Journal of
Sensor and Actuator Networks

Article

Optimising Performance for NB-IoT UE Devices through Data Driven Models

Omar Nassef [1,*], Toktam Mahmoodi [1], Foivos Michelinakis [2], Kashif Mahmood [3] and Ahmed Elmokashfi [2]

1 Center of Telecommunication Research, Kings College London, London WC2R 2LS, UK; toktam.mahmoodi@kcl.ac.uk
2 Center for Digital Engineering, Simula Metropolitan, 0167 Oslo, Norway; foivos@simula.no (F.M.); ahmed@simula.no (A.E.)
3 Telenor Research, 1360 Fornebu, Norway; kashif.mahmood@telenor.com
* Correspondence: omar.nassef@kcl.ac.uk

Abstract: This paper presents a data driven framework for performance optimisation of Narrow-Band IoT user equipment. The proposed framework is an edge micro-service that suggests one-time configurations to user equipment communicating with a base station. Suggested configurations are delivered from a Configuration Advocate, to improve energy consumption, delay, throughput or a combination of those metrics, depending on the user-end device and the application. Reinforcement learning utilising gradient descent and genetic algorithm is adopted synchronously with machine and deep learning algorithms to predict the environmental states and suggest an optimal configuration. The results highlight the adaptability of the Deep Neural Network in the prediction of intermediary environmental states, additionally the results present superior performance of the genetic reinforcement learning algorithm regarding its performance optimisation.

Keywords: NB-IoT; internet of things; reinforcement learning; gradient descent; genetic algorithm; deep learning; machine learning

Citation: Nassef, O.; Mahmoodi, T.; Michelinakis, F.; Mahmood, K.; Elmokashfi, A. Optimising Performance for NB-IoT UE Devices through Data Driven Models. *J. Sens. Actuator Netw.* **2021**, *10*, 21. https://doi.org/10.3390/jsan10010021

Received: 25 January 2021
Accepted: 26 February 2021
Published: 5 March 2021

1. Introduction

Narrow-Band Internet of Things (NB-IoT) [1] is an Internet of Things (IoT) [2] mobile communication technology, which focuses on Low Powered Wide Area Networks (LPWANs) bringing a low cost and power efficient means of communication. Recent advances in NB-IoT catering to energy efficiency and battery longevity have been introduced, such as extended Discontinuous Reception (eDRX) and Power Saving Modes (PSM) [3]. However, there remain many areas to reduce the energy consumption and improve other Quality of Services (QoSs) such as throughput or delay. The overhead stems from the inheritance of protocols from Long-Term Evolution (LTE), which traditionally focused on Human Type Communication (HTC), and can lead to costly overhead when applied in a Machine Type Communication (MTC) environment [4].

One of the main features of NB-IoT is providing coverage enhancement for devices that are hard to reach for the network. Which is achieved by combining repetitions in order to reach a higher effective Signal-to-Noise Ratio (SNR) [5]. This is realised through clustering the User Equipment (UE) into different coverage groups. The coverage group is directly correlated with the received power, where the higher the received power the lower the repetition count.

Although, NB-IoT is designed for battery longevity and reliable connectivity, current research for NB-IoT continues to explore energy consumption, coverage enhancements and latency reduction [6]. Many works have highlighted major deviations of energy assumptions that were published in ref. [7] using standard configurations [8]. As such, applying off the shelf configurations, that affect the energy efficiency such as Tracking Area Updating (TAU) or active timer, which plays a direct role on PSM, may be counter-intuitive.

This leads to a need for bespoke optimisation of pre-configured parameters for the UE and its application, to ensure battery longevity and performance optimality [9]. Further areas to improve the QoS are related to the traffic nature of the application, for instance payload size.

Therefore, this paper explores the employment of data driven models to select the optimal values for packet size and active timer for UEs, in order to achieve the best possible outcome in regard to delay, throughput and energy consumption or a combination of the above. The presented Configuration Advocate (CA) utilises a micro-service architecture [10] which is exhibited and analysed in detail. Reinforcement Learning (RL) is used by the CA to recommend configuration changes to the UE with respect to its application, network and device parameters. Network and application related variables that stem from both the UE and Base Station (BS) are taken into account by the RL, such as coverage, signal strength, packet size and interval [11]. Additionally, two styles of RL are explored: Genetic Algorithm and Gradient Descent. The main contributions of this paper encompass:

- A CA, that delivers bespoke configurations to UEs depending on device, network and application parameters.
- Analysis of environmental prediction models: Machine Learning and Deep Learning with single and multiple output predictions.
- Performance evaluation of two different RL approaches: Gradient Descent and Genetic Algorithm for configuration suggestions.

A plethora of use case are dependant of the adoption of IoT UE as the core of their operation, with the aim of maximising health and safety, minimising the amount of work needed due to its adoption convenience and reducing costs [12]. IoT use cases can be grouped into four categories: transport and logistics such as [13], healthcare, smart environments and personal and social related scenarios [14]. With that in mind, this work specifically targets the reduction of costs in IoT centered use cases, which would be extremely beneficial for NB-IoT scenarios that require communication at a extended range with a relatively low cost and energy consumption, such as smart utility metering, smart cities (parking, lighting, waste management) and agriculture, to name a few.

When compared to the machine learning approaches this paper undertook, the Deep Neural Network (DNN) showcased a superior performance at predicting the metrics, when presented with the configuration set by a factor of 12%. As a result, the RL models could incorporate the environment prediction in its learning phase through the DNN. The Genetic Algorithm incorporated RL approach outperformed the RL model with Gradient Descent at its core through a 10% increase of successful suggestions, albeit taking longer to converge. That being said, the actual metric improvement brought by the configuration suggestion leaned towards the RL with Gradient Descent approach with a minuscule improvement of 3.6% over the Genetic Algorithm infused RL.

The remainder of this paper is structured as follows, Section 2 provides the state of the art and literature review assessing the different approaches taken to improve communication and device performance. Section 3 describes the design of the data-driven models and the CA architecture. A performance analysis is carried out in Section 5. Finally, a conclusion is presented in Section 6.

2. Literature Review

NB-IoT is not a standalone technology, most of the architecture is inherited from LTE, in both Down-link and Up-link, allowing the coexistence of NB-IoT with fourth and fifth generation communication technologies [15].

The focus of NB-IoT is placed on coverage, battery life and device complexity [16]. NB-IoT is more tailored to improving the coverage of transmission for the devices placed in extreme conditions, by repeating the transmission of the packet. There are three different Radio-Frequency (RF) classes in which the repetition count is determined: Normal (N), Robust (R) and Extreme (E). Measurements provided by the power received on the NB-IoT UE determine its class. The BS assigns the class to the UE, to increase the coverage and likelihood of packet success. The extreme RF class is designed to overcome remote areas where coverage conditions are rough [17].

NB-IoT devices operate in two states: connected or idle. Whilst connected, the device exchanges data. When the device is in an idle state, cell re-selection, Down-link paging and Random Access (RA) are carried out [18]. These states are influenced by timers which are directly utilised by energy saving strategies such as eDRX and PSM [19]. The PSM is impacted by both the TAU Timer (T3412) and Active Timer (T3324) as it is the difference between them. The Active Timer is the time the UE should remain reachable after transitioning to the idle state, whilst, TAU Timer is the extended time for the UE to send a periodic TAU.

In ref. [20], PSM, and eDRX values have been explored in order to surmise their effect on the power consumption in smart cities. The authors arrive at the consensus that an artificial RF band would be beneficial in reducing energy consumption in energy-sensitive applications albeit sacrificing throughput. This could be beneficial in remote areas where an increased consumption is a by product of the additional repetitions [21].

A mathematical model is presented in ref. [22] to optimise the up-link of numerous devices connected at different distances from the BS, an analysis for the maximum throughput is carried out, with the success probabilities of the packets given random configurations. The paper concluded that the random configurations are more expensive in terms of resources, as such a bespoke set of configurations are needed to increase the throughput and up-link by orders of magnitude. Additionally, an analysis of transmission latency and reliability is presented in ref. [23], showcasing mathematically the performance of up-link and its detriment to up-link sensitive applications.

A variety of deep learning RL architectures have been readily incorporated as tools to facilitate support in exploding data traffic, to provide assistance in improving a range of QoSs [24]. Work in ref. [25], incorporated RL models, namely Q-Based Deep Neural Network (DNN-Q) and tabular-Q networks, to optimise the up-link of connected NB-IoT devices in real time. The results suggest the use DNN-Q achieves a higher performance with less training time, serving NB-IoT devices whilst reducing communication clashes on the network.

Optimisation carried out on the BS side is considered in ref. [26], where the overall network transmission efficiency is increased through the use of RL, focusing on devices placed at various distances. However, the authors note that the RL model seems to favour immediate rewards, and as such the transmission efficiency of the network may not be guaranteed in the long-term. RL is further utilised, aiming to optimise power ramping and preamble picking in order to improve the energy efficiency of NB-IoT systems [27], shedding light on the inadequacy of RA with respect to energy consumption.

The packet delivery rate and the energy consumption of NB-IoT UE were predicted in ref. [28]. The model achieved a very high accuracy in the prediction of those two variables. Although, the dataset used [29] contained a few pre-select variables, meaning that the data the model was trained on did not reflect the entire spectrum of NB-IoT communication. In comparison to optimising either the UE or BS side, authors in ref. [30] adopt a flavor of Genetic Algorithm to reduce energy consumption in networks, focusing on the architectural design on the network rather than specific configurations.

This paper aims to provide an optimisation framework which incorporates RL models to allow QoS optimisation from the UE perspective opposed to the BS. This places more importance on the UE application and device enabling operators to cater to their different needs. On top of the careful selection of timer values on the UE, for the reduction of energy consumption, this paper goes further, by optimally suggesting the best packet size value for the UE application to adopt in order to serve alternative QoSs concurrently.

3. Architecture and System Model

This section describes the CA, which hosts various data-driven models. The CA obtains measurements from the BS and the UEs and generates configurations that enhances performance metric(s) on the UE. The performance metrics focused in this work are delay, energy consumption, throughput and a combination of all three.

3.1. Scenario

We consider a scenario that is presented in Figure 1. In this scenario, we assume the architecture supports distributed cloud, and consists of edge-cloud nodes. Where arbitrary UEs communicate with the edge-cloud nodes that house a CA, to receive a tailored configuration. When a connection is established, the CA queries the UE through the use of Over-the-Air (OTA) "AT" commands about its measurements and configurations, from signal strength to timer values. The CA generates an optimised configuration with the help of the data driven models and sets them on the UE with "AT" commands. After-which, the UE can carry out its purpose.

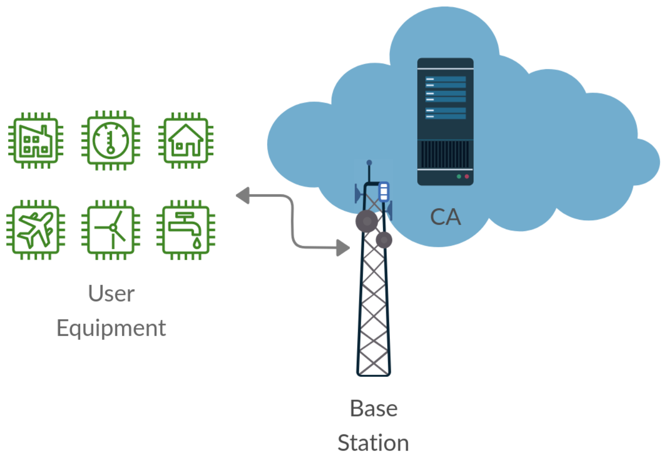


Figure 1. UE devices connects to edge-cloud node housing a CA to obtain a tailored configuration.

On the edge-cloud nodes, a micro-service architecture is utilised to house the implementation this work undertakes, as depicted in Figure 2. Specifically, in the CA micro-service, the Network and Detection Interface is responsible for receiving and sending communication to and from each UEs. Both interfaces act as an intermediary that parses configurations to and from OTA "AT" commands. The Exchange Interface, has the task to instantiate a UE abstraction within the CA, saving the configurations obtained from the Detection Interface in order to store and update the UE configurations, together with the ability of being utilised by the models layer. Additionally, the Exchange Interface allows arbitrary adoption of algorithms and models at the higher layers of the micro-service through a consistent set of communication modes, making way for ease of model changes and updates.

Docker is the micro-service manager enabling connectivity across independent containers forming a scalable solution. The Logging micro-service houses an Elastic, Logstash and Kibana (ELK) stack, implemented to monitor and log the CA interactions in real-time, in an easy to search platform. The ELK stack is a set of well integrated open source software components designed for this purpose: Logstash for data collection, Elasticsearch to store and to query data and Kibana for data visualisation. The stack allows for another means of online data collection, to periodically train the models on the CA. Thus, ensuring that the models utilised by the different edge-cloud nodes at BSs can be specialised to their UE interaction. The ELK stack also serves as a tool to measure network interaction between the CA and the UEs.

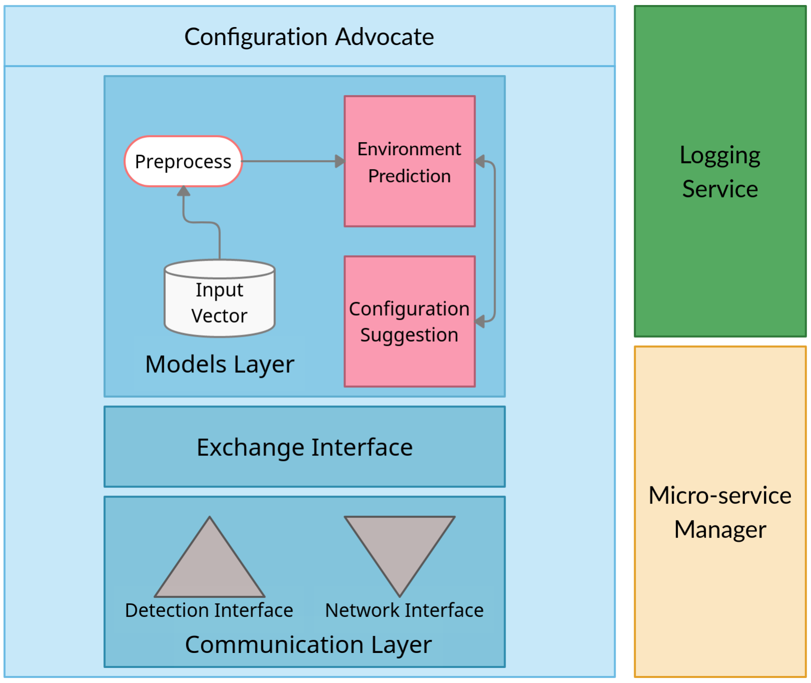


Figure 2. Proposed micro-services architecture which resides in the edge-cloud nodes.

3.2. Data Collection and Manipulation

The dataset used in this work was collected using two UEs, namely: u-blox SARA-N211-02B [31] and Quectel BC95-G [32]. Which are commercial off-the-shelf LTE Cat NB1 UEs. A power measurement device (Otii Arc [33]) was equipped onto the UE for noting the energy consumption, by specifically measuring the voltage and current, under different conditions and configurations. The UEs are connected to two commercial networks located in Norway. The network operators implemented guard-band to reduce interference between NB-IoT and LTE UEs.

By emulating normal, robust and extreme coverage classes on the UEs with the use of attenuators, a sparse dataset with instances covering the broad state space can be collected. This allows a more encompassing data-driven model, that has observed the domain breadth. A variety of packet sizes were chosen to cover different payload sizes that may arise from an application: 12,20,128,256 and 512 bytes. For example, in a smart metering use case, the range of sizes stemmed from 64 to 512 bytes [19]. In our previous work, measurements were collected from a real-world scenario, where the collection spanned over a year, and close to 1.3×10^4 instances were collected [34].

The dataset is split into two unique subsets: $Dataset_{Base}$ and $Dataset_{Equipment}$. $Dataset_{Equipment}$ extends $Dataset_{Base}$ with the addition of readings from power measurement device (Otii Arc) for voltage and current readings on the UE. Whereas, $Dataset_{Base}$ instances focus on Round Trip Time (RTT), packet loss, SNR, BS set timers (active and TAU), packet intervals, payload sizes, UE reported energy consumption, Reference Signal Received Quality (RSRQ) and RRC Connection and Release events. The sizes of the dataset differ greatly, where $Dataset_{Base}$, contains 10^4 instances compared to 1100 for $Dataset_{Equipment}$. Due to the different features in each dataset and the changing requirements during the data collection phase, the metrics modeled by each dataset also vary, $Dataset_{Base}$ models both energy consumption and delay, where as $Dataset_{Equipment}$ expands with the addition of throughput.

In order to ensure that the data-driven models proceed with a smoother learning experience, the dataset needs to be cleaned. As such, the interquartile range was used in order to define outliers, specifically removing the box-plot lower and upper whiskers. After-which z-score was used in order to sieve the dataset removing forced configurations as a result of UE and BS parameter negotiation, which wildly contrasted the measurements recorded for similar configurations.

The definition of Z-score is given by:

$$Z\text{-}score = \frac{x - \mu}{\sigma} \tag{1}$$

Further manipulation is carried out on the dataset, where the dataset is normalized by utilising MinMaxScaler [35], to be in the range of [0, 1] inclusive. This specific range aids model learning, specifically for use in Neural Network (NN) activation functions such as Linear activation function. Constraining the range of values in the dataset for an activation function increases both accuracy and convergence speed, the activation function used in this work is discussed further in Section 4. For inference, the same scaling needs to be implemented on the CA detection interface received parameters to ensure correct input values for the models. The MinMaxScaler is given by:

$$x\_scaled = \frac{(x - \min)}{(\max - \min)} \times (\max - \min) + \min \tag{2}$$

4. Data-Driven Models

The data-driven models serve two purposes. Firstly, it allows a simulation of the environmental state space, predicting the affect of different configurations on the performance metrics. Secondly, aids the configuration suggestion process by learning and broadly generalising the state space, thus configurations are unhindered by any sampling bias that stems from the dataset.

4.1. Environment Prediction

In order to provide a stateful environment for the RL model to generalise efficiently and correctly assign the most optimal configuration, the performance metrics have to predicted with respect to the changing configurations. Deep Learning (DL), supervised and unsupervised learning are explored to achieve environment prediction. Specifically, focusing on Random Forest, K-nearest Neighbours, Decision Tree, Stochastic Gradient Descent, Logistic Regression, Naive Bayes, Perceptron, Linear Regression and DNN. Note that scikit-learn is used for the adoption of Machine Learning (ML) algorithms, whereas, Pytorch is reserved for the DL approach. Both frameworks allow the exporting of the trained models to be employed in the CA.

The architecture of the DNN was designed to provide a regressive model with the least number of layers and neurons reducing computational complexity. From Table 1, there are 3 hidden layers, each increasing the number of neurons with a factor of 2, the layers are coated with a Rectified Linear Unit (RelU) activation function. The RelU function was chosen over other functions such as the linear or sigmoid functions for its ability to become monotonic with its derivative and its ability to deal with non-linearity within the layers. Since any negative input given to the function may result in a zero value, the input features have been normalised beforehand, to conform to such constraints.

Table 1. DNN architecture for $Dataset_{Base}$ and $Dataset_{Equipment}$.

Type	Input ($n \times d$)	Output ($n \times d$)
Linear	$1 \times d$	$d \times 64$
Linear	$d \times 64$	64×128
Linear	64×128	128×256
Linear	128×256	256×512
Linear	256×512	512×3

4.2. Configuration Suggestion

This work employs RL as the main driver of generalising the state space for the sake of suggesting optimal configurations. A novel approach is explored, by incorporating RL models as individuals for a genetic algorithm approach. This method will be compared with the traditional gradient descent infused RL to ascertain its feasibility. Training the RL models will take a mini-batch approach, in order to inject a sufficient amount of noise into the data with each update yet maintaining a speedy convergence.

4.2.1. Reward Function, Loss Function and Actions

The reward function is a critical deciding factor on how the model behaves. For maximum optimality, the reward function must codify improvement of the metrics for both short and long term. To reduce complexity, the standardised values of both the features and labels serve as a basis of the reward. Since, the *MinMaxScaler* scales the features proportional to their value, the rewards can be assigned as follows, where ec is energy consumption, d is delay and tp is throughput:

$$r(x) = \begin{cases} r_{ec} = 1 - state_{x+1}(x).ec & \text{if metric is energy consumption} \\ r_d = 1 - state_{x+1}(x).d & \text{if metric is delay} \\ r_{tp} = state_{x+1}(x).tp & \text{if metric is throughput} \\ \frac{1}{3}(r_{ec} + r_d + r_{tp}) & \text{if metric is combination in } Dataset_{Equipment} \\ \frac{1}{2}(r_{ec} + r_d) & \text{if metric is combination in } Dataset_{Base} \end{cases} \quad (3)$$

Using a step function would be inadvisable in this scenario, as the model will not be receiving any rewards until the terminal condition is met, leading the model to traverse a large state space with zero indication on performance, inevitably leading to a longer convergence time. Therefore, the reward function is shaped asymptotically to guide the agent towards to the highest reward possible in between the terminal states.

Cases presented in Equation (3) ensure that the reward is positive and between the range of $[0-1]$. The positive allocation of rewards gives the agent a certain degree leniency when suggesting a configuration, for a gentler convergence. Opposing to a negative reward allocation, which rushes the agent to reach the most minimal reward, leading to a slower convergence and a poorer ability to generalise when adopted in a real-world environment. When a combination of performance metrics is adopted, the reward function enforces an average weighting scheme, evenly placing importance on each of the metrics, the reward function could be further evolved by placing more importance on an individual metric whilst still catering for the other metrics.

In both the energy consumption and the delay optimisation, the goal is to achieve the lowest possible values for those metrics and as such, the value of the metric in the next state is inversely proportional to the reward the agent receives, hence the incentive for the agent is to decrease the value of these metrics. For the case of the throughput, the aim is to increase the metric value as much as possible and as such, the reward is left as the value obtained from the next state.

The actions suggested in this work address a large number of UE devices with different capabilities, without requiring external equipment. The active timer (T3324) is usually set by the BS, however, the operator can receive a local configuration suggested by the UE device. Since the active timer directly affects the PSM, optimally selecting the timer would alter the UE energy consumption. On the other hand, the packet size has a more encompassing effect, affecting transmission time and transport block size. Hence, the actions are as follows: Nothing, Increase Active Timer, Decrease Active Timer, Increase Packet Size, and Decrease Packet Size.

The NN loss function calculates the difference from the true value and the predicted value, which is important to determine the performance of the model during training. There are numerous loss functions, and each pertain to individual scenarios. This work adopts SmoothL1Loss [36] due to its ability to adapt to features with unexpected values, stemming from different coverage conditions, UEs and BS. SmoothL1Loss is less sensitive to outliers and exploding gradients in addition to removing unnecessary amplification of large losses. This loss function is given by:

$$\mathcal{L}(x,y) = \begin{cases} 0.5(x-y)^2, & \text{if } |x-y| < 1 \\ |x-y| - 0.5, & \text{otherwise} \end{cases} \tag{4}$$

4.2.2. Exploration versus Exploitation

Exploration and exploitation is a crucial aspect that concerns RL model quality and generalisability. The exploitation phase, capitalises on the learnt state space to obtain the highest reward possible, whereas, exploration phase probes the search state for a more favourable outcome. Implementing the correct ratio of exploration to exploitation is essential to ensure a well-rounded model that can generalise to different scenarios. Therefore, an epsilon greedy approach is utilised. An initial large epsilon value at the start of training is applied for a larger exploration probability. The epsilon value gradually decreases as the training progresses, decreasing the probability of exploration. Random actions governs the exploration of the state space.

Conversely, the Genetic Algorithm does not require additional measures to explore the state space. This is solely based on the inherent nature of the Genetic Algorithm itself. As the solution space is traversed owing to offspring creation with different Deep Q-Based Network (DQN) weights every generation. The exploitation phase is proportional to the number of generations the Genetic Algorithm undertakes until convergence, where the later generations cultivate a better performing model that exploits the state space effectively.

4.2.3. Gradient Descent

Gradient Descent is typically applied to RL at each iteration of model training to increase performance. The architecture proposed in Table 2 pertains to a Dueling Deep Q-Network (Dueling-DQN) [37], where d is the dimension of the input variables described in Section 3.2. The input given by $(n \times d)$ array, represents the shape of the input parameters from the UE device. The input features are mapped to a maximum of 2^8 neurons through linear functions. The advantage and value layers both project to an array of 5 elements which corresponds to the actions available. Additionally, the layers have a unique purpose where each action is mapped with the "value" in the environment, assessing the "advantage" of undertaking the chosen action. A minimum number of neurons was chosen, through trial and error, for the Dueling-DQN mindful of the computational overhead utilising a complex architecture incurs.

Table 2. Dueling-DQN architecture for $Dataset_{Base}$ and $Dataset_{Equipment}$.

Architecture	**Type**	**Input $(n \times d)$**	**Output $(n \times d)$**
Feature	Linear	$1 \times d$	$d \times 256$
	RelU	-	-
Advantage	Linear	$1 \times d$	$d \times 256$
	RelU	-	-
	Linear	$d \times 256$	1×5
Value	Linear	$1 \times d$	$d \times 256$
	RelU	-	-
	Linear	$d \times 256$	1×5

4.2.4. Genetic Algorithm

Contrasting Gradient Descent, adopting Genetic Algorithm theoretically guarantees arrival at a global optimum. The key components of a Genetic Algorithm approach consists of selecting, mating and finally mutating the individuals. The Genetic Algorithm employs

8 individuals as an initial population. The individuals correspond to RL models shown in Table 3. DQN individuals reduces the complexity of the algorithm and running time to achieve a faster convergence, in lieu of using a Dueling-DQN. Each individual is initialised with a random weighting and bias to be evolved in subsequent generations, which play a key role in the behaviour of the model. Xavier uniform Weighting [38] is adopted to allocate random weights to the model, Equation (5), due to the utilisation of uniform distributions. n_{in} and n_{out} is number of neurons in and leaving the layer, respectively:

$$var(w) = \frac{2}{n_{in} + n_{out}} \tag{5}$$

Table 3. RL architecture for Genetic Algorithm for $Dataset_{Base}$ and $Dataset_{Equipment}$.

Type	Input ($n \times d$)	Output ($n \times d$)
Linear	$1 \times d$	$d \times 64$
ReLU	-	-
Linear	$d \times 64$	64×5

The loss set for an individual in a generation is their fitness score. The Natural Selection rate is based on the individual fitness score and is placed at 50%. A single-point cross-over is adopted for parent mating, where a point in the weights of each layer is chosen and swapped between the mother and father. This method along with the random selection of parents is chosen to reduce computational complexity. Lastly, elitism is implemented, preventing the highest ranking individual from being mutated. The number of mutated weights to change is shown in Equation (6), where N_{pop} is the population number, μ is the mutation rate placed at 0.01. μ was selected as a low number to deter a large change in the individual and favour exploitation rather than exploration. Since the weights of the DQN are very sensitive, an increase in mutation may render a significant change in the individuals causing a domino affect for later generations, incurring a longer convergence time. The actual change to the weights of the DQN model is applied with a Gaussian function, Equation (7), in order to keep the newly mutated values within an acceptable range of the pre mutated values, to reduce any unexpected behaviour within the individual.

$$N_{neurons}\ for\ mutation = \mu * (N_{pop} - 1) * N_{neurons} \tag{6}$$

$$neuron_{new} = \frac{1}{\sigma\sqrt{2\pi}} \exp \frac{-1}{2} (\frac{neuron_{old} - \mu}{\sigma})^2 \tag{7}$$

5. Results

The results are split into two subsections. The first subsection ascertains the performance of environmental state prediction, with respect to the variety of machine and deep learning algorithms. Whereas, the second, determines the feasibility of utilising Genetic Algorithm and Gradient Descent RL for optimal configuration suggestion. The datasets are further split into three smaller subsets: training, testing and validation subsets, with 70%, 20% and 10% of the dataset size, respectively. The training subset is used to learn the model weightings, the testing subset serves as unseen data instances to determine the accuracy of each model, and finally the validation subset is used to fine-tune the machine learning hyper parameters for a more accurate model.

5.1. Prediction of Environmental States

The prediction of intermediary environmental states has a direct impact on the utility of the RL algorithm, as it facilitates the ability of the RL algorithm to learn different intermediary states through simulation.

The machine learning models being used to predict the single and multi performance metrics are presented in Table 4 with respect to the testing data subsets. The majority of the algorithms preformed exceptionally well obtaining an accuracy approximately greater than 99%, excluding Gradient Boosting and Linear Regression. This proves that the dataset can be predicted with a variety of models without the fear of over fitting. Although decision trees performed exceptionally well with the single metric prediction, the ensemble of decision trees curated by gradient boosting under-performed. However, in the multi metric prediction, the gradient boosting approach managed to obtain a relatively high accuracy in-line with the other approaches. It is worth noting, that the perceptron obtained a very high accuracy in all the single metric predictions, which can hint at the ability of a multi layer perceptron or a deeper model such as DNN, to be able to accurately predict the combination of metrics which motivated the use of DNN in this work.

The single performance metric prediction can be employed to create an ensemble method for combined metric predictions which has been shown to provide a high accuracy as seen from the gradient boosting approach. However, it may suffer to generalise as different UE devices and BS are introduced into the state space. This could be especially troubling with forced configurations set from the BS.

Table 4. Accuracy of machine learning models for single and combined performance metrics prediction with respect to $Dataset_{Base}$ and $Dataset_{Equipment}$.

Machine Learning Model	**Testing Accuracy**						
	$Dataset_{Base}$			$Dataset_{Equipment}$			
	Delay	**Energy Consumption**	**Combination**	**Delay**	**Energy Consumption**	**Throughput**	**Combination**
Random Forest	100	100	95	100	100	100	83
Decision Tree	100	100	95	100	100	100	83
K-neighbours	100	100	95	100	100	100	71
Gaussian	99.87	99.2	-	100	100	100	-
Perceptron	99.98	99.98	-	100	100	99.7	-
Linear SVC	99.98	99.98	-	100	100	99.77	-
Logistic Regression	99.98	99.98	-	99.7	99.7	99.7	-
Gradient Boosting	0	0	96	0	0	0	81
Linear Regression	0	0	84	0	0	0	50

Taking the multi-metric prediction further, a DNN is adopted, where the accuracy of the DNN is plotted against the iteration number for both data subsets are shown in Figure 3. The DNN demonstrates superior accuracy that both the data subsets obtained, surpassing the accuracies of ML multi performance metric prediction algorithms, attaining an accuracy of 99.8% for $Dataset_{Base}$, and 95% for $Dataset_{Equipment}$ on the testing dataset. The DNN surpassed the highest $Dataset_{Base}$ ML approach by 3.9%, but the difference is more stark with the $Dataset_{Equipment}$ reaching 12%. Since the DNN exceeds the accuracy of all the models recorded previously for the prediction of multi performance metrics, it is adopted to facilitate learning for the RL model.

5.2. Optimal Configuration Suggestion

The comparison of Gradient Descent and Genetic Algorithm for configuration suggestion is explored through the SmoothL1Loss, reward each model achieves for the performance metrics, percentage of successful configurations and the performance improvement between the default configuration with respect to the suggested configuration. This is to gain a holistic evaluation of the configuration suggestion approach that not only focuses on final model but its training performance as well.

The final RL model is also evaluated on the testing data for each data subset to indicate its performance, in the case of Genetic Algorithm the best individual is adopted as the model to be tested. The reward allocated to the model indicates the quality of suggested configuration speaking to its performance based on Equation (3). Whereby, a larger reward is proportional to the performance improvement on the UE.

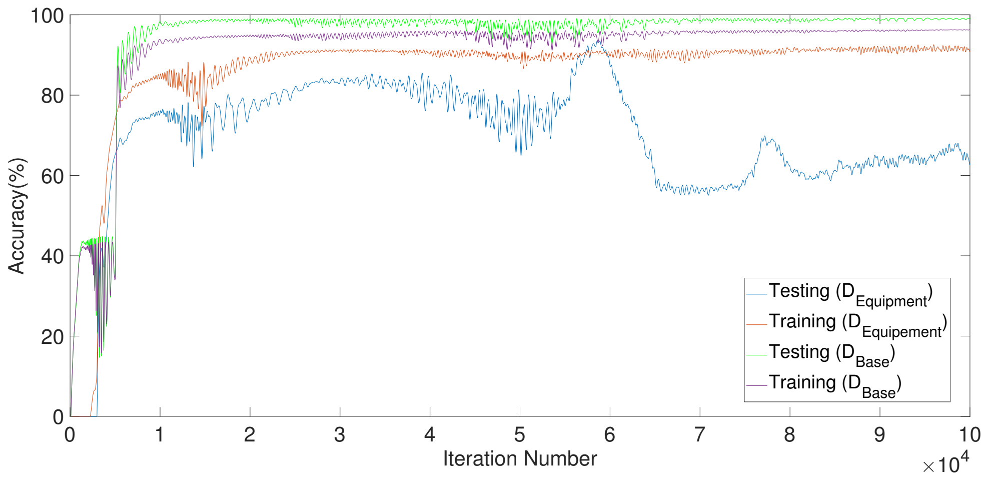


Figure 3. Training and testing accuracy of DNN predicting combined performance metrics for $Dataset_{Base}$ and $Dataset_{Equipment}$ during training phase.

Table 5 depicts the SmoothL1Loss obtained for each model, with the iteration number of its convergence concerning each data subset. It is immediately notable that the $Dataset_{Equipment}$, converged at a much lower number of iterations that its counter part. This can be attributed to the inequality of the subset sizes. Nevertheless, the convergence of the models is not sparse, reaching the same approximate optimum for all the performance metrics. The loss attained by the RL Genetic Algorithm against the number of generation, is presented in Table 6 for both data subsets and catering to all the metrics. An immediate inspection of the loss concludes that the Genetic Algorithm obtained a much higher loss than that of the Gradient Descent counter part by a factor of 10^{-3}. The number of generation the Genetic Algorithm needed to converge is somewhat misleading, as each generation contained a population of agents placed in the simulated environment to learn, which roughly equates to 10^4 instances in the Gradient Descent approach. The best loss had a min of 10^{-7}, which is given by the $Dataset_{Base}$ delay, though by a negligible margin. It can be seen that the $Dataset_{Equipment}$ had a more consistent loss at convergence for all the metrics compared to $Dataset_{Base}$ which fluctuates at the converged loss, this hints at the stability of values in $Dataset_{Equipment}$ compared to $Dataset_{Base}$, which may make the $Dataset_{Equipment}$ a more attractive option for deploying the RL model.

Table 5. Iteration number convergence of RL with Gradient Descent for single and combined performance metrics suggestion on $Dataset_{Base}$ and $Dataset_{Equipment}$.

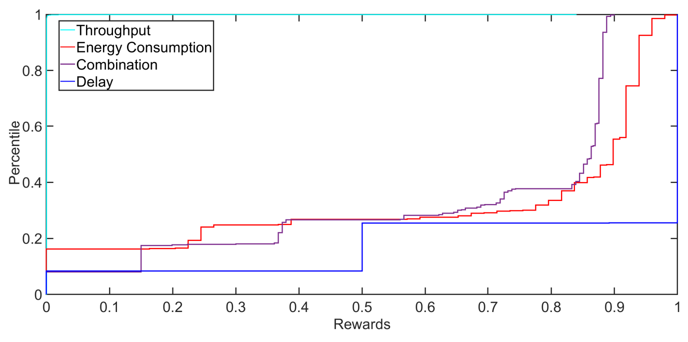
Data Subset	Performance Metric	SmoothL1Loss	Iteration Number
Base	Delay	7.36×10^{-11}	1×10^{6}
	Energy Consumption	2.01×10^{-10}	1×10^{6}
	Combination	1.33×10^{-10}	1×10^{6}
Equipment	Delay	1.09×10^{-9}	4×10^{5}
	Energy Consumption	9.23×10^{-10}	4×10^{5}
	Throughput	2.54×10^{-10}	4×10^{5}
	Combination	2.91×10^{-10}	4×10^{5}

Table 6. Generation convergence of RL with Genetic Algorithm belonging to the highest fitness scored model for single and combined performance metrics suggestion for $Dataset_{Base}$ and $Dataset_{Equipment}$.

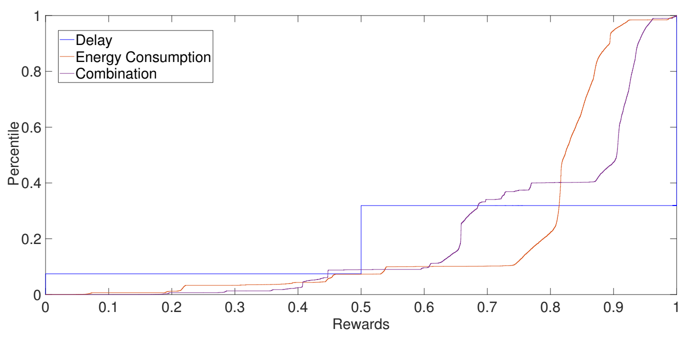
Data Subset	Performance Metric	SmoothL1Loss	Generation
Base	Delay	7.44×10^{-7}	30
	Energy Consumption	2.93×10^{-5}	86
	Combination	7.86×10^{-7}	10
Equipment	Delay	3.24×10^{-5}	27
	Energy Consumption	1.47×10^{-5}	3
	Throughput	2.12×10^{-5}	25
	Combination	3.22×10^{-5}	7

Figures 4 and 5 show the Empirical Distribution Function (ECDF) of rewards obtained by the most optimal model with respect to the performance metrics for both data subsets. The delay metric reward resembles a step function, reaching a reward of 1 about 70% of the time as observed in Figure 4b, suggesting that the model has found the optimal set of actions to undertake. Contrarily, the energy consumption metric, displays a continuous like function where the model performed well in obtaining a reward of greater than 0.5 90% of the time, there is a sharp incline in the rewards obtained between the ranged of 0.8 to 0.9 accounting for 85% of the rewards obtained. As for the multi performance metrics, the ECDF indicates that 50% of the rewards stem from the range 0.2 to 0.9 and the other half of the rewards obtained a reward greater than 0.9. Figure 4a unsurprisingly shows the same reward structure. This is attributed to the metric representation in both subsets.

However, it is obvious that the $Dataset_{Base}$ performed better, as shown by the larger concentration of rewards located at the higher end of the reward spectrum. That being said, the throughput metric, showed a reward of 0 suggesting that the model assigned every action incorrectly, even though, it was evident that the model converged with the same approximate loss as the other metrics in the same subset. The $Dataset_{Base}$ models outperformed the $Dataset_{Equipment}$ models in every performance metric consolidating the importance of the dataset size regarding Gradient Descent for the models effectiveness and ability to generalise. It is intriguing to note that the rewards obtained by the RL Genetic Algorithm (Figure 5) is almost identical to the rewards obtained by the RL Gradient Descent method in Figure 4. In this regard, RL Genetic Algorithm resulted in a more optimal model, mimicking the rewards achieved by the Gradient Descent approach, as well as achieving a better performance for the throughput metric as seen in Figure 4a, where roughly 70% of the configuration suggestion resulted in the highest reward.

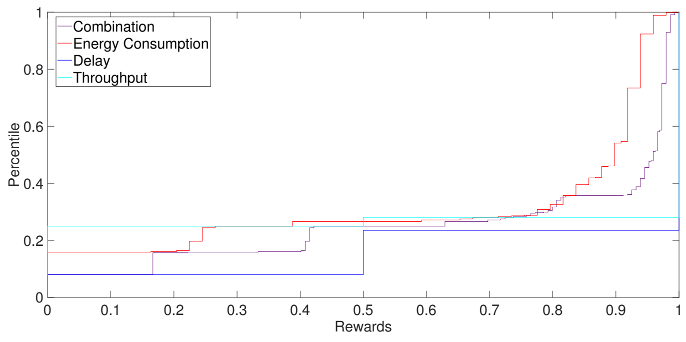


(**a**) Reward allocated for $Dataset_{Equipment}$ metrics

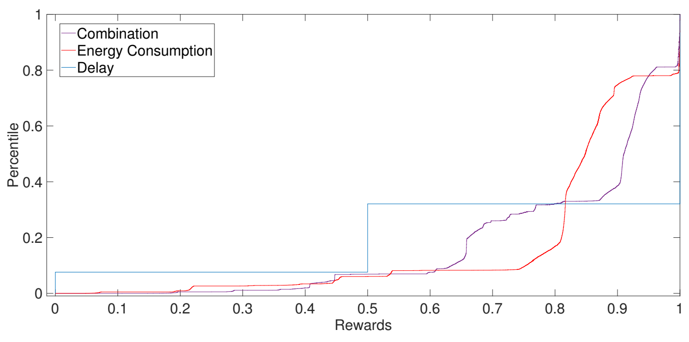


(**b**) Reward allocated for $Dataset_{Base}$ metrics

Figure 4. ECDF of reward for single and combined performance metric suggestion for $Dataset_{Base}$ and $Dataset_{Equipment}$ utilising the most optimal RL with Gradient Descent.



(**a**) Reward allocated for $Dataset_{Equipment}$ metrics



(**b**) Reward allocated for $Dataset_{Base}$ metrics

Figure 5. ECDF of rewards for single and combined performance metric suggestion for $Dataset_{Base}$ and $Dataset_{Equipment}$ utilising the most optimal RL individual with Genetic Algorithm.

Finally, the accuracy of the RL models with the performance improvement percentage is shown in Table 7 with respect to the testing data subset. The performance difference between Gradient Descent and Genetic Algorithm is almost negligible concerning $Dataset_{Base}$, although the accuracy of successful suggestions leans towards the Genetic Algorithm approach by a minuscule factor of 0.2% if the sum of average performance improvement is taken as the deciding factor. That being said, the actual metric improvement increase is dominated by the Gradient Descent for both data subsets by an average of 3.6%. On the other hand, the percentage of successful suggestions were superior in Genetic Algorithm with a difference of 10%. What is intriguing in this set of results, is the comparable performance of the two approaches albeit the adoption of a simpler DNN architecture for Genetic Algorithm. This can pave a way for creation of RL architectures with less computational complexity without sacrificing model accuracy.

Table 7. Successful improvement and suggestion percentage shown for every performance metric with its respective data subset and the RL approach taken. Where the successful suggestion percentage stems from the suggested configuration compared to the original on the test dataset.

RL Mechanism	Data Subset	Metric	Successful Suggestion (%)	Average Performance Improvement (%)
Gradient Descent	*Base*	Delay	93.7	13.02
		Energy Consumption	96.2	7.9
		Combination	92.6	19.1
	Equipment	Delay	56.1	6.8
		Energy Consumption	57.5	7.7
		Throughput	39.4	28.6
		Combination	43.6	13.9
Genetic Algorithm	*Base*	Delay	94.7	12.0
		Energy Consumption	95.2	9.0
		Combination	97.7	11.4
	Equipment	Delay	47.2	10.5
		Energy Consumption	76.6	6.7
		Throughput	64.4	3.1
		Combination	33.1	19.1

Therefore, gathering the results from the optimisation suggestion, with a special focus on $Dataset_{Base}$, since the models flourished better in that data subset, the Genetic Algorithm approach outperformed the Gradient Descent approach. This is a combination of the increased convergence speed, the performance improvement of the metric compared to the default configuration and the number of successful suggestions. Although, the Gradient Descent approach obtained better rewards during training phase, the Genetic Algorithm approach obtain consistent performance in terms of successful suggestions coupled with the percentage metric performance between the default configuration and the Genetic Algorithm suggested configuration.

6. Conclusions

In this paper, we have presented a Configuration Advocate that utilises data driven models to select the optimal values for packet size and active timer for UEs, to achieve the best possible delay, throughput and energy consumption or a combination of the above. The RL Genetic Algorithm model has been identified as the most optimal approach compared to Gradient Descent, obtaining a higher successful suggestion by 10%, although, experiencing a negligible drop in metric improvement of 3.6% and a larger SmoothL1Loss. Additionally, the feasibility of adopting DL was put into question contrasting between the accuracy of DNN and various ML models to predict the environment state space, the results showcased the accuracy advantage of DNN with multi-metric prediction of 12% over the best performing ML model. Future work needs to be carried out to explore the use of a more sophisticated RL algorithm as the building block for the Genetic Algorithm approach to fully comprehend the limitations of Genetic Algorithm infused RL in a real world application. In addition, future work will revolve around utilising the Genetic Algorithm infused RL to provide configurations from the BS perspective, for the range of UEs that the BS encounters, to harmonize with this work and cover optimisation at both ends of the communication channel.

Author Contributions: Conceptualization, O.N., T.M. and K.M.; methodology, O.N. and T.M.; software, O.N.; data curation, F.M.; writing—original draft preparation, O.N.; writing—review and editing, O.N., T.M., F.M., K.M. and A.E.; visualization, O.N.; supervision, T.M., K.M. and A.E. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by EPSRC INITIATE project, a UK national scale programmable Internet testbed. Additionally, the work is partially supported by the European Community through the 5G-VINNI469 project (grant no. 815279) within the H2020-ICT-17-2017 research and innovation program.

Data Availability Statement: The data presented in this study are available on request from an author at Simula Research Laboratory.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sinha, R.S.; Wei, Y.; Hwang, S.H. A survey on LPWA technology: LoRa and NB-IoT. *ICT Express* **2017**, *3*, 14–21. [CrossRef]
2. Fortino, G.; Russo, W.; Savaglio, C.; Viroli, M.; Zhou, M. Modeling opportunistic IoT services in open IoT ecosystems. *CEUR Workshop Proc.* **2017**, *1867*, 90–95.
3. Non-Access-Stratum (NAS) Protocol for Evolved Packet System (EPS). 3GPP 24.301. Available online: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1072 (accessed on 20 February 2020).
4. Jermyn, J.; Jover, R.P.; Murynets, I.; Istomin, M.; Stolfo, S. Scalability of Machine to Machine systems and the Internet of Things on LTE mobile networks. In Proceedings of the 2015 16th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), Boston, MA, USA, 14–17 June 2015; pp. 1–9.
5. Harwahyu, R.; Cheng, R.; Tsai, W.; Hwang, J.; Bianchi, G. Repetitions Versus Retransmissions: Tradeoff in Configuring NB-IoT Random Access Channels. *IEEE Internet Things J.* **2019**, *6*, 3796–3805. [CrossRef]
6. Qiu, T.; Wang, X.; Chen, C.; Atiquzzaman, M.; Liu, L. TMED: A Spider-Web-Like Transmission Mechanism for Emergency Data in Vehicular Ad Hoc Networks. *IEEE Trans. Veh. Technol.* **2018**, *67*, 8682–8694. [CrossRef]
7. NB-LTE - Battery Lifetime Evaluation. 3GPP 151393. Available online: https://www.3gpp.org/DynaReport/TDocExMtg--RP-69--31198.htm (accessed on 20 February 2020).

8. Yeoh, C.Y.; bin Man, A.; Ashraf, Q.M.; Samingan, A.K. Experimental assessment of battery lifetime for commercial off-the-shelf NB-IoT module. In Proceedings of the 2018 20th International Conference on Advanced Communication Technology (ICACT), Chuncheon, Korea, 11–14 February 2018; pp. 223–228.
9. Pirayesh, H.; Sangdeh, P.K.; Zeng, H. EE-IoT: An Energy-Efficient IoT Communication Scheme for WLANs. In Proceedings of the IEEE Conference on Computer Communications (INFOCOM), Paris, France, 29 April–2 May 2019; pp. 361–369.
10. Alshuqayran, N.; Ali, N.; Evans, R. A systematic mapping study in microservice architecture. In Proceedings of the 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), Macau, China, 4–6 November 2016; pp. 44–51.
11. Alfakih, T.; Hassan, M.M.; Gumaei, A.; Savaglio, C.; Fortino, G. Task Offloading and Resource Allocation for Mobile Edge Computing by Deep Reinforcement Learning Based on SARSA. *IEEE Access* **2020**, *8*, 54074–54084. [CrossRef]
12. Mocnej, J.; Pekar, A.; Seah, W.; Zolotova, I. *Network Traffic Characteristics of the IoT Application Use Cases*; Technical report series; School of Engineering and Computer Science, Victoria University of Wellington: Wellington, New Zealand, 2018.
13. Nassef, O.; Sequeira, L.; Salam, E.; Mahmoodi, T. Building a Lane Merge Coordination for Connected Vehicles Using Deep Reinforcement Learning. *IEEE Internet Things J.* **2021**, *8*, 2540–2557. [CrossRef]
14. Atzori, L.; Iera, A.; Morabito, G. The internet of things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805. [CrossRef]
15. Wang, Y.E.; Lin, X.; Adhikary, A.; Grovlen, A.; Sui, Y.; Blankenship, Y.; Bergman, J.; Razaghi, H.S. A primer on 3GPP narrowband Internet of Things. *IEEE Commun. Mag.* **2017**, *55*, 117–123. [CrossRef]
16. Wang, H.; Fapojuwo, A.O. A Survey of Enabling Technologies of Low Power and Long Range Machine-to-Machine Communications. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2621–2639. [CrossRef]
17. Feltrin, L.; Tsoukaneri, G.; Condoluci, M.; Buratti, C.; Mahmoodi, T.; Dohler, M.; Verdone, R. Narrowband IoT: A Survey on Downlink and Uplink Perspectives. *IEEE Wirel. Commun.* **2019**, *26*, 78–86. [CrossRef]
18. Moon, Y.; Ha, S.; Park, M.; Lee, D.; Jeong, J. A Methodology of NB-IoT Mobility Optimization. In Proceedings of the Global Internet of Things Summit (GIoTS), Bilbao, Spain, 4–7 June 2018; pp. 1–5.
19. Martinez, B.; Adelantado, F.; Bartoli, A.; Vilajosana, X. Exploring the Performance Boundaries of NB-IoT. *IEEE Internet Things J.* **2019**, *6*, 5702–5712. [CrossRef]
20. Jörke, P.; Falkenberg, R.; Wietfeld, C. Power Consumption Analysis of NB-IoT and eMTC in Challenging Smart City Environments. In Proceedings of the 2018 IEEE Globecom Workshops (GC Wkshps), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6.
21. Liu, P.; Wu, K.; Liang, J.; Chen, J.; Tseng, Y. Energy-Efficient Uplink Scheduling for Ultra-Reliable Communications in NB-IoT Networks. In Proceedings of the 29th IEEE Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Bologna, Italy, 9–12 September 2018; pp. 1–5.
22. Feltrin, L.; Condoluci, M.; Mahmoodi, T.; Dohler, M.; Verdone, R. NB-IoT: Performance estimation and optimal configuration. In Proceedings of the 24th European Wireless Conference, Catania, Italy, 2–4 May 2018; pp. 1–6.
23. Li, H.; Chen, G.; Wang, Y.; Gao, Y.; Dong, W. Accurate Performance Modeling of Uplink Transmission in NB-IoT. In Proceedings of the 24th IEEE International Conference on Parallel and Distributed Systems (ICPADS), Singapore, 11–13 December 2018; pp. 910–917.
24. Zhang, C.; Patras, P.; Haddadi, H. Deep Learning in Mobile and Wireless Networking: A Survey. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 2224–2287. [CrossRef]
25. Jiang, N.; Deng, Y.; Nallanathan, A.; Chambers, J.A. Reinforcement Learning for Real-Time Optimization in NB-IoT Networks. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1424–1440. [CrossRef]
26. Chen, L.; Chung, W.; Chen, I.; Kuo, S. Adaptive Repetition Scheme with Machine Learning for 3GPP NB-IoT. In Proceedings of the 23rd IEEE Pacific Rim International Symposium on Dependable Computing (PRDC), Taipei, Taiwan, 4–7 December 2018; pp. 252–256.
27. Guo, Y.; Xiang, M. Multi-Agent Reinforcement Learning Based Energy Efficiency Optimization in NB-IoT Networks. In Proceedings of the IEEE GLOBECOM Workshops, Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6.
28. Ateeq, M.; Ishmanov, F.; Afzal, M.K.; Naeem, M. Multi-parametric analysis of reliability and energy consumption in IoT: A deep learning approach. *Sensors* **2019**, *19*, 309. [CrossRef] [PubMed]
29. Fu, S.; Zhang, Y. CRAWDAD Dataset Due/Packet-Delivery (v. 2015-04-01). 2015. Available online: https://crawdad.org/due/packet-delivery/20150401/delay (accessed on 20 February 2020).
30. Goudos, S.K.; Deruyck, M.; Plets, D.; Martens, L.; Psannis, K.E.; Sarigiannidis, P.; Joseph, W. A Novel Design Approach for 5G Massive MIMO and NB-IoT Green Networks Using a Hybrid Jaya-Differential Evolution Algorithm. *IEEE Access* **2019**, *7*, 105687–105700. [CrossRef]
31. SARA-N2 Series Power-Optimized NB-IoT (LTE Cat NB1) Modules Data Sheet. Available online: https://www.u-blox.com/sites/default/files/SARA-N2_DataSheet_(UBX-15025564).pdf (accessed on 20 February 2020).
32. Quectel BC95-G Multi-Band NB-IoT Module with Ultra-Low Power Consumption. Available online: https://www.quectel.com/UploadFile/Product/Quectel_BC95-G_NB-IoT_Specification_V1.9.pdf (accessed on 20 February 2020).
33. Otii Arc. Available online: https://www.qoitech.com/otii/ (accessed on 11 November 2020).
34. Michelinakis, F.; Al-Selwi, A.S.; Capuzzo, M.; Zanella, A.; Mahmood, K.; Elmokashfi, A. Dissecting Energy Consumption of NB-IoT Devices Empirically. *IEEE Internet Things J.* **2021**, *8*, 1224–1242. [CrossRef]

35. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
36. Berrada, L.; Zisserman, A.; Kumar, M.P. Smooth loss functions for deep top-k classification. *arXiv*, **2018**, arXiv:1802.07595.
37. Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; Freitas, N. Dueling network architectures for deep reinforcement learning. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1995–2003.
38. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; pp. 249–256.