Traffic-Driven Sounding Reference Signal Resource Allocation in (Beyond) 5G Networks

Claudio Fiandrino*, Giulia Attanasio*†, Marco Fiore* and Joerg Widmer*
*IMDEA Networks Institute, Madrid, Spain
†Universidad Carlos III de Madrid, Spain
Email: {claudio.fiandrino, giulia.attanasio, marco.fiore, joerg.widmer}@imdea.org

Abstract*—Beyond 5G mobile networks have to support a wide range of performance requirements and unprecedented levels of flexibility. To this end, massive MIMO is a critical technology to improve spectral efficiency and thus scale up network capacity, by increasing the number of antenna elements. This also increases the overhead of Channel State Information (CSI) estimation and obtaining accurate CSI is a fundamental problem in massive MIMO systems. In this paper, we focus on scheduling uplink Sounding Reference Signals (SRSs) that carry pilot symbols for CSI estimation. Under the large number of users and high load that are expected to characterize beyond 5G systems, the limited amount of resources available for SRSs makes the legacy 3GPP periodic allocation scheme largely inefficient. We design TRADER, an SRS resource allocation framework that minimizes the age of channel estimates by taking advantage of machine learning-based short-term traffic forecasts at the base station level. By anticipating traffic bursts, TRADER schedules SRS resources so as to obtain CSI for each user right *before* the corresponding traffic arrives. Experiments with extensive real-world mobile network traces show that our solution is efficient and robust in high load scenarios: with respect to a round robin schedule of aperiodic SRS, TRADER provides more often CSI within the coherence time (up to $5\times$ for given scenarios), leading to channel gains of up to 2 dB.

I. Introduction

As the fifth generation of mobile networks (5G) is now being deployed worldwide, opportunities to further improve data rates, number of simultaneously connected devices, reliability and latency start being investigated for beyond-5G systems. The increasing complexity of mobile architectures makes self-configuration and self-optimizationkey traits, enabled by data-driven management and control of network components [1].

In this work, we apply a data-driven approach to the specific problem of efficiently configuring signals for channel state information (CSI) estimation in massive MIMO systems – a crucial technology to scale the capacity of beyond-5G mobile networks. Indeed, acquiring accurate CSI between each transmitter and receiver antenna pair is paramount to fully exploit the potential of massive MIMO where at least 64 antennas per base station are used [2]. However, Channel Quality Indicator (CQI) reports are either too coarse (when single reports are used for the entire band), or they come with a prohibitively large overhead (when performed on each sub-band) under massive MIMO [3]. Instead, for TDD systems with channel reciprocity, uplink CSI is also valid for the downlink, and uplink SRS pilots are an effective solution to obtain accurate CSI for downlink MIMO precoding. Therefore, when and for which user to schedule SRS resources becomes a crucial problem in future massive MIMO systems.

978-1-6654-4108-7/21/$31.00 ©2021 IEEE

In the high-load scenarios with hundreds of active users at a time that are expected to characterize beyond-5G networks, the periodic SRS scheduler defined by 3GPP [4] can be inefficient. On the one hand, the available SRS resources are insufficient to guarantee that each user equipment (UE) can be scheduled continuously, and the base station (BS) often has to use stale CSI. On the other hand, not all active UEs have traffic over subsequent frames, and a periodic allocation risks to waste substantial SRS resources for UEs with no traffic to be served.

To make the best use of SRS resources, one should schedule SRSs just *before* a burst of traffic starts, as well as *during* a burst as frequently as possible. This requires operating directly at the level of the BS scheduler, where the only information available is the Modulation and Coding Scheme (MCS), Physical Resource Blocks (PRB) and Transport Block Size (TBS) at each Transmit Time Interval (TTI). Such traffic allocations occur irregularly over very fast timescales of tens of milliseconds, and must be accurately predicted for every user independently. However, current solutions proposed in the vast literature on mobile traffic forecasting typically target much lower time granularity, in the order of minutes to hours [5], [6]. Even fine-grained prediction mechanism like LinkForecast [7] and PERCEIVE [8] still aim at predictions over hundreds of milliseconds to seconds. The only model considering timescales aligned with our needs operates on traffic that is aggregated at the BS level rather than the much more challenging case of per-user traffic [9].

We propose TRADER, a TRAffic-DrivEn Resource allocation framework that employs per-user TTI-level traffic forecasts to inform an aperiodic SRS scheduling – an alternative to the default periodic approach specified by the standard [4]. This aligns with ITU-T recommendations for ML inclusion in network operational lifecycle [10]. Specifically, TRADER strives to minimize the *age of channel estimates* to ensure both frequent channel measurements within bursts and to anticipate the start of a burst with an SRS transmission. To this end, TRADER takes advantage of the time series prediction capabilities of Long Short-Term Memory (LSTM) neural networks, by feeding them with traffic allocation information on the burst size, duration and *gap* (*i.e.*, the idle time between subsequent traffic allocations) generated by individual users.

We evaluate the performance of TRADER by using real-world LTE traffic traces, which we collect from production BSs of two different major European mobile network operators. Specifically, we use passive measurement tools based on software-defined radios (SDRs), FALCON [11] and OWL [12], to decode the unencrypted information of the Physical Downlink Control CHannel (PDCCH) of LTE in a fully privacy-preserving manner. Our results show that TRADER largely outperforms a round robin schedule of aperiodic SRS, as it is capable of triggering more often an SRS right before the user generates traffic. For example, while a round-robin heuristic provides a median of 10 frames of anticipation, TRADER is able to lower this number down to 2, which is the minimum possible value. Thanks to this, TRADER provides more often CSI within the coherence time (up to $5\times$ more for given scenarios), and achieves channel gains up to 2 dB.

II. Background, Dataset and Motivation

A. Sounding Reference Signals

To obtain CSI, mobile networks rely on the CSI-RS for the downlink and on the SRS for the uplink. In time division duplexing (TDD) systems with channel reciprocity, uplink SRS feedback can also be used to estimate the downlink channel [3]. In line with both LTE and 5G specifications, we consider 10 ms TDD frames subdivided into subframes of 1 ms, and 14 OFDM symbols per subframe for a 20 MHz channel. Uplink and downlink share the same frequency band, hence the switching delay between transmission and reception is accounted for by using a guard period. Depending on the uplink and downlink switching periodicity, up to 7 types of TDD frame structure configurations can be used (see Table 4.2-2 of [4]). The guard period is announced in a special frame, along with the Downlink Pilot Time Slot (DwPTS) and the Uplink Pilot Time Slot (UpPTS), whose main purpose is to carry pilot signals including the SRS. Specifically, SRS signals are transmitted during the UpPTS and can span 1, 2 or 4 symbols mapped to the last 6 symbols of the subframe.

The SRS is configured at the Radio Resource Control Layer (RRC) and can be periodic or aperiodic [4]. Periodic SRS is scheduled with a periodicity that ranges from 2 ms to 320 ms. In contrast, an aperiodic SRS has to be actively triggered for each occurrence. In addition, other parameters can be configured for the SRS. The *cyclic shift* allows to send multiple orthogonal signals, *e.g.*, a cyclic shift equal to 4 allows the BS to configure 4 UEs in the same subframe. The *transmissionComb* parameter is a flag defining whether SRSs are transmitted in every even or odd subcarrier; it also provides the BS with the capability of multiplexing two UEs by assigning them the same cyclic shift, frequency and time resources, but different *transmissionComb*.

B. Dataset

For our study, we collect a dataset of LTE traffic allocations from multiple BSs located in different areas of Madrid, Spain. For completeness, we run both the SDR-based LTE sniffer tools FALCON [11] and OWL [12] on a Linux laptop connected to a USRP B210 to decode the unencrypted information of the TTI-level traffic allocation that LTE BSs send to the UEs over the PDCCH channel. Specifically, we gather the temporary user ID (C-RNTI), the ID of the frame containing the traffic allocation for the C-RNTI, and the associated transport block size (TBS). This information is sufficient to determine the size and duration of per-user traffic bursts and idle times between transmissions, *i.e.*, the gaps. From the collected data, we filter out background traffic by removing RNTIs that have less than 5 active TTIs over the entire activity period; we also discard RNTIs reserved for random access (RA-RNTI with ID 1-960), paging and system notification (P-RNTI with ID 65534), and broadcast system information (SI-RNTI with ID 65535).

Fig. 1 shows our measurement data. The plots illustrate the number of UEs that are simultaneously connected with active or inactive RRC states to two BSs. The left plot refers to a BS monitored for 3 h with OWL, and covering a typically crowded touristic area. The plot on the right concerns a BS monitored for over 13.5 h with FALCON, and covering a quiet residential area. Note that the user count takes into consideration that the RNTI is a temporary ID: when the time elapsed from the last transmission exceeds the RNTI refresh timer interval of 10.15 s, the UE[1] performs an RRC re-establishment to obtain a new C-RNTI, without affecting the number of users.

C. Motivation

The scenarios portrayed in Fig. 1 illustrate how scheduling SRSs periodically can be highly inefficient: in both BSs, the number of UEs that can be potentially scheduled to transmit largely exceeds the capacity of the periodic SRS.[2] What is needed is a more sophisticated mechanism to trigger SRSs at the right moment in time and for the right users, so as to maximize the usefulness of SRS resources. More precisely, if a traffic burst starts at frame t, an SRS triggered at $t-2$ would be optimal, leaving time for the BS to inform the UE through control information ($t-2$), receive the feedback ($t-1$), and exploit the fresh CSI (t).

However, the bursty and heterogeneous nature of user traffic patterns at millisecond granularity makes achieving an ideal SRS schedule particularly difficult. Fig. 2 shows the distribution of gaps in the traffic generated by individual users during the 3 h dataset in Fig. 1: in 37% of the cases, the gap is lower than 5 LTE frames (*i.e.*, 50 ms), and in 50% of the cases the gap is less than 27 frames; however, in the remaining 50% of the cases, the gap sizes can take any value between 27 and 1015 frames (the latter matches the RNTI refresh timer duration). Such distribution calls for a framework capable of providing accurate forecasts at the frame level that can steer the decision mechanism of SRS resource allocation. Our solution tackles precisely such a problem, using machine learning to anticipate per-user traffic gaps, and then leveraging such information to self-configure aperiodic SRSs to the most appropriate users.

[1]We use the terms *user* and *UE* interchangeably.

[2]For instance, with a *cyclic shift* of 4, a *transmissionComb* set to multiplex 2 UEs and a MIMO system with 2 RF chains at the UE and 4 chains at the BS, the periodic SRS offers 64 signals during each 10-ms frame, which is much lower than the number of users observed in our measurements.

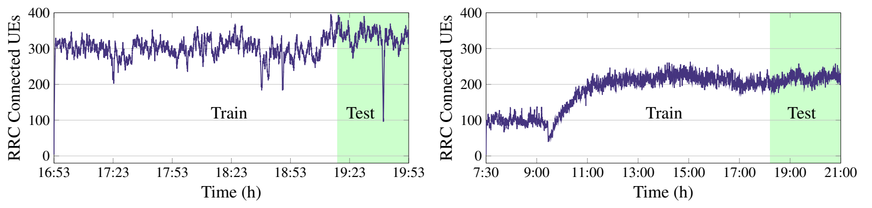


Fig. 1. Number of RRC connected UEs over the period of analysis. On the left the 3 h dataset collected with OWL [12], on the right the 13.5 h dataset collected with FALCON [11].

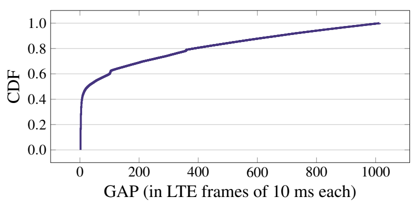


Fig. 2. CDF of gaps (*i.e.*, idle times in subsequent traffic allocations of a user) for the 3 h dataset.

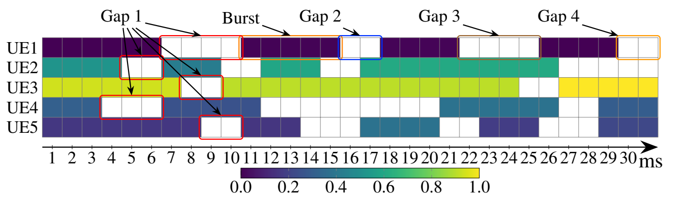


Fig. 3. Snapshot of 30 ms of traffic generated by 5 UEs. Bursts are colored according to their normalized size (min 88 Bytes, max 20 kBytes). Per-user traffic is sporadic and irregular which makes it hard to find patterns. The graph shows that the per-user time series composed of gaps are not time aligned.

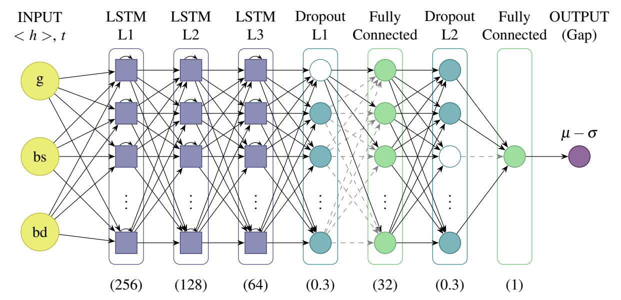


Fig. 4. Diagram illustrating the architecture of the three-stacked LSTM network employed by TRADER for traffic burst gap forecasting at the user level considering a history of N <g, bs, bd> past features.

Our approach, which improves scalability and avoids wasting SRS resources, is detailed next.

III. Traffic Prediction at Frame Granularity

We start by discussing how to forecast user-level traffic at single-frame 10-ms resolution.

A. Forecasting Model

Classical network traffic prediction aims at anticipating the volume of data transmitted in the next time slot(s). However, the nature of per-user traffic at the very short timescales we target is inherently hard to forecast: transmissions occur in rapid bursts that are irregularly spaced in time, as exemplified by Fig. 3 for five users in a sample from our real-world measurements. Moreover, the motivation set out in § II sets our forecasting problem apart from traditional traffic prediction: instead, we aim at learning when the next traffic burst generated by a user will start. We thus design input features and a neural network architecture tailored to that specific objective, as follows.

Input features. We use three input features, illustrated in Fig. 3, that can be easily collected on a per-user basis by analyzing transmission events for each registered active device.

- A *gap* (g - measured in ms) represents the time between two consecutive transmissions of the same user, and inherently indicates when the next transmission will take place. Since two frames are enough to trigger an SRS on the channel, gaps correspond to user silence periods above 20 ms. As users disconnect after 10.15 s of inactivity, gaps are upper-bounded by this value.
- The *burst size* (bs - measured in Byte) is obtained by summing all TBS generated by a user between two subsequent gaps. The resulting aggregate describes the total volume of traffic generated by a user, continuously over time.
- The *burst duration* (bd - measured in ms) is defined as the time interval elapsed between the last and the first frame with non-zero TBS within the corresponding burst size, *i.e.*, how long a burst lasts in time.

Neural network architecture. The input features above are fed to a deep neural network. We experiment with two configurations. In a first case, we employ hidden Stacked Long Short-Term Memory (LSTM) layers with multiple memory cells [13]. The rationale for this design is that LSTM recurrent neural networks (RNN) are known to perform well with time series [14], and our prediction problem can be seen as an instance of (per-user) time series forecasting, where samples map to gaps, and the goal is anticipating the magnitude of the next gap. The leaky version of the Rectified Linear Unit (ReLU) is used as the activation function for neurons in the LSTM layers, with a negative ReLU slope coefficient of 0.01 to avoid the dying ReLU phenomenon [15]. As a second option we use a pure regression model, *i.e.*, a standard feed-forward Multi-Layer Perceptron (MLP). This neural network has similarly been widely applied to time series forecasting [16], although it does not have the memory properties of LSTM. In this case we use a standard ReLU activation function since no dying ReLu problem was faced with MLP.

In both LSTM and MLP models, we experiment with different RNN depths, by stacking LSTM or fully connected layers on top of each other. Such layers are followed by a hidden fully connected layer and by an output layer with a single hidden unit for the actual prediction. We also test various configurations of the input layer, by varying the number of hidden units from 32 to 256 and halving the number of neurons at each subsequent layer.

Dropout layers are interleaved with the last two hidden layers to avoid overfitting during training [17]. Based on extensive experiments, we set the dropout layer rates to 0.3. In fact, an unconventional design choice we make is to keep dropout layers active also during testing, and perform multiple concurrent forward passes on the same test data. This returns, for each forecast instance, a distribution of predicted burst gaps instead of a single output; the mean (μ) and standard deviation (σ) of the distribution approximate those obtained via Bayesian inference in (computationally prohibitive) deep Gaussian processes, as proven by recent findings in deep neural network operation [18]. The mean and deviation provide a richer information than usual univariate prediction, and we take advantage of them for SRS scheduling. Specifically, positive errors in the prediction cause the SRS to be delayed after the start of the user transmission, and hence are not available to estimate the channel state when needed, which leads to a substantial performance degradation. This is a much more severe situation than that entailed by negative errors (of comparable magnitude), which cause an anticipation of the SRS with respect to the ideal allocation, and hence a less up-to-date (but usable) CSI. To minimize the chance of positive errors, we use the uncertainty expressed by the deviation as a *safety margin*: the predictor returns the mean minus the standard deviation of the predicted next gap for each forecast instance.

Fig. 4 summarizes the architecture for the case of a three-stacked LSTM that employs 256 neurons in the first layer. The MLP model can be obtained by replacing the LSTM layers depicted in Fig. 4 with fully connected layers.

Model training. The deep neural network receives N past observations of the three input features, *i.e.*, gap, burst size and burst duration, and aims at forecasting the following gap, so as to inform TRADER about the expectation (and uncertainty) of when the next transmission of a given user will take place.

In order to ensure the highest prediction accuracy, we test and compare different loss functions. We consider the well-known MAE and MSE, but also the Pinball-Loss (PL) proposed in [19]. Formally, the three functions are defined as:

$$\text{MAE} = \frac{1}{n} \cdot \sum_{i=1}^{n} |x_i - \overline{x}_i|, \tag{1}$$

$$\text{MSE} = \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - \overline{x}_i)^2, \tag{2}$$

$$\text{PL} = \begin{cases} (x - \overline{x}) \cdot \kappa & \text{if } x \geq \overline{x}, \\ (x - \overline{x}) \cdot (1 - \kappa) & \text{if } x \leq \overline{x}, \end{cases} \tag{3}$$

where x and $\overline{x}$ are the predicted and observed values respectively and κ is the conditional quantile: for instance, $\kappa = 0.5$ is an estimator of the conditional median. While MAE and MSE are legacy loss functions used for RNN training, the rationale for experimenting with a Pinball-Loss function is that we aim at minimizing overestimation: as explained above, avoiding positive errors is key for SRS scheduling, hence we set the Pinball-Loss threshold to $\kappa = 0.2$ to favor underestimation.

In all variants, the model is trained using the popular Adam optimizerwith a learning rate of 0.001 during 120 epochs.

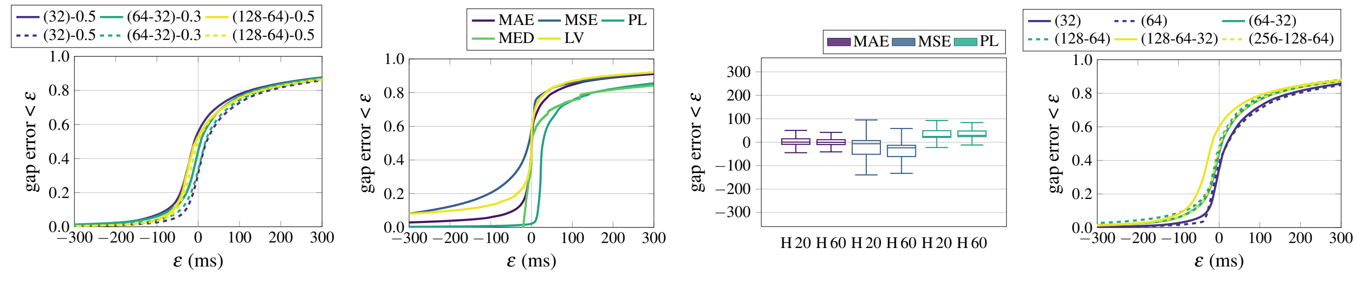
B. Prediction Accuracy

We assess the performance of the proposed deep learning predictor with the FALCON dataset (see § II-B). We use a standard 80:20 training-testing ratio to split the data, as highlighted by the different backgrounds in Fig. 1. We thus use $3,041,118$ <g, bs, bd> tuples to train and validate the neural network architecture. Then, we test the model over $760,280$ new observation samples.

Comparative evaluation. We compare the forecast accuracy in an extensive set of cases, by varying the architecture, loss function, number of stacked layers, neurons per layer and drop rates. We start by analyzing the performance of LSTM and MLP using MAE as the loss function. Fig. 5(a) summarizes the results, portrayed as the cumulative distribution of the error ϵ (with $\epsilon = x - \overline{x}$) incurred by each model during testing. We restrict the error interval to $[-300, 300]$ ms as we are particularly interested in short-term predictions, and since the vast majority of the recorded deviations are in that range. A perfect predictor would yield a step function in $\epsilon = 0$: the closer to this ideal performance is the curve, the better the forecasting model. Also, positive errors should be avoided as much as possible because, as explained before, they are particularly problematic for SRS allocation. The legend denotes LSTM with solid lines and MLP with dashed lines, and in round brackets the number of neurons followed by the drop rate. Varying all these settings did not produce a clear winner, which we ascribe to the extremely noisy nature of the input data, which does not present easy-to-learn patterns. As a result, all the predictors produce forecasts that are primarily based on inference of the statistical distribution of the burst gaps, and any neural network configuration suffices to that end.

We then analyze the quality of the gap forecast under different loss functions. In order to provide a reference for the results obtained with MAE, MSE and PL, we consider two simple statistical predictors: (i) a static forecast (MED) corresponding to the median of the gap distribution observed over the training dataset, and (ii) a naive model that uses the last value (LV) of observed gap as the prediction for the next gap. Other than serving as baselines, these benchmarks help us understand if a deep learning approach is strictly necessary, or if much simpler options are also valid. Fig. 5(b) summarizes the results for LSTM with 1 stacked layer, 64 neurons and drop rate 0.3. We observe that the predictor using MAE as the loss function outperforms all other predictors including the simple ones, with 68% of the cases falling within the interval $[-100, 100]$ ms and 91% of the cases falling within the interval $[-300, 300]$ ms. This allows to operate sufficiently well in the regime of interest, *i.e.*, with the small gaps seen in Fig. 2, whose accurate prediction can inform decisions on whether to trigger an SRS or not. For longer gaps, such a decision can be revisited later with lower urgency. MAE is more robust to outliers than MSE, and this is the reason for its better performance. PL with $\kappa = 0.2$ favors underestimation: this can be appreciated in the curve, as the incidence of negative values of ϵ is very small. However, the error curve is shifted from $\epsilon = 0$, which means that the price paid in terms of underestimation is excessive, and would not be of help for SRS allocation.[3] The MED estimator

[3]This behavior could be partially alleviated by manually shifting the curve, but this overcomplicates the model, makes it less automated, and still would not outperform the simpler MAE: see the curve for positive values of ϵ.



(a) Comparison of LSTM and MLP. (b) Prediction error distributions. (c) Impact of past observations. (d) Impact of number of stacked layers.

Fig. 5. Performance of the burst gap prediction, under different (a) LSTM and MLP architectures, (b) loss functions, (c) history sizes, and (d) stacked layers.

drops to zero overestimation for all those gaps bigger than the observed median, which is desirable, but the accuracy of underestimations is significantly lower than MAE. As a result, the gain obtained by preventing overestimation is achieved at the cost of a lower accuracy in the region for which we are interested in obtaining accurate predictions. Finally, LV performs similarly to MAE for positive values of ϵ, but its accuracy in overestimation is significantly worse, which justifies building a learning model for this problem.

Model parametrization. Having confirmed that a simple deep learning architecture trained with a MAE loss function outperforms other approaches, we investigate how its parameters affect the quality of the forecast.

We first evaluate the impact of different input lengths, considering 20 and 60 previous samples fed to the model. The minimum number of past observations required by the LSTM network has been set to 20 since, after several experiments, such length has proven to be effective in obtaining an accurate prediction. Furthermore, we decided to increase the history sequence length up to 60, in order to evaluate the impact of a larger number of past observations on the prediction accuracy. Fig. 5(c) shows that the accuracy of MAE is very similar with 20 or 60 samples: there are minor differences in the interquartile range, in favor of a history of 20. We also include the results for the other predictors, MSE and PL for which the median of the distribution is not centered in zero.

As a second test, we assess the impact of the depth of the neural network structure in the forecasting performance. To this end, we compare LSTM architectures with a varying number (1, 2 or 3) of stacked layers, and different numbers of neurons per layer. Fig. 5(d) does not show relevant performance differences, which makes us favor the simpler variant with 1-stacked layer and 64 neurons because of the better accuracy in over-estimation.

Summary. Based on extensive experiments, the gap-burst time series under analysis do not fall into the category for which LSTM models perform exceedingly well, most likely because of the highly irregular timings of transmission bursts at the LTE frame granularity. Still, a very simple 1-stacked LSTM model with 64 input neurons can learn the basic statistics of the gap distribution better than simple predictors, and is computationally less demanding[4] than MLP architectures. In the light of these results, we assist the SRS scheduling with a MAE predictor, configured with a 1-stacked LSTM layered architecture with 64 neurons, fed with 20 past samples.

[4]In our tests, training of a 1-stacked LSTM network is ~ 1.16 times faster than the corresponding MLP model.

IV. The TRADER Resource Allocation Strategy

We now discuss the aperiodic SRS resource allocation problem. The objective is to minimize the age of the channel estimate, *i.e.*, the time elapsed between the last SRS that was scheduled for a user and its next traffic allocation. Call $\mathcal{U}$ the set of RRC connected UEs, where u denotes a UE, $u \in \mathcal{U}$. Let t be the current time (or equivalently the current frame number) and f_u be the (unknown) LTE frame that carries the next traffic allocation for u (with $t < f_u$). f_u occurs after a gap τ_u from the last traffic allocation for u that we denote with z_u, i.e., $f_u = z_u + \tau_u$. Let R be the amount of SRS resources available in each TDD frame and let

$$r_{u,t} = \begin{cases} 1 & \text{if an SRS for } u \text{ is triggered in } t, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

The time of the most recent SRS triggered for UE u, l_u is:

$$l_u = \arg\max_{j \le t} \{ j | r_{u,j} = 1 \}. \tag{5}$$

At time t, the objective is to allocate SRS resources $r_{u,t}$ to minimize the maximum age of channel estimate:

$$\min \max_{u \in \mathcal{U}, f_u - t = 2} (f_u - t - l_u), \tag{6}$$

provided that the future allocation is not far ahead in time than 2 frames and without exceeding the R SRS signals available in each frame:

$$\sum_{u=1}^{U} r_{u,t} \le R. \tag{7}$$

At the present time t, the arrival of future traffic f_u is unknown (recall that $f_u > t$). We thus leverage the forecasting methodology presented in § III to obtain an estimate $\widetilde{f}_u$ of such time. The term $\widetilde{f}_u$ can not be directly estimated, but our gap prediction methodology allows to estimate $\widetilde{\tau}_u$ and given that the former traffic allocation z_u is known, we can compute $\widetilde{f}_u = z_u + \widetilde{\tau}_u$. We recall that our deep learning model returns both the expectation (here denoted by $\overline{\tau}_u$) and the standard deviation (σ_u) of the inter-burst gap, via multiple concurrent forward passes with active dropout layers. We leverage the standard deviation as a measure of the uncertainty of the prediction, and set $\widetilde{f}_u = t + \widetilde{\tau}_u$, where $\widetilde{\tau}_u = \overline{\tau}_u - \sigma_u$: in this way, a *safety* margin informed by the model uncertainty anticipates the SRS, and limits the chances that an incorrect forecast triggers the SRS too late with respect to the actual traffic arrival. We then reformulate (6) as:

$$\min_{u \in \mathcal{U}, \widetilde{f}_u - t = 2} \max (\widetilde{f}_u - t - l_u). \tag{8}$$

Algorithm 1 summarizes the workflow of TRADER. At time t, the output of the algorithm is a set of UEs $\mathcal{X} \subseteq \mathcal{U}$ for which an SRS is triggered. $\mathcal{X}$ depends on the amount of resources R available each frame. We store in U the information about the last triggered SRS l_u, the corresponding age of estimate a_u and the last frame with traffic allocation z_u for each *active* UE. The latter component allows determining whether the current UE should be removed from the set of active UEs or not. For this, we compute the age of last transmission as the difference between the current frame t and z_u, and, if this difference exceeds T, u is considered not eligible for allocation (line 3).

A new UE joins the system if, during t, there is traffic allocated for him/her. As no past SRS is available in this case, $l_u = \varnothing$ and we include in U a new tuple $< a_u, l_u, z_u >$ with infinite age for the last estimate and last SRS; the current frame is set as the last frame with traffic allocation (lines 4-5). Then, u acquires the highest priority to be scheduled in the current round. For UEs already in the system and for which predictions are available (*i.e.*, with at least N past samples), we compute the age of the last estimate as the future allocation $\widetilde{f}_u$ or the current frame t with traffic. The future allocation is given by the gap estimate τ computed from the last traffic allocation z_u (lines 8-13). If the future allocation is two frames ahead from t, the SRS should be scheduled at time t. Otherwise, there is no urgency and the SRS can be scheduled later on (line 9). We then sort U in descending order of a_u (line 19) and schedule the first UEs in the set until the available R resource units are used. This builds the schedule $\mathcal{X}$ and for these users we update the information in U by setting the age of estimate to 0 and the last scheduled SRS as the current frame t (lines 20-24).

We benchmark TRADER against a simpler Round Robin heuristic which does not make use of predictions. Round Robin's workflow is also described in Algorithm 1 with the exception of the if condition (lines 7-11). During each iteration, Round Robin computes the age of the estimate of u with respect to the current frame t and schedules UEs accordingly. When no predictions are available for the reasons discussed above, TRADER falls back to Round Robin and blindly triggers SRS for those UEs that currently have the highest age.

Algorithm 1 TRADER Resource Allocation Algorithm

```
    Input: $T$ RNTI refresh timer, $R$ SRS resources per frame,
           $\mathcal{U}$ set of connected UEs
    Output: $\mathcal{X}$ set of UE with SRS triggered in $t$
    Var: $a_u$ age of last estimate, $l_u$ last SRS, $z_u$ last traffic allocation
 1: for each $u \in \mathcal{U}$ do
 2:     $U \leftarrow \{0\}$
 3:     if $t - z_u < T$ then                    ▷ Check RNTI Refresh expiration
 4:         if $l_u == \emptyset$ then             ▷ New UE with no previous SRS
 5:             $a_u \leftarrow \infty$, $l_u \leftarrow \infty$, $z_u \leftarrow t$
 6:         else                                          ▷ Existing active UE
 7:             if prediction available then
 8:                 $\widetilde{f}_u = z_u + \overline{\tau}_u$
 9:                 if $\widetilde{f}_u - t = 2$ then      ▷ $\widetilde{f}_u$ requires SRS at $t$
10:                     $a_u \leftarrow \widetilde{f}_u - l_u$
11:                 end if
12:             else                                   ▷ No predictions available
13:                 $a_u \leftarrow t - l_u$
14:             end if
15:         end if
16:         $U[u] \leftarrow < a_u, l_u, z_u >$            ▷ Update UE's information
17:     end if
18: end for
19: sort $U$ in descending order of $a_u$
20: for $r \leftarrow 1$ to $R$ do                                  ▷ Schedule SRS
21:     $\mathcal{X} \leftarrow \mathcal{X} \cup \{U[r]\}$
22:     $a_r \leftarrow 0$, $l_r \leftarrow t$
23:     $U[r] \leftarrow < a_r, l_r >$
24: end for
```

V. Performance Evaluation

A. Methodology and Metrics

We benchmark TRADER by comparing its performance against both the Round Robin heuristic and an Oracle. The latter is an omniscient predictor with perfect knowledge on future traffic allocations, including the gaps and bursts.

For a fair comparison, we evaluate TRADER, Oracle and Round Robin under the same network conditions, *i.e.*, we do trace-driven simulation using as input the test sets of the LTE traces (see § II). For the number of SRS resources for each 10 ms frame we consider 64, 32 and 16 signals. 64 represents the number of SRS signals at disposal when the TDD system works with a MIMO configuration of 2T4R (2 Transmitter and 4 Receiver), *transmissionComb* equal to 2 and *cyclic shift* equal to 4. Such setting allows to assign 8 SRS signals in one symbol. Within a 10 ms timeframe, the number of symbols available for SRS is 8, which leads to a total of 64 SRS signals.

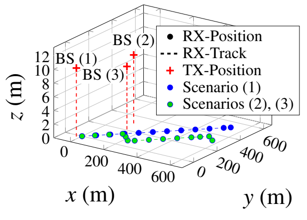
For evaluation, we focus on the following three metrics.

Delay: given our objective of obtaining fresh CSI as often as possible, this metric measures how close an SRS s is with respect to the frame t that carries actual traffic for the UE. The best result possible happens when $t - s = 2$, which means that the SRS was triggered timely.

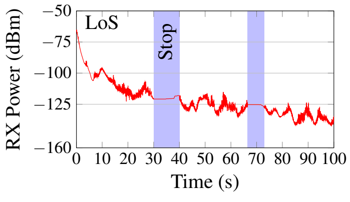
Allocation within coherence time: this metric measures how often the SRS is triggered within the channel coherence time T_c of data traffic. During T_c, the channel is assumed to be constant, thus gathering a channel estimation with SRS within T_c yields the highest utility. T_c can be approximated as $T_c = 9/16\pi f_{max}$, where f_{max} is the maximum frequency shift that is proportional to the velocity v of the UE and the carrier frequency f_c: $f_{max} = \frac{v}{c} \cdot f_c$, where c is the speed of light [20].

Impact on Received Signal Power (RSP): this metric quantifies the gain that TRADER provides over Round Robin.

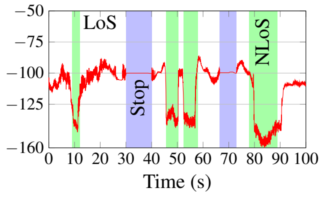
To compute the two last metrics, we employ the well known QuaDRiGa channel model [21] and set up three different urban scenarios. For each one, the UE moves for 100 s with the same sequence of speed variations (the maximum speed is 10 m/s) and stops in two parts of the trajectory. We characterize both Line of Sight (LoS) and Non-Line of Sight (NLoS) propagation by using the `3GPP_3D_UMa_LOS` and `3GPP_3D_UMa_NLOS`, where UMa stands for Urban



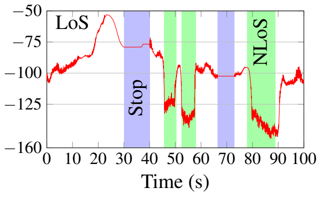
(a) Layout



(b) Channel profile Scenario (1)

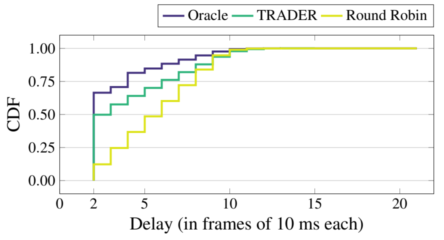


(c) Channel profile Scenario (2)

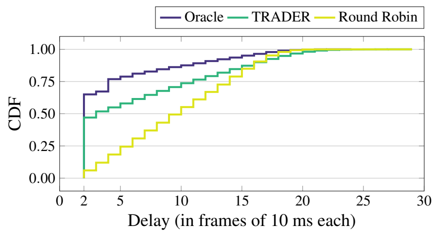


(d) Channel profile Scenario (3)

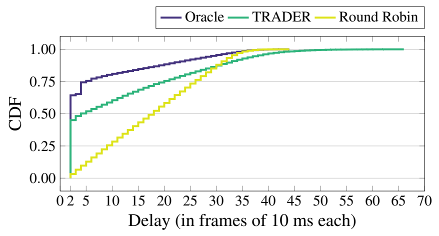
Fig. 6. The QuaDRiGa settings



(a) 64 SRS signals each 10 ms



(b) 32 SRS signals each 10 ms



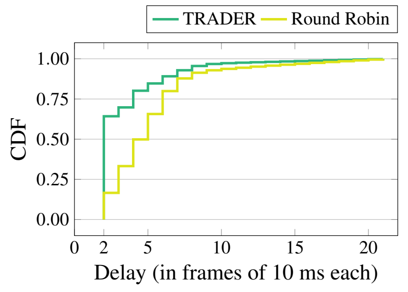
(c) 16 SRS signals each 10 ms

Fig. 7. Evaluation of delay for different amount of resources for the 3 h long trace

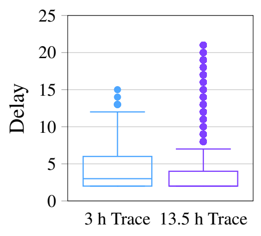
Macrocell deployment. The BS is at a height of 10 m, with a carrier frequency in the 2 GHz band. Scenario (1) in Fig. 6(b) is the simplest: the UE moves in a straight line away from the BS. As a result, its received power (RX) slowly decreases over time. Scenario (2) in Fig. 6(c) is more complex: the UE moves on a trajectory with a sequence of turns and encounters NLoS areas on its way. This makes the RX power profile more variable. Finally, Scenario (3) in Fig. 6(d) is an elaboration of Scenario (2) where the BS is closer to the same UE's trajectory and the NLoS period corresponding to second 10 of the trace is missing. As a result, the trajectory of Scenario (3) is characterized by relatively high RX power for the first part until second 40. We sample the channel coefficients with a granularity of 10 ms, corresponding to the frame duration.

B. Results

Delay. Fig. 7 compares Oracle, TRADER and Round Robin and shows CDF curves for the delay for the two datasets and for different resources reserved for SRS in each 10 ms frame. Obviously, by lowering the amount of SRS resources, the UEs obtain SRS on average less frequently, depending on the number of simultaneously active UEs (see Fig. 1). As a consequence, the tail of delay assumes the highest values when the amount of available resources is the lowest. With 64 SRS per frame (see Fig. 7(a)), TRADER provides intermediate gains between Round Robin and Oracle. TRADER triggers more often SRS exactly with 2 frames of delay (in 50% of the cases) than Round Robin. With 32 SRS each 10 ms (see Fig. 7(b)), we observe that TRADER obtains higher gains over Round Robin with respect to the former case: in nearly 50% of the cases the delay is 2 frames for TRADER and 10 frames for Round Robin. However, we note that for a delay of 15 frames or more, Round Robin marginally outperforms TRADER. By spending SRS close to when the predictions occur, TRADER subtracts resources from the one or more UEs that should have been scheduled according to the principle of minimization of the age of last estimate. These have to be re-scheduled later, thus increasing their delay. Such behaviour becomes even more evident with only 16 SRS signals each 10 ms (see Fig. 7(c)). Here the tradeoff that TRADER enforces in obtaining a better delay on the short term at the expense of some degradation at longer delays is more evident.



(a) Delay



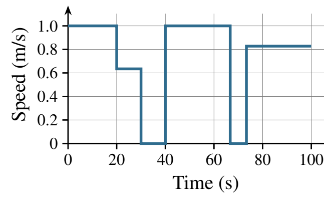
(b) Trace comparison

Fig. 8. Evaluation delay for the 13.5 h long trace of TRADER over Round Robin and comparison among the two traces

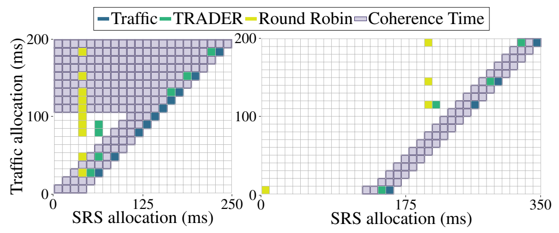
Fig. 8(a) evaluates the delay of TRADER and Round Robin for the 13.5 h long dataset. As the results are in line with what we obtained for the 3-h long trace, we only provide the result using 64 SRS signals each 10 ms. Fig. 8(b) instead compares the results obtained with the different traces. Given the different traffic patterns (see Fig. 1), such a comparison provides an insight into the gains of TRADER in such different scenarios. We observe that in the longer trace, TRADER guarantees more often the lowest delay at the expense of a longer tail (see the outliers). Vice versa, on the shorter trace, with TRADER the tail of long delays is considerably reduced.

Allocation within T_c. We set $f_c = 2$ GHz and determine T_c according to the velocity v shown in Fig. 9(a). Fig. 9(b) shows the duration of T_c, data traffic and SRS allocation for TRADER and Round Robin. TRADER triggers SRS during coherence time more often than Round Robin especially during mobility when the coherence time is shorten than when the UE stands still. We crosscheck for all the 9250 UEs of the 3 h long trace: Fig. 9(c) confirms that this result holds in general.

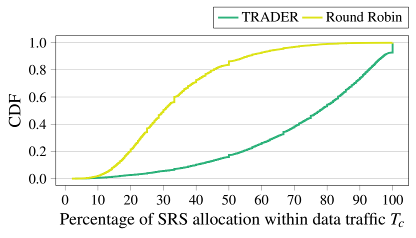
Impact on RSP. We now assess the channel gains due to the reduced SRS age. Fig. 10 shows that TRADER outperforms



(a) User speed

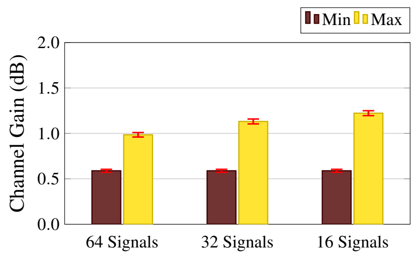


(b) Traffic and SRS allocation with respect to coherence time

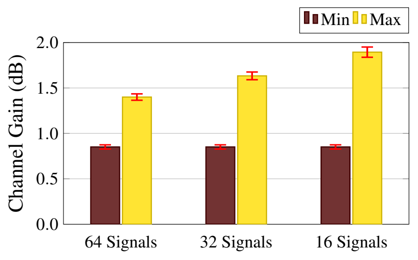


(c) CDF

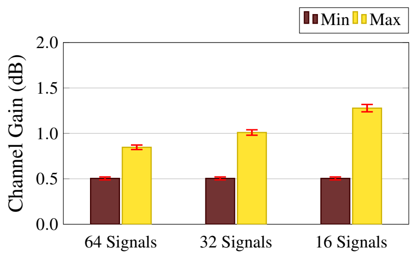
Fig. 9. Evaluation of the gain TRADER obtains over Round Robin in allocating SRS within the channel coherence time of actual data traffic



(a) Scenario (1)



(b) Scenario (2)



(c) Scenario (3)

Fig. 10. Evaluation of the channel gain TRADER obtains over Round Robin for the 3 h long trace

Round Robin (we include mean and 95% confidence intervals). Specifically, for the two schemes we use the minimum and maximum frame delay from the above results for each of the resources considered (64, 32 and 16 signals for each 10 ms frame). For example, in Fig. 7(a)) the maximum delay is 3 frames and the minimum is 1 frame. Then we compute $|c_{\text{TRADER}}(t) - c_{\text{Round Robin}}(t)|$ for each instant t of the channel profiles (see Fig. 6(b), 6(c) and 6(d)).

Fig. 10(a), 10(b) and 10(c) show the results for the OWL dataset. We observe that the average channel gains increase as the amount of SRS signals available each 10 ms frame diminish. Fig. 10(a) shows that for a simple scenario with linear trajectory the variability in channel gain is minimal. The difference of the means obtained with minimum and maximum frame delay spans from 0.4 dB to 0.6 dB (for 64 and 16 respectively), while for Scenario (3) the differences are 0.3 dB to 0.8 dB (for 64 and 16 respectively). The highest gains occurs for Scenario (2) which is the most complex one.

VI. Related Works

Relevant to our work are studies on the prediction of mobile network traffic, and on the allocation of SRS resources. We discuss hereafter the novelty of TRADER with respect to previous efforts in those two areas.

Mobile network traffic prediction. Real-time data traffic forecast are a paramount input to emerging strategies in traffic engineering and resource management that take advantage of the increasing virtualization of mobile networks, as repeatedly demonstrated by many recent studies [22], [23], [24]. In this context, traditional models relying on information theory [25], Markovian [26] or autoregressive [27] approaches have been supplanted by deep learning architectures.

Specifically, a variety of neural networks have been proposed to predict the traffic demands observed at individual BSs [5], [28] or antenna sectors [29], possibly over long time horizons [6], and separately across mobile services [30]. All these solutions aim at estimating traffic volumes over timescales in the order of minutes, whereas our focus is on much faster dynamics at the millisecond level.

The literature on network state prediction at the short timescales we target is much thinner. Previous research has mostly proposed deep learning predictors that operate on time steps of seconds or less for physical layer indicators, such as bandwidth [7], transmission inactivity [31], signal strength [32], uplink throughput [8], Physical Resource Blocks (PRBs) [33], or channel state [34].Unlike these works, we are interested in user-generated traffic, and not physical layer properties.

The study that is the closest to ours in spirit aims at mobile traffic volume forecasting over time steps of tens of LTE Transmission Time Intervals (TTI), *i.e.*, tens of ms, by using LTE Physical Downlink Control CHannel (PDCCH) information [9]. However, neither this research nor any of those listed before addresses the problem of anticipating traffic burst duration or gaps, as done by our forecasting model.

SRS allocation. Standardization bodies are still debating on the amount of dedicated resources and allocation strategies for SRS. In the light of such uncertainty, several recent works have investigated optimizations to the use of SRS when there is a very large number of users for which channel estimation must be performed in parallel. The approaches proposed to date have tackled the problem from different angles.

Some studies have focused on coordinating uplink SRS allocation across neighboring base stations to mitigate pilot contamination, by using fractional reuse [35].

A different line of research has considered enhancing Channel Estimation Capacity from SRS information, *e.g.*, by using deep learning tools, hence inherently improving the utilization of SRS sequences [36]. Also, multi-user grouping has been proposed as a strategy to minimize SRS resource requirements, by bringing together users based on channel state information [37], or channel correlation [38].

However, none of the works above addresses the problem of anticipatory SRS allocation based on per-user traffic prediction. Further, all the above works use periodic SRS while both TRADER and Round Robin mechanisms use aperiodic SRS which makes all previous studies fully orthogonal to ours.

VII. Conclusions

For future mobile networks, obtaining fresh and accurate CSI is especially important. Technologies such as massive MIMO impose a high overhead to acquire CSI, which scales with the number of antenna elements involved in the measurement. In this work, we tackle the problem of efficient configuration of SRS pilots that are used to obtain accurate CSI for downlink MIMO precoding in TDD systems. Specifically, we design TRADER, a TRAffic-DrivEn Resource allocation framework that leverages per-user ms-level traffic forecasts for aperiodic SRS scheduling. TRADER uses an LSTM network to predict the idle time between subsequent user traffic allocations that are essential to determine a good SRS schedule. We extensively evaluate the performance of the predictor and of TRADER with real world mobile data. Unlike an aperiodic round robin heuristic, the trace-driven simulations show that TRADER is able to trigger more often SRS right before the actual traffic and performs well in high load scenarios like the ones tested. The more accurate scheduling of SRSs close to future user traffic translates into channel gains of up to 2 dB.

Acknowledgments

This work is partially supported by the Juan de la Cierva grant from the Spanish Ministry of Science and Innovation (IJC2019-039885-I), by the European Union's Horizon 2020 research and innovation programme under grant agreement no.101017109 "DAEMON," and by the Madrid Regional Government through the TAPIR-CM program (S2018/TCS-4496).

References

[1] K. B. Letaief *et al.*, "The roadmap to 6G: AI empowered wireless networks," *IEEE Communications Magazine*, vol. 57, no. 8, pp. 84–90, 2019.
[2] J. Zhang *et al.*, "Prospective multiple antenna technologies for beyond 5G," *IEEE JSAC*, pp. 1–24, 2020.
[3] E. Hong *et al.*, "A study on channel estimation algorithm with sounding reference signal for TDD downlink scheduling," in *Proc. of IEEE PIMRC*, 2017, pp. 1–6.
[4] 3GPP TS 36.211, "Evolved universal terrestrial radio access (E- UTRA); physical channels and modulation," 2020, (Release 16).
[5] J. Wang *et al.*, "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach," in *Proc. of IEEE INFOCOM*, May 2017, pp. 1–9.
[6] C. Zhang *et al.*, "Long-term mobile traffic forecasting using deep spatio-temporal neural networks," in *Proc. of ACM Mobihoc*, 2018, p. 231–240.
[7] C. Yue *et al.*, "LinkForecast: Cellular link bandwidth prediction in LTE networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 7, pp. 1582–1594, 2018.
[8] J. Lee *et al.*, "PERCEIVE: Deep learning-based cellular uplink prediction using real-time scheduling patterns," in *Proc. of ACM MobiSys*, 2020, p. 377–390.
[9] H. D. Trinh *et al.*, "Mobile traffic prediction from raw data using LSTM networks," in *Proc. of IEEE PIMRC*, Sep. 2018, pp. 1827–1832.
[10] C. Fiandrino *et al.*, "A machine learning-based framework for optimizing the operation of future networks," *IEEE Communications Magazine*, vol. 58, no. 6, 2020.
[11] R. Falkenberg *et al.*, "FALCON: An accurate real-time monitor for client-based mobile network data analytics," in *Proc. of IEEE GLOBECOM*, Dec 2019, pp. 1–7.
[12] N. Bui *et al.*, "OWL: A reliable online watcher for lte control channel measurements," in *Proc. of ACM Workshop on All Things Cellular: Operations, Applications and Challenges*, 2016, pp. 25–30.
[13] A. Graves *et al.*, "Speech recognition with deep recurrent neural networks," in *Proc. of IEEE ICASSP*, 2013, pp. 6645–6649.
[14] Y. Hua *et al.*, "Deep learning with Long Short-Term Memory for time series prediction," *IEEE Communications Magazine*, vol. 57, no. 6, pp. 114–119, 2019.
[15] A. L. Maas *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. of ICML*, vol. 30, no. 1, 2013, p. 3.
[16] T. Koskela *et al.*, "Time series prediction with multilayer perceptron, fir and elman neural networks," in *Proc. of World Congress on Neural Networks*, 1996, pp. 491–496.
[17] N. Srivastava *et al.*, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, p. 1929–1958, Jan. 2014.
[18] Y. Gal *et al.*, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proc. of ICML*, 2016, p. 1050–1059.
[19] I. Steinwart *et al.*, "Estimating conditional quantiles with the help of the pinball loss," *Bernoulli*, vol. 17, 02 2011.
[20] U. Karabulut *et al.*, "Low complexity channel model for mobility investigations in 5G networks," in *Proc. of IEEE WCNC*, 2020, pp. 1–8.
[21] S. Jaeckel *et al.*, "QuaDRiGa: A 3-D multi-cell channel model with time evolution for enabling virtual field trials," *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 6, pp. 3242–3256, 2014.
[22] I. Alawe *et al.*, "Improving traffic forecasting for 5G core network scalability: A machine learning approach," *IEEE Network*, vol. 32, no. 6, pp. 42–49, Nov 2018.
[23] J. Ayala-Romero *et al.*, "vrAIn: A deep learning approach tailoring computing and radio resources in virtualized RANs," in *Proc. of ACM Mobicom*, 2019, pp. 1–16.
[24] L. Chen *et al.*, "Data-Driven C-RAN Optimization Exploiting Traffic and Mobility Dynamics of Mobile Users," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2020.
[25] R. Li *et al.*, "The prediction analysis of cellular radio access network traffic: From entropy theory to networking practice," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 234–240, 2014.
[26] M. Z. Shafiq *et al.*, "Characterizing and modeling internet traffic dynamics of cellular devices," in *Proc. of ACM SIGMETRICS*, 2011, p. 305–316.
[27] F. Xu *et al.*, "Big data driven mobile traffic understanding and forecasting: A time series approach," *IEEE Transactions on Services Computing*, vol. 9, no. 5, pp. 796–805, 2016.
[28] X. Wang *et al.*, "Spatio-temporal analysis and prediction of cellular traffic in metropolis," *IEEE Transactions on Mobile Computing*, vol. 18, no. 9, pp. 2190–2202, 2019.
[29] D. Bega *et al.*, "DeepCog: Optimizing resource provisioning in network slicing with AI-based capacity forecasting," *IEEE JSAC*, vol. 38, no. 2, pp. 361–376, 2020.
[30] C. Zhang *et al.*, "Multi-service mobile traffic forecasting via convolutional long short-term memories," in *Proc. of IEEE M&N*, 2019, pp. 1–6.
[31] P. Brand *et al.*, "Adaptive predictive power management for mobile LTE devices," *IEEE Transactions on Mobile Computing*, pp. 1–18, 2020.
[32] A. Kulkarni *et al.*, "DeepChannel: Wireless channel quality prediction using deep learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 443–456, 2020.
[33] C. Gutterman *et al.*, "RAN resource usage prediction for a 5G slice broker," in *Proc. of ACM MobiHoc*, 2019, p. 231–240.
[34] C. Luo *et al.*, "Channel state information prediction for 5G wireless communications: A deep learning approach," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 227–236, 2020.
[35] L. G. Giordano *et al.*, "Uplink sounding reference signal coordination to combat pilot contamination in 5G massive MIMO," in *Proc. of IEEE WCNC*, 2018, pp. 1–6.
[36] T. Zeng *et al.*, "Channel estimation capacity enhancement for multigroup multicasting multimedia networks with DnCNN," *IEEE Access*, vol. 7, pp. 177 616–177 627, 2019.
[37] T. Zeng *et al.*, "CSI-RS based joint grouping and scheduling scheme with limited SRS resources," in *Proc. of IEEE PIMRC*, 2018, pp. 1–6.
[38] Q. Zhang *et al.*, "SRS limited user grouping scheduling algorithm for downlink massive MIMO systems," in *Proc. of IEEE WCNC*, 2019, pp. 1–6.