Transfer Learning with Input Reconstruction Loss

Wei Cui and Wei Yu
Department of Electrical and Computer Engineering, University of Toronto, Canada
Email: {cuiwei2, weiyu}@ece.utoronto.ca

***Abstract*—Neural networks have been widely utilized for wireless communication optimizations. In most of the literature, a dedicated neural network is trained for each specific optimization problem. However, under many scenarios, several distinct objectives are worth optimizing on the same wireless environment. Instead of exhaustively training a new model for every objective, it is more efficient to exploit the correlations between these objectives to train models with shared model parameters and feature representations. In the deep learning literature, *transfer learning* has been proposed to encourage knowledge transfer among models solving correlated problems. Unlike a majority of transfer learning applications where the high level features are relatively easy to locate in the neural networks, this paper considers wireless communication problems, in which it is much more difficult to identify high level features transferable to correlated tasks. To address this issue, this paper proposes to add an additional *reconstruction loss* when training the model. This new loss is for reconstructing the problem inputs starting from a selected neural network hidden layer. This approach encourages the features learnt to be general and descriptive about the inputs, instead of being solely responsible for minimizing the specific task-based loss. When a new objective is to be optimized, these features can be readily used for transfer learning. Simulation results in device-to-device wireless network power allocation optimization suggest that the proposed approach is highly efficient in data and model complexity, resilient to overfitting, and supports competitive optimization performances.**

I. Introduction

Deep learning has gained increasing popularity as a flexible, generalizable, and computationally efficient approach for solving a great variety of wireless communication problems, such as resource allocations [1]–[4], detection and sensing [5]–[8], and so on. In most literature on applying deep learning for wireless communication optimizations, a specialized neural network is trained from scratch for solving each specific optimization task, therefore requiring a large number of training data and model parameters just for obtaining satisfactory performances on that task. This approach lacks scalability and generalization ability when multiple objectives are considered. However, in wireless communication applications, the optimization problems are often based on the same transmission environment, and they differ from each other only in terms of their objective functions. In this paper, we exploit the similarities between these optimization tasks, and purpose a novel transfer learning approach to train neural networks for different tasks in a highly efficient way, in terms of both data and the model complexity.

In the machine learning literature, researchers have explored the transfer of knowledge between machine learning models to tackle similar tasks, known as *transfer learning* [9]. Firstly, a model is fully trained from scratch with abundant training data and computation resources for one task (i.e. the *source task*). Secondly, when a new task correlated to the source task is presented (i.e. the *target task*) with only a small amount of data available, the trained model is further fine tuned based on the limited available data to solve this new task. Transfer learning is most popular in computer vision (CV) [10]–[14], natural language processing (NLP) [15]–[17], and so on.

To better understand transfer learning with neural networks, we interpret the neural network input-to-output computation flow as a two-stage process, i.e. a *feature learning stage* followed by an *optimization stage*:

1) *Feature learning stage*: the stage where the input characterizing features are learned.
2) *Optimization stage*: the stage where the final task-specific outputs are obtained from the features.

Many transfer learning approaches can be viewed as transferring the feature learning stage across neural network models, while each model learns its own distinct optimization stage corresponding to its specific task. For the mainstream transfer learning researches in the fields such as CV and NLP, the inputs in the problems are highly structured, while the targets are in much lower dimensions. Consequentially, it is clear where the feature learning stage and the optimization stage are within the neural network computation flow. However, conducting transfer learning on general mathematical optimization problems is a different story: the inputs, the outputs, and their mappings often lack discernible structures, resulting no clear distinction between the feature learning stage and the optimization stage in neural network computations. Consequentially, it is difficult to determine where within the trained model is the transferable knowledge (in the form of internal features), or even worse, if such transferable knowledge exists at all. Several researches have explored transfer learning on wireless communication problems [18]–[20]. These works achieved promising results, through techniques based on the application settings or objective characteristics that are problem specific.

In this paper, we propose a novel transfer learning approach to explicitly enforce the learning of transferable features within the neural network computation flow. Specifically, when training the model on the source task, besides the task-specific loss, we introduce an additional *reconstruction loss* to be minimized jointly: we let the neural network reconstruct the inputs using features from a specific hidden layer (which we refer to as the *feature layer*), and compute the input reconstruction loss.

Through minimizing this reconstruction loss, we encourage the features learned at the feature layer to be general and descriptive of the inputs, instead of being solely responsible for optimizing the source task. To perform transfer learning on the target task, we take the trained model parameters up to this feature layer as the optimized feature learning stage for the new model, and further train the remaining model parameters. We emphasize that with this approach, the features learned in the feature layer of the model are highly general and are not restricted to any specific task, and thus can be transferred for solving multiple tasks simultaneously without any re-training. Essentially, the proposed transfer learning approach is *target-task agnostic*. This is in contract to some transfer learning researches [17], [20]–[22] where the training is dedicated to a given source-task-to-target-task pair. We note that several works also explore the similar idea of encouraging input reconstruction from model internal features, in the field of semi-supervised learning [23], [24], and multi-task learning (a different field from transfer learning despite certain similarities) [25]. Nonetheless, as these works belong to different fields, they focus on different scenarios and problems. For transfer learning, we aim for high-level features that support neural networks to quickly adapt to new tasks under limited training and adjustments of model parameters.

For numerical simulations, we focus on a prevalent class of mathematical optimization problems: the resource allocation optimizations. We implement the proposed transfer learning approach to train neural networks to perform power control utility optimization for device-to-device (D2D) wireless networks. Specifically, we explore transfer learning between two different yet correlated objectives: the min rate and sum rate objectives. Optimization results suggest that our approach achieves knowledge transfer and mitigates over-fitting on limited target task data more effectively as compared to the regular transfer learning method.

II. Problem Formulation

A. General Setup: Source Task and Target Task Optimization

Among many variants of transfer learning formulations, we focus on the transfer learning setting where the source task and the target task share the same distribution on the inputs, and differ by their respective objectives. Let $\mathcal{S}$ denote the source task and $\mathcal{T}$ denote the target task, consider the optimization problems summarized by the following components:

- Input parameters $\mathbf{p}$ summarizing all environment information essential for optimization, which follow the same distribution in both $\mathcal{S}$ and $\mathcal{T}$;
- Optimization variables for $\mathcal{S}$: $\mathbf{x}_s$;
- Objective (or utility) for $\mathcal{S}$: $u_s(\mathbf{x}_s)$;
- Optimization variables for $\mathcal{T}$: $\mathbf{x}_t$;
- Objective (or utility) for $\mathcal{T}$: $u_t(\mathbf{x}_t)$.

If $\mathcal{T}$ and $\mathcal{S}$ are supervised learning tasks, then $u_s(\mathbf{x}_s)$ and $u_t(\mathbf{x}_t)$ are also dependent on the ground-truth labels, which we omit in our notations as they are not variables to be optimized.

To optimize $\mathcal{S}$ and $\mathcal{T}$, we utilize neural networks to compute the optimal values for optimization variables. Specifically:

- Let $\mathcal{F}_{\Theta_s}$ with trainable parameters Θ_s denote the neural network mapping for optimizing $\mathcal{S}$:
$$\mathcal{F}_{\Theta_s}(\mathbf{p}) = \mathbf{x}_s \tag{1}$$
- Let $\mathcal{F}_{\Theta_t}$ with trainable parameters Θ_t denote the neural network mapping for optimizing $\mathcal{T}$:
$$\mathcal{F}_{\Theta_t}(\mathbf{p}) = \mathbf{x}_t \tag{2}$$

Under the transfer learning setting, $\mathcal{S}$ and $\mathcal{T}$ are correlated, in the sense that features extracted from $\mathbf{p}$ for optimizing $u_s(\mathbf{x_s})$ and $u_t(\mathbf{x_t})$ are partially similar. Nonetheless, $\mathcal{S}$ and $\mathcal{T}$ are still two different tasks. Features exclusively learned for one task are not optimal for the other one. Therefore, the features still need to be relatively general for being transferable between $\mathcal{S}$ and $\mathcal{T}$. Moreover, due to reasons such as the cost and overhead of data acquisition, the data is limited for training Θ_t on $\mathcal{T}$. This assumption is particularly relevant for scenarios where the target task $\mathcal{T}$ is adapted on the fly after the model deployment. Consequentially, to effectively learn $\mathcal{F}_{\Theta_t}$, we need to utilize the correlation between $\mathcal{S}$ and $\mathcal{T}$, and transfer over the knowledge already learned in $\mathcal{F}_{\Theta_s}$ that is also generalizable enough for solving $\mathcal{T}$.

B. Wireless Network Utilities

For engineering applications, we study the power control optimization for different link rate utilities in D2D wireless networks. Consider a wireless network with N D2D links that transmit independently over the frequency band of bandwidth w with full frequency reuse. Let $\mathbf{G} = \{g_{ij}\}_{i,j\in\{1\ldots N\}}$ denote the channel gain from the j-th transmitter to the i-th receiver. Let P_i denote the maximum transmission power for the i-th transmitter. The problem of power control is to find the optimal values of the variables $\mathbf{x} = \{x_i\}_{i\in\{1\ldots N\}}$, where $x_i \in [0,1]$ denote the percentage of P_i that the i-th transmitter transmits at. Under a specific power control solution $\mathbf{x}$, the i-th link has the following achievable rate:

$$r_i = w \log\left(1 + \frac{g_{ii}P_i x_i}{\sum_{j\neq i} g_{ij}P_j x_j + \sigma^2}\right), \tag{3}$$

where σ^2 denotes the background noise power level.

For optimization objectives, we consider two link rate utility functions which are important under different application scenarios:

- The sum rate:
$$u = \sum_{i=1}^{N} r_i \tag{4}$$
- The min rate
$$u = \min_{i=1\ldots N} r_i \tag{5}$$

Examining (4) and (5), they are correlated in the sense that a set of higher rates over all links leads to a higher objective value. Both objectives would benefit from proper interference mitigation. On the other hand however, these two objectives differ greatly in term of fairness among links: (5)

ensures complete fairness by optimizing the worst link rate; (4) largely ignores fairness since the optimal sum rate might be achieved through heavily utilizing strong links. With this distinction, conducting transfer learning between (4) and (5) is challenging, as certain features crucial in optimizing one objective could be irrelevant or even counter-productive in optimizing the other objective.

III. Transfer Learning with Reconstruction Loss

A. *Information within Neural Network Computation Flow*

A neural network consists of consecutive hidden layers of neurons computing non-linear functions (i.e. activations), forming a computation flow as the input-to-output mapping. For a regular neural network learning one specific input-to-output mapping, the features computed by each hidden layer show a general pattern: from the input to the output, the hidden features in each layer gradually reduce the information contained about the inputs, while maintaining only the information necessary for predicting the outputs [26]–[28]. Nonetheless, this information flow pattern might not be desirable in transfer learning. Instead, we desire the features learned by the model to be generalizable and retain sufficient information on inputs for being transferable to new incoming tasks.

When using transfer learning on CV or NLP applications, it is relatively clear which features hold sufficient information about inputs. Specifically, with highly structured inputs, the neural network computation flow learned under regular training is likely to be structured already: the entire flow can be divided into a feature learning stage and an optimization stage. Take convolutional neural networks as examples, the feature learning stage includes the convolution layers that compute general high-level features (such as edges, pixel intensities, or color gradients over the input image), followed by the optimization stage consisting of fully connected layers that process these high-level features into task-specific features (such as the existence of objects of interest) and compute the outputs (e.g. classification class scores). To conduct transfer learning over correlated CV tasks, the convolutional layers are shared among the models as the feature learning stage, while the fully connected layers of each model are further trained on the per-task basis, such as the approaches in [11], [13], [14].

However, for general mathematical optimization problems, the inputs lack structures in most cases. Under regular training methods, the resulting neural network computation flow is not clearly divided by stages, with internal features gradually becoming more and more task-specific layer by layer. As the result, it is difficult to identify or explicitly encourage the learning of transferable features.

B. *Source Task Training with Added Reconstruction Loss*

To tackle the challenges of transfer learning on general mathematical optimization problems, we propose a novel transfer learning approach to encourage the learning of transferable features. Specifically, when training the model on the source task, on top of the regular task-based loss, we introduce in addition a loss term for input reconstruction.

Fig. 1 illustrates the proposed transfer learning approach. We adopt the most general fully-connected neural network architecture. Within the neural network, we select a hidden layer as the *feature layer* where we encourage the transferable features to be computed. There is no particular constraint on the selection of the feature layer, as long as there is sufficient transformation capacity (by having hidden layers) both from inputs to the selected layer and from the selected layer to outputs. From this feature layer, we add in a *reconstruction stage* as a separate branch in the computation flow, in parallel to the optimization stage. The reconstruction stage aims to construct the inputs $\mathbf{p}$ starting from the feature layer. We denote the corresponding reconstruction loss by $\mathcal{L}_R$. Together with the original loss associated with optimizing the source task utility $u_s(\mathbf{x}_s)$, which we denote by $\mathcal{L}_S$, the loss function $\mathcal{L}$ that we use to train $\mathcal{F}_{\Theta_s}$ is:

$$\mathcal{L} = \mathcal{L}_S + \alpha \mathcal{L}_R \tag{6}$$

where α is the relative weighting scalar between the loss terms. With $\mathcal{F}_{\Theta_s}$ trained on $\mathcal{L}$ as in (6), the feature layer computes features that are pertinent to the source task optimization, while still being comprehensive and informative about the inputs. Therefore, these features are highly transferable to different tasks that are relevant to the source task.

Corresponding to our previous discussion on the neural network computation flow, the computation in the neural network up to the feature layer can be regarded as the feature learning stage, while the computation after the feature layer can be regarded as the optimization stage. We emphasize that our approach is *target-task agnostic*, since no knowledge of the target task is needed throughout the training procedure.

C. *Transfer Learning by Sharing Feature Layer*

After training the neural network $\mathcal{F}_{\Theta_s}$ on the source task $\mathcal{S}$ as described in Section III-B, transfer learning on the target task $\mathcal{T}$ is straightforward. We first transfer the subset of the neural network parameters Θ_s up to the feature layer to Θ_t, and leave the remaining parameters in Θ_t unassigned. When training $\mathcal{F}_{\Theta_t}$, we freeze all the transferred parameters so they would remain unchanged during training. With the proposed transfer learning approach, the features computed in the feature layer of $\mathcal{F}_{\Theta_t}$ are already valuable for optimizing $\mathcal{T}$ before any target task training. The parameters after the feature layer in Θ_t are further trained specifically for $\mathcal{T}$. We train these parameters with the regular loss associated with the target task utility $u_t(\mathbf{x}_t)$, which we denote by $\mathcal{L}_{\mathcal{T}}$. With the number of trainable parameters greatly reduced, along with the features from the feature layer already being relevant, only a small amount of data for $\mathcal{T}$ would already be sufficient for training a well-performing $\mathcal{F}_{\Theta_t}$.

IV. Numerical Simulations

A. *Wireless Network Settings*

We simulate each wireless network containing $N = 10$ links randomly deployed in a 150m$\times$150m region. We first generate the locations of transmitters uniformly within the

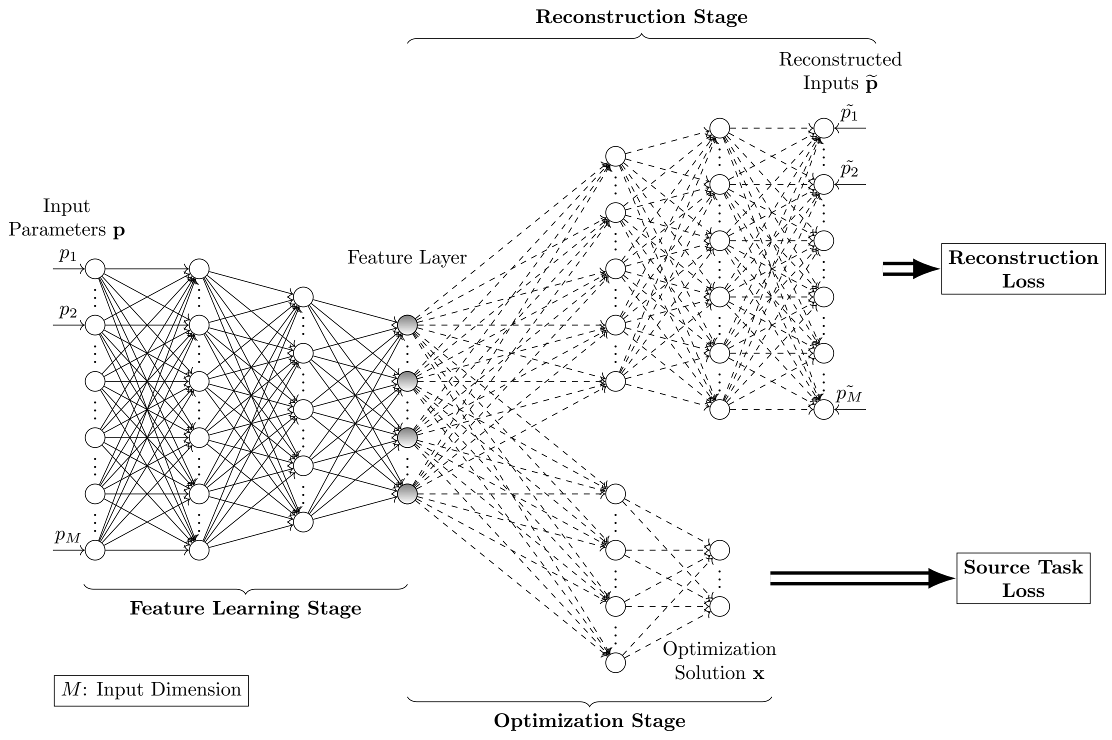

Fig. 1. Transfer Learning with Reconstruction Loss.

region, and then generate the locations of the receivers such that the direct-channel transceiver distances follow a uniform distribution in the interval of 5m∼25m. We impose a minimum of 5m distance between any transmitter and receiver that would form a cross-channel. We assume each transmitter has the maximum transmission power at 30dBm with a direct-channel antenna gain of 6dB, while the noise level is -150dBm/Hz. We assume 5MHz bandwidth at the mmWave frequency is available for transmission with full frequency reuse across the entire wireless network.

To simulate wireless channels, we assume that the channel gain of each channel is determined by three components:

- *Path-Loss*: modeled by the short-range outdoor model ITU-1411.
- *Shadowing*: modeled by the log-normal distribution with 8dB standard deviation.
- *Fast Fading*: modeled by Rayleigh fading with i.i.d circular Gaussian distribution of unit variance.

B. Neural Network Specifications

We collect N^2 channel gains for each layout into a N^2-dimensional vector as the input $\mathbf{p}$. We utilize the same neural network specifications for both $\mathcal{F}_{\Theta_s}$ and $\mathcal{F}_{\Theta_t}$, summarized in Table I. To train $\mathcal{F}_{\Theta_s}$, we use $\alpha = 3$ in (6) for the loss $\mathcal{L}$. With $N = 10$, the total numbers of trainable parameters for all three stages (including both weights and biases) are as following:

- *Feature Learning Stage*: 52900 parameters;
- *Optimization Stage*: 5070 parameters;
- *Reconstruction Stage*: 40300 parameters.

TABLE I
NEURAL NETWORK ARCHITECTURE (N: NUMBER OF LINKS)

Stages	Layers	Number of Neurons
Feature Learning Stage	1st	$1.5N^2$
	2nd	$1.5N^2$
	Feature Layer	N^2
Optimization Stage	1st	$4N$
	2nd	$2N$
	Output Layer	N
Reconstruction Stage	1st	$2N^2$
	Reconstruct Layer	N^2

As the optimization stage only has a small number of parameters, training on $\mathcal{T}$ via transfer learning requires little data.

We introduce two neural network based benchmarks, each with identical network architecture for the feature learning stage and the optimization stage as in Table I:

- *Conventional Transfer Learning*: train $\mathcal{F}_{\Theta_s}$ on the source task on the loss $\mathcal{L}_S$, then transfer all the parameters in Θ_s up to the feature layer to Θ_t.
- *Regular Learning*: train $\mathcal{F}_{\Theta_s}$ and $\mathcal{F}_{\Theta_t}$ the same way as regular training, without knowledge transfer in between.

In the following context, we refer our proposed transfer learning method by *Transfer Learning with Reconstruction*.

In terms of existing transfer learning researches on wireless communications [18]–[20], [18] adopts the conventional transfer learning method. [19] and parts of [20] focus on the setting

TABLE II
DATA SET SPECIFICATIONS

Task	Training Set Samples	Validation Set Samples
$\mathcal{S}$	5×10^5	5000
$\mathcal{T}$	1000	5000*

* A large validation set is used to ensure accurate early stopping. May not be available in realistic scenarios when the target task data is limited.

where the inputs $\mathbf{p}$ come from different distributions for $\mathcal{S}$ and $\mathcal{T}$. Other relevant applications in [20] focus on knowledge transfer among models operating at different parts or locations within the same environment. For the transfer learning setting in Section II-A, it is sufficient to study the two above-mentioned neural network based methods as benchmarks.

To train each neural network, we directly formulate the task-based losses $\mathcal{L}_{\mathcal{S}}$ and $\mathcal{L}_{\mathcal{T}}$ as the utilities $u_s(\mathbf{x}_s)$ and $u_t(\mathbf{x}_t)$. This is accomplished by letting the neural network compute the utility with $\mathbf{p}$ after $\mathbf{x}$ is computed at the training stage. The model is then trained via gradient ascent on the utility value. This loss formulation is shown to be effective in [2]–[4].

C. Data Set Specifications

We first specify the training and validation datasets. For $\mathcal{F}_{\Theta_s}$, the entire model including all three stages need to be trained from scratch. Meanwhile for $\mathcal{F}_{\Theta_t}$, only the optimization stage needs to be trained. Correspondingly, we utilize the data set sizes listed in Table II. Each sample of a D2D wireless network is generated according to Section IV-A. We note that the data set sizes in Table II are smaller than the number of trainable neural network parameters, especially for the training data on $\mathcal{T}$. We select these small data sets to illustrate that new target tasks can be adapted on-the-fly with minimal training overhead, as well as to show that the proposed transfer learning approach is effective in training and is robust to over-fitting. We use large validation sets to ensure accurate early-stopping [29] when training under each approach. However, realistically we may not have sufficient data for validation on $\mathcal{T}$. Therefore, in Section IV-E, we also include numerical results where no early-stopping is performed during target task training.

For the test data set (on which both utilities are evaluated), we generate 2000 new samples according to Section IV-A to obtain performance statistics over all the methods.

D. Evaluation Settings

Besides the two neural network based benchmarks, we also include the following traditional mathematical optimization algorithms serving as performance upper-bound baselines (with the cost of having much higher computational complexities):

- *Geometric Programming (GP)* [30]: mathematical algorithm for solving the min-rate optimization.
- *Fractional Programming (FP)* [31]: mathematical algorithm for solving the sum-rate optimization.

We train and evaluate each neural network based method under the transfer learning direction: Sum Rate $\mathcal{S} \rightarrow$ Min Rate $\mathcal{T}$, while only evaluating FP on $\mathcal{S}$ and GP on $\mathcal{T}$.

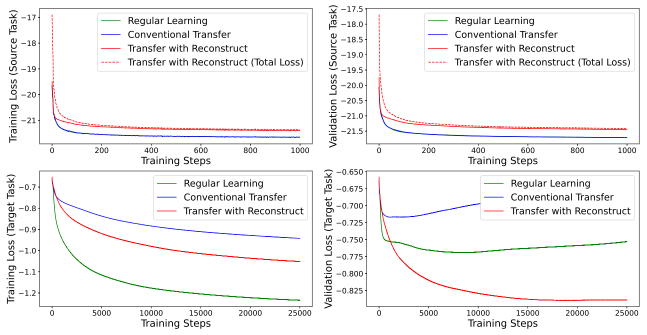

Fig. 2. Training Curves for Transfer Learning (the top two figures provide training and validation curves for the source task; the bottom two figures provide training and validation curves for the target task).

E. Numerical Results and Analysis

We present both $\mathcal{S}$ and $\mathcal{T}$ performances, averaged over all 2000 test wireless networks, in Table III. We first examine the *results with early stopping*, which requires additional data reserved as the validation set. Shown by the numerical results, the transfer learning-with-reconstruction approach achieves the best target-task performance among the neural network based methods, with a 11% improvement over the regular learning approach, and a 17% improvement over the conventional transfer learning approach. Our approach achieves these improvements with the trade-off of a 1% reduction on the sum-rate results as compared to both neural network based benchmarks. This slight performance loss on $\mathcal{S}$ is expected since our approach utilizes the training loss (6) that does not exclusively target to optimizing the source-task utility.

To better understand the results, we provide the training curves on $\mathcal{S}$ and $\mathcal{T}$ for all the methods in Fig. 2. Note that for our proposed approach, we have plotted two losses on $\mathcal{S}$: the source task based loss shown by the solid line ($\mathcal{L}_{\mathcal{S}}$ in (6)), and the total loss shown by the dotted line ($\mathcal{L}$ in (6)). As evident by the validation curves on $\mathcal{T}$ (in the bottom-right figure), while both the conventional transfer learning and the regular learning approaches plateau early in validation loss and then regress due to over-fitting, our approach enables the model to learn at a much more sustainable pace from the very limited training data, without any noticeable over-fitting.

The effects of over-fitting are better shown when no early-stopping is performed in training each model on $\mathcal{T}$. Shown by the *results without early stopping* in Table III, our approach maintains its performance (indicating the model does not over-fit throughout training) and achieves even larger margins on $\mathcal{T}$, with 13% and 25% improvements over the regular learning and conventional transfer learning approach respectively.

V. CONCLUSION

Transfer learning has great potential and wide applicability in general mathematical optimizations such as wireless communication problems. However, as the inputs and computation in such optimizations often lack structures, it is challenging to identify within the neural network the transferable features.

TABLE III
TRANSFER LEARNING PERFORMANCES

Task	Method	Result with Early Stopping (Mbps)	Result without Early Stopping (Mbps)
$\mathcal{S}$: Sum-Rate	Regular Learning	155.69	N/A
	Conventional Transfer Learning	155.67	
	Transfer Learning with Reconstruction	153.96	
	FP	157.45	
$\mathcal{T}$: Min-Rate	Regular Learning	5.39	5.32
	Conventional Transfer Learning	5.13	4.80
	Transfer Learning with Reconstruction	6.00	6.00
	GP	9.16	N/A

This paper proposes a novel transfer learning approach for learning general and transferable features at specified locations in neural networks. Specifically, we introduce a reconstruction stage to the neural network starting from a pre-specified hidden layer, i.e., the feature layer. By enforcing input reconstruction from the feature layer, the learned features are descriptive of the inputs, and therefore transferable to target tasks. Simulation results on wireless network utility optimizations suggest that the proposed approach outperforms the conventional transfer learning and is robust against over-fitting. We hope this work could open up further exploration on bridging transfer learning with general mathematical optimizations.

REFERENCES

[1] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to optimize: Training deep neural networks for interference management," vol. 66, no. 20, pp. 5438–5453, Aug. 2018.

[2] W. Cui, K. Shen, and W. Yu, "Spatial deep learning for wireless scheduling," *IEEE J. Sel. Areas Commun.*, vol. 37, pp. 1248–1261, June 2019.

[3] F. Liang, C. Shen, W. Yu, and F. Wu, "Towards optimal power control via ensembling deep neural networks," *IEEE Trans. Commun. (TCOM)*, vol. 68, no. 3, pp. 1760–1776, Mar. 2020.

[4] W. Cui, K. Shen, and W. Yu, "Deep learning for robust power control for wireless networks," in *IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, May 2020.

[5] K. Gregor and Y. Lecun, "Learning fast approximations of sparse coding," in *Int. Conf. Mach. Learn. (ICML)*, Jun. 2010.

[6] S. Khobahi and M. Soltanalian, "Model-based deep learning for one-bit compressive sensing," *IEEE Trans. Signal Process.*, vol. 68, pp. 5292–5307, Sep. 2020.

[7] K. M. Attiah, F. Sohrabi, and W. Yu, "Deep learning approach to channel sensing and hybrid precoding for TDD massive MIMO systems," in *IEEE Global Commun. Conf. (GLOBECOM) Workshops*, Dec. 2020.

[8] F. Sohrabi, T. Jiang, W. Cui, and W. Yu, "Active sensing for communications by learning," *IEEE J. Sel. Areas Commun.*, Mar. 2022, early access. doi:10.1109/JSAC.2022.3155496.

[9] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, Jul. 2020.

[10] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in *Int. Conf. Mach. Learn. (ICML)*, Jul. 2015.

[11] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Int. Conf. Mach. Learn. (ICML)*, Jul. 2015.

[12] B. Sun and K. Saenko, "Deep CORAL: Correlation alignment for deep domain adaptation," in *Eur. Conf. Comput. Vision (ECCV)*, Oct. 2016.

[13] K. Gopalakrishnan, S. K. Khaitan, A. Choudhary, and A. Agrawal, "Deep convolutional neural networks with transfer learning for computer vision-based data-driven pavement distress detection," *Construction Building Mater.*, vol. 157, pp. 322–330, Dec. 2017.

[14] S. Tammina, "Transfer learning using VGG-16 with deep convolutional neural network for classifying images," *Int. J. Sci. Res. Publications*, vol. 9, no. 10, pp. 143–150, Oct. 2019.

[15] F. Zhuang, P. Luo, H. Xiong, Q. He, Y. Xiong, and Z. Shi, "Exploiting associations between word clusters and document classes for cross-domain text categorization," *Statist. Anal. Data Mining*, vol. 4, no. 1, pp. 100–114, Feb. 2011.

[16] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *Int. Conf. Mach. Learn. (ICML)*, Jun. 2011.

[17] J. Deng, Z. Zhang, E. Marchi, and B. Schuller, "Sparse autoencoder-based feature transfer learning for speech emotion recognition," in *Humaine Assoc. Conf. Affect. Comput. Intell. Interact.*, Sep. 2013.

[18] H. Yang, J. Jee, G. Kwon, and H. Park, "Deep transfer learning-based adaptive beamforming for realistic communication channels," in *Int. Conf. Inf. Commun. Technol. Convergence (ICTC)*, Oct. 2020.

[19] Y. Yuan, G. Zheng, K. Wong, B. Ottersten, and Z. Luo, "Transfer learning and meta learning-based fast downlink beamforming adaptation," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1742–1755, Mar. 2021.

[20] M. Wang, Y. Lin, Q. Tian, and G. Si, "Transfer learning promotes 6G wireless communications: Recent advances and future challenges," *IEEE Trans. Rel.*, vol. 20, no. 3, pp. 1742–1755, Mar. 2021.

[21] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, "Analysis of representations for domain adaptation," in *Conf. Neural Inf. Process. Syst. (NeurIPS)*, Dec. 2006.

[22] F. Zhuang, X. Cheng, P. Luo, S. J. Pan, and Q. He, "Supervised representation learning: Transfer learning with deep autoencoders," in *Int. Joint Conf. Artif. Intell. (IJCAI)*, Jul. 2015.

[23] M. A. Ranzato and M. Szummer, "Semi-supervised learning of compact document representations with deep networks," in *Int. Conf. Mach. Learn. (ICML)*, Jul. 2008.

[24] J. Weston, F. Ratle, and R. Collobert, "Deep learning via semi-supervised embedding," in *Int. Conf. Mach. Learn. (ICML)*, Jul. 2008.

[25] X. Lai, X. Wu, and L. Zhang, "Autoencoder-based multi-task learning for imputation and classification of incomplete data," *Appl. Soft Comput.*, vol. 98, no. 1, Oct. 2020.

[26] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *IEEE Inf. Theory Workshop (ITW)*, Apr. 2015.

[27] R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," Mar. 2017, [Online] Available: https://arxiv.org/abs/1703.00810.

[28] A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox, "On the information bottleneck theory of deep learning," in *Int. Conf. Learn. Representations (ICLR)*, Feb. 2018.

[29] R. Caruana, S. Lawrence, and L. Giles, "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping," in *Conf. Neural Inf. Process. Syst. (NeurIPS)*, Dec. 2000.

[30] M. Chiang, C. W. Tan, D. P. Palomar, D. O'nell, and D. Julian, "Power control by geometric programming," *IEEE Trans. Wireless Commun.*, vol. 6, no. 7, pp. 2640–2651, Jul. 2007.

[31] K. Shen and W. Yu, "FPLinQ: A cooperative spectrum sharing strategy for device-to-device communications," in *IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2323–2327.