FabToys: Plush Toys with Large Arrays of Fabric-based Pressure Sensors to Enable Fine-grained Interaction Detection

Ali Kiaghadi
University of Massachusetts Amherst
Amherst, MA, USA
akiaghadi@umass.edu

Jin Huang
University of Massachusetts Amherst
Amherst, MA, USA
jinhuang@cs.umass.edu

Seyedeh Zohreh Homayounfar
University of Massachusetts Amherst
Amherst, MA, USA
shomayounfar@umass.edu

Trisha Andrew
University of Massachusetts Amherst
Amherst, MA, USA
tandrew@umass.edu

Deepak Ganesan
University of Massachusetts Amherst
Amherst, MA, USA
dganesan@cs.umass.edu

ABSTRACT

Recent advances in fabric-based sensors have made it possible to densely instrument textile surfaces on smart toys without changing their look and feel. While such surfaces can be instrumented with traditional sensors, rigid elements change the nature of interaction and diminish the appeal of plush toys.

In this work, we propose FabToy, a plush toy instrumented with a 24-sensor array of fabric-based pressure sensors located beneath the surface of the toy to have dense spatial sensing coverage while maintaining the natural feel of fabric and softness of the toy. We optimize both the hardware and software pipeline to reduce overall power consumption while achieving high accuracy in detecting a wide range of interactions at different regions of the toy. Our contributions include a) sensor array fabrication to maximize coverage and dynamic range, b) data acquisition and triggering methods to minimize the cost of sampling a large number of channels, and c) neural network models with early exit to optimize power consumed for computation when processing locally and autoencoder-based channel aggregation to optimize power consumed for communication when processing remotely. We demonstrate that we can achieve high accuracy of more than 83% for robustly detecting and localizing complex human interactions such as swiping, patting, holding, and tickling in different regions of the toy.

CCS CONCEPTS

• **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**; *Gestural input*.

KEYWORDS

interaction detection, ubiquitous sensing and computing, smart toys

MobiSys '22, June 25–July 1, 2022, Portland, OR, USA

ACM ISBN 978-1-4503-9185-6/22/06…$15.00
https://doi.org/10.1145/3498361.3538931

ACM Reference Format:
Ali Kiaghadi, Jin Huang, Seyedeh Zohreh Homayounfar, Trisha Andrew, and Deepak Ganesan. 2022. FabToys: Plush Toys with Large Arrays of Fabric-based Pressure Sensors to Enable Fine-grained Interaction Detection. In *The 20th Annual International Conference on Mobile Systems, Applications and Services (MobiSys '22), June 25–July 1, 2022, Portland, OR, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3498361.3538931

1 INTRODUCTION

Stuffed toys are often a child's first friend and play an important role in a child's cognitive, physical and emotional development. Stuffed toys are also very important for building social skills through pretend play and role playing. For example, when a child grooms or feeds a stuffed toy, they mimic everyday interactions which then transition into a social world. Through the process of caring for a stuffed toy, they also build empathy and kindness. Such interactions also play an important role in language skills since children act out stories and scenarios with their toy.

While highly instrumented stuffed toys can be useful for parents and experts to observe and understand how children are developing in their natural environment, a challenge is how to incorporate sensing elements in smart toys without changing their feel and texture. Despite significant development in high-tech toys that incorporate a variety of sensors and actuators, they lose the soft feel and touch due to the need for rigid sensor elements to be placed near the surface of the toy. As a result, they are not as attractive to children who are drawn to the softer and more squishy plush toys [8]. To compromise, smart toy manufacturers usually only place a small number of sensors on the surface of the toy, typically only one or two. This in turn diminishes the ability to measure fine-grained interaction with different regions of the toy.

Recent works on fabric-based sensors [10, 11, 13, 18, 42–44] present a promising alternative to overcome traditional limitations in designing instrumented soft toys. These sensing techniques use familiar garments made of cotton and silk thread, and imperceptibly adapt them to enable sensing of pressure and touch signals to yield natural fabric-based sensors. In addition, fabric-based sensors have improved in their sensitivity which now make it possible to instrument soft toys with such sensors beneath the surface of the felt on the toy, thereby making them even less obtrusive.

These advances present a new possibility to create highly instrumented toys that integrate a large number of sensors to measure

fine-grained and spatio-temporally complex interactions. In contrast to interaction with most rigid toys, interaction with soft toys is much more complex often involving both hands and body contact. These interactions can range from holding, patting, or tickling the toy while simultaneously squeezing or holding it. The high-dimensional nature of the interaction possibilities and methods, dynamic pressure intensity based on the type of interaction and the location it is applied to, and varying sensing region surface area require an equivalently high degree of instrumentation of the toy in order to accurately determine the type of interaction and makes single-threshold based pressure sensing methods obsolete. As a result, conventional binary buttons fail to track such complex interactions, while rigid pressure sensors change the feel of the toy and can interrupt natural behavior with the toy.

In this paper, we describe an end-to-end hardware and software design of such a highly instrumented soft toy, FabToy (Fabric-enhanced Toy), and how we tackle the myriad challenges at the different system design layers. In designing FabToy, we tackle engineering and design challenges across the stack from optimization of form-factor, sensor data acquisition, low-power analytics and low-power communication.

From a sensor design perspective, FabToy is designed to comprehensively capture a range of complex interactions that are expected with smart toys by equipping it with a dense array of two dozen fabric-based pressure sensors to enable fine-grained sensing while covering the majority of the interaction surface of the toy. The sensor array is embedded under the felt surface of the toy such that it is fully invisible and imperceptible to the user. To ensure high sensitivity despite the fact that the sensors are placed beneath the felt surface, the sensors are optimized by reducing impedance using an ionized solution to ensure that it can capture key interactions. In addition, sensor conditioning circuits are designed to have high dynamic range by exposing both amplified and unamplified channels thereby providing high signal to noise ratio while capturing both gentle and rough interactions with the toy.

From a hardware design and data acquisition perspective, we design an ultra-low power and small-factor hardware that is placed deep within the toy to amplify, filter, and acquire signals from 48 sensor channels (two channels per sensor). We optimize the amplification and data acquisition circuits using low-power analog multiplexers and optimized sampling to acquire data from 48 analog channels simultaneously with very low power consumption while rejecting common noise sources like powerline interference.

In terms of data analytics, we consider both local and remote processing — *local processing* would be suitable when the smart toy is intended to execute autonomously without requiring connection to an external device or the Internet, and *remote processing* allows it to stream data to a phone or computer to enable a broader range of web-based interactive storytelling applications. By efficiently supporting both modes, we provide maximal flexibility in terms of use-cases of the smart toy.

To optimize local processing, we utilize machine learning to fuse the sensor data from the large number of sensor channels to classify simple and spatio-temporally complex interactions with the smart toy, as well as localize these interactions. We optimize this model to deal with several issues such as cross-talk between sensor elements as well as other confounders, while also ensuring that it is lightweight enough to fit within the resource constraints of a low-power microcontroller. To achieve this, we design a resource-aware Convolutional Neural Network model with early exit at intermediate layers such that overall computational overhead can be minimized.

To optimize the remote processing pipeline, we compress the multi-channel data by leveraging correlations between the different streams and transmit this over a BLE radio. The aggregation technique is an auto-encoder that aggregates streams that have similar data to reduce transmission overhead. The remote model can be more resource-intensive than the local model since the remote device has more resources.

Our end-to-end implementation and evaluation shows that

- FabToy can classify single interactions with accuracy of 86% and complex interactions with accuracy of 83%, and this is better than several alternative resource-constrained ML models. We show that this accuracy can further increase to 92-94% for medium-grained and coarser-grained classification.
- In case of local processing, the use of early exit reduces processing power consumption by 45% while losing only 4% accuracy, enabling embedded signal processing on a low-power microcontroller for real-time classification and interaction.
- In case of remote processing, we show that dynamic channel dimension reduction using an auto-encoder reduces transmission power consumption over a BLE radio by 43%, while sacrificing only 2% accuracy.
- We implement the full FabToy system from sensor to processor to radio. Our hardware power and latency benchmarks on the nRF52840 low-power microcontroller with BLE radio show that our implementation is lightweight and can be executed with low delay and low power consumption and is practical for real-world deployment.

2 FABTOY HARDWARE DESIGN

The hardware architecture of FabToy designed to achieve three main goals. First, the final prototype needs to look and feel identical to a typical plush toy. Second, we wish to achieve complete coverage across the toy to be able to capture a wealth of complex interactions across different locations of the toy. Third, we wish to ensure that we obtain high signal quality across a range of pressures applied during interaction, from very gentle pressure during tickling to moderate pressure during rubbing to orders of magnitude higher pressure while squeezing the toy.

We now describe the fabric-based sensors that we use and the design of the sensor array in FabToy.

2.1 Fabric-based Sensors on the FabToy

FabToy is designed to look and feel identical to a typical plush toy to ensure that children do not alter their behavior towards using the toy. To achieve this, we need sensors to be placed beneath the felt surface of the smart toys and need to be imperceptible in terms of look and feel of the toy. This means that the sensors need to be extra-sensitive to capture to a range of interactions from gentle tickling to squeezing despite being under the felt surface.

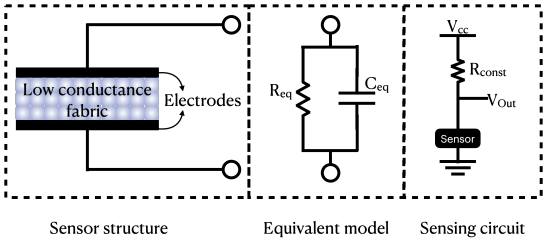


Figure 1: Fabric-based pressure sensor used in FabToy The sensor responds to changes in pressure by changing its resistance which can be measured using a voltage divider.

Towards this end, we designed FabToy by leveraging recent developments in textile-based sensors. The fabric pressure sensors are implemented using a design presented in [14, 18]. The sensors' structure, electrical equivalent, and measurement method is shown in Fig. 1. The sensors detect pressure applied on two electrodes by changing the resistance of the layer within.

Since typical fabrics present very high resistance, we lowered the resistance of the middle layer by coating the fabric with an ion-conductive polymer that provides ion-conductivity to its fibers. As a result, the resistance of the fabric can be lowered by multiple orders of magnitude and increases the sensitivity by introducing more conductive paths between electrodes.

The use of fabric as the sensing substrate enables the sensors to conform to the shape of the toy to maximize sensing efficiency in the region of interest. In contrast, traditional force sensors are usually point sensors due to their rigid nature; even if flexible force sensors are used, they are not capable of bending in multiple directions, thus changing the feel of the toy. In addition, flexible and push-button sensors are more susceptible to mechanical aging and wearing out. Additionally, fabric sensors provide the flexibility to fit to odd 3D shapes such as noses and ears; this is an advantage we utilized in designing FabToy

2.2 Array of Fabric-based Sensors

To achieve fine-grained sensing of interaction, the toy needs to be covered with a large number of sensors so that we can detect which part of the toy is being interacted with and the specific type of interaction. High spatial fidelity is particularly necessary given that interactions with soft toys tend to involve both hands and sometimes also the torso (e.g. while hugging the toy). A single large sensor that covers a large surface of the toy would only capture the overall pressure on the toy, so we would lose both information about the location of interaction (e.g. stomach vs leg) and information about more nuanced interactions such as tickling a toy with one hand while holding the toy with the other.

To achieve high spatial fidelity, FabToy has 24 sensors placed at strategic locations such as hands, arms, feet, ears, stomach, nose, and some other locations that might typically undergo various interactions from children. Figure 2 shows a high-level view of the multi-sensor array used in FabToy.

Amplified and unamplified sensor streams: while designing FabToy, we consider dealing with the wide variation in pressures across which we need to obtain a good signal. Soft toys can be handled in a variety of ways from very gentle pressure to tight squeeze and this results in a few orders of magnitude change in the signal strength.

While increasing the sensitivity of the fabric-based sensor achieves some of this goal, it is not sufficient by itself. Therefore, we also split each textile sensor into two sensor streams — an unamplified stream to deal with medium to tight handling and an amplified stream for very gentle handling. The amplified data stream uses a band-pass filter to increase the signal to noise ratio (SNR) and helps us acquire very weak signals such as during tickling, whereas the unamplified streams helps us acquire large signals such as squeezing hard or strong swiping actions.

Analog multiplexing the sensor channels: The large number of channels (48 in total across amplified and unamplified streams) introduces a number of downstream challenges in sampling and processing the signal. At the hardware level, we need to deal with the fact that typical microcontrollers used on low-power devices have only a few Analog to Digital Converters (ADCs). However, we can take advantage of the fact that we do not need very high sampling rates for the individual sensors.

The data acquisition pipeline is shown in Figure 2. We start with a voltage divider that senses the resistance of the pressure sensors. The output voltage represents the pressure applied on the fabric surface. Then, we use analog multiplexers in our design to uniformly sample all the channels using control signals issued by the microcontroller.

While low sampling rates can suffice to capture the interactions of interest, we are limited by the need to filter powerline noise. While power line noise is typically not large for rigid electronics force sensors that have a very small surface area, the large surface area and relative large sensor impedance of textile sensors in FabToy make them very good receptors for electromagnetic noise. We use a sufficiently high sampling rate (160 Hz per channel) to be able to filter out powerline noise. We then apply a simple moving average filter inside the microcontroller with a cut-off frequency of $\approx 12\ Hz$ to remove powerline interference. Since the frequency of even the faster interactions like tickling is well below 12Hz, this allows us to retain the signal of interest while removing noise.

Hardware power consumption: To keep the electronics' power consumption as low as possible, we carefully chose ultra-low power Op-Amps, regulators, and analog multiplexers that, in combination, consume two orders of magnitude lower power than a low-power microcontroller.

2.3 Computational Challenges

The choices made in our FabToy hardware impacts the design of the downstream modules in several ways.

Large number of channels The overarching problem is that the large number of sensor channels increases computation and communication overhead. Even though each channel is sampled at a low rate, the fact that we have 48 channels makes the cumulative sampling rate quite high, which increases overhead for downstream both analytics on the low-power microcontroller and communication via a low-power radio.

Cross-talk between sensors In contrast to electronic rigid force sensors, the fabric pressure sensor we use are double-sided, i.e. they

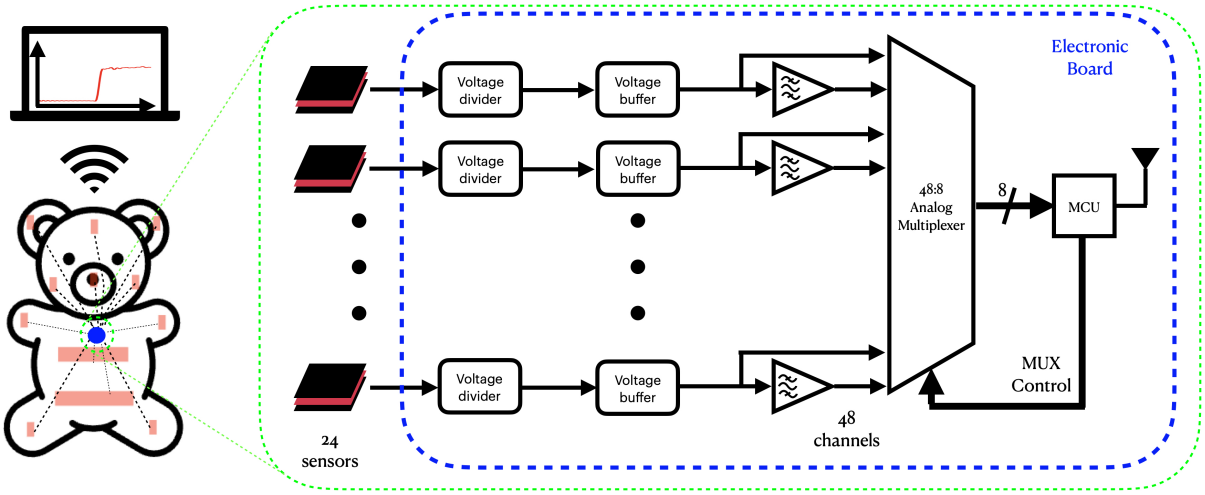


Figure 2: High level hardware design in FabToy

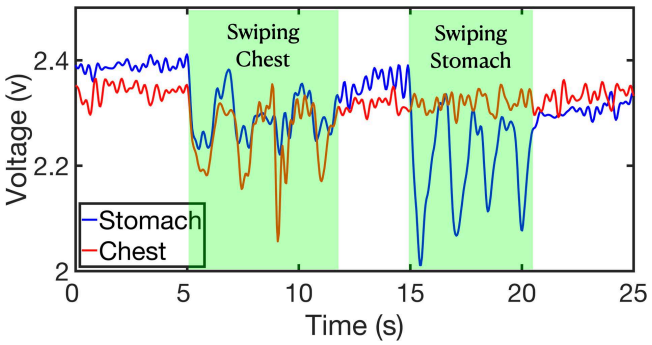


Figure 3: The data streams from stomach and chest sensors during four repetitive swiping interaction on each of these sensors. As can be seen in the figure, the stomach sensors captures pressure changes during swiping the chest.

sense pressure applied from both sides of the sensor. As a result, they are also affected by internal movements of the toy stuffings, which might be caused by interactions with other sensors. Cross-talk also occurs when interaction with one sensor leads to pressure being applied on other sensors. For instance, the close proximity of nose and mouth sensors will lead to interactions with the toy's nose to cause changes in the mouth sensor as well.

Cross-talk also occurs during complex interactions that involve holding the toy with one hand while interacting with the other hand. Since the sensors are interconnected through the toy, there are cross-talk between the two sensors involved in the interaction i.e. anchor sensor that is being held and the interaction sensor. As a result, while the child is performing an interaction, a reverse force is applied to the anchor sensor as well. For instance, holding the hand while swiping the forehead can lead to the swiping signal being visible at the hand sensor as well.

An example of cross-talk is presented in Fig. 3. We see that the chest and stomach sensors clearly pick up their corresponding interaction signals, however, the stomach sensor is also affected by swiping the chest.

Effect of Humidity A second challenge is that fabric-based pressure sensors are affected by humidity as well as pressure i.e. these sensors will have reduced resistance in a humid environment, similar to being under pressure. As a result, the output baseline will depend on base pressure as well as humidity.

3 DATA ANALYTIC PIPELINE

In this section, we present building blocks constructing the data analytic pipeline in FabToy. The aim of data analytic pipeline is to overcome the data volume and cross-talk challenges presented in Section 2.3 to achieve high accuracy while optimizing power consumption.

3.1 Signal processing pipeline overview

The overall computational pipeline with the local and remote processing branches is shown in Figure 4. The initial stage is a triggering stage wherein the sampled data from 48 analog channels are fed into a trigger block to ignore *idle* states when no interaction is happening with the toy.

Once interaction is detected, we have two possible downstream pipelines depending on whether the data is locally or remotely processed.

Local processing pipeline: The first scenario is locally processing on a low-power microcontroller to facilitate the design of a fully self-contained smart toy that does not need to interact with an external device in-order to operate. (While feedback is not the focus of this work, we envisage that tactile or auditory feedback can be incorporated into FabToy to enable such a smart toy).

Since we use TensorFlowLite [37] to build our model for microcontroller-class platforms, we focus on convolutional neural network models that are supported by this framework. For example, bidirectional LSTM layers are not supported by TensorFlowLite, so we do not use models that leverage temporal dependencies.

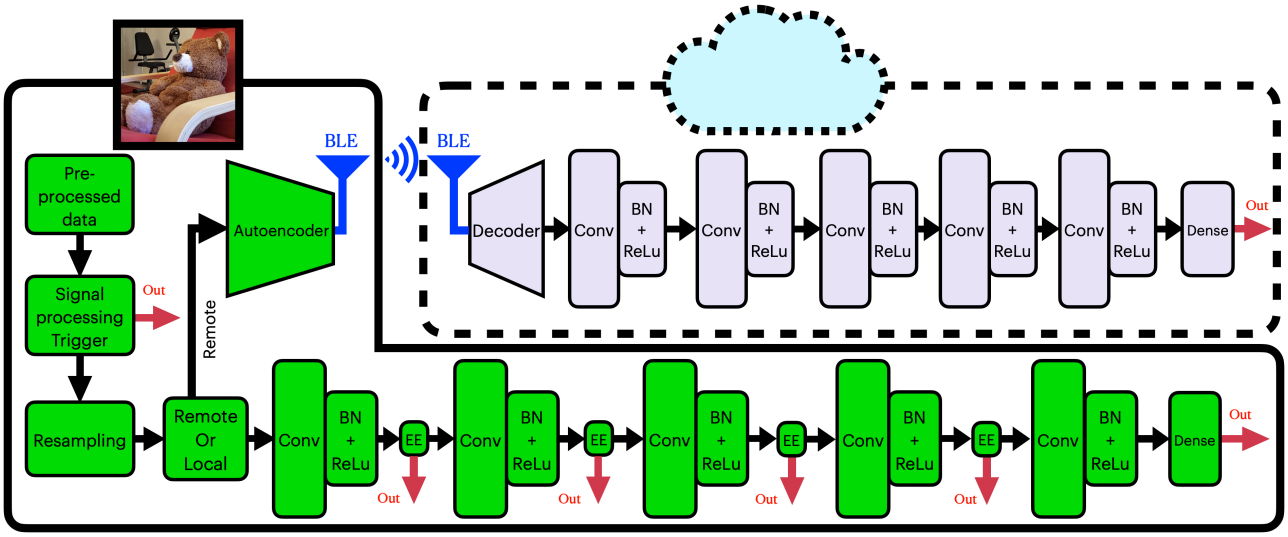


Figure 4: Structure of our implemented machine learning model in FabToy.

To reduce computational overhead of our model, the main key idea in this pipeline is to introduce *early exit blocks* between layers of our neural network to reduce computation time and power.

Remote processing pipeline: The second is remote processing, where raw data is transmitted to a smartphone-class device to offload computation and enable interaction that involves an external device. This can enable a range of digital applications where the smart toy may be part of a larger story-telling or educational platform.

Here, the key idea is to insert a *dynamic channel aggregation block* reduces the size of the data to be transmitted to reduce radio power consumption.

We explain these building blocks in more detail.

3.2 Wake-up trigger

Since interactions tend to be bursty, the first stage of our signal processing pipeline is a wakeup trigger to detect when the Fabtoy is in *idle* state where there is no interaction versus *active* state when there is interaction.

The triggering module is based on the intuition that any activities happening on the toy will cause signal distortions, especially, on the amplified signals. As a result, standard deviation of the amplified channels is a simple indicator of activity. Please note that some interactions cause very weak vibrations on the surface of the toy, which will be naturally filtered as the vibrations penetrate through the toy's stuffing, which is why an IMU-based triggering cannot detect such interactions and will cause false negatives.

To adapt this block to dynamic changes such as changes in ambient powerline noise, and the fabric resistance due to temperature and humidity (which affects the standard deviation of analog channels), we use a dynamic threshold based on summation of the standard deviations of all amplified channels, as shown in Equation 1.

$$S_{th} = \alpha \times \min_{t>0}\Big(\sum_{\forall ch \in Ch_{amp}} std(v_{ch})\Big) \quad (1)$$

Here Ch_{amp} is the set of 24 amplified channels and α represents the tuning coefficient. For each time window, the sum of standard deviation of voltage traces in amplified channels is compared with this threshold to decide whether to trigger a wakeup.

We note that this module would normally be implemented in analog to avoid the MCU having to wakeup and sample the channels. In our current implementation, we implement the module on the MCU for ease of prototyping.

3.3 Local Processing

The local processing model in FabToy consists of 5 layers of neural networks with batch norms and ReLu layers. To optimize computation power consumption, we introduce early exits in between neural network layers.

Convolutional layers: In our design, we apply and stack convolutional layers[19] to help learn the cross-talk between sensors. A single convolutional layer can only capture the patterns within a limited range which we call the *Receptive Field*. For example, a convolutional kernel of size three maps sensor data from three adjacent sensors into one data point of the feature map. However, cross-talk can extend to farther away sensors, particularly for complex interactions where it is difficult to precisely determine which sensors are being impacted by the pressure at the anchor and interaction locations.

Since the physically adjacent sensors cannot always be adjacent in the input matrix to the neural network, we stack convolutional layers in a multi-scale manner in our design so that the early layers with smaller receptive field can capture near-field patterns of the sensors that are adjacent in the data plane; the latter layers with larger receptive field could perceive the potential cross-talk relationship between the sensors that are far away in the data plane.

Batch norm: In order to overcome the humidity-related artifacts, we embed *Batch Norm* [16] layers in the feature extractor stage to standardize the internal features after each convolutional layers.

The Batch Norm layer not only contributes to the steadying of our training process by whitening the features and mitigates the biased distribution but also help rectify the biased input due to the humidity during the model inference time.

Local early exit: To reduce the computation time and energy cost, our computational pipeline endows the network with early exits, thereby allowing us to obtain predictions at intermediate stages in the pipeline. Early exit is a well-known strategy to optimize the computational efficiency of a neural network [20, 33]. The intuition is that the majority of data cases can be classified with only a few layers, and only a few more complex data cases require the entire deep learning pipeline.

Table 1 shows the structure of our model — the **Feature Extractor** consists of five blocks of 1-dimensional convolutional layers **Conv1-5** following by **Batch Norm** (BN) layer as well as a **Rectified Linear Unit**[22] (ReLU6), and a fully connected layer (Dense1).

We make the prediction based on 48× raw sensor data at the granularity of a second. The classification result will only be generated once within the given time window, which indicates the categories of the interaction and position for the given time period. In our experiments, we use a time window of 3 seconds with 2 seconds of overlap between windows.

3.4 Remote Classification Model

The remote model is similar to the local model except for two differences. First, we introduce a data reduction module to minimize communication overhead. Second, we remove the early exit modules since we do not have as stringent resource limitations at the remote device.

Dynamic channel aggregation: To reduce communication cost, we reduce the input dimension by down-sampling the time series data for all channels. We aggregate channels by using an auto-encoder that aggregates the streams to reduce the number of data streams to the desired value. The auto-encoder structure consists of a pair of encoder and decoder. In our setting, the encoder will be on the IoT devices to efficiently aggregate and encode the original streams into smaller size. While the the auto-encoder can work in an unsupervised manner, we jointly train the encoder and decoder together with our prediction pipeline for better performance. We manually assign the number of the streams that the auto-encoder should learn to aggregate during each training.

The number of output streams directly affects the size of required data to be transmitted through the radio, and as a result, directly reduces communication power consumption. The trade-off between reducing data streams and the drop in system's accuracy is studied in Section 5.3.

Remote computational model: The remote classification model that we use is similar to the local one (it has the same *convolutional* and *batch norm* layers), however, it does not have the early exit modules since compute resources are not as limited in the remote device.

We note that one of the major advantages of remote processing is the ability to leverage more complex models. For example, the local model does not maintain state over time but the remote model can take advantage of temporal context to place the current interaction within a larger interaction session. Our datasets focus currently on individual interactions (both simple and complex ones), hence we do not leverage such models, but these present a rich area for further exploration.

4 IMPLEMENTATION

In this section, we describe the implementation of our FabToy prototype.

We used a plush teddy bear to implement FabToy, shown in Fig. 5a. The figure also highlights the placement of 24 fabric sensors on the toy. We note that the fabric sensors are placed *underneath the felt*, therefore there is no change to the feel and texture of the exterior of the toy.

Fabric sensor design and placement: The textile pressure patches are made from 3 layers. The two outer layers are silverized nylon fabric acting as electrodes that cover the middle layer, which is the functionalized cotton gauze. The size of these patches varies from $2 \times 2\ cm^2$ to $3 \times 3\ cm^2$ depending on their placement. The cotton gauze is sonicated in deionized water for 15 minutes, rinsed with isopropanol, then heated at $100^{\circ}C$ for 2 hours. The treated cotton gauze patches are rinsed once more in isopropanol and dried overnight. Finally, this layer is coated with perfluorosilane using vapor deposition to add wash-stability to the sensor.

To measure the resistance of the sensors, they are connected to a voltage divider circuit where one of the electrodes is grounded (as shown in Figure 1). For shielding purposes, we place the grounded electrode outward and closer to human skin during interaction. This is due to the fact that human body carries electrical charge, which can be coupled into the sensors and confound the interaction signal. By grounding the outer plane, we make sure this extra charge is routed to ground and that it will not show up in the output signal.

Electronics board: The sensors are internally routed to a PCB (Fig. 5c). The board uses off the shelf components listed in Fig. 5d. It receives signals from 24 sensors, filters them, and then creates two sub-channels, one amplified and one unamplified. Then the resulting 48 channels are multiplexed into 8 ADC channels of the MCU using 4 analog multiplexer ICs. Each multiplexer outputs two out of its 12 inputs according to the address bits provided by the microcontroller. Finally, the microcontroller digitizes and transmit the data to a laptop using Bluetooth Low Energy (Fig. 2).

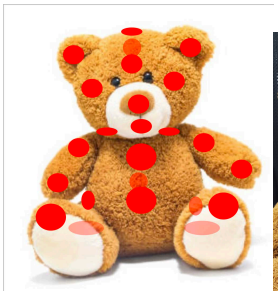
The microcontroller runs an application that provides address control signals for the multiplexers, reads the analog channels from 8 analog input pins, creates packets from the samples and an index for packet loss detection. A moving average filter over seven samples and duty cycling is used to reduce power consumption.

Implementation of classification model: For the classification task, we have both single interactions (interaction at one location) and complex interactions (two concurrent interactions). For the given time window, we use the tuple (interaction@position) that is most frequently shown as the ground truth for the single interaction; and similarly for complex interactions, we use the most frequently seen two single interactions as the ground truth.

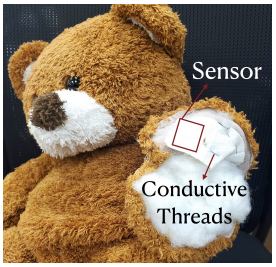
We use leave one out as cross validation for the model training. Due to the fact that many of the ground truth labels are "no interaction" in the collected data and the labels are unbalanced, we use a data sampling technique to balance the labels in the dataset for

Table 1: Structure of Feature Extractor

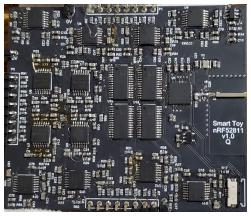
Layers	Specifications	Input	Output
Conv1(+BN+ReLU6)	kernel: 3, stride=1	(1, 48)	(12, 48)
Conv2(+BN+ReLU6)	kernel: 3, stride=1	(12, 48)	(24, 48)
Conv3(+BN+ReLU6)	kernel: 3, stride=1	(24, 48)	(48, 48)
Conv4(+BN+ReLU6)	kernel: 3, stride=1	(48, 48)	(24, 48)
Conv5(+BN+ReLU6)	kernel: 3, stride=2	(24, 48)	(24, 24)
Dense	output: (37,) or (103,)	(576,)	(37,) or (103,)



(a)



(b)



(c)

IC name	Description	Supply curent
ADG1604	Analog multiplexer	$1\mu A$
TLV8544	ULP OpAmp	$550nA$
NRF52811	MCU + BLE	$\sim 1.6\ mA$
S-1318	3.3 V regulator	$55\ nA$

(d)

Figure 5: a) The locations where sensors are placed on the toy. b) The fabric sensor placed underneath the toy felt. c) The main electronics components used in FabToy, d) approximate supply current (current for MCU is measured while ADCs and filtering were running).

the model training. We use cross entropy loss to train the FabToy model and the loss function is:

$$Loss = -\frac{1}{N}(\sum_{i=1}^{N} \mathbf{y_i} \cdot \log(\hat{\mathbf{y}}_\mathbf{i})) + ||W||^2 \quad (2)$$

where $\mathbf{y_i}$ and $\hat{\mathbf{y}}_\mathbf{i}$ are the ground truth label and prediction label, $||W||^2$ is the $l2-$regularization of the model parameters and N is the size of the dataset.

We implemented the local and remote execution pipelines of FabToy on the nRF52840 platform [36]. nRF52840 is a low-power ARM-based embedded device with Bluetooth Low Energy (BLE) protocol support, which allows us to implement both the local and remote processing pipelines. The local model requires 18.5KB of RAM and 153.7 KB of flash memory, corresponding to 7% and 15.3% of available space in the MCU, respectively.

5 EVALUATION

In order to validate the performance of FabToy, we performed a series of experiments that highlights the benefits of our hardware platform and machine learning pipeline, and provides a breakdown of the contribution of various building blocks in FabToy.

5.1 User Study

We asked 18 participants to perform several interactions with a toy as part of an IRB-approved study. The interactions include holding, patting, tickling, and swiping the toy. We chose this combination as an example of meaningful interactions that can be performed with a toy (e.g. holding a hand while patting the head). This separation of single and complex interactions gives us the opportunity to monitor the impact of complex interactions and compare the performance of FabToy with other models. We chose 37 single interactions from the pool of possible interaction/locations pairs to prove the possibility of detecting the interaction and its location. In addition, we chose 65 complex interactions that include holding one of the toy's hands and performing the other interactions with users' free hand. Including the *idle* state, we have 103 different combinations of single and complex interaction/location as labels.

The users varied in age (25 - 35) and gender (6 females). They were free to choose their own method of performing each specific interaction which includes the speed and the intensity of the action. We used adults rather than children since it is difficult to collect good quality longitudinal datasets with small children, particularly when it involves data collection involving several repetitions of each interaction. In addition, it has been significantly more challenging to involve small children in studies due to IRB restrictions. We feel that our current dataset is sufficient to demonstrate feasibility of FabToy and our algorithms, and these can be further tuned in future work for more real-world use with children.

The feel and look of the toy conveys no clue regarding the whereabouts of sensors underneath the felt and the users were not informed about it either so as not to alter users' behaviours. We placed a video camera zoomed in on the toy to be used as ground truth for labeling the data. Each user went through slightly more than 16 minutes of study where they were asked to perform a series of interactions covering all the 103 labels, each during a ten second window with 3-4 seconds rest, counted toward *idle* case. Overall, we gathered around 5 hours of data from our participants while they were interacting with FabToy.

We start by presenting the overall classification results and then provide benchmarks about the individual processing blocks.

5.2 Overall Classification Performance

We start by looking at the overall performance of FabToy. Since cross-talk is present between adjacent sensors, we look at the classification performance for three spatial granularities:

- *Fine-grained*, where we want to precisely determine which of the twenty four sensor locations is being interacted with.
- *Medium-grained*, where we merged some adjacent sensors into one location including: hand and arm on each side, foot and thigh on each side, nose and mouth, forehead and top of the head, and chest and stomach. Other sensors are considered individually. This results in 24 sensor locations being merged into eight regions of interaction.
- *Coarse-grained*, where we merged the following sensors and counted them as one: both arms i.e. hand and arm on each side; both feet i.e. foot and thigh on each side, head including nose, mouth, cheeks, ears and forehead and top of the head, body including chest, stomach, waist and back. Thus, we transform 24 sensor locations into 4 coarser regions.

Figure 6a shows the results. We see that for precise fine-grained classification, we achieve 86% accuracy for single interactions and 83% for complex interactions.

While classifying single interactions, the mis-classifications can be broken down as follows:

- 10.3% of mis-classifications are instances where an interaction is missed or an idle state is detected as an interaction,
- 33.1% of mis-classifications are instances where the location is correctly determined, but the interaction is mis-classified,
- 28.7% of mis-classifications are instances where the interaction is correctly determined, but the location is mis-classified, and
- 27.9% of mis-classifications are instances where the occurrence of an interaction is detected but the detected location and the interactions are false.

As we progressively coarsen the prediction granularity, classification performance increases from 86% to 94% for single interactions and from 83% to 93% for complex interactions. Thus, we see that FabToy can be very effective at both fine-grained and coarse-grained classification with increasing performance as the spatial fidelity reduces.

5.3 Comparison against alternate models

We now demonstrate the superiority of the FabToy model to some other well-established machine learning models. We compare against models that are relatively lightweight and can be executed on a microcontroller-class platform. The Multi-Layer Perception (MLP) model [29] uses features extracted from lightweight convolutional layers which are fed into multiple fully connected layers in the MLP model to compute the predictions. For the other models (Random Forest, xgBoost, and Nearest Neighbors), we pre-process the raw sensor data to help them better capture the time-series contextual information. We then use histogram density features to map the time-series data from each sensor into a 10-bin histogram. The histogram features ($10 \times 48 = 480$) are fed into the the three machine learning pipelines to train the models. The Random Forest model [26] constructs a collections of decision trees and performs an ensemble classification; the xgBoost model [6] boosts decision trees via applying gradients to correct the previous mistakes and minimize the losses; the k-Nearest Neighbors model [5] deterministically finds k instances in the dataset which are most close to the input data and use the most commonly seen label of the k instance as the label of the input.

Figure 6b compares FabToy versus other models for fine-grained classification. As can be seen, the FabToy model outperforms all others both for single and complex interactions. The FabToy model can achieve more than 5% higher accuracy compared with the MLP model and more than 10% higher accuracy compared with the other three machine learning methods.

Importance of Amplified and Non-amplified Channels: Second, we look at the advantage of using both amplified and unamplified streams in terms of overall performance. To evaluate, we separately choose amplified and non-amplified signals to train our model and check the performance against the combined version that uses both data streams. For the amplified-only and non-amplified only cases, we modified the FabToy model so that it could accept input signals that are half the size.

The result is shown in Figure 7. We see that the combination of amplified and non-amplified signals provide more information for the machine learning model compared with either of them separately. For single interactions, using both streams gives us 3 - 4% improvement in accuracy over using one of the streams. For complex interactions, we get 1 - 5% improvement over just using one of the streams. This plot shows the importance of generating amplified versions of the 24 base channels.

Benefit of early exit: While the Feature Extractor Layers are common to the local and remote models, the use of early exit blocks is specific to the local model. This allows us to bypass a portion of neural network by leveraging intermediate exit points.

The effect of early exit is studied on model accuracy (Figure 8a), model power consumption (Figure 8b), and latency on the nRF52840 (Figure 8c). These plots are calculated based on nRF52840 datasheet [23] using active CPU power and runtime for each early exit route.

We see that early exit at layer 3 reduces accuracy by about 4% but has around 8× better computation energy efficiency and latency compared to executing the full model. Early exits at intermediate layers between these ends provide progressively better accuracy but offer less gains in performance. Thus, early exit has significant advantages in terms of overall performance.

Benefit of adaptive aggregation: Specific to the remote model, we employ data dimensionality reduction by adaptive aggregation. We now look at the performance of this module. We produced this benchmark by using Nordic Online Power Profiler for BLE [2]. These numbers roughly agree with the empirical power breakdown figure as well, shown in Figure 11.

Comparison of local and remote processing: Having analyzed the local and remote versions of FabToy, we can compare the performance of these methods. Figure 9c shows the results. We see that the local model is generally more efficient than the remote model. This is unsurprising since the radio consumes more

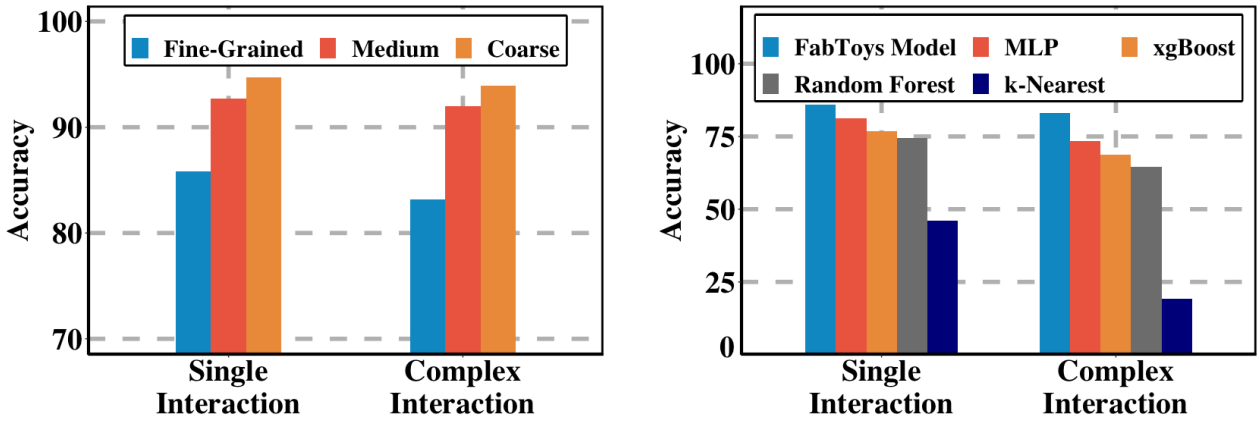


Figure 6: a) Performance of FabToy for fine-grained and coarse-grained localization. Accuracy increases as we reduce the spatial resolution of the sensors, and coarse-grained classification accuracy is around 94%. b) Comparison between FabToy model and other ML models. FabToy achieves higher performance for both simple and complex interactions.

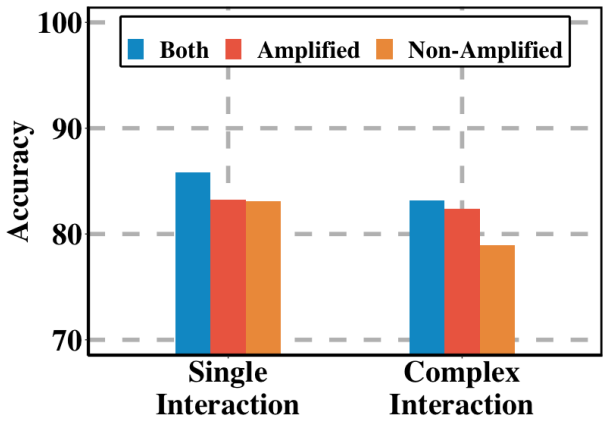


Figure 7: Breakdown of the contributions of amplified and non-amplified channels to the overall accuracy.

power than the MCU. However, the gap narrows in the regime where higher accuracies are desirable since the remote version is able to take advantage of more complex models being used on the remote device. We note that the choice between local and remote processing may be to enable specific applications rather than power consumption. From this perspective, the main takeaway is that both methods are viable at low power.

Figure 9a and 9b illustrate the trade-off between number of streams transmitted and the model's accuracy and power consumption, respectively. We see that the accuracy rapidly increases until about 10 streams, and then plateaus. This allows us to reduce transmission power consumption by about 2× compared to a system that transmits all the channels with no compression.

5.4 Execution latency of local model

We now provide a breakdown of execution latency of our model on both the nRF52811 that we use as well as other popular and emerging low-power embedded devices: GAP8 [34], Raspberry Pi 4B[27] and Jetson TX2[24]. The compute ability of embedded devices can vary widely depending on their power needs. For example, the GAP8 can execute the deep learning models with core frequency of 50MHz and power consumption of 25mW; the power consumption of a Raspberry Pi 4B is around 1.5W; and the power consumption of a Jetson TX2 is around 7.5W. We uses the system clock in Linux and hardware cycle counters in GAP8 to estimate the execution latency for each layer of the FabToy model. The model is executed multiple times and the average execution latency per layer is measured.

Figure 10 shows the latency of different layers in FabToy. As expected, the nRF52811 platform takes longer than other platforms to perform the operations. The overall average latency for local processing is 260 ms. This gives the processor enough time to sleep between each two calculations, which take place every one second.

5.5 Power benchmarks

Figure 11 shows the processing power consumption breakdown for FabToy across the different blocks. Since power consumed for the remote model depends on the number of channels transmitted and power for the local model depends on the early exit point, we provide numbers for three different channel aggregation values and three different early exit points.

Overall, FabToy consumes between 2.9mW to 4mW depending on the number of channels transmitted or the early exit point. This amount of power consumption corresponds to more than a month of operation on a small 950mAh rechargeable battery(3 $cm \times 5$ cm) before it needs recharging.

We see that a significant fraction of the power is consumed by the sampling block. This is the case since we are sampling each

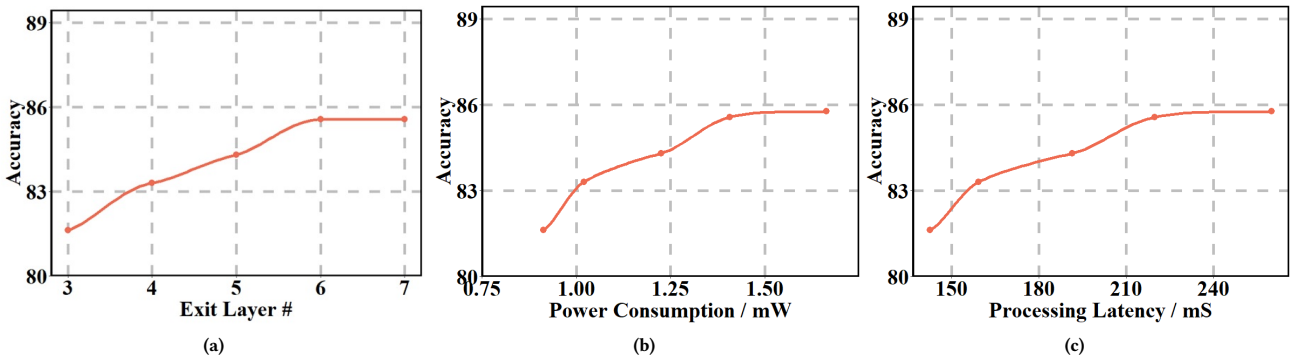


Figure 8: a) The effect of adjusting early exit layer number on system's accuracy, processing power consumption, and processing latency shown in a), b), and c), respectively.

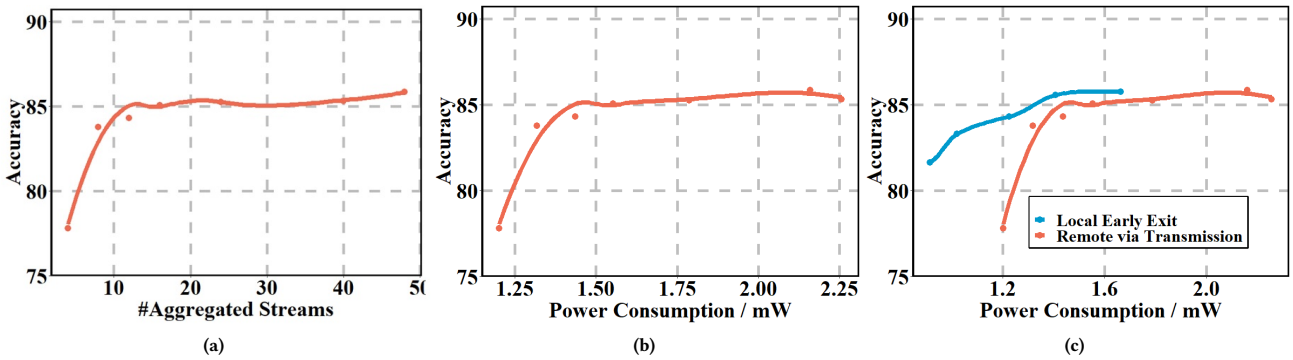


Figure 9: (a) and (b) show the effect of channel aggregation on system's accuracy and power consumption. (c) compares the power consumed by FabToy in local versus remote processing modes for the nRF52811

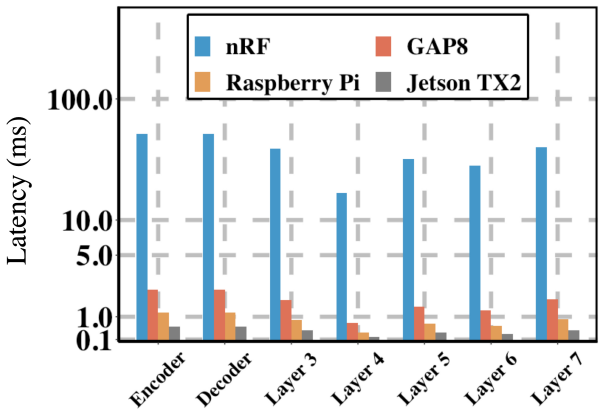


Figure 10: Compute latency across different platforms.

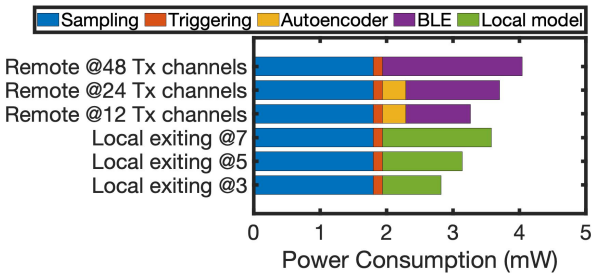


Figure 11: The breakdown of processing power consumption in various scenarios. note that layer numbers in local mode start from 3 to match the order in remote model.

channel at 160Hz (to remove powerline noise), hence the system is sampling at 7.68kHz. We have not focused on optimizing this subsystem in this paper but expect that this can be optimized with better hardware design. The triggering block can also improve efficiency if done in analog before the MCU is turned on.

Finally, we note that since the board and battery can be placed inside the toy, they are physically isolated using waterproof packaging and as a result, the battery capacity can be increased to allow for months of operation per full charge.

6 RELATED WORK

In this section, we describe relevant state of the art in using toys as means to extract physical and physiological signals from children, and the some works regarding the usage of fabric-based sensors in measuring humans health and activity parameters.

Augmenting toys with sensing capabilities There has not been much work on exploring the use of array of sensors in detecting user interactions with toys. Vega-barbas et al. implemented a prototype toy that can measure grip and stretch in a toy using fabric sensors [39]. Yonezawa et al. [45] incorporate textile pressure sensors to detect intensity and duration of pressure on toy's surface and localize it as head, body, or back. Using the information they gather from the pressure sensors, they estimate user's emotional state. There have also been some works on enabling toys to gather biomedical information on the toddler playing with the toy [41]. There are several products available on the market that use microphones to listen and provide feedback according to child's vocal inputs such as CogniToys Dino [1]. Dolphin Sam [7] is another example of a design that embeds several sensing techniques, such as touch sensors and RFID readers, and provides feedback in the form of light and sound.

Using array of fabric sensors to detect physical and physiological signals Recent advances in smart fabrics have led to sensing, actuation, and energy harvesting fabric elements. Liu et al. used multiple fabric pressure sensors to measure finger joint dynamics, physiological signals, and created an input matrix using an array of those sensors [21]. Kiaghadi et al. designed and implemented a loosely-worn sleepwear that can track sleeping posture, heart rate, and respiration rate [18]. This work relies on four pressure sensors and a triboelectric sensor to capture dynamic and static pressure signals.

There have been several works that use the textile sensors to capture user inputs [3, 25]. These methods generally use capacitive or resistive pressure sensors by implementing multiple sensing cells or using weaving structures to create arrays of sensing elements.

Fabric based sensors have also been used to detect various human-related parameters — some examples include using strain sensors for human motion detection [42], inserting triboelectric sensors into a loosely-worn shirt to measure joint dynamics and perspiration [17], and using strain sensors in detecting respiration rate [4].

However, none of these efforts enables what we have proposed i.e. fine-grained interaction detection and localization on plush toys. Our application and the techniques we use to deal with large-scale sensor arrays are not addressed in prior work.

Early exit and Autoencoder The machine learning components we use, Early exit and Autoencoder, are well known techniques in edge AI. BranchyNet [33] introduced early exit as a mean to reduce power consumption and latency [31, 32], and this method has since become quite popular. Reducing data dimension using Autoencoders have also been studied thoroughly in literature (e.g. [12, 40]). Our contribution is that we leverage these techniques in the context of a unique smart toy system rather than the machine learning components by themselves.

7 DISCUSSION

In our design of the FabToy, we have encountered several roadblocks and opportunities for future work. We briefly discuss these issues.

Application studies involving toddlers and children: Our dataset is collected from adults to enable repeatable and dense data collection since we have a large number of {interaction,location} pairs that we need to be able to distinguish with our machine learning model. Due to the vaccination status of very small children who would normally play with plush toys, we have not conducted a user study with this age group. Once restrictions ease, we plan to study how small children interact with the FabToy in naturalistic conditions and validate the algorithms that we have trained using our current dataset. We are actively pursuing collaboration with psychologist experts to investigate whether interaction measures automatically extracted from the FabToy can be used as a screening procedure for emotion regulation-related psychopathological disorders.

Multi-Toy Interaction: One of the exciting new opportunities presented by FabToy is detecting a much richer set of interactions with soft toys. So far, we have focused on a single FabToy but our work can potentially be extended to interactions with more than one toy. This can allow us to explore more complex storytelling applications that involve interaction with different toys that play different roles in the story [9]. For example, a group of children may perform a detailed story using a set of FabToy toys as if they were real. This story may involve handling, exploring, interacting and focusing on a toy's features in an interactive manner [15]. While such multi-toy interaction with smart toys can be enabled by today's technology, the key advantage in our approach is the naturalistic feel for the smart toy making it more engaging for children.

Platform for introducing computational thinking: Programmable robots are extensively used for introducing computational thinking in early childhood education, where children learn about computing via physical interactions with robots (e.g. KIBO [30], Dash Robot [35], Bee-Bot [38]). Unlike rigid robotic toys, FabToy is a plush toy that can engage with children more naturally, thereby providing an alternate platform to engage children physically while teaching computational techniques. For example, children can map voice responses from the plush toy to interactions performed in the physical world such as patting or tickling. This can also be connected to a platform such as Scratch Studio [28] that enables creation of interactive storytelling applications for a more compelling learning experience.

Sensing and Actuation in FabToy: While FabToy is focused on dense sampling of a soft toy, there are many new opportunities if we can pair it with other modalities like audio to expand the vocabulary of interaction. This have been significant advances in natural language understanding and dialog, and we can potentially pair richer tactile sensing of interactions with the FabToy sensors with more sophisticated audio-based dialog methods to enhance how children interact with smart toys.

Dynamics in natural environments: We have designed, implemented, and analyzed an interaction-aware toy and validated

the performance in semi-stationary scenarios. However, there may be a broader range of interactions in the natural environment, for example, a child may interact with a toy during walking. This can create new signal dynamics as walking causes vibrations all over the toy, which will be seen by all the channels. Such global actions can complicate the interaction detection process as the signal caused by the walking may drown some desired signals such as weaker tickling. To deal with this issue, we will need to add active motion artifact canceling methods where an algorithm detects walking by distinguishing the vibrations in most of the sensors with similar rhythm, and recreates the motion signal and subtracts it from the original. This is one of the solutions we plan to explore as a future work.

8 CONCLUSION

In this paper, we presented FabToy, an end-to-end platform for detecting and localizing users' interactions with soft toys in a fine-grained real-time manner. Our design addresses a number of challenges including ensuring unobtrusiveness and natural look and feel while still achieving high signal quality, high spatio-temporal fidelity, as well as low power operation. To enable this, we have optimizations across the hardware-software stack including a highly optimized array of fabric sensors, low-power signal conditioning and acquisition, as well as low-power embedded machine learning and data compression. Our evaluation shows that the device can enable accurate detection across a range of simple and complex interactions across the entire surface of the toy. Overall, FabToy offers a very promising path forward for smart plush toys and has significant potential to enable a new class of interactive toys for kids.

ACKNOWLEDGEMENTS

This work was funded by the National Science Foundation under agreement CSR Medium 1763524. We also acknowledge support from the National Science Foundation under agreements 1815347 and 1839999.

REFERENCES

[1] Cognitoys dino - kids cognitive electronic learning toy. https://www.amazon.com/CogniToysPowered-Cognitive-Electronic-Learning/dp/B014IO4HYS.
[2] Online power profiler for bluetooth LE.
[3] R. Aigner, A. Pointner, T. Preindl, P. Parzer, and M. Haller. Embroidered resistive pressure sensors: A novel approach for textile interfaces. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–13, New York, NY, USA, 2020. Association for Computing Machinery.
[4] S. Ali, S. Khan, A. Khan, and A. Bermak. Developing conductive fabric threads for human respiratory rate monitoring. *IEEE Sensors Journal*, 21(4):4350–4356, 2021.
[5] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.
[6] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.
[7] S. Colombo, F. Garzotto, M. Gelsomini, M. Melli, and F. Clasadonte. Dolphin sam: A smart pet for children with intellectual disability. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '16, page 352–353, New York, NY, USA, 2016. Association for Computing Machinery.
[8] C. Day and A. Midbjer. *Environment and Children: Passive Lessons from the Everyday Environment*. Architectural, 2007.
[9] W. Fontijn and P. Mendels. Storytoy the interactive storytelling toy. In *Second International Workshop on Gaming Applications in Pervasive Computing Environments at Pervasive*, 2005.
[10] A. Golgouneh, M. T. I. Molla, and L. E. Dunne. A comparative feasibility analysis for sensing swelling with textile-based soft strain sensors. In *Proceedings of the 23rd International Symposium on Wearable Computers*, ISWC '19, page 60–65, New York, NY, USA, 2019. Association for Computing Machinery.
[11] J. Gong, Y. Wu, L. Yan, T. Seyed, and X.-D. Yang. Tessutivo: Contextual interactions on interactive fabrics with inductive sensing. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, UIST '19, page 29–41, New York, NY, USA, 2019. Association for Computing Machinery.
[12] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
[13] S. Z. Homayounfar and T. L. Andrew. Wearable sensors for monitoring human motion: A review on mechanisms, materials, and challenges. *SLAS TECHNOLOGY: Translating Life Sciences Innovation*, 25(1):9–24, 2020. PMID: 31829083.
[14] S. Z. Homayounfar, A. Kiaghadi, D. Ganesan, and T. L. Andrew. PressION: An all-fabric piezoionic pressure sensor for extracting physiological metrics in both static and dynamic contexts. *Journal of the Electrochemical Society*, 168(1):017515, jan 2021.
[15] P. Ihamäki and K. Heljakka. The internet of toys, connectedness and character-based play in early education. In *Proceedings of the future technologies conference*, pages 1079–1096. Springer, 2018.
[16] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.
[17] A. Kiaghadi, M. Baima, J. Gummeson, T. Andrew, and D. Ganesan. Fabric as a sensor: Towards unobtrusive sensing of human behavior with triboelectric textiles. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, SenSys '18, page 199–210, New York, NY, USA, 2018. Association for Computing Machinery.
[18] A. Kiaghadi, S. Z. Homayounfar, J. Gummeson, T. Andrew, and D. Ganesan. Phyjama: Physiological sensing via fiber-enhanced pyjamas. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 3(3), Sept. 2019.
[19] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
[20] E. Li, L. Zeng, Z. Zhou, and X. Chen. Edge ai: On-demand accelerating deep neural network inference via edge computing. *IEEE Transactions on Wireless Communications*, 19(1):447–457, 2019.
[21] M. Liu, X. Pu, C. Jiang, T. Liu, X. Huang, L. Chen, C. du, J. Sun, W. Hu, and Z. Wang. Large-area all-textile pressure sensors for monitoring human motion and physiological signals. *Advanced Materials*, 29, 09 2017.
[22] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.
[23] Nordic Semiconductor. *nRF52840*, 2019. V1.1.
[24] Nvidia. Nvidia jetson tx2. *online].(https://developer.nvidia.com/embedded/jetson-tx2*, 2019.
[25] K. Ono, S. Iwamura, A. Ogie, T. Baba, and P. Haimes. Textile++: Low cost textile interface using the principle of resistive touch sensing. In *ACM SIGGRAPH 2017 Studio*, SIGGRAPH '17, New York, NY, USA, 2017. Association for Computing Machinery.
[26] M. Pal. Random forest classifier for remote sensing classification. *International journal of remote sensing*, 26(1):217–222, 2005.
[27] R. Pi. Raspberry pi 4 model b. *online].(https://www. raspberrypi. org*, 2015.
[28] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, et al. Scratch: programming for all. *Communications of the ACM*, 52(11):60–67, 2009.
[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
[30] A. Sullivan, M. Elkin, and M. U. Bers. Kibo robot demo: engaging young children in programming and engineering. In *Proceedings of the 14th international conference on interaction design and children*, pages 418–421, 2015.
[31] S. Teerapittayanon, B. McDanel, and H. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469, 2016.
[32] S. Teerapittayanon, B. McDanel, and H. Kung. Distributed deep neural networks over the cloud, the edge and end devices. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 328–339, 2017.
[33] S. Teerapittayanon, B. McDanel, and H.-T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469. IEEE, 2016.
[34] https://greenwaves-technologies.com/gap8-product/. GAP8: Ultra-low power, always-on processor for embedded artificial intelligence.
[35] https://www.makewonder.com. Wonder Workshop.
[36] https://www.nordicsemi.com/Products/nRF52840. Multiprotocol Bluetooth 5.2 SoC supporting Bluetooth Low Energy, Bluetooth mesh, NFC, Thread and Zigbee.
[37] https://www.tensorflow.org/lite/. TensorFlow Lite: Deploy machine learning models on mobile and IoT devices.
[38] http://www.terrapinlogo.com. Terrapin Inc.
[39] M. Vega-Barbas, I. Pau, J. Ferreira, E. Lebis, and F. Seoane. Utilizing smart textiles-enabled sensorized toy and playful interactions for assessment of psychomotor

development on children. *Journal of Sensors*, 2015:898047, May 2015.

[40] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 1096–1103, New York, NY, USA, 2008. Association for Computing Machinery.

[41] E. Vonach, M. Ternek, G. Gerstweiler, and H. Kaufmann. Design of a health monitoring toy for children. In *Proceedings of the The 15th International Conference on Interaction Design and Children*, IDC '16, page 58–67, New York, NY, USA, 2016. Association for Computing Machinery.

[42] J. Wang, C. Lu, and K. Zhang. Textile-based strain sensor for human motion detection. *ENERGY & ENVIRONMENTAL MATERIALS*, 3(1):80–100, 2020.

[43] T. Wu, S. Fukuhara, N. Gillian, K. Sundara-Rajan, and I. Poupyrev. Zebrasense: A double-sided textile touch sensor for smart clothing. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, UIST '20, page 662–674, New York, NY, USA, 2020. Association for Computing Machinery.

[44] T.-Y. Wu, L. Tan, Y. Zhang, T. Seyed, and X.-D. Yang. Capacitivo: Contact-based object recognition on interactive fabrics using capacitive sensing. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, UIST '20, page 649–661, New York, NY, USA, 2020. Association for Computing Machinery.

[45] T. Yonezawa and H. Yamazoe. Analyses of textile pressure-map sensor data of a stuffed toy for understanding human emotional physical contact. In *Proceedings of the 6th International Conference on Human-Agent Interaction*, HAI '18, page 191–198, New York, NY, USA, 2018. Association for Computing Machinery.