2024 IEEE 35th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC): Track 2: Networking and MAC

2024 IEEE 35th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC) | 979-8-3503-6224-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/PIMRC59610.2024.10817307

Multi-Task Learning for Resource Allocation in Wireless Networks of Dynamic Dimensionality

Nikos A. Mitsiou*, Pavlos S. Bouzinis‡, Panagiotis D. Diamantoulakis*,
Panagiotis G. Sarigiannidis†, George K. Karagiannidis*§
*Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece
e-mail: {nmitsiou, padiaman, geokarag}@auth.gr
†Department of Informatics and Telecommunications Engineering, University of Western Macedonia, 50100 Kozani, Greece
e-mail: psarigiannidis@uowm.gr
‡MetaMind Innovations P.C., 50100 Kozani, Greece, e-mail: pbouzinis@metamind.gr
§Artificial Intelligence & Cyber Systems Research Center, Lebanese American University (LAU), Lebanon

***Abstract*—Deep neural networks (DNNs) have demonstrated their efficacy in delivering accurate solutions to a range of optimization problems. However, in the context of wireless communications, the size of these problems may vary across adjacent time slots, due to fast changes in the networks' architecture, e.g., the number of users. It is essential to note that this time-varying dimensionality of optimization problems in wireless networks necessitates adjustments in the DNN architecture, resulting in different numbers of input and output nodes. To address this challenge, in our paper, optimization problems of varying size are treated as distinct tasks. To tackle these tasks, a multi-task learning (MTL) approach based on modular sharing is proposed. The multi-task approach consists of a DNN, which is used to extract the solutions for all the optimization problems, and a router which manages which nodes and layers of the input and output layer of the DNN to be used during the forward propagation of each task. Consequently, all tasks share common parameters of the DNN, while the DNN dynamically adjusts to the number of nodes of its output and input layers. Numerical results demonstrate the superiority of the suggested approach over zero-padding, which is the current solution for handling resource allocation problems of varying size.**

***Index Terms*—multi-task learning, deep neural networks (DNNs), non-convex optimization**

I. Introduction

Future wireless networks are envisaged as multi-band, decentralized, fully autonomous, and highly flexible user-centric systems, encompassing satellite, aerial, terrestrial, and underwater communications [1]. As a consequence, various key performance indicators (KPIs) need to be considered to meet the diverse, and often contradictory, requirements of these subsystems. To achieve this goal, incorporating intelligence into the physical (PHY) and medium-access control (MAC) layers of future networks is essential.

Deep neural networks (DNNs) have been widely used for tackling various complex optimization problems, especially in the context of wireless communications [2]–[7]. Specifically, in [2]–[5], DNN-based frameworks were proposed for resource allocation in wireless communications. In [2], the class of "learnable algorithms" and the design of DNNs to approximate some algorithms of interest in wireless were given, while in [3], [4] a distributed and unsupervised learning (UL) based framework for constrained optimization was proposed. Both frameworks relied on primal-dual optimization for handling the constraints via the DNNs. Furthermore, in [5], the notion of intelligent resource allocation for wireless networks was discussed, while in [6] intelligent resource management based on online-learning for non-convex problems was investigated. In the same line, [7] online-learning was utilized for orthogonal frequency-division multiplexing (OFDM) resource allocation. Despite their novelty, none of the aforementioned works considered the case of input data to the DNN with varying dimensions. To this end, the ZP technique has been employed in various challenging wireless communication problems [8]–[12]. Specifically, in [8] a deep reinforcement learning (DLR) approach for LoRa wireless networks was proposed, where the ZP was used to ensure the same vector length for all devices. In [9], ZP for a resource allocation scheme based on convolutional DNNs (CDNNs) was studied, while in [12] the ZP technique was used to perform a power allocation scheme for green Internet-of-Things (IoT) networks. Also, in [10], [11] ZP was used to handle the changing dimensionality of the input data associated with the resource management of the network, under the scenario of multiple access points (APs) and D2D communications in respect.

As such, in this work, we consider optimization problems of different size as separate inference-based tasks, and an appropriate mapping which describes the solutions to these tasks is given. Subsequently, we propose a dynamic DNN, based on multi-task learning (MTL), which is capable of effectively capturing this mapping. Towards realizing the proposed multi-task DNN, the concept of *modular sharing* is implemented. Specifically, we let all tasks utilize the common DNN towards their inference phase, as such, each task "interferes" with each other and the DNN tunes its parameters to better satisfy all tasks. Also, a router manages which nodes and layers of the input and output layer of the DNN to be used during the forward propagation of each task, due to their need for a different number of nodes. This enables the DNN to be dynamic with respect to both its input and output layers. The proposed multi-task DNN is evaluated under two different re-

979-8-3503-6224-4/24/$31.00 ©2024 IEEE

source allocation scenarios. The numerical results demonstrate that its performance is near the performance of the single-task DNN, which is specifically designed for optimization problems of standard dimensionality, while it is superior to ZP.

II. The Single-task Problem Formulation

First, we consider the set $\mathcal{N} = \{1, 2, ..., N\}$, and the following optimization problem

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f_0\left(\boldsymbol{x}; \boldsymbol{a}\right) \\ \text{s.t.} \quad & f_n\left(\boldsymbol{x}; \boldsymbol{a}\right) \leq 0, \ \ \forall n \in \mathcal{N}, \\ & \boldsymbol{x} \in \mathcal{X}, \end{aligned} \tag{$\mathbf{P}_1$}$$

where the function $f_0 : \mathbb{R}^N \to \mathbb{R}$ describes the networks cost function and the functions $f_1, ..., f_N : \mathbb{R}^N \to \mathbb{R}$ indicate the devices local constraints. The functions $f_0, f_1, .., f_N$ are not necessarily convex in the general case, but are differentiable. The set $\mathcal{X} \subseteq \mathbb{R}^N$ is a nonempty, compact, and convex set, reflecting the set of global constraints. Let us denote the optimal solution of problem ($\mathbf{P}_1$) as $\boldsymbol{x}^*(\boldsymbol{a}) \in \mathcal{X}$. The optimal solution is parameterized by the parameter $\boldsymbol{a} \in \mathcal{A} \subseteq \mathbb{R}^N$. Thus, there exists an optimal mapping between the parameter set $\mathcal{A}$ and the set $\mathcal{X}^*$ which is the set containing all optimal solutions of problem ($\mathbf{P}_1$), i.e.,

$$\begin{aligned} \mathcal{X}^* \triangleq \Big\{ \boldsymbol{x}^* \in \mathcal{X} \ \Big| \ & f_0\left(\boldsymbol{x}^*; \boldsymbol{a}\right) \leq f_0\left(\boldsymbol{x}; \boldsymbol{a}\right), \forall \boldsymbol{x} \in \mathcal{X}, \\ & f_n\left(\boldsymbol{x}^*; \boldsymbol{a}\right) \leq 0, \, \forall n \in \mathcal{N} \Big\}. \end{aligned} \tag{1}$$

This mapping will be denoted as $\mathcal{F}$ and is given below

$$\mathcal{F}: \ \mathcal{A} \xrightarrow{\mathbf{P}_1} \mathcal{X}^*. \tag{2}$$

Essentially, finding the mapping $\mathcal{F}$ that provides the optimal set of solutions for problem ($\mathbf{P}_1$), can be conceived as an individual *task* with input and output dimensionality of $\mathbb{R}^N$, where we define the input of the task to be the parameters $\boldsymbol{a}$ and its output to be the optimal value $\boldsymbol{x}^*(\boldsymbol{a})$. [1] Next, we present two conventional DL approaches for approximately obtaining the desired mapping $\mathcal{F}$.

A. Unsupervised Learning

Model-based methods for solving problem ($\mathbf{P}_1$), and subsequently, SL, require that some conditions for $f_0, ..., f_N$, such as convexity, hold. This is mandatory for obtaining $\boldsymbol{x}^*(\boldsymbol{a})$, $\forall \boldsymbol{a} \in \mathcal{A}$. However, several fundamental problems in the field of wireless communications are not convex. These problems often are not solvable by standard optimization tools [3]. For this reason, UL is a promising approach to overcome this challenge [3], [13] and solve problems of the form given in ($\mathbf{P}_1$), when conditions such as convexity do not hold. We note that, in [3], [14], it was shown that any convex projection on $\mathcal{X}$ can be realized via DNNs. For instance, an often-encountered constraint is of the form $\mathbf{1}^\intercal \boldsymbol{x} \leq 1$, which can be handled by using the Softmax function as the output layer of the DNN. Nonetheless, in the case that the projection onto set $\mathcal{X}$ is not convex, or it is not straightforward to be implemented via the DNN, we can simply concatenate the constraints imposed by the set $\mathcal{X}$ into the rest inequality constraints of ($\mathbf{P}_1$) and proceed without needing to project $\boldsymbol{x}$ onto the set $\mathcal{X}$ explicitly. Thus, hereinafter, without loss of generality, the constraint $x \in \mathcal{X}$ of problem ($\mathbf{P}_1$) will be discarded.

Let us assume a feed-forward fully-connected DNN, with $L+1$ layers and d_l neurons per layer. Let $\boldsymbol{y}^{l-1}$ to be the input to the l-th layer of the network, with $\boldsymbol{y}^0$ being the input to the input layer of the DNN, while in our case it holds that $\boldsymbol{y}^0 = \boldsymbol{a}$. Then, for all $l = 1, ..., L+1$ and $n = 1, ..., d_l$ the output $\boldsymbol{y}^l(n)$ of node n in layer l is obtained as

$$\boldsymbol{y}^l(n) = g_{n,l}(z_{n,l}), \ z_{n,l} = \boldsymbol{w}_{n,l}^\intercal \boldsymbol{y}^{l-1} + b_{n,l} \tag{3}$$

wherein $\boldsymbol{w}_{n,l} \in \mathbb{R}^{d_{l-1}}$ with $\boldsymbol{w}_{n,l}(k)$ being the weight of the link between the k-th neuron in layer $l-1$ and the n-th neuron in layer l, $b_{n,l} \in \mathbb{R}$ is the bias term of neuron n in layer l, while $g_{n,l}$, is the activation function of neuron n in layer l. Also, $\boldsymbol{\Theta} \triangleq \{\boldsymbol{W}_l, \boldsymbol{b}_l\}_{l=1}^{L+1}$ denotes the weights and biases of all layers, i.e., the training parameters, $\boldsymbol{W}_l \in \mathbb{R}^{d_{l-1} \times d_l}$ is the matrix containing all the weights between the $(l-1)$-th and the l-th hidden layer, and $\boldsymbol{b}_l \in \mathbb{R}^{d_l}$ is the vector which contains all the biases of the l-th layer. The output of the unsupervised DNN is given as $\boldsymbol{y}^{L+1}\left(\boldsymbol{\Theta}; \boldsymbol{a}\right)$. The aim of the UL approach is to obtain a $\boldsymbol{y}^{L+1}\left(\boldsymbol{\Theta}; \boldsymbol{a}\right)$, so that $\boldsymbol{y}^{L+1}\left(\boldsymbol{\Theta}; \boldsymbol{a}\right) \approx \boldsymbol{x}^*(\boldsymbol{a})$, $\forall \boldsymbol{a} \in \mathcal{A}$. However, the dataset $\mathcal{D} = \{\boldsymbol{a}^u, \boldsymbol{x}^*(\boldsymbol{a}^u)\}_{u=1}^D$ cannot be created since there is no tractable way to obtain the values of $\boldsymbol{x}^*(\boldsymbol{a}^u)$ for each $\boldsymbol{a}^u$. Therefore, in the UL case, the dataset will be given as $\mathcal{D} = \{\boldsymbol{a}^u\}_{u=1}^D$, and the UL-based DNN has to be trained in such a fashion so that $\boldsymbol{y}^{L+1}\left(\boldsymbol{\Theta}; \boldsymbol{a}^u\right) \approx \boldsymbol{x}^*(\boldsymbol{a}^u)$, $\forall \boldsymbol{a}^u \in \mathcal{D}$. To this end, it is imperative to define the loss function of the UL-based DNN so that it includes both the objective function of problem ($\mathbf{P}_1$), and its constraints. A possible approach to address this is the Lagrange duality method [3]. First, the Lagrangian of ($\mathbf{P}_1$) is given as follows

$$\mathcal{L}_g(\boldsymbol{x}, \boldsymbol{\lambda}) = f_0\left(\boldsymbol{x}; \boldsymbol{a}\right) + \sum_{n \in \mathcal{N}} \lambda_n f_n\left(\boldsymbol{x}; \boldsymbol{a}\right), \tag{4}$$

where the non-negative λ_n corresponds to the dual variable associated with the n-th constraint in ($\mathbf{P}_1$). $\boldsymbol{\lambda} \in \mathbb{R}^N$ denotes the vector containing all $\lambda_n, \forall n$. Moreover, the dual function $\mathcal{G}(\boldsymbol{\lambda})$ is given as

$$\mathcal{G}(\boldsymbol{\lambda}) = \inf_{\boldsymbol{x}} \mathcal{L}_g(\boldsymbol{x}, \boldsymbol{\lambda}), \tag{5}$$

while the respected dual problem is defined as

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \quad & \mathcal{G}(\boldsymbol{\lambda}) \\ \text{s.t.} \quad & \boldsymbol{\lambda} \succeq 0 \end{aligned} \tag{$\mathbf{P}_2$}$$

To solve problem ($\mathbf{P}_2$) the primal-dual method can be employed [3]. By taking into account that $\boldsymbol{x} = \boldsymbol{y}^{L+1}\left(\boldsymbol{\Theta}\right)$, we define the loss function of the UL method as

$$L_{\text{UL}} = \mathcal{L}_g(\boldsymbol{y}^{L+1}\left(\boldsymbol{\Theta}^{t-1}\right), \boldsymbol{\lambda}), \tag{6}$$

[1] It is clarified that the paper can be easily generalized to the case that the problem ($\mathbf{P}_1$) has different input and output dimensionality.

which takes into account both the objective function and the constraints of problem ($\mathbf{P}_1$). Then, by using the mini-batch stochastic gradient method, the classic primal-dual algorithm for solving problem ($\mathbf{P}_2$) can be written as follows [3]

$$\boldsymbol{\Theta}^t = \boldsymbol{\Theta}^{t-1} - \eta^t \left(\frac{1}{B} \sum_{u \in \mathcal{B}} \nabla_{\boldsymbol{\Theta}} f_0 \left(\boldsymbol{y}^{L+1} \left(\boldsymbol{\Theta}^{t-1} \right); \boldsymbol{a}^u \right) + \sum_{n \in \mathcal{N}} \lambda_n \left(\frac{1}{B} \sum_{u \in \mathcal{B}} \nabla_{\boldsymbol{\Theta}} f_n \left(\boldsymbol{y}^{L+1} \left(\boldsymbol{\Theta}^{t-1} \right); \boldsymbol{a}^u \right) \right) \right) \quad (7)$$

$$\lambda_n^t = \lambda_n^{t-1} + \eta^t \left(\frac{1}{B} \sum_{u \in \mathcal{B}} f_n \left(\boldsymbol{y}^{L+1} \left(\boldsymbol{\Theta}^{t-1} \right); \boldsymbol{a}^u \right) \right)^+, \quad (8)$$

where t is the iteration index of the iterative procedure, $\mathcal{B} \subseteq \mathcal{D}$ and $B = |\mathcal{B}|$ is the batch size of the training dataset is the batch while $(\cdot)^+$ denotes the operation $\max\{0, \cdot\}$. First, by updating the training parameters $\boldsymbol{\Theta}$ through back-propagation, the DNN learns to minimize the Lagrangian function of (6). Afterwards, by updating $\boldsymbol{\lambda}$ the dual function given in (7) is maximized. The iterative updates of (9)-(10) provide approximate solutions to the optimal value of $\boldsymbol{x}^*$, and to the optimal value of $\boldsymbol{\lambda}^*$.

III. Resource Allocation Problems of Varying Dimensionality

Despite the potential of UL in addressing optimization problems, the mapping of (2) is influenced by the set $\mathcal{N}$, which reflects the dimensionality of the considered optimization problem. In the context of wireless communications, the set $\mathcal{N}$ may represent the number of active users in the network, thus, directly affecting the total number of optimization variables. In practical systems, the dimensionality of problem ($\mathbf{P}_1$) may change over time. For instance, the number of active users in the network may vary between adjacent time slots. As a consequence, changes of the problem's dimensionality, result in solving a different problem, i.e., to undertake a different inference task, when the optimization problem is tackled through DNNs. However, the SOTA DNNs have a fixed architecture with a predetermined number of input and output nodes. Subsequently, a specific DNN can capture the mapping (2) for only a particular dimensionality value of problem ($\mathbf{P}_1$) [15]. A naive approach to address the challenge of a task's changing dimensionality is to pretrain multiple DNNs for each individual task. However, training different DNNs for all possible scenarios, such as varying numbers of users in the network, can be time-consuming and impractical. This is especially challenging in scenarios like the open RAN architecture, where network resource allocation operates as an xApp, since whenever the number of users changes, it is necessary to remove the outdated xApp and launch an updated one [16]. This not only complicates the xApps' life-cycle management but also limits the network orchestrator's ability to adapt to to environmental changes.

Towards addressing this challenge, we first need to consider a new mapping $\mathcal{F}^{\mathrm{MT}}$ that can simultaneously handle optimization problems of the form of ($\mathbf{P}_1$), but of different

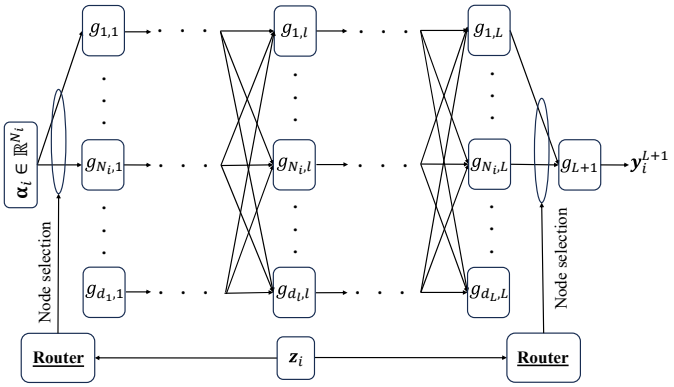


Fig. 1: A modular sharing example for MTL.

dimensionality. As the mapping that represents the solution of an optimization problem of a specific dimensionality can be treated as a different task, the new mapping reflects a multi-task operation, involving tasks of different dimensionality values. In consequence, the DNN which will be used to approximate the mapping $\mathcal{F}^{\mathrm{MT}}$ should be able to adjust its number of nodes of its input and output layers based on the dimensionality values of all considered tasks. With a slight abuse of notation, we define the set $\mathcal{N}^{\mathrm{MT}} = \{\mathcal{N}_1, \mathcal{N}_2, ..., \mathcal{N}_K\}$, with cardinality K, to be the set which contains all sets that reflect the dimensionality values of all considered tasks. Moreover, for each $\mathcal{N}_i, \forall i \in \{1, 2, ..., K\}$, it holds that $\mathcal{N}_i = \{1, 2, ..., N_i\}, N_i \in \mathbb{Z}^+$, where N_i is the input and the output dimensionality of the i-th task. Then, the corresponding mapping of (2) in the multi-task case is given as follows

$$\mathcal{F}^{\mathrm{MT}} : \mathcal{A}_i \xrightarrow{(\mathbf{P}_1)} \mathcal{X}_i^*, \ \forall \mathcal{N}_i \in \mathcal{N}^{\mathrm{MT}}, \quad (9)$$

where $\boldsymbol{a}_i \in \mathcal{A}_i$ is the input parameter of the i-th task, $\mathcal{A}_i \subseteq \mathbb{R}^{N_i}$, $\mathcal{X}_i \subseteq \mathbb{R}^{N_i}$, $\boldsymbol{x}_i^*(\boldsymbol{a}_i)$ is the optimal solution of ($\mathbf{P}_1$) for the i-th task, and $\mathcal{X}_i^*$ is the set containing the optimal values for the i-th task, $\forall \boldsymbol{a}_i \in \mathcal{A}_i$, and is defined similar to (1). In essence, to create the mapping $\mathcal{F}^{\mathrm{MT}}$, multiple mappings $\mathcal{F}_i, \forall \mathcal{N}_i \in \mathcal{N}^{\mathrm{MT}}$, as given in (2), need to be handled simultaneously by a single DNN.

A. Modular Sharing-based MTL

As such, in this paper, a multi-task DNN, which is dynamic with respect to its input and output layers is proposed, and that can extract the mapping $\mathcal{F}^{\mathrm{MT}}, \forall \mathcal{N}_i \in \mathcal{N}^{\mathrm{MT}}$. task [17]. Let us consider the computational graph of Fig. 1. We assume that the DNN network has a number of input and output nodes which equal the maximum dimensionality value of all the considered tasks. Then, it has to hold that

$$d_1 = d_L = \max\{N_1, ..., N_K\}, \quad (10)$$

while the values of $d_i, \forall i \in \{2, ..., L-1\}$, can be chosen in an arbitrary way. We are interested in designing a single DNN which can be used for completing multiple tasks, though, during forward propagation, different tasks may flow through different sub-networks within the same DNN. Specifically,

the aforementioned technique is known as *modular sharing* [17] and enables a DNN to adapt between different tasks, by allowing the tasks to share some common DNN parameters. Let $\boldsymbol{\Theta}_i, \forall i \in \{1, 2, ..., K\}$ denote the training parameters of the subnet of the single DNN, dedicated to the i-th task. From the illustrative example in Fig. 1, it can be observed that different tasks flow through some common pathways in the computational graph of the DNN, and thus, each task causes "interference" to other tasks, in the sense that common weights should be tuned to exhibit satisfactory performance for all the considered tasks. From Fig. 1 is it also shown that each task utilizes a different number of input and output nodes, which is due to the fact that each task has a different dimensionality. To achieve this, the notion of routing is used [17], [18]. The router is responsible for dynamically selecting which parts of the DNN' input and output layers will be activated during the forward propagation of each task [17], [18]. Essentially, for each different task a slightly different DNN, which is a subnet of the original DNN, is instantiated based on the router's output. Therefore, the overall training parameters of the i-th subnet, for the i-th task, during its forward propagation through the bDNN, are then given as

$$\begin{aligned}\boldsymbol{\Theta}_i = \Big\{ &\boldsymbol{W}_1[1:N_i,:], \boldsymbol{b}_1, \{\boldsymbol{W}_l, \boldsymbol{b}_l\}_{l=2}^{L}, \\ &\boldsymbol{W}_{L+1}[1:N_i,:], \boldsymbol{b}_L \Big\}, \forall i \in \{1, 2, ..., K\}.\end{aligned} \tag{11}$$

We note that the router takes as input the parameter $\boldsymbol{z}_i \in \{0,1\}^K$, which is a one-hot vector that indicates the task's index, and then outputs the parameters that will be used per task at the input and the output layers, as these given above.

1) The bDNN's Loss Function Selection: We denote with $L_i(\boldsymbol{\Theta}_i)$ the loss function of the proposed method for the i-th task. Then, the multi-task loss function, is given as follows

$$L(\boldsymbol{\Theta}) = F\left(L_1\left(\boldsymbol{\Theta}_1\right), ..., L_K(\boldsymbol{\Theta}_K)\right), \tag{12}$$

where $F : \mathbb{R}^K \rightarrow \mathbb{R}$ is a function which receives as an input all the loss functions which are associated with each considered task. Many multi-tasks loss functions have been proposed [17], however to make the multi-task loss function applicable to optimization via UL, we will assume that F is the convex combination of all $L_i(\boldsymbol{\Theta}_i)$. Therefore, the multi-task loss function of the proposed scheme will be given as

$$L(\boldsymbol{\Theta}) = \sum_{i=1}^{K} \beta_i L_i(\boldsymbol{\Theta}_i,), \tag{13}$$

with $\sum_{i=1}^{K} \beta_i = 1$. The optimal tuning of the hyperparameters β_i is left for a future extension of this work. It should also be noted that we are currently investigating an intelligent router that not only manages the selection of input-output nodes for each task but also optimizes the selection of different paths per task within the shared DNN architecture. Nonetheless, the results and analysis provided in this paper are the stepping stone to that direction, since they showcase that MTL is feasible and practical for simultaneously solving multiple resource allocation problems.

Another major challenge for the multi-task DNN is to provide an output which satisfies all the constraints, of all different tasks. To this end, a new dual variable, $\lambda_{n,i}$, needs to be defined for the n-th constraint of the i-th different task. Then, during the backward propagation, the following holds

$$\begin{aligned}\boldsymbol{\Theta}^t = \boldsymbol{\Theta}^{t-1} - \eta^t \sum_{i=1}^{K} \beta_i \Bigg[&\frac{1}{B_i} \sum_{u \in \mathcal{B}_i} \nabla_{\boldsymbol{\Theta}} f_{0,i}\left(\boldsymbol{y}_i^{L+1}\left(\boldsymbol{\Theta}_i\right); \boldsymbol{a}_i^u\right) \\ &+ \sum_{n=1}^{N_i} \lambda_{n,i}^{t-1} \left(\frac{1}{B_i} \sum_{u \in \mathcal{B}_i} \nabla_{\boldsymbol{\Theta}} f_{n,i}\left(\boldsymbol{y}_i^{L+1}\left(\boldsymbol{\Theta}_i\right); \boldsymbol{a}_i^u\right) \right) \Bigg],\end{aligned} \tag{14}$$

$$\lambda_{n,i}^t = \lambda_{n,i}^{t-1} + \eta^t \left(\frac{1}{B_i} \sum_{u \in \mathcal{B}_i} f_{n,i}\left(\boldsymbol{y}_i^{L+1}\left(\boldsymbol{\Theta}_i\right); \boldsymbol{a}_i^u\right) \right)^+, \tag{15}$$

where $\boldsymbol{a}_i^u$ is the u-th parameter sample of the i-th task from the dataset $\mathcal{D}_i = \{\boldsymbol{a}_i^u\}_{u=1}^{D_i}$, which consists of $D_i = |\mathcal{D}_i|$ samples. Moreover, $\mathcal{B}_i \subseteq \mathcal{D}_i$, $B_i = |\mathcal{B}_i|$ is the batch size used per task, and $\boldsymbol{y}_i^{L+1}$ is the output of the bDNN for the i-th task. Moreover, $f_{0,i}$ is the cost function of the i-th task, and $f_{n,i}$ is the n-th constraint of the i-th task.

Table I: Simulation parameters.

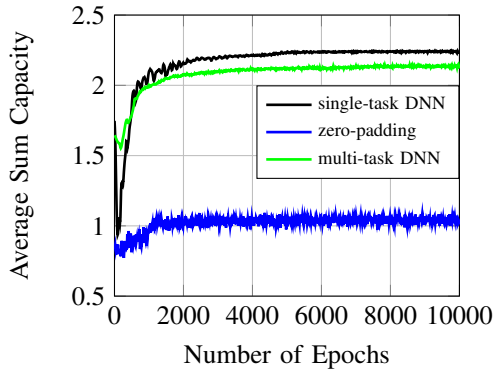
Parameter	Unsupervised scheme
K	7
N_i	$\{5, 8, 10, 12, 15, 18, 20\}$
D	40 k
$L+1$	7
d_1,d_5	20
d_2 - d_4	16
B_i	32
η	0.0005
P_{tot}	1 W
P_{av}	0.5 W
W	1 MHz
β_i	$1/K$
N_0	-174 dBm/Hz

IV. Application to Average Sum Capacity Maximization

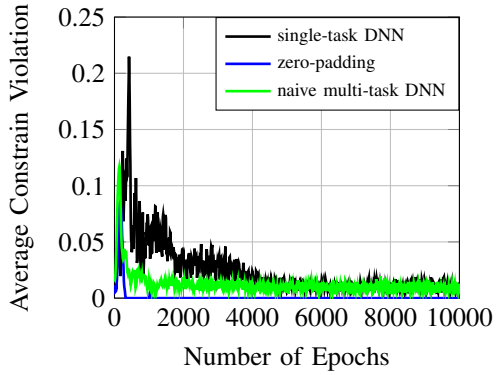
Following [3] we will formulate the average sum capacity maximization problem, under an average and max power constraint in order to verify the proposed UL-based multi-task approach. The problem is given below

$$\begin{aligned}\max_{\boldsymbol{P}} \quad & \mathbb{E}_{\boldsymbol{h}} \left[\log\left(1 + \sum_{n=1}^{N} |h_i|^2 P_i\right) \right] \\ \textbf{s.t.} \ \ \mathrm{C}_1: \quad & \mathbb{E}_{\boldsymbol{h}} \left[\sum_{n=1}^{N} P_i \right] \leq P_{\text{av}} \\ \mathrm{C}_2: \quad & 0 \leq P_n \leq P_{\text{tot}}, \forall n \in \{1, ..., N\}.\end{aligned} \tag{$\mathbf{P}_4$}$$

We observe that ($\mathbf{P}_4$) is a stochastic optimization problem, making it notably more complex to address in the general scenario. Nevertheless, the expectation within the objective function and the constraints can be readily managed using the SDG method as presented in (9) [3]. Hence, there is no requirement for further analysis, and problem (P4) can be

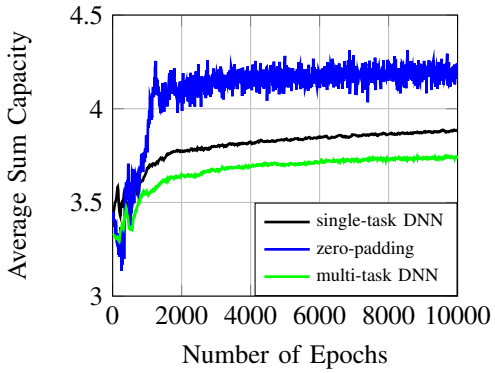


(a) The average sum capacity.

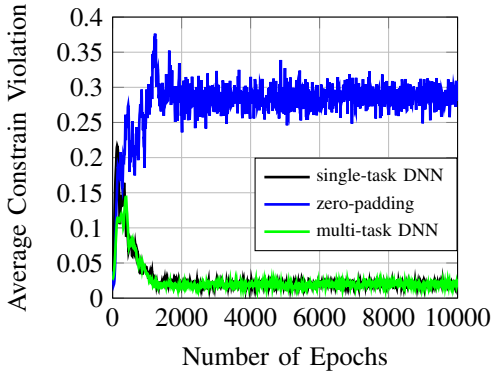


(b) The constraint violation.

Fig. 2: The performance for $N = 5$.



(a) The average sum capacity.



(b) The constraint violation.

Fig. 3: The performance for $N = 20$.

resolved using either (9) in the single-task case, or alternatively, (14),(15) in the multi-task scenario. Consequently, the proposed multi-task approach remains applicable to stochastic optimization problems as well. The dataset comprises only of the feature vectors $\boldsymbol{h}$, thus, $\mathcal{D} = \{\boldsymbol{h}^u\}_{u=1}^{D}$. This dataset will be also established for all, $\mathcal{N}_i$, $\forall i \in \{1, ..., K\}$. In this case, the bDNN architecture also follows that of Fig. 2, where all hidden layers are dense, followed by the ReLU function, with exception of the last layer, which is followed by the Sigmoid activation function, such that the constraint C_2 of problem ($\mathbf{P}_4$) always hold. The constraint C_1 will be forced to hold via the primal-dual optimization.

V. Numerical Simulations

In this section, the proposed multi-task approach is evaluated. All parameters are given in Table I. The results were averaged using a Monte Carlo approach over 100 iterations, while both resource allocation scenarios were studied under Rayleigh fading. The Adam optimizer was used to train all the illustrated schemes, following Algorithm 1. The training dataset consists of $30\,k$ samples, while the testing dataset consists of $10\,k$ samples. To verify the performance of the proposed multi-task DNN, its performance, i.e., the testing evaluation, will be compared to the performance of the following DNN schemes:

- *single-task DNN*: The single-task DNN has an overall number of training parameters equal to the number of the bDNN's training parameters. A different single-task DNN is trained for each different task. Therefore, the performance of the single-task DNN can be considered as the upper (lower) bound for the performance of the proposed multi-task DNN.
- *zero-padding*: The architecture and the number of training parameters of the ZP-based DNN is the same with the single-task DNN, but the dataset of each task is padded with zeros until the feature and label dimensions of all tasks become equal to $\max\{N_1, ..., N_K\}$. This way, a single DNN can be trained subject to all tasks. We note that ZP may not always be applicable to UL-aided optimization problems, but it is applicable to problem ($\mathbf{P}_4$). This is in contrast to the proposed multi-task approach which is always applicable to UL-aided problems.

In Fig. 6, the performance of the UL scheme, for $N = 5$, is plotted. The single-task DNN outperforms the multi-task DNN, but its average constraint violation converges slower compared to that of the multi-task DNN. This is attributed to the fact that interference between different tasks might accelerate the convergence of the respected single tasks components [17]. Nonetheless, all DNNs have the same constraint violation. The proposed multi-task DNN outperforms the naive multi-

Table II: Evaluation of the MTL approach.

Scheme \ Task	$\mathbf{N=5}$	$\mathbf{N=8}$	$\mathbf{N=10}$	$\mathbf{N=12}$	$\mathbf{N=15}$	$\mathbf{N=18}$	$\mathbf{N=20}$
single-task DNN	2.236	2.797	3.1053	3.263	3.515	3.754	3.883
proposed multi-task DNN	2.233	2.783	3.047	3.244	3.5086	3.7309	3.863

task DNN, while ZP converges poorly, obtaining a solution which provides the least average sum capacity.

Finally, in Fig. 8, the convergence for $N = 20$ is shown. First, ZP is shown to completely fail to provide a solution which approximately satisfies the average constraint violation. This is the reason that the average sum capacity of ZP outperforms all schemes, since all, since the ZP-based solution allows the data transmission to utilize more average power than it is actually available. Again, ZP does not have the ability to generalize its performance to all considered tasks. Nonetheless, both multi-task approaches achieve a satisfactory performance, with the proposed approach having a comparable performance to that of the single-task scheme. We also note that the given average sum capacity for the case of $N = 20$ is greater that the capacity of the case $N = 12$.

To further showcase the performance of the proposed multi-task approach, Table II shows the performance of the single-task DNN and of the proposed multi-task DNN. ZP has already shown to perform poorly on several tasks, thus, it was not included. The improvement (%) of the proposed approach compared to the benchmark is also given. It can be noticed that indeed the multi-task DNN generalizes well to all considered tasks. Specifically, the improvement lies between $1.2 - 7.9\%$.

VI. Conclusion

By treating optimization problems of different dimensionalities as distinct tasks, we utilized modular sharing to create a multi-task DNN architecture capable of addressing multiple optimization problems. Simulation results validated the superiority of the proposed multi-task scheme over the method of ZP. Future research directions could explore improving the proposed architecture by optimizing modular sharing, i.e. the subnet selection per task. This can be done via reinforcement learning and evolutionary algorithms.

Acknowledgment

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101070450 (AI4CYBER). Disclaimer: Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Commission. Neither the European Union nor the European Commission can be held responsible for them. Also, the work of G. K. Karagiannidis and P. D. Diamantoulakis was implemented in the framework of H.F.R.I call "Basic research Financing (Horizontal support of all Sciences)" under the National Recovery and Resilience Plan "Greece 2.0" funded by the European Union – NextGenerationEU (H.F.R.I. Project Number: 15642).

References

[1] Z. Zhang, Y. Xiao, Z. Ma, M. Xiao, Z. Ding, X. Lei, G. K. Karagiannidis, and P. Fan, "6g wireless networks: Vision, requirements, architecture, and key technologies," *IEEE Veh. Technol. Mag.*, vol. 14, no. 3, pp. 28–41, 2019.

[2] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to optimize: Training deep neural networks for wireless resource management," in *2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2017, pp. 1–6.

[3] H. Lee, S. H. Lee, and T. Q. S. Quek, "Deep Learning for Distributed Optimization: Applications to Wireless Resource Management," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2251–2266, 2019.

[4] W. Lee, O. Jo, and M. Kim, "Intelligent Resource Allocation in Wireless Communications Systems," *IEEE Commun. Mag.*, vol. 58, no. 1, pp. 100–105, 2020.

[5] M. Eisen, C. Zhang, L. F. O. Chamon, D. D. Lee, and A. Ribeiro, "Learning Optimal Resource Allocations in Wireless Systems," *IEEE Trans. Signal Process.*, vol. 67, no. 10, pp. 2775–2790, 2019.

[6] J. Gao, C. Zhong, G. Y. Li, and Z. Zhang, "Online Deep Neural Network for Optimization in Wireless Communications," *IEEE Wirel. Commun. Lett.*, vol. 11, no. 5, pp. 933–937, 2022.

[7] N. A. Mitsiou, P. D. Diamantoulakis, P. G. Sarigiannidis, and G. K. Karagiannidis, "Energy Efficient OFDM with Intelligent PAPR-aware Adaptive Modulation," *IEEE Commun. Lett.*, pp. 1–1, 2023.

[8] R. Hamdi, E. Baccour, A. Erbad, M. Qaraqe, and M. Hamdi, "LoRa-RL: Deep Reinforcement Learning for Resource Management in Hybrid Energy LoRa Wireless Networks," *IEEE Internet Things J.*, vol. 9, no. 9, pp. 6458–6476, 2022.

[9] W. Lee, M. Kim, and D.-H. Cho, "Deep Power Control: Transmit Power Control Scheme Based on Convolutional Neural Network," *IEEE Commun. Lett.*, vol. 22, no. 6, pp. 1276–1279, 2018.

[10] N. Naderializadeh, J. J. Sydir, M. Simsek, and H. Nikopour, "Resource Management in Wireless Networks via Multi-Agent Deep Reinforcement Learning," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 6, pp. 3507–3523, 2021.

[11] E. Pei and G. Yang, "A Deep Learning based Resource Allocation Algorithm for Variable Dimensions in D2D-Enabled Cellular Networks," in *2020 IEEE/CIC International Conference on Communications in China (ICCC)*, 2020, pp. 277–282.

[12] S. N. Chaudhri, N. S. Rajput, S. H. Alsamhi, A. V. Shvetsov, and F. A. Almalki, "Zero-Padding and Spatial Augmentation-Based Gas Sensor Node Optimization Approach in Resource-Constrained 6G-IoT Paradigm," *Sensors*, vol. 22, no. 8, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/8/3039

[13] A. Zappone, M. Di Renzo, and M. Debbah, "Wireless Networks Design in the Era of Deep Learning: Model-Based, AI-Based, or Both?" *IEEE Trans. Commun.*, vol. 67, no. 10, pp. 7331–7376, 2019.

[14] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 136–145.

[15] M. Akrout, A. Mezghani, E. Hossain, F. Bellili, and R. W. Heath, "From Multilayer Perceptron to GPT: A Reflection on Deep Learning Research for Wireless Physical Layer," *arXiv preprint arXiv:2307.07359*, 2023.

[16] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," *IEEE Commun. Surv. Tutor.*, vol. 25, no. 2, pp. 1376–1411, 2023.

[17] M. Crawshaw, "Multi-task learning with deep neural networks: A survey," *arXiv preprint arXiv:2009.09796*, 2020.

[18] C. Rosenbaum, T. Klinger, and M. Riemer, "Routing networks: Adaptive selection of non-linear functions for multi-task learning," *arXiv preprint arXiv:1711.01239*, 2017.