<http://www.diva-portal.org>

Postprint

This is the accepted version of a paper presented at *IEEE Conference on Distributed Computing in Sensor Systems (DCOSS), Online, June 15-17, 2020*.

Citation for the original published paper:

Hazra, S., Brachmann, M., Voigt, T. (2020)

Shadow-based Hand Gesture Recognition in one Packet

In: *16th Annual International Conference on Distributed Computing in Sensor Systems (DCOSS 2020)* (pp. 27-34). IEEE

<https://doi.org/10.1109/DCOSS49796.2020.00018>

N.B. When citing this work, cite the original published paper.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-427589>

Shadow-based Hand Gesture Recognition in one Packet

Saptarshi Hazra*, Martina Brachmann* and Thiemo Voigt*[†]

*RISE Research Institutes of Sweden, Stockholm, Sweden

[†]Uppsala University, Uppsala, Sweden

Email: {firstname.lastname@ri.se}

Abstract—The ubiquity of wirelessly connected sensing devices in IoT applications provides the opportunity to enable various types of interaction with our digitally connected environment. Currently, low processing capabilities and high energy costs for communication limit the use of energy-constrained devices for this purpose. In this paper, we address this challenge by exploring the new possibilities highly capable deep neural network classifiers present. To reduce the energy consumption for transferring continuously sampled data, we propose to compress the sensed data and perform classification at the edge. We evaluate several compression methods in the context of a shadow-based hand gesture detection application, where the classification is performed using a convolutional neural network. We show that simple data reduction methods allow us to compress the sensed data into a single IEEE 802.15.4 packet while maintaining a classification accuracy of 93%. We further show the generality of our compression methods in an audio-based interaction scenario.

I. INTRODUCTION

We are increasingly surrounded by small Internet of Things devices that sense our environment to enable applications such as smart locks [11], [29], [35], smart thermostats [9], and smart lighting solutions [27]. The ubiquitous nature of these devices can also be used to enable various types of interactions such as gesture [17] and activity detection [32], especially in combination with machine learning. However, due to the resource-constrained character of these wireless devices w.r.t processing power, memory, and energy-budget, deploying large, complex models for interaction classification and detection on the devices is currently difficult. Successfully detecting user interactions requires continuous sensor readings. Streaming the sensor readings to a more powerful device such as an edge cloud incurs heavy communication costs, resulting in fast energy depletion and a short device lifetime. In addition, the dense deployment of these devices results in a high channel utilization and eventually wireless interference.

To avoid streaming sensor readings and to reduce the amount of sensor data that needs to be transmitted, previous work has proposed novel data compression mechanisms [21], [22], [34], in-network processing (i.e., data aggregation) [15], [19], [30], and data prediction algorithms [24], [26]. These approaches are primarily focused on monitoring the environment and therefore require a low sampling frequency. In contrast, interaction-based applications demand fast sensing as the responsiveness of the system is going to affect the user experience [31]. Furthermore, the aforementioned approaches target high accuracy between original sensor readings and approximated sensor values.

However, in interaction-based applications, each user's action – even if it is the same action performed by the same user – produces slightly different sensor readings and highly accurate individual sensor readings are, thus, not necessary.

Through the introduction of deep learning, machine learning has made a gigantic step forward in recent years [16]. Today, deep learning is easily accessible through advanced and powerful libraries such as FASTAI [7]. Deep learning is particularly strong in processing images, video, speech and audio [16] and even commercial players [4] have applied deep learning-based computer vision to existing wireless systems and protocols including WiFi and LTE.

In this paper, we combine vision-based deep learning classification with data reduction methods in the context of interaction-based applications. In particular, we aim to fit the light sensor data of a single hand gesture captured by a resource-constrained device into one IEEE 802.15.4 packet. The packet is received by an edge device to correctly classify the captured interaction. Before classification, the edge device converts the one-dimensional time series sensor data to an image with the light intensity value plotted against the data point indices (time progression). We propose and implement two data reduction methods, compare their performance with existing methods that we adapt to our needs and evaluate our solutions in a shadow-based gesture detection applications.

Our main contributions are the following:

- We highlight the feasibility of deep learning classifiers in sensor-based interaction applications for reducing the amount of data that needs to be sent from a sensor for processing.
- We propose two lightweight methods and adapt DBP [24], [25] and APCA [3], two state-of-the-art compression methods, for reducing the number of sensed data in a shadow-based gesture detection application.
- We demonstrate the capability of our methods to fit the amount of data needed for a gesture in one IEEE 802.15.4 packet while maintaining a high gesture classification accuracy.
- In order to evaluate the generality of our solution, we also experiment with audio-based interactions. Our results indicate that our compression methods can be generalized to different interaction-based applications with different sensor ranges and sampling frequencies.

Fig. 1: Hand gestures and their corresponding shadow patterns. The shadow patterns of a swipe and a one tap look similar. A quickly performed one tap can, hence, be easily confused with a swipe.

II. SHADOW-BASED HAND GESTURE RECOGNITION

Shadow-based hand gesture detection enables users to communicate with devices and to interact with the environment by means of hand movements and gestures that trigger corresponding actions. More precisely, light sensors, photo-diodes or solar cells measure the shadow that a user and its hands cause [17], [18], [33]. The measured sensor values are then sent to a more powerful device such as an edge cloud to perform hand gesture recognition. The hand gesture recognition itself is normally done via machine learning techniques.

Our system follows the same principle: we use a light sensor on a node to detect shadows caused by hand gestures. Once the start of a gesture is detected, we sample and then transmit light sensor readings to an edge cloud to perform hand gesture recognition via vision-based deep learning [28]. We perform the processing and classification on the edge cloud since the general problem of processing short as well as long temporal patterns is complicated on the most resource-constrained sensor nodes. Furthermore, updating models on the edge is easier and more resource-efficient than on multiple resource-constrained devices.

Detecting the start of hand gestures. A light sensor persistently monitors the environment as a user may perform a gesture at any time. To capture small nuances between different hand gestures, a continuous stream of sensor data is required. In applications such as smart implicit homes [8], it is envisioned that a large number of sensors measure light and other measures of interest to understand users' intentions and needs. Transmitting all these sensor readings to the edge cloud would incur heavy communication and high interference, resulting in high data loss and energy consumption. For gesture sensing, it is thus of utmost importance to detect the start of a gesture already at the sensor nodes and only transmit the necessary amount of data required to correctly classify the hand gestures.

Hand gestures. We consider three hand gestures: swipe, one tap, and two taps. A swipe is a quick hand movement over a light sensor from the right to the left side or vice versa. A one tap is a single movement above the light sensor downwards and upwards again. A two tap consists of two consecutively performed one taps. Figure 1 depicts the three hand movements and the corresponding shadow patterns based on 400 data points

each taken every 8 ms from the light sensor.¹

The figure shows that the shadow patterns of a swipe, illustrated in Figure 1a, and a one tap, shown in Figure 1b are very similar. A fast execution of a one tap can easily be confused with a swipe, making it challenging for the deep learning model to correctly classify the gestures.

The amount of data points needed to describe a gesture highly depends on the speed at which a gesture is performed. We found heuristically with 13 users that 400 data points are sufficient to describe the three hand gestures that we consider in this paper. With light values of 2 byte, 800 byte (2 byte x 400 data points), and thus approximately eight IEEE 802.15.4 frames must be sent in case of no data compression to the edge cloud for detecting a single gesture. We aim at reducing the amount of light values so that only a single packet must be transmitted to the edge cloud for detecting a gesture.

III. METHODS TO REDUCE DATA

In this section, we describe methods for reducing the amount of data that nodes must send and for re-constructing the reduced data at the edge cloud. We first present our own two methods and afterwards explain existing methods that we adapt for shadow-based hand gesture detection in one packet.

A. Removing and Interpolating Samples (RI)

One intentionally simple method to reduce the amount of data to be transferred from the sensor to the edge cloud is to *not* transmit all samples. This approach corresponds to sampling at lower frequency. However, sampling at a lower frequency may make it difficult to recognize the start of a gesture. Hence, our approach is to only transmit every n th sample once a gesture has been recognized. We interpolate the missing samples on the edge computer using linear interpolation provided by Python's SciPy package. By this means, the decompressed graph keeps the interesting features that also the original graph has.

Figure 2 shows an example of a one tap both as the original graph without removal of any samples and with the removal and interpolation when only every 8th and 16th sample, respectively, is kept. While in particular the graph which keeps only every 16th looks different to the original graph, it still keeps its interesting features and is classified correctly at the edge cloud.

¹We have chosen 8 ms as this is the most fine-grained timer for Contiki's etimer.

Fig. 2: Example RI and FIP. The one tap is still correctly classified after interpolation.

This approach requires an additional method to detect the start and end of a gesture on a resource-constrained node. For this, we use the well-established median absolute deviation (MAD) that has also been used by Kaholokula et al. [13] for detecting hand gestures, but with several photo diodes.

RI is very lightweight for the resource-constrained nodes. It has also has the advantage that the nodes only have to send the n th samples. No additional information must be transmitted to the edge cloud as long as long as node and edge cloud use the same n . The compressed size of a gesture can, thus be expressed as:

$$\text{compressed size} = 2 \text{ byte} \cdot \# \text{samples}, \quad (1)$$

where $\# \text{samples}$ is the number of samples that we actually send between the start of a gesture and its end described as a 2 byte variable.

B. Finding Inflection Points (FIP)

Transmitting every n th data point in RI might eventuate in removing an inflection point in the shadow trace. In this case, the resulting interpolated trace deviates from the curve and may result in a wrong classification. We therefore propose FIP, which aims at finding the inflection points in sensor readings. FIP measures the slope at a data point with respect to the last saved data point and the next one. The rate of change in the sensor readings at that data point can then be measured by the difference in these two slopes. Defining a threshold on this rate of change allows us to control the size of data we send by varying the rate of changes we measure. If the rate of change is above the threshold, we note the data value and the offset from the last measured data point. Afterward, we linearly interpolate the received data points on the edge cloud to derive the decompressed gesture.

For high thresholds, FIP detects quick changes in the shadow curve, but it detects gradual changes only when they have deviated above the threshold. This results in deviations of the interpolated shadow curve from the original. Hence, we scale the threshold according to the distance from the last measured inflection point to also take into account the gradual changes. The sensitivity of the sensor is affected by the light conditions thereby modifying the range of changes in the trace.

The threshold is set as a percentage of the median of the sensor readings to counteract this dependency on light conditions.

Figure 2 shows an example of a one tap both as the original graph without removal of any samples and with the threshold set to 0.3 and 0.6 respectively. With higher thresholds, FIP only keeps the data points located at a large change in the slope of the curve. Even though the graphs look different from the original ones, they are classified correctly.

FIP also requires an additional method to detect the start and end of the gesture. For this, we use, as in RI, the MAD. The compressed size of a gesture can, thus be expressed as:

$$\text{compressed size} = (2 \text{ byte} + 1 \text{ byte}) \cdot \# \text{samples}, \quad (2)$$

where $\# \text{samples}$ is the number of data points for which the rate of change is above the threshold. We need two bytes for the data point value and one byte for the offset from the previous saved data point.

C. Adaptation of Existing Methods

Reducing the amount of sensor data has been done in the past [10]. However, most methods focus on environmental monitoring with a low sampling frequency. In the following, we describe two existing methods that we adapt for reducing fast sampled data for hand gesture recognition.

1) *Piecewise Constant Approximation (APCA)*: APCA [3] uses a constant model to compress sensor values. It averages all consecutive sensor values that are within a given error threshold to a single value. The averaged value and the amount of values that are summarized describe a window. More precisely, APCA inserts sensor values into the window as long as $\min(\text{window}) - \max(\text{window})$ is smaller or equal than two times $\text{avg}(\text{window})$ times the relative error threshold. Otherwise a new window is created. The compressed size of the gesture values can be computed as:

$$\text{compressed size} = (2 \text{ byte} + 1 \text{ byte}) \cdot \# \text{windows}, \quad (3)$$

with the averaged sensor value being 2 byte, the number of values in a window being 1 byte and $\# \text{windows}$ describing the total amount of windows to be sent for one gesture.

2) *Derivative-Based Prediction (DBP)*: DBP [24], [25] predicts sensor values by creating a linear model during a *learning phase*. During this learning phase, nodes collect sensor values according to a fixed window size and afterwards compute

the derivative between the average of the first and last so-called edge points of the window size. The derivative together with the last sensed value forms the model, which is sent to the edge cloud. After the learning phase follows the actual *prediction phase*, where the nodes use the created model to predict incoming sensor values. A node will not send data as long as the sensed value is within an error threshold of the predicted one. Otherwise, a new model is calculated based on the last acquired values according to the window size from the learning phase.

The error threshold in DBP is computed as the bigger value of an absolute error threshold and $\frac{\text{data point}}{100}$ times a relative error threshold. Window size, edge points, absolute and relative error threshold are thus tuneable parameters in DBP.

DBP assumes that nodes transmit their sensor values periodically to a sink node. If a node suppresses an update and hence a message does not arrive within a certain time at the sink, the sink node can implicitly derive a predicted value. However, as we aim at transmitting only a single IEEE 802.15.4 packet, we need, in addition to the derivative and the last sensed value, a counter that describes how many values the last update accounts. This counter is thus similar to the window size in APCA. The compressed size of a gesture in our modified DBP is thus computed as:

$$\text{compressed size} = (4 \text{ byte} + 2 \text{ byte} + 1 \text{ byte}) \cdot \#updates. \quad (4)$$

We use 4 byte for the derivative and 2 byte for the last data value while the counter is an unsigned 1 byte value. The total number of updates to sent is presented with $\#updates$.

IV. EVALUATION

In this section, we evaluate the presented data reduction methods. We find that all four methods are able to compress a hand gesture in one IEEE 802.15.4 packet with RI achieving the highest gesture classification accuracy of 93%.

A. Deep Learning-based Hand Gesture Recognition

Before evaluating the data reduction methods presented in Section III, we train a deep learning model, running on an edge cloud.

Collecting training data. For collecting training data, we use a Tmote Sky node and sample light values every 8 ms. The raw sensor data is continuously streamed without compression via a serial connection from the node to the edge cloud. For collecting data to train our deep learning model, we perform the gesture start detection at the edge cloud. We run a python script on the edge cloud that reads the light sensor values and detects the start of a gesture based on the MAD also used by Kaholokula [13]. After the start of a gesture has been detected, we collect 400 data points, generate plots, similar to the ones shown in Figure 1, and use these plots to train our model.

Training the deep learning model. We use transfer learning based on a pre-trained deep neural network trained for IMAGENET. In this work, we do not focus on the development of state of art deep neural network architectures. Instead, we

choose already well defined architectures. During training, we tested most of the architectures available in the FASTAI library. Among those, ResNet-18 performed best and hence we used it to train our model and to later classify gestures based on the decompressed sensor readings transmitted by the nodes. To avoid overfitting, we used only a few epochs: we first trained the last layers of the model with 5 epochs and then the entire model with 3 epochs.

For training the model, we collect 112 samples per gesture performed with left and right hands and with different light intensities and distances between hand placement and sensor. In this paper, we do not focus on the tradeoff between the amount of data required and the accuracy of the gesture classification. **Testing the model.** For testing our model and as an additional means to avoid overfitting, we have a separate validation set containing 48 samples for each gesture. Using the validation set our model achieves a gesture detection accuracy of 95%.

B. Evaluation Methodology

We follow a quantitative experimental methodology to evaluate our methods for the aforementioned hand gesture detection application.

Collecting test data. To create a test set we collect 10 samples per gesture from 13 persons. Each person is shown the gestures and then the person is asked to perform the gestures according to their interpretation. This data is collected over the course of a week at two locations of our office environment to test for different ambient lighting conditions. The procedure for collecting the test data is equivalent to the one for collecting the training data. We feed these in total 390 gesture traces into our previously trained model. However, we keep only the 310 gesture traces that were classified correctly in our test set. Our collected samples have discrepancies mainly due to sensor inaccuracy and mistakes of the user in performing the gesture. Hence, we filter out these wrong samples. We use this test set to provide a coherent and fair comparison of the different data reduction methods.

Setup. We implement each data reduction method and its corresponding decompression method in Python. The gesture traces in the test set are then passed to each data reduction method. We collect the resulting decompressed plots and the compressed size for comparing the data reduction methods.

Metrics. We investigate the trade-off between the *compressed size* of a gesture that needs to be transferred and the *accuracy* in classifying the correct gesture. The gesture classification accuracy is dependent on the gesture detection accuracy. But the gesture detection accuracy is a function of our used gesture detection algorithm [13], and evaluating the latter is not the purpose of this study. Hence, we focus on the gesture classification accuracy. We compute the compressed size for each gesture according to Equations 1–4. The accuracy for a particular gesture is computed as:

$$\text{accuracy}_{\text{gesture}} = \frac{N_{\text{gesture}}}{M_{\text{gesture}}}, \quad (5)$$

where N_{gesture} is the number of traces for that particular gesture that are correctly classified after decompression and

$M_{gesture}$ is the total number of traces for that particular gesture in the test set.

C. Comparing Data Compression Methods

We first compare the classification accuracy of the compression methods and find that RI performs best with 93% correctly classified gestures.

Figure 3 shows the confusion matrices of the hand gestures for each method. RI achieves with 93% a high classification accuracy across all gestures. FIP classified ² 87% of the gestures correctly with a tendency of wrongly classifying a swipe as a one tap. This can be attributed to high noise levels in the test data traces, which causes FIP to confuse the slope of noise with the slope of an inflection point. APCA and DBP have a higher bias for one of the gestures. APCA classified 73% of the gestures correctly and DBP achieved the lowest gesture classification accuracy with 70% and also the tendency of confusing a one tap as a swipe.³ Both, APCA and DBP, are designed for slow changing signals, and thus quick changes in a signal – as present in shadow-based hand gestures – result in high deviations from the original individual data points.

To compare the data compression methods, we have set the tuning parameters to the best measured configuration with the constraint that the compressed gesture should fit in one IEEE 802.15.4 packet⁴. In the following section, we investigate the impact of the individual tuning parameters for each data reduction method in more detail.

D. The Impact of the Tuning-Parameters

We now investigate the impact of the individual tuning parameters of the data reduction methods on the accuracy and the compressed size. The tuning parameters mainly determine how much an individual approximated data point of a gesture can deviate from the original one. We find that all methods are able to fit a hand gesture in one IEEE 802.15.4 packet while being able to fine-tune the accuracy.

Figure 4 shows for each method the compressed size in the upper plot and the achieved accuracy in the lower plot. The plots show the average for each gesture.

RI (Figure 4a), FIP (Figure 4b), and APCA (Figure 4c) show a decrease in the compressed size as the data tuning parameter allows for more deviation of the approximated individual data points from the original data points. The absolute error threshold heavily affects the compressed size for DBP as shown in Figure 4d. Due to frequent changes in the signal, DBP is not able to compress the data with lower absolute error threshold⁵ but still generates frequent updates which include the derivate, the sensor value and the counter (which is small).⁶ The high

compressed size for all other tuning parameters is skewed by these cases with lower absolute error thresholds. However, all methods achieve a compressed size of 90 byte and smaller by fine-tuning the method parameters. More precisely, we assume a packet overhead of 30-40 bytes for each IEEE 802.15.4 packet with IPv6 connectivity. A size of 90 byte and less allows, hence, to fit the compressed data points of a gesture to fit in one packet.

There is a general trend for all methods that the accuracy, shown in the lower plots of Figure 4, decreases the more the approximated data points for a gesture deviate from the original one. One reason is that interesting features of a gesture may be removed when approximating data points, resulting in a wrong classification by the deep learning model. However, the accuracy for a two tap and a one tap in RI, and FIP as well as for a swipe in DBP increases with more freedom in deviation to the original data point. We found that removing irrelevant features, especially in noisy data, helps in highlighting the interesting features of this gesture, and thus allowing for a correct classification.

V. GENERALIZATION

We further show the generality of our compression methods with an audio-based interaction application scenario. As we concentrate on the applicability of our compression methods, we do not consider audio-specific compression methods. In this case, the gesture detection is performed on audio samples captured by a microphone following the same principles of our shadow-based hand gesture detection. The audio data is captured on the sensor node. Once a gesture has been detected by the MAD algorithm, we compress the samples using our compression methods and send it to the edge cloud to perform classification.

Audio gestures. We consider three audio gestures: tap, snap and clap. A tap is the sound produced by a person tapping on a surface. A snap is the sound produced by a person creating clicking sound by building tension between a finger and the thumb, followed by forcefully bringing the finger down on the palm. A clap is the sound produced by a person doing a single clap. Figure 5 shows the corresponding audio signatures for these gestures from the microphone-based on a sampling rate of 8 kHz. The figures show that the audio signatures for the tap (Figure 5b) and the clap (Figure 5c) are very similar, with the clap having higher frequency components. We found that 400 data points are sufficient for our deep learning model to accurately classify between the three gestures.

Collecting training data. We use a Silicon Labs Thunderboard Sense 2 node for collecting the audio samples. The raw audio samples are streamed from the node to the edge cloud via a serial connection. For collecting the training data, we follow the same process as for our light-based gesture detection application.

Training the deep learning model. During training based on transfer learning, we tested most of the ImageNet architectures available in the FASTAI library. Among these, ResNet-34 performed best and hence we use this architecture for our model. Similar to our previous approach, we try to avoid

²Thiemo: here we suddenly have past tense (classified not classifies). Good to be consistent

³Thiemo: I think “as” could be replaced by “with”. But maybe rewrite the whole sentence “DBP shows a tendency to confuse a one tap with a swipe”? BTW, I think we mix past and present tense here, see above

⁴The selected tuning parameters are: RI: rate of data points to keep = 8, FIP: rate of change threshold = 0.4, APCA: relative error threshold = 5.0%, DBP: absolute error threshold = 9, relative error threshold = 2.5%, edge points = 3, window size = 6

⁵Thiemo: maybe “a lower.” or “thresholds”?

⁶Thiemo: I would put that instead of which if you refer to counter

Fig. 3: Comparison of classification accuracy.

Fig. 4: Impact of tuning parameters on compression size and accuracy.

overfitting by using only a few epochs: we first trained the last layers of the model with 5 epochs and then the entire model with 10 epochs.

For training the model, we collect 300 samples, with clap and snap being performed with the microphone turned towards and away from the audio source. For tap, we increase the robustness of the model by tapping on different kinds of surfaces: surfaces (e.g. cushions) that absorb the sound waves as well as surfaces (e.g. wooden tables) that vibrate and as a result have a longer decay time

Testing the model. For testing our model and as an additional means to avoid overfitting, we have a separate validation set containing 20 samples for each gesture. Using the validation set, our model achieves a gesture detection accuracy of 95%.

A. Validating the generality of our methods

We use the same evaluation methodology as for the light-based gesture detection application. In this case, we do not collect test samples, but rather work on our training samples. We feed our 300 collected samples into our previously trained model, of which 290 are classified correctly. We use this test set of 290 samples for our data compression methods. We use our previously implemented combination of data compression and decompression methods in Python. Due to space constraints, we focus on the performance of RI and FIP, as they were developed for interaction-based applications.

We first measure the classification accuracy of our compression methods and find that RI and FIP achieve classification accuracies of 88% and 81%, respectively, for their best

Fig. 5: Audio signatures for different gestures. Clap and Tap have similar slowly decaying audio signatures, with clap having higher frequency components. Snap has a sharp high frequency peak.

Fig. 6: Classification accuracy for audio-based gesture recognition.

measured configuration. Figure 6 shows the confusion matrices for audio-based gesture recognition for our methods. RI tends to misclassify a snap as a clap, which can be attributed to the possibility of it skipping the sharp peak in the audio signature of the snap gesture. In the case of FIP, we get the most misclassifications for the clap gesture. FIP compresses by under-sampling the audio sample series. A clap is characterized by its high frequency component. Removal of these high frequency components leads to the misclassification. This is shown by the sharp decrease in the classification accuracy for both of our methods in Figure 7

Fig. 7: Impact of tuning parameters on compression size and accuracy of audio-based gesture detection system.

We also investigate the effect of tuning parameters on the accuracy and the compressed size of the gestures. Figure 7 shows for each method, the compressed size in the upper plot and the achieved accuracy in the lower plot.

Both methods show a decrease in the compressed size with the tuning parameter allowing more deviation of the approximated data points from the individual data points. RI and FIP can compress the data by 50% and 75%, respectively, and achieve a reasonable accuracy. However, compressing these gestures into a single packet would result in considerable loss in accuracy. FIP is unable to compress the clap and snap gestures into a single packet for our measured tuning parameters.

Using the DBP algorithm with the same tuning parameters configuration for compressing these audio signals, leads to 100% accuracy. With DBP, however, the compressed size is larger than the original size. Due to the frequent changes in the signal, DBP is not able to compress the data but still generates updates that include the derivative, the sensor value and the counter (which is 1). Thus, the sampled signal almost equals the compressed signal.

B. Discussion

The nature of the two forms of sensor data considered in our experiments are very different. The audio signature has a oscillating nature with longer series at a much higher frequency. Our methods show that they can perform well, achieving an accuracy of over 80% without much changes to even the tuning parameters. The compressed size for our audio-based gesture recognition system is much larger than for our light-based system. The long oscillating nature of the audio signatures makes it difficult to get good compression ratios. Nevertheless, our methods are able to achieve reasonable compression ratios of 2:1 and 4:1 with only a small loss of accuracy.

VI. RELATED WORK

As mentioned in the introduction, our work is related to efforts that reduce data to be transmitted such as compression [21], [22], [34], in-network processing [15], [19], [30], and data prediction algorithms [24], [26]. While these methods aim for high accuracy between sensed values and approximated values in low sampling frequency applications, we target high sampling frequency applications and only keep the interesting features of the sensed values.

Alippi et al. port deep learning to embedded systems [1]. Their target platforms are not as resource-constrained as our platform that consumes less power for “sense-store-send-sleep” applications [14]. Such platforms are still widely used in industry [6].

Varshney et al. present a visible light sensing system for hand gesture recognition [33]. Their focus is on making the system battery-free. Ma et al. present SolarGest [18], a system that uses a transparent solar cell for efficient gesture recognition. They focus on both energy savings and signal processing to improve classification accuracy. Our work is complementary in that we aim to reduce the amount of sensed data that needs to be transmitted. Li et al. present Aili [17], a system for reconstructing hand poses with visible light. They use a grid of low-cost photodiodes to identify 3D hand skeleton poses in real-time. The communication is not done via a wireless system and hence they do not aim at reducing the amount of data they transmit. The same is true for GestureLight [13] and Huan et al.’s system [5]. The latter focuses on accuracy rather than resource-efficient communication.

Oyedotun et al. use vision-based deep learning to differentiate between the hand gestures from the American Sign Language [20]. They work with static images of hand gestures while we convert one-dimensional light samples caused by hand gestures to 2-dimensional pictures. Oyedotun et al.’s work is similar to many others that try to classify static images [23] or image sequences [2], [12].

VII. CONCLUSIONS

In this paper, we address the challenge of data reduction for interaction-based applications. On the example of shadow-based hand gesture recognition, we demonstrate four different methods that reduce the amount of light values needed to describe a single hand gesture. In particular, we present two methods for reducing data, RI and FIP, and compare our proposed methods with APCA and DBP, two state-of-the-art compression methods. Our evaluation shows that we can fit a shadow-based hand gesture into one IEEE 802.15.4 packet while maintaining a high classification accuracy. In addition, we generalize our results by applying our methods to audio-based interactions. While audio signatures differ substantially from light sensor data, our methods nevertheless achieve reasonable compression ratios at a low loss of accuracy.

ACKNOWLEDGEMENT

This work has been (partially) funded by the H2020 EU/TW joint action 5G-DIVE (Grant #859881) and the Swedish Foundation for Strategic Research (SSF).

REFERENCES

- [1] C. Alippi, S. Disabato, and M. Roveri. Moving convolutional neural networks to embedded systems: the alexnet and VGG-16 case. In *ACM/IEEE IPSN*, 2018.
- [2] M. Asadi-Aghbolaghi, A. Clapés, M. Bellantonio, H. J. Escalante, V. Ponce-López, X. Baró, I. Guyon, S. Kasaei, and S. Escalera. Deep learning for action and gesture recognition in image sequences: A survey. In *Gesture Recognition*, pages 539–578. Springer, 2017.
- [3] K. Chakrabarti, E. J. Keogh, S. Mehrotra, and M. J. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM TODS*, 27, 2002.
- [4] DeepSig. Reinventing Wireless with Deep Learning. [Online; accessed 2019-09-15].
- [5] H. Duan, M. Huang, Y. Yang, J. Hao, and L. Chen. Ambient light based hand gesture recognition enabled by recurrent neural network. *IEEE Access*, 8:7303–7312, 2020.
- [6] J. Eriksson, N. Finne, N. Tsiftes, S. Duquennoy, and T. Voigt. Scaling RPL to Dense and Large Networks with Constrained Memory. In *EWSN*, 2018.
- [7] fast.ai. Making neural nets uncool again, 2019. [Online; accessed 2019-09-5].
- [8] K. Fujinami. Interaction Design Issues in Smart Home Environments. In *IEEE FutureTech*, 2010.
- [9] G. Hernandez, O. Arias, D. Buentello, and Y. Jin. Smart Nest Thermostat A Smart Spy in Your Home. In *Black Hat USA*, 2014.
- [10] N. Q. V. Hung, H. Jeung, and K. Aberer. An Evaluation of Model-Based Approaches to Sensor Data Compression. *ACM TKDE*, 25(11), 2013.
- [11] J. Jeong. A Study on the IoT Based Smart Door Lock System. In K. J. Kim and N. Joukov, editors, *ICISA*, 2016.
- [12] V. John, M. Umetsu, A. Boyali, S. Mita, M. Imanishi, N. Sanma, and S. Shibata. Real-time hand posture and gesture-based touchless automotive user interface using deep learning. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 869–874. IEEE, 2017.
- [13] M.-D. Kaholokula. Reusing Ambient Light to Recognize Hand Gestures. Technical report, Dartmouth College, 2016. TR2016-797.
- [14] J. Ko, K. Klues, C. Richter, W. Hofer, B. Kusy, M. Bruenig, T. Schmid, Q. Wang, P. Dutta, and A. Terzis. Low Power or High Performance? A Tradeoff Whose Time Has Come (and Nearly Gone). In *EWSN*, 2012.
- [15] O. Landsiedel, F. Ferrari, and M. Zimmerling. Chaos: Versatile and Efficient All-to-all Data Sharing and In-network Processing at Scale. In *ACM SenSys*, 2013.
- [16] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521, 2015.
- [17] T. Li, X. Xiong, Y. Xie, G. Hito, X.-D. Yang, and X. Zhou. Reconstructing Hand Poses Using Visible Light. *ACM IMWUT*, 1(3), 2017.
- [18] D. Ma, G. Lan, M. Hassan, W. Hu, M. B. Upama, A. Uddin, and M. Youssef. SolarGest: Ubiquitous and Battery-free Gesture Recognition using Solar Cells. In *ACM MobiCom*, 2019.
- [19] S. Nath, P. B. Gibbons, S. Seshan, and Z. Anderson. Synopsis Diffusion for Robust Aggregation in Sensor Networks. *ACM TOSN*, 4(2), 2008.
- [20] O. K. Oyedotun and A. Khashman. Deep learning in vision-based static hand gesture recognition. *Neural Computing and Applications*, 28(12):3941–3951, 2017.
- [21] S. Papadimitriou, A. Brockwell, and C. Faloutsos. Adaptive, Hands-off Stream Mining. In *ACM VLDB*, 2003.
- [22] N. D. Pham, T. D. Le, and H. Choo. Enhance exploring temporal correlation for data collection in WSNs. In *IEEE RIVF*, 2008.
- [23] S. P. Priyal and P. K. Bora. A robust static hand gesture recognition system using geometry based normalizations and krawtchouk moments. *Pattern Recognition*, 46(8):2202–2219, 2013.
- [24] U. Raza, A. Camerra, A. L. Murphy, T. Palpanas, and G. P. Picco. What does model-driven data acquisition really achieve in wireless sensor networks? In *IEEE PerCom*, 2012.
- [25] U. Raza, A. Camerra, A. L. Murphy, T. Palpanas, and G. P. Picco. Practical Data Prediction for Real-World Wireless Sensor Networks. *ACM TKDE*, 27(8), 2015.
- [26] B. Risteska, D. Solev, and D. Davcev. Data prediction in WSN using variable step size LMS algorithm. In *SENSORCOMM*, 2011.
- [27] E. Ronen and A. Shamir. Extended Functionality Attacks on IoT Devices: The Case of Smart Lights. In *Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016.
- [28] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Noguees, J. Yao, D. Mol-lura, and R. M. Summers. Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. *IEEE TMI*, 35(5), 2016.
- [29] G. Sowjanya and S. Nagaraju. Design and implementation of door access control and security system based on IOT. In *ICICT*, 2016.
- [30] C. Stylianopoulos, M. Almgren, O. Landsiedel, and M. Papatriantafyllou. Continuous Monitoring meets Synchronous Transmissions and In-Network Aggregation. In *DCOSS*, 2019.
- [31] P. Sundström and K. Höök. Hand in hand with the material: designing for suppleness. In *SIGCHI CHI*, 2010.
- [32] E. M. Tapia, S. S. Intille, and K. Larson. Activity Recognition in the Home Using Simple and Ubiquitous Sensors. In *Pervasive Computing*, 2004.
- [33] A. Varshney, A. Soleiman, L. Mottola, and T. Voigt. Battery-free Visible Light Sensing. In *ACM VLCS*, 2017.
- [34] Z. Xiong, A. D. Liveris, and S. Cheng. Distributed Source Coding for Sensor Networks. *IEEE SPM*, 21(5), 2004.

- [35] M. Ye, N. Jiang, H. Yang, and Q. Yan. Security analysis of Internet-of-Things: A case study of august smart lock. In *IEEE INFOCOM WKSHPS*, 2017.