Framework: Clustering-Driven Approach for Base Station Parameter Optimization and Automation (CeDA-BatOp)

Sudharshan Paindi Jayakumar, Alberto Conte

Framework: Clustering-Driven Approach for Base Station Parameter Optimization and Automation (CeDA-BatOp)

Sudharshan Paindi Jayakumar*, Alberto Conte*
*Nokia Bell Labs France

***Abstract*—The growing demand for fast and reliable wireless services has led to the deployment of more base stations, which has made manual optimization of base station parameters more complex and time-consuming. This can lead to suboptimal network performance and a poor user experience. To address this challenge, we propose a Clustering-Driven Approach for Base Station Parameter Optimization and Automation (CeDA-BatOp), an automated for predicting optimized base station parameters. Our framework first compares three clustering algorithms: K-means, DBSCAN, and Agglomerative Clustering, selecting the most suitable one for specific scenarios based on their unique attributes. In parallel to clustering, our framework leverages machine learning (ML) algorithms to predict the optimal parameters for each base station with an evaluation of multiple ML models to identify the best fit for our data. The framework also incorporates data drift monitoring to track gradual changes in data distribution over time, ensuring ML model accuracy through periodic retraining. Our framework achieved an average of 76% reduction in memory overhead on the simulated scenario. Employing a clustering-driven strategy enabled us to simplify the training by utilizing fewer models applicable to multiple base stations, leading to reduced time and computational resources. Additionally, our drift detection system obtained an accuracy of 98.87%, which is the best among the cases we evaluated. The promising results of our study show that our framework can significantly benefit network operators by automating the tuning of base station parameters through a clustering-driven approach. This can lead to reduced human involvement and significant performance gains and cost savings.**

***Index Terms*—Base station parameter, Clustering, Framework, Network automation, Optimization**

I. Introduction

Artificial intelligence (AI) is being used more and more in cellular networks to increase efficiency, optimize network performance and enhance the user experience. The optimization of network resources is one of the key applications of AI that can be found in cellular networks [1], [2]. AI and ML algorithms are able to examine vast volumes of data from a variety of network components, such as base stations and user devices, in order to recognize trends and anticipate network congestion. Based on this, network operators can make proactive adjustments to network resources, such as providing additional bandwidth to regions that see heavy use or increasing the number of base stations located in a certain location [3], [4]. An instance of the use of AI can be seen in self-organizing networks (SON), which are networks capable of self-configuration and optimization without human involvement. This is made possible by the use of AI algorithms that analyze network data and make modifications to enhance coverage, capacity and overall effectiveness. This has the potential to result in remarkable performance improvements and cost reductions for network operators [5].

The integration of AI technology is being applied in cellular networks to improve the user experience. For instance, AI/ML algorithms can be employed to analyze data collected from user devices to tailor the experience for each user. This can include offering personalized content suggestions or altering the network settings to improve battery life [6]. Additionally, AI is put to use in cellular networks to predict and prevent disruptions to network service. Algorithms powered by AI are able to examine data collected from individual network nodes to spot potential problems, such as faulty hardware or software, well in advance of their ability to bring down the network [7], [8]. As a result, network operators can take proactive measures to prevent outages before they occur. This makes the network more reliable for users.

This paper sets the base for the clustering-driven automated framework with the potential to have a substantial impact on the optimization of base station parameters in contemporary wireless networks. This automated framework's inherent capabilities can be expanded to effectively address issues such as outages, disruptions, personalizing user experiences and more. Especially in the current global landscape, where base station traffic patterns show dynamic swings across different areas, resulting in uneven demand distribution at different times, it is crucial to optimize base station characteristics by anticipating and characterizing these complex traffic dynamics. Manual collective optimization of base stations within restricted regions is time-consuming and requires domain experts to alter parameters. In the area of optimization problems where machine learning is used, getting enough measurement data to draw accurate conclusions is a big challenge.

In light of the aforementioned difficulties and the capability of modern AI/ML algorithms, this study proposes a proactive approach. We propose CeDA-BatOp version 1.0, a method to optimize critical base station parameters by utilizing data originating from user equipment (UE) endpoints. By minimizing human involvement in base station administration and parameter optimization, we aim to increase operational efficiency and reduce potential errors caused by manual intervention. Additionally, striking a balance between prediction accuracy and memory consumption allows for efficient utiliza-

tion of network resources while ensuring smooth transmission of model parameters in the network. This is achieved by formulating an end-to-end automated framework designed for ORAN's RAN Intelligent Controller (RIC) [9], utilizing a clustering-driven methodology. By seamlessly integrating this framework as xApps and rApps within RIC, we aim to elevate the functional aspects of cellular networks.

Our main contributions are: We present a novel automated system for predicting optimal base station parameter optimizations, assuming that a lookup database is available.

Within the framework of CeDA-BatOp version 1.0:

1) For the clustering part of the framework, we provide a comparative study of three diverse clustering algorithms on our dataset - K-means, DBSCAN and Agglomerative clustering (AHC) and choose the one with the best Silhouette score [10].
2) For the prediction phase, we leverage the cluster information derived from the aforementioned best-performing algorithm and we conduct an extensive performance analysis. This analysis encompasses a diverse range of machine learning algorithms incorporating:
   - Support Vector Machine (SVM)
   - Stochastic Gradient Descent Regressor (SGDRegressor)
   - Gaussian Process Regression (GPR)
   - Fully Connected Neural Networks (FCNN)
3) For the prediction phase, we compare the following cases:
   - One model for all base stations: This approach uses a single model to learn the traffic patterns of all base stations. This is the simplest approach, but it may not be the most accurate, as the traffic patterns of different base stations can vary significantly.
   - N models for N clusters: This approach divides the base stations into N clusters, and then trains a separate model for each cluster. This approach can be more accurate than the first approach, as it takes into account the different traffic patterns of each cluster.
   - A separate model for each base station: This approach trains a separate model for each base station. This is the most accurate approach, but it is also the most complex and computationally expensive.
4) For the drift phase, we compare the accuracy of three diverse drift algorithms on the new online data - Kolmogorov-Smirnov (KS) test [11], Population Stability Index (PSI) [12] and Jensen-Shannon divergence (JS) [13]. These algorithms are commonly used in detecting and quantifying changes in data distribution. The Kolmogorov-Smirnov test measures the maximum difference between the cumulative distribution functions of two samples, while the Population Stability Index evaluates the similarity between two probability distributions. On the other hand, Jensen-Shannon divergence calculates the dissimilarity between two probability distributions using information theory.

The rest of the paper is organized as follows: Section II provides a comprehensive summary of related works. In Section III, we describe the framework's intricate components and their interdependence. In Section IV, we provide a detailed explanation of the framework's components, including the simulation setup and the specific parameters used in the machine learning algorithm. Additionally, we present a thorough analysis of the empirical results obtained from applying the framework, highlighting its effectiveness and potential limitations. In Section V, the paper concludes by summarizing the main findings and implications of the study. Additionally, it discusses potential avenues for future research in order to further explore and expand upon the current findings.

II. Related Work

In this section, we provide the background for the concepts relevant to the framework:

A. Base station clustering

While existing solutions have made significant advancements in base station clustering especially in the application of beamforming, they often suffer from limitations that restrict their applicability to specific problems. For instance, Hong et al. [14] demonstrated the effectiveness of base station clustering and beamforming for partially coordinated transmission in a dense urban environment, where multiple base stations can serve a high number of users simultaneously. Similarly, other notable works such as [15] have utilized zero-forcing techniques for intra-cluster transmission, further improving the efficiency and capacity of base station clustering. By aligning base station clusters with complementary usage profiles, Chen et al. [16] were able to optimize limited processing resources and improve overall network performance. While their solution showed promising results in such an environment, its effectiveness may be limited in rural or suburban settings with different network characteristics. By showcasing these limitations, we emphasize the need for a versatile framework that can be applied to various parameter types and adapt to different network scenarios.

B. Machine Learning assisted base station parameter optimization

Machine learning's impact on handover parameter optimization in self-organizing networks has been substantial [17]–[19]. Wang et al. [20] innovated base station design optimization using machine learning techniques. Power allocation optimization in heterogeneous environments, as demonstrated in [21], is another principal use case. Dreifuerst et al. [22] pioneered Bayesian optimization and reinforcement learning for coverage and capacity optimization. Their work integrated multiple inputs such as throughput, SINR, RSRP, and RSRQ. When optimizing base station parameters, it is important to consider a wide range of inputs. In our case, we encompass RSRQ, RSRP, RSSI, SINR, CQI, DL/UL bitrate, download process status, and neighboring base station parameters as inputs. By leveraging these inputs, we can effectively optimize base station parameters to ensure optimal network performance and user experience.

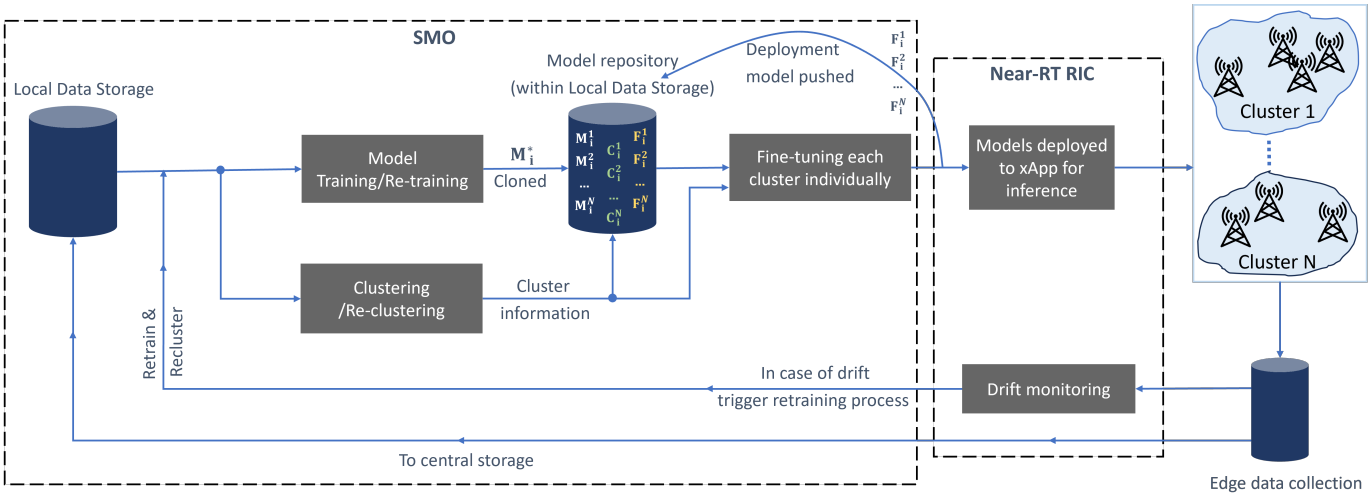

FIG. 1: Base station Parameter Optimization and Automation Framework

C. Data drift in Cellular Networks

Data drift occurs when input data distribution shifts, affecting machine learning model performance. On the other hand, concept drift retains data distribution but alters the interaction between input and target variables, necessitating timely detection. This has significant implications for mission-critical applications.

While finance leverages data and concept drift extensively, their potential remains underutilized in cellular networks. In this domain, most drift-incorporating papers focus on anomaly detection [23], [24]. Recent applications include detecting user behavior changes [25] and network traffic shifts in federated learning [26], indicating a potential area to explore. Our framework continuously monitors data drift and initiates retraining when necessary to maintain model accuracy. This proactive approach enables network operators to adapt their models to changing network conditions, thus enhancing network performance. By integrating data drift detection into cellular networks, operators can proactively identify and address potential issues, ensuring efficient and reliable network operation.

III. FRAMEWORK

Fig. 1 depicts the proposed framework for base station clustering and parameter optimization. Initially, an offline dataset (D_1) consisting of UEs and base station parameters is generated using the in-house simulator and stored in a data repository. UE and base station parameters serve as the input and output of the ML models respectively. The rest of the framework can be divided into three main stages:

A. Stage 1: Training on the overall dataset and clustering in parallel

Initial cycle: Initially, we have a model M_1^* (1 refers to the iteration number) which can be a machine learning/deep learning model that can perform prediction. In our case, it is a 5-layer (4 hidden layers+output layer) FCNN which was obtained through hyperparameter tuning. The purpose of using the M_1^* is to establish a baseline for comparison with its fine-tuned version which is updated through the retraining process. Using an established model as a starting point allows for a more controlled and methodical investigation of our framework. Model M_1^* is trained on the initial offline dataset D_1 for E epochs. In parallel to this, base stations are clustered (data clusters C_1^1,... C_1^N are formed; here subscript 1 refers to the iteration number, initial iteration in this case) using the pairwise constrained clustering technique [27]. The pre-trained FCNN model M_1^* is then cloned to make M_1^1,... M_1^N for N clusters.

Retraining cycle: New models of M_i^1,... M_i^N ($i > 1$ and $i \in \mathbb{N}$) are obtained by training the pre-trained model M_{i-1}^* on the new overall dataset D_i for E epochs and cloning as similar to the above case. Here, $D_i = D_{i-1} \cup O_{i-1}$, where O_i refers to the new data obtained at i^{th} iteration and D_i is the overall dataset available for i^{th} retraining cycle. Simultaneously, base stations are clustered using D_i data.

B. Stage 2: Fine-tuning and Prediction

After stage 1, we now have the data clusters C_i^1,... C_i^N, where $C_1^1 \subset D_1$,... $C_1^N \subset D_1$ for the initial cycle and $C_i^1 \subset D_i$,... $C_1^N \subset D_i$ in case of the retraining cycle. These clusters are then used to fine-tune individually the respective pre-trained models M_i^1,... M_i^N by training for Y ($<$ E) epochs. The resulting fine-tuned models F_N^i,... F_N^i are then pushed to the model repository and deployed to xApp for inference of the base station parameters.

C. Stage 3: Drift monitoring and retraining if necessary

The data from the base stations are collected periodically and checked for data drift. In the case of drift, the retraining process is triggered. The new data o_i (input parameters) $\subset O_i$ (input & output parameters) obtained from UEs via the base stations comprises only the UE parameters that serve as the input for the FCNN model. However, it does not include the base station parameters which are the target labels required for the retraining process. At this stage, we assume that we can obtain the labels for the new data from a large lookup

table. Once we obtain the labels, we can retrain and re-cluster the new data thus making it an automated closed-loop system with a requirement of very less human intervention.

In Fig. 1, the left and right dotted lines enclose the components that can be deployed in Service Management & Orchestrator (SMO) and Near-RT RIC respectively.

IV. Simulation Setup, Discussion and Result

The local dataset (initial) was collected using an in-house dynamic system level simulator tool with 42 base stations with the scenario of a locality in Madrid. It consists of a mix of macro cells with directional antennas (23) and small cells with Omni-antennas (19), UEs moving at three different speeds: 200 UEs at 30 km/hr, 40 UEs at 3 km/hr, and 80 UEs at 3 km/hr. This variation in UE movement speeds introduces realistic mobility patterns and enables the ML model to account for different user scenarios. The combination of diverse scenarios, movement speeds and base station configurations makes the dataset a valuable resource for developing and validating the effectiveness of base station-specific ML models in optimizing network performance and enhancing overall user experience in urban environments like Madrid. A map of the simulated scenario is shown in Fig. 2

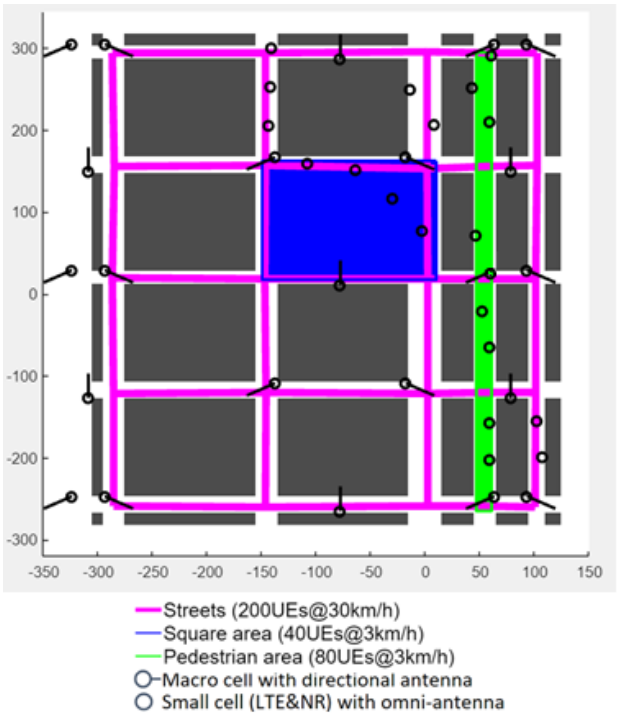

Fig. 2: Map of a simulated locality in Madrid

The UE parameters obtained include Reference Signal Received Quality (RSRQ), Reference Signal Received Power (RSRP), Reference Signal Strength Indicator (RSSI), Signal-to-interference-plus-noise ratio (SINR), Channel quality indication (CQI), Download/upload bitrate (DL/UL bitrate), State - whether the UE is in idle or downloading state (I/D), Reference Signal Received Quality of the Neighbouring cell (NRxRSRQ) and Reference Signal Received Power of the neighboring cell (NRxRSRP). The optimized base station parameters obtained include Antenna tilt - Mechanical tilt and Electrical tilt, Antenna azimuth and Maximum Transmission power. Each parameter consists of several features, for instance, in the case of RSRQ, we have RSRQ values of 42 base stations across time steps. In total, there are 736 input features (UE end) and 4 output features (base station end). A sample of the preprocessed dataset is available here.

In our evaluation, we assessed a diverse array of models to ascertain their effectiveness in optimizing base station parameters. The models encompassed a spectrum of methodologies: SVM Regressor – RBF Kernel, SGDRegressor, Gaussian Process Regression (GPR) - DotProduct() + WhiteKernel(noise_level=0.5), GPR - Exponentiation(RationalQuadratic(), exponent=2), GPR - RBF() + ConstantKernel(constant_value=2), FCNN – 4 layers (12,25,10), FCNN – 5 layers (12,25,12,10), FCNN – 6 layers (12,25,25,12,10).

For SVM, we tuned the regularization parameter 'C', with values ranging from 0.1 to 10, and the kernel coefficient 'gamma', exploring a range from 0.01 to 1. GPR renowned for its adaptability, was deployed with a variety of kernels. For the DotProduct kernel, we adjusted the 'noise_level' parameter, while in the case of Exponentiation of RationalQuadratic kernel, the 'exponent' parameter was set to 2. Additionally, for the combination of RBF and ConstantKernel, the 'constant_value' parameter is set to 2. Our Stochastic Gradient Descent Regressor (SGDRegressor) was fine-tuned with the 'alpha' parameter for L2 regularization and 'learning_rate' ranging from 'constant' to 'adaptive'. Finally, leveraging the potential of deep learning, our FCNN underwent extensive hyperparameter tuning. We explored the optimal number of hidden layers, units per hidden layer (12, 25, 12, 10), ReLU activation functions, an initial learning rate of 0.001 with adaptive optimization, and a regularization term with an L2 strength of 0.0001. By immersing ourselves in this meticulous hyperparameter tuning, we aimed to tailor each model to the unique complexities of our dataset and the intricacies of BTS-specific optimization tasks. Unless otherwise stated, unspecified parameters are assumed to be configured at their default settings as provided by the scikit-learn library [28]. Experiment results were obtained using a Samsung 970 EVO for data storing and loading, Intel Core i7-10750H for training.

The flow of the training and prediction in the case of FCNN – 5 layers (12,25,12,10) are as follows (similarly performed for other models): Initially, the FCNN is trained on the overall dataset and then the model is stored in the repository. For the prediction of base station parameters with FCNN before fine-tuning, we obtain a Root Mean Square Error (RMSE) of 28.34 at epoch 67, which is obtained through early stopping. The results of ML models when trained on the overall dataset are listed in Table II.

From Table II, we can see that the GPR model with Exponentiation(RationalQuadratic(), exponent=2) achieves the best predictive performance with the lowest average RMSE of 24.43 among all models. It outperforms all other models,

including the SVM Regressor, SGDRegressor, and FCNN models. The FCNN models show improvement in predictive accuracy as the number of layers increases. The 6-layer FCNN performs better than the 4-layer FCNN and the 5-layer FCNN in terms of RMSE. The FCNN can be considered a practical option if faster training times and smaller model sizes are required and the slightly higher RMSE is acceptable for the application. It can still provide reasonably good predictive performance, with a trade-off for faster experimentation and deployment.

Parallel to this, we perform Agglomerative Clustering (AHC) using Ward linkage and Euclidean affinity [29] on the overall dataset to obtain cluster information such as the number of clusters and the cluster IDs of the data points. Using the method of dendrograms [30], we found the best cluster size to be 9 and the average Silhouette Score obtained is 0.779. For the above scenario, AHC was chosen among the other cluster algorithms as it provided the best Silhouette score and also allowed us to investigate at varying levels of granularity without the need to rerun the clustering algorithm. The complete results of the clustering algorithms' performance are tabulated in Table I.

Clustering Algorithm	Avg. Silhouette Score	Avg. Time Taken
K-means	0.527	$\sim$11 mins
DBSCAN	0.596	$\sim$13 mins
AHC	**0.779**	$\sim$23 mins

TABLE I: Comparison of clustering algorithms (K-means, DBSCAN, AHC)

With the help of cluster information, we fine-tuned the FCNN – 5 layer models M_1^i,... M_N^i on their respective cluster data C_1^i,... C_N^i. After fine-tuning, we obtain an RMSE of 16.98 at epoch 8, which is obtained through early stopping. We observe a notable decrease in the value of RMSE and we believe fine-tuning the pre-trained model particularizes the prediction for the respective clusters and thus makes a better prediction. The performance for other models is listed in Table III. Note that the total storage occupancy here includes the cluster information. Since clustering is done in parallel to the overall training, it does not affect the training time as the time taken to cluster is less than the time taken to train on the overall dataset. The above-obtained fine-tuned models can then be deployed as xApp for inference of optimized base station parameters.

From Table III, we can decipher that, for fast training and smaller model size, consider using SVM with the RBF kernel or SGDRegressor. However, they may not provide the best prediction. If prediction accuracy is a priority and computational resources are sufficient, Gaussian Process Regression models with appropriate kernels offer improved performance. For deep learning models, the FCNN with 5 or 6 layers performs better than the shallow 4-layer FCNN but requires more training time and larger model sizes.

The FCNN with 5 layers was selected for its best RMSE, low RMSE, and manageable training time, striking a favorable balance between predictive performance and computational efficiency, making it ideal for the application's needs. Other models with better RMSE values had impractical and cost-ineffective longer training times. The decision was driven by achieving satisfactory prediction accuracy while minimizing computational burden.

With FCNN, the performance for the case when we have 42 different models (each base station having its own model) is represented in Table IV.

From Table IV, we can observe that the FCNN model achieved a significantly improved average RMSE of 14.34 compared to just having one global model and a fine-tuned model. This indicates that the model when trained individually for each base station, performs better in predicting the target values. It is worth noting that the total model size has increased substantially to 466 MB. This increase in size is expected because we are now storing individual models for each base station (42 in our case), which leads to a higher overall memory requirement. Training a separate model for each base station can be beneficial if there are substantial variations or unique patterns in the data across different stations. However, it also comes with the trade-off of increased storage requirements. Depending on the resources available and the importance of individual station predictions, this approach can be a suitable choice for specific scenarios. Also, each model is more sensitive to drifts and variations in the operational conditions of each base station. As we can see from Table III and Table IV, there is no major difference in the RMSE for FCNN - 5 layers network between the fine-tuned model and individual model. Our preference therefore lies with the fine-tuned model, predominantly due to its 76% reduction in total storage occupancy. To study the effect of drift on the framework, we randomly drifted the data with a 50% probability across time steps up to an amount of $\pm$5% of the value for all input features. We then separate the drifted data (DF) and non-drifted data (NDF). Similarly, it was done for the drift of maximum $\pm$10% of the value for all input features. In addition to this, to reduce the dimension we performed Kernel Principal component analysis (KPCA) [31] on DF and NDF using poly kernel. DF and NDF were reduced to 100 feature and 350 feature datasets. We name these KDF1, KNDF1, KDF3 and KNDF3 respectively.

We then leveraged similarity tests to detect the drift for this new data. For our case, we benchmarked on Kolmogorov-Smirnov (KS) test [11], Population Stability Index (PSI) [12] and Jensen-Shannon divergence (JS) [13]. For the KS test, we chose a confidence level of 97%, implying we reject the null hypothesis in favor of the alternative if the p-value is less than 0.03. For PSI, the index ranges from 0 to ∞ (PSI < 0.1: no significant population change; $0.1 \leq$ PSI < 0.2: moderate population change; PSI ≥ 0.2: significant population change), in our case, we set the threshold to 0.05. For JS, we set the threshold at 0.1 (JS distance returns a score between 0 and 1. "0" corresponds to identical). These threshold values can be changed as per need based on the application. New drifted and non-drifted datasets DF, NDF, KDF1, KNDF1, KDF3 and KNDF3 were compared against the existing data using

Model	RMSE	Total Training Time (avg.)	Total storage occupancy (MB)
SVM Regressor – RBF Kernel	46.57	∼1hr 43 mins	∼11
SGDRegressor	37.33	∼2hr 32 mins	∼7
GPR - DotProduct() + WhiteKernel(noise_level=0.5)	31.32	∼4hr 37 mins	∼17
GPR - Exponentiation(RationalQuadratic(), exponent=2)	24.43	∼5hr 28 mins	∼19
GPR - RBF() + ConstantKernel(constant_value=2)	37.38	∼4hr 29 mins	∼18
FCNN – 4 layers (12,25,10)	44.43	∼2hr 13 mins	∼11
FCNN – 5 layers (12,25,12,10)	**28.34**	∼2hr 49 mins	∼14
FCNN – 6 layers (12,25,25,12,10)	**27.23**	∼3hr 23 mins	∼24

TABLE II: Performance of models when trained on the overall dataset without fine-tuning

Model	RMSE	Total Training Time (avg.)	Total storage occupancy (MB)
SVM Regressor – RBF Kernel	38.23	∼1hr 48 mins	∼21
SGDRegressor	32.73	∼2hr 39 mins	∼24
GPR - DotProduct() + WhiteKernel(noise_level=0.5)	19.74	∼4hr 42mins	∼32
GPR - Exponentiation(RationalQuadratic(), exponent=2)	17.32	∼5hr 34mins	∼44
GPR - RBF() + ConstantKernel(constant_value=2)	17.92	∼4hr 41mins	∼51
FCNN – 4 layers (12,25,10)	37.03	∼2hr 22 mins	∼81
FCNN – 5 layers (12,25,12,10)	**16.98**	∼3hr 01 mins	∼111
FCNN – 6 layers (12,25,25,12,10)	**15.08**	∼3hr 46 mins	∼147

TABLE III: Performance of models when trained on the overall dataset with fine-tuning

Model	RMSE	Total Training Time (avg.)	Total storage occupancy (MB)
FCNN – 5 layers (12,25,12,10)	14.34	∼3hr 46 mins	∼466

TABLE IV: Performance of FCNN - 5 layers (12,25,12,10) in the case of base stations having their own models (average of 42 models)

these similarity metrics. We observed that KS test was quite sensitive to changes and deviations at the tail were hard to detect. Whereas the sensitivity of PSI was lower compared to KS test, i.e., it reacted only to major changes. JS was sensitive when the drift exceeded 5%. For our dataset, we found it to be more sensitive than PSI but less sensitive than KS. The results of the drift detection for values moved by up to $\pm 5\%$ and $\pm 10\%$ are tabulated in Table V and Table VI respectively.

Cases	Algo	Drift Avg accuracy	Avg false negative rate	Consensus avg. accuracy
(a) Values moved by upto $\pm 5\%$	KS	**90.12%**	1.57%	**96.43%**
	PSI	73.64%	39.35%	
	JS	84.32%	27.11%	
(b) Values moved by upto $\pm 5\%$ - KPCA (poly) [100]	KS	72.77%	13.54%	82.30%
	PSI	59.53%	46.21%	
	JS	76.34%	35.54%	
(c) Values moved by upto $\pm 5\%$ - KPCA (poly) [350]	KS	85.43%	9.32%	93.53%
	PSI	78.32%	42.64%	
	JS	78.11%	30.34%	

TABLE V: For data moved by up to ± 5

Cases	Algo	Drift Avg accuracy	Avg false negative rate	Consensus avg. accuracy
(d) Values moved by upto $\pm 10\%$	KS	**94.34%**	1.89%	**98.87%**
	PSI	83.43%	37.34%	
	JS	84.54%	28.68%	
(e) Values moved by upto $\pm 10\%$ - KPCA (poly) [100]	KS	77.65%	16.34%	83.23%
	PSI	67.05%	37.67%	
	JS	77.43%	33.46%	
(f) Values moved by upto $\pm 10\%$ - KPCA (poly) [350]	KS	87.83%	12.46%	96.45%
	PSI	85.90%	28.01%	
	JS	79.07%	27.48%	

TABLE VI: For data moved by up to ± 10

From the tables V and VI, we can decipher that the KS test provides the best accuracy for the drift detection system. For the case when data is moved by upto $\pm 5\%$, we obtain the maximum average accuracy for case (a) using KS. Similarly, in the case when the data is moved by upto $\pm 10\%$, we obtain the maximum average accuracy of 94.34% for case (d) using KS. The system gets even better at detecting the drift once we consider the consensus of the three algorithms. Here, we use majority voting to obtain the Consensus Average Accuracy (CAA). For instance, in the case when the KS test confirms the presence of drift, PSI confirms no drift and JS confirms the drift, we consider it as drift has occurred since two algorithms have voted in favor of the drift and one algorithm voted against the drift. In case (d), employing the majority voting method results in an accuracy rate of 98.87% surpassing the levels of accuracy obtained in all other cases. From Tables V and VI, we can also observe that the algorithms are better at detecting the drift when the values of the features moved by up to $\pm 10\%$. It is interesting to see that, CAA for cases (a) and (c) are quite comparable even though (a) consists of more than double the features of (c). We can also see a similar trend in (d) and (f) of Table VI. In general, the average drift accuracy of JS is considerably higher than that of PSI except for cases (c) and (f). Once the drift is detected using majority voting of the three algorithms, we then trigger the retraining process.

V. Conclusion

The Clustering Driven Approach for Base Station Parameter Optimization and Automation presents a novel solution for optimizing base station parameters in cellular networks. Further, we provided an illustrative example and conducted a comprehensive study of parameter prediction and clustering systems in our framework. Utilizing clustering-driven models allowed us to discern common patterns among base stations, leading

to enhanced network optimization. Specifically by utilizing AHC, we obtain the best Silhouette Score. We then studied how fine-tuning the pre-trained FCNN model with the help of the clustered data enhances the performance of the system and effectively eliminates the memory constraints associated with individual models for each base station. The results of the study demonstrate the effectiveness of the framework, achieving an average of 76% reduction in memory overhead when compared to having individual models for each base station. The drift monitoring system was then investigated using KS, PSI and JS similarity tests, with Consensus approach among these algorithms yielding the best average accuracy of 98.87% followed by the standalone KS test. The ability to monitor the drift and tackle the incoming data further highlights the robustness of this approach. Overall, this clustering-driven approach represents a promising direction for base station parameter optimization and automation in cellular networks.

Our future work will focus on the direction toward pseudo-labeling to generate labels for the retraining process thus eliminating the need for the large lookup table. Further research can explore additional applications of this framework, including its potential for addressing network disruptions, personalizing user experiences, and adapting to dynamic traffic patterns, thereby advancing the field of network automation and optimization.

ACKNOWLEDGMENT

This work is supported by the European Union's Horizon 2020 research and innovation programme through the SEMANTIC project (Grant No. 861165).

REFERENCES

[1] Janne Riihijarvi and Petri Mahonen. Machine learning for performance prediction in mobile cellular networks. *IEEE Computational Intelligence Magazine*, 13(1):51–60, 2018.

[2] Chunxiao Jiang, Haijun Zhang, Yong Ren, Zhu Han, Kwang-Cheng Chen, and Lajos Hanzo. Machine learning paradigms for next-generation wireless networks. *IEEE Wireless Communications*, 24(2):98–105, 2016.

[3] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. Deep learning in mobile and wireless networking: A survey. *IEEE Communications surveys & tutorials*, 21(3):2224–2287, 2019.

[4] Kun Yang, Cong Shen, and Tie Liu. Deep reinforcement learning based wireless network optimization: A comparative study. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1248–1253. IEEE, 2020.

[5] Olav Østerbø and Ole Grøndalen. Benefits of self-organizing networks (son) for mobile operators. *Journal of Computer Networks and Communications*, 2012, 2012.

[6] Nanticha Poonpanich and Jiroj Buranasiri. Factors affecting baby boomers' attitudes towards the acceptance of mobile network providers' ai chatbot. *Jurnal Nasional Pendidikan Teknik Informatika: JANAPATI*, 11(3):176–182, 2022.

[7] Deniz Gündüz, Paul de Kerret, Nicholas D Sidiropoulos, David Gesbert, Chandra R Murthy, and Mihaela van der Schaar. Machine learning in the air. *IEEE Journal on Selected Areas in Communications*, 37(10):2184–2199, 2019.

[8] Yeh-Hong Ping and Po-Chiang Lin. Cell outage detection using deep convolutional autoencoder in mobile communication networks. In *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1557–1560. IEEE, 2020.

[9] ORAN Community. Ric applications (ricapp), Sep 2022.

[10] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[11] Giovanni Fasano and Alberto Franceschini. A multidimensional version of the kolmogorov–smirnov test. *Monthly Notices of the Royal Astronomical Society*, 225(1):155–170, 1987.

[12] Bilal Yurdakul. *Statistical properties of population stability index*. Western Michigan University, 2018.

[13] ML Menéndez, JA Pardo, L Pardo, and MC Pardo. The jensen-shannon divergence. *Journal of the Franklin Institute*, 334(2):307–318, 1997.

[14] Mingyi Hong, Ruoyu Sun, Hadi Baligh, and Zhi-Quan Luo. Joint base station clustering and beamformer design for partial coordinated transmission in heterogeneous networks. *IEEE Journal on Selected Areas in Communications*, 31(2):226–240, 2013.

[15] Quentin H Spencer, A Lee Swindlehurst, and Martin Haardt. Zero-forcing methods for downlink spatial multiplexing in multiuser mimo channels. *IEEE transactions on signal processing*, 52(2):461–471, 2004.

[16] Longbiao Chen, Dingqi Yang, Daqing Zhang, Cheng Wang, Jonathan Li, et al. Deep mobile traffic forecast and complementary base station clustering for c-ran optimization. *Journal of Network and Computer Applications*, 121:59–69, 2018.

[17] FP ICT-SOCRATES. Handover parameter optimization in lte self-organizing networks. In *Proc. 10th Manage. Committee Meeting (COST)*, 2010.

[18] Seppo Hämäläinen, Henning Sanneck, and Cinzia Sartori. *LTE self-organising networks (SON): network management automation for operational efficiency*. John Wiley & Sons, 2012.

[19] Evolved Universal Terrestrial Radio Access. Self-configuring and self-optimizing network use cases and solutions. *Protocol specification (Release 9)*, 2011.

[20] Subin Wang, Qi Wu, Chen Yu, Haiming Wang, Wei Hong, and Weishuang Yin. Dual-polarized base station antenna design using machine learning-assisted optimization method. In *2021 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting (APS/URSI)*, pages 1715–1716. IEEE, 2021.

[21] Roohollah Amiri, Hani Mehrpouyan, Lex Fridman, Ranjan K. Mallik, Arumugam Nallanathan, and David Matolak. A machine learning approach for power allocation in hetnets considering qos. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–7, 2018.

[22] Ryan M. Dreifuerst, Samuel Daulton, Yuchen Qian, Paul Varkey, Maximilian Balandat, Sanjay Kasturia, Anoop Tomar, Ali Yazdan, Vish Ponnampalam, and Robert W. Heath. Optimizing coverage and capacity in cellular networks using machine learning. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8138–8142, 2021.

[23] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

[24] Rosana Noronha Gemaque, Albert França Josuá Costa, Rafael Giusti, and Eulanda Miranda Dos Santos. An overview of unsupervised drift detection methods. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(6):e1381, 2020.

[25] Dimitrios Michael Manias, Ali Chouman, and Abdallah Shami. A model drift detection and adaptation framework for 5g core networks. *arXiv preprint arXiv:2209.06852*, 2022.

[26] Amir Hossein Estiri and Muthucumaru Maheswaran. Attentive federated learning for concept drift in distributed 5g edge networks. *arXiv preprint arXiv:2111.07457*, 2021.

[27] Pierre Gançarski, Thi-Bich-Hanh Dao, Bruno Crémilleux, Germain Forestier, and Thomas Lampert. Constrained clustering: Current and new trends. *A Guided Tour of Artificial Intelligence Research: Volume II: AI Algorithms*, pages 447–484, 2020.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[29] Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*, 2011.

[30] M Forina, C Armanino, and V Raggio. Clustering with dendrograms on interpretation variables. *Analytica Chimica Acta*, 454(1):13–19, 2002.

[31] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *Artificial Neural Networks—ICANN'97: 7th International Conference Lausanne, Switzerland, October 8–10, 1997 Proceeedings*, pages 583–588. Springer, 2005.