

Multi-Task Learning as enabler for General-Purpose AI-native RAN

Hasan Farooq, Julien Forgeat, Shruti Bothe, Kristijonas Cyras and Md Moin
Ericsson Research, Santa Clara, 95054 CA, USA

{hasan.farooq, julien.forgeat, shruti.bothe, kristijonas.cyras, md.moin.uddin.chowdhury}@ericsson.com

Abstract—The realization of data-driven AI-native architecture envisioned for 6G and beyond networks can eventually lead to multiple machine learning (ML) workloads distributed at the network edges driving downstream tasks like secondary carrier prediction, positioning, channel prediction etc. The independent life-cycle management of these edge-distributed independent multiple workloads sharing a resource-constrained compute node e.g., base station (BS) is a challenge that will scale with denser deployments. This study explores the effectiveness of multi-task learning (MTL) approaches in facilitating a general-purpose AI-native Radio Access Network (RAN). The investigation focuses on four RAN tasks: (i) secondary carrier prediction, (ii) user location prediction, (iii) indoor link classification, and (iv) line-of-sight link classification. We validate the performance using realistic simulations considering multi-faceted design aspects of MTL including model architecture, loss and gradient balancing strategies, distributed learning topology, data sparsity and task groupings. The quantification and insights from simulations reveal that for the four RAN tasks considered (i) adoption of customized gate control-based expert architecture with uncertainty-based weighting makes MTL perform either best among all or at par with single task learning (STL) (ii) LoS classification task in MTL setting helps other tasks but its own performance is degraded (iii) for sparse training data, training a single global MTL model is helpful but MTL performance is on par with STL (iv) optimal set of group pairing exists for each task and (v) partial federation is much better than full model federation in MTL setting.

Index Terms—Multi-task Learning; RAN, mixture of experts; partial federated learning.

I. INTRODUCTION

AI-native RAN is a trending concept that is spearheading the evolution of wireless mobile networks. This concept makes AI pervasive in the entire RAN architecture. The reason being AI's pronounced success for virtually every RAN design/optimization aspect [1]. These AI driven RAN workloads can be deployed at the edge of networks e.g., centralized unit (CU)/distributed unit (DU). Edge-deployed ML workloads, in contrast to traditional cloud-centric architecture, are appealing for fulfilling the latency, scalability, reliability, and privacy needs of beyond-5G applications. Additionally, they present an attractive solution for privacy-focused multi-vendor deployment scenarios and data regulatory compliance.

Nevertheless, a notable limitation in the majority of AI-driven RAN studies is the tendency to address specific RAN problems, features, or use cases in isolation. In such cases, AI is often customized to suit a particular use case, referred to as specialized AI models. Implementing solutions with specialized

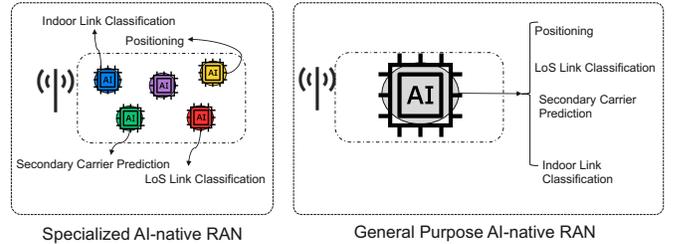


Fig. 1: Specialized AI-native RAN vs General AI-native RAN.

AI models in live RANs may result in an uncontrolled proliferation of specialized ML models per downstream task (radio feature) that will increase both RAN complexity and operational expenditure. In particular, the independent life-cycle management (LCM) of these edge-distributed independent multiple workloads sharing a resource-constrained compute node (BS, CU/DU) will be challenging in terms of availability of labelled data and compute-memory resources that will scale with denser deployments. Contrary to this, general purpose AI-native RAN vision is much more desirable wherein a single AI algorithm would have the capability to learn and manage a wide spectrum of networking operations, spanning the whole protocol stack. Achieving this involves designing an AI algorithm that can concurrently control multiple RAN tasks (Fig. 1).

Multi-task learning (MTL) is one such paradigm that can be used to train a ML model to perform multiple RAN related tasks. MTL jointly learns multiple related tasks using a single model. MTL draws inspiration from human learning, where individuals frequently leverage knowledge acquired from prior tasks to facilitate the learning of a new task. MTL has been shown to improve parameter-efficiency and inference speed compared with learning a separate model for each task. It can also enable features for one task to be available for another and emphasizes generalizable features that are common across tasks. With an MTL approach, different tasks need to share some common structure otherwise it is usually better to use single-task learning. Fortunately, in the context of RAN, there are many tasks with shared structure that may also need concurrent execution. Even if tasks are seemingly unrelated, the laws of physics that govern radio propagation are same for all tasks. In this study, we aim to address three key research questions: (i) *What is the impact of the design space of state-of-the-art MTL approaches in terms of model architecture and weighting strategies on the overall performance of RAN-related*

tasks? (ii) In various scenarios, which among the three modes of distributed learning topology—local model, global model, or partial federated model—is most appropriate? (iii) How does the sparsity of training data and the different groupings of RAN tasks influence the performance of MTL?

A. Related Work

Some works have used ML for RAN tasks considered in this paper, such as [2] for secondary carrier prediction, [3] for location prediction, [4] for indoor/outdoor link classification, and [5] for LoS link classification. However, the conventional approach in these AI for RAN studies is to design a specialized independent ML (STL) model for each task. As opposed to STL, MTL is a promising paradigm that has been largely studied in the domain of computer vision and natural language processing related use-cases only as evident from survey in [6]. Recently very few works have explored MTL paradigm in context of mobile networks like in [7] which has explored MTL to address two RAN use cases, handover management and initial modulation selection. Likewise the work in [8] used MTL for predicting user environment (indoor or outdoor) and mobility (low, medium or high) state using drive test data. The work in [9] has used MTL to simultaneously learn modulation and signal classification tasks. All of these studies have used neural network (NN) based hard parameter sharing architecture without accounting for imbalanced losses or distributed learning topology that can have substantial effect on learning performance. To the best of authors' knowledge, performance analysis of state-of-the-art MTL approaches for RAN tasks considering its key design aspects like model architecture, loss balancing, distributed learning topology is missing in literature. This paper aims to fill this gap.

B. Contributions

The contributions of this paper can be summarized as follows:

- 1) We demonstrate how to design MTL approach to improve RAN related task performance by performing simulations to evaluate the behavior of four RAN tasks in multi-task learning context considering MTL model architecture, loss and gradient balancing strategies, and data sparsity.
- 2) We demonstrate how MTL performance varies when training independent MTL models, a single global MTL model and partial federated MTL model with two variants depending upon the order in which local parameters are updated.
- 3) We present experimental results on how different grouping combinations of tasks affect overall MTL performance.

The rest of the paper is organized as follows: Section II describes the MTL paradigm in context of four RAN use-cases. The performance analysis and associated discussion is presented in Section III while Section IV concludes the paper.

II. MULTI-TASK LEARNING FOR RAN DISTRIBUTED WORKLOADS

In the context of T learning tasks, each having their own dataset $D_t = (x, y)_t$ where x are input features, y are labels and L is the model loss. Each task is characterized by a tuple $\{p(x)_t, p(y|x)_t, L_t\}$ and each task should differ in at least one of them. For the scope of this paper, we have considered four ML driven RAN tasks running in each of the base stations:

- 1) Secondary Carrier Prediction (SC): user equipments (UEs) use AI model to predict coverage on the higher band based on measurements at the serving lower band carrier. This avoids the need to perform measurements on a secondary carrier, thus reducing the energy consumption, throughput degradation and the delay.
- 2) Positioning (PS): location estimation/prediction enables advanced Location-based Services (LBS) and is also helpful for emergency (911) scenarios. It is also helpful in beam management (e.g., reducing search space of initial beam search procedure) and thus, improves the quality of service experienced by users.
- 3) Indoor/Outdoor link Classification (IN): It makes it possible to exploit the environmental context to enhance network operations, in terms of better QoE e.g., handover to indoor small cells, slice selection etc.
- 4) LoS/NLoS link Classification (LOS): identification of LoS/NLoS links in higher bands e.g., mmWave communication is important for range-based localization since utilizing NLoS measurements directly as LoS measurements may result in large localization errors.

We have framed the MTL problem as follows: given the reference signal received power (RSRP) on LTE carrier for all cells, the objective is to predict: (a) RSRP on NR carrier for all cells (b) distance to BSs (c), classify if UE is indoor and (d) classify if user equipment (UE) has LoS link on NR carrier for all cells. In real networks, this dataset can be built by BSs using 3GPP standardized Minimization of Drive Test reporting (MDT) feature. BSs receive MDT reports from UEs that have information about their measured RSRPs for both intra and inter-frequency bands carriers (input features, secondary carrier prediction labels) and using it together with geographical datasets containing 3D Earth terrain and building layouts of the city to capture labels for location prediction and LoS link classifications tasks. Once the MTL model is trained, it can run in BSs requiring only intra-frequency radio measurements from UEs and inferring labels for the four tasks to be consumed in downstream tasks e.g., instructing UE to handover to a specific beam. In our scenario, the input features distribution would be the same across all tasks, as the RSRP levels for intra-frequency cells serve as the input features for each task. However, there is variation in the conditional distribution of labels given input features across tasks, as the labels differ among them. Loss function is also different across the tasks.

Let each task t has a training dataset D_t , MTL aims to learn a model on D_t to learn T tasks together to improve the learning

of a model for each task by using knowledge in all or some of other tasks. The most common way of doing MTL is through hard parameter sharing wherein NN parameters are split into task-sharing parameter θ^{sh} and task-specific parameters θ^{tsk} . Let $L_t(D_t; \theta^{sh}, \theta^{tsk})$ denote the average loss on D_t for task t using $(\theta^{sh}, \theta^{tsk})$. The objective function of MTL is:

$$\min_{\theta^{sh}, \theta^{tsk}} \sum_{t=1}^T \omega_t L_t(D_t; \theta^{sh}, \theta^{tsk}) \quad (1)$$

where ω_t is the task weight for task t . It makes sense to share some network structures, so each task doesn't need to learn everything from scratch independently. While hard parameter sharing (HPS) [10] serves as the foundational and frequently utilized structure in MTL, it can encounter challenges with negative transfer. This is attributed to potential conflicts among tasks, as parameters are shared directly. Some tasks have loss functions on different scales (classification vs regression), Some tasks are asymmetric in terms of importance, difficulty, data availability or noise.

To cope with these challenges, we formulated the research problem as follows: How to design MTL approach to improve RAN task performances by analyzing the state-of-the-art MTL approaches characterized by a tuple of {architecture, weighting, distributed learning topology}. Searching for an optimal combination of all these elements of tuple (optimization variables) with all of their possible search space is prohibitively expensive. Therefore for holistic optimization of MTL, we resorted to heuristic strategy by first exhaustively searching all possible combinations of architecture and weighting keeping distributed learning topology fixed. With the best performing combination of architecture and weighting, all possible distributed learning topologies are then evaluated in context of four RAN tasks. The search space for these elements used in experiments are described in next subsections.

A. Architecture

To deal with task conflicts we investigate three additional MTL model architectures (a) MMoE [11], (b) Deselect-k [12] and (c) CGC [13] (see Fig. 2). In MMoE, the approach differs from having a single shared bottom network for all tasks. Instead, it employs a set of bottom networks, referred to as experts, along with a gating network for each task. The gating networks learns to assign weights to the experts on a per-example basis, and MMoE produces an output as a weighted combination of these experts. This per-example weighting mechanism enables distinct tasks to leverage the experts in varying ways making it more flexible and capable of capturing intricate relationships among tasks.

Dselect-k as opposed to MMoE uses sparse gates. CGC differs from MMoE in that it incorporates both shared and task-specific experts. Each expert module within CGC consists of multiple sub-networks referred to as experts. Shared experts in CGC focus on learning patterns that are common across tasks, while task-specific experts are responsible for extracting patterns specific to individual tasks.

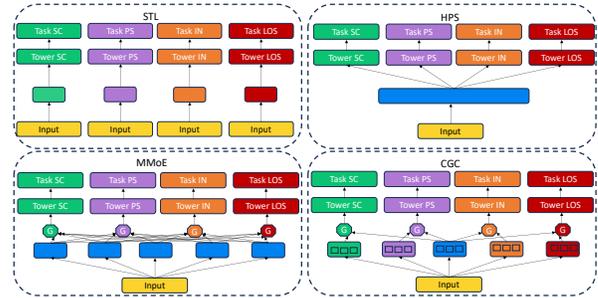


Fig. 2: MTL Architectures. Here blue box shows shared parameters and nodes named 'G' are gates.

B. Weighting

In addition to architecture which decides which parameters are shared and how to share, the other research thrust is devising strategies to optimize MTL. Balancing multiple training losses in MTL affects how task-shared parameters are updated and so methods like (i) loss balancing and (ii) gradient balancing are developed. Although, these two lines of research have been pursued independently in literature as the optimization methods are mainly related to the objective function, while the design of the architecture is to learn relationships between tasks. We in this work investigate loss balancing and gradient balancing schemes combined with architectures to cope with task negative correlation and further enhance MTL's performance. Instead of equally weighting (EW) all tasks simultaneously, loss balancing methods aim at weighting task losses computed dynamically according to different measures such as homoscedastic uncertainty (UW) [14]. DWA [15] learns to dynamically average task weighting over time by assessing the rate of loss change for each task. GLS in [16] expresses the total loss of a multi-task learning problem as the geometric mean of individual task losses while RLW [17] uses normalized random weights for the tasks.

From the gradient perspective, the update of task-sharing parameters θ depends on all task gradients. Thus, gradient balancing methods aim to aggregate all task gradients under different constraints. For example, MGDA [18] formulates MTL as a multi-objective optimization problem and directly solves the optimal weights in every iteration by finding a common descending direction among all the gradients via solving a quadratic programming problem. CAGrad [19] improves MGDA by constraining the aggregated gradient to around the average gradient. GradNorm [20] learns task weights by constraining the gradient magnitude of each task to be similar. PCGrad [21] projects the gradient of one task onto the normal plane of the others if their gradients conflict while GradVac [22] aligns the gradients regardless of whether the gradients conflict or not. GradDrop ([23] randomly masks out the gradient values with inconsistent signs.

C. Distributed Learning Topology

We also consider distributed learning mode depending upon where training of MTL models happen within RAN context.

(a) local mode (MTL_local, STL_local) wherein each model will reside in BS and have its own data and will train model independently without sharing anything with centralized server, (b) global mode (MTL_global, STL_global) wherein all sites' data is aggregated into a global dataset at a centralized place and a single global model is trained for all BSs, and (c) partial federated approach wherein only the weights of shared layers are used in federation. In this setting, eq (1) modifies to (2):

$$\min_{\theta^{sh}, \{\theta_i^{tsk}\}_{i=1}^{N_{BS}}} \frac{1}{N_{BS}} \sum_{i=1}^{N_{BS}} \sum_{t=1}^T \omega_{i,t} L_{i,t}(D_{i,t}; \theta^{sh}, \theta_i^{tsk}) \quad (2)$$

where N_{BS} is the total number of BSs. During each communication round ($k = 1, 2, \dots, K$) of partial federation, the server broadcasts the current global version of the shared parameters (e.g., shared_experts in CGC) to participants BSs. Each BS uses the global shared representation part θ^{sh} received from the server to join with the task-specific parameters θ_i^{tsk} maintained locally to obtain the local model. It then performs one or more steps of (stochastic) gradient descent to update both the shared parameters and the personal parameters (e.g., task specific experts in CGC) and sends only the updated shared parameters to the server for aggregation (Fig. 3). We explored two partial federation algorithms FedAlt and FedSim [24] that differ in way how local updates are performed:

- FedAlt: the BSs first update the personal parameters with the received shared parameters fixed for τ_{tsk} head local iterations and then update the shared parameters with the new personal parameters fixed for τ_{sh} local iterations.
- FedSim: the BS makes $\tau_{sh,tsk}$ local gradient-based updates to both its shared and the task-specific layers synchronously.

In the beginning, we let the BSs run their MTL locally for few epochs ($\tau_{initial}$) and then start federation rounds that showed to improve performance as compared to starting federation from start like in FedVanila benchmark we used in our experiments. We explored hyper-parameters for FedAlt and FedSim and found that FedAlt is more sensitive to its hyper-parameter values. For FedSim best parameters were $\{\tau_{initial} = 10, \tau_{sh,tsk} = 10, K = 9\}$ while for FedAlt, $\{\tau_{initial} = 10, \tau_{tsk} = 17, \tau_{sh} = 1, K = 5\}$. For FedAlt, keeping τ_{sh} smaller than τ_{tsk} gave better performance. We also experimented by sharing task specific parameters of CGC in federation rounds while keeping shared experts out but the performance degraded. We also compared performance with FedVanila in which all model parameters are averaged by server.

III. RESULTS AND DISCUSSION

We generated the dataset from Ericsson propriety radio network simulator with propriety 3D ray tracing propagation model. We generated data from multiple city scenarios each having three base stations. Each base station has three sectors supporting one primary LTE and one secondary NR carrier. The simulated area consisted of a 2 km x 2 km slice of large cities from Europe, Asia and USA containing several buildings

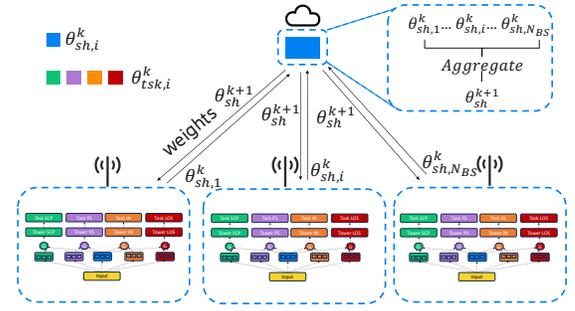


Fig. 3: Partial Federated MTL.

of varying heights causing scattering effects. Each base station data was divided into training/validation and testing with ratio 60/20/20. The input features are intra-RSRP measurements of all cells that will be same for all RAN tasks. The task labels are:

- 1) Secondary Carrier Prediction (SC): Measured inter-RSRP on secondary frequencies for all cells with dimension $1 \times N_{cells}^{fs}$.
- 2) Positioning (PS): Distance of UE to all BSs with dimension $1 \times N_{BS}$.
- 3) Indoor/Outdoor link Classification (IN): Binary scalar indicator if UE is indoor.
- 4) LoS/NLoS link Classification (LOS): Binary indicator if UE has LoS link for each of the secondary carrier cells with dimension $1 \times N_{cells}^{fs}$.

where N_{cells}^{fs} is the number of secondary carrier cells. The input features and output task labels for each of the BSs of a city comprises of cell measurements for only that city. For the MTL architectures and weighting strategies, we used the implementations available in python open-source library LibMTL [25]. The network simulation parameters are given in Table I. Mean Squared Error (MSE) loss was used for SC and PS with mean absolute error (MAE) as their validation metric while binary cross entropy loss was used for IN and LOS tasks with accuracy as their validation metric. For MTL model training, we set training budget in terms of 100 epochs and we used Adam optimizer ($\text{lr} = 0.0001$, momentum = 0.9), 512 layer size for shared parameters, batch size of 64 and number of experts to be 2. Each expert was single layer feed-forward network. Rest of the parameters used were set to default values as provided by [25].

First, we explored the MTL design search space with distributed learning topology fixed to local mode. Each experiment is repeated multiple times and averaged results among all BSs across all four cities are reported. The box plot depicted in Fig. 4 illustrates the variability in values for the objective function $\Omega = Accuracy_{IN} + Accuracy_{LOS} - MAE_{SC} - MAE_{PS}$ across all conceivable combinations of MTL model architectures and weighting strategies. In this context, higher values indicate better performance. Based on lower 25 percentile, it appears that CGC with UW and HPS-GLS perform best among all. CGC with UW had better convergence and therefore this

combination was chosen for rest of experiments. Next, we evaluate the performance for all four RAN tasks considering all modes of distributed learning and performance is shown in Figs. (5-9) for held-out test set. The inset bar plot shows percentage improvement as compared to using STL in local mode.

For secondary carrier prediction task with MAE as metric (Fig. 5), FedAlt (1.9% improvement as compared to STL local) and FedSim (1.8% improvement as compared to STL local) outperformed all others then comes MTL global and MTL_local. The performance of STL_global is worst. Both global based approaches (MTL_global and STL_global) have faster convergence while FedVanila has slowest convergence. In STL, the difference between global and local is much larger than in MTL local and global. Both global and local based MTLs performed better than STLs. For location prediction task with MAE as metric (Fig. 6), MTL_global showed best convergence and outperformed all by large margins (9.6% improvement) and later followed by FedSim and FedAlt. For both MTLs and STLs, global performed better than local. FedVanila has slowest convergence and STL_local is worst. Difference between STL_global and local is much larger than MTL_local and global. Overall globals are better than locals and MTL is better than STL. For indoor link classification with accuracy as metric (Fig. 7), MTL_local is slightly better by STL_local (0.05% improvement). This is contrary to previous two where globals performed better than locals. Both MTL_global and STL_global have quick convergence to suboptimal value and at end STL_global and MTL_global are worst. For LoS link classification task and accuracy as metric (Fig. 8), STL_local is best followed closely by MTL_local. This is different from previous three where MTL performed best than STLs. FedVanila has unstable progression and slowest convergence. Both locals are better than global while MTL and STL are almost on par.

Overall, MTL performs either best among all or on par with STL. This is still an improvement as LCM costs are reduced i.e., one model instead of managing multiple models. Federation with partial model is better than with full model parameters. For secondary carrier prediction, collaboration through partial federation and for location prediction, collaboration through aggregating training data for global model or sharing model weights in partial federation is helpful. Global based approach works best for location prediction and for secondary carrier prediction, it is either on par with local (MTL_local) or show degraded performance (STL_global)

TABLE I: Network Simulation Parameters.

Parameter	Value
Num of cities	4
Num of BSs	3 per city
Propagation model	3D ray tracing
LTE carrier	900 MHz
NR carrier	4.5 GHz
UE density	1K uniformly distributed UEs per snapshot
Num of snapshots	350 for four cities

while both classification tasks are better in local modes. One possible explanation for this can be the influence of local context. For location prediction, distance to base station is the task label that directly affects the input features (RSRPs at primary carrier) and therefore aggregating the dataset from diverse environments helps to learn a better general model as this mapping of UE-to-BS distance and RSRP is not heavily influenced by the environment. For other three tasks, mapping of input features to task labels serve as proxy of environment and local context is hard coded in this relation. Therefore tasks labels are highly influenced by the local context (e.g., layout of buildings within a particular city will affect line of sight and indoor probability as well as secondary carrier strength). Therefore for these three tasks, site-specific model outperforms as compared to aggregating the data samples from diverse cities and base stations for training a general model that hurts the model performance.

Next we experimented with 1% training dataset size. For space brevity, we have only included visualization for SC task. For secondary carrier prediction (Fig. 9), both globals have quickest convergence by a large margin and outperform all at the end (around 59% improvement as compared to STL_local). FedVanila is worst. FedSim is better than FedAlt and FedVanila. Difference between global and local in STL is almost same as for MTL. All federation based methods fall behind the locals and globals. For location prediction, overall trend was similar with MTL_global outperforming all (27.3% improvement as compared to STL_local) followed very closely by STL_global and both have very quick convergence. For indoor classification, both globals are better than locals (11.8% improvement). For LoS link classification, both locals are better than both globals. Both globals and FedVanila perform worse than all while others are on par with each other. Overall with small training dataset availability, globals perform better than locals except in LOS where MTL is on par with STL. For first three tasks, either training a single global model or independent local models is best while for LOS local mode is best. When compared to full dataset size trained model, the performance is worse with models trained on sparse data as can be seen by comparing y-axis of Fig. 5 and Fig. 9.

Lastly, we analyzed the effect of grouping of tasks and their effect on MTL learning performances using CGC-UW combination and keeping distributed learning fixed to local mode. The convergence were almost on par with each other. Overall,

- 1) For SC: grouping with PS and LOS outperform all (2.09% improvement as compared to STL) and second best is with PS (2.06%). Just having SC as STL is worst.
- 2) For PS: grouping with IN outperform all (8.02% improvement) and second best is with SC (7.38%). Just PS is worst.
- 3) For IN: grouping with all is best (0.09% improvement) and second best is grouping with SC, PS and IN (0.08%). Just IN is worst.
- 4) For LOS: having it as STL outperformed all and followed

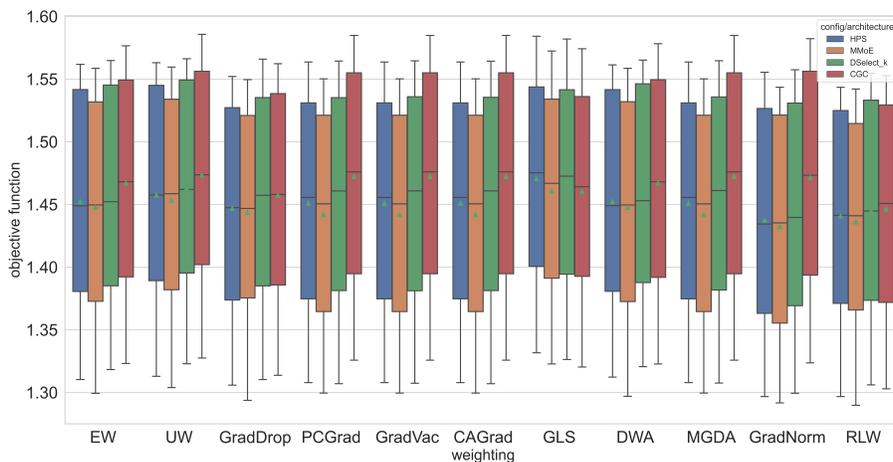


Fig. 4: Effect of MTL architectures and weighting strategies.

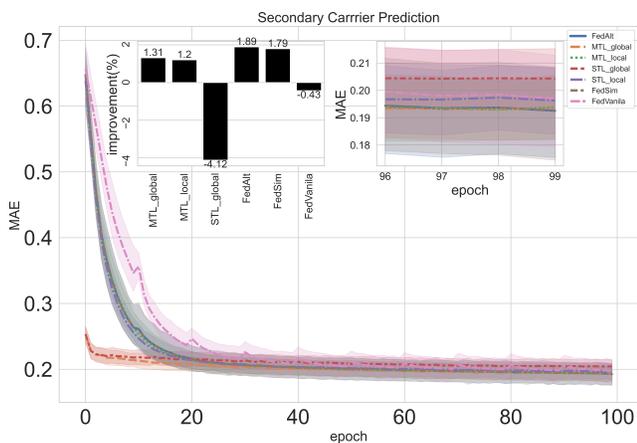


Fig. 5: Secondary Carrier prediction task performance. Improvement over STL_local = {MTL_global: 1.31%, MTL_local: 1.2%, STL_global: -4.12%, FedAlt: 1.89%, FedSim: 1.79%, FedVanilla: -0.43%}

very closely by grouping it with SC and LOS. Grouping together with PS and LOS is worst.

In addition to loss/accuracy metric, model sizes and communication costs are also important. MTL using CGC has 62650 trainable parameters while MTL using HPS has much less (16406) trainable parameters. For comparison, STL has 9737, 6657, 5633 and 9737 trainable parameters for the four RAN tasks respectively. Communication cost is highest in global mode then comes FedVanilla followed by FedSim and FedAlt and least with local mode.

IV. CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this work, we have benchmarked the performance of state-of-the-art multi-task learning (MTL) approaches applied to RAN related tasks. As per results, MTL was found to perform either best among all or on par with STL. SC, PS and IN tasks benefited from knowledge sharing (global model or partial federation) while LOS performance degraded. With sparse training data, collaboration through dataset aggregation

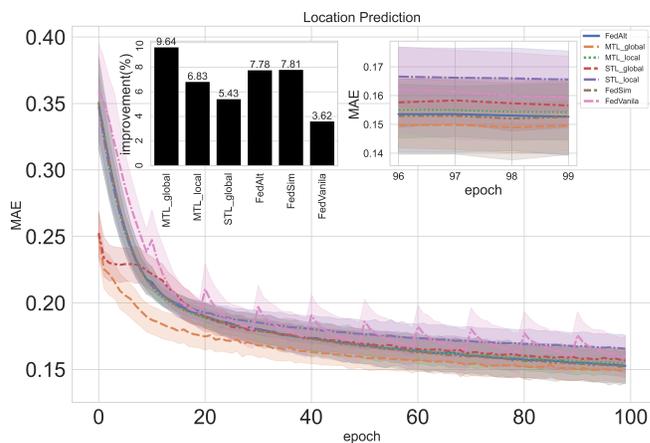


Fig. 6: Location Prediction task performance. Improvement over STL_local = {MTL_global: 9.64%, MTL_local: 6.83%, STL_global: 5.43%, FedAlt: 7.78%, FedSim: 7.81%, FedVanilla: 3.62%}

for a single global model helped significantly but MTL was found to be on par with STL. Federation with partial parameters is much better than full federation with MTL model. The gain of MTL can be further analyzed by devising a utility value based on model performance and resource utilization costs. For future work, we will investigate scenarios where UEs collect some of the labels of these tasks at the same spatiotemporal point and so labels for all tasks are not available as well as its generalization ability for unseen tasks. We will also investigate distributed algorithms to search for optimal MTL related hyper-parameters of edge nodes (e.g., MTL architecture per node, weighting strategy per node, task grouping per node, number of experts, learning rate etc).

REFERENCES

- [1] P. V. Klaine, M. A. Imran, O. Onireti, and R. D. Souza, "A survey of machine learning techniques applied to self-organizing cellular networks," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2392–2431, 2017.
- [2] H. Ryden, J. Berglund, M. Isaksson, R. Cöster, and F. Gunnarsson, "Predicting strongest cell on secondary carrier using primary carrier data," in *IEEE WCNC Workshops*, 2018, pp. 137–142.

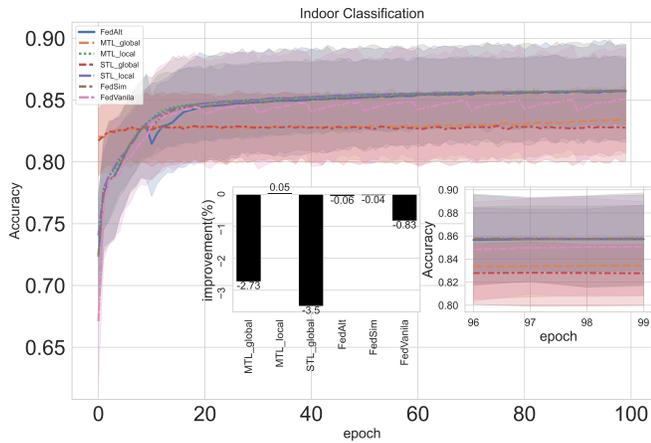


Fig. 7: Indoor link classification task performance. Improvement over STL_local = {MTL_global: -2.73%, MTL_local: 0.05%, STL_global: -3.5%, FedAlt: -0.06%, FedSim: -0.04%, FedVanila: -0.83%}

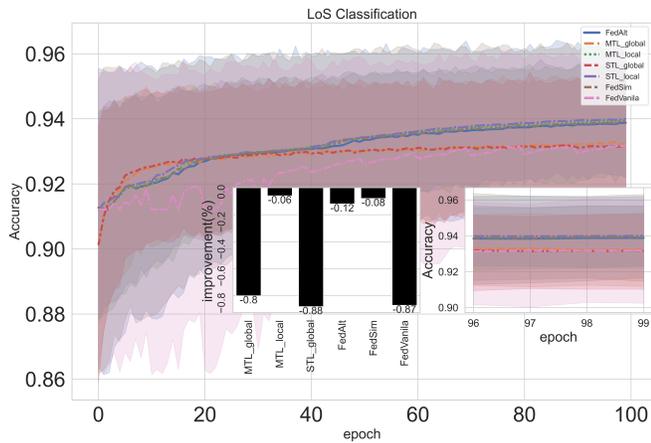


Fig. 8: LoS link classification task performance. Improvement over STL_local = {MTL_global: -0.8%, MTL_local: -0.06%, STL_global: -0.88%, FedAlt: -0.12%, FedSim: -0.08%, FedVanila: -0.87%}

[3] Y. Gao, J. Chen, Z. Liu, L. Liu, and N. Hu, "Deep learning based location prediction with multiple features in communication network," in *IEEE WCNC*, 2021, pp. 1–5.

[4] S. A. Hamideche, M. L. Alberi Morel, K. Singh, and C. Viho, "Indoor-outdoor detection using time series classification and user behavioral cognition," in *IFIP WMNC*, 2022, pp. 7–14.

[5] A. Kirmaz, D. S. Michalopoulos, I. Balan, and W. Gerstacker, "Los/nlos classification using scenario-dependent unsupervised machine learning," in *IEEE 32nd PIMRC*, 2021, pp. 1134–1140.

[6] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 12, pp. 5586–5609, 2022.

[7] Z. Ali, L. Giupponi, M. Miozzo, and P. Dini, "Multi-task learning for efficient management of beyond 5g radio access network architectures," *IEEE Access*, vol. 9, pp. 158 892–158 907, 2021.

[8] M. L. A. Morel, I. Saffar, K. D. Singh, and C. Viho, "Multi-task deep learning based environment and mobility detection for user behavior modeling," in *WiOPT*, 2019, pp. 1–7.

[9] A. Jagannath and J. Jagannath, "Multi-task learning approach for automatic modulation and wireless signal classification," in *ICC 2021 - IEEE International Conference on Communications*, 2021, pp. 1–7.

[10] R. Caruana, "Multitask learning: A knowledge-based source of inductive bias," in *ICML*, ser. ICML'93, 1993, pp. 41–48.

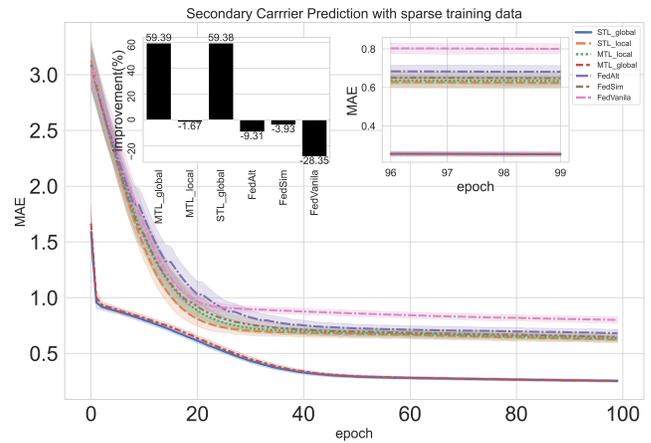


Fig. 9: Secondary carrier prediction task performance with sparse training data. Improvement over STL_local = {MTL_global: 59.39%, MTL_local: -1.67%, STL_global: -59.38%, FedAlt: -9.31%, FedSim: -3.93%, FedVanila: -28.35%}

[11] J. Ma, Z. Zhao, X. Yi, J. Chen, L. Hong, and E. H. Chi, "Modeling task relationships in multi-task learning with multi-gate mixture-of-experts," in *Proceedings of the 24th ACM SIGKDD*, 2018, pp. 1930–1939.

[12] H. Hazimeh, Z. Zhao, A. Chowdhery, M. Sathiamoorthy, Y. Chen, R. Mazumder, L. Hong, and E. Chi, "Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 29 335–29 347.

[13] H. Tang, J. Liu, M. Zhao, and X. Gong, "Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations," in *ACM RecSys*, 2020, pp. 269–278.

[14] R. Cipolla, Y. Gal, and A. Kendall, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *2018 IEEE/CVF CVPR*, jun 2018, pp. 7482–7491.

[15] S. Liu, E. Johns, and A. J. Davison, "End-to-end multi-task learning with attention," in *2019 IEEE/CVF CVPR*, jun 2019, pp. 1871–1880.

[16] S. Chennupati, G. Sistu, S. Yogamani, and S. A. Rawashdeh, "Multi-net++: Multi-stream feature aggregation and geometric loss strategy for multi-task learning," in *2019 IEEE/CVF CVPRW*, jun 2019, pp. 1200–1210.

[17] B. Lin, F. YE, Y. Zhang, and I. Tsang, "Reasonable effectiveness of random weighting: A litmus test for multi-task learning," in *Transactions on Machine Learning Research*, 2022.

[18] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[19] B. Liu, X. Liu, X. Jin, P. Stone, and Q. Liu, "Conflict-averse gradient descent for multi-task learning," in *NeurIPS*, 2021.

[20] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *ICML*, 2018.

[21] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 5824–5836.

[22] Z. Wang, Y. Tsvetkov, O. Firat, and Y. Cao, "Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models," in *ICLR*, 2021.

[23] Z. Chen, J. Ngiam, Y. Huang, T. Luong, H. Kretschmar, Y. Chai, and D. Anguelov, "Just pick a sign: Optimizing deep multitask models with gradient sign dropout," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 2039–2050.

[24] K. Pillutla, K. Malik, A. rahman Mohamed, M. G. Rabbat, M. Sanjabi, and L. Xiao, "Federated learning with partial model personalization," in *ICML*, 2022.

[25] B. Lin and Y. Zhang, "Libmtl: A python library for multi-task learning," in *arXiv preprint arXiv:2203.14338*, 2022.