likelihood lower bound can be maximized using Expectation Maximization algorithm [22] in two steps:

E-step:

$$
q^{k+1} = arg \underset{q}{max} \mathcal{L}(\theta^k, q) \tag{3.3}
$$

M-step:

$$
\theta^{k+1} = arg \underset{\theta}{max} \mathbb{E}_{q^{k+1}} \log P(X, C|\theta) \tag{3.4}
$$

However, in this unsupervised formulation, clusters do not necessarily contain contextual information. This framework can be enhanced to use contextual data opportunistically in a semi-supervised manner. This formulation does not consider the fact that by knowing the locality, we have environmental information as it treats the clusters as unobservable. So the clusters do not necessarily contain environmental information. In order to inject environment meaning into the latent variables and identify the optimum, the E-step is regularized with the Kullback-Leibler (KL) divergence. The KL divergence was used as a measure of similarity between two distributions to penalize the posteriors that were less similar to the context distribution collected from the user's annotations. Therefore, the new E-step becomes:

E-step:

$$q^{k+1} = arg\underset{\theta}{max}\mathbb{E}_{q^{k+1}}\log P(X, C|\theta) - \lambda KL\left(q||context\right) \tag{3.5}$$

By using this formulation, we promote posterior distributions which are more similar to environmental distribution in order to inject environment meaning into them. The M-step remains as standard to improve activity recognition task.

Another question that remains to be answered is how the number of contexts is determined as it needs to be known before starting the training. The answers are different for the cases of unsupervised and semi-supervised. In the unsupervised manner, there is no ground truth using which we can get the number of contexts. Therefore, we use the number of contexts which gives the best results in terms of performance in the internal loop of the training set, such as f1-score. However, in the case of semi-supervised, we have ground truth for the labels and we set the number of contexts as the dataset as we want to use those labels in the training phase.

Additionally, we have formulated another method for semi-supervised mixture of experts which separates the training points in which the context is observed or not. In our formulation below, we have assumed a discriminative model $D(Y|X)$ which accepts X as the input and outputs Y. The model is assumed to be a MoE with Z as the latent variable governing the mixture component assignment for each sample. The data consists of two parts, a part with the latent variable observed (with o as the subscription) and a part that the latent variable is not observed. The primary goal of our formulation is to estimate the distribution of the latent variable when not observed, given all the data. This is done using variational inference and is started by writing the KL divergence between the distribution of unobserved latent variable and the variational distribution q in Eq. 3.6. Note that the variational probability is conditioned on X because the model is discriminative.

$$\begin{aligned}&KL(q(Z|X)||P_\theta(Z|X,Y,X_o,Y_o,Z_o))\\=&\mathbb{E}_{q_{|X}}[\log q(Z|X)] - \mathbb{E}_{q_{|X}}[\log P_\theta(Z|X,Y,X_o,Y_o,Z_o)]\end{aligned} \tag{3.6}$$

In Eq. 3.6, the distribution of Z conditioned on all the observed data can be written as:

$$\begin{aligned}&\log P_\theta(Z|X,Y,X_o,Y_o,Z_o)\\=&\log P_\theta(Z,X,Y,X_o,Y_o,Z_o) + \log P_\theta(X,Y,X_o,Y_o,Z_o)\\=&\log P_\theta(Z,X,Y,X_o,Y_o,Z_o) + const\end{aligned} \tag{3.7}$$

Since unobserved and observed data are independent, the full data log likelihood in Eq. 3.7 can be further simplified as:

$$\begin{aligned}&\log P_\theta(Z,X,Y,X_o,Y_o,Z_o)\\=&\log P_\theta(Z,Y|X) + \log P_\theta(X)\\+&\log P_\theta(Z_o,Y_o|X_o) + \log P_\theta(X_o)\\=&\log P_\theta(Z,Y|X) + \log P_\theta(Z_o,Y_o|X_o) + const\end{aligned} \tag{3.8}$$

Therefore, the KL divergence in Eq. 3.6 can be written in the form of:

$$\begin{aligned}
&KL(q(Z|X)||P_\theta(Z|X, Y, X_o, Y_o, Z_o)) \\
=&\mathbb{E}_{q_{|X}}[\log q(Z|X)] \\
-&\mathbb{E}_{q_{|X}}[\log P_\theta(Z, Y|X) + \log P_\theta(Z_o, Y_o|X_o)] \\
+&const
\end{aligned} \tag{3.9}$$

Consequently, the Evidence Lower Bound (ELBO) [37] is:

$$\begin{aligned}
&ELBO \\
=&\mathbb{E}_{q_{|X}}[\log P_\theta(Z, Y|X)] + \mathbb{E}_{q_{|X}}[\log P_\theta(Z_o, Y_o|X_o)] \\
-&\mathbb{E}_{q_{|X}}[\log q(Z|X)]
\end{aligned} \tag{3.10}$$

In Eq. 3.10, since the second term corresponds to the data with observed latent variable(s), it can be simplified into:

$$\begin{aligned}
&\mathbb{E}_{q_{|X}}[\log P_\theta(Z_o, Y_o|X_o)] \\
=&\log P_\theta(Z_o|X_o) + \log P_\theta(Y_o|Z_o, X_o)
\end{aligned} \tag{3.11}$$

By plugging Eq. 3.11 into Eq. 3.10 we obtain:

$$\begin{aligned}
&ELBO \\
=&\mathbb{E}_{q_{|X}}[\log P_\theta(Z, Y|X)] - \mathbb{E}_{q_{|X}}[\log q(Z|X)] \\
+&\log P_\theta(Z_o|X_o) + \log P_\theta(Y_o|Z_o, X_o)
\end{aligned} \tag{3.12}$$

The form of ELBO in Eq. 3.12 is interesting. The first two terms correspond to the conventional case where the latent variable is unobserved. The second two terms correspond to the case where we have a hierarchical model which has all the variables observed. In such model, selection of Z_o given X_o is decoupled from the selection of Y_o given Z_o and X_o and their models can be trained as such.

3.2 Evaluation

In order to show the success of our proposed work, we will perform three different evaluation processes. The first process will evaluate the impact of our context aware modeling (unsupervised and semi-supervised) on activity predictions.

The opportunistic labeling algorithm will be evaluated in terms of classification performance. We will use commonly used metrics in machine learning that show an overall performance on the classification process. We propose to use F-1 score as our main indicator of performance since it accounts for both precision and recall which makes it more fair than precision and recall by themselves. We will also use overall accuracy to show how are our labels being classified. Therefore, to demonstrate that our Alpha-Beta algorithm provides a performance boost, we will compare the F-1 score and accuracy of the prediction of different activities. We expect to see an improvement on the F-1 score and accuracy.

4. EXPERIMENTS*

4.1 Dataset

UCI data. We used the UCI OPPORTUNITY dataset [38] for context-aware human activity recognition. The dataset contained 18 different activities performed in five different contexts and sensed by 72 different sensors. Each of the 18 activities had one of the five contexts, but not all contexts contained every activity. Therefore, the UCI OPPORTUNITY dataset resembles the true situation in which a context may have several activities and not all the activities happen with the same probability in a specific context. This dataset contains seven levels of hierarchical labels. Higher level labels are involved with more overall and high-level details such as subject posture, while lower level labels described the hand movements or interactions with other subjects. In this study, we chose a higher level label (e.g., cleaning time) as the context and a lower level (e.g., opening a door) label as the activity.We used all the body-worn sensors which included seven inertial measurement units (IMUs) and twelve 3D acceleration sensors. Five IMUs were on the upper body while two were on user's shoes. Accelerometers were on the upper body, hip, and leg, which translated to 133 columns in the raw dataset.

In-house data. In order to experiment in a more noisy setting, we collected our data on human motion performing different activities that may be confounded by environments factors. It is made publicly available, serving as extra part of our contribution. Unlike data collected in a laboratory setting, less rules were given to subjects such as the amount of time they want to rest and the activity intensity. Also, the set of contexts to be detected was tried to be more realistic. On the other hand the data is much noisier than previous ones. This dataset has 3 contexts with corresponding movements (1) outdoors: Crawling, Jogging, Riding Bike, Sprinting, Walking, Walking with One Shoe (simulated limping), Walking with Weight in Arms, Walking with Weight on Back. (2)

*Reprinted with permission from "Uncertainty Quantification for Deep Context-Aware Mobile Activity Recognition and Unknown Context Discovery" by Huo, Zepeng and PakBin, Arash and Chen, Xiaohan and Hurley, Nathan and Yuan, Ye and Qian, Xiaoning and Wang, Zhangyang and Huang, Shuai and Mortazavi, Bobak, 2020. AISTATS, PMLR 108:3894-3904, Copyright 2020 by PMLR. URL: http://proceedings.mlr.press/v108/huo20a/huo20a.pdf.

Table 4.1: In house data description

Collected contexts	indoor	outdoor	in-vehicle
# of subjects participated	20	20	20
# of phone employed	3	3	3
# of unique activities	10	10	10
list of unique activities	[watching TV etc.]	[sprinting [etc.]	[driving etc.]

Movements that happen indoors: Escalator Up, Escalator Down, Elevator Up, Elevator Down, Lying Down, Sitting, Stairs Down, Stairs Up, Standing, Walking, Walk with One Shoe (simulated limping), Cooking, Dancing, Eating, Reading, Sleeping, Talking on phone, Talking to Another Person, Using PC, and (3) movements that happen outdoors but in vehicles: Driving Car, Riding Car, Riding Bus, Reading, Device Usage. The movements, however, are not unique to each context, and the position of the phone may change through usage. Data was collected on 20 people, each doing 32 activities while having 3 phones, one in hand, one in the pocket and one in the backpack. The applications used on the phones for data collection was *Readisens* [39]. The sensors used are: acceleration (in three axes), altitude, compass, gyroscope (in three axes), GPS information (latitude, longitude), screen time information, and phone speed. This study was reviewed and approved by the Texas A&M Institutional Review Board (IRB # 2018-210D). The data has been made available for public use.[*]

4.2 Implementation

In UCI dataset, all the body-worn sensors were fed into the network. They were concatenated in each window. Time series data were divided into non-overlapping segments of 1 second (30 samples). We used a five-fold cross-validation to evaluate our models with testing accuracy and micro F score as performance metrics. Following [40], in our implementation, we used 3 convolutional layers followed by 2 fully connected layers for both α and β networks. Note that the only difference these networks had in their architecture was the number of their outputs. Networks were

[*]RealActivity: https://github.com/Erakhsha/RealActivity

trained using stochastic gradient descent with an initial learning rate of 0.001 and a momentum of 0.9, which provided the best results in cross-validation.

Pre-training. Training in neural networks is an stochastic process and since their optimization problem is very non-convex and they find (usually good) local maxima, initialization plays a major role in their training. Our $alpha$-$beta$ framework is no exception. Specially because of its structure, if not initialized well, the model collapses into only selecting one β network without utilizing other networks, defeating the purpose of having several β networks. This phenomena has been observed in several cases and other authors have reported that too [23]. To solve this problem, we have come up with our own way of pretraining the α network so that it would have a warm start. We have compared such results with regularization where the network output is compared to when we regularize alpha networks to have high entropy outputs, which is a bad thing. As we will see, pretraining has shown to be more effective. Pre-training is an important stage in the α-β network training. Pretraining proves useful in finding subgroups of data as well as in yielding a better performance. We used the idea presented in [41] to cluster the activity data. In detail, a base network was trained with sensor readings as input and activities as output. Next, the CNN segment of the network was used to embed the sensor readings into the features which have proven to be descriptive of the input data [42, 43]. Subsequently, we used K-means to cluster the input into a fixed number of clusters. Finally, we trained our α network to learn the mapping between sensor readings of activity to clusters.

4.3 Test Cases

In this section we discuss different scenarios and test cases under which we have tested our methodology. The results presented are as follows:

- Unsupervised Context Detection using α-β network
  - Results on UCI OPPORTUNITY dataset
  - Results on In-house dataset
- Opportunistic sensor selection using α-β network in environments where additional data is

Table 4.2: Accuracy (F score) for the α-β network and baseline which is a β network with the same number of parameters for each specific number of clusters. For 1 cluster both the models are the same.

Clusters	2	3	4
α-β	**0.89(0.90)**	**0.89(0.90)**	**0.91(0.91)**
baseline	0.81(0.81)	0.84(0.84)	0.84(0.83)

Clusters	5	6	7
α-β	**0.92(0.91)**	**0.91(0.92)**	**0.96(0.96)**
baseline	0.83(0.82)	0.86(0.86)	0.85(0.85)

Clusters	8	9	10
α-β	**0.96(0.96)**	**0.96(0.96)**	**0.95(0.95)**
baseline	0.84(0.84)	0.86(0.86)	0.86(0.86)

noisy/costly to access

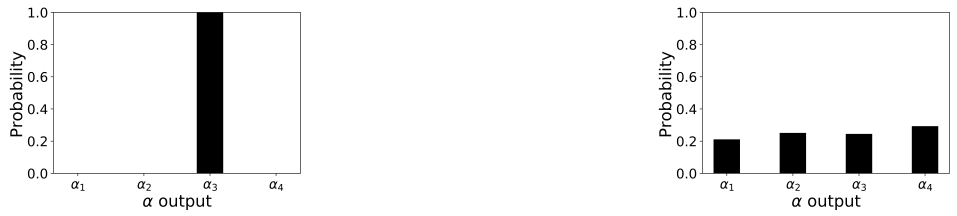
- Results on UCI OPPORTUNITY dataset

We now begin discussing each of the results in its own section.

4.3.1 Unsupervised Context Detection

4.3.1.1 UCI OPPORTUNITY Dataset

As our first case, we perform activity recognition in the UCI OPPORTUNITY dataset using our proposed α-β network compared to baselines. For k contexts, a baseline is a single β network which has $k + 1$ wider hidden layers in order to have the number of parameters equal to the α-β output (similar size). This is done to make a comparison to a network with relatively the same capacity but with a conventional structure. Table 4.2 cmakes a comparison between accuracies of the α-β network and the baseline for predicting activities in the UCI OPPORTUNITY dataset. The testing accuracies are averaged among cross validation instances. Table 4.2 shows that the mixture of the context-specific neural networks improved on the accuracy of the baseline from 86% to 96% when using nine contexts. Thus, our context-aware α-β network was able to find subgroups in the data in an unsupervised manner with different numbers of contexts. This fact was reflected in the



(a) Without pre-training (b) With pre-training

Figure 4.1: Average of α network output in testing without (a) and with (b) pre-training. The model collapses into a single network in the former. This shows the effectiveness of pre-training in making sure that the α network finds subgroups in the data and hence can take advantage of context-aware recognition.

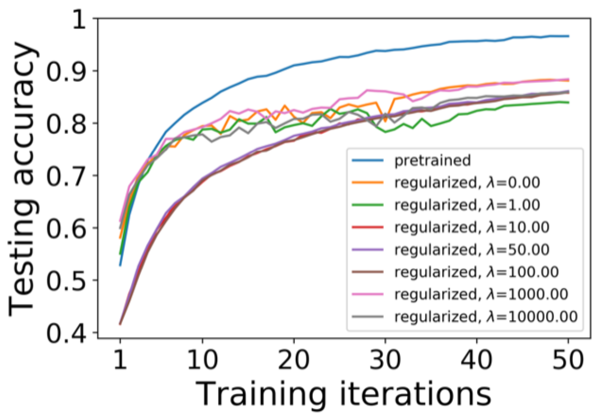


Figure 4.2: Comparison between pre-training and regularization.

accuracy boost due to the subgroup modeling by different β networks.

The α-β network is equipped with pre-training. Figure 4.1 shows that pre-training is of paramount importance and without it the model model would collapse into selecting just one network as discussed earlier. The effect of pre-training can also be seen in the network performance. Figure 4.2 shows that pre-training is much more effective than regularization since regularization just penalizes single network usage but does not use any knowledge about the data in the process. We took the best number of contexts in terms of accuracy and F-score, 9, and tried to train networks with the same number of contexts but without pre-training and by using only regulariza-

Table 4.3: In house data, Accuracy and F score for the α-β network and baseline which is a β network with the same number of parameters for each specific number of clusters. For 1 cluster both the models are the same.

Performance Measures	Accuracy	F score
α-β Network	0.84	0.86
baseline	0.80	0.80

tion with different regularization coefficients. Without proper initialization, the α-β network drops from 96% to 87% in accuracy and 0.96 to 0.87 in F-score. This is 4% less than the performance of an identical α-β network with proper initialization in Table 4.2. This is roughly equal to the performance achieved by a single β network (refer to baseline results in Table 4.2).

4.3.1.2 In-house Dataset

We have performed the same test done on UCI OPPORTUNITY dataset on our In-house dataset to see the assess the performance of our framework in a noisier environment Table 4.3 shows the results we obtained. It can be seen that our method still outperforms the baseline (84% vs. 80%). Note that our model did drop accuracy compared to the average results from Table 4.2 (84.3% (std 0.015)), showing the dataset is noisier and getting the same accuracy is harder. Since the number of context was known to be exactly 3 in this dataset, we just made a comparison with that number of contexts.

4.3.2 Opportunistic Sensor Selection

In this part we talk about another use case of α-β network. In some scenarios, accessing more sensors is expensive or even if accessing them is not necessarily expensive, there might be noise in their measurements so accessing as much sensors as possible is not necessarily a good action to take. In such cases, opportunistically selecting sensor sets would be an effective way of handling the situation. In order to make this scenario happen, we make the β selection a hard assignment, meaning we only select one single β in each iteration, otherwise it would be defeat the purpose of opportunistic label selection. We discuss several study cases here:

Table 4.4: List of sensors used by β networks in the case studies. In case studies 1 and 2 the α networks has access to the watch sensors. In case study 3 the α networks has access to watch and phone sensors. β networks have access to mutually exclusive nearable sensors.

Name	Sensors list
Watch	Wrist Accelerometer and Arm Gyroscope
Phone	Hip Accelerometer
Expert 1	Sensors on Door 1, Door 2, Fridge
Expert 2	Sensors on Drawer 1, Drawer 2, Drawer 3
Expert 3	Dishwasher, Chair, Objects on Table

- **Sensor Requesting**: This is the first use case where we have access to a limited part of the data and accessing more would be costly. In this scenario, we have one single α network which has access to the restricted set and several β networks that have access to mutually exclusive sensors in addition to the base sensor set α has access to.
- **Intelligent Sensor Selection**: In the second use case, cost of accessing more data is low but the data is noisy so we still want to minimize the number of sensors we are accessing. In this scenario, α makes a prediction based on the inputs and still gives a distribution over β networks. Based on the uncertainty it has about the output, it decides whether it needs to ask any of the β networks about the final output.
- **Augmented Selector Network**: This is identical to the *Intelligent Sensor Selection* test case and the only difference would be more sensors provided to the α network.

We have done the test cases on the UCI OPPORTUNITY dataset. Before getting started on the results, let us briefly talk about the sensor groups used in this study. Figure 4.3 shows the breakdown of different sensors in the dataset. As can be seen, miscellaneous sensors are the biggest group and there is less organization in them. But doors and drawers are more organized. We have aggregated different sensors in different groups which can be seen in Table 4.4.

Now we discuss the test cases one by one and present their results:

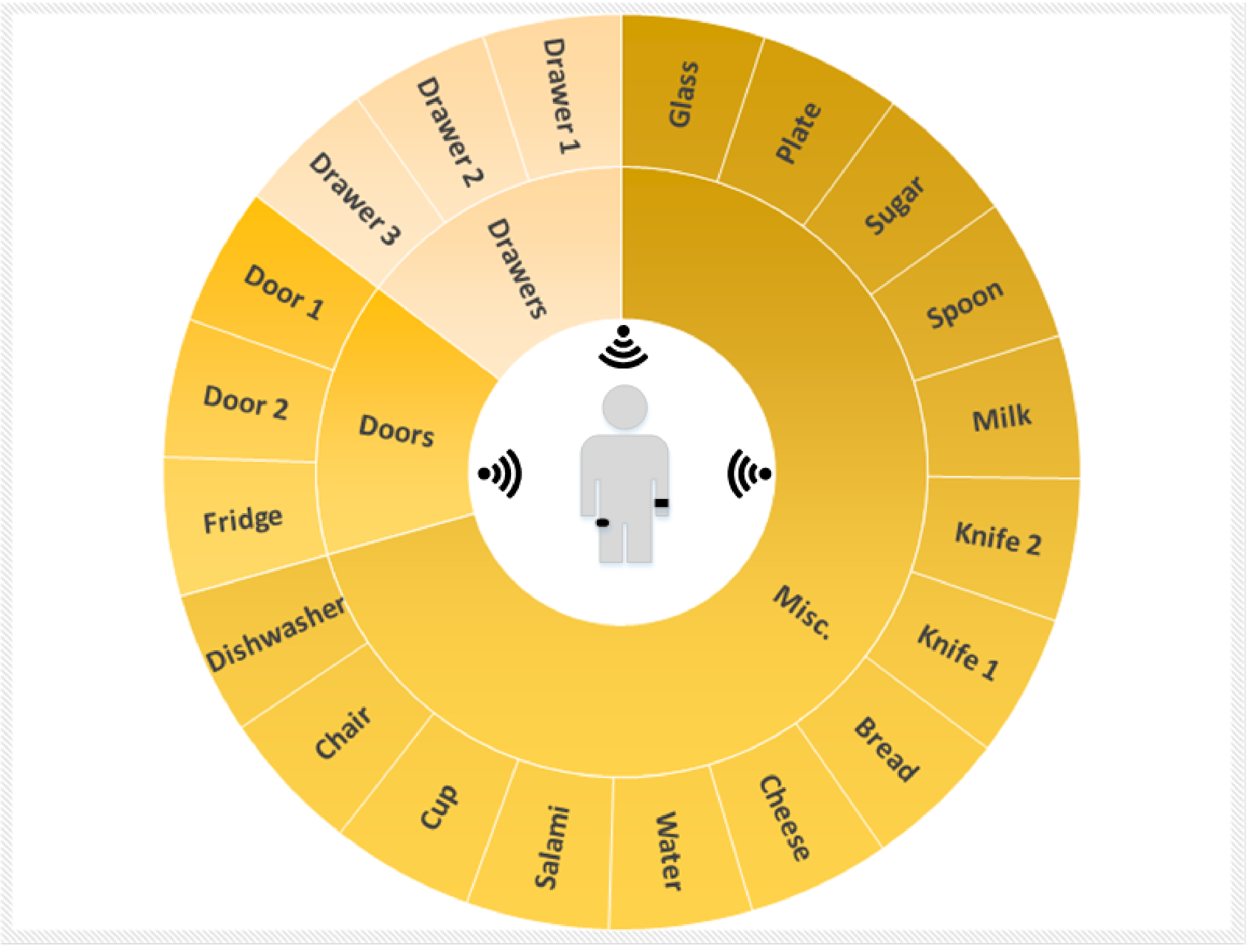


Figure 4.3: Opportunity selected sensors and IoT sensors broken up by category. Notice that the Misc. sensors represent a greater number but that the data provided by all the sensors are roughly evenly distributed.

Table 4.5: Case study 1: Comparison between accuracies and F1 scores of α-β network and the baselines, a single β network. The recognition task was activity recognition in the OPPORTUNITY dataset.

Model	Sensors	Accuracy	F1 Score	F1 Gain
β network	Watch	0.24 (0.03)	0.32 (0.05)	-
β network	Watch & Expert 1	0.44 (0.08)	0.43 (0.02)	34.4%
β network	Watch & Expert 2	0.39 (0.06)	0.39 (0.04)	21.9%
β network	Watch & Expert 3	0.26 (0.01)	0.37 (0.04)	15.6%
α-β network	Watch & Experts (All)	0.44 (0.05)	0.40 (0.06)	25.0%

Table 4.6: Case study 2: Comparison between accuracies and F1 scores of α-β and the baselines, a single β network, when noise is present in the data. The recognition task was activity recognition in the Opportunity dataset.

Model	Sensors	Accuracy	F1 Score	F1 Gain
β network	Watch	0.26 (0.03)	0.29 (0.06)	-
β network	Watch & Expert 1	0.46 (0.04)	0.42 (0.02)	44.8%
β network	Watch & Expert 2	0.36 (0.09)	0.34 (0.08)	17.2%
β network	Watch & Expert 3	0.35 (0.08)	0.31 (0.08)	6.9%
α-β network	Watch & Experts (All)	0.42 (0.04)	0.38 (0.06)	31.0%

4.3.2.1 Sensor Requesting

In this scenario, accessing more data is expensive. α network has access to a limited number of sensors and in order to make a prediction, needs to select one β network among several of them. Each of the β networks have access to mutually exclusive sensor sets in addition to the original set α network has access to. The results of such comparison can be seen in Table 4.5. As we can see, a single α network fails to capture the complexities in the data and the need for additional information through more sensors is needed. This also shows that in unconstrained environments using only watch is not sufficient. The results in general are promising since α-β network outperforms the baselines in most cases.

Table 4.7: As the noise of the dataset increases, the α network becomes more likely to rely on itself than to utilize separate β networks. These α-β networks incorporate signals from the Watch and from all Experts.

Dataset	Accuracy	F1 Score	Implicit Self-Selection Rate
Clean	0.43 (0.03)	0.39 (0.05)	3.62%
Noisy	0.41 (0.06)	0.36 (0.07)	6.15%

Table 4.8: Multitask learning. As the noise of the dataset increases, the α network becomes more likely to rely on itself than to utilize separate β networks. These α-β networks incorporate signals from the Watch and from all Experts.

Dataset	Accuracy	F1 Score	Implicit Self-Selection Rate
Clean	0.43 (0.03)	0.39 (0.05)	3.62%
Noisy	0.41 (0.06)	0.36 (0.07)	6.15%

4.3.2.2 Intelligent Sensor Selection

In this scenario, accessing more data is not costly but data is noisy so adding more data does not necessarily make α network still has access to a restricted set of sensors. In order to implement noisiness, in each time window we randomly add noise to various nearable sensors. In this approach, each object is given a chance to have each of three types of noise added. All channels from a given object's sensors (i.e. all acceleration and gyroscope axes of a particular object) are subjected to the same type of noise. The chance of each of these are independent, with the possibility for multiple types of noise to be added. The types of noise are a) zeroing the signal for the entire window, b) adding uniform random noise of the within the amplitude of the signal, and c) adding Gaussian noise with 0 mean and σ equal to the amplitude of the signal. The first type of noise represents a malfunctioning sensor that is inappropriately offline or unable to connect. The second noise represents multiple users interacting within the environment, and activating sensors with signals that are not of interest to the system. The final noise represents random noise from

Table 4.9: Case study 3: Comparison between accuracies and F1 scores of α-β network and the baselines, a single β network, when implemented with a α network that includes both a smartwatch and a smartphone.

Model	Sensors	Accuracy	F1 Score	F1 Gain
β network	Watch & Phone	0.23 (0.02)	0.29 (0.06)	-
β network	Watch, Phone, & Expert 1	0.48 (0.01)	0.44 (0.01)	51.7%
β network	Watch, Phone, & Expert 2	0.42 (0.06)	0.39 (0.06)	34.5%
β network	Watch, Phone, & Expert 3	0.29 (0.06)	0.37 (0.02)	27.6%
α-β network	Watch, Phone, & Experts (All)	0.38 (0.09)	0.33 (0.12)	13.8%

potential communication cross-talk in a highly instrumented environment. The difference here is that α network makes a prediction itself and if it finds it necessary, it accesses a β network to make the final call. The way network is trained is multi-task learning in which the loss function is an addition of the loss function at the β network end and at the α network end. The *necessity* is computed based on the entropy in the output. The results can be seen in Table 4.6. Even though the results seem to be lower compared to 4.5, the performance boost is larger. So the importance of such framework is underscored in noisier environments. Another interesting result lies in Table 4.8 which shows that with more noise the α network puts more emphasis on choosing itself rather than choosing any of the β networks. It seems like the model accounts for the lack of reliability.

4.3.2.3 Augmented Selector Network

In the third case study we repeat the second case study *Intelligent Sensor Selection* with a change which is adding more sensors to the α network. The results are shown in Table 4.9. Table 4.9 compares the selector-expert network featuring this larger selector network with the baselines of single expert networks trained using different sensors in Table I. The results here show that even though the baseline selector network is a poorer raw predictor than the baseline selector network in earlier studies, this baseline coupled with the expert net works give improved performance. This result underscores a limitation in our design where it appears that the baseline model which always utilizes Expert network 1 (accelerometers on doors) outperforms any other system, including the opportunistic selector-expert network. As this dataset is limited in environmental size, it stands to reason that certain expert networks have signals containing information that allows for greater overall predictive power than others. Applying this model to a dataset with more expert networks spread out more in space would likely remove this increased performance that this particular expert network has.

5. SUMMARY AND CONCLUSIONS

In this work, we developed a novel α-β networkk in tackling a range of realistic situations where context information was unknown but critical for enhanced situation awareness and human activity recognition. Experiments on a real-world datasets showed that this framework led to superior performances by its efficacy and efficiency in extracting context information and consequently achieving superior results.

This work can be used as a foundation for a more comprehensive analysis of contextual discovery in a variety of modeling efforts in the future. Multiple levels of contextual information can be gathered and learned from the system, and understanding those in a truly unsupervised setting can enhance a number of recognition tasks and create a flexible and more realistic ontology for how to define context in human activity recognition tasks, rather than relying on high-level but general descriptions of context or too restrictive pre-defined contexts. In addition, other sources of uncertainty in prediction which result from data of new distributions or noisy data from old distributions should be considered as well.

REFERENCES

[1] M. I. Jordan and R. A. Jacobs, “Hierarchical mixtures of experts and the em algorithm,” *Neural computation*, vol. 6, no. 2, pp. 181–214, 1994.

[2] Z. Huo, A. PakBin, X. Chen, N. Hurley, Y. Yuan, X. Qian, Z. Wang, S. Huang, and B. Mortazavi, “Uncertainty quantification for deep context-aware mobile activity recognition and unknown context discovery,” vol. 108 of *Proceedings of Machine Learning Research*, (Online), pp. 3894–3904, PMLR, 26–28 Aug 2020.

[3] G. Schiboni and O. Amft, “Automatic dietary monitoring using wearable accessories,” in *Seamless healthcare monitoring*, pp. 369–412, Springer, 2018.

[4] J. C. Rawstorn, N. Gant, A. Meads, I. Warren, and R. Maddison, “Remotely delivered exercise-based cardiac rehabilitation: design and content development of a novel mhealth platform,” *JMIR mHealth and uHealth*, vol. 4, no. 2, p. e57, 2016.

[5] E. Jovanov, “Preliminary analysis of the use of smartwatches for longitudinal health monitoring,” in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 865–868, IEEE, 2015.

[6] T. D. Gunter and N. P. Terry, “The emergence of national electronic health record architectures in the united states and australia: models, costs, and questions,” *Journal of medical Internet research*, vol. 7, no. 1, p. e3, 2005.

[7] O. B. Sezer, E. Dogdu, and A. M. Ozbayoglu, “Context-aware computing, learning, and big data in internet of things: a survey,” *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 1–27, 2017.

[8] B. J. Mortazavi, M. Pourhomayoun, G. Alsheikh, N. Alshurafa, S. I. Lee, and M. Sarrafzadeh, “Determining the single best axis for exercise repetition recognition and counting on smart-

watches," in *2014 11th International Conference on Wearable and Implantable Body Sensor Networks*, pp. 33–38, IEEE, 2014.

[9] R. Solis, A. Pakbin, A. Akbari, B. J. Mortazavi, and R. Jafari, "A human-centered wearable sensing platform with intelligent automated data annotation capabilities," in *Proceedings of the International Conference on Internet of Things Design and Implementation*, pp. 255–260, ACM, 2019.

[10] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE communications surveys & tutorials*, vol. 16, no. 1, pp. 414–454, 2013.

[11] D. Guan, W. Yuan, S. J. Cho, A. Gavrilov, Y.-K. Lee, and S. Lee, "Devising a context selection-based reasoning engine for context-aware ubiquitous computing middleware," in *International Conference on Ubiquitous Intelligence and Computing*, pp. 849–857, Springer, 2007.

[12] A. Bikakis, T. Patkos, G. Antoniou, and D. Plexousakis, "A survey of semantics-based approaches for context reasoning in ambient intelligence," in *European Conference on Ambient Intelligence*, pp. 14–23, Springer, 2007.

[13] A. K. Dey, "Understanding and using context," *Personal and ubiquitous computing*, vol. 5, no. 1, pp. 4–7, 2001.

[14] J. Y. Xu, H.-I. Chang, C. Chien, W. J. Kaiser, and G. J. Pottie, "Context-driven, prescription-based personal activity classification: methodology, architecture, and end-to-end implementation," *IEEE journal of biomedical and health informatics*, vol. 18, no. 3, pp. 1015–1025, 2013.

[15] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni, "A survey of context modelling and reasoning techniques," *Pervasive and Mobile Computing*, vol. 6, no. 2, pp. 161–180, 2010.

[16] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, “Activity recognition from accelerometer data,” in *Aaai*, vol. 5, pp. 1541–1546, 2005.

[17] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, “Deep learning for sensor-based activity recognition: A survey,” *Pattern Recognition Letters*, vol. 119, pp. 3–11, 2019.

[18] S. E. Yuksel, J. N. Wilson, and P. D. Gader, “Twenty years of mixture of experts,” *IEEE transactions on neural networks and learning systems*, vol. 23, no. 8, pp. 1177–1193, 2012.

[19] A. Miech, I. Laptev, and J. Sivic, “Learnable pooling with context gating for video classification,” *arXiv preprint arXiv:1706.06905*, 2017.

[20] M. Courbariaux, C. Ambroise, C. Dalmasso, M. Szafranski, M. Consortium, *et al.*, “A mixture model with logistic weights for disease subtyping with integrated genome association study,” 2018.

[21] M. I. Jordan and L. Xu, “Convergence results for the em approach to mixtures of experts architectures,” *Neural networks*, vol. 8, no. 9, pp. 1409–1431, 1995.

[22] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the em algorithm,” *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

[23] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, “Outrageously large neural networks: The sparsely-gated mixture-of-experts layer,” *arXiv preprint arXiv:1701.06538*, 2017.

[24] X. Li, M. Eckert, J.-F. Martinez, and G. Rubio, “Context aware middleware architectures: survey and challenges,” *Sensors*, vol. 15, no. 8, pp. 20570–20607, 2015.

[25] T. Strang and C. Linnhoff-Popien, “A context modeling survey,” in *Workshop Proceedings*, 2004.

[26] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, “Mimic-iii, a freely accessible critical care database,” *Scientific data*, vol. 3, p. 160035, 2016.

[27] P. Nurmi and P. Floréen, “Reasoning in context-aware systems,” *Helsinki Institute for Information Technology, Position paper*, 2004.

[28] K. Altun and B. Barshan, “Human activity recognition using inertial/magnetic sensor units,” in *International workshop on human behavior understanding*, pp. 38–51, Springer, 2010.

[29] R. Mayrhofer, H. Radi, and A. Ferscha, *Recognizing and predicting context by learning from user behavior*. na, 2003.

[30] A. Malinin and M. Gales, “Predictive uncertainty estimation via prior networks,” in *Advances in Neural Information Processing Systems*, pp. 7047–7058, 2018.

[31] D. Hendrycks and K. Gimpel, “A baseline for detecting misclassified and out-of-distribution examples in neural networks,” *arXiv preprint arXiv:1610.02136*, 2016.

[32] M. Sensoy, L. Kaplan, and M. Kandemir, “Evidential deep learning to quantify classification uncertainty,” in *Advances in Neural Information Processing Systems*, pp. 3179–3189, 2018.

[33] S. Liang, Y. Li, and R. Srikant, “Enhancing the reliability of out-of-distribution image detection in neural networks,” *arXiv preprint arXiv:1706.02690*, 2017.

[34] K. Lee, K. Lee, H. Lee, and J. Shin, “A simple unified framework for detecting out-of-distribution samples and adversarial attacks,” in *Advances in Neural Information Processing Systems*, pp. 7167–7177, 2018.

[35] R. D. Levine and M. Tribus, “Maximum entropy formalism,” in *Maximum Entropy Formalism Conference (1978: Massachusetts Institute of Technology)*, Mit Press, 1979.

[36] T. Jaakkola, M. Meila, and T. Jebara, “Maximum entropy discrimination,” in *Advances in neural information processing systems*, pp. 470–476, 2000.

[37] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[38] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, *et al.*, "Collecting complex activity datasets in highly rich networked sensor environments," in *Networked Sensing Systems (INSS), 2010 Seventh International Conference on*, pp. 233–240, IEEE, 2010.

[39] Lockheed Martin, "Anonymized phone usage data collector," 2019.

[40] J. Yang, M. N. Nguyen, P. P. San, X. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition.," in *Ijcai*, vol. 15, pp. 3995–4001, 2015.

[41] J. Guérin, O. Gibaru, S. Thiery, and E. Nyiri, "Cnn features are also great at unsupervised classification," *arXiv preprint arXiv:1707.01700*, 2017.

[42] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 806–813, 2014.

[43] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *International conference on machine learning*, pp. 647–655, 2014.