FrugalLight : Symmetry-Aware Cyclic Heterogeneous Intersection Control using Deep Reinforcement Learning with Model Compression, Distillation and Domain Knowledge

SACHIN KUMAR CHAUHAN, CSE, Indian Institute of Technology Delhi, India
RIJUREKHA SEN, CSE, Indian Institute of Technology Delhi, India

Developing countries need to better manage fast increasing traffic flows, owing to rapid urbanization. Else, increasing traffic congestion would increase fatalities due to reckless driving, as well as keep vehicular emissions and air pollution critically high in cities like New Delhi. State-of-the-art traffic signal control methods in developed countries, however, use expensive sensing, computation and communication resources. How far can control algorithms go, under resource constraints, is explored through the design and evaluation of FrugalLight (FL) in this paper. We also captured and processed a real traffic dataset at a busy intersection in New Delhi, India, using efficient techniques on low cost embedded devices. This dataset (*https://delhi-trafficdensity-dataset.github.io*) contains traffic density information at fine time granularity of one measurement every second, from all approaches of the intersection for 40 days. FrugalLight (*https://github.com/sachin-iitd/FrugalLight*) is evaluated on the collected traffic dataset from New Delhi and another open source traffic dataset from New York. FrugalLight matches the performance of state-of-the-art Convolutional Neural Network (CNN) based sensing and Deep Reinforcement Learning (DRL) based control algorithms, while utilizing resources less by an order of magnitude. We further explore improvements using a careful combination of knowledge distillation and domain knowledge based DRL model compression, with employing Model-Agnostic Meta-Learning to quickly adapt to traffic at new intersections. The collected real dataset and FrugalLight therefore opens up opportunities for resource efficient RL based intersection control design for the ML research community, where the controller should have limited carbon footprint. Such intelligent, green, intersection controllers can help reduce traffic congestion and associated vehicular emissions, even if compute and communication infrastructure is limited in low resource regions. This is a critical step towards achieving two of the United Nations Sustainable Development Goals (SDG), namely sustainable cities and communities and climate action.

CCS Concepts: • **Computing methodologies** → **Reinforcement learning**; *Q-learning*; *Neural networks*; • **Social and professional topics** → *Sustainability*.

Additional Key Words and Phrases: Traffic Signal Control, Real Traffic Dataset

ACM Reference Format:
Sachin Kumar Chauhan and Rijurekha Sen. 2024. FrugalLight : Symmetry-Aware Cyclic Heterogeneous Intersection Control using Deep Reinforcement Learning with Model Compression, Distillation and Domain Knowledge. In . ACM, New York, NY, USA, 33 pages. https://doi.org/XXXXXXX.XXXXXXX

Manuscript submitted to ACM

1 INTRODUCTION

Traffic signal control, a historical application of automated planning and scheduling, is seeing a dramatic shift with recent advances in Internet of Things (IoT), Computer Vision (CV), and Reinforcement Learning (RL). As shown in [47, 52], researchers, in collaboration with city authorities, are collecting traffic videos at scale using IoT devices, processing them using the latest Convolutional Neural Networks (CNN) based computer vision methods, and using the processed data for traffic light control using Deep Reinforcement Learning (DRL).

The problem of traffic congestion is acute in developing countries like India, further contributing to the high air pollution in cities like New Delhi [10]. Therefore the state-of-the-art intersection control algorithms should also be used to benefit these low resource communities. There is, however, a key challenges involved in transferring technology as it is: the current intersection control algorithms [45, 46] use exact vehicle counts, and work perfectly in developed countries with orderly laned traffic. Traffic in developing countries is non-lane based (sample images in Figure4). Thus exact vehicle counts computed using state-of-the-art CNNs like YOLO [36], even if work fine at daytime [8], start dropping in accuracy at night. But intersection control is needed at all times, independent of the lighting condition. Hence intersection control algorithms should use information that can be computed at all times. There is, therefore, a clear gap in data and algorithms for automated intersection control in developing regions. This paper seeks to bridge this gap, with the following three contributions:

❶ We release the first traffic density dataset (*https://delhi-trafficdensity-dataset.github.io*) from a developing country intersection (attributed by non-lane behaviour, connectivity/compute limitations), under a Creative Commons Attribution 4.0 International License [7]. We deployed 6 cameras at a busy 3-approach intersection in New Delhi. The location and camera placements is indicated in Figure 1. Due to lack of broadband connectivity from the road to the cloud, the video feeds could not be transmitted. This lack of cloud connectivity will hold in any actual intersection control deployment for a developing country. So we performed *in-situ* processing of the traffic videos, using computer vision algorithms like background subtraction [6], that also work well at night, unlike CNNs. The released dataset contains traffic density information at fine time granularity of one measurement every second, from all approaches of the intersection for 40 days between Sep-Dec 2020 (§ 3).

❷ We carefully design a reinforcement learning based intersection control algorithm FrugalLight (FL) in this paper, that needs traffic density information as input and not exact vehicle count. FL (*https://github.com/sachin-iitd/FrugalLight*) is shown to improve transportation metrics like throughput and travel time over traditional and state-of-the-art policies, while showing better transferability and fast adaptability. More importantly, FL and all computer vision inputs to FL, are computationally so efficient, that we can run them on low cost embedded platforms at the intersection in real time, without cloud connectivity (§ 5).

❸ We compare FL with state-of-the-art intersection control algorithms [45, 46] on this new Delhi dataset, as well as an open source traffic dataset from New York, the latter containing exact vehicle count information. Surprisingly, FL matches the performance of the baseline RL algorithms, while utilizing computation and communication resources less by an order of magnitude. This shows that algorithms designed for constrained datasets (only traffic density) as released in this paper, can work well even in resourceful countries with unconstrained data (exact vehicle counts). Thus if ML researchers start considering efficiency metrics like model size and inference latency, in addition to accuracy metrics, ML algorithms for critical applications like intersection control can have less carbon footprint than the state-of-the-art [45, 46] (§ 6).

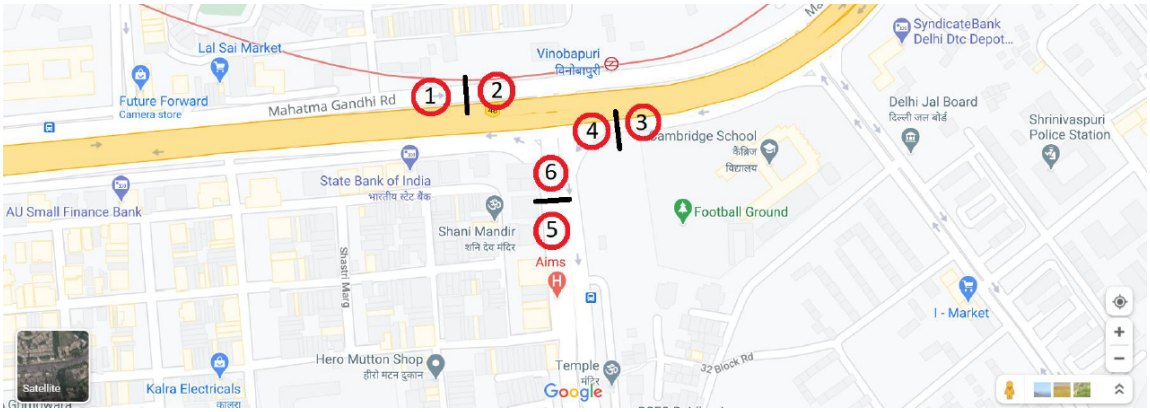

Fig. 1. Google Maps Location of the intersection and installed cameras in New Delhi. Approach 1 has cameras 1 and 2, approach 2 has cameras 3 and 4, approach 3 has cameras 5 and 6.

❹ We utilize Knowledge Distillation based guidance approaches [18, 27] to improve FrugalLight , and use MAML based Meta-Learning approaches [17, 51] to scale them, paving a way for efficient and effective learning of the traffic situations (§ 7).

❺ The challenges of directly importing the FrugalLight model are three fold – (i) extreme budget constraints, which allows for only very low cost, compute and RAM constrained, embedded platforms to be deployed (ii) poor network connectivity between the road and the servers, forcing all analysis to happen in-situ on the road and (iii) chaotic non-laned driving behavior in developing regions, which makes accurate video analysis for exact counting and classification of vehicles harder.

We hence extend FrugalLight for learning efficient Look-Up Table (LUT) based or threshold based intersection control. This solution, termed EcoLight [9], performs at par with the compute intensive methods, at a mere fraction of runtime overhead. Optimizing computational overhead while not losing accuracy has been challenging for EcoLight. We reduce DRL states from over a thousand dimensions in state-of-the-art papers [45, 46] to one or two dimensions. We remove the DNN based RL computation at runtime using static LUTs. We quantize the original continuous values of DRL states for finite sized LUTs. All these optimizations needed to be carefully tuned for accuracy. We experiment with both open-source developed country dataset and a custom developing region dataset, created by us from our deployed cameras. As a result of careful tuning, EcoLight gives comparable benefits and sometimes even improves upon the compute-intensive methods, on both performance metrics (throughput, average travel time etc.) and fairness metrics (worst case travel time, vehicles stuck etc.) (§ 7).

❻ Finally, an end-to-end system has also been demonstrated in this paper. This incorporates video feeds from cameras at a real intersection and computer vision based traffic density estimation for input to the control algorithms. Our results show great promise towards practical adaptive intersection control at extreme budget and network constraints, a vital necessity for sustainability (§ 8).

2 RELATED WORK

Researchers have identified and modelled traffic congestion in developing countries using different methods [5, 23, 24]. To mitigate the problem of traffic congestion, in most developing countries today, traditional traffic signal control methods are still being used. These are static systems that work by changing the phase periodically with a fixed cycle length. Research has suggested that such traffic lights even become the cause of traffic jams in some cases as shown in [4]. In order to deal with the issue of dynamic-updation, researchers have modeled the problem of traffic control as an optimization problem based on certain assumptions and come up with rules for setting the phase based on the traffic densities in the network as shown in [28, 29, 38, 42]. Even with this approach, the rules obtained are pre-defined and cannot be dynamically adjusted for real-time traffic. A variety of Reinforcement Learning works, like [2, 11, 14, 15, 32, 35, 41], have proposed different state and reward formulations for different action choices.

Recent Reinforcement-Learning based methods [45, 46] outperform the traditional traffic signal control methods in simulation environments. These methods, however, require significant computational and communication resources as each agent uses a Deep Neural Network to compute the current Phase and communicates to a central server in real-time, making them infeasible for deployment in developing countries. A recent work EcoLight [9] uses very small dimensional states, with fairness optimizations, to generate lookup tables to be deployed in lieu of the DRL models. The lookup tables being static, cannot fully leverage the dynamic traffic conditions.

Our method FrugalLight is an extension of the above work and can adapt to changing traffic conditions and can be cost-effectively deployed in developing countries.

High quality real traffic data is necessary to develop ML/RL based control algorithms for efficient intersection control. Researchers have been generating real traffic data through various means. [20] used loop sensors based dataset creation and [39] used loop sensors to calculate traffic queue length, but as per [16, 34], though cheap and widely used, there are several operational constraints with loop sensors, like they degrade with the conditions of the road and water penetration affects the performance. They are prone to damage due the poor state of our road surfaces, and through weaknesses caused by the installation of loops or other damage over time such as potholes. Loop tails are also often cut in the course of other road works such as utility companies accessing their infrastructure. These findings were used by different city authorities for improvements in transport infrastructure, As per [13], Swansea City Council opted for an alternate solution considering such factors. We found similar case for London [25, 48].

Besides, [45, 46] used taxi data to approximate real traffic data in New York. [37] recorded the traffic in Doha, Qatar in the peak hour of weekday and reproduced in the SUMO [26] simulator. [44] used open source camera data from Hefei (China) to analyze and use a peak hour flow for the experiments. [2] used the 24 hour primary vehicle Origin-Destination data collected from the municipality of Tehran and adjusted it by one-hour interval traffic count data obtained from traffic sensors and gathered (impatient) pedestrians data via fieldwork. [3], used the six hourly origin-destination matrices calculated by the municipality of Tehran for the traffic demands between 6am and 12pm on a workday.

Continuing the effort of creating datasets for traffic intersection control, we alongside provide multiple days traffic density data that can be used to train ML models directly or can be used to generate smaller datasets to be used in various simulators like cityflow [52] and SUMO [26].

3 REAL DATA DESCRIPTION

3.1 Data Collection Challenges

Installing traffic cameras at a busy intersection in New Delhi, involved a series of challenges. Proper infrastructure to mount the cameras and draw connections for power and communication, was needed. All work had to be done by minimally affecting the flow of traffic. Permissions were needed from the traffic authorities. We collaborated with an industry partner[1] for the deployment. It was also not feasible to send raw video from cameras to cloud server, hence edge computing was required. Camera video was processed at the intersection itself, to generate traffic density numbers, and the density values were stored locally for periodic retrieval.

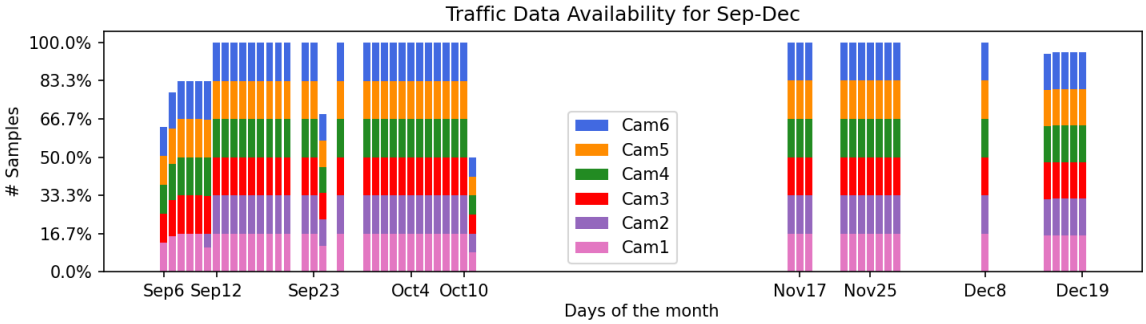

Fig. 2. Traffic Data Availability for the six cameras from Sep-Dec 2020, each colour denoting the data coming from separate camera.

During deployment, we observed multiple issues hampering the sound operation of the deployed system: ❶ One camera power adapter failure (1 camera down), ❷ Power failure for one approach (2 cameras down), ❸ Communication line failure from one approach (2 cameras down), ❹ Multiple times Local Processing Unit hang or power-off (all cameras down). In issues related to the camera device, a crane was required for the repair. For issues at ground level, efforts had to be made to trace down the point of failure, and then replacement of the faulty component. All of these require days to weeks to get done, due to many dependencies involved in the maintenance process. We finally had 40 days of complete data in a 4-month duration (Sep-Dec 2020), as shown in Figure 2. Due to winter, heavy fog and late sunrise, during Dec 15-19, we started delayed data collection as compared to previous days, hence the plot shows a bit smaller lines for these 5 days.

3.2 Dataset Processing

Before discussing the processing of the traffic density data, we first describe our density estimation method based on background subtraction. As presented in Figure 3, a continuously updated Background Filter (with learning rate τ) is subtracted from each frame to get the foreground, and the ratio *foreground/background* denotes Queue Density. To find Stop Density, we need to discard the density caused by the moving/dynamic traffic. Using the optical flow algorithm to detect moving pixels, we computed the standing traffic (Stop Density). The inbuilt adaptation in Background Subtraction makes the processing robust to the changing ambient light conditions.

The density estimation code [2] deployed on the road to generate queue and stop densities contains the methods to receive the video frames from the camera and process it using background subtraction to generate Queue Density and Stop Density values for the each camera (with cuda optimized version). There are background images for each camera

[1] Aabmatica Technologies http://aabmatica.com/new/
[2] https://github.com/sachin-iitd/TrafficDensity

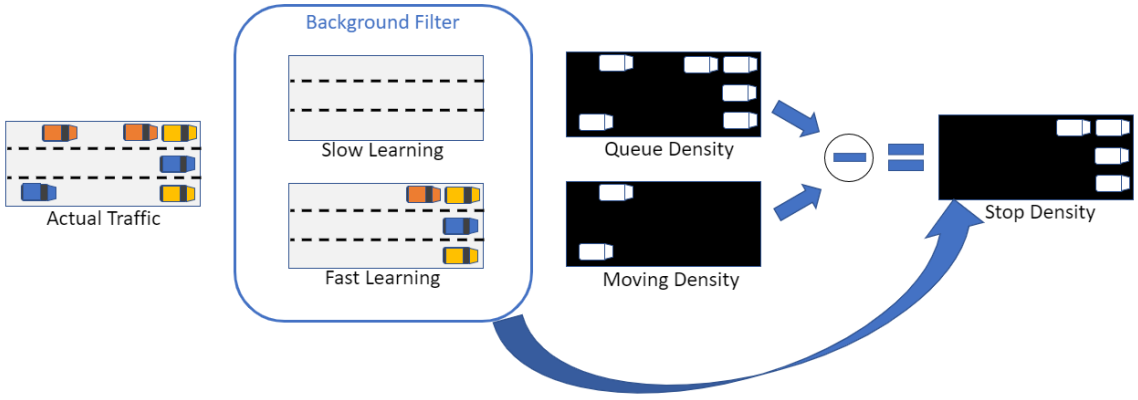

Fig. 3. Background Subtraction for Queue and Stop densities.

used to initiate the background subtraction algorithm in the morning and projection coordinates to transform 3D view[3] of the camera to 2D vertical view [4].

3.3 Dataset Quality

To work with intersection control algorithms, camera frames are processed to get representative traffic summary. CNN based Computer Vision methods like YOLO [36] are used to identify the vehicles present on the road, and enumerate the identified vehicles to get vehicle count. Figure 4 shows two good examples of YOLO traffic identification on the Left, and two fairly miserable detections due to occlusion in heavy traffic in the Middle. Poor lighting conditions also drastically affect YOLO accuracy.

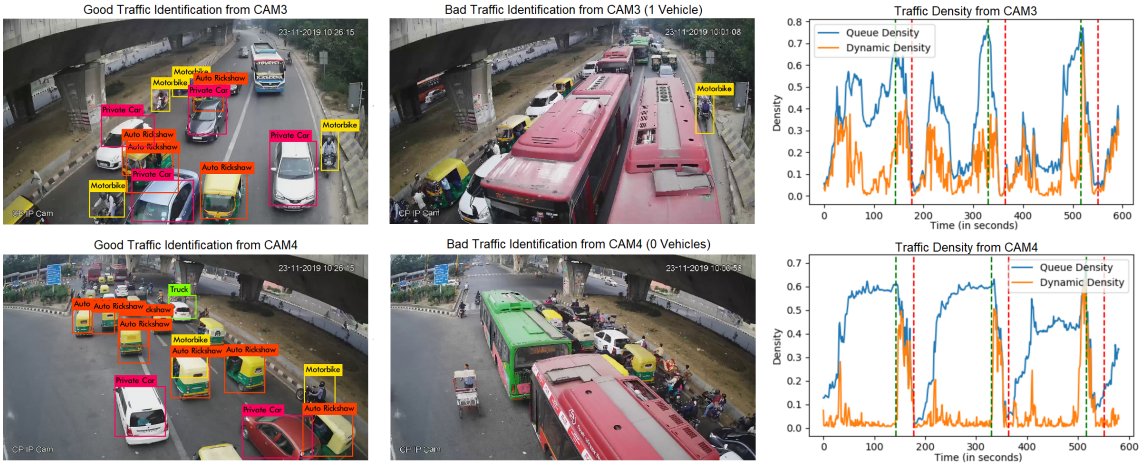

Fig. 4. Good Yolo labeling (Left), Bad Yolo labeling (Middle) and Good traffic density (queue density and dynamic density) (Right). Green/red vertical lines in traffic density indicates the event of green/red signal.

[3] *https://github.com/sachin-iitd/TrafficDensity/blob/main/traffic/6.jpg*
[4] *https://github.com/sachin-iitd/TrafficDensity/blob/main/traffic/proj_traffic_6.jpg*

CNN accuracy vs. latency: We use the labeled dataset from [8] to explicitly check the accuracy vs. latency trade-off of different CNN models. We split the dataset into train and test data as shown in Table 1, and train YOLO V2 and Tiny-YOLO models.

Table 1. Train and test data for YOLO V2 and Tiny-YOLO

Test images	Train images	Test annotations	Train annotations
1670	3896	9963	22657

Table 2. Accuracy-vs-latency trade-off of CNN models

Method	Size	mAP	Precision	Recall	CPU FPS	GPU FPS
YOLO V2	194 MB	54.63	0.38	0.62	0.1	6-7
Tiny-YOLO	34 MB	34.99	0.63	0.49	0.4-0.5	20-21

On test data, the average accuracies are reported in terms of mAP, precision and recall in Table 2. As seen from the values, Tiny-YOLO reports low mAP and low recall compared to YOLO V2. In terms of latency, as measured on NVIDIA Jetson TX2, CPU frame rates are really low for both models, whereas GPU rates are low for YOLO V2 and moderate for Tiny-YOLO. We observe that the Traffic Density calculation using background subtraction [6] works pretty well, in both natural and street lighting. Figure 4(Right) shows two example density computations, in light to moderate traffic (Top) and in heavy traffic (Bottom). The Top graph corresponds to Left of YOLO detections, and the Bottom graph corresponds to Middle of YOLO detections. Queue density grows correctly between red and green signal vertical lines, signifying the red phase for the approach. In contrast, queue density drops between green and the next red signal, signifying the green phase for the approach. Dynamic/moving density remains close to 0 in heavy traffic (as seen in below bottom plot) during red cycle and only rises in green cycle. As per our observations for other dataset collection using loop-based or camera data, intermittent validation is a feasible way forward. We have observed high density (~1) when the frame was full with vehicles, and very low (almost 0) in case of no traffic or when the vehicles pass completely across the intersection. In between also, we have observed density values in the similar ratio as traffic is present on the road. So, over many manually verified parts of the dataset, we repeatedly observed this perfect density calculation vs. high YOLO errors. This is understandable as density estimation is a easier task than detecting vehicle bounding boxes. The traffic density processing code is available at *https://github.com/sachin-iitd/TrafficDensity*.

In addition to being accurate, our density estimation code runs at 6 FPS on low cost embedded platform (1.8 GHz Intel(R) Atom(TM) CPU D525 with 4 logical cores and 8GB RAM) budgeted by our deployment partners, and gives us Queue Density and Stop Density values per second for the 6 cameras. The dataset, thus also has fine granularity of recorded traffic density measurements.

Histogram of Queue and Stop densities: We analyzed the histogram of the queue and stop densities of our dataset. Both the densities are available in the scale of 0 - 1, where 0 means empty road with no traffic and 1 means road with full traffic. Figure 5 shows the histogram of Queue and Stop densities for the 6 cameras. By manual observations of the traffic images, we have seen that approach 3 usually had limited traffic stopped for the red light, the same was found evident in the StopDensity5 histogram which explicitly shows high occurrence for almost none waiting traffic beyond the camera 6 scope.

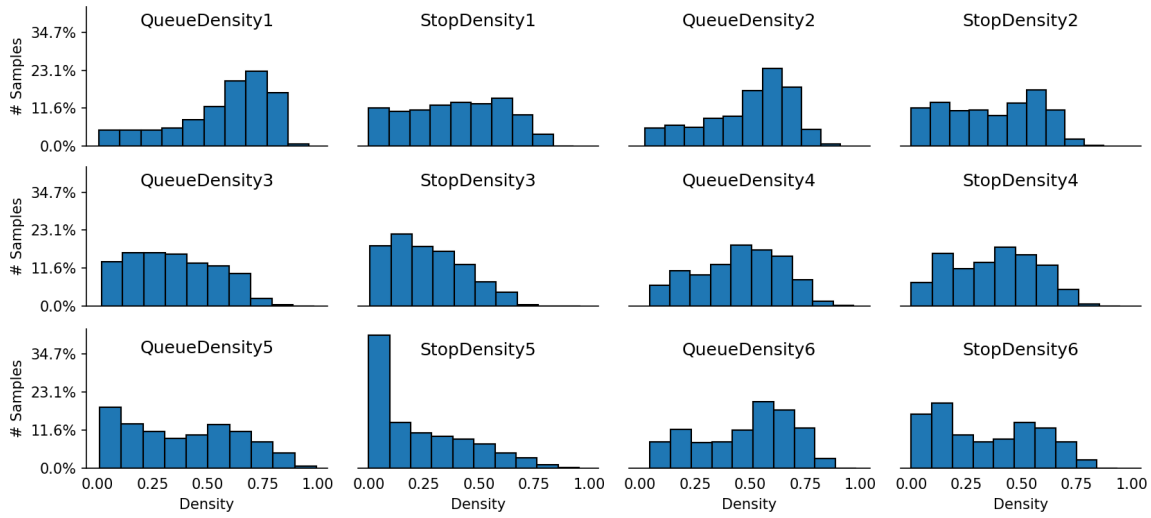

Fig. 5. Histogram of Queue and Stop densities for the 6 cameras for the 40 days for total 2,160,040 samples.

3.4 Dataset Uniqueness

Our dataset is a longitudinal dataset which is collected over many days from all approaches at an intersection, which is needed specifically for traffic intersection control algorithms. Using the background subtraction and optical flow techniques, our dataset contains traffic density and stop density for each approach per second. We further convert the density values from to traffic dataset suitable for simulator evaluations with methods similar to other works.

Table 3. Open source real traffic datasets.

Inter'n Layout	Num App	Vehs/ 5min	Hr	Location	Process Method
16x1=16	4	569	1	8th Ave, NY	Taxi data
16x3=48		235	1	8-10 Ave, NY	
1x1=1	3	614	1	New Delhi	Cam+YOLO
		346	1		Cam+ BackSub
		254	2		
		185	4		
		113	6		
3x4=12	4	525	1	Jinan, Hongqi	Cam
4x4=16		249	1	Hangzhou, Gudang	

Table 3 enlists the properties for various open source datasets. *Inter'n Layout*, denotes how many roads cross to create how many intersections, describing the road network architecture. E.g. *16x1=16* indicates there is one road perpendicular to 16 roads, crossing each of them creating 16 intersections. *Num App* denotes the (fixed) number of approaches at intersections. The next three columns indicate the traffic volume (in vehicles arriving per 5 minutes),

duration of datasets (from 1 to 6 hours), and the geographical location (from USA/India/China) from which the datasets are collected. The last column points the source of the data (Taxi trip information, Camera) and the processing method (YOLO: [36], Background Subtraction: [6]) The single intersection data from New Delhi corresponding to Cam+BackSub process method is generated from 1-6 hours portions of the shared 40 days traffic density dataset.

The uniqueness of this dataset lies in ❶ limited features (density, not vehicle count), which we show is a practical information to obtain in real time in a developing country, and ❷ longitudinal nature. The analysis of multiple days data from our New Delhi dataset is shown in Figure 6 [5], the horizontal axis denote the hour of the day, and vertical axis denote the Queue Density. The box-plot shows the peak traffic during morning and evening hours for the Approaches 1 and 2. The approach 3, which joins the other approaches to form a T junction, has an independent pattern where the traffic increases as the day progresses. The variation in density at each hour over the 40 days, show how dynamic traffic is at this Delhi intersection, and how ML researchers can use this to benchmark their RL based intersection control algorithms.

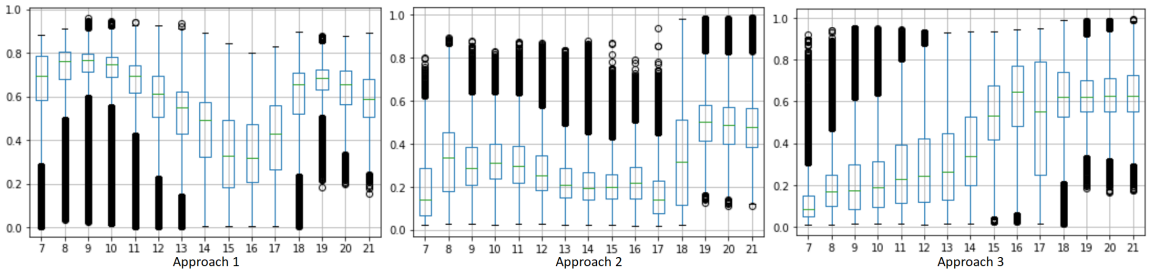

Fig. 6. Traffic Patterns observed at an intersection in developing country.

4 NEED FOR INTELLIGENT TRAFFIC LIGHT CONTROL

Traffic in both developed and developing countries is very dynamic, which is very hard to be approximated, predicted or calculated with simple equations and formulas. It is also heavily dependent on location, time, and many other local constraints. Utilization of the features of AI and DRL is necessary to approximate the traffic behaviour in a better way. Such time varying and complex traffic information needs suitable methods, to be processed and transformed into an effective policy to help control the traffic more effectively. Reinforcement Learning (RL) methods have shown great promise to learn effective control policies from such situations. Traditional RL methods used in the research either could not capture the varying traffic (limiting experiments to hourly traffic), or miss out to process the data in a structured way (by simply trying to fit to the raw data). We try to overcome this problem by presenting a structured and formalized way to learn from the real data effectively.

Existing methods work with vehicle count (and their lanes and distance from the intersection), which required expensive *in-situ* processing to convert camera videos to vehicle count, with models such as YOLO [36]. As we could efficiently process the real-time traffic data in terms of density, the new RL method should be able to effectively work using limited information such as only traffic density as the input parameters.

[5]This traffic density dataset has already been used in another paper on thermal controller design for embedded GPU platforms and the graph is reproduced from that paper [40]. But the dataset is being made public for the first time, with a more important use-case analysis for the ML research community, namely RL based intersection controller design.

The state-of-the-art models are big in size which makes them unsuitable for deployment over low cost edge devices. Our new model should be having small size, making it fast in processing and hence easily deployable on edge platform. As no existing RL algorithms [45, 46] meets these requirements, we formulate the traffic control problem as a Markov Decision Process (MDP) and design FrugalLight .

4.1 Problem Definition

To start-with, we define the problem of traffic signal control as a Markov Process. Each intersection in the system is controlled by an agent running independently, and without any communication with the others. In this setting, each agent observes part of the total system, and decides for its own intersection whether to keep the same phase or switch to the next, so as to minimize the average traffic density on the approaches around the intersection. Specifically, the problem can be characterized by the following major components $< \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{P}, r, \pi, \gamma >$ as described in detail below.

❶ With system state space $\mathcal{S}$ and observation space $\mathcal{O}$, we assume that there are N intersections in the system and each agent can observe part of the system state $s \in \mathcal{S}$ as its observation $o \in \mathcal{O}$. We define o_i^t for agent i at time t, which consists of traffic density in one or two dimensions as described later.

❷ With set of actions $\mathcal{A}$, at time t, an agent i would choose an action a_i^t from its candidate action set $\mathcal{A}_i$ as a decision for the next Δt period of time. Here, each agent would choose either 0 or 1 as its action a_i^t, indicating that from time t to $t + \Delta t$, this intersection would be in same phase or under transition to the next phase.

❸ With transition probability $\mathcal{P}$, given the system state s_i^t and actions a_i^t of agent i at time t, the system arrives at the next state s_i^{t+1} according to the state transition probability $P(s_i^{t+1}|s_i^t, a_i^t)$.

❹ With reward r, each agent i obtains an immediate reward r_i^t from the environment at time t. In this paper, we want to minimize the travel time for all vehicles in the system, which is hard to optimize directly. Therefore, we define the reward for intersection i as $r_i^t = -\sum_a d_{i,a}^t$ where $d_{i,a}^t$ is the stop density on the approach a of intersection i at time t.

❺ With Policy π and discount factor γ, as the independent actions have long-term effects on the system, we want to minimize the expected stop density of each intersection in each episode. Specifically, at time t, each agent chooses an action following a certain policy $\mathcal{O}$ x $\mathcal{A} \rightarrow \pi$, aiming to maximize its total reward $G_i^t = \sum_{t=\tau}^{T} \gamma^{t-\tau} r_i^t$, where T is total time steps of an episode and $\gamma \in [0, 1]$ differentiates the rewards in terms of temporal proximity.

In this paper, we use the action-value function $Q_i(\theta_n)$ for each agent i at the n^{th} iteration (parameterized by θ) to approximate total reward G_i^t with neural networks by minimizing the loss:

$$\mathcal{L}(\theta_n) = E[(r_i^t + \gamma \max_{a'} Q(o_i^{t\prime}, a_i^{t\prime}; \theta_{n-1}) - Q(o_i^t, a_i^t; \theta_n))^2] \tag{1}$$

where $o_i^{t\prime}$ denotes the next observation for o_i^t. These earlier snapshots of parameters are periodically updated with the most recent network weights and help increase the learning stability by de-correlating predicted and target q-values.

5 FRUGALLIGHT

Based on a recent and comprehensive survey of intelligent traffic light control methods [47], we choose two [45, 46] most promising state-of-the-art DRL based traffic light control algorithms to be our baselines. FrugalLight uses domain knowledge and careful optimizations to match the performance of these complex baselines, at a tiny fraction of computational resources, utilizing practical sensor inputs for developing region traffic. Our work, in principle, follows the current research trend on efficient machine learning, mostly for optimizing CNN models for computer vision tasks [19, 21, 30, 49, 50]. We extend the efficiency question to a new application domain of traffic light control and optimize a

different machine learning model, namely DRL. Our innovations come from practical constraints (not addressed in prior work) that developing regions pose on the intended application. We also utilize Knowledge Distillation based guidance approaches [18, 27] to improve our methods, and use MAML based MetaLearning approaches [17, 51] to scale them, paving a way for efficient and effective learning of the traffic situations.

5.1 Design Prerequisites

There are several environment level design considerations before discussing the DRL based methodology.

Control agents – coordinated vs decentralized: While absence of continuous network connectivity to the cloud necessitates *in situ* computations for the computer vision and traffic light control algorithms, the same connectivity issue also necessitates the design of independent traffic light control agents. Real time communication across agents of different intersections cannot be taken for granted. So we design individual agents for each intersection in this paper, without assuming mutual communication.

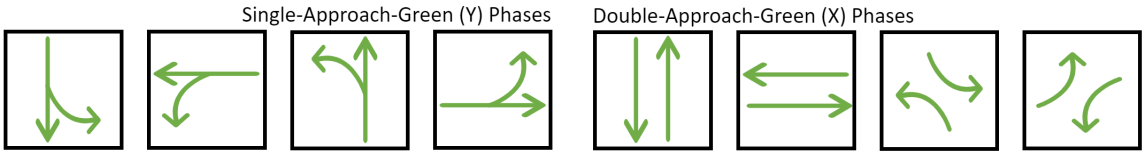

Fig. 7. Allowed phases at intersections

Phase characteristics at intersections: As traffic in developing regions is non-laned and chaotic, giving green simultaneously to different approaches increases chance of collisions at an intersection. Single-approach-green (Y) pattern is, therefore, typically used in intersection design in developing countries. This scheme comprises phase sequences as shown in Figure 7, where in each phase, the straight flow and the turning flow are given green simultaneously. The number of such phases depend on the number of approaches at a particular intersection. We analyze our methods primarily for this Y-scheme, and later show that our method works better for the other phase schemes as well, such as Double-approach-green (X) pattern and a mix of both (XY).

Expected output of control algorithm: Our agents can take one of the two kinds of decisions, at every decision making time point. The decision making time point comes at fixed periodicity for the agent. ❶ *Switch to the next phase*: In this setting, the scheduler delivers a binary decision either to continue current green signal or to switch to the next phase in a cyclic order. ❷ *Set any phase*: In this setting, the scheduler switches to the best phase, which can be any of the allowed phases. *Set any phase* is more flexible and can potentially give better values for the traffic metrics being optimized (travel time or throughput). But the fixed phase cycle in *switch to the next phase* is better to set commuter expectations as to who will get the next green. In developing countries where traffic is already extremely chaotic and drivers are unruly, phase cycle is kept constant to set predictable expectations to drivers. Thus our control agents should follow *switch to the next phase* decision scheme. We evaluate both and show that the additional flexibility of *set any phase* gives minor improvements in metric values, over the more practical and safer *switch to the next phase* scheme.

Optimization metrics: The primary metric usually used to quantify the performance of a traffic light control system, is average *travel time* of vehicles passing through that intersection. We use this metric in our evaluations. The second metric evaluated is *throughput*, which is the percentage of vehicles cleared by the intersection. A third metric is *total time*, which combines the time spent by the vehicles which clear the intersection and also those stuck at the intersection. Throughput needs to be maximized while travel time and total time needs to be minimized.

5.2 FrugalLight DRL Architecture

We use a DQN based DRL architecture with fully connected layers, comprising two hidden layers of size H each ($5 \leq H \leq 20$). Suppose we use M states to represent an intersection. Further suppose N phases, so our DRL can choose to stay in the current phase, or choose among the remaining $N-1$ phases, giving N possible actions. Then our DRL has an M x H x H x N architecture, as shown in Figure 8. Our method is independent of underlying loss function and optimizer choice, and we use *MeanSquareError* [43] and *RMSprop* [22] respectively in our experiments.The simple architecture also allows us to explore and emphasize the benefits of using other enhancements in State and Reward design. Such enhancements would give further improvements when a complex DNN architecture is utilized.

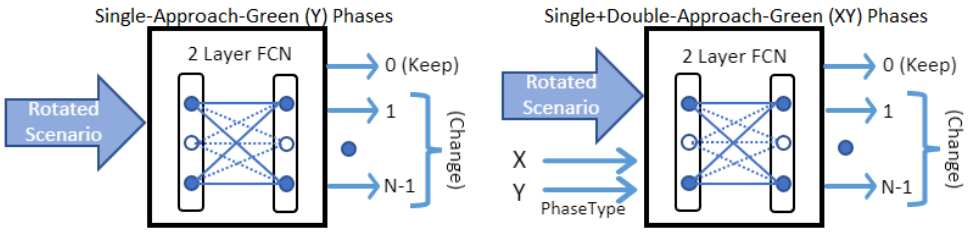

Fig. 8. DRL architecture and actions.

5.3 FrugalLight Rewards

Rewards need to be carefully crafted, so that the DRL algorithm can train to convergence. Rewards need to be fed back to the DRL algorithm, from the environment in which the DRL is applying control. So essentially, the rewards need to be computed from the approaches at the traffic intersection, by the computer vision algorithms. Listed next are some rewards, in increasing order of complexity of the vision algorithms.

- **Queue density:** This is the set of traffic densities of all incoming approaches. It can be calculated using background subtraction [6], which we find works even in poor lighting conditions, as the vehicles' head and tail lights create enough features.
- **Stop density:** This is density of vehicles stopped (halted) at the intersection in all the approaches, waiting for their opportunity to be in motion. It can again be calculated using background subtraction [6], having a different learning rate than for queue density, as queue density considers both halted and moving vehicles. Optical flow method [33] also facilitates motion detection, thus identifying stalled vehicles.
- **Max pressure:** Max pressure can be approximated by the difference of queue densities at incoming and outgoing approaches. It requires data capture and computation at both incoming and outgoing approaches, thus involving camera and embedded computer deployment in subsequent intersections.
- **Cross count:** Exact count of vehicles crossing the intersection is called cross count. This requires tracking vehicles entering and exiting any approach. Thus YOLO (You only look once) [36] or similar CNN based vehicle detection and subsequent tracking of each individual vehicle needs to be done, which have not been shown to work in poor lighting conditions [8], for example during evening peak hours. Also, there are intermittent poor vehicle detections as discussed in § 3.3 (Figure 4)

We use the *Stop density* reward, which can be easily computed by background subtraction [6] at the edge device, for our evaluations in § 6.5. We also evaluate our DRL algorithms with different rewards in § 6.6.

5.4 DRL compression using domain knowledge

We explored carefully crafted DRL states utilizing domain knowledge of traffic light control. The goal is to reduce the computational and memory overheads of the DRL algorithm along-with reducing dependencies on computer vision methods that might be infeasible for developing region traffic. We describe below our four domain-specific optimizations (illustrated in Figure 9 and Figure 10).

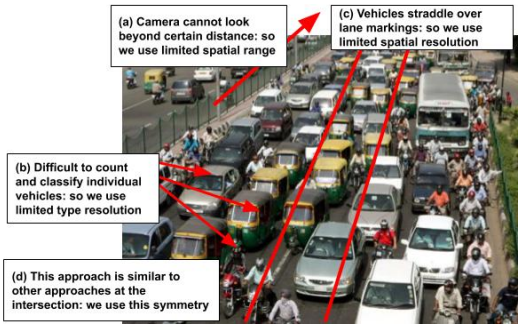

Fig. 9. Domain knowledge based optimizations

❶ **Reduced spatial range of information:** Developing region cameras placed at an intersection can see traffic up to a limited distance, depending on their angle of elevation. It is not frugally feasible to repeat cameras throughout the approach to measure traffic for the whole approach. Therefore our DRL state will be constrained with limited information within a certain distance from the intersection, and only for incoming traffic. The first column of Figure 10, termed *Actual Traffic Layout*, shows in color the limited camera vision, though there are vehicles in the regions that the camera cannot see. The next column *Transformed Layout*, in Figure 10, removes vehicles beyond camera's visual range, which will not be part of DRL state.

❷ **Reduced type granularity of information:** In our DRL states, instead of exact vehicle count requiring computation intensive CNN based YOLO [36], we use traffic density, based on less computation intensive background subtraction [6]. Density is also better than queue length (which indicates the distance at which the last vehicle is standing), as chaotic driving in developing regions sometimes create long queues with haphazard gaps in between. Smaller vehicles like auto-rickshaws and motorbikes can trickle in those gaps, keeping the queue length same but increasing traffic density. Density thus better captures the traffic state on the road, independent of heterogeneous vehicle sizes and their chaotic placement.

❸ **Reduced spatial granularity of information:** As drivers in developing regions do not follow lane markings, vehicles straddle across lanes. So to reduce DRL state complexity, we reduce the information granularity from per lane to per approach, averaging the traffic over the lanes. We further explore a DRL state of just two numbers for each phase, one representing density for the approach with green signal and another for the total densities of all other approaches getting red. We finally reduce the DRL state for a particular phase to only one value, where we take the ratio of the green approach density to the overall density of all approaches. These subsequent reduction in DRL states per phase is shown as *Lane*, *Approach*, *Group* and *Relative* Density in Figure 10.

❹ **Exploiting symmetry:** We finally remove the current phase information from the DRL state. Utilizing the symmetry of traffic intersections, we opt to rotate the DRL state to make the data of the current green phase as the first (or any constant) position. In Figure 10, instead of two columns representing two phases, there is thus a single phase with one approach green and others red. When the approaches differ in properties (like number of lanes, width,

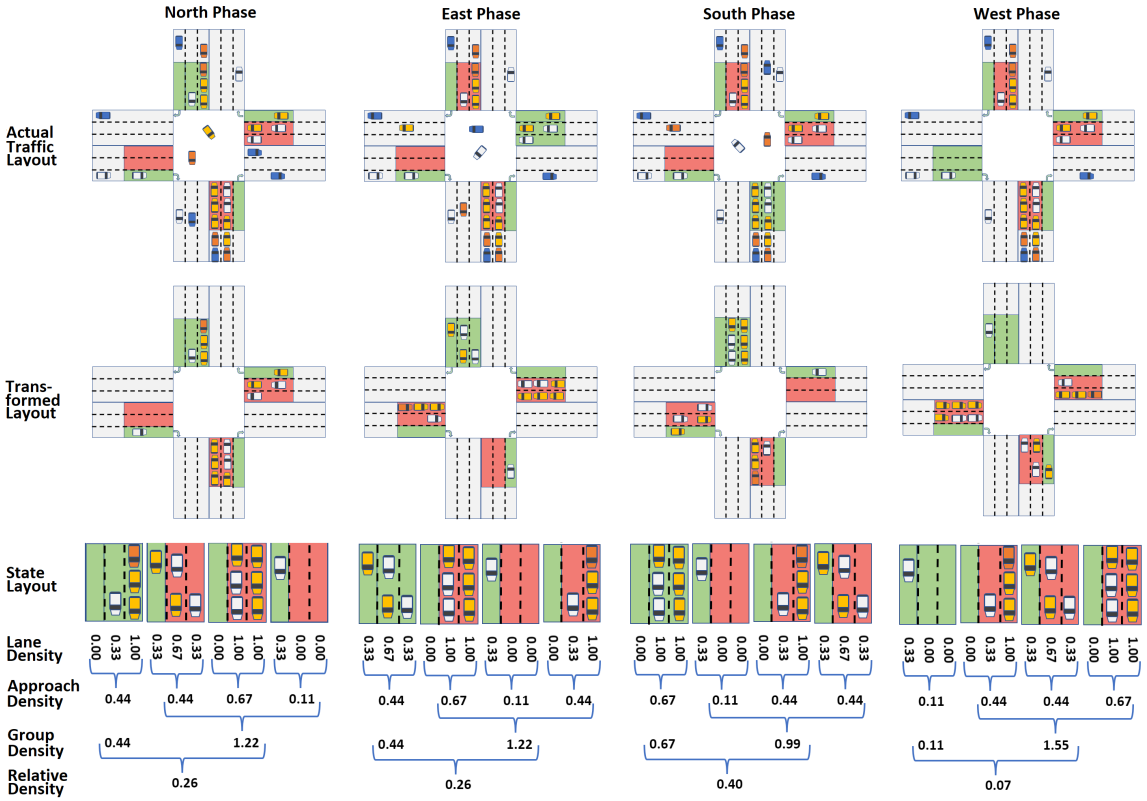

Fig. 10. DRL states from four signal phases

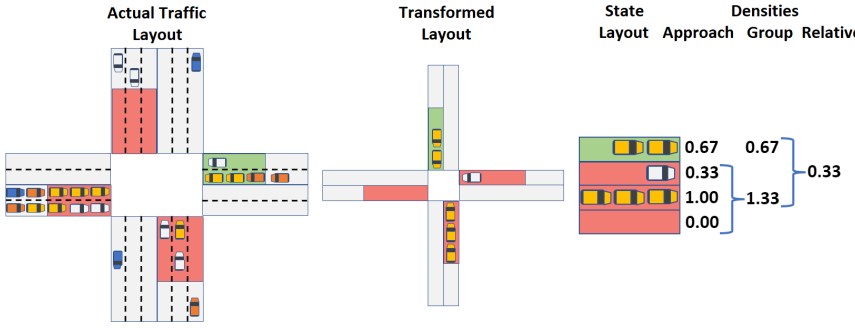

Fig. 11. Heterogeneous intersection, with both two-laned and three-laned approaches

etc. as shown in Figure 11), we cannot perform a simple rotation to get a phase-free-state. There are two options to make a heterogeneous intersection homogeneous – **i. Padding:** We can pad empty lanes in the smaller approaches to make the intersection homogeneous. This would leave some lanes with zero traffic. **ii. Normalizing to unit lane:** We can normalize the density of heterogeneous approaches, scaling down each approach's density to unit lane. ∀ i in approaches, where w_i is a scale factor based on width, number of lanes or other appropriate parameters for approach i.

$$approach\_density[i] \leftarrow \frac{1}{w_i} \sum_{lane \in i} lane\_density[lane]$$

6 FRUGALLIGHT EVALUATION

Giving several design choices for efficient DRL algorithms with constrained inputs, we next evaluate whether any of them can match the performance of computationally intensive state-of-the-art methods.

6.1 Baselines:

We compare our FrugalLight algorithms over state-of-the-art RL models: ❶ *Presslight (PL)* [45], for decentralized multi-intersection processing and ❷ *Colight (CL)* [46], for centralized multi-intersection processing. We also utilized two NonRL algorithms, popular in recent research as standard baselines: ❶ *Max Pressure (MP)* [42], where phase shift occurs based on the difference of vehicles on the incoming and outgoing lanes. ❷ *Self-Organizing Traffic Light (SOTL)* [12], where after a minimum phase duration, the signal is switched based on traffic level in green and red approaches.

6.2 Benchmarks:

We use multiple real road datasets in our experiments, as described in Table 3 in § 3.4. The New York datasets are publicly available[6], which are already processed and used for experiments in prior work [45, 46]. The state-of-the-art works need richer traffic information which cannot be gathered with Computer Vision methods, so to be fair with them we too use their advertised datasets [45, 46] in our experiments, and validate/present the performance of our input-constrained efficient control for global scenarios as well. We also use self-curated developing region datasets[7] which are 1 to 6 hours long and extracted from larger duration of traffic flows available in the original density data[8] for weekdays (8AM - 2PM).

6.3 Simulator:

We use the CityFlow traffic simulator [52] in our experiments. It takes the road network structure, traffic phase information and incoming traffic details through files. We create these files based on the real road datsets described. CityFlow allows us to set the desired phase using API calls. For every phase switch, a 5-second combined yellow and all-red interval exists to clear the intersection. CityFlow also provides the traffic information i.e. what happened on applying the phase switch/hold advised by a particular traffic light control algorithm. This output list of vehicles, along with their locations, is processed to compute our throughput, travel time and total time metrics, to compare across the traffic light control algorithms.

6.4 DRL Efficiency:

Given our primary goal is to have more *efficient* DRL models for resource constrained settings in developing countries, we first quantify how efficient our optimizations are, compared to the baselines. In Table 5, our solution FrugalLight is denoted as FL, with Lane(L), Approach(A), Group(G) and Relative(R) indicating increasing optimizations. The DNN parameters are a range of values, as they depend on the control choice of *Switch to Next* with binary output vs. *Switch to Any* with multinomial outputs. The DRL state size and DNN parameters indicate the significant lower FrugalLight overhead. Such a small DRL model can be deployed on a moderate cost embedded system for roadside deployment, as currently being piloted with our industry partner. Other properties of the underlying DNN are given in Table 4 and properties of Non RL baselines are given in Table 6.

[6] *https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page*
[7] *https://github.com/sachin-iitd/FrugalLight*
[8] *https://delhi-trafficdensity-dataset.github.io*

Table 4. DNN properties

State	Reward	Weight	Loss Fn	Optimizer	LearnRate	Discount
Queue Density	Stop Density	-0.25	MSE	RMSprop	0.001	0.8

Table 5. Model property for 4approach x 3lane intersection

Model	RL State Size	DNN Arch	DNN Params
Presslight	80	-	2082-2124
Colight	1600-12480	-	6018-6084
FL-Lane	12	20x20	722-764
FL-Approach	4	15x15	347-379
FL-Group	2	10x10	162-184
FL-Relative	1	5x5	52-64

Table 6. Properties for Baseline NonRL Algorithms

Algorithm	Properties
MaxPressure	5s Min Green
SOTL	2/4 veh, 5s Min Green

6.5 FL performance on existing open-source datasets

The critical question to evaluate is whether FrugalLight's efficiency comes at a trade-off for throughput or travel time metric values. We present these results next, evaluated on Nvidia DGX Workstation (with 4X Tesla V100 GPUs). As shown in Figure 12, the FrugalLight models converge faster both in terms of number of episodes, and time taken per episode. The FL models converge fairly well by 30 episodes whereas it takes 60+ episodes for the PL model to converge. Also, FL models take 71% to 74% time per episode as compared to the time taken by PL (i.e. saving 26% compute resources per episode).

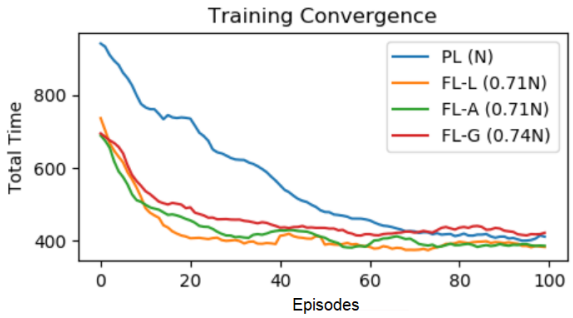

Fig. 12. Training Convergence over RL models. For time taken per episode, PL takes N seconds and FL-L variant takes 0.71N seconds.

As the DRL models converge well within 200 episodes as, we perform training for 250 simulation episodes and the performance is measured and averaged over next 50 (i.e. 251-300) unseen episodes, to vouch for stable experimental results. Figure 13 show the throughput, travel time and total time metrics when the DRL and NonRL based traffic light control algorithms are evaluated for the New York datasets. Two control choices of *Switch to Next* (denoted by Next) and *Switch to Any* (denoted by Any) are shown. We use stop-density as the DRL reward in these plots. For NonRL algorithms, the metrics are consistent for each round, and hence a single round metrics are shown. Higher throughput and lower travel and total times indicate better performance.

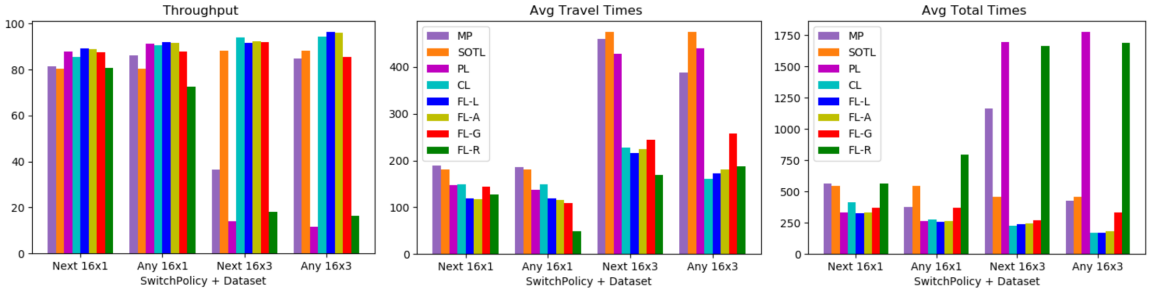

Fig. 13. Evaluation metrics for various algorithms on different Switch Policy and Dataset

Among the two DRL baselines, PL with de-centralized control for multiple intersections, works poorly for the larger 16x3 road network (seems due to incompatible MaxPressure reward for 2D grids), while CL with centralized control does well. Both CL and PL perform well for the smaller 16x1 network. The NonRL baselines (MP and SOTL) show lower performance than the DRL baselines. The two control choices do not show significant performance differences. *Switch to Any* with more freedom to choose outputs and therefore with potential to perform better, is less usable in a developing country, where to avoid more unruliness than already is on the road, *Switch to Next* is mandated at intersections to have fixed precedence among waiting drivers. Thus the lack of performance difference between the two control choices is encouraging.

More encouraging, however, is FrugalLight's performance. Our algorithms FL-L, FL-A and FL-G do as well as CL for all road settings. The most optimized version FL-R degrades for the 16x3 network (and slightly for 16x1 too), possibly due to the incapacity in capturing absolute traffic density. But even the second most optimized version FL-G (showing a slight under-performance for *Any 16x3*), with only 2-sized DRL states and 184 DNN parameters, can match the performance of CL with upto 12480 sized DRL states and 6084 DNN parameters!

FL-G is also de-centralized, not requiring network communication across multiple intersections to have centralized control as in CL. The 2-sized DRL states and the stop density reward can all be computed with simple background subtraction [6] based computer vision methods. This is a tremendous result for developing regions, that a de-centralized DRL algorithm with constrained computer vision inputs and very efficient model parameters, can match the performance of a centralized, more computation intensive, much larger state-of-the-art DRL model.

Different Phase Schemes (Y,X,XY): We next analyze the FrugalLight performance over different phase schemes, such as *Double-approach-green* (X) pattern and a mix of both (XY), as depicted in Figure 14(Left). Colight(CL) is centralized and alongside local information, it seeks neighbouring intersections' information to decide policy for any given intersection. This creates network-dependency and data-latency, alongside complicating the model by increasing state space significantly larger than other models (refer Table 5. While gathering the real data for single intersection, we also observed issues related to power line failures, camera faults, and broken communication. Any fault requires a manual repair, which is a costly and time consuming effort. Hence, network-dependent solution (Colight) is less feasible for deployment in developing countries and single intersections. As shown in Figure 14(Right), we see that FL performs better than baseline PL for all phase schemes.

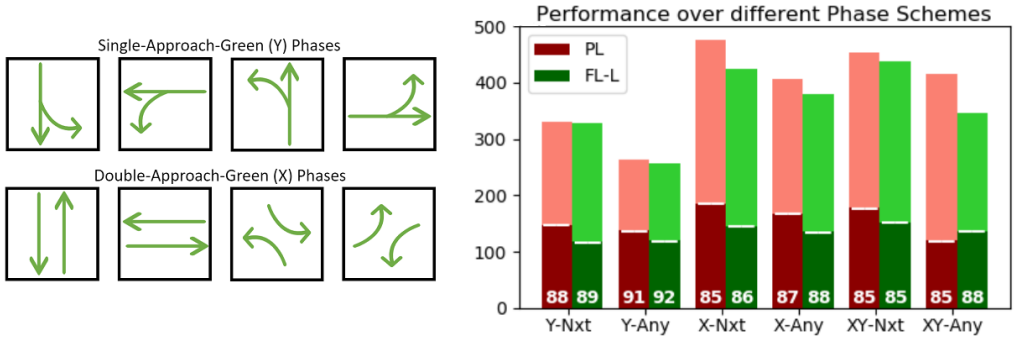

Fig. 14. (Left) Allowed phases at intersections (Right) Performance over different phase schemes, numbers at bottom of bars denote throughput, darker bars denote travel time and lighter bars (at the top) denote total time.

Scaled Road Lengths: Real roads have varying lengths, causing different capacity of waiting-traffic at the approaches. The longer the length of the road, the more traffic it can hold, which may require one lengthy green phase or multiple green phases to pass through the intersection. To see the scaling of FrugalLight on various road lengths, we experiment with different road lengths in the simulator (for 16x1 network) in Figure 15. Compared to state-of-the-art Presslight (PL), FL shows improvements in all metrics, which enhances further as we scale-up the road lengths. Thus, FrugalLight can potentially scale to any road dimensions with similar benefits in average metric values.

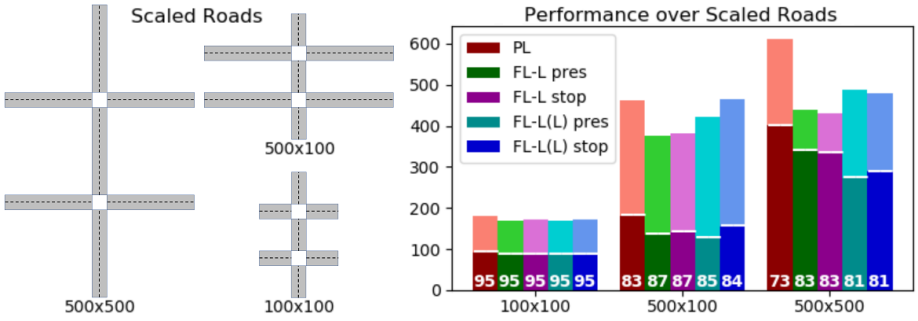

Fig. 15. Performance over scaled road lengths, numbers at bottom of bars denote throughput, darker bars denote travel time and lighter bars (at the top) denote total time.

6.6 FL performance on our New Delhi dataset

We finally analyzed FrugalLight's performance on our collected data for different datasets of increasing duration. As cityflow simulator has a proprietary format for accepting traffic information via input json files, we utilize the density-to-simulator [9] conversion script to convert the real datasets for use with the simulator. In case of real deployments, the density from our Background Subtraction algorithm can be directly fed to FrugalLight models. Along with StopDensity, we did experiments over other rewards (MaxPressure, CrossCount and QueueDensity) to gauge their suitability with our method. The weight for CrossCount is 1 and -0.25 for others. The experiments are shown in Table 7.

We can see that FrugalLight gives good performance, over the developing region data processed with methods different than NY data, for different state sizes and different rewards, except QueueDensity. QueueDensity is the least

[9] *https://github.com/sachin-iitd/TrafficDensity/density.py*

Table 7. Performance on different duration 1x1 developing region data (total time @ throughput)

Model	Reward	1 (1Hour)	2 (1Hour)	3 (2Hour)	4 (4Hour)	5 (6Hour)
PL	MaxPressure	273.3 @ 92.3	254.6 @ 92.8	187.9 @ 97.2	174.0 @ 98.7	112.7 @ 99.4
FL-L	StopDensity	246.7 @ 93.0	**243.7 @ 93.1**	**178.4 @ 97.3**	**170.8 @ 98.7**	98.6 @ 99.5
FL-A		260.6 @ 92.6	253.2 @ 92.8	205.1 @ 96.9	189.5 @ 98.5	**95.2 @ 99.5**
FL-G		246.0 @ 93.0	246.4 @ 93.0	199.2 @ 96.9	191.0 @ 98.4	103.2 @ 99.5
FL-R		**242.8 @ 93.2**	254.5 @ 92.8	225.8 @ 96.7	363.7 @ 97.3	273.8 @ 98.6
FL-L	MaxPressure	261.6 @ 92.6	244.7 @ 93.0	181.8 @ 97.3	170.7 @ 98.7	111.7 @ 99.4
FL-L	CrossCount	*238.5 @ 93.2*	*231.2 @ 93.4*	*169.2 @ 97.4*	*140.4 @ 98.9*	*96.2 @ 99.5*
FL-L	QueueDensity	446.6 @ 87.4	446.1 @ 87.4	458.6 @ 93.3	437.6 @ 96.3	343.4 @ 98.3

computationally intensive reward, but it is not as performant as the other rewards. As computer vision constraints make some rewards harder to compute in developing region, more compute intensive rewards CrossCount (needs vehicle identification and counting) or MaxPressure (needs camera and computing on successive intersections) are not suitable for efficient deployments despite good/comparable performance. However, StopDensity, which is easily computable using background subtraction [6] and performs similar to other rewards, gives a good trade-off between computability and DRL performance, and therefore has been used as the default DRL reward in all our experiments.

7 ENHANCED FRUGALLIGHT

We next explore the inclusion of state-of-the-art techniques and focused optimizations to further enhance FrugalLight for training and deployment scenarios.

7.1 Student-teacher knowledge distillation, with FrugalLight's domain knowledge

Knowledge distillation is the standard method of compressing large machine learning models (teacher) into smaller more efficient models (student) [18, 27]. We analyze two methods of student-teacher learning in this paper - Blind and Explored.

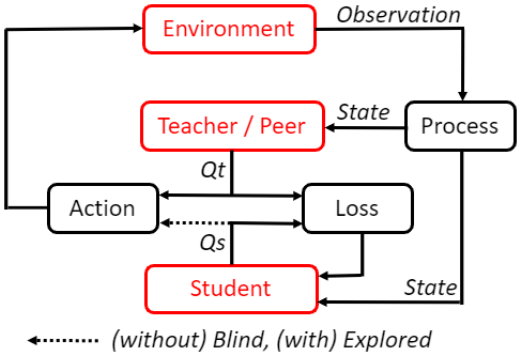

Fig. 16. Blind and Explored learning methods.

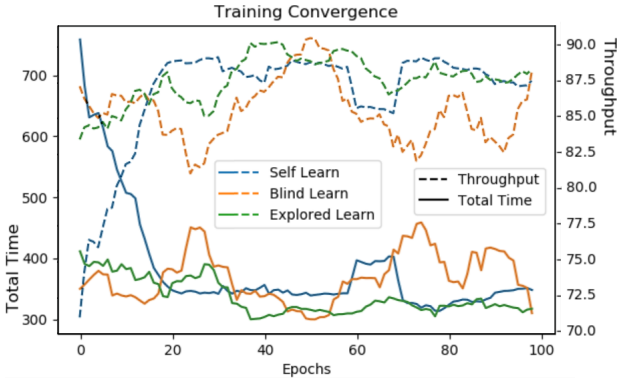

Fig. 17. Convergence of Teaching Strategies, Dotted Lines denote Throughput, and Solid Lines denote Total Time.

In Blind learning (Figure 16 without dotted line), student model learns the environment by blindly following teacher's steps for every state, hence teacher controls the learning. In Explored learning (Figure 16 with dotted line), student

model learns the environment with self exploitation, asking teacher only during exploration phase, hence student controls the learning. For both scenarios, the teacher provides the Q-values for every experience tuple, and local rewards are ignored, thus student tries to fit to teacher's understanding of the environment.
We use the following loss function -

$$MeanSquareErrorLoss \leftarrow \frac{1}{N}\sum_{i=1}^{N}(Qt_i - Qs_i)^2$$

where, Qs_a denotes the Q-values predicted by the student network and Qt_a denotes the target Q-values given by the teacher/peer network, for N training samples.

We hence consider a combination of domain knowledge based DRL state compression along-with knowledge distillation, where we use distillation between models of similar dimensions (peers). These experiments are performed on a 300x300 length 16x1 NY road network, for *switch to the next phase* signal policy. We train for 250 epochs, then average metrics (total time @ throughput) for next 50 epochs are reported in Table 8 and Figure 17.

Table 8. Knowledge distillation (total time @ throughput). Peers (same row in **bold**) teach better than PL (in *italics*).

	Self	**Blind Learning**				**Explored Learning**			
	Learn	**PL**	**FL-L**	**FL-A**	**FL-G**	**PL**	**FL-L**	**FL-A**	**FL-G**
Teacher		316@89.7	286@90.7	279@91.0	340@88.4	316@89.7	286@90.7	279@91.0	340@88.4
PL	345@87.7	1005@59.3	1051@59.1	988@58.5	905@63.6	**335@87.7**	301@90.1	291@89.8	371@86.8
FL-L	328@88.4	1179@52.3	433@84.5	310@89.7	394@85.4	*1118@56.0*	**309@89.7**	331@87.9	355@87.7
FL-A	332@88.3	1173@54.0	496@78.9	267@91.6	365@86.9	*1126@55.4*	342@87.9	**310@89.6**	357@87.6
FL-G	385@86.5	1145@54.2	421@85.6	359@85.9	375@85.8	*880@64.6*	376@85.9	343@88.2	**360@87.7**

The values against the "*Teacher*" row at the top, depicts the performance of the single Teacher model, selected based on *best metric values* from the epochs towards the end. The "*Self Learn*" column on the left, gives the average metric values without knowledge distillation. Self-learning converges slower than distillation, as expected (Figure 17). Explored learning gives better results than blind learning (right side of Table 8 has better values than left), and also gives faster convergence and better stability (Figure 17) than blind.

As seen in Table 8, *Explored Learning* gives better average metric values than *Self Learn*, when models of same size and architecture (called peers) are used for teaching. This improvement in *average metric* through peer learning, is evident if we compare the bold values in the same rows. The values in *Teacher* row show the best metric, not the average, and hence are not suitable for comparison with the bold values. Using the state-of-the-art large PL model [45] as teacher, however, greatly degrades average metrics (indicated in italics in each row). Thus knowledge distillation from a significantly larger model to a significantly smaller student model is more tricky, while similar model as teacher boosts utility [10]

So, large teacher models (e.g. the DRL model in [45]) cannot efficiently train smaller student models. However, peer based knowledge distillation among comparable small models, boosts the optimization metrics for all peers. While recent literature uses collaborative peer learning in an online setting [18, 27], we use this offline, where peers use knowledge distillation to boost optimization metrics before deployment. Using carefully crafted small DRL models with domain knowledge, along with peer learning, therefore further boosts the application utility in our scenario.

[10] Similar results have been independently obtained in a recent research [31].

7.2 FrugalLight's Transferability and Adaptability

Figure 18 shows the transferability (how well a DRL model trained on one dataset performs on a new unseen dataset) and adaptability (how quickly a DRL model fits to the new dataset) of FL vs. baseline PL. The first leg in Figure 18 shows the regular training on the 16x1 NY dataset, depicting improved training convergence for our FL compared to PL. For the next two legs, we utilize two different 1 hour datasets from the 16x1 NY network. We train each model for 300 epochs using the default dataset, then switch the traffic pattern to second dataset and allow training for next 300 epochs, and finally do the same for third dataset. FL is significantly more transferable and adaptable than PL, over different unseen traffic patterns. This can be explained by the large size of the PL model, that tends to overfit to a given training dataset and generalizes poorly to new data.

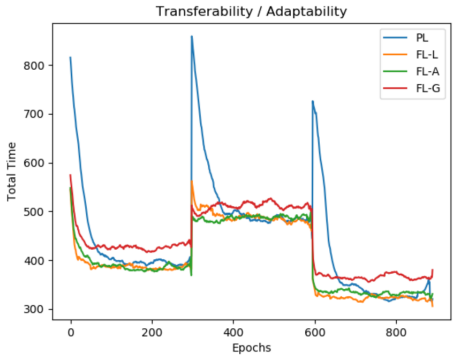

Fig. 18. Transferability/Adaptability of the methods.

	Self	PL	FL-L	FL-A	FL-G
PL	1096@58	1074@60	1018@60	1091@59	1218@54
FL-L	730@73	1187@53	934@61	793@68	642@75
FL-A	599@76	1142@56	708@66	661@75	598@78
FL-G	**568@80**	**922@65**	**618@77**	**481@85**	**588@78**
FL-R	921@67	999@62	1120@59	1125@58	844@69

Fig. 19. Transfer to other dataset for ExploredLearn models.

We further take an ensemble of 50 models from epochs 251-300 for each training experiment, and use them to train 50 peer models using the Explored Learn method. We then evaluate these peer trained models on an unseen dataset and present the average performance of these models in the table in Figure 19.

FL-G is the best generalizable model, both self-learnt and peer-teacher guided, for transferring an ensemble of models to unseen dataset.

7.3 Enhanced Adaptability using Gradient based Meta Learning (MAML)

Meta Learning enables a Machine Learning system to learn fast. Model-Agnostic Meta-Learning (MAML) is a general optimization algorithm suitable for models employing gradient descent. Given multiple tasks, the parameters of a model are trained such that few iterations of gradient descent with few training data from a new task will lead to good generalization performance on that task. MAML *trains the model to be easy to fine-tune* [1]. MAML gradient can be shown by the standard expression [17]:

$$g_{MAML} = \nabla_{\theta_k} \mathcal{L}^{(1)}(\theta_k) \cdot \Pi_{i=1}^{k} (I - \alpha \nabla_{\theta_{i-1}} (\nabla_{\theta} \mathcal{L}^{(0)}(\theta_{i-1})))$$

where θ are model parameters, α is the step-size, and $\mathcal{L}$ is the Loss Function to-be-minimized evaluated over k training samples.

FirstOrder MAML (FOMAML) ignores the 2^{nd} derivative, resulting in a simplified and efficient implementation.

$$g_{FOMAML} = \nabla_{\theta_k} \mathcal{L}^{(1)}(\theta_k)$$

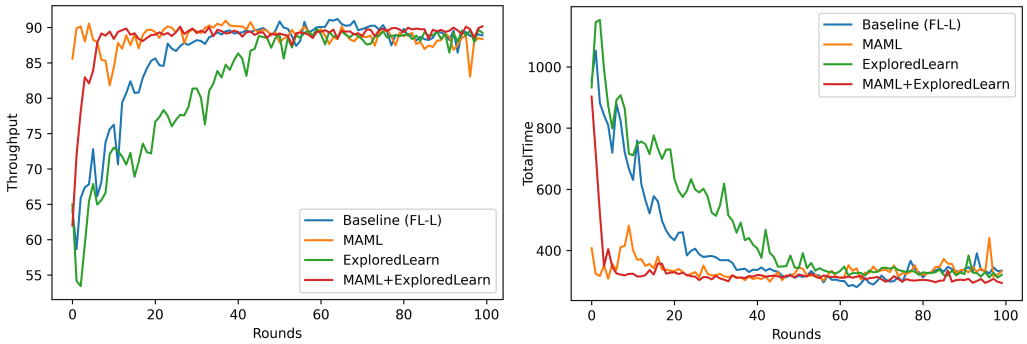

Fig. 20. MAML based Enhanced Adaptability.

So, we consider the first-order gradient based MAML in our experiments. The FOMAML algorithm provides a good improvement without much computational hindrance, making the overall training/optimization process computationally efficient. For the NY dataset of 16x1 intersections, we randomly select a group of 5 nodes and train the networks for these 5 nodes as usual. We train another network using first-order MAML approach with data samples from these 5 nodes. Now this pre-trained model is utilized to train the remaining 11 nodes. For the purpose of metrics calculation, we train all 16 nodes with the pre-trained MAML model. The results for the same are depicted in Figure 20 (for 100 rounds, averaged over 5 runs).

We take a subset of total intersection nodes, the data for which act as meta-data to train the meta-network. This meta-network acts as pre-trained model for other nodes, enabling faster convergence. We also combine MAML with our Explored Learning technique and train the student model from pre-trained MAML model and non-MAML teacher model. We observe a more stable training with added benefits of the two.

7.4 Doing Away with Runtime DRL: Lookup Table based Intersection Control (Goodness EcoLight)

We also seek to do away with running the DRL at runtime at the deployment site. The first reason is efficiency: on low cost embedded systems, compute power is limited. The inputs for the control algorithms anyway needs to be computed on the embedded devices, using computer vision algorithms on the real time video data from all approaches. Using these inputs, if the control algorithm can be made more efficient than running a neural network for DRL, it becomes more practical to meet the low computational budget. The second reason to do away with runtime DRL, is the lack of confidence on the DRL black box. Based on anecdotal evidence through discussions with our deployment partners, adaptive intersection control that can be visualized and verified by human experts before deployment, is much more preferred than algorithms which are free to choose actions at runtime without any human supervision/comprehension, as a runtime DRL would do.

We therefore seek to use static Lookup Tables (LUT) at deployment, where each cell in the table will represent a state in our DRL. The value contained in that cell will represent a boolean action: stay in the current phase vs. switch to the next phase, referred to as *keep-change* actions henceforth. The actions are learnt using offline DRL training. This training can be compute heavy and high latency, as it is run on powerful GPU servers before deployment for real time intersection control. During training, computer vision based processed video datasets are collected from the road, and fed in traffic simulator to create all possible DRL states (cells in the LUT). Actions corresponding to each state are then learnt by training the DRL algorithm.

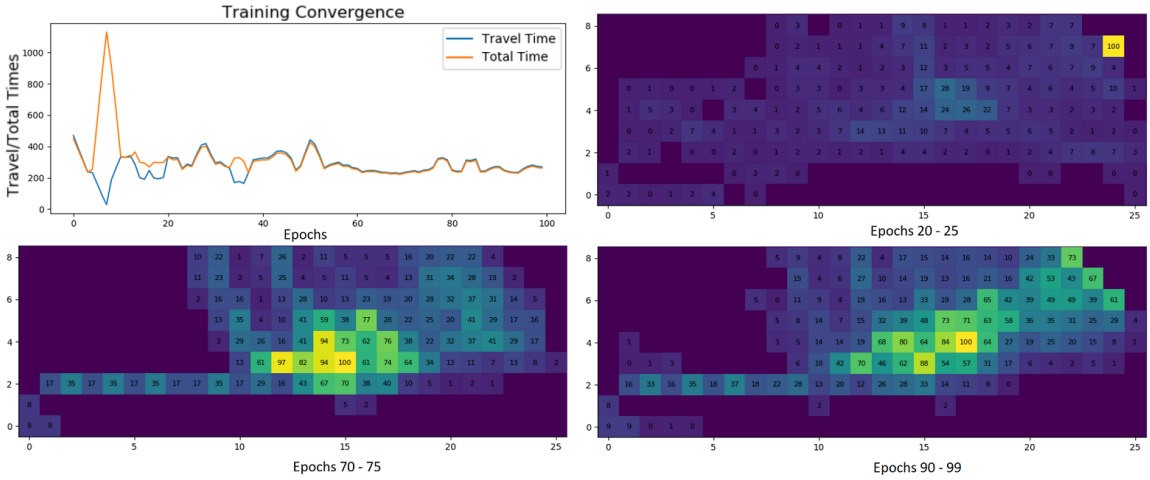

Fig. 21. DRL training and LUT structure

The first graph in Figure 21 shows how metrics Total time and Travel time improve over many epochs of offline DRL training. The other three images show how many times different DRL states are seen by the DRL training algorithm as training progresses. The lighter the color, the more a DRL state is seen. These three images also describe the LUT structure, where the two axes represent quantized values of $x1$ and $x2$ for the 2-dimensional state DRL (FL-G). Instead of "how many times a DRL state is seen" presented in these images, the LUT contains a boolean action value in each cell, learnt by DRL training. Verified by developing country traffic control experts for sanity and safety checks, the LUT is eventually deployed on road. At runtime, the current state is computed using computer vision methods on incoming video, and the action corresponding to that state in the stored LUT is taken by the traffic signal controller.

While storing DRL decisions for different states in LUT is efficient and verifiable, we need to ensure that the learnt decisions are good for subsequent use at runtime. It is important to choose good DRL models to populate the static LUT, as unlike running DRL at runtime, the LUT will not be able to dynamically update these decisions.

As measure of DRL model goodness, we define two metrics:

(a) FairShare: We hypothesize that a good RL tries to achieve FairShare of traffic densities among approaches i.e. fit the traffic among at the intersection such that each approach maintains equal/similar density of traffic. To quantify this FairShare property of a given DRL model, we project all instances of observed states (factored by the distance) onto the equal density segments of LUT (corresponding to the diagonal starting at 0,0) in Figure 21. We sum this vector of the projections to get a single scalar, which will be high for models with most states with equal density (like Epoch 90-99 in Figure 21), and low otherwise. This scalar quantifies how balanced traffic is among the approaches for a particular DRL model.

(b) DecisionConsistency: If a model predicts to hold/keep the signal for a state, we hypothesize that a good or stable model should continue to predict the same for all states having higher traffic in the green approach (or low traffic in the red approaches). We name this model property of sticking to the same decision under similar traffic scenarios as DecisionConsistency. To quantify DecisionConsistency, for each green density level ($x1$) we take the ratio of two numbers, the large range of red density ($x2$) over which the keep decision is maintained vs the range followed with opposite decision. The sum of all such ratios gives rise to a scalar which will be larger for models with better DecisionConsistency.

In addition to hypothesizing what properties good DRL models might have, and defining scalar metrics to quantify those goodness properties, we also need mechanisms to use these goodness metrics. We do this in the following two ways:

❶ **DRL training using model goodness metrics:** We use the FairShare and DecisionConsistency scalars during the DRL training process to identify and favour better RL models. We maintain a threshold θ for these scalars, as training progresses. As presented in Figure 22, at each epoch we hold a model if its goodness metric is below θ, lower θ by a factor, and start the training for a fresh model in that epoch. We approve the best model so far (new or on hold), if its goodness metric exceeds θ, or after fixed number (η=5) of retries in that epoch, and move on with the metric value of this model as new θ.

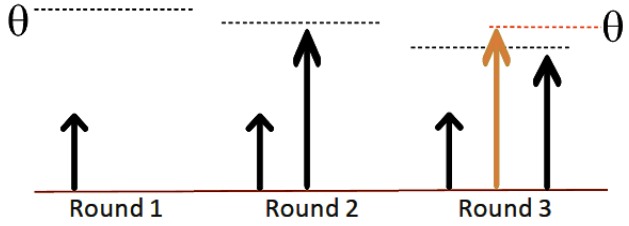

Fig. 22. Goodness metrics based DRL training

❷ **DRL selection using model goodness metrics:** Figure 23 shows the correlation between Total Time performance metric and DRL model's goodness metric values. We discard models with goodness metric values lower than the average of all the models, to remove outliers (see Perspective 1 of Figure 23). In order to select the good models among the remaining ones, we pick the best model (again based on the goodness metric values) among a set of (ψ=20) models, and restart the process from the model next to the selected one (see Perspective 2 of Figure 23). This final set of high performing models can be effectively used to generate the LUT to be deployed at the intersection.

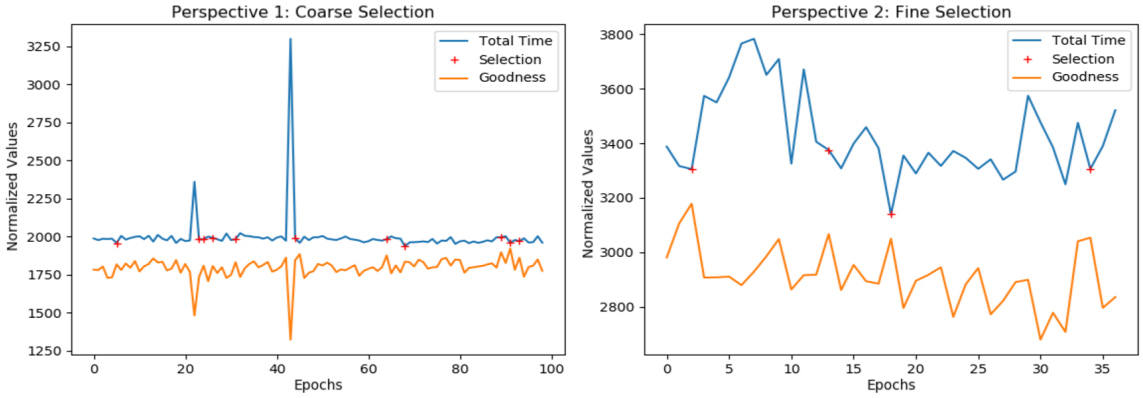

Fig. 23. Goodness metrics based DRL selection

We need to evaluate this LUT based signal control, compared to the FL-G that we designed in § 5, and also the state-of-art DRL methods Presslight [45] and CoLight [46]. Static LUTs lose performance due to quantization of the traffic density values, while runtime DRL can use continuous values of traffic density. But the quantization is unavoidable, as the table needs to be of finite dimensions. Whether our training and training+selection with goodness metrics can overcome the quantization related performance loss, needs to be quantified.

Table 9 shows the average case performance metric values ❶ nOut (number of vehicles cleared by the intersection), ❷ Travel (time spent by cleared vehicles) and ❸ Total (time spent by all vehicles). The T in model names denotes Goodness based Training only experiments, whereas TS includes Goodness based Selection as well. We continue the training for 200 epochs, allowing all methods to converge and then average the next 50 epochs for performance metrics calculation for T, and the selected few out of these for TS. As can be seen from the table, performance loss compared to FL-G due to quantization, is gracefully recovered by both our goodness metrics. DecisionConsistency performs significantly better than FairShare for all datasets.

Table 9. Performance of Goodness EcoLight for average case metrics

	1x1			16x1			16x3		
Model	nOut	Travel	Total	nOut	Travel	Total	nOut	Travel	Total
PressLight	1246	254.4	252.0	4866	219.6	362.8	1355	560.3	930.3
CoLight	1248	222.3	250.9	4986	259.5	374.8	2589	318.9	311.3
FL-G	1282	237.9	243.4	5010	252.4	376.3	2574	328.1	322.6
FairShare(T)	1287	251.3	243.4	4976	244.0	377.0	2561	331.2	330.1
Decision(T)	1292	224.6	239.7	5081	251.3	359.1	2586	327.6	318.6
FairShare(TS)	1285	251.6	243.7	5137	239.9	343.8	2583	327.3	318.1
Decision(TS)	1298	186.4	234.5	5186	*277.4*	357.8	2586	325.5	316.2

Table 10. Performance of Goodness EcoLight for worst case (fairness) metrics for 16x3

Model	WrstTime	WrstWait	MaxWait	Stuck75	Stuck50	Stuck25	Stuck0
Presslight	3516.4	2481.5	255.6	99.2	338.5	843.9	1405.9
Colight	834.4	900.8	45.9	0.0	0.4	2.7	234.7
FL-G	985.3	1396.5	47.6	0.9	2.4	6.0	250.1
FairShare(T)	1207.1	1524.2	48.7	2.1	6.2	14.1	261.3
Decision(T)	924.7	1352.2	47.7	0.0	0.0	1.4	238.3
FairShare(TS)	929.0	1320.0	48.5	0.0	0.0	0.5	241.0
Decision(TS)	675.2	1007.0	46.8	0.0	0.0	0.0	237.6

We further show the value of worst case or fairness metrics for 16x3 benchmark dataset in Table 10. Our fairness metrics are: (a) WrstTime (maximum time spent in the network by any stuck vehicle), (b) WrstWait (maximum wait time at any intersection by any vehicle), (c) MaxWait (maximum of average wait times at any intersection) and (d) StuckX (vehicles stuck in network at X% time from simulation end). Fairness loss due to quantization is not only gracefully recovered by our goodness metrics, but we significantly outperform all baselines as well.

Using a finite sized LUT with (a) quantized traffic density values as rows and columns, and (b) cells containing binary decisions learnt using DRL model training, and model selection based on some goodness metrics, gives us performance and fairness comparable to the state-of-the-art DRL algorithms. This is extremely encouraging in terms of practical deployment in developing countries.

7.5 Doing Away With Look-up Tables: Threshold based Intersection Control (Threshold EcoLight)

Based on anecdotal discussions with intersection control companies, while most intersections in developing regions will be able to support LUTs, some intersections might be budget constrained to such an extent that the controller's RAM will not be enough to even store LUTs. In this section, we therefore consider how to design such a stateless controller, with better performance and fairness metrics compared to other widely deployed stateless controllers. We start by examining the FL-R tried in § 5, and gradually build performant and fair stateless control.

1-dimensional state RL (FL-R) did poorly on the Throughput and TotalTime metrics in Figure 13, especially for the 16x3 road network. Wondering what is being learnt by the RL for the case of 1-dimensional state (in FL-R), we checked the model behaviour for the whole range of this state variable $< x3 = x1/(x1 + x2) >$ from 0.0 to 1.0. We calculate the expected value of signal change for all 16 intersections (of 16x1 NY road network) for continuous 50 rounds after training for 500 rounds.

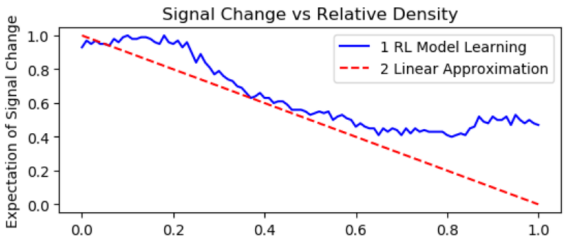

Fig. 24. Density vs action

Figure 24 plots the expected signal change along y-axis, with relative density along x-axis. The signal change expectation is high when relative density is low (top left) and vice-versa (red line given for reference for exact negative correlation between signal change expectation and relative density). The blue curve shows a near-linear response following the red line, but is still non-linear. Thus 1-dimensional state FL-R with ratio $x1/(x1 + x2)$ is not enough to capture the necessary non-linearity and overall traffic concentration - empty vs. moderate vs. saturation. It only captures relative density among approaches, while absolute values retained in 2-dimensional state of FL-G are clearly important.

We explore the options of both 1-dimensional relative density (FL-R) $< x1/(x1 + x2) >$ and 2-dimensional absolute densities (FL-G) $< x1, x2 >$ in the simple algorithm next. The algorithm does not use any LUT to store the signal switching decisions learnt by RL for all possible states. It only uses few empirically learned thresholds. This is to support embedded hardware, that cannot use LUTs due to RAM constraints and would need the control algorithm to be completely stateless, possibly using only a few thresholding parameters.

The intuition behind the algorithm is ❶ to take the *CycleTime* (i.e. the cumulative duration of all phases), and divide it among phases in proportion to their relative densities and ❷ to increase *CycleTime* based on increasing absolute densities. At each decision making point, the agent allows the green signal to continue until the relative density for that approach has not fallen below a threshold α. Below α, signal can be switched. When *CycleTime* is defined (we call this variant Timed), the agent uses it in proportion to the relative density (Timed (1dim)), with optionally increasing the given *CycleTime* in response to absolute densities (Timed (2dim)). When *CycleTime* is undefined (we call this variant Random), it would switch randomly, but still proportional to the relative density. The various hyper parameters are listed in Table 11.

GetNextAction (*cur_phase*, *phase_time*, *density_list*)**:**

```
action ← 0
total_density ← sum(density_list)
if total_density > 0 and phase_time ≥ CONFIG[MinGreen] then
  relative_density ← density_list[cur_phase]/total_density
  if relative_density < CONFIG[α] then
    if CONFIG[Mode] is Random then
      ratio ← random(0.0, 1.0)
    else
      cycleTime ← CONFIG[CycleTime]
      if CONFIG[Mode] is Timed(2dim) then
        cycleTime ← cycleTime × total_density × 2/CONFIG[MaxDensity]
        cycleTime ← MAX(CONFIG[MinGreen], cycleTime)
      end if
      ratio ← phase_time/cycleTime
    end if
    if ratio > relative_density then
      action ← 1
    end if
  end if
end if
return action
```

Table 11. Algorithm Hyper Parameters

Param	Description
α	Hold green above this threshold
MinGreen	Minimum green per phase
CycleTime	Total green time over phases
MaxDensity	Maximum density at intersection
Mode	Random / Timed(1dim or 2dim)

Table 12. Empirically Learnt Values

Algorithm	Properties
FixedTiming	20s Min/Max Green
MaxPressure	5s Min Green
SOTL	2/4 veh, 5s Min Green
Random	α=0.17, 5s Min Green
Timed	α=0.17, 150s Cycle

We compare the performance of our stateless algorithms against below baselines. These baselines also do not use any state, but work with few parameters as listed in Table 12. State-of-the-art research based RL methods like Presslight and CoLight are still in literature and not adopted in the real world. So these simpler baselines are the widely deployed intersection control algorithms across the world. Developing countries, typically, still use Fixed Timing signals.

❶ **Fixed Timing:** Signal switches in cyclic order to the next approach after fixed time intervals.

❷ **Max Pressure:** Pressure is calculated by the difference of vehicles on the incoming and outgoing lanes for the possible movements in each phase [42]. Signal is switched to the phase with maximum pressure. If current phase pressure is not the maximum, we switch to the next phase.

❽ **Self-Organizing Traffic Light (SOTL):** This is a vehicle actuated mechanism [12]. There is a minimum phase duration. Once the minimum phase duration is over, the switch signal is generated if the traffic in green approach is less than a threshold and traffic in any other approach is more than another threshold.

Table 13. Performance of EcoLight Thresholding Algorithms for average case metrics

	1x1			16x1			16x3		
Algo	nOut	Travel	Total	nOut	Travel	Total	nOut	Travel	Total
FixedTiming	1249	260.6	252.0	3743	*193.0*	583.5	1489	723.2	985.7
MaxPressure	1160	280.8	272.8	4106	*214.4*	504.2	1840	649.7	768.8
SOTL	1305	246.7	239.0	4640	*264.7*	436.3	2462	485.9	465.1
Random	1361	231.6	224.8	5076	*354.9*	427.8	2540	378.5	364.9
Timed(1dim)	1358	231.7	255.4	5104	*355.3*	428.9	2516	380.6	368.3
Timed(2dim)	1358	231.7	255.4	5268	*346.6*	406.3	2553	375.2	361.7

Table 13 shows the average case metric values (a) nOut (number of vehicles cleared by the intersection), (b) Travel (time spent by cleared vehicles) and (c) Total (time spent by all vehicles). Our algorithms Random, Timed (1dim) and Timed (2dim), clear many more vehicles at lower Travel and Total times than the baselines, for all benchmark datasets. The Travel times for 16x1 network is higher (italicized in Table 13) for our algorithms, though other metrics improved. This is due to the fact that it is a linear network of 16 intersections and the traffic pattern is such that a good part of the traffic enters around one end and exits around the other (and vice-versa), making the vehicles cross many intersections in a sequence. Supported by increased nOut, our algorithms make more vehicles to exit the network. The extra vehicles which exit are mostly the ones with larger travel times, thus pushing the average travel time for all cleared vehicles higher. Similar behaviour is observed for the baselines as well, where SOTL Travel time (with more nOut) is higher than other baselines (with less nOut).

Table 14. Performance of EcoLight Thresholding Algo for worst case (fairness) metrics for 16x3

Algo	WrstTime	WrstWait	MaxWait	Stuck75	Stuck50	Stuck25	Stuck0
FixedTiming	3443	2671	255.6	82	348	741	1203
MaxPressure	3100	2347	261.6	11	154	449	942
SOTL	1229	2188	79.8	0	0	11	362
Random	841	526	56.1	0	0	0	284
Timed(1dim)	839	524	56.5	0	0	0	308
Timed(2dim)	719	516	54.2	0	0	0	271

We further show the value of worst case or fairness metrics for 16x3 benchmark dataset in Table 14. For our Random variant, we take average of 5 rounds of simulation. For all others, the results are consistent for every round. Our algorithms significantly outperform the baselines for all fairness metrics for 16x3 network, and also for other benchmarks (omitted here for space constraints).

Table 15 shows performance of EcoLight Algorithms on two addition datasets collected at different times on the same intersection in New Delhi (India).

Table 15. Performance on other 1x1 datasets

	2			3		
Algo	nOut	Travel	Total	nOut	Travel	Total
PressLight	225	30.0	29.4	529	163.1	177.6
Colight	221	31.2	49.1	514	163.2	181.7
FL-G	225	30.4	29.9	540	182.3	178.6
FairShare(T)	225	31.3	30.9	517	194.5	189.5
Decision(T)	225	30.3	29.8	538	185.2	180.2
FairShare(TS)	225	32.2	31.7	538	181.7	178.4
Decision(TS)	225	30.4	29.9	564	172.8	167.8
Timed(2dim)	225	31.9	31.3	566	168.9	162.5
Timed(1dim)	225	31.9	31.3	563	174.8	166.9
Random	225	31.9	31.3	562	172.8	166.3
SOTL	223	50.7	53.8	539	196.0	186.7
MaxPressure	224	48.4	47.3	487	205.5	202.0
FixedTiming	224	70.9	69.0	512	198.4	188.8

Based on these results, in situations where running RL based control or maintaining LUTs are not feasible due to RAM constraints, our stateless algorithms can be deployed, vastly improving both performance and fairness metrics, compared to the currently deployed intersection control baselines.

8 INPUT TO CONTROL ALGORITHMS: COMPUTER VISION FOR END-TO-END SYSTEM

All intersection control algorithms designed in this paper – ❶ FrugalLight DRLs (§ 5), ❷ Explored Learn FrugalLight DRLs (§ 7.1), ❸ MAML based FrugalLight DRLs (§ 7.3), ❹ LUTs built from offline DRL training using quantized states (§ 7.4) and ❺ Stateless threshold based FrugalLight algorithms (§ 7.5), use traffic density as input. More specifically, the algorithms need density of standing traffic (also called stop density), discarding vehicles which have started moving.

Given the hardware constraints, we need to make sure that this input is available to our control algorithms at an acceptable latency, with limited computation and no communication to a back-end server. As efficient computer vision candidates, we use background subtraction and optical flow techniques as discussed in § 3.2. Background subtraction based density estimates comprise both standing and moving traffic, whereas the control algorithms need to discard density contributed by the moving vehicles. So we additionally use optical flow algorithm, to detect moving pixels between frames, and compute standing traffic density from the stationary parts of the frames.

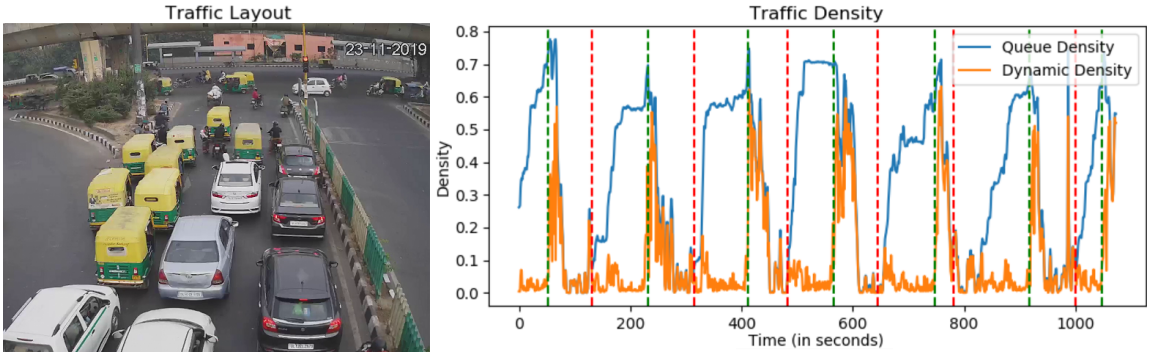

Fig. 25. Developing Region Traffic Density Estimation

Figure 25 on the left shows a high traffic density frame, from one approach of a developing region intersection we are working at. The graph on the right shows for this location: (a) background subtraction based density (Queue Density in blue curve) and (b) optical flow based density (Dynamic Density in orange curve), over a span of over 15 minutes. Queue density starts to rise when signal turns red (indicated by vertical red lines), and starts to fall when signal turns green (indicated by vertical green lines). Dynamic density is zero when red signal is on (between red and green vertical lines) and rises when signal turns green and vehicles start moving. The difference between these two curves gives the density of standing vehicles, the input required by our control algorithms.

The density estimation code runs at 5 FPS on low cost embedded platform (1.8 GHz Intel(R) Atom(TM) CPU D525 with 4 logical cores and 8GB RAM) budgeted by our deployment partners. With signal keep-change decisions taken every 5-10 seconds using LUT or threshold based control algorithms, this FPS is good enough to get inputs for all approaches.

9 CONCLUSION AND FUTURE WORK

This paper shows the feasibility of deployable intelligent traffic light control methods for developing regions, using efficient and optimized computations on low-cost edge devices. Our shared dataset is peculiar in terms of its traffic representation properties despite the various functional challenges. Our proposed traffic control method FrugalLight, which supports using the simplified traffic data, is evaluated on many hours of real world data, both existing open-source from New York, USA[11] and now open-source from New Delhi, India[12]. Though our problem statement comes from developing country, our data and models are useful everywhere empowered by their efficiency and simplicity. We do equally well in both orderly and chaotic situations. FrugalLight also demonstrates that control can be made computationally efficient resulting in less carbon footprint, without losing utility in terms of metric optimization. We will continue to explore how such deployable systems will actually benefit the sustainability goals like air pollution reduction.

[11] *https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page*
[12] *https://delhi-trafficdensity-dataset.github.io*

REFERENCES

[1] 2021. Meta learning (computer science). https://en.wikipedia.org/wiki/Meta_learning_(computer_science).

[2] Mohammad Aslani, Mohammad Saadi Mesgari, and Marco Wiering. 2017. Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events. *Transportation Research Part C: Emerging Technologies* 85 (2017), 732–752. https://doi.org/10.1016/j.trc.2017.09.020

[3] Mohammad Aslani, Stefan Seipel, Mohammad Saadi Mesgari, and Marco Wiering. 2018. Traffic Signal Optimization through Discrete and Continuous Reinforcement Learning with Robustness Analysis in Downtown Tehran. *Adv. Eng. Inform.* 38, C (oct 2018), 639–655. https://doi.org/10.1016/j.aei.2018.08.002

[4] UK Bewiser. 2016. Traffic lights cause traffic jams, new research suggests. https://www.bewiser.co.uk/news/car-insurance/traffic-lights-cause-traffic-jams-new-research-suggests.

[5] Ankit Bhardwaj, Shiva R. Iyer, Sriram Ramesh, Jerome White, and Lakshminarayanan Subramanian. 2023. Understanding sudden traffic jams: From emergence to impact. *Development Engineering* 8 (2023), 100105. https://doi.org/10.1016/j.deveng.2022.100105

[6] Thierry Bouwmans. 2014. Background modeling and Foreground Detection for video surveillance: Traditional and Recent Approaches, Benchmarking and Evaluation. http://www.crcpress.com/product/isbn/9781482205374.

[7] CC-by4. 2013. Attribution 4.0 International (CC BY 4.0). Retrieved June 7, 2023 from https://creativecommons.org/licenses/by/4.0

[8] Mayank Singh Chauhan, Arshdeep Singh, Mansi Khemka, Arneish Prateek, and Rijurekha Sen. 2019. Embedded CNN Based Vehicle Classification and Counting in Non-Laned Road Traffic. In *Proceedings of the Tenth International Conference on Information and Communication Technologies and Development.*

[9] Sachin Chauhan, Kashish Bansal, and Rijurekha Sen. 2020. EcoLight: Intersection Control in Developing Regions Under Extreme Budget and Network Constraints. *34th Conference on Neural Information Processing Systems (NeurIPS 2020).* https://proceedings.neurips.cc/paper/2020/file/97e49161287e7a4f9b745366e4f9431b-Paper.pdf

[10] Sachin Chauhan, Sayan Ranu, Rijurekha Sen, Zeel B Patel, and Nipun Batra. 2023. AirDelhi: Fine-Grained Spatio-Temporal Particulate Matter Dataset From Delhi For ML based Modeling. *37th Conference on Neural Information Processing Systems (NeurIPS 2023) Track on Datasets and Benchmarks.* https://openreview.net/pdf?id=n2wW7goGky

[11] Sachin Chauhan and Rijurekha Sen. 2023. RealLight: DRL based Intersection Control in Developing Countries without Traffic Simulators. In *NeurIPS 2023 Computational Sustainability: Promises and Pitfalls from Theory to Deployment.* https://openreview.net/pdf?id=dmjT841VuV

[12] Seung-Bae Cools, Carlos Gershenson, and Bart D'Hooghe. 2006. Self-Organizing Traffic Lights: A Realistic Simulation. *Advances in Applied Self-Organizing Systems* (10 2006). https://doi.org/10.1007/978-1-84628-982-8_3

[13] Anthony Davis. 2019. Inductive loops or wireless magnetometers for traffic signal control. https://highways.today/2019/03/22/inductive-loops-wireless-magnetometers).

[14] Samah El-Tantawy and Baher Abdulhai. 2010. An agent-based learning towards decentralized and coordinated traffic signal control. In *13th International IEEE Conference on Intelligent Transportation Systems.* 665–670. https://doi.org/10.1109/ITSC.2010.5625066

[15] Samah El-Tantawy, Baher Abdulhai, and Hossam Abdelgawad. 2013. Multiagent Reinforcement Learning for Integrated Network of Adaptive Traffic Signal Controllers (MARLIN-ATSC): Methodology and Large-Scale Application on Downtown Toronto. *IEEE Transactions on Intelligent Transportation Systems* 14, 3 (2013), 1140–1150. https://doi.org/10.1109/TITS.2013.2255286

[16] FHWA. 2006. Traffic Detector Handbook: Chapter 4. In-Roadway Sensor Design. https://www.fhwa.dot.gov/publications/research/operations/its/06108/04.cfm).

[17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (Sydney, NSW, Australia) *(ICML'17).* JMLR.org, 1126–1135.

[18] Qiushan Guo, Xinjiang Wang, Yichao Wu, Zhipeng Yu, Ding Liang, Xiaolin Hu, and Ping Luo. 2020. Online Knowledge Distillation via Collaborative Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

[19] Philipp Gysel, Jon Pimentel, Mohammad Motamedi, and Soheil Ghiasi. 2018. Ristretto: A Framework for Empirical Study of Resource-Efficient Inference in Convolutional Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* 29, 11 (2018), 5784–5789. https://doi.org/10.1109/TNNLS.2018.2808319

[20] Jiang Han, John W. Polak, Javier Barria, and Rajesh Krishnan. 2010. On the estimation of space-mean-speed from inductive loop detector data. *Transportation Planning and Technology* 33, 1 (2010), 91–104. https://doi.org/10.1080/03081060903429421 arXiv:https://doi.org/10.1080/03081060903429421

[21] Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both weights and connections for efficient neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1* (Montreal, Canada) *(NIPS'15).* MIT Press, Cambridge, MA, USA, 1135–1143.

[22] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. 2012. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on* 14, 8 (2012), 2.

[23] Shiva R Iyer, Ulzee An, and Lakshminarayanan Subramanian. 2020. Forecasting sparse traffic congestion patterns using message-passing rnns. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 3772–3776.

[24] Vipin Jain, Ashlesh Sharma, and Lakshminarayanan Subramanian. 2012. Road traffic congestion in the developing world. In *Proceedings of the 2nd ACM Symposium on Computing for Development.* 1–10.

[25] Nicole Kobie. 2018. London is hacking its traffic lights to slash waiting times. https://www.wired.co.uk/article/traffic-lights-uk-london).

[26] Daniel Krajzewicz, Georg Hertkorn, Christian Feld, and Peter Wagner. 2002. SUMO (Simulation of Urban MObility); An open-source traffic simulation. *4th Middle East Symposium on Simulation and Modelling (MESM2002)*, 183–187.

[27] xu lan, Xiatian Zhu, and Shaogang Gong. 2018. Knowledge Distillation by On-the-Fly Native Ensemble. In *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.). Curran Associates, Inc., 7517–7527. http://papers.nips.cc/paper/7980-knowledge-distillation-by-on-the-fly-native-ensemble.pdf

[28] John DC Little, Mark D Kelson, and Nathan H Gartner. 1981. MAXBAND: A versatile program for setting signals on arteries and triangular networks. (1981).

[29] PR Lowrie. 1990. SCATS: Sydney Co-Ordinated Adaptive Traffic System: a traffic responsive method of controlling urban traffic.

[30] Huizi Mao, Song Han, Jeff Pool, Wenshuo Li, Xingyu Liu, Yu Wang, and William J. Dally. 2017. Exploring the Granularity of Sparsity in Convolutional Neural Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 1927–1934. https://doi.org/10.1109/CVPRW.2017.241

[31] Vaishnavh Nagarajan, Aditya Krishna Menon, Srinadh Bhojanapalli, Hossein Mobahi, and Sanjiv Kumar. 2023. On student-teacher deviations in distillation: does it pay to disobey. *37th Conference on Neural Information Processing Systems (NeurIPS 2023)*.

[32] Tomoki Nishi, Keisuke Otaki, Keiichiro Hayakawa, and Takayoshi Yoshimura. 2018. Traffic Signal Control Based on Reinforcement Learning with Graph Convolutional Neural Nets. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 877–883. https://doi.org/10.1109/ITSC.2018.8569301

[33] OpenCV. 2021. Optical Flow. Retrieved Oct 29, 2023 from https://docs.opencv.org/3.4.15/d4/dee/tutorial_optical_flow.html

[34] Srinivas Peeta and Pengchang Zhang. 2002. Counting Device Selection and Reliability: Synthesis Study. *Joint Transportation Research Program* (2002).

[35] L A Prashanth and Shalabh Bhatnagar. 2011. Reinforcement learning with average cost for adaptive control of traffic lights at intersections. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. 1640–1645. https://doi.org/10.1109/ITSC.2011.6082823

[36] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 779–788. https://doi.org/10.1109/CVPR.2016.91

[37] Stefano Giovanni Rizzo, Giovanna Vantini, and Sanjay Chawla. 2019. Time Critic Policy Gradient Methods for Traffic Signal Control in Complex and Congested Scenarios. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 1654–1664. https://doi.org/10.1145/3292500.3330988

[38] Roger P Roess, Elena S Prassas, and William R McShane. 2004. *Traffic Engineering*. Pearson/Prentice Hall. Retrieved Jan 3, 2024 from https://books.google.co.in/books?id=OYNPAAAAMAAJ

[39] Andrea Sassella, Francesco Abbr., Simone Formentin, Andrea G. Bianchessi, and Sergio M. Savaresi. 2023. On queue length estimation in urban traffic intersections via inductive loops. In *2023 American Control Conference (ACC)*. 1135–1140. https://doi.org/10.23919/ACC55779.2023.10156258

[40] Omais Shafi, Sachin Chauhan, Gayathri Ananthanarayanan, and Rijurekha Sen. 2022. DynCNN: Application Dynamism and Ambient Temperature Aware Neural Network Scheduler in Edge Devices for Traffic Control. In *ACM SIGCAS/SIGCHI Conference on Computing and Sustainable Societies (COMPASS)* (Seattle, WA, USA) *(COMPASS '22)*. Association for Computing Machinery, New York, NY, USA, 513–528. https://doi.org/10.1145/3530190.3534823

[41] Elise van der Pol and Frans A. Oliehoek. 2016. Coordinated Deep Reinforcement Learners for Traffic Light Control. Retrieved Jan 3, 2024 from https://api.semanticscholar.org/CorpusID:198950131

[42] Pravin Varaiya. 2013. Max pressure control of a network of signalized intersections. *Transportation Research Part C: Emerging Technologies* 36 (2013), 177–195. https://doi.org/10.1016/j.trc.2013.08.014

[43] Shiva Verma. 2019. Understanding different Loss Functions for Neural Networks. https://towardsdatascience.com/loss-functions-neural-networks-dd1ed0274718.

[44] Y. Wang, T. Xu, X. Niu, C. Tan, E. Chen, and H. Xiong. 2022. STMARL: A Spatio-Temporal Multi-Agent Reinforcement Learning Approach for Cooperative Traffic Light Control. *IEEE Transactions on Mobile Computing* 21, 06 (2022), 2228–2242.

[45] Hua Wei, Chacha Chen, Guanjie Zheng, Kan Wu, Vikash Gayah, Kai Xu, and Zhenhui Li. 2019. PressLight: Learning Max Pressure Control to Coordinate Traffic Signals in Arterial Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. 1290–1298.

[46] Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yamin Zhu, Kai Xu, and Zhenhui Li. 2019. CoLight: Learning Network-level Cooperation for Traffic Signal Control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (Beijing, China) *(CIKM '19)*.

[47] Hua Wei, Guanjie Zheng, Vikash V. Gayah, and Zhenhui Jessie Li. 2019. A Survey on Traffic Signal Control Methods. *ArXiv* abs/1904.08117 (2019). https://api.semanticscholar.org/CorpusID:119116017

[48] David Williams. 2012. Too many traffic lights make congestion worse. https://www.standard.co.uk/hp/front/too-many-traffic-lights-make-congestion-worse-6676646.html).

[49] T. Yang, Y. Chen, and V. Sze. 2017. Designing Energy-Efficient Convolutional Neural Networks Using Energy-Aware Pruning. (jul 2017), 6071–6079. https://doi.org/10.1109/CVPR.2017.643

[50] Jiecao Yu, Andrew Lukefahr, David Palframan, Ganesh Dasika, Reetuparna Das, and Scott Mahlke. 2017. Scalpel: Customizing DNN Pruning to the Underlying Hardware Parallelism. *SIGARCH Comput. Archit. News* 45, 2 (June 2017), 548–560. https://doi.org/10.1145/3140659.3080215

[51] Xinshi Zang, Huaxiu Yao, Guanjie Zheng, Nan Xu, Kai Xu, and Zhenhui Li. 2020. MetaLight: Value-Based Meta-Reinforcement Learning for Traffic Signal Control. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 01 (Apr. 2020), 1153–1160. https://doi.org/10.1609/aaai.v34i01.5467

[52] Huichu Zhang, Siyuan Feng, Chang Liu, Yaoyao Ding, Yichen Zhu, Zihan Zhou, Weinan Zhang, Yong Yu, Haiming Jin, and Zhenhui Li. 2019. CityFlow: A Multi-Agent Reinforcement Learning Environment for Large Scale City Traffic Scenario. (2019), 3620–3624. https://doi.org/10.1145/3308558.3314139