Enabling Resource-Efficient Edge Intelligence with Compressive Sensing-Based Deep Learning

Alina Machidon
alina.machidon@fri.uni-lj.si
University of Ljubljana
Ljubljana, Slovenia

Veljko Pejović
veljko.pejovic@fri.uni-lj.si
University of Ljubljana
Ljubljana, Slovenia

ABSTRACT

Billions of sensor-enabled computing devices open tremendous opportunities for AI-powered context-aware services. Yet, democratizing AI so that heterogeneous devices can enjoy deep learning (DL) revolution requires significant reduction in computing and energy burden posed by DL models. Inspired by the concept of compressive sensing (CS), and guided by the observation that reduced sampling rates often suffice for successful classification, we devise an adaptive CS-DL pipeline. Our approach dynamically adjusts the sensing rate according to input properties and performs classification through an input-flexible DL model, demonstrating classification accuracy rates on par with uncompressed models while using up to 46% less battery energy.

CCS CONCEPTS

• **Human-centered computing → Ubiquitous and mobile computing systems and tools**; **Ubiquitous computing**; • **Computing methodologies → Neural networks**.

KEYWORDS

neural networks, compressive sensing, human activity recognition

ACM Reference Format:
Alina Machidon and Veljko Pejović. 2022. Enabling Resource-Efficient Edge Intelligence with Compressive Sensing-Based Deep Learning. In *19th ACM International Conference on Computing Frontiers (CF'22), May 17–19, 2022, Torino, Italy*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3528416.3530230

1 INTRODUCTION

Rapid advances in smartphone device manufacturing resulted in an increase and diversification in the range and the level of complexity of sensors that these devices are equipped with. This further lead to an explosion of data collection. However, in mobile computing, sensing may have very high energy costs and can quickly deplete the mobile's battery. Moreover, intensive and sometimes unnecessary sampling also burdens the computational, storage, and wireless transmission capabilities of a device. Given that there is a need for more judicious energy-efficient sampling strategies producing only the minimum amount of data necessary to complete a computing task, a promising approach is to employ a smart, dynamic sensor sampling strategy, that uses the minimum amount of samples to obtain the desired quality of service.

Inspired by Compressive Sensing (CS), a concept that allows successful reconstruction of sparse signals sampled at rates far below those prescribed by the Nyquist theorem, in this paper we propose a novel rate-adaptive mobile sensing and signal classification pipeline. The key novelties of our pipeline are: i) the integration of the sampling layer with a deep learning (DL) network that enables processing (e.g. signal class prediction) of samples taken at different rates, and ii) context awareness that intelligently adapts the sampling rate according to the nature of the signal at the input.

Our approach is based on the observation that the sampling rate required to achieve high classification accuracy strongly correlates with the input's "difficulty". By this term we refer to all possible predictors of the degree to which the inherent characteristics of the input represent a demanding classification problem for the neural network. For example, in case of human activity recognition (HAR), a domain we use as a working example in this paper, we analyze a commonly used dataset [2] and conclude that certain human activities may be sampled with an on-body sensor and accurately classified when using only 20% of the sampling rate commonly employed in smartphones, while other more complex activities require a higher, more resource expensive, sampling rate. We then uncover the relationship between the input data "difficulty" and the inference confidence of the softmax layer of our integrated DL network, and devise an algorithm for dynamic adaptation of the sampling rate according to the input's properties.

We implement and evaluate our solution on an edge computing device and experimentally show the benefits of our context-aware sampling rate adaptation. We assess the savings enabled by our solution and show that up to 46% energy savings can be achieved, without a degradation of inference accuracy.

To summarize, the contributions of our work include:

- We develop a rate-flexible mobile sensing and inference pipeline merging compressive sensing and deep learning;
- We devise a sampling rate tuning algorithm that aims to satisfy the given quality of service requirements with the least amount of sampling (thus, energy);
- We evaluate our approach on a well-established human activity recognition dataset and demonstrate that it achieves the accuracy on par with the full-fledged uncompressed processing pipeline using on average only 30% of the sensor samples;
- We experimentally validate our solution on an edge computing platform with live acquisition of inertial data and show

it provides significant energy savings with no inference accuracy loss.

The reduction in energy usage brought by our solution could be integral for enabling continuous sensing to resource-frugal devices, especially in areas where untethered operation is crucial, such as with on-body monitoring in assisted living scenarios, or in wildlife tracking. Furthermore, rather than merely switching the embedded AI on or off, our solution enables dynamic balancing between the achieved quality of service and resource usage of AI on ubiquitous computing devices. Finally, being rather generic, our solution opens space for further exploration of sampling-DL optimization space, for instance through dynamic neural network weight quantization [11] and pruning [12].

The code for our pipeline is publicly available at: https://gitlab.fri.uni-lj.si/lrk/adaptive-cs-dl.

2 PRELIMINARIES

2.1 Joint Compressive Sensing & Deep Learning for Edge Intelligence

Compressive Sensing (CS) [5, 7] revolutionized the signal acquisition by proving that it is possible to successfully reconstruct a signal acquired at sampling rates far lower that thought before. The two conditions that need to be meet for accurately restoring a compressed sensed signal are: *signal sparsity* – a property that in a certain domain most of the signal's information is concentrated along a few dimensions, and *incoherence* – a property of low correlation between the sparsity domain and the sampling process.

CS states that a K-sparse signal vector $x \in \mathcal{R}^N$ (i.e. a signal with only K non-zero components), can be accurately recovered from the undersampled measurements $y \in \mathcal{R}^M$, taken with a known *measurement matrix* A, that satisfies the restricted isometry principle (i.e. does not distort the measured signal, thus enabling accurate reconstruction):

$$y = Ax$$

In general, the above equation is under-defined ($M \ll N$), having an infinite number of solutions for the original signal x. CS seeks to solve the above equation by finding the sparsest signal that produces the measurement y, through a l_0-norm minimization approach:

$$\begin{aligned} \text{minimize} \quad & \|x\|_0 \\ \text{subject to} \quad & Ax = y \end{aligned} \tag{1}$$

where $\|\cdot\|_0$ is the l_0-norm and denotes the number of non-zero components in x. Since Equation 1 leads to an intractable combinatorial search problem, the l_1-norm is used instead, as a proxy for the l_0-norm, and the NP-hard problem becomes solvable, using convex optimization or greedy algorithms.

More recently, neural networks' ability to solve convex optimization problems was also exploited in the context of CS. The main idea behind using DL for CS is to train a deep neural network to model the non-linear reconstruction process. Deep learning brought tremendous improvements to CS and a variety of CS-DL implementations emerged, ranging from approaches based on unfolding the CS iterative algorithms through the various layers of a neural network [42], to solutions that learn, independently from traditional CS algorithms, to reconstruct the original data, with [25] or without [14] jointly learning the measurement matrix.

One of the most common recent uses of sampled (and then reconstructed) signals is high-level inference, i.e. diagnosing diseases from magnetic resonance images, inferring physical activity from body-worn accelerometers, and others. Consequently, direct merging of CS concepts with DL-based classification appeared [22, 36]. These so-called "compressed learning" approaches, are based on training a DL model to guide both the sampling and the classification together, bypassing signal reconstruction[1]. By skipping the expensive and often unnecessary phase, consistent computational speedups can be achieved [26]. In addition, learning directly in the compressed domain allows very high compression rates that would otherwise be prohibitive for the reconstruction-based approaches [21]. In turn, these higher compression rates foster CS-DL implementations on resource-constrained devices, where such inference tasks were previously not supported.

2.2 Sparse Sensing and Classification

Mathematically, compressive sensing relies on signal sparsity, and the number of measurements M that need to be taken to guarantee accurate reconstruction depends on the sparsity level K of the signal ($M \approx Klog(N/K)$). In other words, the sparser the signal (in a certain domain), the fewer measurements are required. In practice however, assessing the sparsity level for a given signal is a challenging task. Furthermore, the estimation of the sparsity for time-varying signals is usually performed on the reconstructed signal, which for reconstruction-free approaches such as the one developed in this paper is missing. Intuitively however, the accuracy of compressed learning will reflect the appropriateness of a certain sparsely sampled input – signals with higher K would require more samples for correct classification.

The above observation is a cornerstone of our method. To estimate the number of measurements that will be sufficient for classification at each time step we utilize a historical data test set and compute the minimum required sampling rate to get a correct prediction, further grouping the results by the classification label.

As an illustrative example we show how the minimum sampling rate[2] needed for correct classification of physical activity from on-body accelerometer and gyroscope samples (dataset [2] described in Section 3) processed thorough a CS-DL pipeline (detailed in the following section) varies with the nature of the conducted activity. The results in Figure 1 show that some activities can almost always be successfully inferred even with a low minimum sampling rate: e.g. lying, an activity for which 95% of the instances are correctly classified with the lowest sampling rate (10%). At the same time however, there are classes for which this assertion does not hold: sitting requires the highest sampling rate (100%) for 10% of the samples pertaining to this class, 20% of the samples require sampling rate between 80% and 40%, while for 70% of the instances, the lowest sampling rate (10%) is sufficient to get a correct classification.

[1]While these approaches often get presented within the CS community, we note that without a full signal reconstruction it is difficult to claim that the resulting sampling is indeed complying with the CS requirements. Therefore, in this paper we term our approach as "CS-inspired".

[2]Since we are using an existing dataset collected at a uniform sampling rate, what we term an "N% sampling rate" is actually a subset containing N% of the original samples.

Other activities such as walking are even more difficult to map to a certain sampling rate (45% of the samples are distributed over a large interval of sampling rates: 100% – 20%).

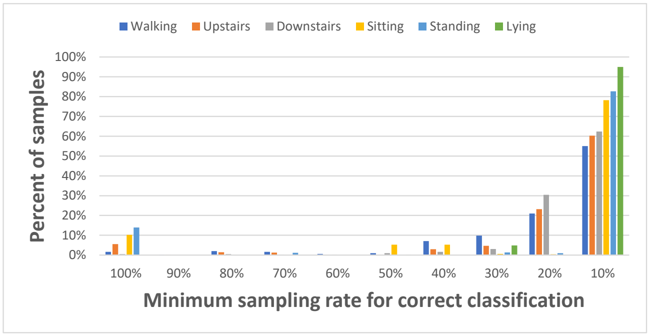

Figure 1: The distribution of samples in each class and the minimum sampling rate required for obtaining a correct classification in the UCI HAR dataset.

Our analysis shows that there is a correlation between the input's activity class label and a sufficient sampling rate for some activities though this correlation is stronger than for the others, for which additional factors must be considered. Consequently, we design an adaptation approach where the sampling rate selection is driven by *the individual difficulty of each input*, on the grounds that within the same class activity label, there are "difficult" inputs that require a higher sampling rate but also "easier" inputs for which even a lower sampling rate gives good results.

3 DYNAMIC COMPRESSIVE SENSING-INSPIRED NEURAL NETWORK ADAPTATION FRAMEWORK

In this section we develop an end-to-end CS-inspired DL inference pipeline. As a running example, we again use human activity recognition (HAR) from triaxial accelerometer and gyroscope data, of the UCI HAR dataset [2]. We opted for a small and computationally lightweight deep learning model that meets the limitations and constraints of the devices usually used for HAR, such as smartphones, smartwatches, and fitness wristbands. While supporting on-device physical inference, these devices are nevertheless highly constrained on the available memory, computational power, and battery energy. Thus, our goal is to have a network with a modest number of parameters and operations, yet powerful enough to give results comparable with those of other state-of-the-art methods.

Our proposed end-to-end deep learning solution for compressive learning is illustrated in Figure 2. Note that our approach can be seamlessly adapted to other types of network architectures as well. For sensing we opt for a dense layer followed by several convolutional layers responsible for the inference, i.e., activity recognition. In this manner, the sensing and the inference stages can be optimized simultaneously, but also detached and used separately, if needed, for instance in scenarios where the sensing is done on-device and the inference in the cloud.

The sensing layer is a key part of our solution and enables rate-adaptive sampling and processing. Dense layers at the input have

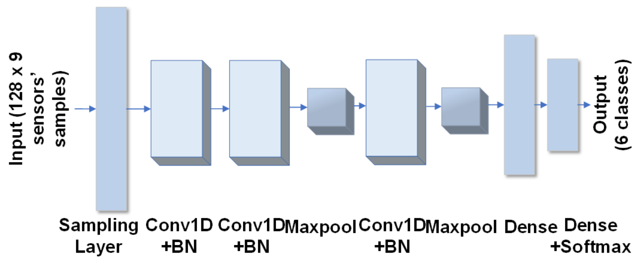

Figure 2: Simplified illustration of the proposed network's architecture.

been shown to perform well in CS-DL implementations [20, 29], yet, what is unique to our approach is that through our dense layer we model a data-driven measurement matrix A enabling "picking" of individual entries from the input time signal x. We mathematically modeled the individual time steps measurements by using a diagonal measurement matrix (Figure 3). This measurement process can further be implemented as a dense layer with a kernel constraint enforcing the training of only those weights that correspond to the diagonal values, while the rest are set to zero. In this manner, a sampling matrix for time domain data is learned. Using this sampling matrix, we can later design a sampling pattern that specifies at the sensor-level which samples must be read and which can be skipped, while maximizing the inference's accuracy (more details on this in Section 3.1).

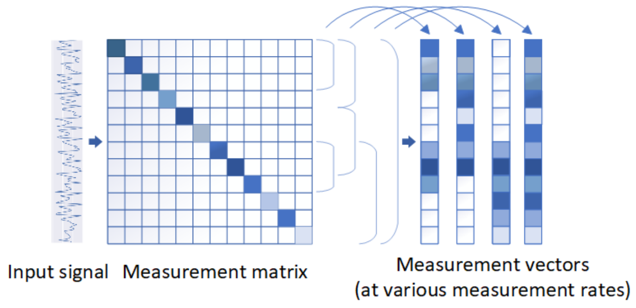

Figure 3: Proposed sampling scheme. The measurement matrix has non-zero values only on the main diagonal, while the rest of its entries are zero. By masking different sequences from the measurement matrix, measurement vectors of various sizes are obtained, representing signals acquired at various sampling rates.

The classification network's architecture is also inspired by a classic CNN model (three convolutions with max pooling, followed by a dense layer and a final softmax layer), that was successfully validated on time series data [6]. We added batch normalization to our architecture, since this operation is known to prevent the internal covariate shift across one mini-batch training of time series [16]. At the same time, having only 1.5 million parameters the proposed model is light, whereas MobileNet V1 [13], commonly used in mobile and embedded applications, has 4.2 million parameters.

We trained the network on the training subset of the publicly available UCI HAR dataset [2], with the categorical crossentropy loss, using the stochastic gradient descent (SGD) optimizer with a learning rate of 0.001, and a momentum of 0.7.

3.1 Enabling Sampling Rate Adaptivity

A major downside of the DL-based approaches for CS [14, 25] is the fact that once trained, a network is only valid for a single sampling rate, whereas for the optimal performance the measurement rates should vary with the changing context of use. As demonstrated in Section 2.2 there might not always be a need to sample at the same (maximal) rate – certain inputs might be "easier", enabling accurate classification even under lower sampling rates. Consequently, adaptive sampling rate could require less computational and energy resources for the same end result.

To achieve real-time adaptivity of the DL model, we need to train a model that is able to classify series of data samples acquired at different sampling rates. First, to simulate sparser temporal sampling scenarios, we developed a weight pruning strategy focused on pruning groups of consecutive weights from the sampling layer, corresponding to all the sensors' samples acquired at a certain time step (representing the three-axis total acceleration, three-axis body acceleration and three-axis angular velocity, thus nine values). Instead of pruning certain percentages of the weights with the lowest magnitude, we opted for pruning groups of nine consecutive weights (corresponding to samples acquired by both the accelerometer and the gyroscope at a certain time step), whose magnitudes sum scores the lowest. The number of groups that will be pruned corresponds to the sampling rate (i.e. more groups are pruned in case of lower sampling rates). This strategy of skipping time steps, however, leads to a faster degradation of the network's accuracy. Figure 4 illustrates the sensor sampling and group-based pruning strategy.

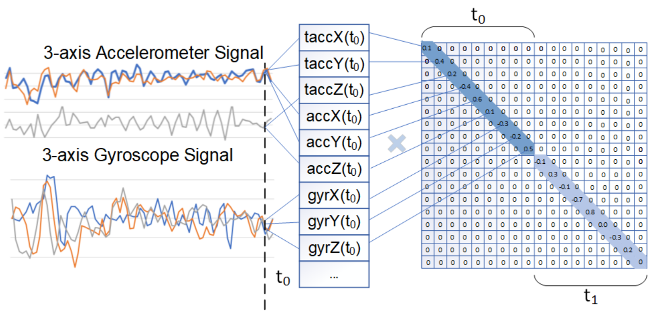

Figure 4: Sampling the tri-axial accelerometer and gyroscope signals with a measurement matrix trained to preserve sensor data pertaining to selected time steps only. Values at t_0 are sampled, pre-processed, and forwarded for processing in the DL pipeline, while values at t_1 are skipped.

To prevent the degradation of the network's performance with weight pruning in the sampling layer we adopt a refined training scheme. We split the initially trained network into two parts: the sensing part and the inference part. The sensing sub-network on which the previously described weight pruning strategy was applied is used to artificially generate outputs corresponding to 10 different sampling rates scenarios, based on the training input samples from the training set. The sampling rates range from 100% of the full scale sampling (50Hz in our HAR example) to 10% of it. Irrespective of the sampling rate, the output vectors are packed in the same format (see Figure 3). Next, using the generated (sub-)samples we further train the inference sub-network to classify inputs acquired with this range of sampling rates. This network refinement process enables real time runtime adaptivity of the neural network with a more graceful degradation in accuracy and without the need for re-training when switching the sampling rate at runtime.

3.2 Sampling Rate Adaptation Algorithm

With the mechanism for adjusting the sampling rate on-the-fly in place, we now turn to the algorithm for selecting the most appropriate rate at any given moment. We start from the hypothesis that *guiding the sampling rate based on the level of difficulty of the input* uses higher sampling rates only when necessary, and consequently reduces the overall computational costs without severely impacting the inference accuracy.

Assessing an input's difficulty for classification can be efficiently proxied by interpreting the probability scores a classifier assigns to each of the possible classes [4]. In a neural network, the reported probabilities from the final softmax layer can be used to estimate how confident the model was when making the prediction. A very confident model would predict one class with probability 1 and all the other classes with probability 0, while a model with very low confidence provides equal probabilities for all classes. A common way of capturing the uncertainty in the model's prediction is to evaluate the *entropy* of the classes' probabilities [33]. The entropy measures the uncertainty provided by the distribution of the classes probabilities:

$$H = -\sum_{k=1}^{L} P(y_k) \cdot \log(P(y_k)) \tag{2}$$

where y_k is the k-th element in the set of class labels L, and $P(y_k)$ is the probability that y_k is the correct label for that input.

Because the range of the entropy score depends on the number of classes, we normalize the entropy to:

$$\tilde{H} = -\sum_{k=1}^{L} P(y_k) \cdot \log(P(y_k))/\log(L) \tag{3}$$

We empirically show that the entropy of the classes' probabilities remains inversely correlated with the accuracy of the model. Intuitively, low entropy values correspond to low uncertainty which also implies that the classification is likely accurate and vice versa. For our running example of HAR, we analyze the entropy of the distribution of classes probabilities for classifications performed on inputs representing different sampling rates. For each input sample we compute the normalized entropy of the prediction achieved at every sampling rate, for both correctly classified samples and incorrectly classified samples, and we illustrate the results in Figure 5.

The entropy values for the correct predictions (blue) are on the average about 70% lower than the entropy values of incorrect

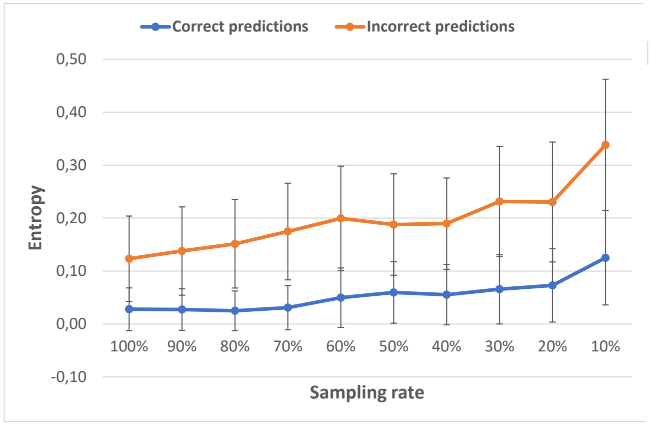

Figure 5: Average entropy values for correct (blue line) and incorrect (orange line) predictions at different sampling rates (error bars represent two standard deviations).

predictions (orange), for samples taken with the same sampling rate (sparsity). In addition, as the sampling rate decreases, increasing the sparsity of the inputs, the entropy levels raise significantly for both correct and incorrect predictions, indicating that in general sparser samples are inherently more difficult to classify.

These results suggest that since there is clear gap between the entropy values of the correct and incorrect predictions, entropy offers a good indicator of the correctness of the prediction and as such we can use it to guide the sampling rate selection. Nevertheless, choosing the right threshold that separates correct and incorrect predictions, based on the entropy value, is not a trivial task, since the results discussed above represent average values. The actual distribution of the entropy values shows more intertwined correct and incorrect values, thus, examining softmax layer entropy for predicting whether a signal sampled at a certain rate will be classified correctly is not straightforward.

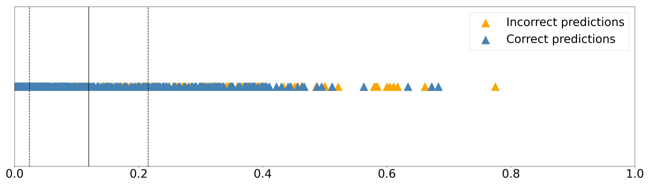

Figure 6: SVM decision boundary (vertical continuous line) and margins (vertical dotted lines) on predictions' entropy values.

To find an entropy-driven boundary that separates potentially correct and potentially incorrect classification of a certain input at a certain sampling rate we harness the Support Vector Machine (SVM) classifier. SVM seeks to find the decision boundary that maximizes the separation between two or more classes. For cases like ours, where two classes seem impossible to separate using a linear separator, SVM performs a projection of the data (in this case, numeric values of the entropy of the predictions, with their associated labels: correct or incorrect prediction) into a higher dimensional space where the separation is feasible. In that higher dimensional space, SVM learns the most suitable decision boundary. As such, we use the decision boundary of the SVM (Figure 6) for defining the threshold between the correct and incorrect entropy values for a given sampling rate.

Next, we devise an algorithm that based on the *current* predicted activity label and the entropy of the current prediction, sets the sampling rate to be used for sampling and classifying the *next* input. Choosing the next sampling rate based on the inference result for the current sample is based on the fact that human activity inertial data is not characterized by rapid variations [17] and there is usually a high similarity degree between consecutive samples. Moreover, previous work [19] showed a medium-strong level of correlation between the prediction's confidence value of each two consecutive samples of the UCI HAR dataset.

At each inference point, the algorithm takes the entropy of the current prediction and compares it to the pre-computed entropy threshold for the current sampling rate. Based on this comparison, the next sampling rate is chosen: an above-the-threshold result (indicating a "difficult" prediction) will lead to a higher sampling rate, while a below-the-threshold situation indicates an "easy" prediction and enables selecting a lower sampling rate. For each activity class, the algorithm can choose between two sampling rates, one achieving highest possible accuracy and the other maximizing compression with minimum accuracy loss.

The specific sampling rate options for each activity class were established empirically based on the average accuracy values across all sampling rates, for each activity class label. The analysis showed that there is no clear monotonic degradation in inference accuracy as the compression increases. The sitting activity for example, achieves the top accuracy when sampled at a 50% sampling rate and laying at a 40% sampling rate. As such, the specific sampling rates for each class label were chosen to maximize energy savings while maintaining the maximum average accuracy achievable by the network for that particular class label.

The algorithm allows threshold tuning, enabling accuracy vs. energy savings trade-off according to the context of use. The goal of the adaptation algorithm is to achieve the desired accuracy with the minimum sampling (and consequently energy consumption). We start from the trade-off point between the accuracy and the energy consumption determined by the SVM boundary between entropy values of correct and incorrect predictions. To enable different trade-off points between accuracy and energy consumption, we add or subtract from this point the SVM margin, thus obtaining the SVM− MARGIN, and the SVM+MARGIN points (Figure 7). We can also maximize/minimize either the savings or the accuracy of the inference, by setting the threshold to the maximum/minimum possible value for the entropy, obtaining the HIGH_COMPRESSION point- for the entropy threshold of 1 and the HIGH_ACCURACY point-for the entropy threshold set to 0 (Figure 7).

4 IMPLEMENTATION

We first validated our approach on the UCI HAR [2] dataset. The UCI HAR dataset provides recordings of embedded tri-axial sensors of accelerometer and gyroscope from a smartphone carried by 30 volunteers while performing six different activities (walking,

walking upstairs, walking downstairs, sitting, standing and lying). The activity recognition dataset was collected at a sampling rate of 50Hz, and the collected samples were grouped in fixed-width sliding windows of 2.56 seconds with a 50% overlap, thus each data segment has 128 readings of 9 sensor values: the triaxial total acceleration from the accelerometer, the triaxial body acceleration, and the triaxial angular velocity from the gyroscope. In our experiments, we aim to evaluate how the algorithm scales with different sampling rates, so we artificially created data samples corresponding to different percentages of the original sampling rate: thus 100%, 90%, 80%, etc., up to a sampling rate of 10% of the original one. In order to simulate a reduced sampling rate, we zero-filled a certain number of groups of 9 consecutive values, corresponding to all the sensor's readings at a specific time step. For example, if the desired sampling rate is 50% of the original one, half of the 128 groups of 9 sensors readings will be replaced by zeros. The selection of the time steps to be skipped from sampling, is made according to the learned measurement matrix elaborated in Section 3.1. In Section 5.1 we show results on the UCI HAR dataset analysis.

Next, we assess the energy efficiency of our approach, so for that purpose, we implement a fully-functioning version of our solution on an embedded computing device to measure the energy consumption for both the sampling process and the entire sampling and inference pipeline. All the energy measurements were performed on an UDOO Neo Full [39] board. This is an IoT platform with processing hardware similar to that found in modern low-end smartphones and having embedded accelerometer and gyroscope sensors, that can run both Android and Linux OS, thus allowing us to easily prototype and test our model. The experiments were performed on Ubuntu 18.04.6 LTS, the CS-DL pipeline was executed using Python [40] version 3.5.9 and Tensorflow[1] 1.5.0, and the sensor sampling was implemented using the Neo.GPIO library [10]. The power consumption was measured using a Monsoon power monitor tool [15], a commonly used tool for power measurements in mobile and embedded computing [35]. The results of these experiments are presented in Section 5.2.

5 EXPERIMENTAL RESULTS

5.1 Accuracy Analysis

We start by evaluating the performance of the DL model across the different sampling rates (static sampling scenario) and notice its accuracy scores vary from 73% (using the lowest sampling rate, 10%) to 88% (for full sampling), with a general downward trend for accuracy as the compression increases (moving towards lower sampling rates) as shown in Figure 7.

We then compare the results of this static sampling scenario with the proposed adaptive sensing framework discussed in Section 3.2 for different accuracy thresholds. We explored several different scenarios, based on the level of the target reliability, ranging from lower reliability/lower sampling rate (HIGH_COMPRESSION scenario), to higher reliability/higher sampling rate (HIGH_ACCURACY scenario). In all of these scenarios, the sampling rate selection is driven by the previous input's activity class and its classification difficulty, with a tolerance for the reliability/accuracy of the inference ranging from high to low. The tolerance level is set through the entropy threshold value as explained in Section 3.2.

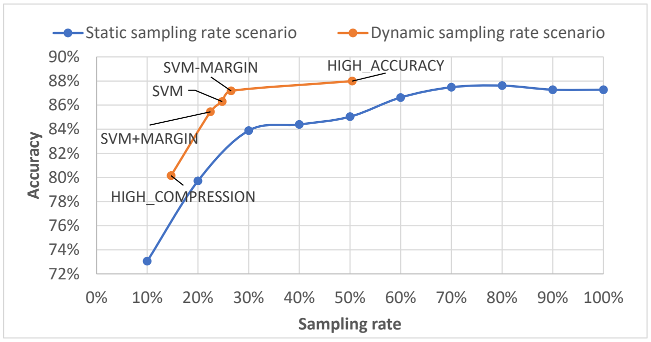

Figure 7: Average accuracy vs. average sampling rate in a static scenario (blue line) and various adaptive scenarios (orange line) on the UCI HAR dataset.

Figure 7 demonstrates that across the board our adaptive rate approach leads to a higher accuracy than the static approach for the same average sampling rate. For example, the high accuracy scenario achieves an average accuracy of 88%, using on average a sampling rate of only 50%, while in a static scenario, the average accuracy would have been 85% at the same sampling rate. In other words, our dynamic approach achieved the accuracy of the full-scale sampling model, when using on average, a sampling rate of only 50% of the original one. Moreover, the accuracy drops with just 1%, if the sampling rate is 25% of the original one. This is due to the fact that our algorithm chooses at each time step a more appropriate sampling rate, which is not necessarily the highest one. For instance, surprisingly, some activities are more accurately classified at a lower compression rate.

5.2 Energy Savings Analysis

Intuitively, reducing the sampling rate directly translates into less power being consumed by the sensors. Theoretically, if we move from sampling at 50Hz to sampling at 20Hz, 60% of the energy used on sampling could potentially be saved. Yet, various intricacies of the actual hardware and the system call for a careful examination of the actual savings enabled by reduced sampling rates. Previous studies, such as [18], analyzed the impact of varying the sampling rate of a wearable accelerometer on the battery consumption and showed that reducing the sampling rate from 100 Hz down to 25 Hz, for example, would more than double the battery's lifetime. Furthermore, additional processing, transmission, storage savings can be achieved. To see how these theoretical savings translate into practice in our case, we created two test benchmarks for measuring the energy consumption: the first for the energy consumed by the sampling process only (reading the sensors' values to create the input vector for our inference pipeline) and the second one for measuring the energy used for the entire CS-DL inference pipeline. Both scenarios were deployed three times by capturing multiple consecutive samples at the desired sampling rate and the averaged results are reported.

For the first experiment, we measured the energy consumed when sampling 128 consecutive readings of the UDOO board's accelerometer and gyroscope sensors at 50Hz, versus skipping the readings according to the *mask* given by the CS sampling layer (our measurement matrix). A value is sampled if the corresponding measurement matrix element is a non-null value and skipped otherwise. The mask was varied for each compression level and the measurements, illustrated by the orange bars in Figure 8, focus on the energy used for sampling with different masks, i.e. rates. These results include only the energy consumed by the "reading" script (the idle power consumption of the system has been subtracted).

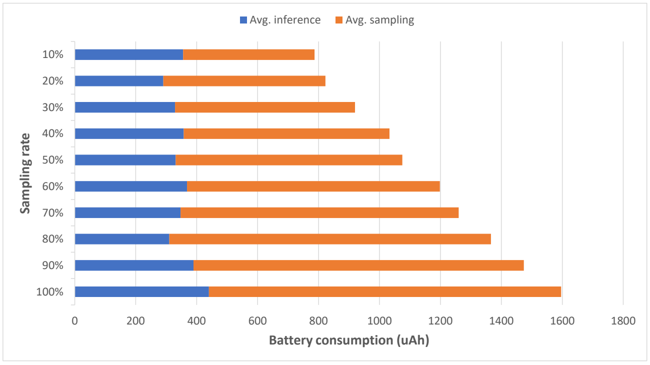

Figure 8: Average energy consumed for real-time sampling (orange bars) and human activity inference (blue bars) from a single 2.56s data instance on a UDOO board.

For the second experiment, we measured the energy consumption of the complete CS-DL pipeline, which includes reading the sensors' values and processing them through the DL model. Again, we measure the energy consumed when sampling 128 consecutive readings of the UDOO board's accelerometer and gyroscope sensors at 50Hz, versus skipping the readings according to the mask given by the CS sampling layer. These measurements, also illustrated in Figure 8, reflect the relationship between a given sampling rate and the energy consumed for reading, filtering, formatting and processing the sampled data through the neural network. Unlike the sampling part that scales linearly with the sampling rate, the processing of the data takes almost the same amount of energy irrespective of the sampling rate, because although we read less data, we still have to up-sample it by substituting with zero the missing values, so the number of operations performed for filtering, formatting and processing remains the same.

The final step in our energy analysis is the assessment of the energy savings enabled by our context adaptive algorithm. For this purpose, we aim to evaluate the energy-accuracy trade-off between running the adaptive algorithm with different threshold values and running the static sampling scenario on the same test set used for evaluating the classification accuracy. We measure the energy consumption in each case and plot the result in Figure 9.

6 DISCUSSION

The experiments presented in Figure 8 show that on a resource-constrained IoT platform the sampling process takes significantly more energy than the inference part, being up to three times more energy consuming. In addition, the energy usage scales linearly with the sampling rate. For example shifting from a 50 Hz sampling

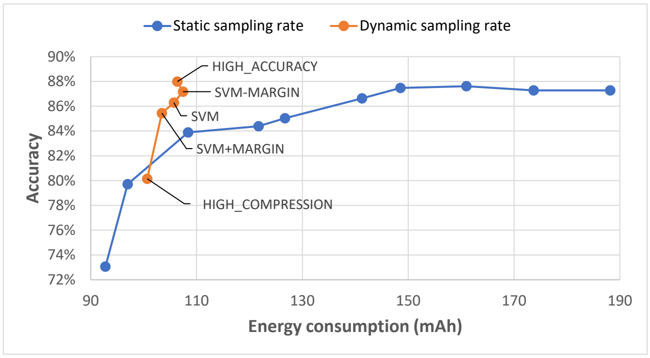

Figure 9: Energy-accuracy trade off in a static scenario (blue line) and various adaptive scenario (orange line).

rate (100%) to a 30 Hz sampling rate (60%), saves 50% of the energy used for sampling alone. *This implies the foremost need for reducing the sampling rate and using each available sample judiciously.*

On the other hand, the energy used for inference does not show a linear decrease, as one would expect given that both the input and the weights of the dense layer have a higher sparsity at reduced sampling rates. This is not to say that popular optimizations on the DL-level, such as weight pruning, are irrelevant. Instead, to truly benefit from processing sparser data, additional optimizations of the deep-learning model are required before deploying it on a device. While some steps have already been made in this direction (such is the TensorFlow Lite XNNPACK inference engine [38] that offers support for accelerating sparse inference for CNN models), it is to be expected that future advances in libraries and compiler architectures will make such optimizations inherently supported, fostering the acceleration of sparse models.

The key benefit of our solution is adaptability. First, adapting to the input difficulty, up to 43% of the necessary energy can be saved with the same average accuracy compared to the 100% static sampling scenario (Figure 9, HIGH_ACCURACY option). Second, depending on the users' needs, context of operation, or currently available resources, our solution enables on-the-fly adaptation of the sampling-processing pipeline (see orange trade-off line in Figure 9). For instance, if the goal is to maximize energy savings, the adaptive algorithm can save up to 46% energy compared to the full static sampling scenario, with a 6% drop in the average accuracy.

The energy savings enabled by our solution translate into an extended battery lifetime. This aspect is critical for applications that rely on continuous long-term sensing, and where the battery's lifetime is a concern, such as healthcare on-body monitoring, wildlife tracking, or infrastructure monitoring applications. Previously, the approach for this kind of applications was a performance trade-off between the prediction accuracy and the battery lifetime through the sampling rate: either using a lower sampling rate, that gives poor accuracy, but a longer monitoring period, or opting for a higher sampling rate, with better accuracy, but a shorter monitoring period. Our approach achieves the same quality of service with much lower energy consumption and, thus, enables operation over a longer period of time.

In this paper we present the first framework for CS-inspired DL inference on edge devices. Being preliminary, our solution faces certain practical limitations. First, in our work the deep learning model performs the same amount of operations irrespective of the number of samples acquired. This restrains the energy savings enabled by our solution mostly to the energy saved on sampling, while the inference energy consumption remains more or less the same. By optimizing the proposed model using the recently proposed DAP layer [28] for example, the energy efficiency of our pipeline could be further expanded. Second, at the moment our adaptation algorithm decides between two possible choices of sampling rates per inference class. A higher number of options could allow for a finer energy-accuracy tuning, yet would require a more complex decision method. Finally, while not an immediate topic of our work, we note that our solution opens space for further optimization of joint sampling and learning process, which we have not explored in this paper. Recent research advances that target network complexity and redundancy, such as quantization [11], pruning [12], slimmable neural networks [44], or early exiting [37], for example, can be harnessed in parallel with sampling optimization. Furthermore, by exploiting these techniques in a dynamic context, such as the one in our approach, additional savings might be achieved with a minimal impact on the performance of the application, by adapting the computations to the variations in the input's complexity [9, 23].

7 RELATED WORK

During the past few years, a tremendous research effort focused towards exploring DL-based implementations for CS [20, 30, 32, 43]. Several of these research works targeted CS-DL implementations for identifying human activities using wearable sensors [24, 46]. A common feature of these approaches is that they all assume a static sampling rate scenario for a trained network. In other words, the neural network model is trained for one single sampling rate, defined before the training, and thus can only infer from data acquired at the same sampling rate.

The first CS-DL solution allowing adaptive sampling rate was proposed by Lohit et al. [27] and is based on a three-steps training process: first, for the highest desired sampling rate, second, for the lowest sampling rate (all other parameters frozen) and finally, for an a additional sampling rate between the highest and the lowest (with the rest of parameters frozen). In another work [41], more closely related to our research, the authors also address the rate adaptivity aspect in the CS-DL context. Their solution is to create measurement vectors of various lengths, simulating samples acquired at different measurement rates, and use them to re-train a neural network with a fixed input layer size. Unlike [41], where the measurement matrix is a binary sensing one, our approach is based on a data-driven, *learned* diagonal matrix. In addition, the structure of the measurement vectors also differs: in [41] these vectors are zero-padded at the end to compensate for the size mismatch between their variable length and the fixed input size of the inference network, while in our case, the measurement vectors have zero values in the locations corresponding to sampling pauses. The main bottleneck of DL-based rate adaptive approaches is the lack of available neural network architectures that could handle variable dimensions of their inputs, without up-sampling or imputing the data. A solution for this was recently proposed in [28], where a DL model invariant to sampling rates changes was proposed.

Another key question of an adaptive sampling framework is how to harness the context sensitivity for efficiently guiding the adaptation mechanism. Most of HAR studies rely on estimating the activity type [8], the signal type [31] or the user profile [45] or on deriving some statistical features from the input data [34] as triggers for adaptivity. When it comes to deep-learning based implementation, the confidence of the network is often exploited as a marker for the need to adjust the energy-accuracy trade off [3]. Nonetheless, deriving efficient methods for context sampling and adaptation is still a challenge.

Our context-adaptive algorithm complements the above research and is based on the entropy of the probability distributions of the predictions for tuning the sampling rate to satisfy predefined (yet possibly dynamic) accuracy requirements. Uniquely in this work, we guide the adaptation based not only on the difficulty of the class of activities or signal type that the sample belongs to, but also on each individual sample's difficulty. This allows a more fine-grain tuning of the sampling rate sensitive to the hidden particularities of each individual input. In addition, our method supports very flexible levels of approximation, based on the desired reliability-accuracy trade off of the inference, ranging from lower reliability/lower sampling rate, to higher reliability/higher sampling rate.

8 CONCLUSION

In this paper we proposed a pipeline for dynamically adjusting the sampling rate on the fly and performing classification on inputs whose sizes that may vary, while preserving the inference accuracy with the least amount of sampling. We implemented our CS-based DL pipeline and evaluated its performance on the well-established UCI human activities dataset. We find that changing the sampling rate affects the accuracy inference differently depending on the context (i.e. the physical activity), but also depending on each individual input's difficulty, thus, we used the previously predicted activity label and the confidence of that prediction to dynamically adapt the sampling rate at each inference point.

The experiments validated our approach, and revealed that a trade-off between the classification accuracy and energy consumption can be struck with our adaptive sampling rate algorithm. In comparison to the static sampling and inference approach, our method used 43% less energy while achieving the same inference accuracy. The practical energy savings enabled by our framework are of paramount importance for edge computing applications which rely on a very limited energy budget; by extending the battery lifetime and maintaining the same level of inference accuracy, our CS-DL framework contributes to advancing deep learning at the edge.

ACKNOWLEDGMENTS

The research presented in this paper was funded by projects "Bringing Resource Efficiency to Smartphones with Approximate Computing" (N2-0136), "Context-Aware On-Device Approximate Computing" (J2-3047), and by the Slovenian Research Agency research core funding No. P2-0098 and P2-0426.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 265–283.

[2] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, Jorge Luis Reyes-Ortiz, et al. 2013. A public domain dataset for human activity recognition using smartphones.. In *Esann*, Vol. 3. 3.

[3] Konstantin Berestizshevsky and Guy Even. 2019. Dynamically sacrificing accuracy for reduced computation: Cascaded inference based on softmax confidence. In *International Conference on Artificial Neural Networks*. Springer, 306–320.

[4] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. 2017. Adaptive neural networks for efficient inference. In *Int. Conf. on Machine Learning (ICML)*. Sydney, Australia.

[5] Emmanuel J Candès, Justin Romberg, and Terence Tao. 2006. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory* 52, 2 (2006), 489–509.

[6] Zhicheng Cui, Wenlin Chen, and Yixin Chen. 2016. Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995* (2016).

[7] David L Donoho. 2006. Compressed sensing. *IEEE Transactions on Information Theory* 52, 4 (2006), 1289–1306.

[8] Ramin Fallahzadeh, Josue Pagan Ortiz, and Hassan Ghasemzadeh. 2017. Adaptive compressed sensing at the fingertip of internet-of-things sensors: An ultra-low power activity recognition. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 996–1001.

[9] Xitong Gao, Yiren Zhao, Łukasz Dudziak, Robert Mullins, and Cheng-zhong Xu. 2018. Dynamic channel pruning: Feature boosting and suppression. *arXiv preprint arXiv:1810.05331* (2018).

[10] Neo GPIO. 2016. *A python library to control the Gpios, Accel, Gyro, Temp, Baro, Magno sensors/pins easily.* Retrieved February 9th 2022 from https://github.com/smerkousdavid/Neo.GPIO/tree/master/neo

[11] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. 2015. Deep learning with limited numerical precision. In *International conference on machine learning*. PMLR, 1737–1746.

[12] Song Han, Jeff Pool, John Tran, and William J Dally. 2015. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626* (2015).

[13] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[14] Michael Iliadis, Leonidas Spinoulas, and Aggelos K Katsaggelos. 2018. Deep fully-connected networks for video compressive sensing. *Digital Signal Processing* 72 (2018), 9–18.

[15] Monsoon Solutions Inc. 2015. *Monsoon Solutions high voltage power monitor.* Retrieved February 3rd 2022 from http://msoon.github.io/powermonitor/HVPM.html

[16] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. PMLR, 448–456.

[17] Roua Jabla, Félix Buendía, Maha Khemaja, and Sami Faiz. 2019. Balancing Timing and Accuracy Requirements in Human Activity Recognition Mobile Applications. In *Multidisciplinary Digital Publishing Institute Proceedings*, Vol. 31. 15.

[18] Aftab Khan, Nils Hammerla, Sebastian Mellor, and Thomas Plötz. 2016. Optimising sampling rates for accelerometer-based human activity recognition. *Pattern Recognition Letters* 73 (2016), 33–40.

[19] Timotej Knez, Octavian Machidon, and Veljko Pejović. 2021. Self-Adaptive Approximate Mobile Deep Learning. *Electronics* 10, 23 (2021), 2958.

[20] Kuldeep Kulkarni, Suhas Lohit, Pavan Turaga, Ronan Kerviche, and Amit Ashok. 2016. Reconnet: Non-iterative reconstruction of images from compressively sensed measurements. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 449–458.

[21] Kuldeep Kulkarni and Pavan Turaga. 2015. Reconstruction-free action inference from compressive imagers. *IEEE transactions on pattern analysis and machine intelligence* 38, 4 (2015), 772–784.

[22] Chiman Kwan, David Gribben, Akshay Rangamani, Trac Tran, Jack Zhang, and Ralph Etienne-Cummings. 2020. Detection and confirmation of multiple human targets using pixel-wise code aperture measurements. *Journal of Imaging* 6, 6 (2020), 40.

[23] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. 2017. Runtime neural pruning. *Advances in neural information processing systems* 30 (2017).

[24] Guocheng Liu, Rui Ma, and Qi Hao. 2018. A Reinforcement Learning Based Design of Compressive Sensing Systems for Human Activity Recognition. In *2018 IEEE SENSORS*. IEEE, 1–4.

[25] Suhas Lohit, Kuldeep Kulkarni, Ronan Kerviche, Pavan Turaga, and Amit Ashok. 2018. Convolutional neural networks for noniterative reconstruction of compressively sensed images. *IEEE Transactions on Computational Imaging* 4, 3 (2018), 326–340.

[26] Suhas Lohit, Kuldeep Kulkarni, Pavan Turaga, Jian Wang, and Aswin C Sankaranarayanan. 2015. Reconstruction-free inference on compressive measurements. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 16–24.

[27] Suhas Lohit, Rajhans Singh, Kuldeep Kulkarni, and Pavan Turaga. 2018. Rate-adaptive neural networks for spatial multiplexers. *arXiv preprint arXiv:1809.02850* (2018).

[28] Mohammad Malekzadeh, Richard Clegg, Andrea Cavallaro, and Hamed Haddadi. 2021. DANA: Dimension-Adaptive Neural Architecture for Multivariate Sensor Data. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 3 (2021), 1–27.

[29] Ali Mousavi and Richard G Baraniuk. 2017. Learning to invert: Signal recovery via deep convolutional networks. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2272–2276.

[30] Ali Mousavi, Ankit B Patel, and Richard G Baraniuk. 2015. A deep learning approach to structured signal recovery. *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)* (2015), 1336–1343.

[31] Josue Pagan, Ramin Fallahzadeh, Mahdi Pedram, Jose L Risco-Martin, Jose M Moya, Jose L Ayala, and Hassan Ghasemzadeh. 2018. Toward ultra-low-power remote health monitoring: An optimal and adaptive compressed sensing framework for activity recognition. *IEEE Transactions on Mobile Computing* 18, 3 (2018), 658–673.

[32] Hamid Palangi, Rabab Ward, and Li Deng. 2016. Distributed compressive sensing: A deep learning approach. *IEEE Transactions on Signal Processing* 64, 17 (2016), 4504–4518.

[33] Laurence AF Park and Simeon Simoff. 2015. Using entropy as a measure of acceptance for multi-label classification. In *International symposium on intelligent data analysis*. Springer, 217–228.

[34] Nafiul Rashid, Berken Utku Demirel, and Mohammad Abdullah Al Faruque. 2021. AHAR: Adaptive CNN for Energy-efficient Human Activity Recognition in Low-power Edge Devices. *arXiv preprint arXiv:2102.01875* (2021).

[35] Andreas Schuler and Gabriele Anderst-Kotsis. Houston, TX, USA, 12–14 November 2019. Examining the energy impact of sorting algorithms on Android: an empirical study. In *Proceedings of the 16th EAI Int. Conf. on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous '19)*. ACM, 404–413.

[36] Vanika Singhal, Angshul Majumdar, and Rabab K Ward. 2017. Semi-supervised deep blind compressed sensing for analysis and reconstruction of biomedical signals from compressive measurements. *IEEE Access* 6 (2017), 545–553.

[37] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2464–2469.

[38] Tensorflow. 2021. *Pruning for on-device inference with XNNPACK.* Retrieved January 24, 2022 from https://www.tensorflow.org/model_optimization/guide/pruning/pruning_for_on_device_inference

[39] UDOO. 2021. *UDOO Single Board Computer.* Retrieved February 3rd 2022 from https://www.udoo.org/udoo-neo/

[40] Guido Van Rossum and Fred L. Drake. 2009. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

[41] Yibo Xu, Weidi Liu, and Kevin F Kelly. 2020. Compressed Domain Image Classification Using a Dynamic-Rate Neural Network. *IEEE Access* 8 (2020), 217711–217722.

[42] Y. Yang, J. Sun, H. Li, and Z. Xu. 2020. ADMM-CSNet: A Deep Learning Approach for Image Compressive Sensing. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 3 (2020), 521–538.

[43] Hantao Yao, Feng Dai, Shiliang Zhang, Yongdong Zhang, Qi Tian, and Changsheng Xu. 2019. DR2-net: Deep residual reconstruction network for image compressive sensing. *Neurocomputing* 359 (2019), 483–493.

[44] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. 2018. Slimmable neural networks. *arXiv preprint arXiv:1812.08928* (2018).

[45] Ozgur Yurur, Miguel Labrador, and Wilfrido Moreno. 2013. Adaptive and energy efficient context representation framework in mobile sensing. *IEEE Transactions on Mobile Computing* 13, 8 (2013), 1681–1693.

[46] Abrar Zahin, Le Thanh Tan, and Rose Qingyang Hu. 2019. Sensor-based human activity recognition for smart healthcare: A semi-supervised machine learning. In *International conference on artificial intelligence for communications and networks*. Springer, 450–472.