The University of British Columbia

Advanced Machine Learning Tools for Engineers
EECE 571T

Enhancements to SmartPUR for Mobile C-IoT Devices using Advanced Machine Learning Techniques - Group 3

Author: Flynn Dowey
Author: Nikhileswar Kota

December 14, 2023

1 Goal of Report

Our objectives are twofold:

1. **Enhanced Model Performance:** The first goal of this project is to improve upon the results reported in [8] using alternative machine learning models and to introduce additional features which were overlooked in [8]. This approach is intended to optimize predictive accuracy and model robustness.

2. **Innovative Architecture Proposal:** Secondly, we propose two novel architectures that address the core challenges identified in [8]. These models will be sequential machine learning models, designed to provide a more robust and effective solution to the problem identified in [8].

2 Previous work done

Establishing connections in Cellular Internet-of-Things (C-IoT) networks involves a random access procedure (RAP). During this procedure, the user equipments (UEs) in the network compete with one another for necessary bidirectional communication resources. Generally, a standard contention-based RAP consists of four steps. It mandates at least four message exchanges between the UE and the base station (BS) before a connection can be established [1] and [6].

To minimize this operational overhead in C-IoT networks, early data transmission (EDT) has been introduced. EDT allows simultaneous transmission of short uplink data along with RAP messages [4]. The implementation of EDT reduces the network latency by three seconds and extends the battery life of C-IoT UEs by as much as three years [4].

The advantages of EDT are further improved by preconfigured uplink resources (PURs). With PURs, C-IoT UEs are allocated resources in advance, which eliminates the need for RAP message exchanges when initiating uplink transmissions [5]. However, one of the prerequisites for utilizing PURs is that the UE must apply a valid timing advance (TA) for its transmission. This adjustment ensures that the UE aligns its signal transmission with the propagation delay between the UE and the BS, preventing potential interference with adjacent resources used by other UEs. Therefore, each UE must validate its TA before initiating a PUR transmission.

While validating TAs is a straightforward task for stationary UEs, mobile UEs face challenges due to the dynamically changing environment, frequently making the previously assigned TAs invalid and requiring a new TA request through legacy RAP procedures. This limits the widespread adoption of PURs among mobile UEs, limiting the technology's potential.

To overcome this drawback and to promote the universal adoption of PURs across all mobility scenarios, an innovative machine-learning solution for TA validation and prediction in UEs was introduced in [8], known as smartPUR, where the sTAV and sTAP methods were proposed for managing uplink resources in C-IoT devices. However, initial results for sTAV and sTAP reveal performance gaps in larger cell-size scenarios, such as those extending over 5 kilometres. Sensitivity for sTAV and fallback rates for sTAP in these settings require further improvement.

To address these challenges, our work aims to enhance the performance of smartPUR. By employing advanced machine learning methods, our project aims to improve the results of smartPUR, allowing for the universal adoption of PURs across various C-IoT scenarios.The paper will be organized into two sections, described below:

The paper will be organized into two sections, described below:

1. **Non-Sequential Case**:
    - Use classification to determine if the UE's last allocated TA value is valid based on a set of stationary features.
    - Use prediction to estimate the new TA time for the UE based on a set of stationary features.
2. **Sequential Case**:
    - Use prediction to estimate the new TA time for the UE based on a sequence of features.

3 Strategy

In our project, we will use three distinct cell sizes each having a radial distance of 1.5km, 2.5km and 5km. Each cell is represented by a Euclidean ball in $\mathbb{R}^2$. Let the set $C_r \triangleq \{x \in \mathbb{R}^2 : ||x||_2 \leq r\}$ represent a cell of radial distance of r in km. For simplicity, we will only present the results based on the 5km cell as this is the most challenging case. All data for this project will be simulated using MATLAB.

In the initial phase, we will simulate a User Equipment (UE) moving stochastically within a cell. During this movement, two sets of power measurements will be recorded at times t and $t+1$. Denote the set of power measurements at time t to be $\mathcal{P}_t$, where $\mathcal{P}_t$ is represented as a tuple $\mathcal{P}_t \triangleq \{P^{(t)}_{RSRP}, P^{(t)}_{RSSI}, P^{(t)}_{RSRQ}\}$. The abbreviations in $\mathcal{P}$ mean the following: RSRP represents the Reference Signal Received Power (P_{RSRP}), RSSI represents the Received Signal Strength Indicator (P_{RSSI}), and RSRQ represents the Reference Signal Received Quality (P_{RSRQ}) [7]. In addition to measuring $\mathcal{P}_t$ and $\mathcal{P}_{t+1}$, we will calculate the timing advance (TA) time at time t, TA_{old}, using the formula outlined in [8]. The features for classification and prediction will be represented by the tuple $X_{ns} \triangleq \{\Delta P_{RSRP}, \Delta P_{RSSI}, TA_{old}\}$, where $\Delta P_{(\cdot)} = P^{(t+1)}_{(\cdot)} - P^{(t)}_{(\cdot)}$. It is important to note that P_{RSRQ} is not considered in this phase. The label for *prediction* will be the actual TA value at $t+1$, TA_{new}, which is again calculated using the formula in [8]. The label for *classification* will be based on a threshold: if $\Delta TA = |TA_{old} - TA_{new}| \leq 8$ then the old TA is valid. Our dataset consists of 300,000 training samples and 60,000 testing samples [1]. Refer to algorithm 1 and 2 for pseudocode of the task described above.

Algorithm 1 Predict TA

```
function predictTA(data)
    TA_new ← Model(data)
    return TA_new
end function
```

Algorithm 2 Classify TA

```
function classifyTA(data)
    isValid ← Model(data)
    return isValid
end function
```

[1] In fact, the dataset consists of a concatenation of different path loss models. A path loss model is a mathematical formula or algorithm describing signal strength's attenuation or loss as it propagates through space.

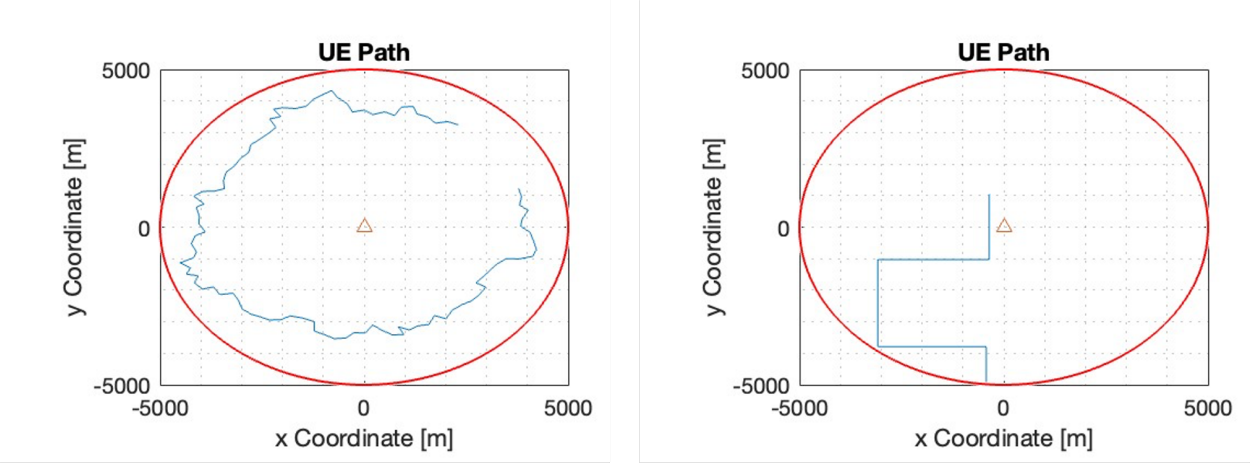

Figure 1: Sample trajectories, clockwise (left) and Manhattan (right).

In the second stage of our project, we will modify the UE's movement to follow four distinct trajectory patterns: Manhattan, Clockwise, Counterclockwise, and Radial (see figure 1). Instead of measuring $\mathcal{P}_t$ and $\mathcal{P}_{t+1}$, a series of 10 measurements will be taken over equally spaced sample intervals. In this phase, the difference in measurements will not be considered and the features now become $X_s \triangleq \{TA_{old}^{(i)}, P_{RSRP}^{(k)}, P_{RSSI}^{(k)}, R_{RSRQ}^{(k)}\}$ for $k = 1, 2, \ldots 10$ and $i = 1, 2, \ldots, 9$. The label will be $TA_{old}^{(10)}$, this is indicated by the separate use of indices in X_s, i.e., we only take the previous nine TA values but we are allowed to measure power as the UE moves. For simplicity, we neglect the task of classification for the sequential case. Algorithm 1 represents the task of the sequential case, where $data = X_s$.

The results we obtain in the non-sequential case will be compared to the models used in [8]. In fact, the models presented in [8] will be trained and tested on the same data we simulate. For the sequential case, we have no benchmarks to compare to as this is a novel technique.

4 What Methods Worked

4.1 Pseudo-code Description

We use four ML models in the classification task and three in the prediction task. The optimizer used in all cases was Adam.

4.1.1 Classification: MLP

(a) The MLP is a traditional ML model dating back to 1958 that has been revived by deep learning. We only test the feasibility of our problem with this model.

(b) We use the following architecture

(i) Input layer & Hidden layers 1 - 4: 100 neurons

(ii) Hidden layer 5: 64 neurons

(iii) Output layer: 1 neuron

(c) We use ReLU between each layer of the network. The loss function was BCEWithLogitsLoss as it is more numerically stable than using a plain Sigmoid followed by a BCELoss. The is because PyTorch combined the two operations into one layer and used the log-sum-exp trick for numerical stability [9].

4.1.2 Classification: Random Forest

(a) The random forest is an ensemble method which creates a 'forest' of decision trees. For classification, the output of the random forest is the class selected by most trees. The RF we consider in our project has a linear layer attached to the output of the leaf nodes which performs logistic regression.

(b) We consider 100 trees in the forest with an arbitrary depth, i.e. the tree nodes are expanded until all leaves are pure or until all leaves contain less than the number of samples [11].

4.1.3 Classification: Random Tree Ensemble

(a) A Random Tree Ensemble is an unsupervised learning task. Its purpose is to transform a dataset to a high-dimensional sparse representation. The features are coded according to which leaf of each tree it is sorted into. This leads to a binary coding with as many ones as there are trees in the forest [10].

(b) Refer to section 4.1.2 (b) for model parameters.

4.1.4 Prediction: MLP

(a) The architecture is identical to what was used in section 4.1.1. The exception is the use of mean squared error as the loss function as this is a prediction problem.

4.1.5 Prediction: LSTM

(a) RNNs are used to model sequential ML problems. The LSTM is an attempt to improve the memory issues found in the original RNN architecture. Within the LSTM cell there are three layers: forget gate, input gate and output gate. Refer to [3] or [13] for a more rigorous discussion.

(b) We use an input size of 4, i.e., the number of features is 4, the hidden size is 10 and the number of layers is 1. We remove the 10th value of TA, $TA_{old}^{(10)}$, from the datasets as this is the item the network needs to predict.

(c) The real part of the features will be passed through one LSTM and the imaginary parts through another. The output of each LSTM is summed and placed through an MLP. Refer to figure 2; T_x is the sequence length and $x^{<T_x>}$ represents the T_xth sequence value.

4.1.6 Prediction: Transformer

(a) Since the publishing of [12] a major shift in deep learning has been focused around the transformer. The transformer architecture has been used by OpenAI in their LLM's. We will only use the encoder portion of the transformer architecture. For a more rigorous discussion on the topic refer to [13]

(b) As discussed in section 4.1.5 (c) we will split the real and imaginary parts of the data after positional encoding and remove the 10th TA value from the datasets. The architecture is summarized below:

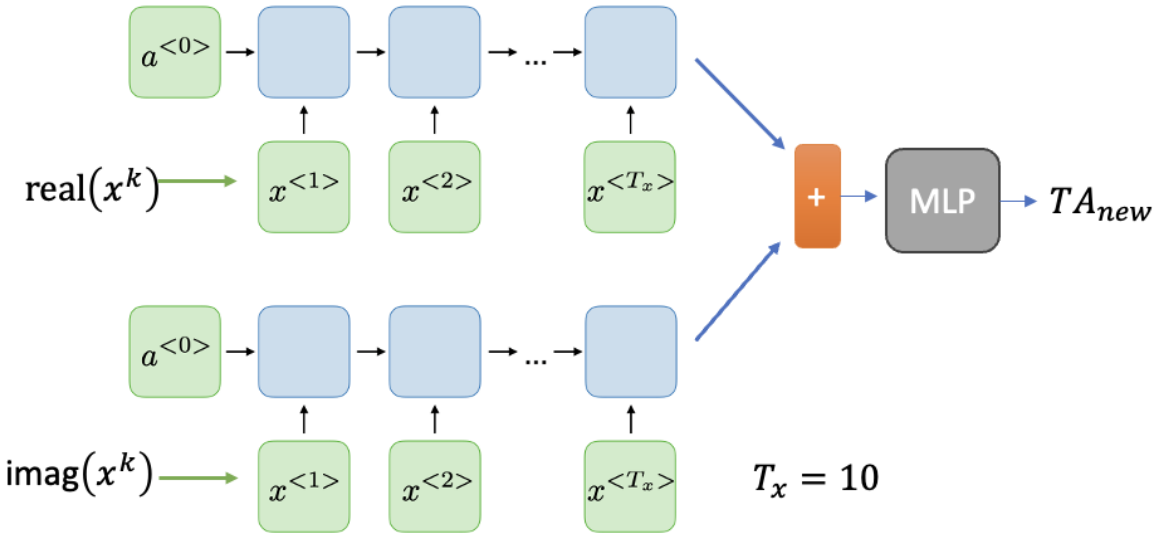

Figure 2: Architecture of the LSTM used in the sequential case. Image modified from [2].

(i) Number of attention heads: 1

(ii) Number of encoder layers: 1

(iii) Output layer: summation + single neuron

(c) We trained the model with and without positional encoding. Positional encoding gave marginal performance gains. The position encoding function is identical to the one used in [12]

4.2 Explanation of Method Used and Results

4.2.1 Classifcation: MLP

We elected to use the feedforward network in the classification task as this architecture is able to handle non separable and non linear data; the more layers and neurons the more probable the model will learn the decision boundaries in some high dimensional space. We also present the MLP as an alternative architecture to what was used in [8]. The models in [8] could have overlooked patterns in the feature space which a MLP could identify, although this is doubtful as kernel methods were employed in [8]. The choice of the number of layers and neurons is entirely by random selection, the architecture outlined in section 4.1.1 gave the best results. See table 1 under MLP for results.

4.2.2 Classifcation: Ensemble methods

The reasoning behind this architecture was twofold: aggregating N classifiers could boost model performance and to transform our features into a higher dimensional, sparse space, i.e., making the data more separable. Once the features are transformed into a higher dimension, we train a linear model on these features in hope

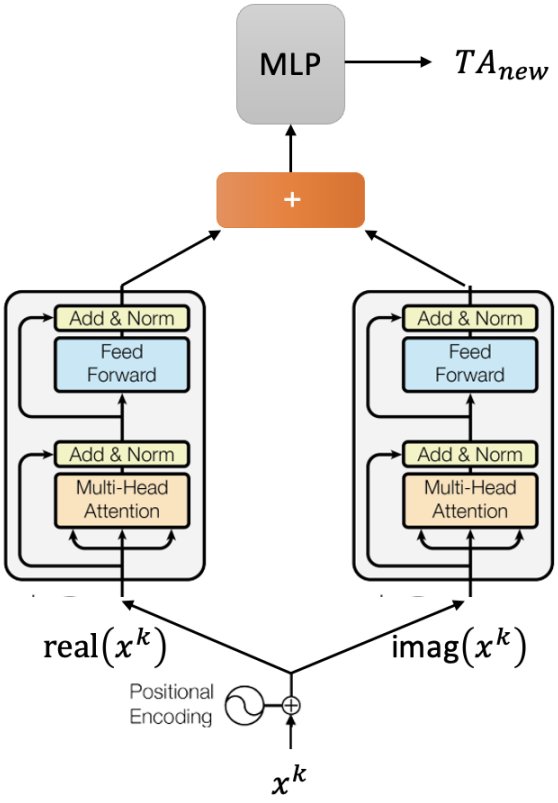

Figure 3: Architecture of the Transformer used in the sequential case. Image modified from [12].

that it outperform a lower dimensional feature space classifier [10]. The choice of selecting 100 classifiers was for computational simplicity. Results for random tree embedding (RT) and random forest with a linear layer (RF) are shown in table 1.

4.2.3 Prediction: MLP

We decided to use the feedforward network in the prediction task as this architecture tends to perform better on non separable and non linear data; the more layers and neurons the more probable the model learn the approximating function in some high dimensional space. To test the prediction capabilities of out classification MLP, we simply transferred this architecture directly from the section 4.2.1 and changed the

Table 1: Cell size 5km. The rows in grey represent benchmarks.

Model	Accuracy	Sensitivity	Specificity	PPV	NPV	F1 Rate
SVM	0.7625	0.0765	0.9971	0.9021	0.7594	0.1411
Boost	0.8147	0.4703	0.9325	0.7046	0.8373	0.5641
MLP	0.8331	0.6782	0.8839	0.6568	0.8934	0.6674
RF	0.8088	0.6001	0.8789	0.6245	0.8675	0.6120
RT	0.8151	0.5311	0.9122	0.6742	0.8505	0.5942

loss function to MSE and removed the activation at the output. Refer to the discussion in section 4.2.1 for parameter analysis. Results are presented in table 2 under MLP.

4.2.4 Prediction: LSTM

The authors in [8] suggested the use of sequential ML models for predicting TA. The reasoning to use a sequential model is as follows: As the cell size increases, the measurements of power become more corrupt with noise causing the signal condition to become less predictable. Additionally, as the distance increases the ΔP between some some finite increment approaches 0, i.e. $\Delta P = P(d+\epsilon) - P(d) \approx 0$ for some small ϵ and distance $d > 2.5$km.

Thus we propose the idea of a sequence of features to enable the model to learn these properties mentioned above. Specifically, if we give the model k samples for $k > 3$, we would expect that as the UE moves away from the center of the cell the values of TA will increase while power measurements will begin converge, see figure 4.

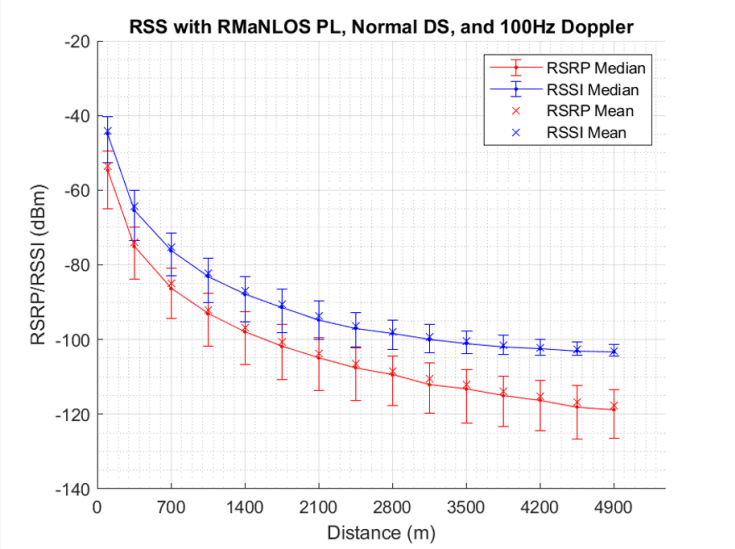

Figure 4: Plot of P_{RSSI} and P_{RSRQ} vs a UE's radial distance from the cell's center.

Although our sequence length of 10 is not large, we used the LSTM as it is an improved sequential model over the traditional RNN. The choice of splitting the data into real and imaginary parts was the result of vanishing gradients during training, i.e., when the features were complex, the loss values after the first epoch would be `NaN`.

4.2.5 Prediction: Transformer

We use the transformer to compare its performance with the LSTM. Essentially, we wanted to see if the self-attention mechanism would be enough to predict the next TA value given a sequence of X_s. We neglected to use the decoder layer as this is not a sequence-to-sequence problem, rather a many-to-one problem.

We used one encoder layer and one attention head to prevent over-fitting and these parameters gave the best results based on the data we had obtained; we suspect that if the data set was larger or the sequence length increased using more than one attention head would be appropriate. We trained the transformer with and without positional encoding, however, the choice to use PE gave a marginal performance gain; the

Table 2: Cell size 5km. The rows in grey represent benchmarks.

Model	Accuracy	Error Standard Deviation	Mean Absolute Error	Mean Error	RMSE
LS + Boost	73.1817	8.6129	6.3890	-0.4816	8.6263
MLP	76.1400	8.4348	5.8827	-0.0164	8.4348
LSTM	98.6119	2.9185	2.4234	0.6750	2.9956
Transformer	98.0458	3.2771	2.4234	-0.1585	3.2809

reasoning behind this is that adding PE allows the model to 'see' ordering in the data. The PE functions can be found in [12].

Predicting the last sequence value of TA was done in two ways: masking the entire tenth value of features or only masking the tenth TA value, $TA_{old}^{(10)}$. Accuracy was 4% higher when only masking the last TA value. This should be clear since the model has access to all ten measurements of $\mathcal{P}$. We note that the Transformer has the potential to outperform the LSTM if trained on more data.

5 What did not work

5.1 Non-sequential Models: MLP & Ensemble Methods

As discussed before, the seperation boundary between a valid and invalid TA is non-linear and overlapping. This causes issues in low complexity or classical models as they tend to perform well on separable and low dimensional feature spaces.

In an attempt to make the decision boundary more linear by projecting the features in a high dimensional space using ensemble methods proved to have little effect on performance. The problem stems from the features themselves and not the model; as the UE moves farther away from the BS, the measurements of power become less predictable and vary less. That is, when UE moves from some initial position $d_i > 2.5$km to some final position $d_f > d_i$, the change power, $\Delta\mathcal{P}$ is very small while the TA would change linearly. This is unlike the case when $d_i < 2.5$km and $d_f < d_i$ where $\Delta\mathcal{P}$ would change quadratically while the TA would remain to change linearly (see figure 4).

5.2 Sequential Models: LSTM & Transformer

These models performed exceptionally well on the data. This reasoning can be partially explained by the concept of indirectly telling the model that the power measurements become less sensitive at large distances. That is, if the UE is moving at a radial path away from the BS then the TA values would signify the UE movement whereas $\mathcal{P}$ would become unreliable. We can also state that in the sequential model case, we had control over the trajectories each UE took. By control we do not mean deterministic control, rather a UE path that would be programmed at a mean velocity and variance. An important note: the testing dataset was simulated using a uniform distribution of velocities, variances and trajectory model. We hypothesize that if these models were tested on pure stochastic trajectories, i.e., trajectories which do not follow pre-programmed path with added noise would yield results lower than what was presented here.

6 Conclusions and Future work

In this paper, we introduced two problems to be solved: (1) prediction and classification of TA values based on the change in power and a single TA value based on the UE's previous position, and (2) Prediction of TA values using sequential modelling. The performance of our non-sequential models, MLP, RT and RF, outperform the models presented in [8], however, by marginal standards. We then propose an alternative to use a sequence of power samples and old TA values, which would be fed into a LSTM or Transformer. We conclude that the results using a sequence of 10 past measurements can substantially outperform the models in the non-sequential case.

Noise in our data prevented the non-sequential models to perform adequately, further research should be conducted on methods of signal recovery and de-noising strategies. We suggest the idea of a sequence-to-sequence task using an LSTM or Transformer to predict the next k, for some $k > 1$, TA values. We observed that although our sequential models performed well, they are far from perfect and suspect they would fail on stochastically generated paths. Finally, we propose the idea of introducing an additional layer of complexity to the simulation by increasing the cell size, the number of cells and/or moving beyond the previous generation of cellular communications to 5th and 6th generation.

References

[1] Huda Althumali and Mohamed Othman. "A Survey of Random Access Control Techniques for Machine-to-Machine Communications in LTE/LTE-A Networks". In: *IEEE Access* 6 (2018), pp. 74961–74983. DOI: 10.1109/ACCESS.2018.2883440.

[2] Afshine Amidi and Shervine Amidi. *Cheatsheet: Recurrent Neural Networks*. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks. 2018.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[4] Andreas Hoglund et al. "3GPP Release 15 Early Data Transmission". In: *IEEE Communications Standards Magazine* 2.2 (2018), pp. 90–96. DOI: 10.1109/MCOMSTD.2018.1800002.

[5] Andreas Hoglund et al. "3GPP Release-16 Preconfigured Uplink Resources for LTE-M and NB-IoT". In: *IEEE Communications Standards Magazine* 4.2 (2020), pp. 50–56. DOI: 10.1109/MCOMSTD.001.2000003.

[6] Wanyu Li et al. "Dynamic Allocation of RACH Resource for Clustered M2M Communications in LTE Networks". In: *2015 International Conference on Identification, Information, and Knowledge in the Internet of Things (IIKI)*. 2015, pp. 140–145. DOI: 10.1109/IIKI.2015.38.

[7] MathWorks. *Reference Signal Measurements (RSRP, RSSI, RSRQ) for Cell Reselection*. [Online; accessed December 2023]. Accessed 2023. URL: https://www.mathworks.com/help/lte/ug/reference-signal-measurements-rsrp-rssi-rsrq-for-cell-reselection.html.

[8] Gautham Prasad et al. *SmartPUR: An Autonomous PUR Transmission Solution for Mobile C-IoT Devices*. 2023. arXiv: 2306.12336 [eess.SP].

[9] PyTorch. *BCEWithLogitsLoss*. Accessed: 2023-12-14. 2023. URL: https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html#bcewithlogitsloss.

[10] *Scikit-learn: Feature Transformation Example*. https://scikit-learn.org/stable/auto_examples/ensemble/plot_feature_transformation.html. Accessed: 2023-12-14. 2023.

[11] *sklearn.ensemble.RandomForestClassifier*. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. Accessed: 2023-12-14. 2023.

[12] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].

[13] Aston Zhang et al. *Dive into Deep Learning*. https://d2l.ai/chapter_preface/index.html. Accessed: 2023-12-14. 2023.