Active Learning On Embedded Devices

Frans de Boer
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
f.deboer-1@student.utwente.nl

ABSTRACT

Machine Learning algorithms require a lot of resources to predict, and even more resources to be trained. This makes it very difficult to train machine learning algorithms on embedded devices, which typically have a small battery and low power processors. This low power processor allows the device to last a long time on a small battery, but it also means tasks that require a lot of resources are difficult to implement. This research aims to use active learning to reduce the number of training iterations required to achieve a high accuracy on a machine learning problem, and thus make it more feasible to train machine learning algorithms on low powered devices. In the end we show that while our active learning algorithms had some problems, energy usage was still reduced and it could be a promising way of reducing the energy usage of machine learning algorithms.

Keywords

Machine Learning, Active Machine Learning, Embedded Machine Learning

1. INTRODUCTION

Machine learning is quite an actively researched subject in computer science and has been applied in many fields, ranging from crop production [17] to cancer classification [2]. These learning algorithms can pick up on patterns from a dataset that it is given, and can then be used to recognize these patterns in new never before seen data.

Supervised learning is a type of machine learning where the dataset comes in input-output pairs. Each sample in the dataset has a list of features and a corresponding label. The learning algorithms can then learn to associate each label with a pattern in the input list, and predict labels for new data. Collecting the input data can be done by machines, but the labelling process is usually done by humans and can take a lot of effort, thus researchers have looked at active learning as a way to reduce the amount of labelling required.

Active learning is a training method for machine learning algorithms where the learner can request, in active learning this is called querying, samples from the dataset to be labelled by a human, also called an oracle. We start with a small labelled subset of the dataset called the seed and let the learner train on this. After this, the learner chooses one or more samples from the dataset that it finds confusing, in other words, it queries labels for the samples that it thinks will give it the most information.

Active learning has been used in applications like image classification [13, 18], classifying cancer pathology reports (natural language processing) [4] and animal activity recognition [16]. Using active learning the number of labelled training examples required to achieve a high accuracy can be reduced, thus reducing the workload for human labellers.

In embedded machine learning a machine learning algorithm is implemented on an embedded device. Example applications are activity recognition [16, 6], and Internet-of-Things applications [11].

35th Twente Student Conference on IT July 2nd, 2021, Enschede, The Netherlands.

2. RELATED WORK

Studies have been done on the current state of machine learning on embedded devices, and ways to make these machine learning algorithms run faster or more efficient [3, 1]. Neither of these studies however assume the training phase is done on the embedded device and don't mention active learning as a way to make the algorithm learn more efficiently.

In [6] methods of extending the battery lifetime of a wearable sensor are explored. In this research almost all extra battery lifetime was obtained by changing the sampling rate and the data resolution of the sensor. A small extra increase was obtained by putting the machine learning algorithm (a Support-Vector Machine) on the embedded device instead of sending the data to a server to be processed.

In [16] active learning is used for human activity recognition on a smartwatch. Reducing energy usage using active learning is mentioned in the paper but is not further researched.

Most found active learning research either explores active learning methods or the ways active learning can be used to decrease the amount of labelled data needed to achieve a high accuracy machine learning schema. To our knowledge little to no research has been conducted about using active learning specifically as a way to reduce power usage of embedded devices.

Many different methods of active learning exist, and [14, 15] both investigate the current state of active learning. The research [15] proposes three different types of active learning: *membership query synthesis*, *stream-based selective sampling*, and *pool-based active learning*. With membership query synthesis the learner can create its own in-

stances, and query the oracle to label these. Stream-based selective sampling and pool-based active learning are quite similar, but with some notable differences. With stream-based selective learning, the learner evaluates the unlabelled instances one by one using an informativeness measure, and if an instance is informative enough (for example the informativeness measure is above a certain threshold), it queries the oracle to label it. With pool-based active learning, the instances are once again evaluated using an informativeness measure, but now all instances are evaluated, and the oracle queries the most informative instance(s). In [14] two categories of informativeness measures are proposed, *agnostic* and *non-agnostic*. Agnostic strategies ignore any predictions made by the learner on the dataset and instead decide the informativeness of an instance only based on the structure of the sets of labelled and unlabelled data. Non-agnostic strategies use the predictions of the learner to decide the informativeness of an instance.

Many different active learning algorithms exist to query the most informative instances from a dataset. Several of these algorithms are discussed in [14], and in the following paragraphs we will highlight a few of them.

2.1 Random Sampling

One of the simplest strategies of agnostic active learning is random sampling. With random sampling, the learner randomly picks an instance from the unlabelled set and queries the oracle to label it. This sampling method is often used to create the initial seed for the active learner to train on.

2.2 Utility Sampling

Utility sampling is a non-agnostic active learning method where the learner classifies unlabelled instances and queries the instance it is least certain about. Several utility functions exist to quantify this notion of certainty, and a high uncertainty could imply a lot of information can be learnt from the instance. We describe three of these utility functions below.

2.2.1 Uncertainty Sampling

In uncertainty sampling the learner will query the instance where it is the least certain about the label. The learner will look at the probability it predicted of the label being correct for each instance, and it will pick the instance where this probability is the lowest.

This method works well if there are only two possible classes (so it would work well for binary classification), but a lot of information is lost in cases where there are more than two classes. For this other utility measures could work better.

2.2.2 Margin Sampling

With margin sampling the learner will once again look at the probabilities it predicted for the labels of an instance, but this time instead of looking only at the highest probability (the probability of the label it predicted), it looks at the highest two. The learner will then query the instance where the difference between these two probabilities is the lowest. This difference being low, in other words, the two labels had a very similar probability according to the learner, means the learner was not sure which label to pick.

2.2.3 Entropy Sampling

Entropy sampling uses Shannon's entropy function to calculate the entropy of each prediction and then chooses the prediction with the most entropy. In a perfect prediction we expect this entropy to be 0, in this case the prediction for the correct label will be 1, while the prediction for all other labels will be 0. When the entropy gets further from 0, and closer to 1, there is more uncertainty in the prediction of the algorithm.

3. PROBLEM STATEMENT

A machine learning algorithm requires a lot of resources to be trained. This makes it very difficult to put learning algorithms on embedded devices, which typically have a small battery and a slow processor. By applying active learning to a learning algorithm we aim to decrease the amount of energy required to achieve the same accuracy as when a passive learning strategy is used, and thus make it more feasible to train the learning algorithms on embedded devices.

One note to make is that the goal of this research is not to create a classifier with a high accuracy or to increase the accuracy of the classifier using active learning. Instead, the goal is to use active learning to achieve similar accuracies as when passive learning strategies are used, while using less energy.

3.1 Research Question

To research a solution to our problem we have formulated the following research question:

- Can active learning be used to reduce the energy usage of training embedded machine learning algorithms?

Which we have split up into the following subquestions:

1. How much energy does training a passive machine learning algorithm use?
2. Can an active learner be built that gets a similar accuracy as the passive learner?
3. How much energy does training the active learner use relative to the passive learner?

4. METHODOLOGY

In order to find answers to our research questions we first needed to answer questions about what algorithms we are going to use and what hardware we will run these algorithms on, and how we will do the energy measurements. The following sections will explain this.

4.1 Hardware

To emulate the workings of an embedded device we needed something that shared some of the constraints embedded tasks have to deal with, such as a low power processor and very little RAM, while also giving us the ability to easily run a variety of tests on the device. To meet both of these requirements we decided to use the Raspberry Pi family of systems, more specifically the Raspberry Pi 3B+ was used for initial testing and setup, and the Raspberry Pi 0 W was used for the final data collection. The Raspberry Pi 3B+ was chosen to start with because of its higher power, and better software supported (most notably TensorFlow was a lot easier to install in the 3B+). After it was confirmed everything ran on the 3B+ we switched to the Pi Zero W, which due to being more resource-constrained more closely resembles embedded applications. To get TensorFlow on the Raspberry Pi 3B+ we used [9], and for the Raspberry Pi 0 W we used [12]. From here on all mention of "the Pi" will refer to the Raspberry Pi Zero W.

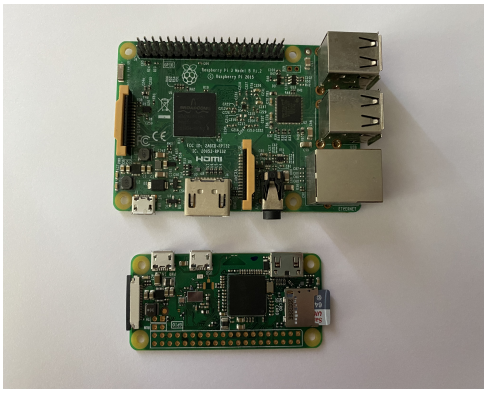

Figure 1. RPI 3B+ (top) and RPI 0 W (bottom)

4.2 Data & Data Processing

One of the applications of machine learning on embedded devices is (animal) activity recognition [16, 10]. For this reason, we used the labelled dataset from [10] to run our experiments with. The dataset consists of sensor data from sensors around the necks of 4 sheep and 2 goats. Data was collected from a 3D accelerometer, a 3D gyroscope, a temperature sensor, and a pressure sensor while the animals were doing activities such as eating, walking, and grazing.

Data processing was done following the guidelines of the original research. In the paper only the accelerometer data was used for training the classifiers, so we will do the same. First, the orientation-independent magnitude of the 3D vector of the accelerometer is calculated for each point of data using the following formula:

$$M(t) = \sqrt{s_x(t)^2 + s_y(t)^2 + s_z(t)^2} \tag{1}$$

The research mentions using a window function on the data, and then calculates the *min*, *standard deviation*, and 25^{th} percentile for each window to use as features. The research however does not mention the window size, overlap, or windowing function that was used. To get a good estimate of what windowing parameters to use we looked at [5]. This research analyses the best window size for activity recognition on sheep and concludes that good window parameters are a 7-second window with 50% overlap. This means that at a sensor polling frequency of 200Hz the window size used to process the dataset is 1400 samples. Neither research mentions the windowing function that was used, so to reduce the number of parameters a rectangular window (also called a boxcar) function was used.

After the input parameters for each sample have been calculated we need to determine the label per parameter. Because we use a window multiple different labels can be in each window. Here we followed the instructions from the original research and chose the label per sample to be the label that occurs most often in the window.

Finally, 10% of this dataset is used for testing, which is randomly selected from the dataset at the start of each test.

4.3 Machine Learning Algorithm

Together with the data [10] also describes its learning algorithms. In total 7 different classifiers were used, from these we will use the Neural Network and Deep Neural Network for our tests.

Figure 2. FNB28 USB Power Meter Interface

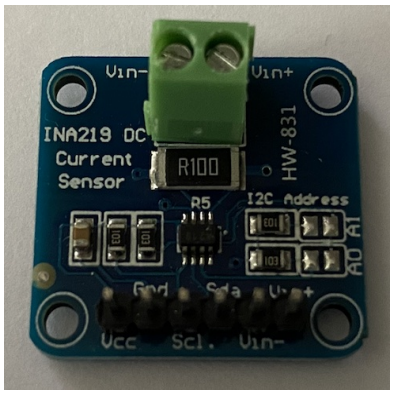

Figure 3. INA219 Current Sensor Board

The neural network described is a multilayer perceptron, a type of feedforward neural network. The network has 1 hidden layer, where the number of neurons is defined by:

$$\psi = \frac{\gamma + \rho}{2} + 1 \tag{2}$$

Where ψ is the number of neurons, γ is the number of features, and ρ is the number of classes. In our case we have 3 features and 9 classes, so we get 7 neurons in the hidden layer. We used the Adam optimizer included in TensorFlow and found we got the best results with a learning rate of 0.05. After some testing we saw that the neural network achieved its maximum testing accuracy after about 20 epochs (it was able to get about 2% higher, but this required about double the epochs), so any passive learning tests were ran with 20 epochs. Any other parameters were left to their Tensorflow defaults.

We also implemented a version of the deep neural network (DNN) described in the research. The DNN we implemented is also a multilayer perceptron, with 10 hidden layers each with 50 neurons. However after some initial tests we could not get the accuracy of this DNN above 45%, thus for any testing we only used the 1 layer neural network.

4.4 Energy Measuring

Finally, if we want to measure the difference in energy usage, we need a way to measure the energy usage of the Pi. Solutions already exist to do this, such as lab power supplies or USB power monitors with a screen. For this research the plan was initially to use a USB power monitor, so an FNB28 USB power meter was bought (see figure 2). While this device was able to measure the power usage and store this data locally, there was no way (as far as we could find) to get this data off of the device. To solve this problem a custom power meter was built using an

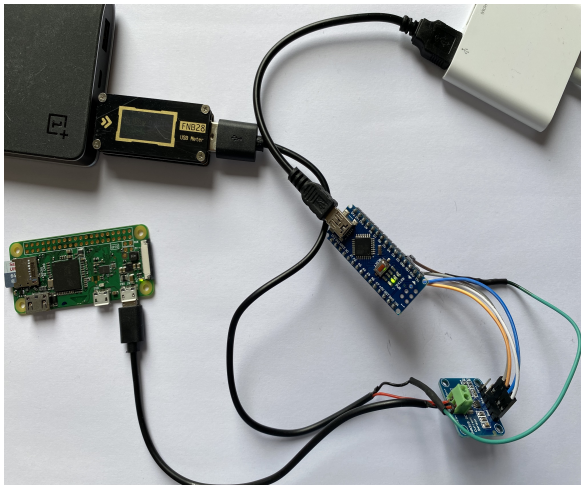

Figure 4. Energy Measurement Setup

Arduino and an INA219 current sensor board (see figure 3) following the instructions from [7, 8]. The full energy measurement setup can be seen in figure 4. The Arduino measures both the voltage and current draw of the Pi every 200 milliseconds using the INA219 sensor, and sends this data over USB to a laptop. From here the data can be stored and analyzed. From the voltage and current we can calculate the power usage using the following formula:

$$P = V * I \tag{3}$$

Up next we can use the difference in time between samples to calculate the energy usage. One problem here was that the Arduino did not consistently send the data every 200 milliseconds and would sometimes send a batch of samples at the same time. Due to this some of the times on our samples weren't correct, so to fix this we used the average time between samples instead of the measured time.

Now that we have a power measurement per sample, and can calculate the difference in time between samples we can calculate the energy usage over our average period of time. To do this we take the time difference between two samples, and multiply this by the power usage of the second sample:

$$E_i = P_i * (t_i - t_{i-1}) \tag{4}$$

Where i is the index of a sample. Summing this for every sample gives us a total energy usage over our measurement period.

The FNB28 was used to visually confirm the results obtained from the Arduino, but was removed for any later tests so its power draw would not influence the results.

To ensure we only measure the energy used for training, and not the energy used for setting up the training code (for example just importing TensorFlow on the Pi can take around 30 seconds) we start measuring the energy usage only when the training starts. One problem we encountered here is that TensorFlow does some setup before it starts training which we were not able to account for.

4.5 Active Learning

For our tests we implemented active learning, margin sampling, uncertainty sampling, and (Shannon) entropy sampling. There are a lot of parameters to consider while running these algorithms, such as the seed size, the number of iterations to run, and the number of samples to query per iteration. After some testing we opted to use the following configuration: Random sampling was used to create a seed with 100 instances, on which the network was trained for 5 epochs. After this 20 iterations of querying are done, where each iteration 10 samples are queried using the active learning algorithm and the network is trained for 3 more epochs. This made the final size of our reduced dataset 300 instances or about 1.69% of the total dataset. We got better results when training the active learners with a slightly higher learning rate, so the learning rate was set to 0.1 for all active learning tests (for the passive learning tests it was kept at 0.05). This configuration reached a similar accuracy to our passive learning solution, and early testing indicated the energy usage was at or below the energy usage of our passive learner.

Table 1. Baseline Training Average Results

% of Training Set	Energy (J)	Test Accuracy
100%	230	81.6%
10%	38	77.7%
1%	19	66.2%

Table 2. Active Learning Results

Active Learning Method	Energy (J)
Random	58
Margin	189
Uncertainty	179
Entropy	521

5. RESULTS

To get started with the results we first wanted a baseline of the energy usage of the Pi Zero W. The energy usage of the Pi Zero W was measured for 5 minutes, once while it was doing nothing, and once while it had two ssh connections. This last measurement was done because for most of the testing the Pi had multiple ssh connections, so we wanted to measure if this had a noticeable effect on energy usage. The results of this measurement can be seen in figure 5. The results of these tests were mainly used so we could visually see in the incoming data if the Pi was drawing current above the passive baseline, which gave us an indication about the current activity level of the Pi.

Up next we wanted a good baseline for the energy usage of the Pi while training. To do this we trained the neural network for 20 epochs with the full dataset. After this we randomly selected 10% of the entire dataset and trained the algorithm once again with this, and then once more with 1% of the entire dataset. All of these measurements were ran 3 times using the same seed and the final average was used for the results. Because we used a seed for these tests the final accuracies were the same in all 3 tests, so to get a better idea of what accuracies each dataset would get we ran the same tests 3 more times this time without a random seed. The average results of these tests can be seen in table 1. One interesting observation is that the energy usage does not decrease linearly with the dataset size, so training with a dataset that is a tenth the size does not consume a tenth of the energy. This is probably caused by the setup code Tensorflow executes (mentioned in 4.4), which adds some extra execution time (and thus energy usage) to any training session.

We wanted to measure the energy usage of our active

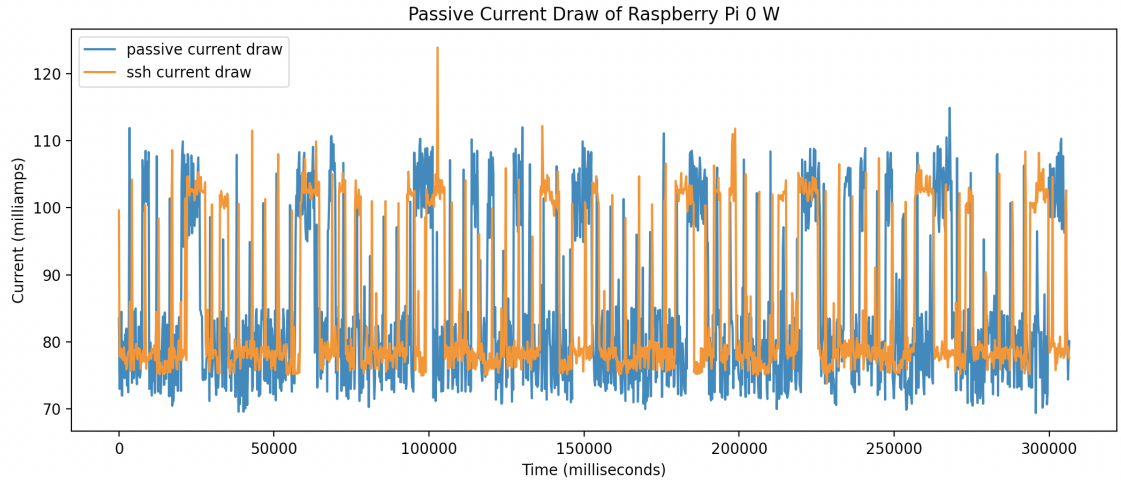

Figure 5. Passive Current Draw of Raspberry Pi 0 W

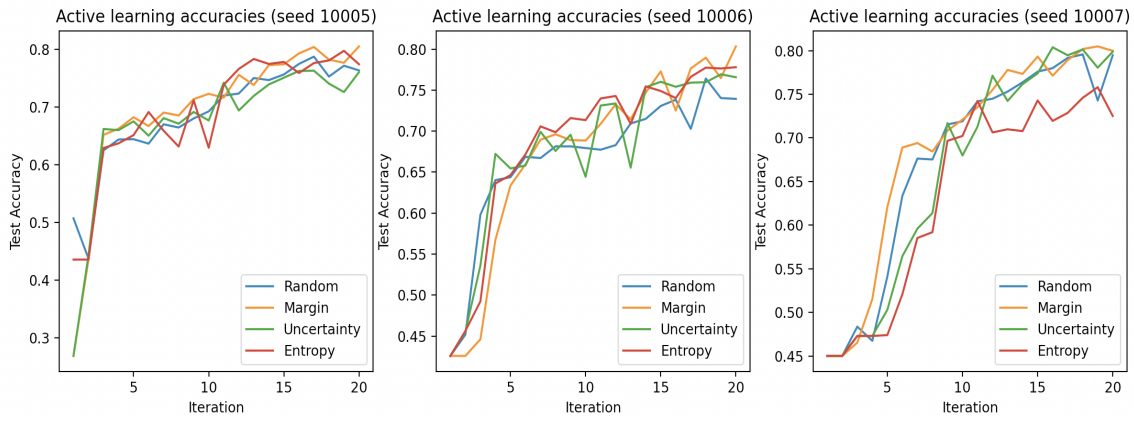

Figure 6. Active Learning Training Results

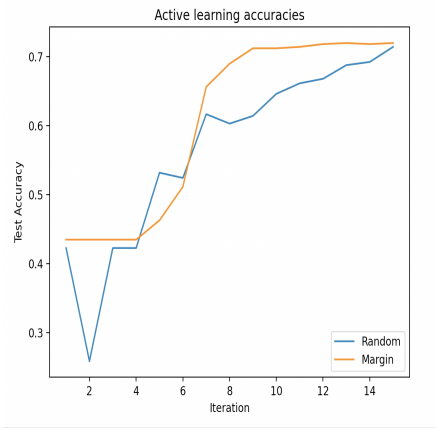

Figure 7. Early Active Learning Results

learning algorithms, and their accuracy on each iteration. Due to the fact that the training dataset is constantly changing, getting the accuracy on this training dataset is not very helpful. To get a better estimate of the performance of our network we opted to get the accuracy on the test dataset each iteration while it was training. A problem with this was that getting the test accuracy each iteration consumed more energy which would make our energy measurements come out higher, so we ran multiple versions of the test. For the first test we ran 3 iterations of each algorithm, each with the same seed, and only measured the energy usage (so no accuracy was stored here). After this we ran 1 more iteration with the same seeds, this time storing the accuracy, and then 2 more iterations both with new seeds to get more data on the accuracy of each algorithm. The energy measurements can be seen in table 2 and the accuracies per iteration can be seen in figure 6.

Comparing the results from table 1 and table 2 we can see that random, margin, and uncertainty sampling use less energy than the training on the full dataset (respectively 74.8%, 17.8%, and 22.2% less). Entropy sampling on the other hand uses more than twice the energy of training on the full dataset (126.5% more energy), due to the complexity of the entropy algorithm.

When we look at the accuracy results in figure 6 we encounter a problem. None of the more complicated active learning algorithms (margin, uncertainty, and entropy sampling) perform significantly better than random sampling. Each of the algorithms seems to perform as if it were randomly picking instances from the dataset instead of using the informativeness measures. We would expect the results to look somewhat like the results in figure 7, where margin sampling grows faster than random sampling, until they both reach the same peak. This lines up with the results obtained in [14]. The results in figure 7 were obtained early on in this research, so they don't match the results from our other active learning tests (these earlier tests were ran with fewer iterations and fewer queries, which is why they only obtained an accuracy of about 70%). This means we can't use them as a comparison to our passive learning solutions, but they do show what results we expected to get for figure 6. We were unable to recreate similar results in later stages of the research.

An interesting thing to look at with active learning is how the final dataset is composed. The results of this can be seen in figure 8. The first thing we can observe from this is that the original full dataset is quite unbalanced, there are a lot more instances of some labels than there are of others. We can see that random sampling ends up with about the same distribution as the full dataset, while margin, uncertainty, and entropy sampling deviate quite a bit from the original distribution. We have also added the distribution of the test dataset of one of our tests (with the seed set to 10005) so we can validate if the distribution is similar to that of the full dataset. The full dataset being quite unbalanced, and the fact that our active learning solutions (except for random sampling) do not create datasets with a distribution similar to the one of the full dataset could be reasons why our active learning algorithms did not perform better than random.

6. CONCLUSION

The results from table 1 show us how much energy is used for regular passive learning. From here we can see that this energy usage very much depends on the size of the dataset, a machine learning algorithm requires less energy to be trained with a smaller dataset.

In figure 6 we can see that active learning algorithms can reach about the same accuracy as passive learning, although there is somewhat more variance. One problem here is that, as discussed before, our more complicated active learning algorithms did not work better than random sampling.

When we compare just the energy usage of the passive learning and active learning solutions, active learning looks to be a way of reducing the energy usage of machine learning. Random, margin, and uncertainty sampling all had a reduction in energy usage. The problem however is that none of the more complicated active learning algorithms (margin, uncertainty, and entropy sampling) had a noticeable better accuracy than random sampling, and just using a randomly selected smaller dataset used less energy than even random sampling. Of course with this smaller dataset the accuracy was lower, but this could be increased by increasing the dataset size. So while active learning did bring a decrease in energy usage, from the results we cannot conclude that it is a good way of reducing energy usage. What we do think is that with an active learning algorithm that performs better than our random sampling (such as the results we got in figure 7), and does not use more energy than our margin sampling, the energy usage of machine learning algorithms could be reduced with minimal loss of accuracy.

7. FUTURE WORK

This research tries to lay the building blocks for future active learning research. Active learning is quite a complicated topic, and a lot remains to be researched. One of these things is to research the improvements and implementations of more complicated active learning algorithms. This research was not able to get a significant accuracy improvement using active learning algorithms such as margin and uncertainty sampling. If future research can implement this properly it could be used to reach a higher accuracy, and perhaps reach it faster to save even more energy. Other facets of active learning and embedded machine learning that this research did not look at could be researched in the future.

7.1 Hardware Acceleration

Hardware acceleration could not only allow the machine learning algorithms to run faster, but also more efficiently, and using this the energy usage of the algorithms could be reduced even further. This could be especially helpful

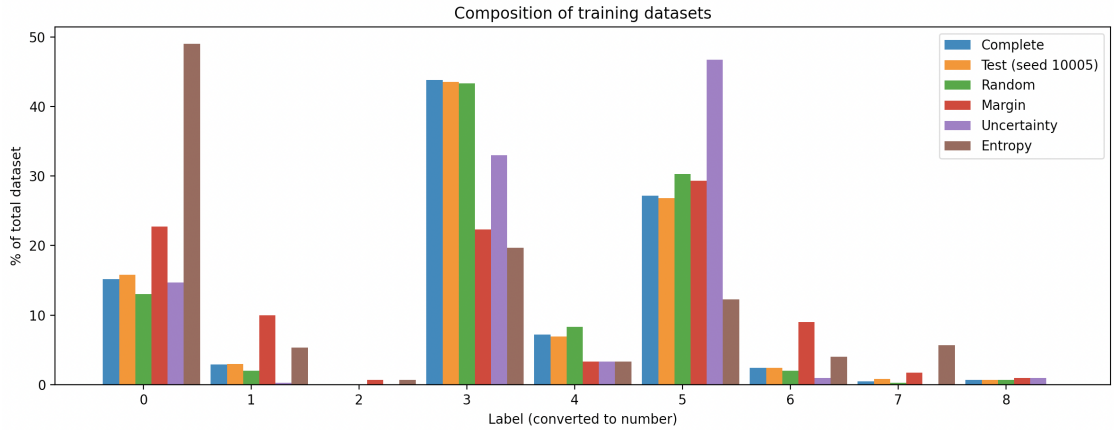

Figure 8. Compositions of Final Training Datasets

for non-agnostic active learning algorithms, which use the learner to predict on the unlabelled dataset. If this predicting can be made more efficient the non-agnostic active learning algorithms will use less energy. This extra energy saved could be either used to make the embedded device last longer on a battery or to increase the training iterations done to achieve a higher accuracy.

7.2 Stream-Based Selective Sampling

The active learning algorithms tested in this research were all pool-based, so they had to predict the entire unlabelled dataset to create a query. More energy savings could be obtained by using stream-based selective sampling, where the unlabelled dataset is predicted until an instance is found that meets a threshold of informativeness. The chance exists that this instance is not the most informative instance in the entire dataset so less will be learnt from it, but the trade-off is that there will be less predicting and thus less energy is used.

7.3 Sensor Data

Because the main focus of this research is the effects of active learning on energy usage we opted to use an existing labelled dataset, but a big application of embedded machine learning is activity recognition, where data is continuously collected by a sensor. By combining this incoming sensor data with active learning the algorithm could learn only from select incoming data, and continuously learn while being in use.

7.4 Other optimisations

Embedded applications are not only constrained by their batteries, but also by other resources such as RAM or CPU power. This research only tries to optimise the energy usage, but active learning could also be used to for example reduce the number of training samples that have to be stored, thus reducing RAM and storage usage.

References

[1] L. Andrade, A. Prost-Boucle, and F. Pétrot. Overview of the state of the art in embedded machine learning. In *Proceedings of the 2018 Design, Automation and Test in Europe Conference and Exhibition, DATE 2018*, volume 2018-January, pages 1033–1038, 2018.

[2] A. Bharathi and K. Anandakumar. Investigation on cancer classification using machine learning approaches. *Journal of Biomaterials and Tissue Engineering*, 4(6):492–500, 2014.

[3] Y. Chen, B. Zheng, Z. Zhang, Q. Wang, C. Shen, and Q. Zhang. Deep learning on mobile and embedded devices: State-of-the-art, challenges, and future directions. *ACM Comput. Surv.*, 53(4), Aug. 2020.

[4] K. De Angeli, S. Gao, M. Alawad, H. . Yoon, N. Schaefferkoetter, X. . Wu, E. B. Durbin, J. Doherty, A. Stroup, L. Coyle, L. Penberthy, and G. Tourassi. Deep active learning for classifying cancer pathology reports. *BMC Bioinformatics*, 22(1), 2021.

[5] W. Emily, C. Christy, M. Jurgen, V.-D. J. A., Y. Juan, D. Tania, E. K. A., W. Anthony, and K. Jasmeet. Evaluation of sampling frequency, window size and sensor position for classification of sheep behaviour. 2018.

[6] X. Fafoutis, L. Marchegiani, A. Elsts, J. Pope, R. Piechocki, and I. Craddock. Extending the battery lifetime of wearable sensors with embedded machine learning. In *IEEE World Forum on Internet of Things, WF-IoT 2018 - Proceedings*, volume 2018-January, pages 269–274, 2018.

[7] GreatScott! Make your own power meter/logger. `https://www.youtube.com/watch?v=lrugreN2K4w/`, 2016.

[8] GreatScottLab. Make your own power meter/logger. `https://www.instructables.com/Make-Your-Own-Power-MeterLogger/`, 2016.

[9] L. Johnson. Tensorflow 2 on raspberry pi. `https://bit.ly/3x44FVS`, 2020.

[10] J. Kamminga, H. Bisby, D. Le, N. Meratnia, and P. Havinga. Generic online animal activity recognition on collar tags. In *UbiComp'17*, pages 597–606. ACM Press, 2017. 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp 2017, UbiComp ; Conference date: 11-09-2017 Through 15-09-2017.

[11] J. Lee, M. Stanley, A. Spanias, and C. Tepedelenlioglu. Integrating machine learning in embedded sensor systems for internet-of-things applications. In *2016 IEEE International Symposium on Signal Processing and Information Technology, ISSPIT 2016*, pages 290–294, 2017.

[12] L. L. [lhelontra]. tensorflow-on-arm. `https://github.com/lhelontra/tensorflow-on-arm/release/`, 2020.

[13] X. Li and Y. Guo. Adaptive active learning for image classification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 859–866, 2013.

[14] D. Pereira-Santos, R. B. C. Prudêncio, and A. C. P. L. F. de Carvalho. Empirical investigation of active learning strategies. *Neurocomputing*, 326-327:15–27, 2019.

[15] B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[16] F. Shahmohammadi, A. Hosseini, C. E. King, and M. Sarrafzadeh. Smartwatch based activity recognition using active learning. In *Proceedings - 2017 IEEE 2nd International Conference on Connected Health: Applications, Systems and Engineering Technologies, CHASE 2017*, pages 321–329, 2017.

[17] B. Sharma, J. K. P. S. Yadav, and S. Yadav. Predict crop production in india using machine learning technique: A survey. In *ICRITO 2020 - IEEE 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)*, pages 993–997, 2020.

[18] T. Yao, W. Wang, and Y. Gu. A deep multiview active learning for large-scale image classification. *Mathematical Problems in Engineering*, 2020.