TrafficNNode: Low Power Vehicle Sensing Platform for Smart Cities

Justin Nguyen
Electrical and Computer Engineering
Carnegie Mellon University
Mountain View, CA USA
justin.nguyen@sv.cmu.edu

Reese Grimsley
Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA USA
reeseg@cmu.edu

Bob Iannucci
Electrical and Computer Engineering
Carnegie Mellon University
Mountain View, CA USA
bob@sv.cmu.edu

***Abstract*—Automating traffic monitoring and management can materially contribute to the realization of Smart Cities, but this demands detailed, accurate and timely characterization of traffic flows. Current methods employ video capture (high installation and operating costs), pneumatic-tube-based counters (limited detail about vehicle types and often only installed temporarily), or manual data capture (high human cost).**

We have developed an intelligent, inexpensive, pavement-mountable device capable of collecting information about vehicle types and speeds using an embedded, power-optimized neural network. The device is designed to fit within a raised pavement marker (RPM). RPMs are deployed throughout the world as lane markers. Packaging our sensing technology into RPMs offers the potential to significantly improve the spatial and temporal resolution of traffic flow information across a city.

We outline our methodology for energy-optimized machine learning on a small, resource-constrained sensor device. We present the results of our work in terms of accuracy (96% classifying vehicle type and 89% classifying vehicle speed) and battery life (three years).

***Index Terms*—tinyML; embeddedML; traffic sensing; Smart Cities; IoT; RNN; Low-Power.**

I. Introduction

Smart Cities employ pervasive sensing networks that enable intelligence in city infrastructure, resource management, and policy to improve the well-being of the city and its inhabitants [1]. Roads are a critical component of city infrastructure, and applying concepts from smart cities to their improvement can yield substantial, widespread benefit.

To better understand roadways and their use, cities need to measure quantities like traffic volumes, the types of vehicles, and their speed [2]. But such measurements become actionable only when they are precise in space (down to the road segment) and time (based on time-of-day, day-of-week, season); discriminating of vehicle types; and available within minutes or seconds. For instance, city planners might learn through such measurements that a disproportionate number of large utility vehicles travel through side streets during rush hour to avoid a left turn at a busy traffic light, resulting in congestion for that side street. Armed with this knowledge, signage, traffic signal re-timings and other simple interventions can be implemented to reduce the congestion and re-balance the traffic flows in that area. Such improvements benefit drivers, residents, business-owners, and policymakers [3].

To facilitate fine-grain data collection at a city-scale, tens of thousands of sensors (or more) will be needed [4]. Existing traffic monitoring solutions (video, inductive loop, pneumatic tube) would incur high installation and operating costs to achieve the necessary level of detail. Suitable traffic sensors need to be inexpensive to manufacture, deploy, and maintain, and should last as long as the rest of the city's roadway system, *i.e.*, multiple years without constant attention.

A long-lasting, battery-powered solution is the most cost-effective one. Battery powered platforms are cheaper to deploy because they avoid the cost of installing mains power. Battery-powered sensors with short lifetimes incur significant maintenance costs, especially if they are situated in the roadways, as technicians must frequently visit thousands of devices to swap batteries, temporarily closing roadways in the process. Such sensors must last for years between required maintenance, just like a stop sign or a traffic light.

Nellore [2] provides an overview of known traffic sensing technologies. Given our power constraints, the only sensors that meet our requirements are low-powered MEMS transducers such as accelerometers, magnetometers, or microphones. Other studies have proposed sensing based on inductive loops [5]–[7] which use traditional signal processing techniques to determine speed or vehicle type. In this work, we demonstrate that it is possible to extract vehicle speed and type by combining mature three-axis magnetometer devices with embedded machine learning, all of which can be packaged in a compact, battery-operated device that can live on road surfaces for three to five years without maintenance. We present the somewhat counter-intuitive result that performing this computation in the device and only sending summary information wirelessly actually runs at lower average power than streaming sensed readings and performing the neural network computations at the edge or in the cloud.

In this work, we develop a gated recurrent unit (GRU) [8] classification model which can be deployed on a microcontroller. We evaluate this on our custom hardware (based on a commodity ARM Cortex-M4 processor) using vehicular data collected for this purpose. The entire sensor–the *TrafficNNode*–includes a low-power wide-area network radio and a suitable battery, and is designed to fit within a commodity Raised Pavement Marker (RPM). RPMs are widely deployed on roads

world wide and are excellent vantage points for observing traffic flows. Our classification model is capable of counting vehicles, classifying them by type and speed range with 96% and 89% accuracy, respectively. Our device has an estimated life time of up to three years for a 2 Ah $Li - SOCl_2$ battery.

Our contributions include:

- An in-depth feasibility analysis of a battery-powered IoT sensor with embedded machine learning for a Smart City traffic measurement system
- Domain-specific optimization techniques and methodologies to improve the battery life for embedded machine learning without sacrificing accuracy

II. Related Work

Traffic measurement systems (TMS) for Smart Cities are the building blocks upon which we build our work. Table-1 of [2] provides a concise overview of infrastructure-based traffic sensing technologies. TMS's and their impact are discussed in [2], [9]–[11]. These works examine various measurement methods, the majority of which would not scale for fine-grained measurements since they either rely too heavily on existing infrastructure or their sensing modalities are too power-hungry for deployment on cheap, battery-powered sensors. Additionally, [12], [13] explore the use of vehicle-based and fog-computation data collection methods while [14] covers sensor-fusion/modelling strategies. While these methodologies do not have any reliance on infrastructure, they raise several privacy and security concerns [15] that make their general acceptance a challenge for city-scale deployment.

Prior literature has identified magnetic flux as a useful modality for sensing vehicles, given the large amount of ferromagnetic material present in automobiles [5]–[7]. Traffic lights are often informed of vehicle presence by inductive loops installed beneath the road surface, which requires costly infrastructure changes. MEMS magnetoresistive sensors have become common transducers in a variety of devices, often acting as compasses in mobile phones. These sensors produce 3-axis, time-series flux readings at up to 1 kHz while consuming less than 1mA, and they can be used with a variety of signal processing techniques. A vehicle speed estimation study using AMR sensors [7] implemented a graph based approach to determine vehicle speed on an RPM-like wireless sensor. However, they are unable to determine the traffic composition, *i.e.,* vehicle types, and their system will only last one year on a 5 Ah battery. We build upon these studies with a neural-network based approach that was infeasible until recent advances in tinyML and modern microcontrollers.

Successful Smart Cities need to leverage big data and machine learning. This could be implemented in the cloud, but the growing rate of data collection has quickly outpaced network infrastructure growth [16]. The dichotomy between cloud and edge computing has been thoroughly discussed in [17] where, in addition to data collection rates and network limitations, energy usage of pervasive IoT sensors warrant local computation whenever possible.

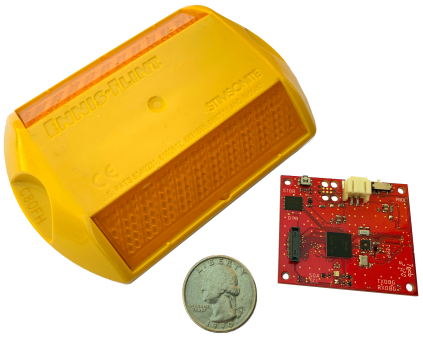


Fig. 1: CommonSense Hardware Platform for Low-Power Sensing

Processor Speed	120 MHz
RAM	256 kB
FLASH	1024 kB
Active Current	21.26 ± 0.072 mA
Sleep Current	35.8 ± 1.80 µA
Magnetometer Active Current (400 Hz)	575 µA
Magnetometer Sleep Current (12.5 Hz)	40µA
LoRa Transmit Power (SF10+14dBm)	33.5 mA

TABLE I: CommonSense Hardware Parameters

The field of tinyML, *i.e.*, machine learning on deeply embedded devices, is relatively new. Two leading tools, TFLM [18] and uTensor [19] make the deployment of neural networks on microcontrollers computationally feasible: Microcontrollers have limited memory and computation resources, so these tools seek to reduce related requirements without losing more than a few percent of accuracy. Common improvements include quantizing floating-point weights, which require many compute cycles to use, into fixed-point weights as well as replacing small weights with zeros, *i.e.,* sparsifying, to reduce the total number of operations. These improvements allow neural networks to fit within the small RAM sizes (<1MB) and run the pre-trained model more quickly. Several studies have been performed [20]–[22] to look at how neural network architecture can modified to improve energy efficiency, generally by reducing the computation time, which has a linear relationship with the total energy consumed. This can be further optimized by leveraging application knowledge, which we describe in Section IV.

III. Sensor, Dataset and Baseline Models

For our system, we target a custom hardware design, CommonSense, that is built for prototyping low-power IoT sensors with substantial compute capabilities. CommonSense, shown in Figure 1 uses the ATSAMD51 Cortex-M4 processor with floating point support, and has an extensible interface for supporting new radios or sensors. We use an NXP FXOS8700CQ 3-axis magnetometer and a Semtech SX1261 LoRa radio for this application. Table I describes the relevant parameters of this platform with respect to computation and energy consumption.

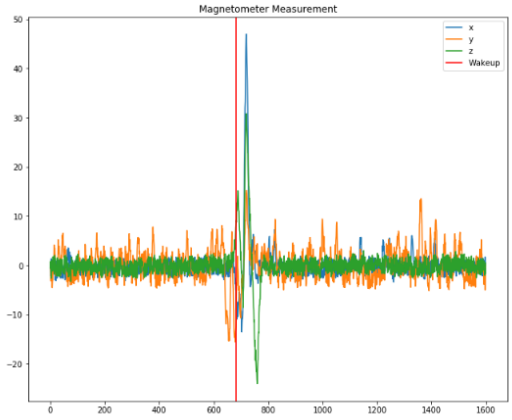
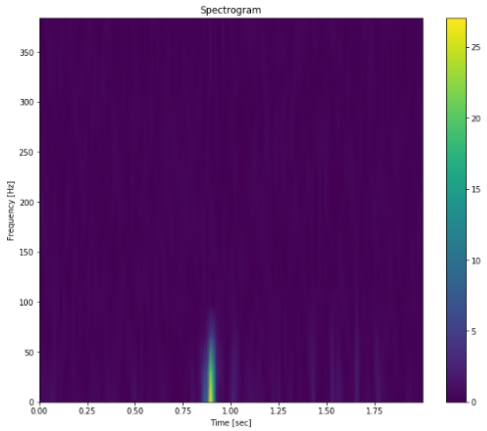


Fig. 2: 3-axis magnetometer trace of a passing vehicle and associated spectrogram of total magnitude. High ambient noise requires debouncing to reliably detect a vehicle.



Our training and test datasets for classification of vehicle types and speeds were built across three separate events. Drivers were recruited and test tracks were set up at two safe, off-highway locations. Several prototype TrafficNNodes were mounted to the pavement in the center of and offset from the centerline of travel.

The vehicles were queued a few hundred meters up the road from the sensors to give ample distance to stabilize at target speed. A radar gun was used to capture actual speed as each vehicle transited the sensor area. Each TrafficNNode recorded three-axis magnetometer traces for each passing event. This was repeated for each of the five speed classes for each of the eight vehicle six times identically on two separate days. The vehicles utilized consists of three SUVs, four sedans, and one police cruiser (a sedan that is heavier than all other vehicles in the dataset). In total, the dataset consists of 500 vehicle measurements, evenly split among the five speed classes and eight vehicle types, which we augment with AWGN noise, time-shifts, and amplitude scaling to increase the dataset size by a factor of 20 for a total of 10,000 measurements.

IV. Methodology

This section follows the step-by-step approach that we took to design and optimize the classification model for energy efficient deployment of TrafficNNode. We begin from the baseline model with no constraint on network size, computation time, or memory footprint. We implement optimizations to reduce power without sacrificing more than a few percent of recognition accuracy for vehicle type and speed. We start with standard tinyML optimizations like reducing network size, quantizing weights, and sparsifying the model [22], although in accordance with that survey, there we found no improvement by quantizing weights to integers on our microcontroller that has hardware floating point support. Next, we perform application-specific optimizations, such as reducing the sample rate, to further reduce energy usage on a single sensor. Finally, we propose several network-level optimizations, which leverages load-balancing across several sensors to reduce energy usage even further.

A. TrafficNNode Model

Our traffic-sensing model is built using the Tensorflow and TFLM [18]. TFLM provides a portable runtime which reads in a compiled Tensorflow Graph binary. TFLM then constructs the graph and executes it at runtime.

The network is a many-to-one recurrent classifier where the recurrent neural network (RNN) is applied over the entire length of the sequence (here, 3-axis magnetometer samples), and the final hidden state is passed to a fully-connected layer with softmax activation to perform the inference. Our network has two dense, *i.e.,* fully-connected, output layers, which are jointly trained to classify both the vehicle and the speed class. We ran several experiments with Elman (traditional RNN), long-short term memory (LSTM), and gated-recurrent unit (GRU) cells as the recurrent node. The GRU cell was selected as it showed the best balance of accuracy vs. computation time; this is due to the reduced number of instructions compared to the LSTM cell (GRU does not have an output gate) and better long-term memory compared to the Elman network [8]. Our initial RNN achieved 99.2% accuracy where the RNN hidden layer contains 18 neurons.

B. Metrics

To evaluate performance, we will use accuracy and computation time as our primary metrics. Computation time is a direct proxy for energy usage because it has a direct linear correlation as shown by [21]. Intuitively, more time spent computing means more time spent in a high-powered active state *vs.* putting the device into an ultra-low power sleep state. Equations 2 and 4 translate the computation time into energy usage over the course of a day.

$$t_{comp/day} = \text{AADT} \times (Lt_{RNNstep} + t_{projection}) \quad (1)$$

$$E_{comp/day} = V_{batt} I_{comp} \frac{t_{comp/day}}{86400 \frac{s}{day}} \quad (2)$$

$$E_{sleep/day} = V_{batt} I_{sleep} (1 - \frac{t_{comp,perday}}{86400 \frac{s}{day}}) \quad (3)$$

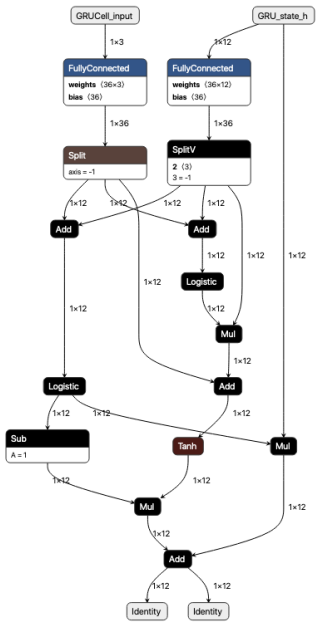


Fig. 3: Operations within the GRU cell.

where AADT is the annual average daily throughput (a metric describing vehicles per day), L is the sequence length [1], and t_{layer} is the time in seconds to compute a single forward propagation for the $layer$ in the model. I_{comp} is the average measured current consumption of the processor when performing the inference and V_{batt} is the supply voltage from the battery, which we assume to be constant throughout the vast majority of the battery's life cycle. I_{comp} and I_{sleep} are shown in Table I. $E_{comp/day}$ and $E_{sleep/day}$ represent the amount of energy spend over the course of a day performing computation to classify passing vehicles and sleeping, respectively.

The batteries lifetime is represented as:

$$t_{lifetime} = \frac{E_{batt}}{(E_{comp/day} + E_{sleep/day})} \tag{4}$$

where $t_{lifetime}$ is in days and E_{batt} is the battery capacity in Amp-hours.

As we optimize the classification model with respect to energy per vehicle classification, we expect to see a trade off with accuracy.

C. Model Size Tuning

Our first energy optimization is a byproduct of ordinary network hyperparameter tuning, which is necessary regardless of our system constraints. Specifically, we tune the hyperparameter for the number of hidden units, which represents the size of the internal state vector within the GRU cell. A high dimension hidden state may lead to over fitting, and a low dimension will incur under fitting. Here, we prune the network size to reduce overfitting and minimize inference time; this has a significant impact on energy usage.

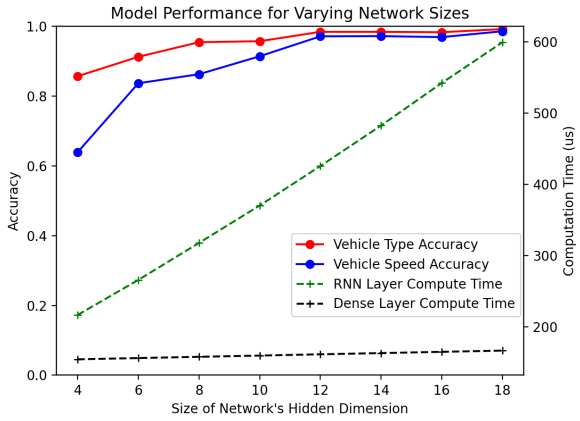


Fig. 4: Classification Model Performance for Varied Network Size

The operations within the GRU [8] cell are displayed in Figure 3. There is a linear relationship between the recurrent hidden dimension and time-complexity as shown in Figure 4, and thus, a linear relation between the number of hidden neurons and the energy cost of a single inference.

We empirically select the best value for the recurrent hidden dimension by sweeping this hyperparameter and analyzing the accuracy and computation time as shown in Figure 4[2]. As expected, computation time changes linearly with this hyperparameter. The accuracy drops as this hyperparameter decreases, but is nonlinear: we observe a 'knee' in this curve around $hiddendim = 6$ and 12, where the accuracy starts to drop more sharply. A $hiddendim = 8$ offers a good tradeoff of high accuracy and low computation time, although we could use a smaller dimension to improve the energy efficiency further at the loss of accuracy. The results following in the rest of the paper will use $hiddendim = 8$.

D. Minimal Sufficient Sampling Rate

Another opportunity to reduce computation time and therefore save energy is by reducing the sampling rate. This would also result in shorter sequence lengths. Our baseline model used every sample, which was collected at 800 Hz, the maximum sampling rate of the magnetometer. However, spectral analysis of our data, an example of which is shown in Figure 2, shows that these signals have little information above 80 Hz, even for vehicles moving at above 60 mph, although the majority of energy is within 0-50 Hz. Therefore, we know that the data sampled at 800 Hz is higher than necessary and that a sampling rate of 160 Hz should be sufficient based on the Nyquist rate of twice the maximal frequency component.

[1]The number of magnetometer samples that contains the vehicle's passing magnetic signature; L is selected as an upper bound across the speed classes, as the signature is longer at lower speeds.

[2]Our assessment of the model performance for analyzing energy usage, the accuracy figures are highly dependent on the dataset we collected which is still quite limited. This is further discussed in Section V.

Decimation Factor	Label Acc.	Speed Acc.	Compute Time (s)
1	0.99496924	0.9860256	0.19136
4	0.99105644	0.98658466	0.09601
8	0.99385130	0.98937952	0.04812

TABLE II: Classification accuracies and compute time for an inference of a single vehicle over a range of decimation factors. The accuracies are for the model predicting vehicle type and speed.

This can be achieved by training and running the network on data at a lower sample rate. When applying this transformation to the data, it is crucial that the downsampling operation does not reduce the data to a point that the model cannot generalize due to the reduced volume of input data resulting in an overfit model. With this in mind, we ran a series of experiments where our network was trained on downsampled variants of the input data, synthesized using a decimation filter that applies an anti-aliasing filter before downsampling such that the apparent sampling rate is no lower than 160Hz.

Since we have shown via the spectrogram that the information we care must reside below 80Hz, downsampling by up to a factor of four should not reduce the information content within the signal. This is observed in Table II where the network accuracy is consistent across a range of decimation factors and the computation time decreases linearly.

As energy consumption is linearly related to L, as shown in Equation 1, so downsampling the signal to the shortest possible representation will save energy with respect to the original signal. To reduce concerns of overfitting the data, we select for a sampling rate of 200 Hz, *i.e.*, a decimation factor of 4, to ensure all spectral content is retained. This reduces the energy consumption per inference by a factor of 4, although the gains in battery life are smaller than this due to power usage by other components, as will show in Section V

E. Delayed Inference

To achieve a long battery life, the processor needs to be duty-cycled such that the hardware spends as much time in the low-powered sleep state as possible. Measuring the entire magnetic signature of a passing vehicle is impractical, since the sensor would need to poll continuously to find the first moment the vehicle nears the magnetometer.

The processor should be duty-cycled such that it is active only when a car is passing. Ideally, the magnetometer too would be duty-cycled to use a lower sampling rate when cars are not passing, but react to significant changes in the magnetic field such that it wakes itself and the processor. The magnetometer we selected implements this functionality via a programmable threshold and debounce timing, which in combination, allow for energy efficient presence detection of vehicles.

Therefore, low-power operation will prevent us from seeing the entire signature of the vehicle; however, we have found that it is sufficient to classify on a subset of that signal. We introduce a degree of freedom called ”delayed inference”, in

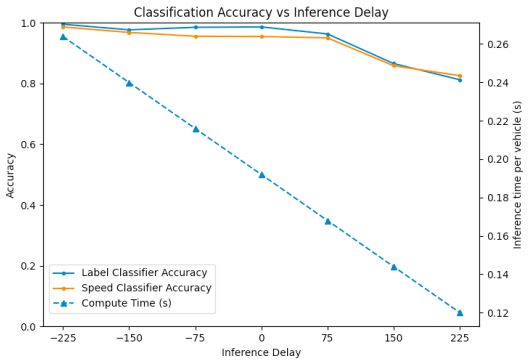


Fig. 5: Classifier accuracy when trained on delay inference from -225 to 225 samples. The delay is performed prior to the downsampling operation.

which the recurrent classifier is not run on the immediately available samples. Instead, we wait a small amount of time such that the initial portion of the signal is ignored, which provides more time to debounce the signal and wake up the processor. We ran an experiment the model's performance while varying this delay from the first magnetometer until we start using current magnetometer samples within the recurrent model. We vary this delay from -225ms to 225ms. A negative delay is meant to emulate the case where we begin running the model before the vehicle affects the magnetic field. A large positive delay may miss the vehicle entirely (depending on its speed). Figure 5 shows the impact on accuracy.

Since there is a linear relationship between energy and sequence length, L, as shown in Equation 1, a larger inference delay should be selected that still exhibits acceptable classification performance. For our network, we observe that an inference delay of 75 samples (with respect to the full 800Hz sampling rate, this is 94ms of delay) would result in the best tradeoff of performance to energy efficiency. Independent of other optimizations, using this 75 sample inference delay would reduce the energy consumption per inference by 12.5%.

F. Network Level Energy Optimization

All the methods discussed previously reduce energy usage for a single sensor, yet sensors such as the TrafficNNode are meant to be deployed en-masse as a sensor network. We consider additional optimizations to improve battery-life across the network of devices by load-balancing between sensors, although this was not strictly measured given that we did not engage in a full deployment of TrafficNNode with our proposed traffic measurement algorithm.

Several load-balancing techniques include:

- All sensors running at once and randomly ignoring some vehicle events independently on each sensor
- N^{th} order strided measurements for sensors placed along the same road, meaning sensors would only perform inference on every one of N vehicles detected; each sensor should be out of phase (not measuring the same vehicles) with those nearby.

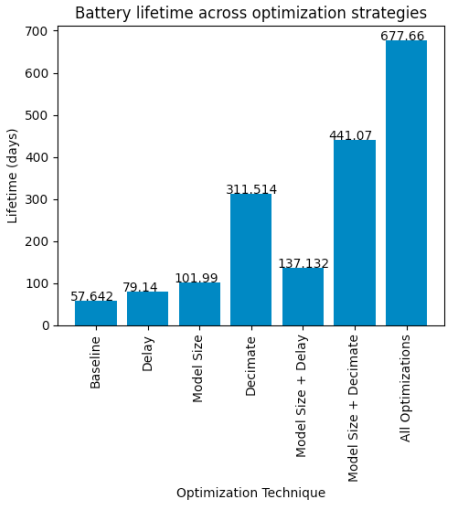


Fig. 6: Battery Life impacts of our optimizations, assuming 10,000 AADT.

- Sensors take alternating time 'shifts', such as being active every other day or hour

As shown in [22] and our results thus far, the linear relationship between the number of classifications and battery life can be further exploited by leveraging multiple sensors. These network-level optimziations all reduce the total computation time by reducing the apparent AADT at each sensor. The selection of each of these techniques is application specific, and can stretch the lifetime of each sensor, although there is an upper limit of 6.5 years due to the unavoidable power spent maintaining the device in sleep state (assuming the magnetometer can be turned off during inactive load cycles).

V. Results

For our dataset, we believe the number of measurements is small (500 measurements total, pre-augmentation) to the point that overfitting is difficult to avoid. For this reason, we qualify that our accuracy results should be taken with a grain of salt. Instead, we aim to communicate our methodology for reducing energy consumption of on-board, time-series machine learning, specifically for traffic measurement. We believe these strategies for reducing sampling rate, early interference, and tuning model size are general to other time-series tinyML applications, although the exact parameters are highly application dependent.

Taking our optimization techniques into account, we estimate the battery life of our IoT sensor using Equation 4.

Figure 6 shows the impact of our optimizations, particularly reduction in sampling rate and delayed inference, on the overall battery life. Clearly, reducing the sampling rate has the largest impact. It also has less risk of impacting accuracy than delayed inference, given knowledge of the magnetometer traces' spectral qualities, whereas some vehicles still may have distinguishing characteristics in the early portion of the signal.

Roadway Type	AADT
Highway	>15000
Arterial	10000 - 15000
Collector	5000 - 10000
Local	1000 - 5000

TABLE III: AADTs the network was evaluated over.

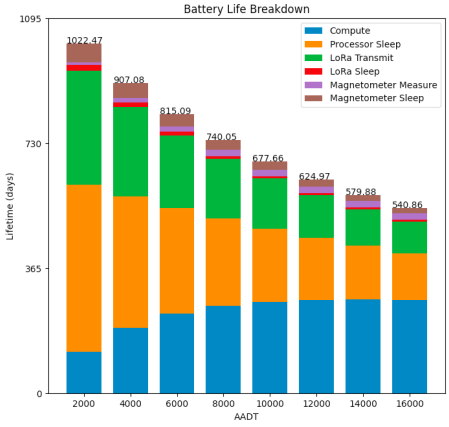


Fig. 7: Estimated lifetime over a range of AADTs for a single sensor without load-balancing. 2Ah Lithium Thionyl Chloride battery with 90% usable capacity, and the LoRA radio transmits aggregates every 15 minutes @ 1 kbps (SF10, +14dBm)

The battery life is dependent on the number of vehicles we perform inference on, the AADT, so we consider several roadway types following the Functional Classifications guidelines outlined by the U.S. Federal Highway Administration. States and local governments have slight variations of these guidelines, but, in general, roadways are classified via travel characteristics, which are distance served, access points, speed limit, distance between routes, usage (AADT/DVMT), significance, and number of lanes, as summarized in [23]. We collected several AADT ranges from CalTrans and the City of San Joe, California to aid our analysis of battery life over a feasible set of AADTs as measured in the real world. These ranges are described in Table III.

The estimated battery life resulting for our fully optimized RNN is shown across a range of feasible AADTs in Figure 7. This figure shows where energy is spent in the device, in which processor sleep state, active (compute) state, and LoRa transmission consume the bulk of the energy. For low AADT, the sleep state will dominate the battery life, and regularly scheduled LoRa transmissions will have an increased contribution throughout the entire lifetime. For high AADT, computation takes on a larger portion and the sleep state will have less impact as we spend less time sleeping, more time computing. If not specified, the reader may assume that we are modeling after an AADT of 10,000 as this is the midpoint between arterial and collector road traffic volumes per day.

Figure 8 shows the impact of network usage on the battery

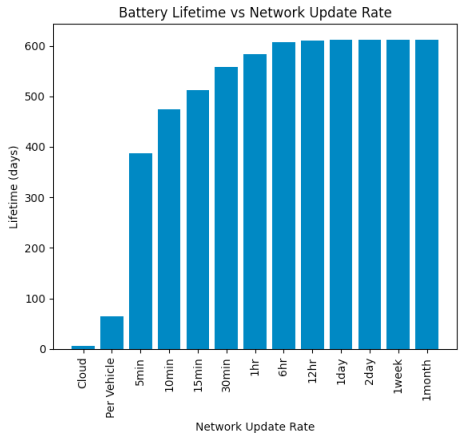


Fig. 8: Battery lifetime for a range of communication intervals

life. We assume to be using a LoRa radio transmitting at SF10 and TX power +14dBm. We show two degenerate cases, in which all samples are uploaded to the Cloud for inference, and when every vehicle detection is communicated; the battery life in these cases is quite poor compared to uplinking aggregates of all vehicles measured in a recent timeframe, such as 15 minutes or a day. This impacts the battery life noticeably until the aggregates are send on a several-hour basis, after which the energy impact of communication is small enough to be ignored.

We envision our sensor being used primarily on collector and arterial roads, in which AADT is likely within 4000 to 16000 cars per day. If sensors are deployed individually, then we would expect 1.5 - 2.8 years of lifetime per sensor. By load-balancing between multiple sensors, we expect life times greater than 3 years which is on par with the expected life time of the battery (which is subjected to intense heating cycles of the road) and the life time of the RPM tself, which is no longer than 5 years. This demonstrates the feasibility of deploying this battery-powered, traffic measurement sensor.

VI. Traffic Measurement System Design

We have designed TrafficNNode to measure traffic conditions in high spatial and temporal resolutions. Here, we discuss the implications of our model and sensor in the broader context of a traffic measurement system, as well as several essential points to consider in its implementation.

Spatial resolution is contingent on the number of sensors deployed and their distribution in space, which should redundantly cover points of primary interest like arterial and collector roads as opposed to low usage residential streets. To achieve the goal of high spatial resolution, the devices must be cheap, and simple to deploy and maintain. Just like an ordinary RPM, the packaged version should require little to no extra labor when installing, such as measuring and adjusting the physical alignment to make an internal antenna point towards the nearest base station. However, there must be advance planning to ensure high likelihood of network coverage at the installation sites by creating a map of signal strength from the gateway or base-station. Pavement mounted sensors may be difficult to communicate with given the higher likelihood of obstacles between the antenna, which must be inside the hardy enclosure, and the base-station. If building a networking infrastructure to support this application, such as LoRaWAN, one must consider the network capacity *vs.* the number of devices in the region, a common consideration that cellular infrastructure has already dealt with.

Temporal resolution is directly related to how quickly data may be retrieved over a wireless link from the deployed devices, which has an inverse relationship with the lifetime as shown in the previous section. The sooner data is made available, the faster it can be acted upon for changing traffic lights, speed limits, etc. Unfortunately, we find that using this on the timescales necessary for real-time traffic control of V2X infrastructure is infeasible if a long-term battery life is required. However, it is well-situated for multi-year installment in which data is available within the hour, which is a substantial improvement over existing traffic measurement systems.

We have focused our efforts on reducing the energy impact of the traffic classification model, given its novelty in battery-operated sensors. However, there are plenty more optimizations to be made at a systems level for each sensor using more standard embedded systems design principles. One such example is to leverage low-power peripherals to write magnetometer samples to memory using DMA and perform inference across those samples in bulk. This helps to mitigate the non-zero overhead of waking the processor from sleep state, which we have not explored in this work.

VII. Future Work

There are several logical next steps from the work presented thus far.

The performance and measurements that we have demonstrated is specific to both the dataset we have curated and the platform that we have deployed our model on. This dataset is quite till limited since it contains only SUVs and sedans (low variability) and only has 500 individual samples. Therefore, we cannot directly speak to the accuracy of our model, given the liklihood of overfitting. The dataset should be expanded to include many more samples per class, and encompass more types of vehicles and more granular speed classes. While the methodology outlined in this study is valid regardless of the dataset's composition, we have concerns that some of the parameters, such as sampling rate decimation factor, may be become less friendly to low power operation. We may require more samples to accurately discern similar vehicles. This also does not include variations in vehicles such as vehicles towing or with trailers. The second concern is with the network size. As the size of the dataset and variability of the data increases, the network needs to be expanded too.

Our investigation on the impacts of sampling rate and decimation within Section IV-D sparked our curiosity in determining which samples to use or ignore. This idea was

recently studied by Tao et al. [24], in which they trained a neural network on the input of their classifier to gate inputs to the RNN. However, using such a solution in a compute and energy constrained system such as our own may have potential energy savings by reducing the sequence length of the measurement but also introduces the overhead of the additional layer. This strategy and alternatives should be considered in the context of our system.

Due to the online nature of RNN inference, we can extract the inference probabilities at any time step by feeding the hidden state to the projection layer. This represents an opportunity to stop inference early, an idea explored by Rußwurm et al. [25], which uses an additional output in the last layer of the RNN to indicate the model's confidence that it has converged and may stop early. Clearly, this would improve the total inference time and thus, the energy per computation. However, this requires the final projection layer to be run at every timestamp, which increases the total compute time per inference and introduces additional overhead. It is worth investigating if this would increase or decrease the overall battery life of our system.

VIII. Conclusion

In this study, we have designed a battery-powered, pavement-mounted sensor that classifies vehicle type and a speed range using three-axis magnetometer readings with a recurrent neural network (RNN). We deploy this RNN to our microcontroller -based system, which must last several years without recharging or maintenance. To achieve this lifetime, we built a methodology to apply standard and application-specific optimizations to reduce the overall energy cost vehicle detection and classification. For this application, the energy cost per inference can be reduced by decreasing the sampling rate and delaying inference beyond the initial detection. These optimizations improve battery lifetime by 11.8x over the original baseline model to achieve a total lifetime of up to 2.8 years. We have demonstrated that is feasible to deploy an inexpensive, battery-powered, intelligent sensor directly to the roadways.

References

[1] H. Chourabi, T. Nam, S. Walker, J. R. Gil-Garcia, S. Mellouli, K. Nahon, T. Pardo, and H. Scholl, "Understanding smart cities: An integrative framework," *45th Hawaii International Conference on System Sciences*, pp. 2289–2297, 01 2012.

[2] K. Nellore and G. P. Hancke, "A survey on urban traffic management system using wireless sensor networks," *Sensors*, vol. 16, no. 2, 2016.

[3] S. Djahel, R. Doolan, G.-M. Muntean, and J. Murphy, "A communications-oriented perspective on traffic management systems for smart cities: Challenges and innovative approaches," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 125–151, 2015.

[4] B. Iannucci and A. Rowe, "Crowdsourced Smart Cities," in *Intelligent Transportation Systems (ITS) World Congress*, (Montreal, Quebec, Canada), 2017.

[5] S. Oh, S. G. Ritchie, and C. Oh, "Real-time traffic measurement from single loop inductive signatures," *Transportation Research Record*, vol. 1804, no. 1, pp. 98–106, 2002.

[6] M. H. Kang, B. W. Choi, K. Koh, J. Lee, and G. T. Park, "Experimental study of a vehicle detector with an amr sensor," *Sensors and Actuators A-physical*, vol. 118, pp. 278–284, 2005.

[7] Z. Zhang, T. Zhao, and H. Yuan, "A vehicle speed estimation algorithm based on wireless amr sensors," in *Big Data Computing and Communications* (Y. Wang, H. Xiong, S. Argamon, X. Li, and J. Li, eds.), (Cham), pp. 167–176, Springer International Publishing, 2015.

[8] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014.

[9] S. Djahel, R. Doolan, G.-M. Muntean, and J. Murphy, "A communications-oriented perspective on traffic management systems for smart cities: Challenges and innovative approaches," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 125–151, 2015.

[10] M. Pla-Castells, J. J. Martinez-Durá, J. J. Samper-Zapater, and R. V. Cirilo-Gimeno, "Use of ict in smart cities. a practical case applied to traffic management in the city of valencia," in *2015 Smart Cities Symposium Prague (SCSP)*, pp. 1–4, 2015.

[11] Q.-J. Kong, Q. Zhao, C. Wei, and Y. Liu, "Efficient traffic state estimation for large-scale urban road networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 398–407, 2013.

[12] J. C. Herrera, D. B. Work, R. Herring, X. J. Ban, Q. Jacobson, and A. M. Bayen, "Evaluation of traffic data obtained via gps-enabled mobile phones: The mobile century field experiment," *Transportation Research Part C: Emerging Technologies*, vol. 18, no. 4, pp. 568–583, 2010.

[13] Z. Ning, J. Huang, and X. Wang, "Vehicular fog computing: Enabling real-time traffic management for smart cities," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 87–93, 2019.

[14] A. Allström, J. Barceló, J. Ekström, E. Grumert, D. Gundlegård, and C. Rydergren, *Traffic Management for Smart Cities*, pp. 211–240. Cham: Springer International Publishing, 2017.

[15] C. Esposito, A. Castiglione, F. Pop, and K.-K. R. Choo, "Challenges of connecting edge and cloud computing: A security and forensic perspective," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 13–17, 2017.

[16] M. Mohammadi and A. Al-Fuqaha, "Enabling cognitive smart cities using big data and machine learning: Approaches and challenges," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 94–101, 2018.

[17] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[18] R. David, J. Duke, A. Jain, V. J. Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, S. Regev, R. Rhodes, T. Wang, and P. Warden, "Tensorflow lite micro: Embedded machine learning on tinyml systems," *CoRR*, vol. abs/2010.08678, 2020.

[19] N. Tan, "utensor-ai inference library based on mbed and tensorflow." online. https://github.com/uTensor/uTensor.

[20] E. Liberis, u. Dudziak, and N. D. Lane, "nas: Constrained neural architecture search for microcontrollers," in *Proceedings of the 1st Workshop on Machine Learning and Systems*, EuroMLSys '21, (New York, NY, USA), p. 70–79, Association for Computing Machinery, 2021.

[21] L. Heim, A. Biri, Z. Qu, and L. Thiele, "Measuring what really matters: Optimizing neural networks for tinyml," *CoRR*, vol. abs/2104.10645, 2021.

[22] C. R. Banbury, C. Zhou, I. Fedorov, R. M. Navarro, U. Thakker, D. Gope, V. J. Reddi, M. Mattina, and P. N. Whatmough, "Micronets: Neural network architectures for deploying tinyml applications on commodity microcontrollers," *CoRR*, vol. abs/2010.11267, 2020.

[23] "Highway functional classification concepts, criteria and procedures: 2013 edition," Tech. Rep. FWHA-PL-13-026, U.S. Department of Transportation Federal Highway Administration, Salt Lake City, UT, 2013.

[24] J. Tao, U. Thakker, G. Dasika, and J. Beu, "Skipping rnn state updates without retraining the original model," in *Proceedings of the 1st Workshop on Machine Learning on Edge in Sensor Systems*, pp. 31–36, 2019.

[25] M. Rußwurm, S. Lefèvre, N. Courty, R. Emonet, M. Körner, and R. Tavenard, "End-to-end learning for early classification of time series," *arXiv preprint arXiv:1901.10681*, 2019.