Addressing the Constraints of Active Learning on the Edge

1st Enrique Nueve
NAISE
Northwestern University
Evanston, Illinois, USA
enrique.iv@northwestern.edu

2nd Sean Shahkarami
Urban CCD
University of Chicago
Chicago, Illinois, USA
sshahkarami@uchicago.edu

3rd Seongha Park
Mathematics and Computer Science Division
Argonne National Laboratory
Lemont, Illinois, USA
seongha.park@anl.gov

4th Nicola Ferrier
Mathematics and Computer Science Division
Argonne National Laboratory
Lemont, Illinois, USA
nferrier@anl.gov

***Abstract*—The design of machine learning methodology often does not take into account the limitations of edge computing. In particular, active learning approaches have not considered the constraints of the edge, such as separate data locations (labeled data is on the cloud whereas unlabeled data is on the edge), cold starting or low initial model performance, limited budget sizes due to bandwidth constraints, and computational constraints due to edge hardware. Active learning on the edge could help decide what data to cache on the edge and what data to prioritize for offloading, facilitating efficient use of memory and bandwidth resources. Active learning on the edge would also allow for a machine learning model to be trained using a minimal amount of data. In this work, we examine the constraints of performing active learning on the edge, propose an active learning method that seeks to address these constraints, and discuss advances needed at large to improve active learning on the edge.**

***Index Terms*—Edge, Active Learning, Caching, Offloading, Edge Inference, Sampling**

I. Introduction

With advances in both machine learning and edge computing, an approach to performing computations at the source of data collection over a distributed network, practitioners seek to circumvent issues with existing cloud-based deployments of AI by running models near input data sources [1], [2]. Running a machine learning model at the source of data collection removes the need to transfer data over a distributed network to be used by a machine learning model but instead only requires the model's output to be transferred. However, to deploy a machine learning model on the edge, sufficiently large sets of training data must be collected. This may be a challenging task because of the computational and bandwidth cost of transferring data over a wireless network [3]. Active learning could be used to minimize the amount of data that needs to be transferred from the edge for model training. In active learning, data samples are ranked by their ability to most improve a machine learning model's performance if the samples were used during training of the model [4]. Thus, active learning on the edge could help decide what data to cache on the edge and what data to prioritize for offloading, allowing for efficient use of memory and bandwidth resources. However, most of the current active learning methodologies, such as [5]–[9], do not take into account the unique constraints of the edge.

In an edge computing architecture, extra constraints are placed on active learning, such as separated data (labeled data on the cloud while unlabeled data is on a remote edge device), cold starting (no initial training of the model on source data), or low initial model performance, limited budget sizes due to bandwidth constraints, and computational constraints depending on the edge device's hardware. In this work, we examine the unique constraints of performing active learning on the edge. We propose and test an approach for active learning that is edge compatible focused around the computational constraints on the edge, and we discuss the needed advancements in edge computing to further the capabilities of active learning on the edge. Our proposed active learning methodology addresses the mentioned constraints of active learning on the edge and can be scaled with respect to the edge network's resources.

Code: https://github.com/EnriqueNueve/Addressing-the-Constraints-of-Active-Learning-on-the-Edge

II. Related work

Qian et al. [10] explore the task of data collection and training over an edge network. They propose a combination of active learning on edge nodes and federated learning over

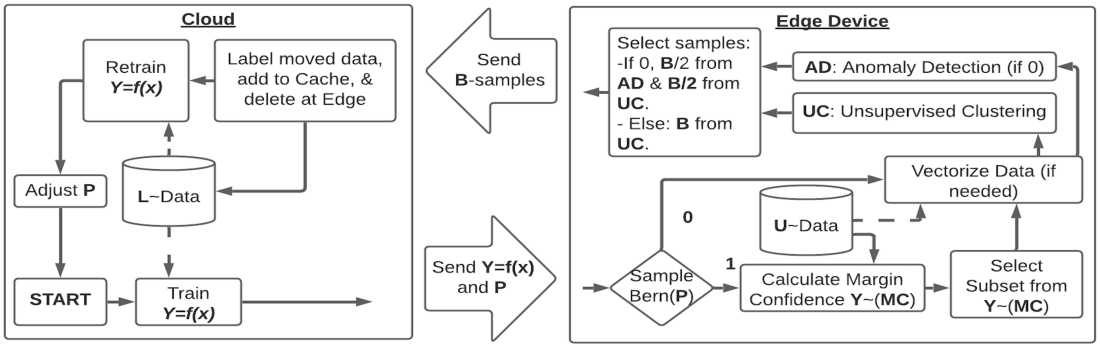

Figure 1: Diagram of proposed active learning framework for the edge. $Y = f(x)$ denotes the underlying model being trained. Bern(P) denotes the Bernoulli process used to determine whether or not to take into account $Y \sim (MC)$'s uncertainty with margin confidence.

intermediary fog nodes. Although [10] their work shows promising results, the assumption that federated learning is possible over a fog node may not be true for all cases.

In itself, federated learning provides many positive aspects, such as privacy of data, efficient use of network bandwidth, and low latency [11]. However, federated learning assumes that one can compute the gradient of a deep learning model from the data cached at the edge. For a supervised model, this would require access to the data sample's label at the edge in order to calculate the gradient. Assuming access to labels at the edge may be unrealistic for applications such as image classification and object identification. Fortunately, strictly using active learning could help optimize training on the edge in more cases since active learning does not assume access to labels. Unfortunately, many current active learning methods are not edge compatible.

A multitude of methods such as Discriminative Active Learning [5], Adversarial Active Learning [6], Deep Bayesian Active Learning [7], Multi-class Active Learning by Uncertainty Sampling with Diversity Maximization [8], and Learning to Sample [9] have shown advancements in active learning. However, these methods all require the edge network to be able to offload batches containing thousands of samples and to be able to train a deep learning model on the edge. Current hardware and bandwidth constraints of most edge networks make such methods infeasible for use on the edge. To address the current limitations of active learning in an edge computing scenario, we propose a new active learning methodology that is edge compatible.

III. Methodology

We propose an active learning framework for edge computing influenced by the seminal work on experimental design [12] and recent work in active learning [13], [14] that uses representative and uncertainty sampling. Our proposed active learning method uses a dynamic Bernoulli process to select the most appropriate method for sampling based on the model's performance. When the model's performance is low, representative and uncertainty sampling are used to get a mix of data near decision boundaries and dense regions of the training data in order to prevent the model from converging to a false minimum. As the performance of the model increases, uncertainty sampling alone is performed more often, allowing for more samples to be taken near the decision boundaries to improve performance in later stages of training.

In the following paragraphs, we present our methodology that meets all of the cited edge constraints, including being computationally efficient, having the ability to start cold, and not needing access to labels at the edge. All of the methods used within ours are computationally lighter than benchmarked Deep learning models that have been shown to perform efficiently on an array of modern edge devices [15]. Thus, our proposed methodology should be compatible with an extensive array of current edge devices. Before discussing our method specifically, we cover the needed nomenclature used to describe our framework.

Let $L = \left[(x_i, y_i)\right]_{i=1}^{l}$ denote a sequence of labeled training data. L is located on the cloud or a local device, and the size of this dataset will change as samples are added. Let $U = \left[(x_i)\right]_{i=1}^{u}$ denote a sequence of unlabeled data. U is located on a remote edge device. Active learning assumes there are a fixed number of rounds, denoted by N, in which a fixed number of samples are chosen from U and then labeled and moved into L each round. Let B, known as the budget, denote the number of samples chosen for each round. Assume there is a machine learning model, denoted by Y. In each round, Y is retrained on the updated labeled dataset L. The overall objective is to maximize the performance of Y with respect to the fixed number of rounds, N, and the fixed budget, B.

To begin, we train Y on the labeled data, L, located on the cloud or a local device. We then upload Y to our remote edge device and begin our active learning process. For our method, the choice of sampling from U on the edge can occur in two ways. One way takes into account the uncertainty of the model, and the other does not. The choice of method is determined by the outcome of sampling a Bernoulli distribution. We denote the parameter of the Bernoulli distribution by $P \in [0, 1]$. If the sampled value is zero, we do not take into account the model's uncertainty and vice versa if the sampled value is one.

We expect the accuracy will improve if we avoid taking into

	A	B	C	D	E	F	G	H
Vector	HOG	HOG	HOG	HOG	Flat	Flat	Flat	Flat
Cluster	Gaussian Mixture	Gaussian Mixture	K-means	K-means	Gaussian Mixture	Gaussian Mixture	K-means	K-means
Anomaly	Isolation Forest	One Class SVM	Isolation Forest	One Class SVM	Isolation Forest	One Class SVM	Isolation Forest	One Class SVM

Table I: Framework configurations used in our experiments. The letters at the top of each column denote the configurations in plotting results.

account the model's uncertainty if the model is starting cold or with relatively low initial performance. Thus, by initially setting P low, we are more likely to not take into account the model's uncertainty. This addresses the dilemma of starting cold. As an example, assume we sampled the value zero. With the data from U, on the edge, we perform clustering as well as anomaly detection. For clustering many suitable methods exist, for example, K-means or a Gaussian Mixture model. As a default, the number of clusters is set to $B/2$. We then select $B/2$ samples that are closest to each of the centroids constructed from the unsupervised clustering algorithm. With this procedure, we obtain a diverse set of expressive samples from U, thus performing representative sampling. Through this diverse sampling procedure, we are also more likely to explore new regions of the data; these will help with possible class imbalance in our initial data [13]. However, we still need to perform anomaly detection.

Through clustering, we were able to select samples that were representative; however, by performing anomaly detection on U, we are able to identify outliers. Anomaly detection can be performed by using algorithms such as Isolation Forest [16] or One-Class SVM [17]. Using an anomaly detection method, we select $B/2$ pieces of data that are considered outliers. Suppose there are overlaps between samples chosen from clustering and anomaly detection. Then the next highest-ranked outliers are selected to replace an overlapped sample, and so on for any samples after that, until we have a total of $B/2$ unique samples chosen from anomaly detection.

The selected samples are sent back to the cloud or local device, labeled, and added to L. Y then is retrained on the updated version of L. Based on the change of performance of Y on test data, the value of P is increased or decreased.

If Y improves, P is increased, thus increasing the chances the Bernoulli samples a one, which causes the sample selection procedure to incorporate the model's interpretation of the data. To see this, assume we performed another round of active learning, and this time a one was sampled from the Bernoulli. We then perform inference on U at the edge with Y. Assuming the task is classification, a prediction vector $\hat{y}_i$ is produced for each sample from U. For each of the prediction vectors $\hat{y}_i$, we calculate the prediction's margin confidence, denoted by c_i. The margin confidence of a prediction vector is calculated as $c_i = y_i^1 - y_i^2$, where y_i^1 is the highest value in $\hat{y}_i$ and y_i^2 is the second highest value in $\hat{y}_i$.

Upon calculating the margin confidence for each sample of U, we select a subset based on the samples with the highest margin confidence scores. The size of this subset depends on the sizes of U and B. The subset must be larger than B. The reason is that this subset is then used to perform clustering (as done when a 0 is drawn), and the B samples closest to the centroids are then selected for labeling.

During our experiments, we set the subset size to be three times as large as B. This is the same procedure as when a zero is sampled from the Bernoulli but incorporates a prefiltering of data based on the uncertainty measurement from margin confidence. Thus, when a one is sampled from the Bernoulli, the sampling process incorporates both uncertainty and representative sampling. The whole procedure is schematically presented in Figure 1.

Although we present a full framework, we developed it as a modular system. We chose margin confidence because it is computationally cheap. However, based on one's respective task and computational resources, this uncertainty measurement could be replaced with a more powerful uncertainty measurement such as MC Dropout [18], Deep Ensembles [19], or Orthonormal Certificates [20]. We used clustering to determine representative samples, but other approaches can be substituted for the K-means or Gaussian Mixture model used here. The adjustment of the P-value for the Bernoulli can also be tuned for a specific application. For our experiments, we set P for each round to be equal to the achieved validation accuracy after retraining Y, and thus P increased as the performance of the model increased. However, P's adjustment could also be altered through more complicated methods such as using an exponential moving average of past validation accuracy or with reinforcement learning.

IV. Experiments and Results

To explore the capabilities of our proposed framework, we conducted three experiments. These experiments involved performing active learning on different image classification datasets using CNNs. The three datasets consisted of one standard benchmark dataset, MNIST, and two non-benchmark natural image datasets, a bee[1] and a monkey[2] image dataset. The bee and monkey datasets were from Systema Naturae, a project for collecting and distributing wildlife datasets.

To evaluate the performance of our proposed framework, we compare it with active learning with both passive sampling (uniformly sampling over the available cache) and margin

[1] https://www.systemanaturae.org/wildlifedatasets/annotated-honey-bee-images/

[2] https://www.systemanaturae.org/wildlifedatasets/10-monkey-species/

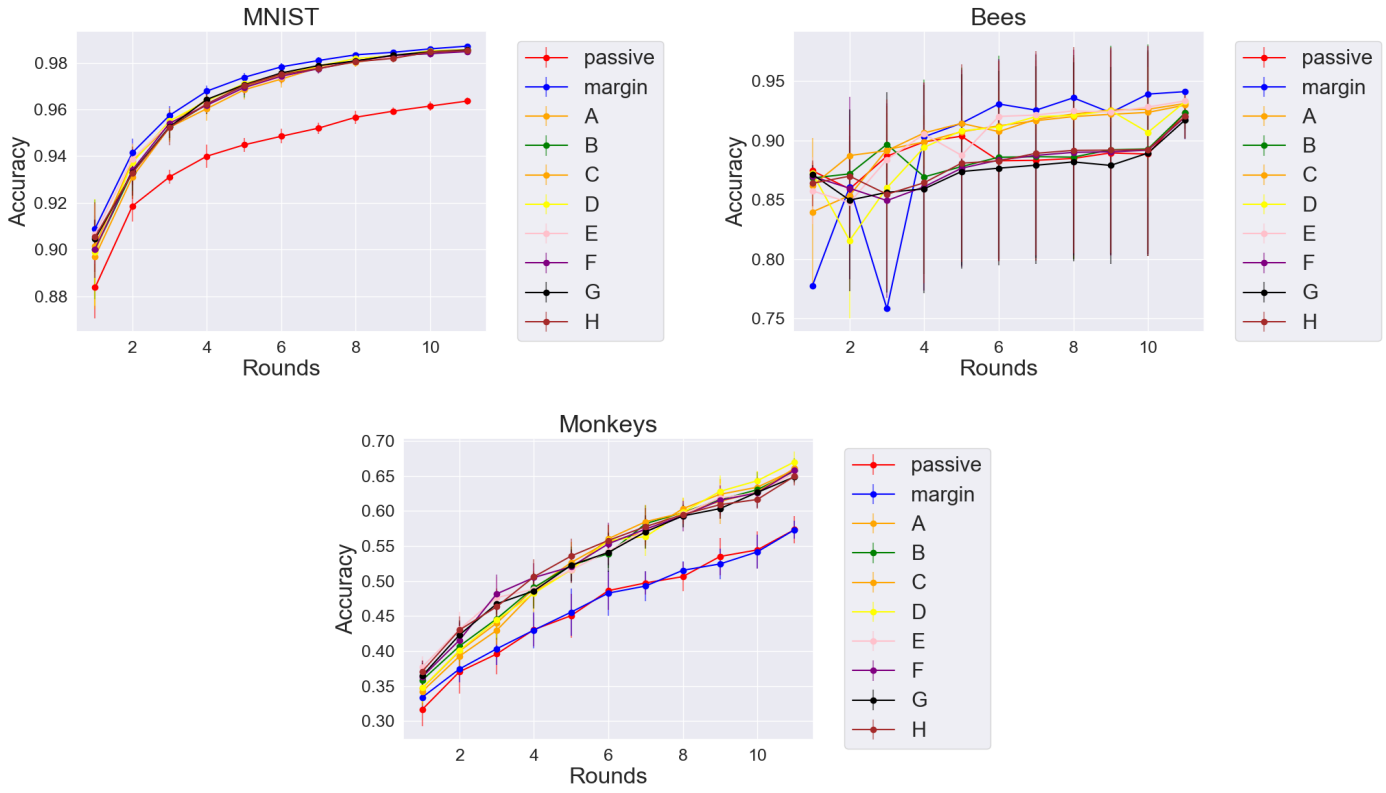

Figure 2: Experiment results: plots of accuracy vs number of rounds of the active learning "cycle." The letters in the legend correspond to the configurations in the columns of Table I. The results express the mean and standard deviation over ten repeated trials for each method over each dataset.

confidence sampling. To explore how our framework could be configured based on one's available computational resources and particular learning problem, we tested a series of eight configurations of our framework. In particular, we substituted different parts of the framework, such as how we converted our image data into vectors, given that our clustering and anomaly detection methods required our data to be shaped as vectors. For vectorizing our data, we implemented flattening and using a histogram of oriented gradients [21]. For clustering, we implemented both Gaussian Mixture models and K-means. For anomaly detection, both Isolation Forest and One-Class SVM were used. The different configurations are expressed by letters A through H, as shown in Table I. For each combination of dataset and active learning method, ten repeated trials were performed in which ten rounds of active learning occurred in each.

For our experiments, P was initially set to the value of the accuracy on the validation set. Before conducting the active learning experiments, 100 data samples from each of the datasets were used to initially train each respective model. Twenty percent of the data was set aside for validation. The remainder was used in a pool for active learning. MNIST contains a total of 70,000 images; the bee dataset contains 4,744 images; and the monkey dataset contains 1,368 images. For each experiment ten rounds of sampling and retraining of the models were performed. In the experiments, 100 samples were chosen each round when testing with the MNIST and bee dataset, while 50 samples were chosen each round during testing with the monkey dataset. These assigned budget sizes were chosen to be small in order to portray the limited abilities to move data over an edge network given bandwidth constraints.

The outcomes of the experiments are shown in Table II and Figure 2. The results indicate that our framework performed on a par with margin sampling using the MNIST dataset and far above passive sampling. For the bees dataset, the results show that our methodology did well during the earlier rounds when accuracy was low, but margin sampling was better in the later rounds. The results of the bee dataset shows rather larger variance in performance. We believe this was due to the bee dataset being rather skewed in regard to class balance. The results show that our proposed methodology outperformed both passive and margin sampling for the monkey dataset.

The results suggest that our proposed framework performs well when the starting accuracy is lower, as demonstrated in the bee and monkey datasets. However, our method performs on a par or slightly worse when the accuracy increases. Because margin confidence outperforms our proposed method when accuracy starts to get closer to perfect, we believe that a method that causes the Bernoulli parameter to favor sampling

Data set (size)	Proposed	Passive	Margin
MNIST (100)	0.99 ± 0.00	0.96 ± 0.00	0.99 ± 0.0
Bees (100)	0.93 ± 0.012	0.92 ± 0.01	0.94 ± 0.00
Monkeys (50)	0.67 ± 0.01	0.57 ± 0.02	0.57 ± 0.01

Table II: Experiment results: The accuracy of the model $\pm\sigma$ is given for our proposed approach over ten trials. The results for our proposed approach used configuration (D) for MNIST and the monkey datasets and (E) for the bees dataset. These configurations, from Table I, produced the highest accuracy using our approach. For comparison, we also present the accuracy of passive and margin sampling.

using model confidence should be utilized more frequently.

V. Future Work

Our proposed methodology is edge compatible and outperforms passive sampling; however, when compared with current best-in-class active learning methods, [5]–[9], our proposed framework underperforms. The extra constraints of performing active learning on the edge limits the choice of feasible methods that can be used. Key challenges such as model's starting cold, how to train model's given the separation of data (labeled data is on the cloud while unlabeled data is on the edge), and smaller than usual budget sizes for sampling during active learning given bandwidth and computational constraints need to be addressed to further the abilities of active learning on the edge.

Further work is also needed to develop our framework's method for adjusting Bernoulli's parameter value. By improving our method's ability to choose whether to select data based on uncertainty or whether the data is representative, our method would likely perform better during the early and later stages of training.

Additionally, we hope to explore better the tradeoffs of budget size and an edge computing network's ability to transfer data. In our experiments, we tested with smaller budget sizes, 50 and 100, since we assumed limited bandwidth resources. However, the ability to move data over an edge network varies from system to system. Thus, we believe it would be insightful to see how performance for active learning on the edge varies given different assumed feasible budget sizes.

VI. Conclusion

Through this work, we provide insight into how constraints on the edge may affect active learning. With our proposed framework, we offer a viable procedure for active learning on the edge that can be easily modified to fit one's machine learning task and edge-related constraints. For future work, we intend to look further into different methodology for our framework's modular pieces in order to understand better what methods serve best for various use cases. With the unique constraints in an edge computing scenario, new edge-compatible active learning algorithms clearly are needed.

Acknowledgments

This work was supported by a National Science Foundation's Mid-Scale Research Infrastructure grant, NSF-OAC-1935984 [22] and by the U.S. Department of Energy, Office of Science, under Contract No. DE-AC02-06CH11357.

References

[1] P. Beckman, C. Catlett, I. Altintas, E. Kelly, S. Collis, N. Ferrier, M. Papka, J. Olds, D. Reed, and R. Sankaran, "Sage: Cyberinfrastructure for AI at the Edge," https://sagecontinuum.org/, October 2019, (accessed: 08/28/2020).

[2] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[3] D. Xu, T. Li, Y. Li, X. Su, S. Tarkoma, T. Jiang, J. Crowcroft, and P. Hui, "Edge intelligence: Architectures, challenges, and applications," 2020.

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

[5] D. Gissin and S. Shalev-Shwartz, "Discriminative active learning," *arXiv preprint arXiv:1907.06347*, 2019.

[6] M. Ducoffe and F. Precioso, "Adversarial active learning for deep networks: a margin based approach," *arXiv preprint arXiv:1802.09841*, 2018.

[7] Y. Gal, R. Islam, and Z. Ghahramani, "Deep Bayesian active learning with image data," *arXiv preprint arXiv:1703.02910*, 2017.

[8] Y. Yang, Z. Ma, F. Nie, X. Chang, and A. G. Hauptmann, "Multi-class active learning by uncertainty sampling with diversity maximization," *International Journal of Computer Vision*, vol. 113, no. 2, pp. 113–127, 2015.

[9] J. Shao, Q. Wang, and F. Liu, "Learning to sample: an active learning framework," in *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2019, pp. 538–547.

[10] J. Qian, S. Sengupta, and L. K. Hansen, "Active learning solution on distributed edge computing," *arXiv preprint arXiv:1906.10718*, 2019.

[11] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, 2020.

[12] D. V. Lindley, "On a measure of the information provided by an experiment," *The Annals of Mathematical Statistics*, pp. 986–1005, 1956.

[13] A. Kazerouni, Q. Zhao, J. Xie, S. Tata, and M. Najork, "Active learning for skewed data sets," 2020.

[14] F. Zhdanov, "Diverse mini-batch active learning," 2019.

[15] P. KANG and J. JO, "Benchmarking modern edge devices for ai applications," *IEICE Transactions on Information and Systems*, vol. 104, no. 3, pp. 394–403, 2021.

[16] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*. IEEE, 2008, pp. 413–422.

[17] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, J. C. Platt *et al.*, "Support vector method for novelty detection." in *NIPS*, vol. 12. Citeseer, 1999, pp. 582–588.

[18] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[19] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *arXiv preprint arXiv:1612.01474*, 2016.

[20] N. Tagasovska and D. Lopez-Paz, "Single-model uncertainties for deep learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 6417–6428.

[21] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1. Ieee, 2005, pp. 886–893.

[22] P. Beckman, C. Catlett, I. Altintas, E. Kelly, and S. Collis, "Mid-scale RI-1: SAGE: A software-defined sensor network (NSF OAC 1935984)," 2019, https://sagecontinuum.org/.