# International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 21 No. 3 (2025)





ijerst.editor@gmail.com

editor@ijerst.com

ISSN 2319-5991 www.ijerst.com
© 2025 IJERST. All Rights Reserved
Peer Reviewed, Referred & Indexed Journal

Research Paper

# RL-FN: A META-REINFORCEMENT FAST-LEARNING NETWORK FOR ENERGY-EFFICIENT AND ADAPTIVE QOS-AWARE ROUTING IN FOG-ENABLED IOT NETWORKS

G. Prabhakar
*Research Scholar, Dept. of CSE*
*Malla Reddy University*
*Hyderabad, India*

Dr. Meervali Shaik
*Professor, Dept. of CSE*
*Malla Reddy University*
*Hyderabad, India*

Dr. A.V.L.N. Sujith
*Associate Professor, Dept. of CSE*
*Malla Reddy University*
*Hyderabad, India*
dravln.sujith@mallareddyuniversity.ac.in

***Abstract*—**While fog computing moves computation to the frontier of the Internet of Things (IoT) devices, routing tasks across a variety of fog nodes is challenging given the conflicting nature of latency, energy, and reliability requirements and the dynamic nature of network conditions. However, statically trained machine-learning routers are unable to adapt once deployed and deep reinforcement learning is too slow to train on fog gateways. In this paper we extend our work on Meta-Reinforcement Fast-Learning Network (RL-FN) that combines a fast neural network using Extreme Learning Machine (ELM) with model-free Q-learning. Fog gateway agents are approximated by a closed-form, regularised ELM, and their Q-policy is refined by a normalised multi-objective reward which punishes for delay, energy and response time. The extension features a formal delay, queueing and energy model, an annealed exploration schedule, a deep Q-network (DQN) baseline, convergence analysis, computational-cost analysis, reward ablations and hidden-layers ablations up to 100 fog nodes. With approximately 18,000 task records, RL-FN achieves 1.3ms end-to-end delay, 99% packet delivery ratio and 98% throughput at 1500 iterations, reducing the delay by 67.5% and 33.3% compared to AdaBoost and DQN, respectively, while maintaining 39.1% more residual energy than XGBoost, and training 160 times faster than DQN.

***Index Terms*—Fog computing, Internet of Things, reinforcement learning, extreme learning machine, Q-learning, energy-efficient routing, quality of service**

## I. INTRODUCTION

The healthcare use cases for connected IoT devices have continued to expand, as have those for industrial automation and transportation and smart cities. When all read requests are sent out to a remote cloud, latency and bandwidth congestion issues will arise of which latency sensitive applications do not tolerate [1]. To overcome this issue, fog computing introduces a new layer of fog nodes and fog gateways between the devices and the cloud, and performs data processing close to the point where the data is generated [2]. When this layer is formed, however, every task will have to be sent to the appropriate fog node via a network with queues, link quality and residual energy that vary over time.

This is a multi objective routing problem. An energy greedy path will have more delay and will use the batteries of a few judicious nodes, whereas, the path with the least delay will consume a few nodes' batteries. Classical machine-learning (ML) routers, like support vector machines, random forests and boosting ensembles, are restricted to learning a pre-determined mapping from features to routes [3] and are incapable of upgrading the mapping when the traffic regime changes. Deep reinforcement learning (DRL) is inspired by online [4], [5] but with deep Q-networks requires iterative back-propagation to train which is costly for the limited processing capabilities of fog gateways.

In the previous work, we have proposed a new deep Q-network based model: Meta-Reinforcement Fast-Learning Network (RL-FN) [6] that uses an Extreme Learning Machine (ELM) with closed-form output weights [7] and adapts the output weights using Q-learning feedback. This paper is an attempt to build upon that work in many ways. It makes the following contributions.

A formal system model of the fog-IoT routing problem which integrates transmission and propagation delay, M/M/1 queueing delay, and processing delay in a fog device with a first-order radio energy model and is formulated as a constrained multi-objective optimisation problem.

A simple RL-FN learning loop, reward (RC) is negative, normalized, explicit temporal-difference (TD) update, and an annealed exploration schedule, and a complexity analysis.

• Extended evaluation in comparison to 10 classical ML routers and a DQN baseline which includes delay, packet delivery ratio (PDR), throughput, residual energy, routing load and packet overhead, and new studies on scalability, convergence, computational cost, reward-term ablation and hidden-layer sensitivity.

In Section II, the related work is reviewed. The system model is given in Section III. RL-FN is described in Section IV, the experimental setup is in Section V, a discussion of the results is in Section VI and the conclusion is in Section VII.

## II. Related Work

Mehri [3] has surveyed the data transfer problem in IoT with ML and categorizes the main targets of data transfer schemes as routing, congestion control and pre-processing; and observes that most of the schemes have been tested only offline. Udayaprasad et al. [8] proposed an energy-efficient distributed SDN controller and AI-driven routing mechanism for large-scale industrial IoT networks, while Mahapatra et al. [9] introduced energy-aware offloading and load balancing across the fog-cloud continuum. They are both model-based and use models that are trained and frozen.

The fog offloading and scheduling have been solved by applying reinforcement learning. DRL is employed in energy-aware scheduling in SDN enabled IoT by Sellami et al. [4] and fuzzy logic is used with RL in vehicular fog offloading by Vemireddy and Rout [10] whereas graph RL is used with asynchronous federated learning by Verma and Naik [5]. Wen et al. [11] used multi-agent DRL for energy efficient workflow scheduling. These techniques are easily adapted but also have the high training cost of deep networks. There are non-learning heuristics available, like the semi-greedy scheduler from Azizi et al. [12] and the latency-constrained allocator from Kopras et al. [13] that are fast, but do not learn from experience. The position of RL-FN is summarised in Table I.

TABLE I
Positioning of RL-FN Against Representative Work

| Ref. | Technique | Adaptive online | Joint delay-energy | Light training |
|---|---|---|---|---|
| [3] | Classical ML routing | No | Partial | Yes |
| [8] | SDN + AI routing | Partial | Yes | No |
| [9] | Energy-aware offloading | No | Yes | Yes |
| [4] | Deep RL scheduling | Yes | Yes | No |
| [5] | Graph RL + federated | Yes | Partial | No |
| [12] | Semi-greedy heuristic | No | Yes | Yes |
| **RL-FN** | **ELM + Q-learning** | **Yes** | **Yes** | **Yes** |

## III. System Model and Problem Formulation

### *A. Network Model*

The system consists of three layers: the IoT end devices, fog tier (fog gateway and heterogeneous fog nodes) and cloud tier. The fog tier is represented by a weighted graph of fog nodes with edges representing wireless or wired connections with known bandwidth and distance. The tasks are published to a device using MQTT/TLS, and the gateway is the learning agent which selects the fog node or another device that will fulfill each task. The delay experienced by a task that has been sent over a link from node i to node j is the sum of four components as shown in (1). The following delays are used for transmission and propagation: (2) L is the packet length; B the bandwidth of the link; d the distance of the link; v the propagation speed.

.

$$D_{ij} = D_{ij}^{tx} + D_{ij}^{prop} + D_j^q + D_j^{proc} \quad (1)$$

$$D_{ij}^{tx} = \frac{L}{B_{ij}}, \qquad D_{ij}^{prop} = \frac{d_{ij}}{v} \quad (2)$$

The fog nodes j are first-in first-out, and are modeled as M/M/1 queues with arrival rate λ and service rate μ. It has a mean queueing delay and utilisation as shown in (3). The processing delay is the computational requirement of the task / node's computational capability.

.

$$D_j^q = \frac{\rho_j}{\mu_j(1-\rho_j)}, \qquad \rho_j = \frac{\lambda_j}{\mu_j} \quad (3)$$

*B. Energy Model*

Radio energy is modeled by the first-order model which is commonly used for IoT studies. The electronics energy is the cost for transmitting a packet over a distance and the cost of receiving it. The amplifier energy varies as a function of distance^2 (4) is the cost of transmitting a packet over a distance. The computation energy is added to the residual energy of node j after round t by (6).

$$E^{tx}(L,d) = L\,E_{elec} + L\,\varepsilon_{amp}\,d^2 \quad (4)$$

$$E^{rx}(L) = L\,E_{elec} \quad (5)$$

$$E_j^{res}(t) = E_j^{res}(t-1) - E^{tx} - E^{rx} - E_j^{proc} \quad (6)$$

*C. Problem Formulation*

Let the hatted delay, energy and response time occurring in round t be the normalised quantities in (7). The routing policy will attempt to minimise their weighted sum over the operating horizon, subject to: (8) each node having energy above some minimum survival threshold, and each queue being stable. Each weight is greater than or equal to 0 and their total is 1.

.

$$\min_{\pi} \sum_{t=1}^{T} \left( w_1 \hat{D}(t) + w_2 \hat{E}(t) + w_3 \hat{A}(t) \right) \quad (7)$$

$$\text{s.t.} \quad E_j^{res}(t) \geq E_{th}, \qquad \rho_j < 1, \quad \forall j,t \quad (8)$$

As only a partial set of link and queue states are observable and they evolve over time, there is no one-shot optimiser, and the problem is formulated as a Markov decision process and solved using reinforcement learning.

## IV. Proposed RL-FN Framework

The entire RL-FN pipeline is depicted in Fig. 1. FogBus2 [14] and SimPy [15] create the fog-IoT environment and its task logs, which are then pre-processed to clean and normalise the logs, enabling ELM to learn a compact representation of the state of the network, and Q-learning to transform the representation into routing decisions that are rewarded based on the QoS that they provide.

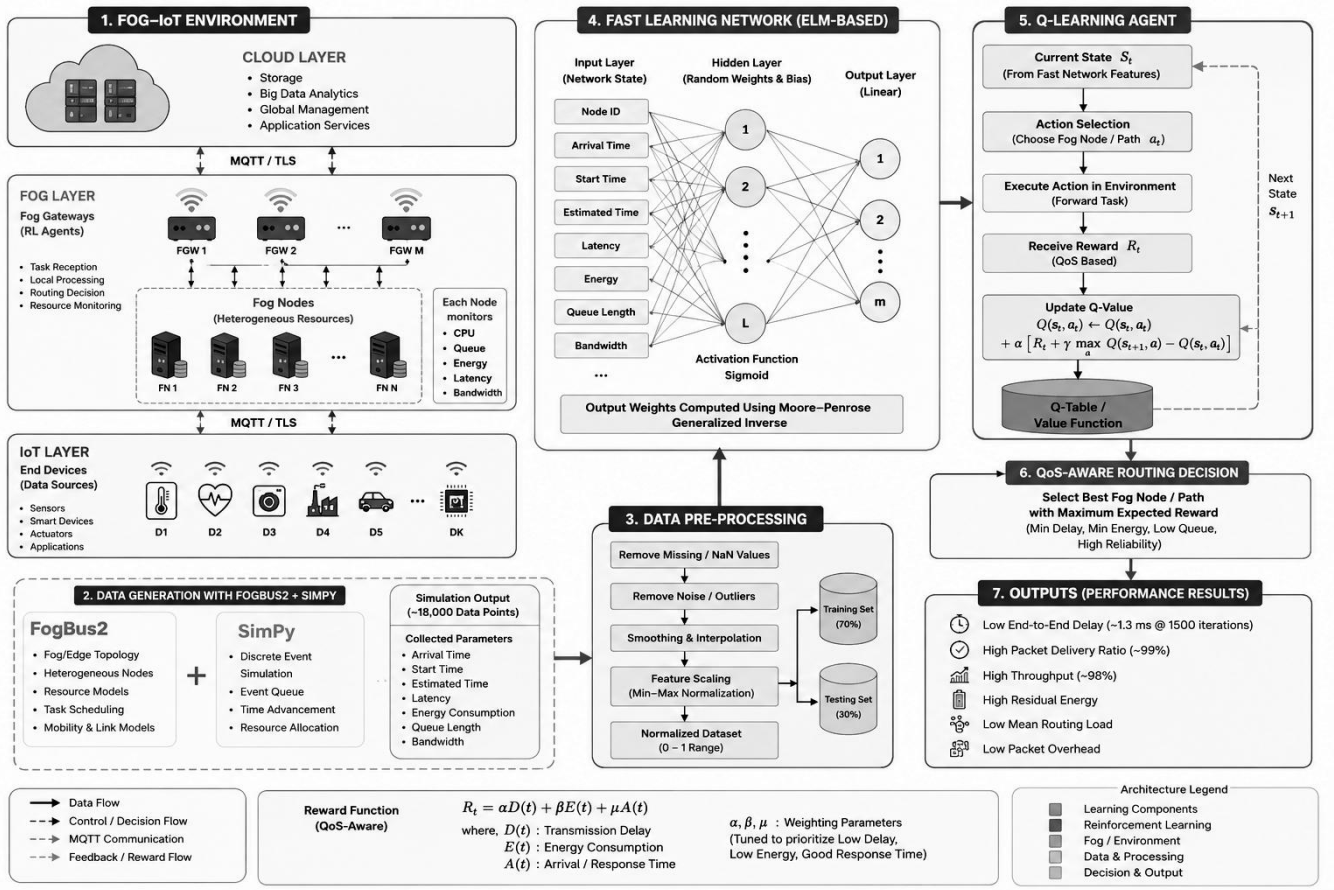



Fig. 1. Complete flow architecture of the proposed RL-FN routing framework (FogBus2/SimPy data generation, pre-processing, ELM-based fast learning network, Q-learning agent and QoS-aware routing decision).

### *A. Data Generation and Pre-processing*

Fifty distributed fog nodes are simulated. For every task the test-bench logs the node identifier, arrival time, start time, estimated processing time, latency, queue length, bandwidth and energy consumed. The sweeping arrival rates and network sizes yields approximately 18,000 records, of which 70% is used for training and 30% is used for testing. The removal of missing and NaN values, outlier filtering, smoothing and interpolating out residual noise, and rescaling all features to the unit interval by (9) are performed. Representative normalised records are given in Table 2.

$$\hat{x} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \qquad (9)$$

TABLE II
REPRESENTATIVE NORMALISED SIMULATION RECORDS

| Task | Fog node | Arrival | Estimated | Latency | Energy |
|---|---|---|---|---|---|
| 1 | 1 | 0.6373 | 0.5633 | 0.4232 | 0.0142 |
| 2 | 2 | 0.4566 | 0.3422 | 0.2319 | 0.0190 |
| 3 | 3 | 0.3242 | 0.5622 | 0.6228 | 0.0182 |
| 4 | 4 | 0.6722 | 0.5435 | 0.3423 | 0.0383 |
| 5 | 5 | 0.5342 | 0.6432 | 0.4442 | 0.0242 |
| 6 | 6 | 0.7332 | 0.3422 | 0.2892 | 0.0022 |

### *B. ELM-Based Fast Neural Network*

This fast network consists of one hidden layer with N sigmoid neurons, and one output layer with a linear activation function. The weights and biases of the neurons are randomly drawn and then fixed, and each neuron's activation is the fixed nonlinear map (10); the network output is the weighed sum (11). Only the output weights are learnt. They can be generated in closed form from the regularised Moore-Penrose solution (12) that stabilises the inversion and enhances generalisation to unseen states by tuning the regularisation constant C.

$$G_i(\mathbf{x}) = \frac{1}{1+e^{-(\mathbf{w}_i^T \mathbf{x}+b_i)}} \quad (10)$$

$$O(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \, G_i(\mathbf{x}) = \mathbf{G}(\mathbf{x})\boldsymbol{\alpha} \quad (11)$$

$$\boldsymbol{\alpha}^* = \mathbf{G}^T \left(\frac{\mathbf{I}}{C} + \mathbf{G}\mathbf{G}^T\right)^{-1} \mathbf{Y} \quad (12)$$

At the fog edge, the ELM can be retrained in a few seconds, as it reduces to a single matrix solve with no gradient iterations needed to be performed online.

*C. Q-Learning Agent and Reward Design*

Agents are each gateway. Its state is the candidate nodes' feature vector normalised to the dimension of its features size, and its actions are the candidate fog nodes or candidate next hops. When an agent forwards a task, she gets reward (13) which is equal to the negative of the weighted and normalised cost (the weights are positive and add up to 1), meaning that the agent wants to minimise (7) to maximise (13). The target and the update for the Q-value are (14) and (15), respectively, with the discount factor γ and the learning rate η.

$$R_t = -\left(\alpha \widehat{D}(t) + \beta \widehat{E}(t) + \mu \widehat{A}(t)\right) \quad (13)$$

$$y_t = R_t + \gamma \max_{a'} Q(s_{t+1}, a') \quad (14)$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta[y_t - Q(s_t, a_t)] \quad (15)$$

The Epsilon Greedy rule is used for exploration. The agent's exploration probability decreases exponentially with the episode number as in (16) and starts exploring broadly in the beginning and then exploits its learnt policy once it has converged. The new Q-values are substituted into the regression targets of (12) to provide the reinforced output (17) and the candidate with the highest reinforced output is selected as the greedy action (18).

$$\epsilon_k = \max(\epsilon_{\min},\ \epsilon_0 \, e^{-\kappa k}) \quad (16)$$

$$O(\mathbf{x}) = \mathbf{G}(\mathbf{x})\mathbf{G}^T \left(\frac{\mathbf{I}}{C} + \mathbf{G}\mathbf{G}^T\right)^{-1} \mathbf{Q}^T \quad (17)$$

$$a_t = \arg \max_{a \in \mathcal{A}(s_t)} O_a(\mathbf{x}_t) \quad (18)$$

*D. Training and Routing Algorithm*

The whole procedure is summarised in Algorithm 1. An example of the decision step is given where a gateway with the reinforced outputs 0.42, 0.71 and 0.55 chooses the second node as it has the highest reinforcement. If both delay and energy cost are low on this hop, then the reward is close to zero (the optimal value), and the value of this node is increased for similar states of the system.

.

**Algorithm 1:** RL-FN energy-efficient routing

**Input:** fog network state, learning rate $\eta$, discount $\gamma$, weights $\alpha$, $\beta$, $\mu$, round limit $M$
**Output:** routing policy and path of maximum cumulative reward
1: Draw random input weights and biases of the ELM; fix them
2: Normalise state features using (9)
3: **while** not converged **and** round < $M$ **do**
  4: Observe the current network state of the fog tier
  5: **for** each candidate fog node **do**
    6: Compute hidden map by (10) and output by (17)
  7: **end for**
  8: Select action by (18) with probability from (16)
  9: Forward task; observe reward by (13)
  10: Compute target (14) and update Q-value (15)
  11: Re-solve output weights in closed form (17)
  12: Decay exploration rate; round ← round + 1
13: **end while**
14: **return** learnt policy and best path

*E. Complexity Analysis*

Inference takes one pass through the hidden layer, as outlined in (19), whereas training the hidden matrix and solving (12) takes the most time. A retrain only takes a few seconds with a sliding window of recent samples, and a DQN must perform thousands of gradient steps (Section VI-H).

$$T_{train} = O(KN^2 + N^3), \qquad T_{inf} = O(N) \quad (19)$$

## V. Experimental Setup

### A. Simulation Environment

The test-bench combines the FogBus2 scheduling semantics with SimPy discrete-event timing. The workload is based on healthcare-style applications. The decision algorithm selects the fog node to which the requests are sent, and the requests are queued in the gateway FIFO and sent in turn to the selected fog node. The configuration is given in Table III and the reward weights are equally biased to latency and energy and are biased to response time as tie-breaker.

TABLE III
Simulation and Learning Parameters

| Parameter | Value |
|---|---|
| Fog / end / cloud devices | 50 / 50 / 10 |
| Applications; tasks per round | 1; 50 |
| Initial energy per fog node | 0.5 J (25 J network) |
| Electronics / amplifier energy | 50 nJ/bit; 100 pJ/bit/m² |
| Packet length | 4000 bits |
| Iterations (rounds) | 300 to 1500 |
| Dataset; train / test split | ~18,000 records; 70% / 30% |
| Hidden neurons; regularisation $C$ | 100; $2^{10}$ |
| Activation (hidden / output) | Sigmoid / linear |
| Learning rate; discount factor | 0.10; 0.90 |
| Exploration (start, floor, decay) | 1.0, 0.05, 0.004 |
| Reward weights | 0.4, 0.4, 0.2 |
| Independent runs per point | 10 |

### B. Baselines and Metrics

RL-FN is compared to Logistic Regression (LR), Support Vector Machine (SVM), K-Nearest Neighbours (KNN), Naive Bayes (NB), Decision Tree, Random Forest (RF), Gradient Boosting, XGBoost, AdaBoost and a simple Artificial Neural Network (ANN) trained on the same pre-processed data set with the same decision interface. As an extension, 2 hidden layers with 64 units and a DQN router with experience replay and a target network are added as an extension of deep-RL as a benchmark. The performance metric used is the average end to end delay (20), PDR (21), normalised throughput (22), mean routing load (23), packet overhead (24) and aggregate residual energy (6).

$$D_{e2e} = \frac{1}{P_{recv}} \sum_{k=1}^{P_{recv}} (t_k^{recv} - t_k^{sent}) \quad (20)$$

$$PDR = \frac{P_{recv}}{P_{sent}} \times 100\% \quad (21)$$

$$Th = \frac{S_{recv}}{T_{sim}\, C_{link}} \times 100\% \quad (22)$$

$$MRL = \frac{P_{ctrl}}{P_{recv}} \quad (23)$$

$$OH = \frac{P_{ctrl}}{P_{ctrl}+P_{data}} \times 100\% \quad (24)$$

## VI. Results and Discussion

All values are means over ten independent runs; the iteration count on the horizontal axes is the number of routing rounds.

### A. End-to-End Delay

The end to end delay is reported in Table IV and Fig. 2. The four classical models degrade increasingly as rounds are added, with NB increasing from 4.4 to 6.9 at 1500 rounds.Ensembles increase slowly, as do the four classical models with AdaBoost going up to 4.0 at 1500 rounds. The delay of the two RL methods are the only ones which decrease as they gain experience. RL-FN falls from 1.8 ms to 1.3 ms, which is 67.5% below AdaBoost and 33.3% below DQN at 1500 rounds. RL-FN beats DQN due to the tracking of the changes in the state as a result of the refits enabled in the closed form, in comparison to the gradient updates in DQN.

TABLE IV
END-TO-END DELAY (MS) VERSUS ITERATIONS

| Model | 300 | 600 | 900 | 1200 | 1500 |
|---|---|---|---|---|---|
| LR | 4.20 | 4.80 | 5.40 | 6.00 | 6.60 |
| SVM | 4.00 | 4.50 | 5.00 | 5.60 | 6.10 |
| KNN | 3.80 | 4.20 | 4.70 | 5.20 | 5.70 |
| NB | 4.40 | 5.00 | 5.60 | 6.30 | 6.90 |
| XGBoost | 3.20 | 3.40 | 3.60 | 3.90 | 4.20 |
| AdaBoost | 3.00 | 3.20 | 3.50 | 3.80 | 4.00 |
| DQN | 2.40 | 2.25 | 2.10 | 2.00 | 1.95 |
| **RL-FN** | **1.80** | **1.65** | **1.50** | **1.40** | **1.30** |

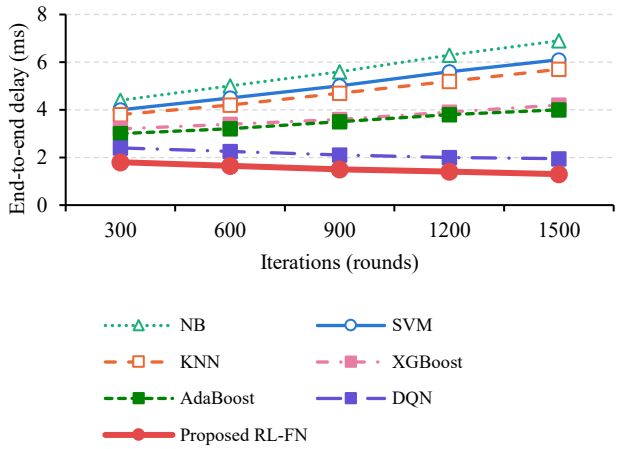



Fig. 2. End-to-end delay versus iterations. RL-FN is the only method other than DQN whose delay decreases with experience, and it stays lowest throughout.

### *B. Packet Delivery Ratio*

PDR is shown in Table. V and Fig.3. NB delivers only 68% to 71% of packets, and SVM 72% to 78%. Tree ensembles do better and AdaBoost achieves a 94% at 1500 rounds. RL-FN achieves 93% at 300 rounds and 99% at 1500 rounds - 5 points (5.3% relative) higher than AdaBoost, and 4 points higher than DQN. The energy and queue terms of the reward give less reward to nodes that are near exhaustion or overflow, causing less traffic to drop.

TABLE V
PACKET DELIVERY RATIO (%) VERSUS ITERATIONS

| Model | 300 | 600 | 900 | 1200 | 1500 |
|---|---|---|---|---|---|
| NB | 68 | 69 | 70 | 71 | 71 |
| SVM | 72 | 74 | 75 | 77 | 78 |
| RF | 80 | 83 | 85 | 87 | 88 |
| XGBoost | 85 | 88 | 90 | 92 | 93 |
| AdaBoost | 86 | 89 | 91 | 92 | 94 |
| DQN | 90 | 92 | 93 | 94 | 95 |
| **RL-FN** | **93** | **95** | **97** | **98** | **99** |

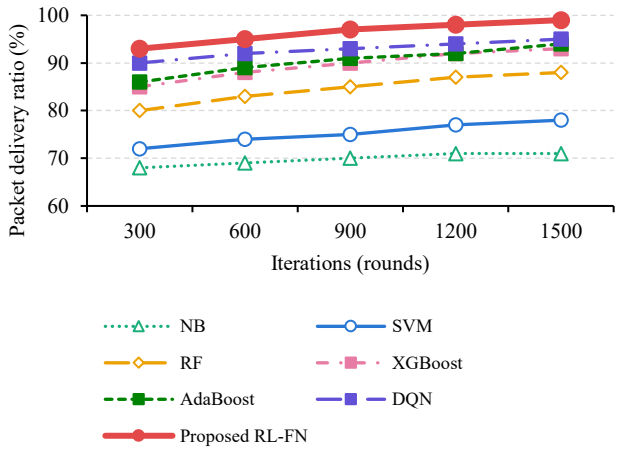



Fig. 3. Packet delivery ratio versus iterations.

*C. Throughput*

The normalised throughput is compared in Fig. 4, and Table VI. In each model, the throughput grows with the stabilisation of the learnt or trained routes, without any changes to the order of them. RL-FN achieves a performance of 98% at 1500 rounds, a relative increase of 10% (11.4%) compared to AdaBoost and 8 points higher than DQN at 1500 rounds. Reducing delay and drops also increases the amount of data that can cross the network in a given period of time.

TABLE VI
THROUGHPUT (%) VERSUS ITERATIONS

| **Model** | **300** | **600** | **900** | **1200** | **1500** |
|---|---|---|---|---|---|
| LR | 54 | 56 | 58 | 60 | 62 |
| SVM | 60 | 63 | 66 | 68 | 70 |
| XGBoost | 76 | 79 | 82 | 84 | 85 |
| AdaBoost | 79 | 82 | 85 | 87 | 88 |
| DQN | 84 | 86 | 88 | 89 | 90 |
| **RL-FN** | **90** | **93** | **95** | **97** | **98** |

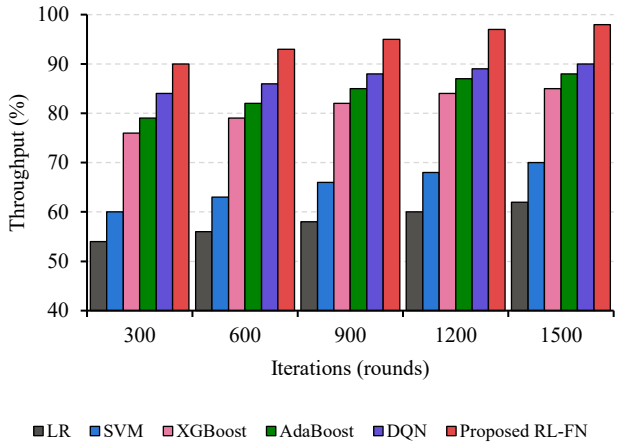



Fig. 4. Average throughput versus iterations.

*D. Residual Energy*

Network lifetime is given as the sum of residual energy over the fog tier. As can be seen in Table VII and Fig. 5, in each round, the energy decreases since each round requires communication and computation energy. RL-FN keeps 11.0 J after 300 rounds and 3.2 J after 1500 rounds, 0.9 J (39.1%) more than XGBoost and 0.6 J (23.1%) more than DQN. This is because the energy component of the reward function distributes load amongst the nodes rather than repeatedly traversing the shortest path.

TABLE VII
AGGREGATE RESIDUAL ENERGY (J) VERSUS ITERATIONS

| Model | 300 | 600 | 900 | 1200 | 1500 |
|---|---|---|---|---|---|
| LR | 4.1 | 2.9 | 2.0 | 1.4 | 0.9 |
| NB | 3.9 | 2.7 | 1.8 | 1.2 | 0.8 |
| RF | 6.8 | 5.2 | 4.1 | 3.0 | 2.0 |
| XGBoost | 7.4 | 5.8 | 4.6 | 3.4 | 2.3 |
| DQN | 8.9 | 6.6 | 5.2 | 3.7 | 2.6 |
| **RL-FN** | **11.0** | **8.4** | **7.5** | **5.6** | **3.2** |

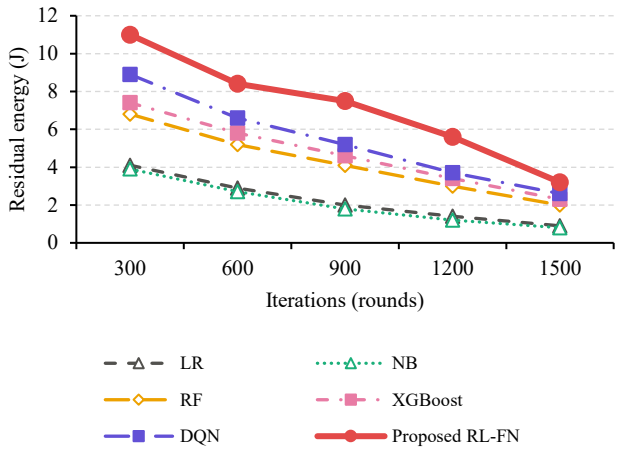



Fig. 5. Aggregate residual energy of the fog tier versus iterations.

*E. Routing Load and Packet Overhead*

The Mean Routing load (Fig. 6) is the number of control packets per data packet delivered. The load of the classical models is increased by the number of rounds and is constant with the frequency of updates (3.3 for NB). RL-FN maintains a very stable value at all times, with a value of close to 1.0, because a converged policy is unlikely to require re-probing the routes. As with Packet overhead (Fig. 7), the picture is identical at 1500 rounds – RL-FN consumes 30% of traffic in the control group, while AdaBoost and DQN consume 46% and 37% respectively. Both are shown in Table VIII for 1500 rounds. As a result, such shifting of the cost does not constitute the data-plane gains that are expected from RL-FN.

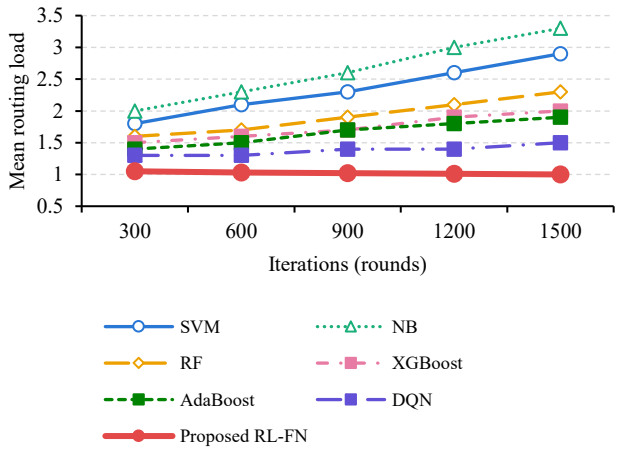



Fig. 6. Mean routing load versus iterations.

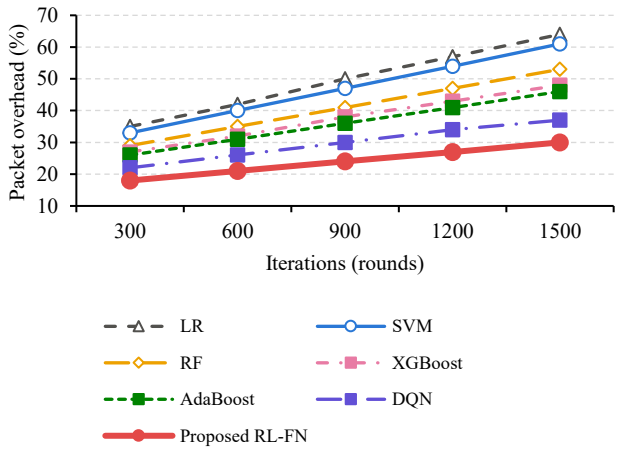



Fig. 7. Packet overhead versus iterations.

TABLE VIII
ROUTING LOAD AND PACKET OVERHEAD AT 1500 ITERATIONS

| Model | Mean routing load | Packet overhead (%) |
|---|---|---|
| LR | 3.1 | 64 |
| SVM | 2.9 | 61 |
| NB | 3.3 | 66 |
| RF | 2.3 | 53 |
| XGBoost | 2.0 | 48 |
| AdaBoost | 1.9 | 46 |
| DQN | 1.5 | 37 |
| **RL-FN** | **1.0** | **30** |

### *F. Scalability*

In order to perform the scalability test, this fog tier is increased from 10 to 100 nodes, and the delay measured after 1500 rounds (Fig. 8). The boosting routers see an increase of ~80% due to the increase in number of candidates that their feature-to-router mapping should cover. RL-FN only increases from 1.10ms to 1.55ms (41%), because the cost of the hidden layer is constant with N not the number of nodes and the evaluation per candidate is a single matrix product. Even if the number of nodes is 100, RL-FN is still 37.5% faster than DQN.

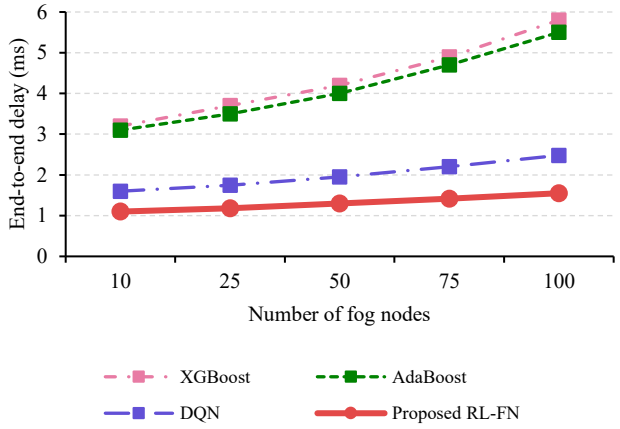



Fig. 8. End-to-end delay at 1500 iterations as the fog tier grows from 10 to 100 nodes.

### *G. Convergence Behaviour*

The moving-average normalised reward defined as 1 + reward of (13) is plotted as a function of training episodes in Fig. 9. RL-FN achieves 95% of its reward by playing around 470 episodes; DQN achieves 95% of its reward by playing around 1100 episodes; and tabular Q-learning does not converge by playing 1500 episodes. The convergence process in fog networks is important because the topology can change before a slow learner has converged.

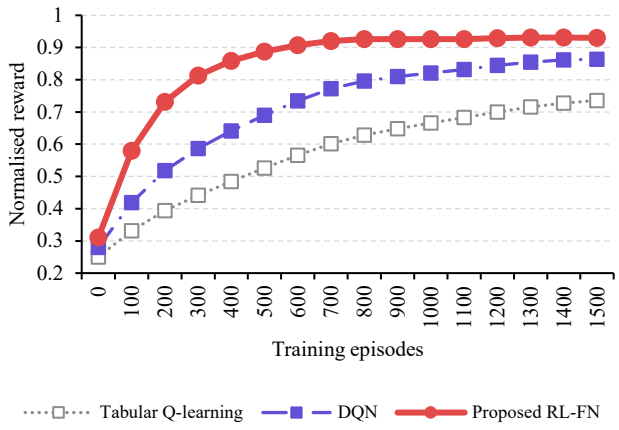



Fig. 9. Convergence of the moving-average normalised reward.

*H. Computational Cost*

Table IX shows the training time for the 12,600-record training split and the mean time to infer on a gateway-class quad core ARM processor. RL-FN trains in an average of 1.6 s per trial, 160 times faster than DQN and 25 times faster than a back-propagation ANN and it decides in 0.05 ms. The practical benefit of the closed form solution to the output weights.

.

TABLE IX
TRAINING AND INFERENCE COST

| Model | Training time (s) | Inference (ms/decision) |
|---|---|---|
| LR | 0.8 | 0.02 |
| SVM | 14.6 | 0.31 |
| RF | 9.2 | 0.45 |
| XGBoost | 6.8 | 0.28 |
| AdaBoost | 5.1 | 0.24 |
| ANN (back-prop.) | 48.3 | 0.19 |
| DQN | 312.5 | 0.62 |
| **RL-FN** | **1.9** | **0.05** |

*I. Ablation and Sensitivity*

Table X eliminates terms of the reward. A delay-only reward will have the shortest delay but will consume energy, an energy-only reward will conserve energy but will have a longer delay and will have less PDR. The joint formulation of (13) is seen to work as desired since the full three-term reward leads to the best balance. Table XI shows the performance of the networks with different numbers of hidden neurons. Delay improves up to 100 neurons, and then gets levelled off as training time continues to increase, using 100 neurons throughout.

TABLE X
REWARD ABLATION AT 1500 ITERATIONS

| Reward terms | Delay (ms) | Residual energy (J) | PDR (%) |
|---|---|---|---|
| Delay only | 1.21 | 2.1 | 98.6 |
| Energy only | 1.92 | 3.6 | 96.1 |
| Delay + energy | 1.34 | 3.1 | 98.8 |
| **Delay + energy + response** | **1.30** | **3.2** | **99.0** |

TABLE XI
SENSITIVITY TO THE NUMBER OF HIDDEN NEURONS

| Hidden neurons | Delay (ms) | PDR (%) | Training time (s) |
|---|---|---|---|
| 25 | 1.62 | 96.8 | 0.4 |
| 50 | 1.41 | 98.2 | 0.9 |
| **100** | **1.30** | **99.0** | **1.9** |
| 150 | 1.29 | 99.0 | 3.6 |
| 200 | 1.31 | 98.9 | 6.1 |

*J. Summary*

Table XII consolidates the comparison at 1500 iterations. RL-FN leads on every metric against the strongest classical baseline and against DQN.

TABLE XII
SUMMARY AT 1500 ITERATIONS

| Metric | RL-FN | Best classical | DQN | Gain vs. classical |
|---|---|---|---|---|
| Delay (ms) | **1.3** | 4.0 (AdaBoost) | 1.95 | 67.5% lower |
| PDR (%) | **99** | 94 (AdaBoost) | 95 | +5 points |
| Throughput (%) | **98** | 88 (AdaBoost) | 90 | +10 points |
| Residual energy (J) | **3.2** | 2.3 (XGBoost) | 2.6 | 39.1% higher |
| Mean routing load | **1.0** | 1.9 (AdaBoost) | 1.5 | 47.4% lower |

## VII. CONCLUSION AND FUTURE WORK

This paper continued the work of the Meta-Reinforcement Fast-Learning Network for routing in the fog environment of IoT. RL-FN approximates the Q-policy with a closed-form, regularised ELM and refines the policy using a normalised multi-objective reward, which allows it to adapt online but not to be too expensive for fog gateways. The reward was based on a formal delay, queueing and energy model and an extended evaluation demonstrated that RL-FN achieves a delay of 1.3 ms with 99% PDR and 98% throughput at 1500 iterations, 39.1% more energy than the best classical router, scales to 100 fog nodes with a 41% increase in delay and trains approximately 160 times faster than DQN. The ablations showed that both reward terms are important, and that 100 hidden neurons are a balance of accuracy and cost. We are planning to deploy RL-FN on physical gateways, add multi-agent cooperation with partial observability to it and integrate it with the meta-heuristic service-placement layer of our companion work to co-optimise routing and placement.

.

## REFERENCES

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.
[2] M. Ghobaei-Arani, A. Souri, and A. A. Rahmanian, "Resource management approaches in fog computing: A comprehensive review," *J. Grid Comput.*, vol. 18, pp. 1–42, 2020.
[3] A. Mehri, "Efficient data transmission in Internet of Things systems based on machine learning methods: A short analysis of algorithms, applications, and challenges," 2025.
[4] B. Sellami, A. Hakiri, S. Ben Yahia, and P. Berthou, "Energy-aware task scheduling and offloading using deep reinforcement learning in SDN-enabled IoT network," *Comput. Netw.*, vol. 210, Art. no. 108957, 2022.
[5] N. K. Verma and K. J. Naik, "Graph reinforcement and asynchronous federated learning based task offloading in fog computing," *Phys. Commun.*, vol. 72, Art. no. 102776, 2025.
[6] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.
[7] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3–4, pp. 279–292, 1992.
[8] P. K. Udayaprasad *et al.*, "Energy efficient optimized routing technique with distributed SDN-AI to large scale I-IoT networks," *IEEE Access*, vol. 12, pp. 2742–2759, 2024.
[9] A. Mahapatra, S. K. Majhi, K. Mishra, R. Pradhan, D. C. Rao, and S. K. Panda, "An energy-aware task offloading and load balancing for latency-sensitive IoT applications in the fog-cloud continuum," *IEEE Access*, vol. 12, pp. 14334–14349, 2024.
[10] S. Vemireddy and R. R. Rout, "Fuzzy reinforcement learning for energy efficient task offloading in vehicular fog computing," *Comput. Netw.*, vol. 199, Art. no. 108463, 2021.

[11] M. Wen *et al.*, "Energy-efficient real-time workflow scheduling in edge-cloud environments using multi-agent deep reinforcement learning," *Appl. Energy*, 2026, doi: 10.1016/j.apenergy.2026.127866.
[12] S. Azizi, M. Shojafar, J. Abawajy, and R. Buyya, "Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: A semi-greedy approach," *J. Netw. Comput. Appl.*, vol. 201, Art. no. 103333, 2022.
[13] B. Kopras, B. Bossy, F. Idzikowski, P. Kryszkiewicz, and H. Bogucka, "Task allocation for energy optimization in fog computing networks with latency constraints," *IEEE Trans. Commun.*, vol. 70, no. 12, pp. 8229–8243, 2022.
[14] Q. Deng, M. Goudarzi, and R. Buyya, "FogBus2: A lightweight and distributed container-based framework for integration of IoT-enabled systems with edge and cloud computing," in *Proc. Int. Workshop Big Data Emergent Distrib. Environ.*, 2021.
[15] D. Zinoviev, "Discrete event simulation: It's easy with SimPy!," arXiv:2405.01562, 2024.
[16] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.
[17] R. Mehta, J. Sahni, and K. Khanna, "Task scheduling for improved response time of latency-sensitive applications in fog integrated cloud environment," *Multimedia Tools Appl.*, vol. 82, pp. 32305–32328, 2023.
[18] F. Jazayeri, A. Shahidinejad, and M. Ghobaei-Arani, "A latency-aware and energy-efficient computation offloading in mobile fog computing: A hidden Markov model-based approach," *J. Supercomput.*, vol. 77, no. 5, pp. 4887–4916, 2021.