UNIVERSITY OF CALIFORNIA SAN DIEGO

Deep Learning with Estimation and Complexity Guarantees for Signal Processing

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Electrical Engineering (Signal and Image Processing)

by

Kuan-Lin Chen

2024

DEDICATION

To my family and teachers.

EPIGRAPH

Stay hungry. Stay foolish.

Steve Jobs

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

It is my pleasure to express my deepest gratitude to my advisor, Prof. Bhaskar D. Rao, for his invaluable guidance, patience, and unwavering support throughout my Ph.D. studies. His mentorship extended beyond academic advice, offering encouragement and insights that have profoundly impacted my professional development and personal growth. Thank you for being a pivotal part of my academic journey. I am forever grateful.

I would also like to thank Prof. Sanjoy Dasgupta, Prof. Philip E. Gill, Prof. Piya Pal, Prof. Xiaolong Wang, and Dr. Harinath Garudadri for serving on my doctoral committee. I am deeply grateful to Prof. Pal for the teaching assistant opportunity in the undergraduate optimization course, where I was honored to receive the 2023-2024 ECE Best TA Award. Her support and advocacy have been invaluable in opening doors and providing me with opportunities that I deeply appreciate. I extend special thanks to Dr. Garudadri for his guidance and support that have been crucial to my research journey and my life at UC San Diego.

Additionally, I would like to express my gratitude to Prof. Nuno Vasconcelos, Prof. fred harris, Prof. Behrouz Touri, and Prof. John Iversen, for their support and guidance.

It was my privilege to collaborate with brilliant graduate students during my PhD studies. I am deeply grateful to Ching-Hua Lee for being an excellent mentor and true friend. His guidance has not only enriched my research but also brought color to my life in San Diego. I cherish the time we spent together at Atkinson Hall. I would also like to thank Ross Greer for being an excellent collaborator. Our proposal won the Qualcomm Innovation Fellowship (QIF) in 2022, of which I am immensely proud. In addition, I extend my gratitude to Tzu-Han Zoe Cheng for her creativity and collaboration. Together, we won the Innovative Research Grants (IRG) Award from the Kavli Institute for Brain and Mind (KIBM) in 2021. I also appreciate the contributions of Dhiman Sengupta, Alice Sokolova, Mingchao Liang, and Wenyu Zhang. I have learned so much from our insightful discussions.

It has been a pleasure to be a part of the UCSD DSP Lab. I would like to express my gratitude to my labmates—Govind R. Gopal, Rohan R. Pote, Aditya Sant, Hitesh D. Khunti, Gokce Sarar, Rushabha Balaji, and Vinay Kanakeri—for their valuable discussions, collaborative brainstorming sessions, cherished friendship, and company during coffee hours and on walks to Price Center.

I am grateful for the invaluable experiences gained during my internships at Meta and Qualcomm. These opportunities provided me with useful skills and exposure to real-world challenges. I extend my thanks to my intern managers, Daniel D. E. Wong, Sarmad Malik, and Asif Mohammad, for their mentorship, guidance, and support throughout these enriching experiences. Also, I would like to thank my collaborators and mentors, Ke Tan, Buye Xu, Anurag Kumar, Vamsi Krishna Ithapu, Julien Pons, Xianwen Wu, and Srikar Sastry, for their constructive feedback and fruitful discussions. The knowledge and insights gained during my internships have been instrumental in shaping my career path and personal growth.

Finally, I am profoundly grateful to my family, A-Chu, Wen-Chung, Li-Fan, and Guan-Hua, for their love, support, and encouragement throughout my academic journey, and my partner, Yun, for her inspiration, care, and love. Their belief in me has been a constant source of strength and motivation. I am deeply appreciative of the sacrifices they have made to help me achieve my goals. Thank you for always being there for me and for believing in my dreams.

This work was supported in part by IEEE Signal Processing Society Scholarship, in part by KIBM Innovative Research Grant Award, and in part by NSF under Grant CCF-2225617 and Grant CCF-2124929.

Chapter 1, in full, is a reprint of the material as it appears in K.-L. Chen, H. Garudadri, and B. D. Rao, “Improved bounds on neural complexity for representing piecewise linear functions,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. The dissertation author was the primary investigator and author of this material.

Chapter 2, in full, is a reprint of the material as it appears in K.-L. Chen, C.-H. Lee, H. Garudadri, and B. D. Rao, “ResNEsts and DenseNEsts: Block-based DNN models with improved representation guarantees,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. The dissertation author was the primary investigator and author of this material.

Chapter 3, in part, is a reprint of the material as it appears in K.-L. Chen and B. D. Rao, "Subspace representation learning for sparse linear arrays to localize more sources than sensors: A deep learning methodology," which was submitted to *IEEE Transactions on Signal Processing* in 2024, with its preprint available at arXiv (`https://arxiv.org/abs/2408.16605`). The dissertation author was the primary investigator and author of this material.

VITA

2016	B.S.E. in Electrical Engineering, National Taiwan University, Taiwan
2019	M.S. in Electrical Engineering, University of California San Diego, USA
2024	Ph.D. in Electrical Engineering, University of California San Diego, USA

PUBLICATIONS

K.-L. Chen and B. D. Rao, "Subspace representation learning for sparse linear arrays to localize more sources than sensors: A deep learning methodology," submitted to *IEEE Transactions on Signal Processing*, 2024. [Online]. Available: `https://arxiv.org/abs/2408.16605`.

K.-L. Chen, C.-H. Lee, B. D. Rao, and H. Garudadri, "A DNN based normalized time-frequency weighted criterion for robust wideband DoA estimation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

K.-L. Chen, D. D. E. Wong, K. Tan, B. Xu, A. Kumar, and V. K. Ithapu, "Leveraging heteroscedastic uncertainty in learning complex spectral mapping for single-channel speech enhancement," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

K.-L. Chen, H. Garudadri, and B. D. Rao, "Improved bounds on neural complexity for representing piecewise linear functions," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

K.-L. Chen, C.-H. Lee, H. Garudadri, and B. D. Rao, "ResNEsts and DenseNEsts: Block-based DNN models with improved representation guarantees," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

K.-L. Chen, C.-H. Lee, B. D. Rao, and H. Garudadri, "Jointly leveraging decorrelation and sparsity for improved feedback cancellation in hearing aids," in *European Signal Processing Conference (EUSIPCO)*, 2020.

K.-L. Chen, C.-H. Lee, B. D. Rao, and H. Garudadri, "A generalized proportionate-type normalized subband adaptive filter," in *Asilomar Conference on Signals, Systems, and Computers*, 2019.

T.-H. Cheng, **K.-L. Chen**, J. Schubert, Y.-P. Chen, T. Brown, and J. Iverson, "Similar hierarchical representation of speech and other complex sounds in the brain and deep residual networks: An MEG study," in *Conference of the International Speech Communication Association (Interspeech)*, 2023.

H. Garudadri, C.-H. Lee, **K.-L. Chen**, F. Harris, and B. D. Rao, “Mitigating acoustic feedback in hearing aids with frequency warping by all-pass networks,” *U.S. Patent No. 11,849,283*, 2023.

A. Sokolova, D. Sengupta, **K.-L. Chen**, R. Gupta, B. Aksanli, f. harris, and H. Garudadri, “Multirate audiometric filter bank for hearing aid devices,” in *Asilomar Conference on Signals, Systems, and Computers*, 2021.

C.-H. Lee, **K.-L. Chen**, f. harris, B. D. Rao, and H. Garudadri, “On mitigating acoustic feedback in hearing aids with frequency warping by all-pass networks,” in *Conference of the International Speech Communication Association (Interspeech)*, 2019.

ABSTRACT OF THE DISSERTATION

Deep Learning with Estimation and Complexity Guarantees for Signal Processing

by

Kuan-Lin Chen

Doctor of Philosophy in Electrical Engineering (Signal and Image Processing)

University of California San Diego, 2024

Bhaskar D. Rao, Chair

This dissertation presents advancements in both the theory and applications of deep learning. The objectives include reducing complexity, enhancing interpretability, and offering superior methodologies for solving signal processing problems.

On the theoretical side, we develop two major advances: one for complexity and another for interpretability. For complexity, we establish tighter upper bounds for the number of computational units required for a rectified linear unit (ReLU) neural network to represent or compute any given continuous piecewise linear (CPWL) function. Specifically, we prove that any CPWL function can be represented by a ReLU network whose number of hidden neurons is at most a quadratic function of the number of pieces of the CPWL function. This quadratic bound is independent of the input dimension and outperforms previous bounds exponentially, holding the state-of-the-art in the literature. When the number of linear components is also known, this bound reduces to a bilinear bound. On the other hand, a new upper bound on the number of pieces in terms of the number of linear components is proved, enabling different descriptions of neural complexity. In addition to existence, a polynomial-time algorithm is developed to identify a neural network that satisfies these tighter bounds, shedding light on reverse-engineering and network pruning. For interpretability, we prove that, the most popular building block in deep learning, the skip connection, can guarantee to improve representations over residual blocks when the expansion layer is sufficiently large. Its implications explain (a) why the residual network (ResNet) can avoid the degradation problem and be scaled to thousands of layers, (b) why the wide ResNet is superior than the ResNet, and (c) why the bottleneck blocks are more economical than the basic block. We design a simplified architecture called the residual nonlinear estimator (ResNEst) and propose a new architecture called the augmented ResNEst to develop guarantees for the ResNet. Under mild assumptions, it is proved that every local minimizer in the ResNEst can be a global minimizer, despite the nonconvex optimization landscape, implying that any local minimizer can be provably better than the best linear predictor.

On the application side, we develop a new deep learning-based methodology, subspace representation learning, for the classic direction-of-arrival (DoA) estimation problem in array processing. The codomain of a deep learning model is defined as a union of Grassmannians reflecting signal subspaces of different dimensions. A family of distance functions on Grassmannians is proposed. In particular, we use geodesic distances to train a model and prove that it is possible for a ReLU network to approximate signal subspaces. Because a subspace is invariant to the selection of its bases, our methodology enlarges the solution space of a model compared to existing approaches learning covariance matrices. Furthermore, due to its geometry-agnostic nature, our methodology is robust to array imperfections. Numerical results show that subspace representation learning significantly outperforms existing semidefinite programming-based and deep learning-based covariance matrix reconstruction approaches for a wide range of scenarios.

Introduction

Over the past decade, machine learning-based methodologies have demonstrated impressive accuracy and capabilities across various fields of signal processing, including audio and speech processing [Lu et al., 2013, Chen et al., 2023b], medical imaging [Ronneberger et al., 2015], data compression [Gregor et al., 2016], array processing [Papageorgiou et al., 2021, Chen and Rao, 2024], and wireless communications [Gao et al., 2023]. Problems in signal processing that were once addressed using linear techniques or handcrafted modeling approaches are now being transformed by data-driven methods employing deep neural networks (DNNs), resulting in significant performance gains. While DNN-based signal processing solutions are increasingly popular, this paradigm shift also introduces many new challenges, such as high complexity and limited interpretability. Unlike conventional signal processing, these challenges make DNN-based solutions costly and occasionally less reliable, impeding their practical application and accessibility. Therefore, the focus of this dissertation is to advance the fundamental theory of DNN models and, based on these improved theoretical principles, develop state-of-the-art signal processing methodologies.

0.1 Deep neural networks

The journey of DNNs began in 1943 with McCulloch and Pitts' seminal work, where they proposed a logical calculus of neural activity, laying the foundation for neural networks. This conceptual framework was further advanced in 1958 by Rosenblatt who introduced the *perceptron* model, a one-layer neural network for binary classification. Then, the multilayer perceptron (MLP) was developed based on the perceptron. Figure 0.1 illustrate a three-hidden layer MLP. While training an MLP or a convolutional neural network (CNN) with more than

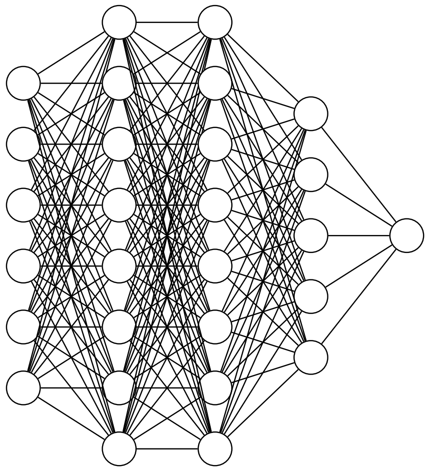
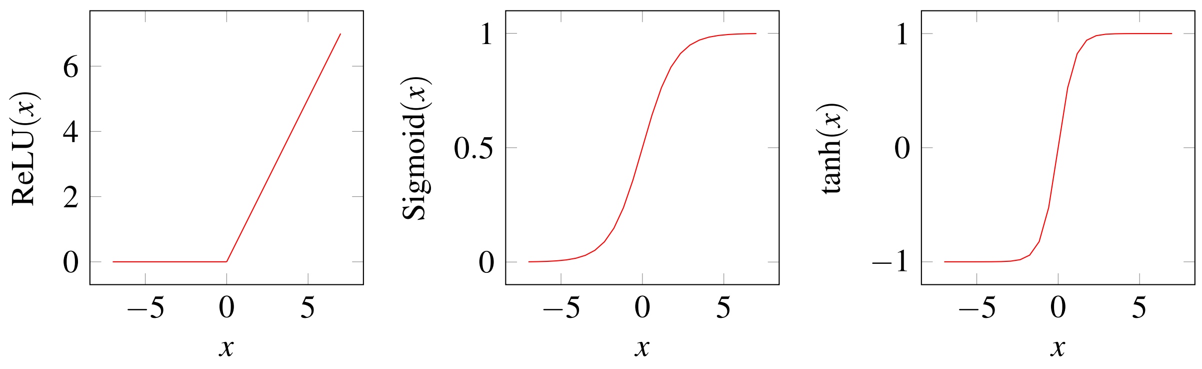


Figure 0.1. An illustration of a three-hidden-layer fully connected network. It has six input neurons and one output neuron.

one hidden layer had been studied for decades prior [Fukushima, 1980, LeCun et al., 1989, 1998], it was not until 2012 that DNNs or deep learning truly came into the spotlight with the advent of AlexNet by Krizhevsky et al. [2012], which has seven hidden layers, including five convolutional layers and two fully connected layers. This deep convolutional neural network architecture revolutionized computer vision by significantly improving image classification accuracy in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [Russakovsky et al., 2015]. The success of AlexNet marked a pivotal moment in the history of deep learning, showcasing the potential of DNNs in machine learning and artificial intelligence (AI). Since AlexNet, many models have been proposed, including VGG [Simonyan and Zisserman, 2015a], U-Net [Ronneberger et al., 2015], ResNet [He et al., 2016a], DenseNet [Huang et al., 2017], and Transformer [Vaswani et al., 2017], to name a few. Each of these models features specific designs, arrangements, or types of layers used in a neural network model. The specific design of a neural network model is commonly referred to as its architecture. Compared to architectures developed prior to the advent of deep learning, the two fundamental differences are the activation function and the number of hidden layers. Modern architectures use the rectifier linear unit (ReLU) activation function and many hidden layers. For example, the commonly used ResNet architecture uses the ReLU activation and can be scaled up to more than $1,000$ layers without degrading its performance [He et al., 2016b].

0.1.1 ReLU and continuous piecewise linear functions

The adoption of nonlinear activation functions is crucial for a neural network to be able to learn a nonlinear function. For example, any number of compositions of affine mappings gives an affine function, yielding the so-called deep linear network. A deep linear network is incapable of learning nonlinear functions because it is within the family of affine mappings. Figure 0.2 shows three different activation functions. Before deep learning gained popularity, many neural networks relied on smooth activation functions such as the sigmoid function and hyperbolic tangent, which introduce nonlinearity and make it possible for a network to approximate any continuous function on a compact subset of $\mathbb{R}^n$ [Hornik et al., 1989, Cybenko, 1989]. However, these activations are limited in their effectiveness for very deep networks due to issues like vanishing gradients [Goodfellow et al., 2016]. In contrast to smooth activation functions, modern DNNs use the ReLU activation function, which is non-smooth due to being non-differentiable at the origin. However, it does not suffer from the vanishing gradient problem, thus enabling the training of very deep neural network models. Furthermore, DNNs using the ReLU activation can be characterized by the family of continuous piecewise linear (CPWL) functions [Arora et al., 2018], contrasting with neural networks using smooth activation functions, which belong to a family of smooth functions. Because the family of CPWL functions is dense in the family of continuous functions, every neural network using the ReLU activation can approximate any continuous function on a compact subset of $\mathbb{R}^n$ if a sufficient number of hidden neurons are provided. Hence, one can prove the following universal approximation theorem. For any continuous function $f : [0,1]^n \to \mathbb{R}$, there exists a ReLU network $g : \mathbb{R}^n \to \mathbb{R}$ with finitely many



(b) ReLU (b) Sigmoid (c) Hyperbolic tangent

Figure 0.2. (a) The ReLU activation function $\text{ReLU}(x) = \max(x, 0)$. (b) The sigmoid function $\text{Sigmoid}(x) = \frac{1}{1+e^{-x}}$. (c) The hyperbolic tangent $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.

hidden neurons such that

$$\sup_{\mathbf{x}\in[0,1]^n} |f(\mathbf{x}) - g(\mathbf{x})| < \varepsilon \tag{1}$$

for every $\varepsilon > 0$. Based on this result, a more interesting question is: *How many hidden neurons does a network require to approximate a continuous function or represent a CPWL function?* Regarding continuous functions, the question has been answered by Yarotsky [2018]. With regard to CPWL functions, we develop a compact neural representation of CPWL functions and provide dimension-independent bounds in Chapter 1.

0.1.2 Architectures and skip connections

Intuitively, a deeper network should always lead to better performance because of increased expressivity. However, adding more layers to a model can lead to worse performance. This is the so-called degradation problem in deep learning. This issue is primarily due to vanishing or exploding gradients, making it difficult for the network to learn effectively. From the perspective of the optimization landscape, as the depth of a neural network increases, its optimization landscape can transition from being nearly convex to being highly chaotic [Li et al., 2018]. Residual networks (ResNets) address this problem through the use of skip connections, also known as residual connections. These connections allow gradients to flow more easily through the network by providing a shortcut around one or more layers, enabling the construction of very deep networks while improving performance [He et al., 2016b]. Many modern architectures also adopt skip connections, such as those used in the Transformer [Vaswani et al., 2017]. In fact, modern architectures often use building blocks or modules stacked in sequence, rather than purely stacking layers. Each block is designed to perform specific operations on a group of layers, helping the network scale its depth and learn better representations. This approach enables improved optimization, resulting in better performance or reduced complexity. For example, ResNet uses residual blocks, Transformer uses attention blocks, and SE network uses Squeeze-and-Excitation (SE) blocks [Hu et al., 2018]. Among many block designs, the residual block appears to be the most fundamental and widely used design. As the adoption of skip connections is one of the key elements in deep learning, we study the ResNet architecture by developing similar architectures, called ResNEst and augmented ResNEst, and provide theoretical guarantees on the optimization landscape of these networks in Chapter 2.

0.2 New learning-based methodologies for signal processing

Many array signal processing problems involve estimating parameters based on given observations. Conventional signal processing methods typically formulate these problems as optimization tasks under a guiding principle, such as the well-known maximum likelihood, while adhering to appropriate constraints. Since the original optimization problems are often nonconvex and high-dimensional, traditional approaches generally rely on convex relaxation or majorization-minimization techniques. However, deep learning offers an alternative. Because data can be collected or generated based on statistical assumptions, the aforementioned problems can be reformulated as supervised learning tasks. DNNs can be leveraged to model complex relationships between observations and target parameters, learning the mapping between them.

Recently, many such learning-based methodologies have become increasingly popular. For example, sparse Bayesian learning in sparse signal recovery [Tipping, 2001, Wipf and Rao, 2004] and the direction-of-arrival (DoA) estimation problem in array processing [Yang et al., 2014, Pote and Rao, 2023], which traditionally rely on optimization techniques, are now being revisited by learning-based methodologies [He et al., 2017, Barthelme and Utschick, 2021a, Wu et al., 2022, Chen and Rao, 2024]. As these methodologies continue to evolve, they are poised to revolutionize signal processing, offering more accurate and robust solutions to complex challenges that were difficult to address within the conventional optimization paradigm. In Chapter 3, we develop a novel methodology for DoA estimation in this context that achieves state-of-the-art performance.

0.3 Outline and contributions of the dissertation

In Chapter 1, we provide new complexity results for a ReLU network to exactly represent any given CPWL function. In this dissertation, we prove the first dimension-independent bounds which are tighter than previous bounds in the literature and establish a polynomial time algorithm to identify a network satisfying these tighter bounds for any given CPWL function. In particular, we prove that the number of hidden neurons required to exactly represent any CPWL function is $\mathscr{O}(q^2)$ where q is the number of pieces. This upper bound is invariant to the input dimension n. When the number of distinct linear components k of the CPWL function is given, we prove that $\mathscr{O}(q^2)$ reduces to $\mathscr{O}(kq)$, leading to a lower complexity since $k \leq q$. In addition, a new upper bound of q in terms of k is proved. If q is unknown, we prove that, in terms of k, the neural complexity of a CPWL function is at most polynomial growth for small n and factorial growth for the worst-case scenario.

In Chapter 2, we contribute to the interpretability and estimation properties of the widely used ResNet architecture by constructing new networks called Residual Nonlinear Estimators (ResNEsts) and augmented ResNEsts (A-ResNEsts). We prove that a ResNEst with a sufficiently large first layer can guarantee that representations over residual blocks improve monotonically. In addition, we prove that any local minimizer obtained from a ResNet outperforms any best linear predictor in terms of training performance under mild conditions. These results not only offer mathematical explanations for why ResNet architectures are immune to the well-known degradation problem associated with stacking layers in a plain neural network (a network without skip connections) but also provide ResNet architectures with optimization guarantees. Furthermore, we prove that, under mild conditions, every local minimizer obtained for the ResNEst architecture is a global minimizer despite the nonconvexity of its optimization landscape, and every saddle point has at least one direction with strictly negative curvature, demonstrating desirable optimization properties.

In Chapter 3, we propose a novel deep learning-based methodology called *subspace representation learning* for DoA estimation [Chen and Rao, 2024]. Given a sparse linear array (SLA), most existing methods for localizing more sources than sensors estimate the covariance matrix of a uniform linear array (ULA) by formulating a constrained optimization problem and solving its semidefinite programming (SDP) relaxation [Yang et al., 2014, Wang et al., 2019]. They then resolve the source directions via subspace methods such as the root-MUSIC algorithm [Barabell, 1983, Rao and Hari, 1989]. In this dissertation, we estimate the matrix by evaluating a learned function at a sample covariance and show that this approach can significantly outperform conventional SDP-based methods such as the sparse and parametric approach (SPA) [Yang et al., 2014] for a wide range of signal-to-noise ratios (SNRs), snapshots, and source numbers. Unlike existing DNN-based methods that learn covariance matrices of a ULA via some distances such as the Frobenius norm [Barthelme and Utschick, 2021a, Wu et al., 2022], our approach trains a DNN to learn signal and noise subspace representations that are invariant to the selection of bases. Due to this invariance, our loss functions expand the solution space and give the DNN more freedom to learn the subspace structures. To learn such representations, we propose novel loss functions that gauge the separation between the desired subspace and the predicted subspace from the DNN. In particular, we propose losses that measure the shortest curve(s) between two subspaces viewed on a union of Grassmannians and prove that a ReLU network can approximate signal subspaces. Our approach is robust to array imperfections because it is geometry-agnostic.

Numerical results show that learning such subspace representations is more beneficial than learning covariance matrices for both perfect and imperfect arrays under standard assumptions.

Chapter 1

Improved Bounds on Neural Complexity for Representing Piecewise Linear Functions

A deep neural network using rectified linear units represents a continuous piecewise linear (CPWL) function and vice versa. Recent results in the literature estimated that the number of neurons needed to exactly represent any CPWL function grows exponentially with the number of pieces or exponentially in terms of the factorial of the number of distinct linear components. Moreover, such growth is amplified linearly with the input dimension. These existing results seem to indicate that the cost of representing a CPWL function is expensive. In this paper, we propose much tighter bounds and establish a polynomial time algorithm to find a network satisfying these bounds for any given CPWL function. We prove that the number of hidden neurons required to exactly represent any CPWL function is at most a quadratic function of the number of pieces. In contrast to all previous results, this upper bound is invariant to the input dimension. Besides the number of pieces, we also study the number of distinct linear components in CPWL functions. When such a number is also given, we prove that the quadratic complexity turns into bilinear, which implies a lower neural complexity because the number of distinct linear components is always not greater than the minimum number of pieces in a CPWL function. When the number of pieces is unknown, we prove that, in terms of the number of distinct linear components, the neural complexities of any CPWL function are at most polynomial growth for low-dimensional inputs and factorial growth for the worst-case scenario, which are significantly better than existing results in the literature.

1.1 Introduction

The rectified linear unit (ReLU) [Fukushima, 1980, Nair and Hinton, 2010] activation has been by far the most widely used nonlinearity and successful building block in deep neural networks (DNNs). Numerous architectures based on ReLU DNNs have achieved remarkable performance or state-of-the-art accuracy in speech processing [Zeiler et al., 2013, Maas et al., 2013], computer vision [Krizhevsky et al., 2012, Simonyan and Zisserman, 2015b, He et al., 2016a], medical image segmentation [Ronneberger et al., 2015], game playing [Mnih et al., 2015, Silver et al., 2016], and natural language processing [Vaswani et al., 2017], just to name a few. Besides such unprecedented empirical success, ReLU DNNs are also probably the most understandable nonlinear deep learning models due to their ability to be "un-rectified" [Hwang and Heinecke, 2019].

The ability to demystify ReLU DNNs via "un-rectifying ReLUs" dates back to a seminal work by Pascanu et al. in 2014. Because each of ReLUs in a hidden layer divides the space of the preceding layer's output into two half spaces whose ReLU response is affine in one half space and exactly zero in the other, the layer of ReLUs can be replaced by an input-dependent diagonal matrix whose diagonal elements are ones for firing ReLUs and zeros for non-firing ReLUs. Based on this rationale, Pascanu et al. [2014] proved that a neural network using ReLUs divides the input space into many linear regions such that the network itself is an affine function within every region. Two excellent visualizations are shown in Figure 2 in [Hanin and Rolnick, 2019a] and [Hanin and Rolnick, 2019b]. At this point, it is quite evident that any ReLU network exactly represents a CPWL function. Pascanu et al. also proved that the maximum number of linear regions for any ReLU network with a single hidden layer is equivalent to the number of connected components induced by arrangements of hyperplanes in general position where each hyperplane corresponds to a ReLU in the hidden layer. Such a number can be computed in a closed form by Zaslavsky's Theorem [Zaslavsky, 1975]. Furthermore, they showed that the maximum number of linear regions can be bounded from below by exponential growth in terms of the number of hidden layers, leading to a conclusion that ReLU DNNs can generate more linear regions than their shallow counterparts. In the same year, Montúfar et al. improved such a lower bound and gave the first upper bound for the maximum number of linear regions. These bounds and their assumptions were later improved by [Montúfar, 2017, Raghu et al., 2017, Arora et al., 2018, Serra et al., 2018, Hinz and van de Geer, 2019], just to name a few. We refer readers to Hinz's doctoral thesis for a thorough discussion on the upper bound of the number of linear regions. Because a CPWL function with more pieces can better approximate any given continuous function and a ReLU DNN exactly represents a CPWL function [Arora et al., 2018], a ReLU DNN with more linear regions in general exhibits stronger expressivity. In summary, this "un-rectifying" perspective provides us a new angle to understand ReLU DNNs, and the results in some ways align with advances in approximation theory demonstrating the expressivity.[1]

Despite these advancements in linear regions, the complexity of a ReLU DNN that exactly represents a given CPWL function remains largely unexplored. One can find that this question is the opposite direction of the above-mentioned line of research. Although Arora et al. [2018] proved that any CPWL function can be exactly represented by a ReLU DNN with a bounded depth, any estimates regarding the width or number of neurons of such a network were not given. The resources required for a ReLU neural network to exactly represent a CPWL function remained unknown until He et al. [2020] provided a bound to the complexity of a ReLU network that realizes any given CPWL function. They proved that the number of neurons is bounded from above by exponential growth in terms of the product between the number of pieces and the number of distinct linear components of a given CPWL function. Such an exponential bound also grows linearly with the input dimension. Because the number of pieces is an upper

[1]The approximation viewpoint is not the focus of this paper. The literature on approximation is vast and we refer readers to [Vardi et al., 2021, Lu et al., 2017, Eldan and Shamir, 2016, Telgarsky, 2016, Hornik et al., 1989, Cybenko, 1989], just to name a few.

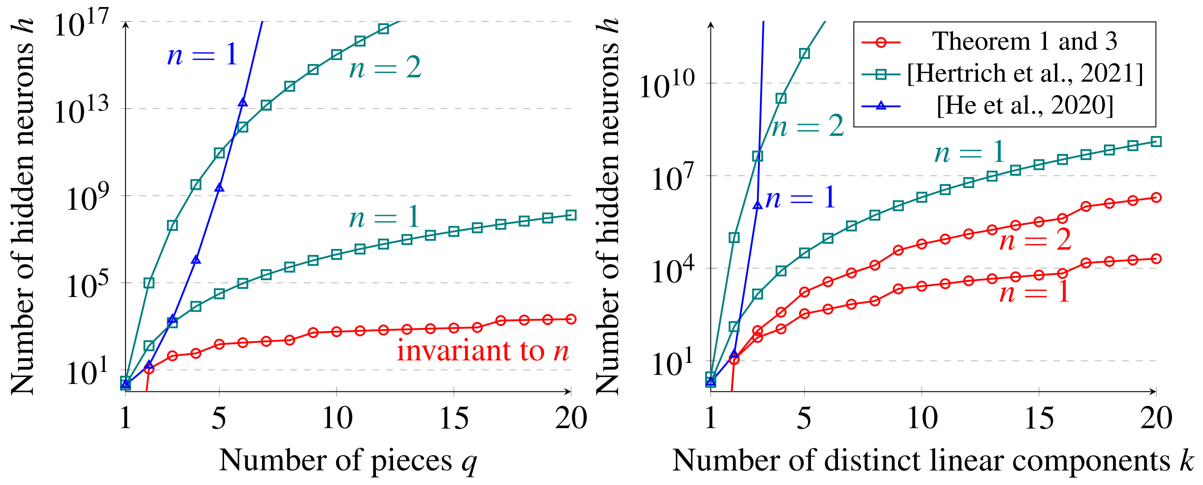


Figure 1.1. Any CPWL function $\mathbb{R}^n \to \mathbb{R}$ with q pieces or k distinct linear components can be exactly represented by a ReLU network with at most h hidden neurons. In Theorem 1 and 3, $h = 0$ when $q = 1$ or $k = 1$. The bounds in Theorem 1 and the worst-case bounds in Theorem 3 are invariant to n. (1.5) is used to infer h based on the depth and width given by Hertrich et al. [2021]. The upper bounds given by Theorem 1 and 3 are substantially tighter than existing bounds in the literature, implying that any CPWL function can be exactly realized by a ReLU network at a much lower cost.

bound of the number of distinct linear components for any CPWL function [Tarela and Martínez, 1999, He et al., 2020], the bound grows exponentially with the quadratic number of pieces, which seems to imply that the cost for representing a CPWL function by a ReLU DNN is exceedingly high.

The most recent upper bound can be inferred from a recent work by Hertrich et al. [2021] although the number of hidden neurons was not directly given. Hertrich et al. [2021] proved a width bound in terms of the number of distinct linear components under the same depth used by Arora et al. [2018] and He et al. [2020].[2] In particular, they proved that the maximum width of a ReLU network that represents any given CPWL function can be polynomially bounded from above in terms of the number of distinct linear components. However, the order of such a polynomial is a quadratic function of the input dimension, which can be immensely large for

[2]The number of "affine pieces" used by Theorem 4.4 in [Hertrich et al., 2021] should be interpreted as the number of distinct linear components to best reflect the upper bound for the maximum width. Such an interpretation of "affine pieces" is different from the convention used by Pascanu et al. [2014], Montúfar et al. [2014], Arora et al. [2018], Hanin and Rolnick [2019a], and this work.

a small number of pieces or linear components when the input dimension is large. This bound grows larger with the input dimension even though the underlying CPWL function is just a one-hidden-layer ReLU network using only one ReLU (see Figure 1.1 for the difference between $n = 1$ and $n = 2$ when $q = 2$ or $k = 2$).

In this paper, we provide improved bounds showing that any CPWL function can be represented by a ReLU DNN whose neural complexity is bounded from above by functions with much slower growth (see Figure 1.1). Our results imply that one can exactly realize any given CPWL function by a ReLU network at a much lower cost. On the other hand, in addition to guaranteeing the existence of such a network, we also give a *polynomial time* algorithm to exactly find a network satisfying our bounds. To the best of our knowledge, our results regarding the computational resource for a ReLU network, i.e., the number of hidden neurons, are the best upper bounds in the existing literature and the algorithm is the first tailored procedure to find a network representation from any given CPWL function. Key results and main contributions of this paper are highlighted below.

1.1.1 Key results and contributions

Quadratic bounds

We prove that any CPWL function with q pieces can be represented by a ReLU network whose number of hidden neurons is bounded from above by a quadratic function of q. We also give the corresponding upper bounds for the maximum width, i.e., the maximum number of neurons per hidden layer, and the number of layers for such a network. The maximum width is bounded from above by $\mathscr{O}(q^2)$ and the number of layers is bounded from above by a logarithmic function of q, i.e., $\mathscr{O}(\log_2 q)$. *These bounds are invariant to the input dimension.* For any affine function, the upper bounds for the maximum width and the number of hidden neurons are zero.

Further improvements on neural complexity

When the number of distinct linear components k of any CPWL function is given along with the number of pieces q, the quadratic bounds $\mathscr{O}(q^2)$ for the number of hidden neurons and the maximum width turn into *bilinear* bounds of k and q, i.e., $\mathscr{O}(kq)$. Such a change reduces the neural complexity because $k \leq q$, and q can be much larger than k. Still, these bounds are independent of the input dimension.

Finding a network satisfying bilinear bounds

We establish a polynomial time algorithm that finds a ReLU network representing any given CPWL function. The network found by the algorithm satisfies the bilinear bounds on the number of hidden neurons and the maximum width, and the logarithmic bound on the number of layers. Note that such an algorithm also guarantees that one can always reverse-engineer at least one ReLU network from the function it computes. Compared to the general-purpose reverse-engineering algorithm proposed by Rolnick and Kording [2020], our algorithm specializes in the situation when pieces of a CPWL function are given.

Improved bounds from a perspective of linear components

When the number of pieces of a CPWL function is unknown and only the number of linear components k is available, we prove that the number of hidden neurons and maximum width are bounded from above by factorial growth. More precisely, $\mathscr{O}(k \cdot k!)$. The number of layers is bounded from above by linearithmic growth, or $\mathscr{O}(k \log_2 k)$. However, when the input dimension n grows sufficiently slower than k, e.g., $\mathscr{O}\left(\sqrt{k}\right)$, then bounds for the number of hidden neurons and maximum width reduce to *polynomial* growth functions of order $2n+1$; and the linearithmic growth reduces to $\mathscr{O}\left(n \log_2 k\right)$ for the depth.

A new approach to choosing the depth

Instead of scaling the depth of a ReLU network with the input dimension [Arora et al., 2018, He et al., 2020, Hertrich et al., 2021], we reveal that constructing a ReLU network whose depth is scaled with the number of pieces of the given CPWL function is more advantageous. Such a scaling turns out to be the key to deriving better upper bounds. This insight is provided by the max-min representation of CPWL functions [Tarela et al., 1990]. The importance of this scaling on the depth in ReLU networks has not been well recognized by existing bounds in the literature. We discuss implications of different representations in Section 1.4.

1.2 Preliminaries

Notation and definitions used in this paper are set up and clarified in this section. The set $\{1,2,\cdots,m\}$ is denoted by $[m]$. $\mathbb{I}[condition]$ is an indicator function that gives 1 if the *condition* is true, and 0 otherwise. The CPWL function is defined by Definition 1 below.

Definition 1. *A function $p\colon \mathbb{R}^n \to \mathbb{R}$ is said to be CPWL if there exists a finite number of closed subsets of $\mathbb{R}^n$, say $\{\mathscr{U}_i\}_{i\in[m]}$, such that (a) $\mathbb{R}^n = \bigcup_{i\in[m]} \mathscr{U}_i$; (b) p is affine on $\mathscr{U}_i, \forall i \in [m]$.*

A family of closed *convex* subsets, say $\{\mathscr{X}_i\}_{i\in[q]}$, satisfying Definition 1 is also referred to as a family of convex regions, affine pieces or simply *pieces* for a CPWL function in this paper. Definition 1 follows the definition of CPWL functions by Ovchinnikov [2002]. Notice that there are different definitions in the literature. For example, Chua and Deng [1988] and Arora et al. [2018] defined a CPWL function on a finite number of polyhedral regions. However, their definitions are essentially the same as Definition 1 because any family of closed subsets satisfying Definition 1 can be decomposed into polyhedral regions. It is possible that some of the closed subsets satisfying Definition 1 are non-convex even though the number of them reaches the minimum (see Figure 2 in [Wang and Sun, 2005]). The continuity is implied by Definition 1 due to the subsets being closed.

Because the goal of this paper is to bound the complexity of a ReLU DNN that exactly represents any given CPWL function, it is necessary to be able to measure the complexity of a CPWL function. The complexity of a CPWL function can be described using two different perspectives. One is the number of pieces q, which is the number of closed convex subsets satisfying Definition 1. Because this number has a minimum and any finite number above the minimum can be a valid m in Definition 1, the bounds become obviously loose when the number of pieces is not the minimum. Without loss of generality, we are interested in the number q when it is the minimum. The other is the number of distinct linear components k. A linear component of a CPWL function is defined in Definition 2.

Definition 2. *An affine function f is said to be a linear component of a CPWL function p if there exists a nonempty subset $\mathscr{M} \subseteq [m]$ such that $f(\mathbf{x}) = p(\mathbf{x}), \forall \mathbf{x} \in \bigcup_{i \in \mathscr{M}} \mathscr{U}_i$ where $\{\mathscr{U}_i\}_{i \in [m]}$ is a family of the minimum number of closed subsets satisfying Definition 1.*

A greater q or k gives a CPWL function more degrees of freedom because a CPWL function allowed to use $q+1$ pieces or $k+1$ arbitrary linear components can represent any CPWL function with q pieces or k distinct linear components and still have the flexibility to modify existing affine maps or increase the number of distinct affine maps of the CPWL function. Although increasing them both leads to a CPWL function with greater flexibility, the speed of upgrading degrees of freedom is different from each other. Note that a CPWL function with q pieces can never have more than q distinct linear components and a CPWL function with k distinct linear components can easily have more than k minimum number of pieces. Such a difference in a 1-dimensional case can be clearly observed from Figure 1 in [Tarela and Martínez, 1999]. Note that it is possible for two disjoint subsets from a minimum number of closed subsets satisfying Definition 1 to share the same linear component. In other words, a linear component can be reused by multiple pieces. Hence, increasing k gives faster growth than increasing q for the complexity and expressivity of CPWL functions.

We define the ReLU activation function in Definition 3. The ReLU network defined in Definition 4 is a simple architecture which is usually referred to as a *ReLU multi-layer perceptron*. Definition 5 defines the corresponding number of hidden neurons, depth, and maximum width.

Definition 3. *The rectified linear unit (ReLU) activation function σ is defined as $\sigma(x) = \max(0, x)$. The ReLU layer or vector-valued rectified linear activation function σ_k is defined as*

$$\sigma_k(\mathbf{x}) = \begin{bmatrix} \sigma(x_1) & \sigma(x_2) & \cdots & \sigma(x_k) \end{bmatrix}^{\mathsf{T}} \tag{1.1}$$

where $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_k \end{bmatrix}^\mathsf{T}$.

Definition 4. *Let l be any positive integer. A function $g\colon \mathbb{R}^{k_0} \to \mathbb{R}^{k_l}$ is said to be an l-layer ReLU network if there exist weights $\mathbf{W}_i \in \mathbb{R}^{k_i \times k_{i-1}}$ and $\mathbf{b}_i \in \mathbb{R}^{k_i}$ for $i \in [l]$ such that the input-output relationship of the network satisfies $g(\mathbf{x}) = h_l(\mathbf{x})$ where $h_1(\mathbf{x}) = \mathbf{W}_1\mathbf{x} + \mathbf{b}_1$ and $h_i(\mathbf{x}) = \mathbf{W}_i\sigma_{k_{i-1}}\left(h_{i-1}(\mathbf{x})\right) + \mathbf{b}_i$ for every $i \in [l] \setminus [1]$.*

Definition 5. *The sum $\sum_{l=1}^{L-1} k_l$ and the maximum $\max_{l\in[L-1]} k_l$ for $L > 1$ are referred to as the number of hidden neurons and the maximum width of an L-layer ReLU network, respectively. Any 1-layer ReLU network is said to have 0 hidden neurons and a maximum width of 0. An l-layer ReLU network is said to have depth l and $l-1$ hidden layers.*

1.3 Upper bounds on neural complexity for representing CPWL functions

The correspondence between CPWL functions and ReLU networks was first clearly confirmed by Theorem 2.1 in [Arora et al., 2018] although a weaker version of the correspondence can be inferred from Proposition 4.1 in [Goodfellow et al., 2013]. Arora et al. [2018] proved that every ReLU network $\mathbb{R}^n \to \mathbb{R}$ exactly represents a CPWL function, and the converse is also true, i.e., every CPWL function can be exactly represented by a ReLU network. One of the key steps used by Arora et al. [2018] to construct a ReLU network from any given CPWL function relies on an important representation result by Wang and Sun [2005], stating that any CPWL function can be represented by a sum of a finite number of *max-η-affine* functions [Magnani and Boyd, 2009] whose signs may be flipped and η is bounded from above by $n+1$ where η is the number of affine functions in the *max-η-affine* function. The implication of using this representation is later discussed in Section 1.4.1 and its *max-η-affine* functions are given therein. The bound $\eta \leq n+1$ in the representation allowed Arora et al. [2018] to further prove that there exists a ReLU DNN with at most

$$\left\lceil \log_2(n+1) \right\rceil \tag{1.2}$$

hidden layers to exactly realize any given CPWL function. However, the computational resource required for a ReLU network to exactly represent any CPWL function had not been available in the literature until the work by He et al. [2020].

1.3.1 Upper bounds in prior work

He et al. [2020] proved that a CPWL function $\mathbb{R}^n \to \mathbb{R}$ with q pieces and k linear components can be represented by a ReLU network whose number of neurons is given by

$$\begin{cases} \mathscr{O}\left(n2^{kq+(n+1)(k-n-1)}\right), & \text{if } k \geq n+1, \\ \mathscr{O}\left(n2^{kq}\right), & \text{if } k < n+1. \end{cases} \tag{1.3}$$

The number of hidden layers in such a ReLU DNN is also bounded from above by $\lceil \log_2(n+1) \rceil$, which is the same as the bound derived by Arora et al. [2018]. One of their significant contributions in our view is that they utilize the number of pieces and linear components of a CPWL function to bound the complexity of the equivalent ReLU network. He et al. [2020] also proved the relationship

$$k \leq q \leq k! \tag{1.4}$$

for any CPWL function. Note that the bounds in (1.4) on the number of pieces q and linear components k were first mentioned by Tarela and Martínez [1999] who developed the lattice representation of CPWL functions. Asymptotically, the bounds in (1.3) for $k \geq n+1$ and $k < n+1$ are amplified linearly with the input dimension n for any fixed k. Due to (1.4), they can be further bounded from above by $\mathscr{O}\left(n2^{q^2+(n+1)(q-n-1)}\right)$ and $\mathscr{O}\left(n2^{q^2}\right)$ in terms of q and n. On the other hand, in terms of k and n, they can be further bounded from above by $\mathscr{O}\left(n2^{k\cdot k!+(n+1)(k-n-1)}\right)$ and $\mathscr{O}\left(n2^{k\cdot k!}\right)$. Because these bounds grow much faster than exponential growth, they seem to suggest that the cost of computing a CPWL function via a ReLU network is exceptionally high.

Hertrich et al. [2021] proved that any CPWL function $\mathbb{R}^n \to \mathbb{R}$ with k distinct linear components can be represented by a ReLU network whose maximum width is $\mathscr{O}\left(k^{2n^2+3n+1}\right)$

under the same number of hidden layers $\lceil \log_2(n+1) \rceil$. Hence, the number of hidden neurons must be bounded from above by

$$\mathscr{O}\left(k^{2n^2+3n+1}\log_2(n+1)\right). \tag{1.5}$$

Note that we infer this bound by taking the product of the depth and the maximum width. Using $k \leq q$, the bound in (1.5) can be expressed in terms of q, leading to $\mathscr{O}\left(q^{2n^2+3n+1}\log_2(n+1)\right)$. Such a bound can grow slower than $\mathscr{O}\left(n2^{q^2}\right)$, but it grows faster than $\mathscr{O}\left(n2^{q^2}\right)$ if the input dimension n grows sufficiently faster than the number of pieces q. Also, $\mathscr{O}\left(k^{2n^2+3n+1}\log_2(n+1)\right)$ grows faster than $\mathscr{O}\left(n2^{k \cdot k!}\right)$ when the input dimension n grows sufficiently faster than the number of distinct linear components k.

1.3.2 Improved upper bounds

We show that any CPWL function can be represented by a ReLU network whose number of hidden neurons is bounded by much slower growth functions. We state our main results in Theorem 1, 2 and 3, and focus on their impact in this subsection. Each one of them is tailored to a specific complexity measure of the CPWL function. Their proof sketches are deferred to Section 1.4.2. We first focus on the case when the number of linear components is unknown and the complexity of the CPWL is only measured by the number of pieces q.

Theorem 1. *Any CPWL function $p\colon \mathbb{R}^n \to \mathbb{R}$ with q pieces can be represented by a ReLU network whose number of layers l, maximum width w, and number of hidden neurons h satisfy*

$$l \leq 2\lceil \log_2 q \rceil + 1, \tag{1.6}$$

$$w \leq \mathbb{I}[q > 1]\left\lceil \frac{3q}{2} \right\rceil q, \tag{1.7}$$

and

$$h \leq \left(3 \cdot 2^{\lceil \log_2 q \rceil} + 2\lceil \log_2 q \rceil - 3\right) q + 3 \cdot 2^{\lceil \log_2 q \rceil} - 2\lceil \log_2 q \rceil - 3. \tag{1.8}$$

Furthermore, Algorithm 1 finds such a network in $\mathsf{poly}(n,q,L)$ *time where L is the number of bits required to represent every entry of the rational matrix* $\mathbf{A}_i$ *in the polyhedron representation* $\{\mathbf{x}\in\mathbb{R}^n|\mathbf{A}_i\mathbf{x}\le\mathbf{b}_i\}$ *of the piece* $\mathscr{X}_i$ *for every* $i\in[q]$.

Algorithm 1. Find a ReLU network that computes a given continuous piecewise linear function

```
Input: A CPWL function $p$ with pieces $\{\mathscr{X}_i\}_{i\in[q]}$ of $\mathbb{R}^n$ satisfying Definition 1.
Output: A ReLU network $g$ computing $g(\mathbf{x})=p(\mathbf{x}),\forall\mathbf{x}\in\mathbb{R}^n$.
1: $f_1,f_2,\cdots,f_k\leftarrow$ run Algorithm 6 to find all distinct linear components of $p$
2: for $i=1,2,\cdots,q$ do
3:     $\mathscr{A}_i\leftarrow\emptyset$
4:     for $j=1,2\cdots,k$ do
5:         if $f_j(\mathbf{x})\ge p(\mathbf{x}),\forall\mathbf{x}\in\mathscr{X}_i$ then
6:             $\mathscr{A}_i\leftarrow\mathscr{A}_i\bigcup\{j\}$
7:         end if
8:     end for
9:     $v_i\leftarrow$ run Algorithm 2 with $\{f_m\}_{m\in\mathscr{A}_i}$ using the minimum type
10: end for
11: $v\leftarrow$ run Algorithm 3 with $v_1,v_2,\cdots,v_q$        ▷ Combine $q$ ReLU networks in parallel
12: $u\leftarrow$ run Algorithm 2 with $\left\{\begin{bmatrix}s_1 & s_2 & \cdots & s_q\end{bmatrix}^{\mathsf{T}}\mapsto s_m\right\}_{m\in[q]}$ using the maximum type
13: $g\leftarrow$ run Algorithm 4 with $v$ and $u$        ▷ Find a ReLU network for the composition $u\circ v$
```

The proof of Theorem 1 is deferred to Appendix 1.6.2 in the supplementary material. Algorithm 6, 2, 3, and 4 used by Algorithm 1 are deferred to Appendix 1.6.3 in the supplementary material and will be discussed soon after the discussion on bounds. Because $2^{\lceil\log_2 q\rceil}<2q$, the upper bound in (1.8) can be further bounded from above by $6q^2+2\lceil\log_2 q\rceil q+3q-2\lceil\log_2 q\rceil-3$, leading to the asymptotic bound $h=\mathscr{O}(q^2)$. Obviously, $l=\mathscr{O}(\log_2 q)$ and $w=\mathscr{O}(q^2)$. Since the bound given by Theorem 5.2 in He et al. [2020] can be lower bounded by $\mathscr{O}\left(n2^{q^2}\right)$, it grows exponentially faster than our bound of h given in Theorem 1. On the other hand, the upper bound given by (1.5) is at least polynomially larger than our bound of h and the order of this polynomial grows quadratically with the input dimension n. Note that such a polynomial becomes an exponential function when the growth in n is not slower than q. Such differences are illustrated by the figure on the left-hand side of Figure 1.1. The bounds in Theorem 1 are independent of the input dimension n. Hence, one can realize any given CPWL function using a relatively small ReLU network even though n is huge.

In terms of the maximum width, the upper bound given by (1.7) is at least polynomially smaller than the one given by Hertrich et al. [2021]. In contrast to the bound for the number of layers in [Arora et al., 2018, He et al., 2020, Hertrich et al., 2021] that grows logarithmically with the input dimension n, our bound in Theorem 1 grows logarithmically with the number of pieces q. Therefore, the ReLU network found by Algorithm 1 in general becomes deeper when the CPWL becomes more complex for a fixed input dimension. On the other hand, the network remains the same depth even for an arbitrarily larger n as long as q is fixed. Taking an affine function for example, a 1-layer ReLU network with 0 hidden neurons is the solution given by Theorem 1. However, the bound given by [Arora et al., 2018, He et al., 2020, Hertrich et al., 2021] keeps increasing the depth for a larger n.

We briefly explain algorithms used by Algorithm 1. Algorithm 2 finds a ReLU network that computes a *max-affine* or *min-affine* function [Magnani and Boyd, 2009]. Algorithm 3 concatenates two given ReLU networks in parallel and returns another ReLU network computing the concatenation of two outputs. Algorithm 4 finds a ReLU network that represents a composition of two given ReLU networks. These algorithms are basic manipulations of ReLU networks. Algorithm 1 is a polynomial time algorithm, following from the proof of Theorem 2. Table 1.1 in Appendix 1.6.3 in the supplementary material gives a complexity analysis for Algorithm 1.

Notice that Algorithm 1 does not need to be given any linear components or completely know the CPWL function because every distinct linear component can be found by Algorithm 6, which only needs to be given a closed ε-ball in the interior of every piece of a CPWL function p and observe the output of p when feeding an input. Algorithm 6 solves a system of linear equations for every piece of p to find the corresponding linear component. Every system of linear equations here has a unique solution because the interior of each of the pieces is nonempty. The nonemptyness is guaranteed by Lemma 12(a) in Appendix 1.6.1 in the supplementary material.

The 5th step of Algorithm 1 can be executed by checking the optimization result of the following linear programming problem

$$\begin{aligned} \text{minimize} \quad & f_j(\mathbf{x}) - p(\mathbf{x}), \\ \text{subject to} \quad & \mathbf{x} \in \mathscr{X}_i. \end{aligned} \tag{1.9}$$

The condition in the 5th step can only be true when the optimal value is nonnegative. Because every piece of p is given to Algorithm 1, the piece $\mathscr{X}_i$ is available for the linear program as a system of linear inequalities. The objective function is also available since p is affine on $\mathscr{X}_i$ and all distinct linear components are available from Algorithm 6. The corresponding linear component of p on $\mathscr{X}_i$ can be found by first feeding at most $n+1$ affinely independent points from the closed ε-ball to p and every candidate linear component, and then matching their output values.

The ellipsoid method [Khachiyan, 1979], the interior-point method [Karmarkar, 1984], and the path-following method [Renegar, 1988] are polynomial time algorithms for the linear programming problem using rational numbers on the Turing machine model of computation. These algorithms are also known to be *weakly polynomial time* algorithms. The strongly polynomial time algorithm requested by Smale's 9th problem [Smale, 1998], i.e., *the linear programming problem*, is still an open question. Given that we run the 5th step of Algorithm 1 by solving the linear programming problem in (1.9), Algorithm 1 is a weakly polynomial time algorithm. The question of whether it is a strongly polynomial time algorithm is not known. The dependency on the number of bits L in the time complexity of Algorithm 1 directly comes from using (1.9) to execute the 5th step. In practice, linear programming problems can be solved very reliably and efficiently [Boyd and Vandenberghe, 2004]. We provide an implementation of

Algorithm 1 and measure its run time on a computer for different numbers of pieces and input dimensions in Appendix 1.6.4 in the supplementary material.

Theorem 2 discusses the case when the number of linear components and pieces are both known.

Theorem 2. *Any CPWL function $p\colon \mathbb{R}^n \to \mathbb{R}$ with k linear components and q pieces can be represented by a ReLU network whose number of layers l, maximum width w, and number of hidden neurons h satisfy $l \le \lceil \log_2 q \rceil + \lceil \log_2 k \rceil + 1$, $w \le \mathbb{I}[k > 1] \left\lceil \frac{3k}{2} \right\rceil q$, and*

$$h \le \left(3 \cdot 2^{\lceil \log_2 k \rceil} + 2 \lceil \log_2 k \rceil - 3\right) q + 3 \cdot 2^{\lceil \log_2 q \rceil} - 2 \lceil \log_2 k \rceil - 3. \tag{1.10}$$

Furthermore, Algorithm 1 finds such a network in $\mathsf{poly}(n,k,q,L)$ time where L is the number of bits required to represent every entry of the rational matrix $\mathbf{A}_i$ in the polyhedron representation $\{\mathbf{x} \in \mathbb{R}^n | \mathbf{A}_i \mathbf{x} \le \mathbf{b}_i\}$ of the piece $\mathscr{X}_i$ for every $i \in [q]$.

The proof of Theorem 2 is deferred to Appendix 1.6.2 in the supplementary material. The bounds in Theorem 2 are in general tighter and always no worse than those in Theorem 1 because q is never less than k but can be much larger than k. Asymptotically, $l = \mathscr{O}(\log_2 q)$, $w = \mathscr{O}(kq)$, and $h = \mathscr{O}(kq)$. The bound given by Theorem 5.2 in He et al. [2020] increases exponentially faster than the bound of h in Theorem 2.

When the number of linear components is the only complexity measure of the CPWL function, we resort to Theorem 3 below.

Theorem 3. *Any CPWL function $p\colon \mathbb{R}^n \to \mathbb{R}$ with k linear components can be represented by a ReLU network whose number of layers l, maximum width w, and number of hidden neurons h satisfy $l \le \lceil \log_2 \phi(n,k) \rceil + \lceil \log_2 k \rceil + 1$, $w \le \mathbb{I}[k > 1] \left\lceil \frac{3k}{2} \right\rceil \phi(n,k)$, and*

$$h \le \left(3 \cdot 2^{\lceil \log_2 k \rceil} + 2 \lceil \log_2 k \rceil - 3\right) \phi(n,k) + 3 \cdot 2^{\lceil \log_2 \phi(n,k) \rceil} - 2 \lceil \log_2 k \rceil - 3 \tag{1.11}$$

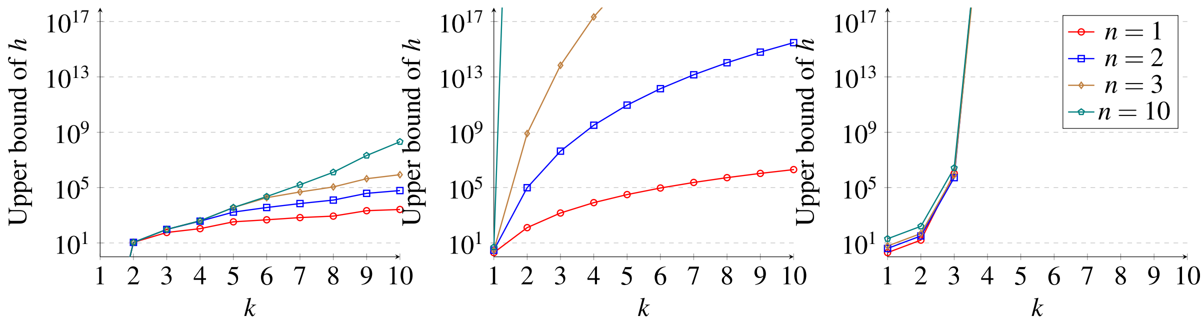


Figure 1.2. Left: The upper bound of h in Theorem 3 grows much slower when n grows sufficiently slower than k, leading to a much better upper bound compared to the worst-case asymptotic bound $\mathscr{O}(k \cdot k!)$ in Theorem 3 when n is sufficiently larger. Middle: the bound in (1.5) inferred from [Hertrich et al., 2021]. Right: Theorem 5.2 in [He et al., 2020].

where

$$\phi(n,k) = \min\left(\sum_{i=0}^{n}\binom{\frac{k^2-k}{2}}{i}, k!\right). \tag{1.12}$$

The proof of Theorem 3 is deferred to Appendix 1.6.2 in the supplementary material. Because $\phi(n,k) \le k!$, the worst-case asymptotic bounds for l, w and h are $l = \mathscr{O}(k\log_2 k)$, $w = \mathscr{O}(k \cdot k!)$, and $h = \mathscr{O}(k \cdot k!)$, respectively. However, it holds that $\sum_{i=0}^{n}\binom{\frac{k^2-k}{2}}{i} \le k^{2n}$, so the asymptotic bounds are $l = \mathscr{O}(n\log_2 k)$, $w = \mathscr{O}\left(k^{2n+1}\right)$, and $h = \mathscr{O}\left(k^{2n+1}\right)$ when n grows sufficiently slower than k. For example, $n = \mathscr{O}\left(\sqrt{k}\right)$. In this case, w and h are bounded from above by a polynomial of order $2c\sqrt{k}+1$ for some constant c, which grows slower than factorial growth. Such an advantage for small n is illustrated by the figure on the left-hand side of Figure 1.2.

Since the bound given by Theorem 5.2 in [He et al., 2020] can be bounded from below by $\mathscr{O}\left(n2^{k\cdot k!}\right)$, it at the minimum grows exponentially larger than the upper bound of h in Theorem 3. Even for a small n, the relative order of growth is gigantic. The figure on the right-hand side of Figure 1.2 illustrates such a large difference. For $k = 5$, $n2^{k\cdot k!} \approx 7.92 \times 10^{28}$ when $n = 1$, while our bound of h is at most 3615 for any n. The difference is extremely large even though k is small under $n = 1$. The middle plot in Figure 1.2 shows that (1.5) increases much faster when n becomes larger. For $k = 3$, $k^{2n^2+3n+1}\log_2(n+1) \approx 8.20 \times 10^{110}$ when $n = 10$, while our bound of h is at most 95 for any n. The upper bound of h in Theorem 3 is much better than (1.5) for any n and k.

Lemma 1. *Let $\mathscr{P}_{n,k}$ be the set of all CPWL functions with exactly k distinct linear components such that $p\colon \mathbb{R}^n \to \mathbb{R}, \forall p \in \mathscr{P}_{n,k}$. Let $\mathscr{C}_{n,k}(p)$ be the collection of all families of closed convex subsets satisfying Definition 1 for any $p \in \mathscr{P}_{n,k}$. Then, $k \leq \min_{\mathscr{Q} \in \mathscr{C}_{n,k}(p)} |\mathscr{Q}| \leq \phi(n,k)$.*

The proof of Lemma 1 is deferred to Appendix 1.6.2 in the supplementary material. Clearly, $\phi(n,k)$ is a better upper bound of q compared to the bound $q \leq k!$ given by He et al. [2020]. When n grows sufficiently slower than k, the bound $\phi(n,k)$ can be exponentially smaller than $k!$.

1.3.3 Limitations

Although these new bounds are significantly better than previous results, it is still possible to find a ReLU network whose hidden neurons are fewer than the bounds in Theorem 2 to exactly represent a given CPWL function. A tight bound for the case when $n = 1$ was first given by Theorem 2.2 in [Arora et al., 2018]. However, it seems more difficult to bound the size of a network from below for $n > 1$. To the best of our knowledge, we are not aware of any tight bounds in the literature for the size of the ReLU network representing a general CPWL function using an arbitrary input dimension.

1.4 Representations of CPWL functions have different implications on depth

We reveal implications of using different representations of CPWL functions and their impact on constructing ReLU networks. We first discuss the popular representation used by prior work and the implicit constraint imposed by such a representation.

1.4.1 Constrained depth

Arora et al. [2018], He et al. [2020], and Hertrich et al. [2021] proved the same bound for the number of layers, relying on the following representation of a CPWL function

$$p(\mathbf{x}) = \sum_{j=1}^{J} \sigma_j \max_{i \in \eta(j)} f_i(\mathbf{x}) \tag{1.13}$$

where $\sigma_j \in \{+1, -1\}$ and $\eta(j)$ is a subset of $[J]$ such that $\left|\eta(j)\right| \leq n+1$ for all $j \in [J]$. That is, a sum of a finite number of *max-η-affine* functions whose signs may be flipped. (1.13) was established by Theorem 1 in [Wang and Sun, 2005] which is essentially the same as Theorem 1 in [Wang, 2004] that emphasizes the difference between two convex piecewise linear functions. This result was also used by Goodfellow et al. [2013] to prove Proposition 4.1 in the maxout network paper.

The depth given by (1.2) does not scale with the complexity of a CPWL function. This feature directly comes from using a ReLU network to realize each of *max-η-affine* functions in (1.13) and concatenating all of them together. Because the size of $\eta(j)$ is bounded from above by $n+1$, the depth can be made to depend solely on n. Such a treatment seems to be the only way if one considers a CPWL function represented by (1.13). As a result, the ReLU network is forced to use a depth constrained by the input dimension to represent the given CPWL function, which in turn requires more hidden neurons. Because we do not use (1.13), our networks are not limited by such an implication.

1.4.2 Proof sketch for the unconstrained depth

We give a proof sketch in this subsection for our main results. By using a different representation, the depth of a ReLU network is able to be scaled with the complexity measure, i.e., the number of pieces, of any given CPWL function to accommodate the high expressivity.

By Theorem 4.2 in [Tarela and Martínez, 1999], any CPWL function p can be represented as

$$p(\mathbf{x}) = \max_{\mathscr{X} \in \mathscr{Q}} \min_{i \in \mathscr{A}(\mathscr{X})} f_i(\mathbf{x}) \tag{1.14}$$

for all $\mathbf{x} \in \mathbb{R}^n$ where $\mathscr{A}(\mathscr{X}) = \left\{ i \in [k] \mid f_i(\mathbf{x}) \geq p(\mathbf{x}), \forall \mathbf{x} \in \mathscr{X} \right\}$ is the set of indices of linear components that have values greater than or equal to $p(\mathbf{x})$ for all $\mathbf{x} \in \mathscr{X}$, and $\mathscr{Q}$ is any family of closed convex subsets of $\mathbb{R}^n$ satisfying Definition 1. We have used $f_1, f_2, \cdots, f_k$ to denote the k distinct linear components of p. Notice that Theorem 4.2 in [Tarela and Martínez, 1999] was first stated by Theorem 7 in [Tarela et al., 1990]. Both are essentially the same, but Theorem 4.2 in [Tarela and Martínez, 1999] emphasizes the convexity of each of the regions in the domain. Both theorems are also fundamentally equivalent to Theorem 2.1 in [Ovchinnikov, 2002]. Notice that one of the concluding remarks in [Ovchinnikov, 2002] pointed out that the convexity of the input space is an essential assumption. The entire space $\mathbb{R}^n$ satisfies such an assumption. In addition, Ovchinnikov pointed out that the max-min representation also holds for vector-valued CPWL functions. Hence, it is possible to generalize our bounds to vector-valued CPWL functions.

Using the representation in (1.14) and Lemma 2 below, we are able to prove Theorem 2 by bounding the size of $\mathscr{Q}$ and $\mathscr{A}(\mathscr{X})$. Theorem 1 and 3 can be proved by applying Lemma 1 to Theorem 2. Note that the size $|\mathscr{Q}|$ in (1.14) is the key for the depth of a ReLU network to be able to scale with q.

Lemma 2. *Let m be any positive integer. Define $l(m) = \lceil \log_2 m \rceil + 1$, $w(m) = \mathbb{I}[m > 1] \left\lceil \frac{3m}{2} \right\rceil$, and the following sequence for any positive integer k,*

$$r(k) = \begin{cases} 0, & \text{if } k = 1, \\ \frac{3k}{2} + r\left(\frac{k}{2}\right), & \text{if } k \text{ is even}, \\ 2 + \frac{3(k-1)}{2} + r\left(\frac{k+1}{2}\right), & \text{if } k \neq 1 \text{ and } k \text{ is odd}. \end{cases} \tag{1.15}$$

Then, there exists an $l(m)$-layer ReLU network $g\colon \mathbb{R}^n \to \mathbb{R}$ with $r(m)$ hidden neurons and a maximum width of $w(m)$ such that g computes the extremum of $f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_m(\mathbf{x})$, i.e.,

$g(\mathbf{x}) = \max_{i\in[m]} f_i(\mathbf{x})$ or $g(\mathbf{x}) = \min_{i\in[m]} f_i(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^n$ under any m scalar-valued affine functions $f_1, f_2, \cdots, f_m$. Furthermore, Algorithm 2 finds such a network in $\mathsf{poly}(m,n)$ time.

The proof of Lemma 2 is deferred to Appendix 1.6.2 in the supplementary material. One can also view $l(m)$, $w(m)$, and $r(m)$ as upper bounds for the number of layers, maximum width, and the number of hidden neurons. Because $r(m) < 6m-3$ by Lemma 6, the bound for the number of hidden neurons $r(m)$ is tighter than the bound $8m-4$ given by Lemma D.3 in [Arora et al., 2018] or Lemma 5.4 in [He et al., 2020] (these two lemmas are essentially the same). The bound for the number of layers remains the same as the one given by Lemma D.3 in [Arora et al., 2018]. By combining Lemma 2 with Lemma 3, Lemma 4, and Lemma 8, we can easily perform the same job on computing the extremum of multiple scalar-valued ReLU networks as Lemma D.3 does in [Arora et al., 2018]. Lemma 6, 3, 4, and 8 are given in Appendix 1.6.1 in the supplementary material.

1.5 Broader impact

Our results guarantee that any CPWL function can be exactly computed by a ReLU neural network at a more manageable cost. This assurance is crucial because CPWL functions are important tools in many applications. Such an assurance also relates DNNs closer to CPWL functions and allows researchers and engineers to understand the expressivity of DNNs from a different perspective. We focus on simple ReLU networks (ReLU multi-layer perceptrons) in this paper, but it may be possible to derive bounds for other activation functions and advanced neural network architectures such as maxout networks [Goodfellow et al., 2013], residual networks [He et al., 2016a], densely connected networks [Huang et al., 2017], and other nonlinear networks [Chen et al., 2021], by making some (possibly mild) assumptions. Our contributions advance the fundamental understanding of the link between ReLU networks and CPWL functions.

1.6 Appendix

1.6.1 Lemmas

Lemma 3. *Let l be any positive integer. There exists an l-layer ReLU network g with $2n(l-1)$ hidden neurons and a maximum width of $2n$ such that $g(\mathbf{x}) = \mathbf{x}$ for all $\mathbf{x} \in \mathbb{R}^n$. Furthermore, Algorithm 5 finds such a network in $\mathsf{poly}(n,l)$ time.*

Proof. Appendix 1.6.2. □

Definition 6. *Let $g_{(l,n,w)}$ denote an l-layer ReLU network with n hidden neurons and a maximum width bounded from above by w.*

Lemma 4. *There exists $g_{\left(l_1+l_2-1,n_1+n_2,\max(w_1,w_2)\right)}$ that represents any composition of $g_{(l_1,n_1,w_1)}$ and $g_{(l_2,n_2,w_2)}$. Algorithm 4 finds such a network computing the composition in $\mathsf{poly}\left(w_{max}, l_{max}\right)$ time where $w_{max} = \max(w_1,w_2)$ and $l_{max} = \max(l_1,l_2)$.*

Proof. Appendix 1.6.2. □

Lemma 5. *The sequence $r(k)$ defined by (1.15) is a strictly increasing sequence.*

Proof. Appendix 1.6.2. □

Lemma 6. *For any positive integer k, the sequence $r(k)$ defined by (1.15) satisfies*

$$r(k) \le 3\left(2^{\lceil \log_2 k \rceil} - 1\right) < 6k - 3. \tag{1.16}$$

Proof. Appendix 1.6.2 □

Lemma 7. *Let m_1 and m_2 be the output dimensions of $g_{(l_1,n_1,w_1)}$ and $g_{(l_2,n_2,w_2)}$, respectively. Define*

$$l = \max(l_1,l_2), \tag{1.17}$$

$$w = w_j + \max(w_i, 2m_i), \tag{1.18}$$

and

$$n = n_1 + n_2 + 2m_i|l_1 - l_2|, \tag{1.19}$$

where $i = \arg\min_{k\in[2]} l_k$ *and* $j = [2] \setminus \{i\}$. *Then, there exists* $g_{(l,n,w)}$ *such that*

$$g_{(l,n,w)}(\mathbf{x}) = \begin{bmatrix} g_{(l_1,n_1,w_1)}(\mathbf{x}) \\ g_{(l_2,n_2,w_2)}(\mathbf{x}) \end{bmatrix} \tag{1.20}$$

for all $\mathbf{x} \in \mathbb{R}^n$.

Proof. Appendix 1.6.2. □

Lemma 8. *Let* $m_1, m_2, \cdots, m_k$ *be the output dimensions of* $g_{(l_1,n_1,w_1)}, g_{(l_2,n_2,w_2)}, \cdots, g_{(l_k,n_k,w_k)}$, *respectively. Define*

$$l = \max_{i\in[k]} l_i, \tag{1.21}$$

$$w = \sum_{i\in[k]} \max(w_i, 2m_i), \tag{1.22}$$

and

$$n = \sum_{i\in[k]} n_i + 2m_i(l - l_i). \tag{1.23}$$

Then, there exists $g_{(l,n,w)}$ *such that*

$$g_{(l,n,w)}(\mathbf{x}) = \begin{bmatrix} g_{(l_1,n_1,w_1)}(\mathbf{x}) \\ g_{(l_2,n_2,w_2)}(\mathbf{x}) \\ \vdots \\ g_{(l_k,n_k,w_k)}(\mathbf{x}) \end{bmatrix} \tag{1.24}$$

for all $\mathbf{x} \in \mathbb{R}^n$. *Furthermore, Algorithm 3 finds such a network in* $\mathsf{poly}\left(\max_{i\in[k]} w_i, k, l\right)$ *time.*

Proof. Appendix 1.6.2. □

Lemma 9. *Let $f_1, f_2, \cdots, f_k$ be any affine functions such that $f_i \colon \mathbb{R}^n \to \mathbb{R}$ for all $i \in [k]$. Define the set of feasible ascending orders as*

$$\mathscr{S}^n_{f_1,f_2,\cdots,f_k} = \left\{(s_1, s_2, \cdots, s_k) \in \mathfrak{S}(k) \,|\, f_{s_1}(\mathbf{x}) \leq f_{s_2}(\mathbf{x}) \leq \cdots \leq f_{s_k}(\mathbf{x}), \mathbf{x} \in \mathbb{R}^n\right\} \tag{1.25}$$

where $\mathfrak{S}(k)$ is the collection of all permutations of the set $[k]$. It holds true that

$$\left|\mathscr{S}^n_{f_1,f_2,\cdots,f_k}\right| \leq \min\left(\sum_{i=0}^{n} \binom{\frac{k^2-k}{2}}{i}, k!\right). \tag{1.26}$$

Proof. Appendix 1.6.2. ☐

Lemma 10. *If Definition 1 is satisfied for a non-affine function, then every nonempty subset has a nonempty intersection with the other subset or at least one of the other subsets.*

Proof. Appendix 1.6.2. ☐

Assumption 1. *The number of closed connected subsets satisfying Definition 1 is a minimum.*

The interior and frontier (boundary) of a set $\mathscr{X}$ are denoted as $\mathrm{Int}\mathscr{X}$ and $\mathrm{Fr}\mathscr{X}$, respectively.

Lemma 11. *Let f_i denote the affine function associated with $\mathscr{X}_i$ for $i \in [I]$ where $\{\mathscr{X}_i\}_{i\in[I]}$ is a family of closed connected subsets satisfying Assumption 1. Then, for any $i \in [I], j \in [I]$ such that $i \neq j$ and $\mathscr{X}_i \bigcap \mathscr{X}_j \neq \emptyset$,*

(a) f_i *and* f_j *are different, and* $\{\mathbf{x} \in \mathbb{R}^n | f_i(\mathbf{x}) = f_j(\mathbf{x})\} \neq \emptyset$.

(b) $\{\mathbf{x} \in \mathbb{R}^n | f_i(\mathbf{x}) = f_j(\mathbf{x})\}$ *is an affine subspace of* $\mathbb{R}^n$ *with dimension* $n-1$.

(c) $\mathscr{X}_i \bigcap \mathscr{X}_j \subseteq \{\mathbf{x} \in \mathbb{R}^n | f_i(\mathbf{x}) = f_j(\mathbf{x})\}$.

(d) $\mathbf{x} \notin \mathrm{Int}\mathscr{X}_i$ *and* $\mathbf{x} \notin \mathrm{Int}\mathscr{X}_j$ *for all* $\mathbf{x} \in \mathscr{X}_i \bigcap \mathscr{X}_j$.

Proof. Appendix 1.6.2, 1.6.2, 1.6.2, and 1.6.2. ☐

Lemma 12. *If a family of closed connected subsets $\{\mathscr{X}_i\}_{i\in[I]}$ satisfies Assumption 1, then, for all $i \in [I]$,*

(a) $\mathrm{Int}\mathscr{X}_i \neq \emptyset$.

(b) $Fr\mathscr{X}_i = \bigcup_{k\in[I]\setminus i} \mathscr{X}_k \bigcap \mathscr{X}_i$.

(c) $\mathrm{Int}\mathscr{X}_i \bigcap \mathrm{Int}\mathscr{X}_j = \emptyset$ *for all* $j \in [I]$ *such that* $j \neq i$.

Proof. Appendix 1.6.2, 1.6.2, and 1.6.2. □

Lemma 13. *Let $\{\mathscr{X}_i\}_{i\in[m]}$ be any finite family of subsets satisfying Assumption 1. Let $\{\mathscr{H}_j\}_{j\in[k]}$ be any finite family of affine subspaces of $\mathbb{R}^n$ with dimension $n-1$. Then, for every $i \in [m]$,*

$$\mathscr{X}_i \bigcap \left(\mathbb{R}^n \setminus \bigcup_{j\in[k]} \mathscr{H}_j \right) \neq \emptyset. \tag{1.27}$$

Proof. Appendix 1.6.2. □

Proposition 1. *For any family of closed connected subsets satisfying Definition 1, all subsets are the largest closed connected subsets if and only if Assumption 1 is satisfied.*

Proof. Appendix 1.6.2. □

1.6.2 Proofs

Proof of Lemma 1

Proof. Let the family of closed connected subsets $\bar{\mathscr{Q}} = \{\mathscr{X}_i\}_{i\in[I]}$ satisfy Assumption 1 for any $p \in \mathscr{P}_{n,k}$. Let the k distinct linear components of p be $f_1, f_2, \cdots, f_k$ and $\mathscr{H}_{lm}$ be the intersection between f_l and f_m for $l \in [k], m \in [k], l \neq m$. Note that every $\mathscr{H}_{lm}$ is an affine subspace of $\mathbb{R}^n$ with dimension $n-1$ (a hyperplane) or an empty set. Because the linear components are distinct, it must be true that $k \leq I$ by Definition 2. If p is an affine function, then it follows that $k = \min_{\bar{\mathscr{Q}}\in\mathscr{C}_{n,k}(p)} |\bar{\mathscr{Q}}| = \phi(n,k) = 1$, the claim holds. For the non-affine case, we must have $k > 1$.

Let $\mathscr{R} = \mathbb{R}^n \setminus \mathscr{H}$ where

$$\mathscr{H} = \bigcup_{k\in[m],l\in[m],k\neq l} \mathscr{H}_{kl}. \tag{1.28}$$

Note that $\mathscr{H} \neq \emptyset$ according to Lemma 10 and 11(c). By Lemma 12(b), the boundary or frontier of $\mathscr{X}_i$ for $i \in [I]$ is given by

$$\mathrm{Fr}\mathscr{X}_i = \bigcup_{j\in[I]\setminus i} \left(\mathscr{X}_i \bigcap \mathscr{X}_j\right). \tag{1.29}$$

Because every $\mathscr{X}_i \bigcap \mathscr{X}_j$ for $i \in [I], j \in [I], i \neq j$ is a subset of some $\mathscr{H}_{lm}$ for $l \in [k], m \in [k], l \neq m$ by Lemma 11(c), it follows that the boundary of $\mathscr{X}_i$, $\mathrm{Fr}\mathscr{X}_i$, satisfies

$$\mathrm{Fr}\mathscr{X}_i \subseteq \mathscr{H} \tag{1.30}$$

for $i \in [I]$. The interior of $\mathscr{X}_i$, $\mathrm{Int}\mathscr{X}_i$, is a nonempty subset of $\mathbb{R}^n$ according to Lemma 12(a). Furthermore, by Lemma 13,

$$\mathscr{X}_i \bigcap \mathscr{R} \neq \emptyset. \tag{1.31}$$

Now, define

$$\mathscr{Z}_i = (\mathrm{Int}\mathscr{X}_i) \bigcap \mathscr{R} \tag{1.32}$$

for $i \in [I]$. Note that $\mathscr{Z}_i = \mathscr{X}_i \bigcap \mathscr{R} \neq \emptyset$ due to (1.30) and (1.31). Let $\mathscr{A}$ be any subset of $\mathbb{R}^n$ and $\lambda(\mathscr{A})$ be the number of connected components of $\mathscr{A}$ in $\mathbb{R}^n$. It must be true that

$$1 = \lambda(\mathscr{X}_i) \leq \lambda\left(\mathrm{Int}\mathscr{X}_i\right) \leq \lambda(\mathscr{Z}_i). \tag{1.33}$$

By Lemma 12(c), $\mathrm{Int}\mathscr{X}_i \bigcap \mathrm{Int}\mathscr{X}_j = \emptyset$ for $i \in [I], j \in [I], i \neq j$. We have

$$I \leq \lambda\left(\bigcup_{i\in[I]} \mathrm{Int}\mathscr{X}_i\right) = \sum_{i\in[I]} \lambda(\mathrm{Int}\mathscr{X}_i) \leq \sum_{i\in[I]} \lambda(\mathscr{Z}_i) = \lambda\left(\bigcup_{i\in[I]} \mathscr{Z}_i\right) = \lambda\left(\bigcup_{i\in[I]} \mathscr{X}_i \bigcap \mathscr{R}\right). \tag{1.34}$$

Notice that

$$\bigcup_{i\in[I]} \mathscr{X}_i \bigcap \mathscr{R} = \mathbb{R}^n \bigcap \mathscr{R} = \mathscr{R} \tag{1.35}$$

by the property $\bigcup_{i\in[I]} \mathscr{X}_i = \mathbb{R}^n$ in Definition 1. Plugging (1.35) into (1.34) leads to

$$I \leq \lambda(\mathscr{R}) \tag{1.36}$$

which states that I is bounded from above by the number of connected components of $\mathscr{R}$ in $\mathbb{R}^n$. Notice that every component is an open convex set because every component is the intersection of a finite number of open half spaces. Therefore,

$$I = \left|\bar{\mathscr{Q}}\right| = \min_{\mathscr{Q}'\in\mathscr{C}'_{n,k}(p)} \left|\mathscr{Q}'\right| \leq \min_{\mathscr{Q}\in\mathscr{C}_{n,k}(p)} \left|\mathscr{Q}\right| \leq \lambda(\mathscr{R}) \tag{1.37}$$

where $\mathscr{C}'_{n,k}(p)$ denotes the collection of all families of closed connected subsets satisfying Definition 1 for any $p \in \mathscr{P}_{n,k}$. Because the ascending order of these k linear components does not change within a connected component of $\mathscr{R}$, $\lambda(\mathscr{R})$ can be bounded from above by the number of feasible ascending orders. Let $\mathfrak{S}(k)$ be the collection of all permutations of the set $[k]$. It follows that

$$\lambda(\mathscr{R}) \leq \left|\left\{(s_1, s_2, \cdots, s_k) \in \mathfrak{S}(k) \,\middle|\, f_{s_1}(\mathbf{x}) \leq f_{s_2}(\mathbf{x}) \leq \cdots \leq f_{s_k}(\mathbf{x}), \mathbf{x} \in \mathbb{R}^n\right\}\right|. \tag{1.38}$$

Finally, Lemma 9 proves the statement by bounding the number of feasible ascending orders. □

Proof of Lemma 2

Proof. It suffices to show that

$$g(\mathbf{x}) = \max_{i\in[k]} x_i. \tag{1.39}$$

for all $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_k \end{bmatrix}^\mathsf{T} \in \mathbb{R}^k$ since the composition of affine functions is still affine. The affine functions can be absorbed into the first layer of the ReLU network g. We prove the case for taking the maximum of m real numbers since the same procedure below can be applied to prove the case of taking the minimum due to the following identity

$$\min_{i\in[k]} f_i(\mathbf{x}) = -\max_{i\in[k]} -f_i(\mathbf{x}). \tag{1.40}$$

Because $\max(x_1,x_2) = \max(0,x_2-x_1)+\max(0,x_1)-\max(0,-x_1)$ for any $x_1 \in \mathbb{R}$ and $x_2 \in \mathbb{R}$, it holds true that

$$\max_{j\in[k]} x_j = \begin{cases} \max_{j\in\left[\frac{k}{2}\right]} \max_{i\in\{2j-1,2j\}} x_i, & \text{if } k \text{ is even} \\ \max_{j\in\left[\frac{k+1}{2}\right]} \alpha(j;x_1,x_2,\cdots,x_k), & \text{if } k \text{ is odd} \end{cases} \tag{1.41}$$

for $x_j \in \mathbb{R}, j \in [k]$ where

$$\alpha(j;x_1,x_2,\cdots,x_k) = \begin{cases} \max_{i\in\{2j-1,2j\}} x_i, & \text{if } j \in \left[\frac{k-1}{2}\right] \\ \max(0,x_k)-\max(0,-x_k), & \text{if } j = \frac{k+1}{2} \end{cases}. \tag{1.42}$$

Let $r(k)$ be the number of operations of taking the maximum between a zero and a real number, i.e., $\max(0,x), x \in \mathbb{R}$ for computing the maximum of k real numbers using (1.41). One can find $r(2) = 3$ and $r(3) = 8$ by expanding all operations in (1.41). Because we do not need any maximum operations to compute the maximum over a singleton, we define $r(1) = 0$. For any positive integer k such that $k \geq 2$, we have the recursion

$$r(k) = \begin{cases} \frac{3k}{2} + r\left(\frac{k}{2}\right), & \text{if } k \text{ is even} \\ 2 + \frac{3(k-1)}{2} + r\left(\frac{k+1}{2}\right), & \text{if } k \text{ is odd} \end{cases} \tag{1.43}$$

according to (1.41). Note that $r(n)$ is the number of ReLUs in a ReLU network g that computes the maximum of n real numbers or a *max-affine* function. The number of ReLUs here is equivalent to the number of hidden neurons according to Definition 4. We shall note that the number of ReLU layers is equivalent to the number of hidden layers.

Obviously, we only need a 1-layer ReLU network with no ReLUs to compute the maximum of a singleton. Suppose that we aim to compute the maximum of $m = 2^n$ real numbers for any positive integer n. Then, every time the recursion goes to the next level in (1.43), the number of variables considered for computing the maximum is halved. Hence, the number of ReLU layers is n. When m is not a power of two, i.e., $2^n < m < 2^{n+1}$, then we can always construct a ReLU network with $n+1$ ReLU layers and 2^{n+1} input neurons, and set weights connected to the $2^{n+1} - m$ "phantom input neurons" to zeros. Because $\lceil \log_2 m \rceil = n+1$ for $2^n < m < 2^{n+1}$, the number of ReLU layers is $\lceil \log_2 m \rceil$ for any positive integer m. By Definition 5, we have $l(m) = \lceil \log_2 m \rceil + 1$.

By Lemma 5, $r(k)$ is a strictly increasing sequence. Therefore, the maximum width of the network is given by the width of the first hidden layer. When $L = 1$ or $m = 1$, the width is 0 due to Definition 5. When $L > 1$ or $m > 1$,

$$\begin{aligned}\max_{l \in [L-1]} k_l &= \begin{cases} \frac{3m}{2}, & \text{if } m \text{ is even} \\ 2 + \frac{3(m-1)}{2}, & \text{if } m \text{ is odd} \end{cases} \\ &= \left\lceil \frac{3m}{2} \right\rceil. \end{aligned} \tag{1.44}$$

Algorithm 2 directly follows from the above construction. Its complexity analysis is deferred to Table 1.2 in Appendix 1.6.3. □

Proof of Theorem 2

Proof. Let $f_1, f_2, \cdots, f_k$ be k distinct linear components of p and $\mathscr{Q}$ be any family of closed convex subsets of $\mathbb{R}^n$ satisfying Definition 1. By Theorem 4.2 in [Tarela and Martínez, 1999], p can be represented as

$$p(\mathbf{x}) = \max_{\mathscr{X} \in \mathscr{Q}} \min_{i \in \mathscr{A}(\mathscr{X})} f_i(\mathbf{x}) \tag{1.45}$$

for all $\mathbf{x} \in \mathbb{R}^n$ where

$$\mathscr{A}(\mathscr{X}) = \left\{ i \in [k] \mid f_i(\mathbf{x}) \geq p(\mathbf{x}), \forall \mathbf{x} \in \mathscr{X} \right\} \tag{1.46}$$

is the set of indices of linear components that have values greater than or equal to $p(\mathbf{x})$ for all $\mathbf{x} \in \mathscr{X}$. A thorough discussion of the representation (1.45) is given in Section 1.4.2.

According to (1.45), there are $|\mathscr{Q}|$ minima required to be computed where each of them is a minimum of $|\mathscr{A}(\mathscr{X})|$ real numbers. Then, the value of p can be computed by taking the maximum of the resulting $|\mathscr{Q}|$ minima. We will show that these operations are realizable by a ReLU network. By Lemma 2, an $l(m)$-layer ReLU network with $r(m)$ hidden neurons and a maximum width of $w(m)$ can compute the extremum of m real numbers given by m affine functions.

We realize (1.45) in three steps. First, we create $|\mathscr{Q}|$ ReLU networks where each of them is an $l\left(|\mathscr{A}(\mathscr{X})|\right)$-layer ReLU network with $r\left(|\mathscr{A}(\mathscr{X})|\right)$ hidden neurons and a maximum width of $w\left(|\mathscr{A}(\mathscr{X})|\right)$ that computes $\min_{i \in \mathscr{A}(\mathscr{X})} f_i(\mathbf{x})$ for $\mathscr{X} \in \mathscr{Q}$. Second, we parallelly concatenate these $|\mathscr{Q}|$ networks, i.e., put them in parallel and let them share the same input to obtain a ReLU network that takes $\mathbf{x}$ and outputs $|\mathscr{Q}|$ real numbers. Finally, we create an $l(|\mathscr{Q}|)$-layer ReLU network with $r(|\mathscr{Q}|)$ hidden neurons and a maximum width of $w(|\mathscr{Q}|)$ that takes the maximum of $|\mathscr{Q}|$ real numbers.

The parallel combination of $|\mathscr{Q}|$ networks in the second step can be realized by Lemma 8. The third step can be fulfilled by Lemma 4. With the above construction, we can now count the number of layers, the upper bound for the maximum width, and the number of hidden neurons for a ReLU network that realizes p. The number of layers is given by

$$l(|\mathscr{Q}|) + \max_{\mathscr{X} \in \mathscr{Q}} l\left(|\mathscr{A}(\mathscr{X})|\right) - 1. \tag{1.47}$$

The maximum width is bounded from above by

$$\max\left(\sum_{\mathscr{X}\in\mathscr{Q}}\max\left(w\left(\left|\mathscr{A}\left(\mathscr{X}\right)\right|\right),2\right),w\left(|\mathscr{Q}|\right)\right). \tag{1.48}$$

The number of hidden neurons is given by

$$r\left(|\mathscr{Q}|\right)+\sum_{\mathscr{X}\in\mathscr{Q}}r\left(\left|\mathscr{A}\left(\mathscr{X}\right)\right|\right)+2\left(\max_{\mathscr{Y}\in\mathscr{Q}}l\left(\left|\mathscr{A}\left(\mathscr{Y}\right)\right|\right)-l\left(\left|\mathscr{A}\left(\mathscr{X}\right)\right|\right)\right). \tag{1.49}$$

Because $\mathscr{A}\left(\mathscr{X}\right)$ for every $\mathscr{X}\in\mathscr{Q}$ is a subset of $[k]$, it holds that

$$1\leq\left|\mathscr{A}\left(\mathscr{X}\right)\right|\leq k \tag{1.50}$$

for all $\mathscr{X}\in\mathscr{Q}$. Therefore, the number of layers in (1.47) can be bounded from above by

$$l\left(|\mathscr{Q}|\right)+l\left(k\right)-1=\left\lceil\log_2|\mathscr{Q}|\right\rceil+\left\lceil\log_2 k\right\rceil+1 \tag{1.51}$$

where we have used the definition of the function l in Lemma 2. Again, using (1.50), the upper bound for the maximum width in (1.48) can be further bounded from above by

$$\begin{aligned}\max\left(\sum_{\mathscr{X}\in\mathscr{Q}}\max\left(w\left(k\right),2\right),w\left(|\mathscr{Q}|\right)\right)&=\max\left(|\mathscr{Q}|\max\left(w\left(k\right),2\right),w\left(|\mathscr{Q}|\right)\right)\\&\leq\max\left(|\mathscr{Q}|\max\left(\left\lceil\frac{3k}{2}\right\rceil,2\right),\left\lceil\frac{3|\mathscr{Q}|}{2}\right\rceil\right)\\&=\left\lceil\frac{3k}{2}\right\rceil|\mathscr{Q}|\end{aligned} \tag{1.52}$$

where we have used the definition of the function w in Lemma 2. Note that the maximum width is zero when the number of layers is one. Finally, again, using (1.50), the number of neurons in

(1.49) can be bounded from above by

$$
\begin{aligned}
& r(|\mathscr{Q}|) - 2l(k) + 2l(1) + \sum_{\mathscr{X} \in \mathscr{Q}} \big(r(k) + 2l(k) - 2l(1) \big) \\
&= r(|\mathscr{Q}|) - 2l(k) + 2l(1) + |\mathscr{Q}| \big(r(k) + 2l(k) - 2l(1) \big) \\
&= r(|\mathscr{Q}|) - 2 \lceil \log_2 k \rceil + |\mathscr{Q}| \Big(r(k) + 2 \lceil \log_2 k \rceil \Big) \\
&\leq 3 \Big(2^{\lceil \log_2 |\mathscr{Q}| \rceil} - 1 \Big) - 2 \lceil \log_2 k \rceil + |\mathscr{Q}| \Big(3 \Big(2^{\lceil \log_2 k \rceil} - 1 \Big) + 2 \lceil \log_2 k \rceil \Big) \\
&= 3 \Big(2^{\lceil \log_2 |\mathscr{Q}| \rceil} - 1 \Big) + 3|\mathscr{Q}| \Big(2^{\lceil \log_2 k \rceil} - 1 \Big) + 2 (|\mathscr{Q}| - 1) \lceil \log_2 k \rceil
\end{aligned}
\tag{1.53}
$$

where we have used Lemma 6 for the upper bound in the fourth line of (1.53). Expanding and rearranging terms in (1.53) lead to (1.10).

Algorithm 1 directly follows from the above construction. Its complexity analysis is deferred to Table 1.1 in Appendix 1.6.3. □

Proof of Theorem 1

Proof. By Lemma 1, the number of distinct linear components k is bounded from above by the number of pieces, i.e., $k \leq q$, implying that the bounds in Theorem 2 can be written in terms of q. Substituting k with q in Theorem 2 proves the claim.

According to Theorem 2, the time complexity of Algorithm 1 is $\mathsf{poly}(n,k,q,L)$. Using the bound $k \leq q$ proves the claim for the time complexity. □

Proof of Theorem 3

Proof. By Lemma 1, the minimum number of closed convex subsets q of a CPWL function $p\colon \mathbb{R}^n \to \mathbb{R}$ can be bounded from above by $\phi(n,k)$, i.e.,

$$
q \leq \phi(n,k) = \min \left(\sum_{i=0}^{n} \binom{\frac{k^2-k}{2}}{i}, k! \right). \tag{1.54}
$$

Substituting q with $\phi(n,k)$ in Theorem 2 proves the claim. □

Proof of Lemma 3

Proof. Obviously, a one-layer ReLU network is an affine function whose weights can be set to fulfill the identity mapping in $\mathbb{R}^n$. We prove the case when the number of layers is more than one in the next paragraph. We start with a scalar case, and then work on the vector case.

For any $x \in \mathbb{R}$, it holds that $\max(0,x) - \max(0,-x) = x$. In other words, a hidden layer of two ReLUs with $+1$ and -1 weights can represent an identity mapping for any scalar. For any vector input in $\mathbb{R}^n$, we can concatenate such structures of two ReLUs in parallel because the identity mapping can be decomposed into n individual identity mappings from n coordinates. Therefore, a two-layer ReLU network with $2n$ hidden neurons can realize the identity mapping in $\mathbb{R}^n$. Stacking such a hidden layer any number of times gives a deeper network that is still an identity mapping. Algorithm 5 follows from the above construction. Its complexity analysis is deferred to Table 1.5 in Appendix 1.6.3. $\square$

Proof of Lemma 4

Proof. Because a composition of two affine mappings is still affine, the first layer of either one of the two networks can be absorbed into the last layer of the other one if their dimensions are compatible. The resulting new network still satisfies Definition 4. The number of layers of the new network is $l_1 + l_2 - 1$. The number of hidden neurons of the new network is $n_1 + n_2$. The maximum width of the new network is at most $\max(w_1, w_2)$. Algorithm 4 follows from the above construction. Its complexity analysis is deferred to Table 1.4 in Appendix 1.6.3. $\square$

Proof of Lemma 5

Proof. For any positive even integer $k \geq 4$, it holds true that

$$r(k) - r(k-1) = \frac{3k}{2} + r\left(\frac{k}{2}\right) - 2 - \frac{3(k-2)}{2} - r\left(\frac{k}{2}\right) = 1. \tag{1.55}$$

For any positive odd integer k such that $k \geq 3$, we have

$$r(k) - r(k-1) = 2 + \frac{3(k-1)}{2} + r\left(\frac{k+1}{2}\right) - \frac{3(k-1)}{2} - r\left(\frac{k-1}{2}\right)$$
$$= \begin{cases} 3, & \text{if } \frac{k+1}{2} \text{ is even} \\ 2 + r\left(\frac{k+1}{2}\right) - r\left(\frac{k+1}{2} - 1\right), & \text{otherwise} \end{cases} \tag{1.56}$$

which is strictly greater than zero. Note that (1.56) is greater than 0 because the equality in (1.56) can be applied over and over again to reach (1.55) or the base case $r(2) - r(1) = 3$. □

Proof of Lemma 6

Proof. By Lemma 5, $r(k)$ is a strictly increasing sequence. Then, it must be true that

$$r(k) = r\left(2^{\log_2 k}\right) \leq r\left(2^{\lceil \log_2 k \rceil}\right). \tag{1.57}$$

According to the recursion (1.15), it holds that

$$\begin{aligned} r\left(2^{\lceil \log_2 k \rceil}\right) &= \frac{3}{2} \sum_{i=1}^{\lceil \log_2 k \rceil} 2^i \\ &= \frac{3}{2}\left(2^{\lceil \log_2 k \rceil + 1} - 2\right) \\ &= 3\left(2^{\lceil \log_2 k \rceil} - 1\right) \\ &< 3\left(2^{(\log_2 k) + 1} - 1\right) \\ &= 3(2k - 1). \end{aligned} \tag{1.58}$$

□

Proof of Lemma 7

Proof. Two ReLU networks can be combined in parallel such that the new network shares the same input and the two output vectors from the two ReLU networks are concatenated together. To see this, we show that the weights of the new network can be found by the following operations. Let $\mathbf{W}_i^1$ and $\mathbf{b}_i^1$ be the weights of the i-th layer in $g_{(l_1,n_1,w_1)}$, and $\mathbf{W}_i^2$ and $\mathbf{b}_i^2$ are the weights of the i-th layer in $g_{(l_2,n_2,w_2)}$. Let $\mathbf{W}_i$ and $\mathbf{b}_i$ be the weights of the new network. Now, we find the weights for the new network. In the first layer, we construct

$$\mathbf{W}_1 = \begin{bmatrix} \mathbf{W}_1^1 \\ \mathbf{W}_1^2 \end{bmatrix} \tag{1.59}$$

and

$$\mathbf{b}_1 = \begin{bmatrix} \mathbf{b}_1^1 \\ \mathbf{b}_1^2 \end{bmatrix}. \tag{1.60}$$

For the i-th layer such that $1 < i \leq \min(l_1, l_2)$, we use

$$\mathbf{W}_i = \begin{bmatrix} \mathbf{W}_i^1 & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_i^2 \end{bmatrix} \tag{1.61}$$

and

$$\mathbf{b}_i = \begin{bmatrix} \mathbf{b}_i^1 \\ \mathbf{b}_i^2 \end{bmatrix}. \tag{1.62}$$

If $l_1 = l_2$, then the claim is proved. If $l_1 \neq l_2$, then we stack a network that implements the identity mapping to the shallower network such that the numbers of layers of the two networks are the same. Because the network $g_{(l_i,n_i,w_i)}$ is shallower than the other network, we append $|l_1 - l_2|$ hidden layers to $g_{(l_i,n_i,w_i)}$ such that the procedure in (1.61) and (1.62) can be used. By Lemma 3, there exists an $(|l_1 - l_2| + 1)$-layer ReLU network $g_{(|l_1-l_2|+1, 2m_i|l_1-l_2|, 2m_i)}$ with $2m_i|l_1 - l_2|$ hidden neurons and a maximum width bounded from above by $2m_i$ for representing the identity mapping in $\mathbb{R}^{m_i}$. By Lemma 4, there exists a network $g_{\left(l_i+|l_1-l_2|, n_i+2m_i|l_1-l_2|, \max(w_i, 2m_i)\right)}$ that represents the composition of $g_{(|l_1-l_2|+1, 2m_i|l_1-l_2|, 2m_i)}$ and $g_{(l_i,n_i,w_i)}$. Now, (1.61) and (1.62) can be used to combine $g_{(l_j,n_j,w_j)}$ and $g_{\left(l_i+|l_1-l_2|, n_i+2m_i|l_1-l_2|, \max(w_i, 2m_i)\right)}$ in parallel because the number of layers in network $g_{(l_i+|l_1-l_2|, n_i+2m_i|l_1-l_2|, \max(w_i,2m_i))}$ is equal to l_j according to the fact that $l_i + |l_1 - l_2| = \max(l_1, l_2) = l_j$. Such a new network has $\max(l_1, l_2)$ layers and

$$n_j + n_i + 2m_i|l_1 - l_2| = n_1 + n_2 + 2m_i|l_1 - l_2| \tag{1.63}$$

hidden neurons. The maximum width of the new network is at most $w_j + \max(w_i, 2m_i)$. □

Proof of Lemma 8

Proof. The case $k = 1$ is trivial. The case $k = 2$ is proved by Lemma 7, which gives a tighter bound on the maximum width. The number of layers and hidden neurons of the claim agree with Lemma 7 when $k = 2$. The claim can be proved by following a similar procedure from the proof of Lemma 7. By Lemma 3, we can stack an identity mapping realized by an $(l - l_i + 1)$-layer ReLU network with $2m_i(l - l_i)$ hidden neurons and a maximum width of $2m_i$ on the i-th network for all $i \in [k]$ such that $l_i < l$. In other words, we increase the number of hidden layers for any network whose number of layers is less than l such that the cascade of the network and the corresponding identity mapping has l layers. For all $i \in [k]$ such that $l_i < l$, the extended network has $n_i + 2m_i(l - l_i)$ hidden neurons and a maximum width at most $\max(w_i, 2m_i)$ according to Lemma 4. Because all the networks now have the same number of layers, we can directly combine them in parallel. Hence, the resulting new network has $\max_{i \in [k]} l_i$ layers and

$$\sum_{i \in [k]} n_i + 2m_i(l - l_i) \tag{1.64}$$

hidden neurons and a maximum width at most

$$\sum_{i \in [k]} \max(w_i, 2m_i). \tag{1.65}$$

Algorithm 3 directly follows from the above construction. Its complexity analysis is deferred to Table 1.3 in Appendix 1.6.3. □

Proof of Lemma 9

Proof. Because $\mathscr{S}^n_{f_1,f_2,\cdots,f_k}$ is a subset of $\mathfrak{S}(k)$ and $\left|\mathfrak{S}(k)\right| = k!$ is the number of permutations of k distinct objects, it follows that

$$\left|\mathscr{S}^n_{f_1,f_2,\cdots,f_k}\right| \leq k!. \tag{1.66}$$

On the other hand, the number of hyperplanes, or affine subspaces of $\mathbb{R}^n$ with dimension $n-1$, induced by the distinct intersections between any two different affine functions is bounded from above by

$$\binom{k}{2}. \tag{1.67}$$

Let the arrangement of these hyperplanes be $\mathscr{A}$, and $|\mathscr{A}|$ be the number of hyperplanes in the arrangement. By Zaslavsky's Theorem [Zaslavsky, 1975], the number of connected components of the set

$$\mathbb{R}^n \setminus \bigcup_{H\in\mathscr{A}} H \tag{1.68}$$

is bounded from above by

$$\sum_{i=0}^{n} \binom{|\mathscr{A}|}{i} \tag{1.69}$$

Because there are at most $\binom{k}{2}$ hyperplanes in $\mathbb{R}^n$, it follows that

$$\left|\mathscr{S}^n_{f_1,f_2,\cdots,f_k}\right| \leq \sum_{i=0}^{n} \binom{\binom{k}{2}}{i}. \tag{1.70}$$

Combining (1.66) and (1.70) proves the claim. Notice that the ascending order does not change within a connected component. □

Proof of Lemma 10

Proof. Let $\mathscr{X}_1, \mathscr{X}_2, \cdots, \mathscr{X}_I$ be a family of nonempty subsets satisfying Definition 1 for a non-affine function. We prove the claim by contradiction. Suppose that there exists at least one nonempty closed subset, say $\mathscr{X}_i$, that is disjoint with every other closed subset $\mathscr{X}_j, j \in [I] \setminus i$. It follows that

$$\mathscr{X}_i \bigcap \bigcup_{j\in[I]\setminus i} \mathscr{X}_j = \emptyset \tag{1.71}$$

which implies

$$\left(\mathbb{R}^n \setminus \mathscr{X}_i\right) \bigcup \left(\mathbb{R}^n \setminus \bigcup_{j\in[I]\setminus i} \mathscr{X}_j\right) = \mathbb{R}^n. \tag{1.72}$$

Because the union of any finite collection of closed sets is closed, it must be true that $\bigcup_{j\in[I]\setminus i}\mathscr{X}_j$ is closed. Notice that $\mathscr{X}_i$ is never the whole space $\mathbb{R}^n$ because the CPWL function is assumed to be non-affine. $\bigcup_{j\in[I]\setminus i}\mathscr{X}_j$ must be nonempty due to Definition 1. Therefore, both $\mathbb{R}^n \setminus \mathscr{X}_i$ and $\mathbb{R}^n \setminus \bigcup_{j\in[I]\setminus i}\mathscr{X}_j$ are nonempty and open. Since $\mathbb{R}^n$ is connected, it cannot be represented as the union of two disjoint nonempty open subsets. It follows that the intersection between $\mathbb{R}^n \setminus \mathscr{X}_i$ and $\mathbb{R}^n \setminus \bigcup_{j\in[I]\setminus i}\mathscr{X}_j$ is nonempty. In other words, there exists an element of $\mathbb{R}^n$ that is not in $\mathscr{X}_i$ and $\bigcup_{j\in[I]\setminus i}\mathscr{X}_j$, contradicting Definition 1. □

Proof of Lemma 11(a)

Proof. If the CPWL function is affine, then there are no intersecting closed subsets because the only closed subset satisfying Assumption 1 is $\mathbb{R}^n$. On the other hand, if the CPWL function is non-affine, then there exist at least two intersecting closed subsets according to Lemma 10. For any two intersecting closed subsets, say $\mathscr{X}_i$ and $\mathscr{X}_j$, we first show that

$$\{\mathbf{x}\in\mathbb{R}^n \mid f_i(\mathbf{x}) = f_j(\mathbf{x})\} \neq \emptyset \tag{1.73}$$

where f_i and f_j are the affine functions corresponding to $\mathscr{X}_i$ and $\mathscr{X}_j$. We prove this statement by contradiction. Suppose that the intersection is empty, i.e., the linear equation $\left(\mathbf{a}_i - \mathbf{a}_j\right)^T \mathbf{x} + b_i - b_j = 0$ does not have a solution where $f_i(\mathbf{x}) = \mathbf{a}_i^T\mathbf{x} + b_i$ and $f_j(\mathbf{x}) = \mathbf{a}_j^T\mathbf{x} + b_j$ for $\mathbf{a}_i, \mathbf{a}_j \in \mathbb{R}^n$ and $b_i, b_j \in \mathbb{R}$. Then, it is necessary that $\mathbf{a}_i = \mathbf{a}_j$ and $b_i \neq b_j$. In other words, the two affine functions are parallel, implying that every point in $\mathscr{X}_i \cap \mathscr{X}_j$ gives two different values, which cannot be true for a valid function.

Next, we prove that there does not exist an intersection that is $\mathbb{R}^n$ by contradiction. Let us assume that there exists at least one intersection that is $\mathbb{R}^n$ between the affine functions corresponding to two intersecting closed subsets, say $\mathscr{X}_i$ and $\mathscr{X}_j$. Then, we can always replace $\mathscr{X}_i$ and $\mathscr{X}_j$ with their union. Such a replacement still satisfies Definition 1 but reduces the number of closed (connected) subsets by at least one, contradicting the fact that the number of closed subsets is a minimum. Because the two affine functions are identical if and only if the intersection is $\mathbb{R}^n$, the two affine functions must be different. $\square$

Proof of Lemma 11(b)

Proof. The claim follows from Lemma 11(a). Because the two affine functions have a nonempty intersection, their intersection must be $\mathbb{R}^n$ or an affine subspace of $\mathbb{R}^n$ with dimension $n-1$. However, the two affine functions must be different, implying that $\mathbb{R}^n$ is never the intersection. $\square$

Proof of Lemma 11(c)

Proof. Let any given two intersecting subsets be $\mathscr{X}_i$ and $\mathscr{X}_j$. The intersection between their corresponding affine functions, say f_i and f_j, is given by $\mathscr{H}_{ij} = \{\mathbf{x} \in \mathbb{R}^n \mid f_i(\mathbf{x}) = f_j(\mathbf{x})\}$. Suppose that there exists a point $\mathbf{a} \in \mathscr{X}_i \bigcap \mathscr{X}_j$ such that $\mathbf{a} \notin \mathscr{H}_{ij}$, then it follows that $f_i(\mathbf{a}) \neq f_j(\mathbf{a})$. Such a result cannot be true for a valid function. We conclude that $\mathscr{X}_i \bigcap \mathscr{X}_j \subseteq \mathscr{H}_{ij}$. $\square$

Proof of Lemma 11(d)

Proof. We prove the statement by contradiction. Suppose there exists a point $\mathbf{c} \in \mathbb{R}^n$ in the intersection of two intersecting closed connected subsets, say $\mathscr{X}_i$ and $\mathscr{X}_j$, such that $\mathbf{c}$ is an interior point of $\mathscr{X}_i$, then there exists an open ε-radius ball $B(\mathbf{c}, \varepsilon)$ such that $\mathbf{x} \in \mathscr{X}_i, \forall \mathbf{x} \in B(\mathbf{c}, \varepsilon)$ for some $\varepsilon > 0$. By Lemma 11(b), the intersection between the two affine functions corresponding to $\mathscr{X}_i$ and $\mathscr{X}_j$ must be an affine subspace of $\mathbb{R}^n$ with dimension $n-1$. Let such an affine subspace be denoted as $\mathscr{H}_{ij}$ and its corresponding linear subspace be denoted as $\mathscr{V}(\mathscr{H}_{ij})$. Then, there exists a nonzero vector $\mathbf{d} \in \mathbb{R}^n$ such that $\alpha\mathbf{d} \perp \mathbf{v}$ for all $\mathbf{v} \in \mathscr{V}(\mathscr{H}_{ij})$ and any $\alpha \neq 0$. Therefore, it follows that $\alpha\mathbf{d}+\mathbf{a} \notin \mathscr{H}_{ij}$ for any $\mathbf{a} \in \mathscr{H}_{ij}$ and any $\alpha \neq 0$. According to Lemma 11(c), $\mathscr{X}_i \cap \mathscr{X}_j \subseteq \mathscr{H}_{ij}$, so we have $\alpha\mathbf{d}+\mathbf{c} \notin \mathscr{X}_i \cap \mathscr{X}_j$ for any $\alpha \neq 0$. When $\alpha = \frac{\varepsilon}{2\|\mathbf{d}\|_2}$ or $\alpha = \frac{-\varepsilon}{2\|\mathbf{d}\|_2}$, $\alpha\mathbf{d}+\mathbf{c} \in B(\mathbf{c},\varepsilon)$. However, one of them must satisfy $\alpha\mathbf{d}+\mathbf{c} \notin \mathscr{X}_i$, contradicting the existence of a point in $\mathscr{X}_i \cap \mathscr{X}_j$ that is an interior point of $\mathscr{X}_i$. The same procedure can be applied to prove that there does not exist a point in $\mathscr{X}_i \cap \mathscr{X}_j$ such that it is an interior point of $\mathscr{X}_j$. We conclude that every element in $\mathscr{X}_i \cap \mathscr{X}_j$ is not an interior point of $\mathscr{X}_i$ or $\mathscr{X}_j$. □

Proof of Lemma 12(a)

Proof. The boundary or frontier of $\mathscr{X}_i$ is given by

$$\begin{aligned}
\mathrm{Fr}\mathscr{X}_i &= \overline{\mathscr{X}_i} \bigcap \overline{\mathbb{R}^n \setminus \mathscr{X}_i} \\
&= \mathscr{X}_i \bigcap \overline{\left(\bigcup_{k\in[I]} \mathscr{X}_k\right) \setminus \mathscr{X}_i} \\
&= \mathscr{X}_i \bigcap \overline{\bigcup_{k\in[I]\setminus i} \left(\mathscr{X}_k \setminus \mathscr{X}_k \bigcap \mathscr{X}_i\right)} \\
&= \mathscr{X}_i \bigcap \bigcup_{k\in[I]\setminus i} \overline{\left(\mathscr{X}_k \setminus \mathscr{X}_k \bigcap \mathscr{X}_i\right)} \\
&= \mathscr{X}_i \bigcap \bigcup_{k\in[I]\setminus i} \overline{\mathscr{X}_k} \\
&= \mathscr{X}_i \bigcap \bigcup_{k\in[I]\setminus i} \mathscr{X}_k \\
&= \bigcup_{k\in[I]\setminus i} \mathscr{X}_k \bigcap \mathscr{X}_i
\end{aligned} \tag{1.74}$$

where $\overline{\mathscr{A}}$ denotes the closure of a subset $\mathscr{A}$. We have used Lemma 11(d) for the equality between the 4-th and 5-th line of (1.74). Now, we prove that the interior of $\mathscr{X}_i$ is nonemtpy by contradiction. Suppose that the interior of $\mathscr{X}_i$ is empty, then it follows that $\mathscr{X}_i = \overline{\mathscr{X}_i} = \mathrm{Fr}\mathscr{X}_i$ because the closure of $\mathscr{X}_i$ is the union of the interior and the boundary of $\mathscr{X}_i$. Combining that with (1.74), we have $\mathscr{X}_i = \bigcup_{k\in[I]\setminus i} \mathscr{X}_k \bigcap \mathscr{X}_i$. which implies every element in $\mathscr{X}_i$ is at least covered by one of the other closed subsets $\mathscr{X}_k$ for some $k \in [I] \setminus i$. In this case, we can delete $\mathscr{X}_i$

from $\mathscr{X}_1, \mathscr{X}_2, \cdots, \mathscr{X}_I$; and the remaining $I-1$ closed subsets still satisfy Definition 1. Such a valid deletion of $\mathscr{X}_i$ contradicts the fact that I is the minimum number of closed subsets. Hence, the interior of $\mathscr{X}_i$ must be nonempty. □

Proof of Lemma 12(b)

Proof. The statement is proved by (1.74) in Lemma 12(a). □

Proof of Lemma 12(c)

Proof. By Lemma 12(a), the interior of every subset is nonempty. Next, by Lemma 11(d), every point in the intersection between any two subsets is a boundary point of both subsets. It follows that the interiors of any two subsets are disjoint. □

Proof of Lemma 13

Proof. By Lemma 12(a), the interior of $\mathscr{X}_i$ is nonempty. Therefore, there exists an open ε-radius ball $B(\mathbf{c}_0, \varepsilon)$ such that $\mathbf{x} \in \mathscr{X}_i, \forall \mathbf{x} \in B(\mathbf{c}_0, \varepsilon)$ for some $\varepsilon > 0$ and $\mathbf{c}_0 \in \mathscr{X}_i$. Let us consider the set

$$\bigcap_{j \in [k]} \left(B(\mathbf{c}_0, \varepsilon) \bigcap \left(\mathscr{H}_j^+ \bigcup \mathscr{H}_j^- \right) \right) \tag{1.75}$$

where $\mathscr{H}_j^+$ and $\mathscr{H}_j^-$ are two open half spaces created by $\mathscr{H}_j$. It suffices to show the nonemptyness of the set in (1.75) to prove the claim. If $\mathscr{H}_j$ and $B(\mathbf{c}_0, \varepsilon)$ do not intersect, then $B(\mathbf{c}_0, \varepsilon)$ completely belongs to $\mathscr{H}_j^+$ or $\mathscr{H}_j^-$. Without loss of generality, we can remove all j such that $\mathscr{H}_j$ does not intersect $B(\mathbf{c}_0, \varepsilon)$ and assume there are k affine subspaces of $\mathbb{R}^n$ with dimension $n-1$ intersecting $B(\mathbf{c}_0, \varepsilon)$. Let us sequentially carry out the intersection in (1.75). Every time before the operation of the j-th intersection between $B(\mathbf{c}_{j-1}, \frac{\varepsilon}{2^{j-1}})$ and $\left(\mathscr{H}_j^+ \bigcup \mathscr{H}_j^- \right)$, there exists an open $\frac{\varepsilon}{2^j}$-radius ball $B(\mathbf{c}_j, \frac{\varepsilon}{2^j})$ for some $\mathbf{c}_j \in B(\mathbf{c}_{j-1}, \frac{\varepsilon}{2^{j-1}})$ such that it does not intersect with $\mathscr{H}_j$. Therefore, at the end of the sequential process, there exists an open ball that does not intersect any of these k affine subspaces of $\mathbb{R}^n$ with dimension $n-1$. The set in (1.75) is nonempty, implying (1.27) holds true. □

Proof of Proposition 1

Proof. We prove the claim by contraposition. If the number of closed connected subsets is not a minimum, i.e., Assumption 1 is not satisfied, then such a number can be decreased by merging some of the intersecting closed connected subsets that have the same corresponding affine functions. Therefore, there exist at least two closed connected subsets that can be made larger.

On the other hand, if the closed connected subsets, say $\mathscr{X}_1, \mathscr{X}_2, \cdots, \mathscr{X}_I$, have at least one of the subsets that can be made larger, then there exist at least two intersecting closed connected subsets, say $\mathscr{X}_i$ and $\mathscr{X}_j$, from $\mathscr{X}_1, \mathscr{X}_2, \cdots, \mathscr{X}_I$ such that their corresponding affine functions are the same. Otherwise, any closed connected subset cannot be made larger than itself. Therefore, $\mathscr{X}_i$ and $\mathscr{X}_j$ can be replaced with $\mathscr{X}_i \bigcup \mathscr{X}_j$ and these $I-1$ closed connected subsets still satisfy Definition 1, implying that I is not the minimum. $\square$

1.6.3 Algorithms and time complexities

Table 1.1. The running time of Algorithm 1 is upper bounded by $\mathsf{poly}(n,k,q,L)$.

Line	Operation count	Explanation
1	$\mathscr{O}\left(nq\max(n^2,q)\right)$	Algorithm 6 (see Table 1.6).
2	$\mathscr{O}(q)$	Repeat Line 3 to Line 9 q times.
3	$\mathscr{O}(1)$	Initialize an empty placeholder.
4	$\mathscr{O}(k)$	Repeat Line 5 to Line 7 k times.
5	$\mathsf{poly}(n,q,L)$	Solve a linear program [Vavasis and Ye, 1996].
6	$\mathscr{O}(1)$	Add an index.
7	-	-
8	-	-
9	$\mathscr{O}\left(k^2\max(k\log_2 k,n)\right)$	Algorithm 2 (see Table 1.2).
10	-	-
11	$\mathscr{O}\left(q\max(n,k)^2\max(n,k,q)\log_2 k\right)$	Algorithm 3 (see Table 1.3).
12	$\mathscr{O}\left(q^3\log_2 q\right)$	Algorithm 2 (see Table 1.2).
13	$\mathscr{O}\left(q^3\max(n,k)^3\log_2 q\right)$	Algorithm 4 (see Table 1.4).

Algorithm 2. Find a ReLU network that computes the extremum of affine functions

Input: Scalar-valued affine functions $f_1, \cdots, f_m$ on $\mathbb{R}^n$ and the type of extremum (max or min).
Output: Parameters of an l-layer ReLU network g computing $g = \max_{i\in[m]} f_i$ or $g = \min_{i\in[m]} f_i$.

```
1: $\mathbf{A} \leftarrow \begin{bmatrix} -1 & 1 & -1 \\ 1 & 0 & 0 \end{bmatrix}^\mathsf{T}, \mathbf{B} \leftarrow \begin{bmatrix} 1 & 1 & -1 \end{bmatrix}, \mathbf{C} \leftarrow \begin{bmatrix} 1 \\ -1 \end{bmatrix}$   ▷ Constant matrices
2: $\boldsymbol{\Psi}(\mathbf{Y},\mathbf{Z}) \leftarrow \operatorname{diag}(\mathbf{Y},\mathbf{Z})$, $\boldsymbol{\Phi}(\mathbf{Y},s) \leftarrow \operatorname{diag}(\mathbf{Y},\mathbf{Y},\cdots,\mathbf{Y})$   ▷ The latter repeats $\mathbf{Y}$ $s$ times
3: $l \leftarrow \lceil \log_2 m \rceil + 1, k_0 \leftarrow n, k_l \leftarrow 1, c_0 \leftarrow m$   ▷ $l$ is the number of layers of $g$
4: for $i = 1,2,\cdots,l-1$ do
5:     if $c_{i-1}$ is even then
6:         $c_i \leftarrow \frac{c_{i-1}}{2}$
7:         $k_i \leftarrow 3c_i$   ▷ Output dimension of the $i$-th layer
8:     else
9:         $c_i \leftarrow \frac{c_{i-1}+1}{2}$
10:        $k_i \leftarrow 3c_i - 1$   ▷ Output dimension of the $i$-th layer
11:    end if
12: end for
13: $\mathbf{W}_1 \leftarrow \begin{bmatrix} \nabla f_1 & \nabla f_2 & \cdots & \nabla f_m \end{bmatrix}^\mathsf{T}, \mathbf{b}_1 \leftarrow \begin{bmatrix} f_1(0) & f_2(0) & \cdots & f_m(0) \end{bmatrix}^\mathsf{T}$
14: if $l > 1$ then   ▷ Find the weights of input and output layers, if any
15:     if $c_0$ is even then
16:         $\mathbf{W}_1 \leftarrow \boldsymbol{\Phi}(\mathbf{A},c_1)\mathbf{W}_1, \mathbf{b}_1 \leftarrow \boldsymbol{\Phi}(\mathbf{A},c_1)\mathbf{b}_1$
17:     else
18:         $\mathbf{W}_1 \leftarrow \boldsymbol{\Psi}(\boldsymbol{\Phi}(\mathbf{A},c_1-1),\mathbf{C})\mathbf{W}_1, \mathbf{b}_1 \leftarrow \boldsymbol{\Psi}(\boldsymbol{\Phi}(\mathbf{A},c_1-1),\mathbf{C})\mathbf{b}_1$
19:     end if
20:     $\mathbf{W}_l \leftarrow \mathbf{B}, \mathbf{b}_l \leftarrow \mathbf{0}_{k_l}$
21: end if
22: if $l > 2$ then   ▷ Find the weights of remaining layers, if any
23:     for $i = 2,3,\cdots,l-1$ do
24:         if $c_{i-1}$ is even then
25:             $\mathbf{T} \leftarrow \boldsymbol{\Phi}(\mathbf{A},c_i)$
26:         else
27:             $\mathbf{T} \leftarrow \boldsymbol{\Psi}(\boldsymbol{\Phi}(\mathbf{A},c_i-1),\mathbf{C})$
28:         end if
29:         if $c_{i-2}$ is even then
30:             $\mathbf{W}_i \leftarrow \mathbf{T}\boldsymbol{\Phi}(\mathbf{B},c_{i-1})$
31:         else
32:             $\mathbf{W}_i \leftarrow \mathbf{T}\boldsymbol{\Psi}\left(\boldsymbol{\Phi}(\mathbf{B},c_{i-1}-1),\mathbf{C}^\mathsf{T}\right)$
33:         end if
34:         $\mathbf{b}_i \leftarrow \mathbf{0}_{k_i}$
35:     end for
36: end if
37: if type of extremum is the minimum then
38:     $\mathbf{W}_1 \leftarrow -\mathbf{W}_1, \mathbf{b}_1 \leftarrow -\mathbf{b}_1$, $\mathbf{W}_l \leftarrow -\mathbf{W}_l, \mathbf{b}_l \leftarrow -\mathbf{b}_l$
39: end if   ▷ See Table 1.2 in Appendix 1.6.3 for complexity analysis
```

Algorithm 3. Find a ReLU network that concatenates a number of given ReLU networks

Input: Weights of k ReLU networks $g_1, g_2, \cdots, g_k$ denoted by $\{\mathbf{W}_i^j, \mathbf{b}_i^j\}_{i=1}^{l_j}$ for $j \in [k]$.

Output: Parameters of an l-layer ReLU network g computing $g(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_k(\mathbf{x}) \end{bmatrix}, \forall \mathbf{x} \in \mathbb{R}^n$.

```
1:  $l \leftarrow \max_{j\in[k]} l_j$
2:  $\mathbf{W}_1 \leftarrow \begin{bmatrix} \mathbf{W}_1^1 \\ \mathbf{W}_1^2 \\ \vdots \\ \mathbf{W}_1^k \end{bmatrix}, \mathbf{b}_1 \leftarrow \begin{bmatrix} \mathbf{b}_1^1 \\ \mathbf{b}_1^2 \\ \vdots \\ \mathbf{b}_1^k \end{bmatrix}$        ▷ Weights of the input layer
3:  for $j = 1, 2, \cdots, k$ do
4:      if $l_j < l$ then    ▷ Append an identity mapping network to the network if it is shallower
5:          $m \leftarrow$ output dimsion of $g_j$
6:          $g_j^c \leftarrow$ run Algorithm 5 with an input dimension $m$ and a number of layers $l - l_j + 1$
7:          $g_j' \leftarrow$ run Algorithm 4 with $g_j$ and $g_j^c$
8:          $\{\mathbf{W}_i^j, \mathbf{b}_i^j\}_{i=1}^{l} \leftarrow$ weights of $g_j'$
9:      end if
10: end for
11: for $i = 2, 3, \cdots, l$ do                ▷ Find the remaining weights
12:     $\mathbf{W}_i \leftarrow \begin{bmatrix} \mathbf{W}_i^1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_i^2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{W}_i^k \end{bmatrix}, \mathbf{b}_i \leftarrow \begin{bmatrix} \mathbf{b}_i^1 \\ \mathbf{b}_i^2 \\ \vdots \\ \mathbf{b}_i^k \end{bmatrix}$
13: end for                  ▷ See Table 1.3 in Appendix 1.6.3 for complexity analysis
```

Algorithm 4. Find a ReLU network computing a composition of two given ReLU networks

Input: Weights of two ReLU networks g_1 and g_2 denoted by $\{\mathbf{W}_i^1, \mathbf{b}_i^1\}_{i=1}^{l_1}$ and $\{\mathbf{W}_i^2, \mathbf{b}_i^2\}_{i=1}^{l_2}$.
Output: Parameters of an l-layer ReLU network g computing $g(\mathbf{x}) = g_2\left(g_1(\mathbf{x})\right), \forall \mathbf{x} \in \mathbb{R}^n$.

```
1:  $l \leftarrow l_1 + l_2 - 1$
2:  for $i = 1, 2, \cdots, l$ do
3:      if $i < l_1$ then     ▷ The first $l_1 - 1$ layers are identical to the corresponding layers in $g_1$
4:          $\mathbf{W}_i \leftarrow \mathbf{W}_i^1, \mathbf{b}_i \leftarrow \mathbf{b}_i^1$
5:      else if $i = l_1$ then        ▷ A composition of affine functions is still an affine function
6:          $\mathbf{W}_i \leftarrow \mathbf{W}_1^2 \mathbf{W}_{l_1}^1, \mathbf{b}_i \leftarrow \mathbf{W}_1^2 \mathbf{b}_{l_1}^1 + \mathbf{b}_1^2$
7:      else              ▷ The last $l_2 - 1$ layers are identical to the corresponding layers in $g_2$
8:          $\mathbf{W}_i \leftarrow \mathbf{W}_{i-l_1+1}^2, \mathbf{b}_i \leftarrow \mathbf{b}_{i-l_1+1}^2$
9:      end if
10: end for                  ▷ See Table 1.4 in Appendix 1.6.3 for complexity analysis
```

Algorithm 5. Find a ReLU network that computes an identity mapping for a given depth

Input: The input dimension n and the number of layers l of the target ReLU network.

Output: Parameters of an l-layer ReLU network g computing $g(\mathbf{x}) = \mathbf{x}, \forall \mathbf{x} \in \mathbb{R}^n$.

```
1: $\mathbf{A} \leftarrow \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \mathbf{B} \leftarrow \begin{bmatrix} 1 & -1 \end{bmatrix}, \mathbf{C} \leftarrow \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$    ▷ Constant matrices
2: $\mathbf{\Phi}(\mathbf{Y}, s) = \begin{bmatrix} \mathbf{Y}^{(1)} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}^{(2)} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{Y}^{(s)} \end{bmatrix}$    ▷ A block diagonal matrix with $\mathbf{Y}$ repeated $s$ times
3: $k_0 \leftarrow n, k_l \leftarrow n, \mathbf{b}_l \leftarrow \mathbf{0}_n$
4: for $i = 1, 2, \cdots, l-1$ do
5:     $k_i \leftarrow 2n$    ▷ The number of hidden neurons at the $i$-th hidden layer
6:     $\mathbf{b}_i \leftarrow \mathbf{0}_{k_i}$
7: end for
8: if $l = 1$ then    ▷ Find the weights of input and output layers, if any
9:     $\mathbf{W}_1 \leftarrow \mathbf{I}_{n \times n}$    ▷ An identity matrix
10: else
11:     $\mathbf{W}_1 \leftarrow \mathbf{\Phi}(\mathbf{A}, k_0)$
12:     $\mathbf{W}_l \leftarrow \mathbf{\Phi}(\mathbf{B}, k_l)$
13: end if
14: if $l > 2$ then    ▷ Find the weights of hidden layers, if any
15:     for $i = 2, 3, \cdots, l-1$ do
16:         $\mathbf{W}_i \leftarrow \mathbf{\Phi}(\mathbf{C}, n)$
17:     end for
18: end if    ▷ See Table 1.5 in Appendix 1.6.3 for complexity analysis
```

Algorithm 6. Find all distinct linear components of a CPWL function

Input: An unknown CPWL function p whose output can be observed by feeding input from $\mathbb{R}^n$ to the function. A center $\mathbf{c}_i$ and radius $\varepsilon_i > 0$ of any closed ε_i-radius ball $B(\mathbf{c}_i, \varepsilon_i)$ such that $B(\mathbf{c}_i, \varepsilon_i) \subset \mathscr{X}_i$ for $i = 1, 2, \cdots, q$ where $\{\mathscr{X}_i\}_{i \in [q]}$ are all pieces of p.

Output: All distinct linear components of p, denoted by $\mathscr{F}$.

```
1:  $\mathscr{F} \leftarrow \emptyset$                ▷ Initialize the set of all distinct linear components
2:  for $i = 1, 2, \cdots, q$ do
3:      $\mathbf{x}_0 \leftarrow \mathbf{c}_i$                ▷ select the center of $B(\mathbf{c}_i, \varepsilon_i)$
4:      $y_0 \leftarrow p(\mathbf{x}_0)$
5:      $\begin{bmatrix} \mathbf{s}_1 & \mathbf{s}_2 & \cdots & \mathbf{s}_n \end{bmatrix} \leftarrow \varepsilon_i \mathbf{I}_{n \times n}$                ▷ scale the standard basis of $\mathbb{R}^n$
6:      $\mathbf{S} \leftarrow \begin{bmatrix} \mathbf{s}_1 & \mathbf{s}_2 & \cdots & \mathbf{s}_n \end{bmatrix}$
7:      $\mathbf{z} \leftarrow \begin{bmatrix} p(\mathbf{s}_1 + \mathbf{x}_0) - y_0 \\ p(\mathbf{s}_2 + \mathbf{x}_0) - y_0 \\ \vdots \\ p(\mathbf{s}_n + \mathbf{x}_0) - y_0 \end{bmatrix}$
8:      $\mathbf{a} \leftarrow \mathbf{S}^{-\mathsf{T}} \mathbf{z}$                ▷ Find the linear map by solving a system of linear equations
9:      $b \leftarrow y_0 - \mathbf{a}^{\mathsf{T}} \mathbf{x}_0$                ▷ Find the translation
10:     $f \leftarrow \mathbf{x} \mapsto \mathbf{a}^{\mathsf{T}} \mathbf{x} + b$                ▷ The affine map on $\mathscr{X}_i$
11:     if $f \notin \mathscr{F}$ then    ▷ Only add the affine map $f$ to $\mathscr{F}$ if $f$ is distinct to all elements of $\mathscr{F}$
12:         $\mathscr{F} \leftarrow \mathscr{F} \bigcup \{f\}$
13:     end if
14: end for                ▷ See Table 1.6 in Appendix 1.6.3 for complexity analysis
```

Table 1.2. The time complexity of Algorithm 2 is $\mathscr{O}\left(m^2 \max(m \log_2 m, n)\right)$.

Line	Operation count	Explanation
1	$\mathscr{O}(1)$	Initialize constant matrices.
2	$\max\left(\mathscr{O}\left(d_1^2\right), \mathscr{O}(s^2 d_1^2)\right)$	Let d_1 be the maximum dimension of $\mathbf{Y}$ and $\mathbf{Z}$.
3	$\mathscr{O}(1)$	Scalar assignments.
4	$\mathscr{O}(\log_2 m)$	Repeat Line 5 to Line 11 $\lceil \log_2 m \rceil$ times.
5	$\mathscr{O}(1)$	Check a scalar is even or not.
6-12	$\mathscr{O}(1)$	Compute a scalar.
13	$\mathscr{O}(mn)$	Assign a matrix and a vector.
14	$\mathscr{O}(1)$	Check a scalar inequality.
15	$\mathscr{O}(1)$	Check a scalar is even or not.
16-19	$\mathscr{O}(m^2 n)$	Matrix creation and multiplication.
20-21	$\mathscr{O}(1)$	Assign a constant matrix and vector.
22	$\mathscr{O}(1)$	Check a scalar inequality.
23	$\mathscr{O}(\log_2 m)$	Repeat Line 24 to Line 34 $\lceil \log_2 m \rceil - 1$ times.
24	$\mathscr{O}(1)$	Check a scalar is even or not.
25-28	$\mathscr{O}(m^2)$	Matrix creation.
29	$\mathscr{O}(1)$	Check a scalar is even or not.
30-32	$\mathscr{O}(m^3)$	Matrix creation and multiplication.
33	-	-
34	$\mathscr{O}(m)$	Assign a vector whose length is at most $\left\lceil \frac{3m}{2} \right\rceil$.
35-36	-	-
37	$\mathscr{O}(1)$	Check the binary data type.
38-39	$\mathscr{O}(mn)$	Reverse the sign of $\mathbf{W}_1$ and $\mathbf{b}_1$, and constants.

Table 1.3. The time complexity of Algorithm 3 is $\mathcal{O}\left(d^2kl\max(d,k)\right)$ where d is the maximum dimension of all the weight matrices in $g_1, g_2, \cdots, g_k$ and $l = \max_{j\in[k]} l_j$.

Line	Operation count	Explanation
1	$\mathcal{O}(k)$	Find the maximum among k numbers.
2	$\mathcal{O}(d^2k)$	Matrix concatenation and assignment.
3	$\mathcal{O}(k)$	Repeat Line 4 to Line 9 k times.
4	$\mathcal{O}(1)$	Check a scalar inequality.
5	$\mathcal{O}(1)$	A scalar assignment.
6	$\mathcal{O}(d^2l)$	Algorithm 5 (see Table 1.5).
7	$\mathcal{O}(d^3l)$	Algorithm 4 (see Table 1.4).
8	$\mathcal{O}(d^2l)$	Assign weights of the network.
9	-	-
10	-	-
11	$\mathcal{O}(l)$	Repeat Line 12 $l-1$ times.
12	$\mathcal{O}(d^2k^2)$	Assign a matrix and a vector.
13	-	-

Table 1.4. The time complexity of Algorithm 4 is $\mathscr{O}\left(d^3 \max(l_1, l_2)\right)$ where d is the maximum dimension of all the weight matrices in g_1 and g_2.

Line	Operation count	Explanation
1	$\mathscr{O}(1)$	Assign a constant.
2	$\mathscr{O}(l)$	Repeat Line 3 to Line 9 l times.
3	$\mathscr{O}(1)$	Check a scalar inequality.
4	$\mathscr{O}(d^2)$	Assign a matrix and a vector (at most $d^2 + d$ elements).
5	$\mathscr{O}(1)$	Check a scalar equality.
6	$\mathscr{O}(d^3)$	Matrix multiplication and assignment.
7	-	-
8	$\mathscr{O}(d^2)$	Assign a matrix and a vector (at most $d^2 + d$ elements).
9	-	-
10	-	-

Table 1.5. The time complexity of Algorithm 5 is $\mathcal{O}(n^2 l)$.

Line	Operation count	Explanation
1	$\mathcal{O}(1)$	Initialize constant matrices.
2	$\mathcal{O}(s^2 d_1 d_2)$	Create a block diagonal matrix from $\mathbf{Y} \in \mathbb{R}^{d_1 \times d_2}$ and $s \in \mathbb{N}$.
3	$\mathcal{O}(n)$	Assign two constant scalars and one constant vector of length n.
4	$\mathcal{O}(l)$	Repeat Line 5 to line 6 l times.
5	$\mathcal{O}(1)$	Assign a scalar.
6	$\mathcal{O}(n)$	Assign a vector whose length k_i is equal to $2n$.
7	-	-
8	$\mathcal{O}(1)$	Check a scalar equality.
9	$\mathcal{O}(n^2)$	Assign an n-by-n matrix.
10	-	-
11	$\mathcal{O}(n^2)$	Assign a $2n$-by-n block diagonal matrix.
12	$\mathcal{O}(n^2)$	Assign an n-by-$2n$ block diagonal matrix.
13	-	-
14	$\mathcal{O}(1)$	Check a scalar inequality.
15	$\mathcal{O}(l)$	Repeat Line 16 $l-2$ times.
16	$\mathcal{O}(n^2)$	Assign a $2n$-by-$2n$ block diagonal matrix.
17	-	-
18	-	-

Table 1.6. The time complexity of Algorithm 6 is $\mathcal{O}\left(nq\max(n^2,q)\right)$.

Line	Operation count	Explanation
1	$\mathcal{O}(1)$	Initialize an empty placeholder $\mathcal{F}$.
2	$\mathcal{O}(q)$	Repeat Line 3 to line 13 q times.
3	$\mathcal{O}(1)$	Select an interior point. Use the center of the ball.
4	$\mathcal{O}(1)$	Evaluate the function on the point.
5	$\mathcal{O}(n^2)$	Scale and assign an n-by-n matrix.
6	-	-
7	$\mathcal{O}(n)$	Translate, evaluate, and subtract n points.
8	$\mathcal{O}(n^3)$	Solve a system of n linear equations with n variables.
9	$\mathcal{O}(n)$	Solve the translation term in the affine map
10	-	-
11	$\mathcal{O}(nq)$	Each affine map has $n+1$ parameters. $\mathcal{F}$ has at most q elements.
12	$\mathcal{O}(1)$	Add a distinct affine map to $\mathcal{F}$.
13	-	-
14	-	-

1.6.4 Software implementation and run time of Algorithm 1

We implement Algorithm 1 in Python. Figure 1.3 shows that the run time of the algorithm is greatly affected by the number of pieces q.

Code is available at `https://github.com/kjason/CPWL2ReLUNetwork`.

1.7 Acknowledgements

We would like to thank the anonymous reviewers for their constructive comments, Tai-Hsuan Chung for answering our mathematical questions, and Christoph Hertrich for his thoughtful comments on the time complexity of Algorithm 1 and for clarifying Theorem 4.4 in

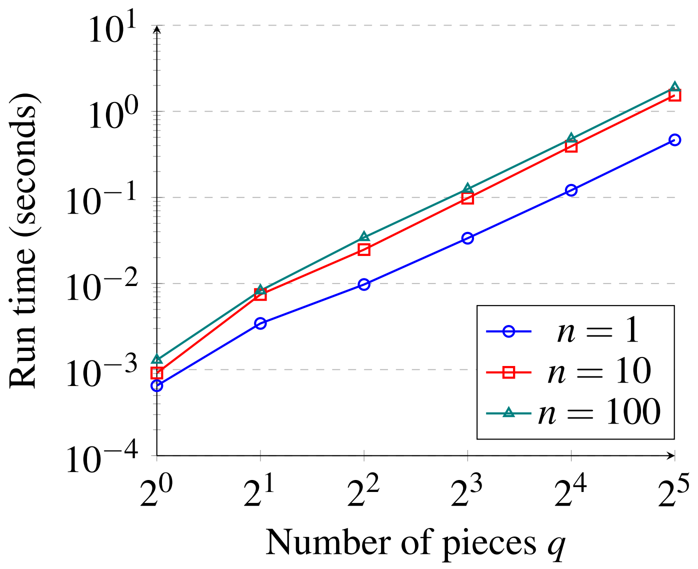


Figure 1.3. The run time of Algorithm 1 is an average of 50 trials. Every trial runs Algorithm 1 with a random CPWL function whose input dimension is n and number of pieces is q. The code provided in the above link is run on a computer (Microsoft Surface Laptop Studio) with the Intel Core i7-11370H.

[Hertrich et al., 2021]. This work was supported in part by NSF under Grant CCF-2225617, Grant CCF-2124929, and Grant IIS-1838897, in part by NIH/NIDCD under Grant R01DC015436, and in part by KIBM Innovative Research Grant Award.

Chapter 1, in full, is a reprint of the material as it appears in K.-L. Chen, H. Garudadri, and B. D. Rao, “Improved bounds on neural complexity for representing piecewise linear functions,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. The dissertation author was the primary investigator and author of this material.

Chapter 2

ResNEsts and DenseNEsts: Block-based DNN Models with Improved Representation Guarantees

Models recently used in the literature proving residual networks (ResNets) are better than linear predictors are actually different from standard ResNets that have been widely used in computer vision. In addition to the assumptions such as scalar-valued output or single residual block, the models fundamentally considered in the literature have no nonlinearities at the final residual representation that feeds into the final affine layer. To codify such a difference in nonlinearities and reveal a linear estimation property, we define ResNEsts, i.e., Residual Nonlinear Estimators, by simply dropping nonlinearities at the last residual representation from standard ResNets. We show that wide ResNEsts with bottleneck blocks can always guarantee a very desirable training property that standard ResNets aim to achieve, i.e., adding more blocks does not decrease performance given the same set of basis elements. To prove that, we first recognize ResNEsts are basis function models that are limited by a coupling problem in basis learning and linear prediction. Then, to decouple prediction weights from basis learning, we construct a special architecture termed augmented ResNEst (A-ResNEst) that always guarantees no worse performance with the addition of a block. As a result, such an A-ResNEst establishes empirical risk lower bounds for a ResNEst using corresponding bases. Our results demonstrate ResNEsts indeed have a problem of diminishing feature reuse; however, it can be avoided by

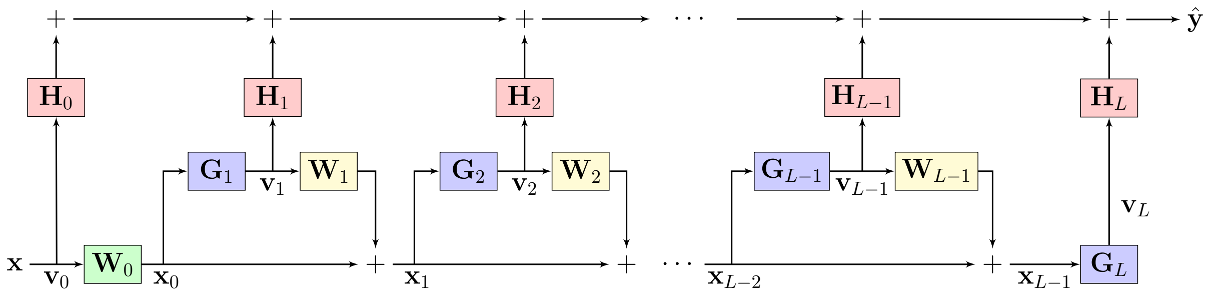


Figure 2.1. The proposed augmented ResNEst or A-ResNEst. A set of new prediction weights $\mathbf{H}_0, \mathbf{H}_1, \cdots, \mathbf{H}_L$ are introduced on top of the features in the ResNEst (see Figure 2.2). The A-ResNEst is always better than the ResNEst in terms of empirical risk minimization (see Proposition 3). Empirical results of the A-ResNEst model are deferred to Appendix 2.7.2 in the supplementary material.

sufficiently expanding or widening the input space, leading to the above-mentioned desirable property. Inspired by the densely connected networks (DenseNets) that have been shown to outperform ResNets, we also propose a corresponding new model called Densely connected Nonlinear Estimator (DenseNEst). We show that any DenseNEst can be represented as a wide ResNEst with bottleneck blocks. Unlike ResNEsts, DenseNEsts exhibit the desirable property without any special architectural re-design.

2.1 Introduction

Constructing deep neural network (DNN) models by stacking layers unlocks the field of deep learning, leading to the early success in computer vision, such as AlexNet [Krizhevsky et al., 2012], ZFNet [Zeiler and Fergus, 2014], and VGG [Simonyan and Zisserman, 2015a]. However, stacking more and more layers can suffer from worse performance [He and Sun, 2015, Srivastava et al., 2015, He et al., 2016a]; thus, it is no longer a valid option to further improve DNN models. In fact, such a *degradation problem* is not caused by overfitting, but worse training performance [He et al., 2016a]. When neural networks become sufficiently deep, optimization landscapes quickly transition from being nearly convex to being highly chaotic [Li et al., 2018]. As a result, stacking more and more layers in DNN models can easily converge to poor local minima (see Figure 1 in [He et al., 2016a]).

To address the issue above, the modern deep learning paradigm has shifted to designing DNN models based on blocks or modules of the same kind in cascade. A block or module comprises specific operations on a stack of layers to avoid the degradation problem and learn better representations. For example, Inception modules in the GoogLeNet [Szegedy et al., 2015], residual blocks in the ResNet [He et al., 2016a,b, Zagoruyko and Komodakis, 2016, Kim et al., 2016, Xie et al., 2017, Xiong et al., 2018], dense blocks in the DenseNet [Huang et al., 2017], attention modules in the Transformer [Vaswani et al., 2017], Squeeze-and-Excitation (SE) blocks in the SE network (SENet) [Hu et al., 2018], and residual U-blocks [Qin et al., 2020] in U^2-Net. Among the above examples, the most popular block design is the residual block which merely adds a skip connection (or a residual connection) between the input and output of a stack of layers. This modification has led to a huge success in deep learning. Many modern DNN models in different applications also adopt residual blocks in their architectures, e.g., V-Net in medical image segmentation [Milletari et al., 2016], Transformer in machine translation [Vaswani et al., 2017], and residual LSTM in speech recognition [Kim et al., 2017]. Empirical results have shown that ResNets can be even scaled up to 1001 layers or 333 bottleneck residual blocks, and still improve performance [He et al., 2016b].

Despite the huge success, our understanding of ResNets is very limited. To the best of our knowledge, no theoretical results have addressed the following question: *Is learning better ResNets as easy as stacking more blocks?* The most recognized intuitive answer for the above question is that a particular stack of layers can focus on fitting the residual between the target and the representation generated in the previous residual block; thus, adding more blocks always leads to no worse training performance. Such an intuition is indeed true for a constructively blockwise training procedure; but not clear when the weights in a ResNet are optimized as a whole. Perhaps the theoretical works in the literature closest to the above question are recent results in an albeit modified and constrained ResNet model that every local minimum is less than or equal to the empirical risk provided by the best linear predictor [Shamir, 2018, Kawaguchi and Bengio, 2019, Yun et al., 2019]. Although the aims of these works are different from our question, they actually prove a special case under these simplified models in which the final residual representation is better than the input representation for linear prediction. We notice that the models considered in these works are very different from standard ResNets using pre-activation residual blocks [He et al., 2016b] due to the absence of the nonlinearities at the final residual representation that feeds into the final affine layer. Other noticeable simplifications include scalar-valued output [Shamir, 2018, Yun et al., 2019] and single residual block [Shamir, 2018, Kawaguchi and Bengio, 2019]. In particular, Yun et al. [2019] additionally showed that residual representations do not necessarily improve monotonically over subsequent blocks, which highlights a fundamental difficulty in analyzing their simplified ResNet models.

In this paper, we take a step towards answering the above-mentioned question by constructing practical and analyzable block-based DNN models. Main contributions of our paper are as follows:

Improved representation guarantees for wide ResNEsts with bottleneck residual blocks

We define a ResNEst as a standard single-stage ResNet that simply drops the nonlinearities at the last residual representation (see Figure 2.2). We prove that sufficiently wide ResNEsts with bottleneck residual blocks under practical assumptions can always guarantee a desirable training property that ResNets with bottleneck residual blocks empirically achieve (but theoretically difficult to prove), i.e., adding more blocks does not decrease performance given the same arbitrarily selected basis. To be more specific, any local minimum obtained from ResNEsts has an improved representation guarantee under practical assumptions (see Remark 2 (a) and Corollary 1). Our results apply to loss functions that are differentiable and convex; and do not rely on any assumptions regarding datasets, or convexity/differentiability of the residual functions.

Basic vs. bottleneck

In the original ResNet paper, He et al. [2016a] empirically pointed out that ResNets with basic residual blocks indeed gain accuracy from increased depth, but are not as economical as the ResNets with bottleneck residual blocks (see Figure 1 in [Zagoruyko and Komodakis, 2016] for different block types). Our Theorem 4 supports such empirical findings.

Generalized and analyzable DNN models

ResNEsts are more general than the models considered in [Hardt and Ma, 2017, Shamir, 2018, Kawaguchi and Bengio, 2019, Yun et al., 2019] due to the removal of their simplified ResNet settings. In addition, the ResNEst modifies the input by *an expansion layer* that expands the input space. Such an expansion turns out to be crucial in deriving theoretical guarantees for improved residual representations. We find that the importance on expanding the input space in standard ResNets with bottleneck residual blocks has not been well recognized in existing theoretical results in the literature.

Restricted basis function models

We reveal a linear relationship between the output of the ResNEst and the input feature as well as the feature vector going into the last affine layer in each of residual functions. By treating each of feature vectors as a basis element, we find that ResNEsts are basis function models handicapped by *a coupling problem* in basis learning and linear prediction that can limit performance.

Augmented ResNEsts

As shown in Figure 2.1, we present a special architecture called *augmented ResNEst* or *A-ResNEst* that introduces a new weight matrix on each of feature vectors to solve the coupling problem that exists in ResNEsts. Due to such a decoupling, every local minimum obtained from an A-ResNEst bounds the empirical risk of the associated ResNEst from below. A-ResNEsts also directly enable us to see how features are supposed to be learned. It is necessary for features to be *linearly unpredictable* if residual representations are strictly improved over blocks.

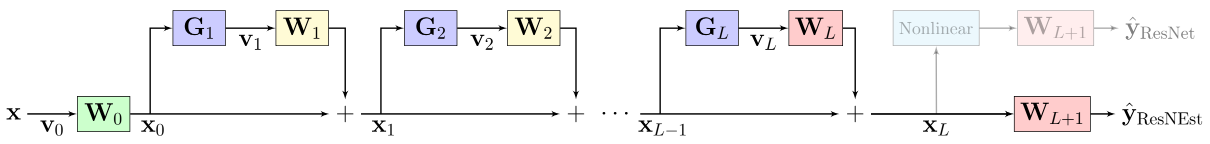


Figure 2.2. A generic vector-valued ResNEst that has a chain of L residual blocks (or units). Redrawing the standard ResNet block diagram in this viewpoint gives us considerable new insight. The symbol "+" represents the addition operation. Different from the ResNet architecture using pre-activation residual blocks in the literature [He et al., 2016b], our ResNEst architecture drops nonlinearities at $\mathbf{x}_L$ so as to reveal a linear relationship between the output $\hat{\mathbf{y}}_{\text{ResNEst}}$ and the features $\mathbf{v}_0, \mathbf{v}_1, \cdots, \mathbf{v}_L$. Empirical results of the ResNEst model are deferred to Appendix 2.7.2 in the supplementary material.

Wide ResNEsts with bottleneck residual blocks do not suffer from saddle points

At every saddle point obtained from a ResNEst, we show that there exists at least one direction with strictly negative curvature, under the same assumptions used in the improved representation guarantee, along with the specification of a squared loss and suitable assumptions on the last feature and dataset.

Improved representation guarantees for DenseNEsts

Although DenseNets [Huang et al., 2017] have shown better empirical performance than ResNets, we are not aware of any theoretical support for DenseNets. We define a DenseNEst (see Figure 2.4) as a simplified DenseNet model that only utilizes the dense connectivity of the DenseNet model, i.e., direct connections from every stack of layers to all subsequent stacks of layers. We show that any DenseNEst can be represented as a wide ResNEst with bottleneck residual blocks equipped with orthogonalities. Unlike ResNEsts, any DenseNEst exhibits the desirable property, i.e., adding more dense blocks does not decrease performance, without any special architectural re-design. Compared to A-ResNEsts, the way the features are generated in DenseNEsts makes linear predictability even more unlikely, suggesting better feature construction.

2.2 ResNEsts and augmented ResNEsts

In this section, we describe the proposed DNN models. These models and their new insights are preliminaries to our main results in Section 2.3. Section 2.2.1 recognizes the importance of the expansion layer and defines the ResNEst model. Section 2.2.2 points out the basis function modeling interpretation and the coupling problem in ResNEsts, and shows that the optimization on the set of prediction weights is non-convex. Section 2.2.3 proposes the A-ResNEst to avoid the coupling problem and shows that the minimum empirical risk obtained from a ResNEst is bounded from below by the corresponding A-ResNEst. Section 2.2.4 shows that linearly unpredictable features are necessary for strictly improved residual representations in A-ResNEsts.

2.2.1 Dropping nonlinearities in the final representation and expanding the input space

The importance on expanding the input space via $\mathbf{W}_0$ (see Figure 2.2) in standard ResNets has not been well recognized in recent theoretical results [Shamir, 2018, Kawaguchi and Bengio, 2019, Yun et al., 2019] although standard ResNets always have an expansion implemented by the first layer before the first residual block. Empirical results have even shown that a standard 16-layer wide ResNet outperforms a standard 1001-layer ResNet [Zagoruyko and Komodakis, 2016], which implies the importance of *a wide expansion of the input space*.

We consider the proposed ResNEst model shown in Figure 2.2 whose i-th residual block employs the following input-output relationship:

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \mathbf{W}_i \mathbf{G}_i \left(\mathbf{x}_{i-1}; \boldsymbol{\theta}_i\right) \tag{2.1}$$

for $i = 1, 2, \cdots, L$. The term excluded the first term $\mathbf{x}_{i-1}$ on the right-hand side is a composition of a nonlinear function $\mathbf{G}_i$ and a linear transformation,[1] which is generally known as a residual function. $\mathbf{W}_i \in \mathbb{R}^{M \times K_i}$ forms a linear transformation and we consider $\mathbf{G}_i(\mathbf{x}_{i-1}; \boldsymbol{\theta}_i) : \mathbb{R}^M \mapsto \mathbb{R}^{K_i}$ as a function implemented by a neural network with parameters $\boldsymbol{\theta}_i$ for all $i \in \{1, 2, \cdots, L\}$. We define the expansion $\mathbf{x}_0 = \mathbf{W}_0 \mathbf{x}$ for the input $\mathbf{x} \in \mathbb{R}^{N_{in}}$ to the ResNEst using a linear transformation with a weight matrix $\mathbf{W}_0 \in \mathbb{R}^{M \times K_0}$. The output $\hat{\mathbf{y}}_{\text{ResNEst}} \in \mathbb{R}^{N_o}$ (or $\hat{\mathbf{y}}_{L\text{-ResNEst}}$ to indicate L blocks) of the ResNEst is defined as $\hat{\mathbf{y}}_{L\text{-ResNEst}}(\mathbf{x}) = \mathbf{W}_{L+1} \mathbf{x}_L$ where $\mathbf{W}_{L+1} \in \mathbb{R}^{N_o \times M}$. M is the expansion factor and N_o is the output dimension of the network. The number of blocks L is a nonnegative integer. When $L = 0$, the ResNEst is a two-layer linear network $\hat{\mathbf{y}}_{0\text{-ResNEst}}(\mathbf{x}) = \mathbf{W}_1 \mathbf{W}_0 \mathbf{x}$.

Notice that the ResNEst we consider in this paper (Figure 2.2) is more general than the models in [Hardt and Ma, 2017, Shamir, 2018, Kawaguchi and Bengio, 2019, Yun et al., 2019] because our residual space $\mathbb{R}^M$ (the space where the addition is performed at the end of each residual block) is not constrained by the input dimension due to the expansion we define. Intuitively, a wider expansion (larger M) is required for a ResNEst that has more residual blocks. This is because the information collected in the residual representation grows after each block, and the fixed dimension M of the residual representation must be sufficiently large to avoid any loss of information. It turns out a wider expansion in a ResNEst is crucial in deriving performance guarantees because it assures the quality of local minima and saddle points (see Theorem 4 and 5).

2.2.2 Basis function modeling and the coupling problem

The conventional input-output relationship of a standard ResNet is not often easy to interpret. We find that redrawing the standard ResNet block diagram [He et al., 2016a,b] with a different viewpoint, shown in Figure 2.2, can give us considerable new insight. As shown in

[1]For any affine function $\mathbf{y}(\mathbf{x}_{\text{raw}}) = \mathbf{A}_{\text{raw}} \mathbf{x}_{\text{raw}} + \mathbf{b}$, if desired, one can use $\mathbf{y}(\mathbf{x}) = \begin{bmatrix} \mathbf{A}_{\text{raw}} & \mathbf{b} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\text{raw}} \\ 1 \end{bmatrix} = \mathbf{A}\mathbf{x}$ where $\mathbf{A} = \begin{bmatrix} \mathbf{A}_{\text{raw}} & \mathbf{b} \end{bmatrix}$ and $\mathbf{x} = \begin{bmatrix} \mathbf{x}_{\text{raw}} \\ 1 \end{bmatrix}$ and discuss on the linear function instead. All the results derived in this paper hold true regardless of the existence of bias parameters.

Figure 2.2, the ResNEst now reveals a linear relationship between the output and the features. With this observation, we can write down a useful input-output relationship for the ResNEst:

$$\hat{\mathbf{y}}_{L\text{-ResNEst}}(\mathbf{x}) = \mathbf{W}_{L+1} \sum_{i=0}^{L} \mathbf{W}_i \mathbf{v}_i(\mathbf{x}) \tag{2.2}$$

where $\mathbf{v}_i(\mathbf{x}) = \mathbf{G}_i(\mathbf{x}_{i-1}; \boldsymbol{\theta}_i) = \mathbf{G}_i\left(\sum_{j=0}^{i-1} \mathbf{W}_j \mathbf{v}_j; \boldsymbol{\theta}_i\right)$ for $i = 1, 2, \cdots, L$. Note that we do not impose any requirements for each $\mathbf{G}_i$ other than assuming that it is implemented by a neural network with a set of parameters $\boldsymbol{\theta}_i$. We define $\mathbf{v}_0 = \mathbf{v}_0(\mathbf{x}) = \mathbf{x}$ as the linear feature and regard $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_L$ as nonlinear features of the input $\mathbf{x}$, since $\mathbf{G}_i$ is in general nonlinear. The benefit of our formulation (2.2) is that the output of a ResNEst $\hat{\mathbf{y}}_{L\text{-ResNEst}}$ now can be viewed as a linear function of all these features. Our point of view of ResNEsts in (2.2) may be useful to explain the finding that ResNets are ensembles of relatively shallow networks [Veit et al., 2016].

As opposed to traditional nonlinear methods such as basis function modeling (chapter 3 in the book by Bishop, 2006) where a linear function is often trained on a set of handcrafted features, the ResNEst jointly finds features and a linear predictor function by solving the empirical risk minimization (ERM) problem denoted as (P) on $(\mathbf{W}_0, \cdots, \mathbf{W}_{L+1}, \boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_L)$. We denote $\mathscr{R}$ as the empirical risk (will be used later on). Indeed, one can view training a ResNEst as *a basis function modeling with a trainable (data-driven) basis* by treating each of features as a basis vector (it is reasonable to assume all features are not linearly predictable, see Section 2.2.4). However, unlike a basis function modeling, the linear predictor function in the ResNEst is not entirely independent of the basis generation process. We call such a phenomenon as *a coupling problem* which can handicap the performance of ResNEsts. To see this, note that feature (basis) vectors $\mathbf{v}_{i+1}, \cdots, \mathbf{v}_L$ can be different if $\mathbf{W}_i$ is changed (the product $\mathbf{W}_{L+1}\mathbf{W}_i\mathbf{v}_i$ is the linear predictor function for the feature $\mathbf{v}_i$). Therefore, the set of parameters $\boldsymbol{\phi} = \{\mathbf{W}_{i-1}, \boldsymbol{\theta}_i\}_{i=1}^{L}$ needs to be fixed to sufficiently guarantee that the basis is not changed with different linear predictor functions. It follows that $\mathbf{W}_{L+1}$ and $\mathbf{W}_L$ are the only weights which can be adjusted without changing the features. We refer to $\mathbf{W}_L$ and $\mathbf{W}_{L+1}$ as prediction weights and $\boldsymbol{\phi} = \{\mathbf{W}_{i-1}, \boldsymbol{\theta}_i\}_{i=1}^{L}$

as feature finding weights in the ResNEst. Obviously, the set of all the weights in the ResNEst is composed of the feature finding weights and prediction weights.

Because $\mathbf{G}_i$ is quite general in the ResNEst, any direct characterization on the landscape of ERM problem seems intractable. Thus, we propose to utilize the basis function modeling point of view in the ResNEst and analyze the following ERM problem:

$$(\mathrm{P}_{\boldsymbol{\phi}}) \min_{\mathbf{W}_L, \mathbf{W}_{L+1}} \mathscr{R}\left(\mathbf{W}_L, \mathbf{W}_{L+1}; \boldsymbol{\phi}\right) \tag{2.3}$$

where $\mathscr{R}\left(\mathbf{W}_L, \mathbf{W}_{L+1}; \boldsymbol{\phi}\right) = \frac{1}{N}\sum_{n=1}^{N} \ell\left(\hat{\mathbf{y}}^{\boldsymbol{\phi}}_{L\text{-ResNEst}}\left(\mathbf{x}^n\right), \mathbf{y}^n\right)$ for any fixed feature finding weights $\boldsymbol{\phi}$. We have used ℓ and $\{(\mathbf{x}^n, \mathbf{y}^n)\}_{n=1}^{N}$ to denote the loss function and training data, respectively. $\hat{\mathbf{y}}^{\boldsymbol{\phi}}_{L\text{-ResNEst}}$ denotes a ResNEst using a fixed feature finding weights $\boldsymbol{\phi}$. Although $(\mathrm{P}_{\boldsymbol{\phi}})$ has less optimization variables and looks easier than (P), Proposition 2 shows that it is a non-convex problem. Remark 1 explains why understanding $(\mathrm{P}_{\boldsymbol{\phi}})$ is valuable.

Remark 1. *Let the set of all local minimizers of $(P_{\boldsymbol{\phi}})$ using any possible features equip with the corresponding $\boldsymbol{\phi}$. Then, this set is a superset of the set of all local minimizers of the original ERM problem (P). Any characterization of $(P_{\boldsymbol{\phi}})$ can then be translated to (P) (see Corollary 2 for example).*

Assumption 2. $\sum_{n=1}^{N} \mathbf{v}_L\left(\mathbf{x}^n\right) \mathbf{y}^{nT} \neq \mathbf{0}$ *and* $\sum_{n=1}^{N} \mathbf{v}_L\left(\mathbf{x}^n\right) \mathbf{v}_L\left(\mathbf{x}^n\right)^T$ *is full rank.*

Proposition 2. *If ℓ is the squared loss and Assumption 2 is satisfied, then (a) the objective function of $(P_{\boldsymbol{\phi}})$ is non-convex and non-concave; (b) every critical point that is not a local minimizer is a saddle point in $(P_{\boldsymbol{\phi}})$.*

The proof of Proposition 2 is deferred to Appendix 2.7.1 in the supplementary material. Due to the product $\mathbf{W}_{L+1}\mathbf{W}_L$ in $\mathscr{R}\left(\mathbf{W}_L, \mathbf{W}_{L+1}; \boldsymbol{\phi}\right)$, our Assumption 2 is similar to one of the important data assumptions used in deep linear networks [Baldi and Hornik, 1989, Kawaguchi, 2016]. Assumption 2 is easy to be satisfied as we can always perturb $\boldsymbol{\phi}$ if the last nonlinear feature and dataset do not fit the assumption. Although Proposition 2 (a) examines the non-convexity for a fixed $\boldsymbol{\phi}$, the result can be extended to the original ERM problem (P) for the ResNEst. That is, if there exists at least one $\boldsymbol{\phi}$ such that Assumption 2 is satisfied, then the objective function for the optimization problem (P) is also non-convex and non-concave because there exists at least one point in the domain at which the Hessian is indefinite. As a result, this non-convex loss landscape in (P) immediately raises issues about suboptimal local minima in the loss landscape. This leads to an important question: Can we guarantee the quality of local minima with respect to some reference models that are known to be good enough?

2.2.3 Finding reference models: bounding empirical risks via augmentation

To avoid the coupling problem in ResNEsts, we propose a new architecture in Figure 2.1 called augmented ResNEst or A-ResNEst. An L-block A-ResNEst introduces another set of parameters $\{\mathbf{H}_i\}_{i=0}^{L}$ to replace every bilinear map on each feature in (2.2) with a linear map:

$$\hat{\mathbf{y}}_{L\text{-A-ResNEst}}(\mathbf{x}) = \sum_{i=0}^{L} \mathbf{H}_i \mathbf{v}_i(\mathbf{x}) . \tag{2.4}$$

Now, the function $\hat{\mathbf{y}}_{L\text{-A-ResNEst}}$ is linear with respect to all the prediction weights $\{\mathbf{H}_i\}_{i=0}^{L}$. Note that the parameters $\{\mathbf{W}_i\}_{i=0}^{L-1}$ still exist and are now dedicated to feature finding. On the other hand, $\mathbf{W}_L$ and $\mathbf{W}_{L+1}$ are deleted since they are not used in the A-ResNEst. As a result, the corresponding ERM problem (PA) is defined on $(\mathbf{H}_0, \cdots, \mathbf{H}_L, \boldsymbol{\phi})$. We denote $\mathscr{A}$ as the empirical risk in A-ResNEsts. The prediction weights are now different from the ResNEst as the A-ResNEst uses $\{\mathbf{H}_i\}_{i=0}^{L}$. Because any A-ResNEst prevents the coupling problem, it exhibits a nice property shown below.

Assumption 3. *The loss function $\ell(\hat{\mathbf{y}}, \mathbf{y})$ is differentiable and convex in $\hat{\mathbf{y}}$ for any $\mathbf{y}$.*

Proposition 3. *Let $\left(\mathbf{H}_0^*, \cdots, \mathbf{H}_L^*\right)$ be any local minimizer of the following optimization problem:*

$$(PA_{\boldsymbol{\phi}}) \min_{\mathbf{H}_0, \cdots, \mathbf{H}_L} \mathscr{A}(\mathbf{H}_0, \cdots, \mathbf{H}_L; \boldsymbol{\phi}) \tag{2.5}$$

where $\mathscr{A}(\mathbf{H}_0,\cdots,\mathbf{H}_L;\boldsymbol{\phi}) = \frac{1}{N}\sum_{n=1}^{N}\ell\left(\hat{\mathbf{y}}^{\boldsymbol{\phi}}_{L\text{-A-ResNEst}}(\mathbf{x}^n),\mathbf{y}^n\right)$*. If Assumption 3 is satisfied, then the optimization problem in (2.5) is convex and*

$$\varepsilon\left(\mathbf{W}_L^*,\mathbf{W}_{L+1}^*;\boldsymbol{\phi}\right) = \mathscr{R}\left(\mathbf{W}_L^*,\mathbf{W}_{L+1}^*;\boldsymbol{\phi}\right) - \mathscr{A}\left(\mathbf{H}_0^*,\cdots,\mathbf{H}_L^*;\boldsymbol{\phi}\right) \geq 0 \tag{2.6}$$

for any local minimizer $\left(\mathbf{W}_L^*,\mathbf{W}_{L+1}^*\right)$ *of* ($P_{\boldsymbol{\phi}}$) *using arbitrary feature finding parameters* $\boldsymbol{\phi}$.

The proof of Proposition 3 is deferred to Appendix 2.7.1 in the supplementary material. According to Proposition 3, A-ResNEst establishes empirical risk lower bounds (ERLBs) for a ResNEst. Hence, for the same $\boldsymbol{\phi}$ picked arbitrarily, an A-ResNEst is better than a ResNEst in terms of any pair of two local minima in their loss landscapes. Assumption 3 is practical because it is satisfied for two commonly used loss functions in regression and classification, i.e., the squared loss and cross-entropy loss. Other losses such as the logistic loss and smoothed hinge loss also satisfy this assumption.

2.2.4 Necessary condition for strictly improved residual representations

What properties are fundamentally required for features to be good, i.e., able to strictly improve the residual representation over blocks? With A-ResNEsts, we are able to straightforwardly answer this question. A fundamental answer is they need to be at least *linearly unpredictable*. Note that $\mathbf{v}_i$ must be linearly unpredictable by $\mathbf{v}_0,\cdots,\mathbf{v}_{i-1}$ if

$$\mathscr{A}\left(\mathbf{H}_0^*,\mathbf{H}_1^*,\cdots,\mathbf{H}_{i-1}^*,\mathbf{0},\cdots,\mathbf{0},\boldsymbol{\phi}^*\right) > \mathscr{A}\left(\mathbf{H}_0^*,\mathbf{H}_1^*,\cdots,\mathbf{H}_i^*,\mathbf{0},\cdots,\mathbf{0},\boldsymbol{\phi}^*\right) \tag{2.7}$$

for any local minimum $\left(\mathbf{H}_0^*,\cdots,\mathbf{H}_L^*,\boldsymbol{\phi}^*\right)$ in (PA). In other words, the residual representation $\mathbf{x}_i$ is not strictly improved from the previous representation $\mathbf{x}_{i-1}$ if the feature $\mathbf{v}_i$ is linearly predictable by the previous features. Fortunately, the linearly unpredictability of $\mathbf{v}_i$ is usually satisfied when $\mathbf{G}_i$ is nonlinear; and the set of features can be viewed as a basis function. This viewpoint also suggests avenues for improving feature construction through imposition of various constraints.

By Proposition 3, the relation in (2.7) always holds with equality, i.e., the residual representation $\mathbf{x}_i$ is guaranteed to be always no worse than the previous one $\mathbf{x}_{i-1}$ at any local minimizer obtained from an A-ResNEst.

2.3 Wide ResNEsts with bottleneck residual blocks always attain ERLBs

Assumption 4. *$M \geq N_o$.*

Assumption 5. *The linear inverse problem $\mathbf{x}_{L-1} = \sum_{i=0}^{L-1} \mathbf{W}_i \mathbf{v}_i$ has a unique solution.*

Theorem 4. *If Assumption 3 and 4 are satisfied, then the following two properties are true in ($P_{\boldsymbol{\phi}}$) under any $\boldsymbol{\phi}$ such that Assumption 5 holds: (a) every critical point with full rank $\mathbf{W}_{L+1}$ is a global minimizer; (b) $\varepsilon = 0$ for every local minimizer.*

The proof of Theorem 4 is deferred to Appendix 2.7.1 in the supplementary material. Theorem 4 (a) provides a sufficient condition for a critical point to be a global minimum of ($\mathrm{P}_{\boldsymbol{\phi}}$). Theorem 4 (b) gives an affirmative answer for every local minimum in ($\mathrm{P}_{\boldsymbol{\phi}}$) to attain the ERLB. To be more specific, any pair of obtained local minima from the ResNEst and the A-ResNEst using the same arbitrary $\boldsymbol{\phi}$ are equally good. In addition, the implication of Theorem 4 (b) is that every local minimum of ($\mathrm{P}_{\boldsymbol{\phi}}$) is also a global minimum despite its non-convex landscape (Proposition 2), which suggests there exists no suboptimal local minimum for the optimization problem ($\mathrm{P}_{\boldsymbol{\phi}}$). One can also establish the same results for local minimizers of (P) under the same set of assumptions by replacing "($\mathrm{P}_{\boldsymbol{\phi}}$) under any $\boldsymbol{\phi}$" with just "(P)" in Theorem 4. Such a modification may gain more clarity, but is more restricted than the original statement due to Remark 1. Note that Theorem 4 is not limited to fixing any weights during training; and it applies to both normal training (train all the weights in a network as a whole) and blockwise or layerwise training procedures.

2.3.1 Improved representation guarantees

By Remark 1 and Theorem 4 (b), we can then establish the following representational guarantee.

Remark 2. *Let Assumption 3 and 4 be true. Any local minimizer of (P) such that Assumption 5 is satisfied guarantees (a) monotonically improved (no worse) residual representations over blocks; (b) every residual representation is better than the input representation in the linear prediction sense.*

Although there may exist suboptimal local minima in the optimization problem (P), Remark 2 suggests that such minima still improve residual representations over blocks under practical conditions. Mathematically, Remark 2 (a) and Remark 2 (b) are described by Corollary 1 and the general version of Corollary 2, respectively. Corollary 1 compares the minimum empirical risk obtained at any two representations among $\mathbf{x}_1$ to $\mathbf{x}_L$ for any given network satisfying the assumptions; and Corollary 2 extends this comparison to the input representation.

Corollary 1. *Let Assumption 3 and 4 be true. Any local minimum of ($P_{\boldsymbol{\alpha}}$) is smaller than or equal to any local minimum of ($P_{\boldsymbol{\beta}}$) under Assumption 5 for any $\boldsymbol{\alpha} = \{\mathbf{W}_{i-1}, \boldsymbol{\theta}_i\}_{i=1}^{L_\alpha}$ and $\boldsymbol{\beta} = \{\mathbf{W}_{i-1}, \boldsymbol{\theta}_i\}_{i=1}^{L_\beta}$ where L_α and L_β are positive integers such that $L_\alpha > L_\beta$.*

The proof of Corollary 1 is deferred to Appendix 2.7.1 in the supplementary material. Because Corollary 1 holds true for any properly given weights, one can apply Corollary 1 to proper local minimizers of (P). Corollary 2 ensures that ResNEsts are guaranteed to be no worse than the best linear predictor under practical assumptions. This property is useful because linear estimators are widely used in signal processing applications and they can now be confidently replaced with ResNEsts.

Corollary 2. *Let* $\left(\mathbf{W}_0^*, \cdots, \mathbf{W}_{L+1}^*, \boldsymbol{\theta}_1^*, \cdots, \boldsymbol{\theta}_L^*\right)$ *be any local minimizer of (P) and*

$$\boldsymbol{\phi}^* = \{\mathbf{W}_{i-1}^*, \boldsymbol{\theta}_i^*\}_{i=1}^{L}.$$

If Assumption 3, 4 and 5 are satisfied, then (a)

$$\mathscr{R}\left(\mathbf{W}_0^*,\cdots,\mathbf{W}_{L+1}^*,\boldsymbol{\theta}_1^*,\cdots,\boldsymbol{\theta}_L^*\right)\leq\min_{\mathbf{A}\in\mathbb{R}^{N_o\times N_{in}}}\frac{1}{N}\sum_{n=1}^{N}\ell\left(\mathbf{A}\mathbf{x}^n,\mathbf{y}^n\right);$$

(b) the above inequality is strict if $\mathscr{A}\left(\mathbf{H}_0^*,\mathbf{0},\cdots,\mathbf{0},\boldsymbol{\phi}^*\right)>\mathscr{A}\left(\mathbf{H}_0^*,\cdots,\mathbf{H}_L^*,\boldsymbol{\phi}^*\right)$.

The proof of Corollary 2 is deferred to Appendix 2.7.1 in the supplementary material. To the best of our knowledge, Corollary 2 is the first theoretical guarantee for vector-valued ResNet-like models that have arbitrary residual blocks to outperform any linear predictors. Corollary 2 is more general than the results in [Shamir, 2018, Kawaguchi and Bengio, 2019, Yun et al., 2019] because it is not limited to assumptions like scalar-valued output or single residual block. In fact, we can have a even more general statement because any local minimum obtained from (P$_{\boldsymbol{\phi}}$) with random or any $\boldsymbol{\phi}$ is better than the minimum empirical risk provided by the best linear predictor, under the same assumptions used in Corollary 2. This general version fully describes Remark 2 (b).

Theorem 4, Corollary 1 and Corollary 2 are quite general because they are not limited to specific loss functions, residual functions, or datasets. Note that we do not impose any assumptions such as differentiability or convexity on the neural network $\mathbf{G}_i$ for $i=1,2,\cdots,L$ in residual functions. Assumption 4 is practical because the expansion factor M is usually larger than the input dimension N_{in}; and the output dimension N_o is usually not larger than the input dimension for most supervised learning tasks using sensory input. Assumption 5 states that the features need to be uniquely invertible from the residual representation. Although such an assumption requires a special architectural design, we find that it is always satisfied empirically after random initialization or training when the "bottleneck condition" is satisfied.

2.3.2 How to design architectures with representational guarantees?

Notice that one must be careful with the ResNEst architectural design so as to enjoy Theorem 4, Corollary 1 and Corollary 2. A ResNEst needs to be wide enough such that

$M \geq \sum_{i=0}^{L-1} K_i$ to necessarily satisfy Assumption 5. We call such a sufficient condition on the width and feature dimensionalities as a *bottleneck condition*. Because each nonlinear feature size K_i for $i < L$ (say $L > 1$) must be smaller than the dimensionality of the residual representation M, each of these residual functions is a *bottleneck design* [He et al., 2016a,b, Zagoruyko and Komodakis, 2016] forming a bottleneck residual block. We now explicitly see the importance of the expansion layer. Without the expansion, the dimenionality of the residual representation is limited to the input dimension. As a result, Assumption 5 cannot be satisfied for $L > 1$; and the analysis for the ResNEst with multiple residual blocks remains intractable or requires additional assumptions on residual functions.

Loosely speaking, a sufficiently wide expansion or satisfaction of the bottleneck condition implies Assumption 5. If the bottleneck condition is satisfied, then ResNEsts are equivalent to A-ResNEsts for a given $\boldsymbol{\phi}$, i.e., $\varepsilon = 0$. If not (e.g., basic blocks are used in a ResNEst), then a ResNEst can have a problem of diminishing feature reuse or end up with poor performance even though it has excellent features that can be fully exploited by an A-ResNEst to yield better performance, i.e., $\varepsilon > 0$. From such a viewpoint, Theorem 4 supports the empirical findings in [He et al., 2016a] that bottleneck blocks are more economical than basic blocks. Our results thus recommend A-ResNEsts over ResNEsts if the bottleneck condition cannot be satisfied.

2.3.3 Guarantees on saddle points

In addition to guarantees for the quality of local minima, we find that ResNEsts can easily escape from saddle points due to the nice property shown below.

Theorem 5. *If ℓ is the squared loss, and Assumption 2 and 4 are satisfied, then the following two properties are true at every saddle point of $(P_{\boldsymbol{\phi}})$ under any $\boldsymbol{\phi}$ such that Assumption 5 holds: (a) $\mathbf{W}_{L+1}$ is rank-deficient; (b) there exists at least one direction with strictly negative curvature.*

The proof of Theorem 5 is deferred to Appendix 2.7.1 in the supplementary material. In contrast to Theorem 4 (a), Theorem 5 (a) provides a necessary condition for a saddle point.

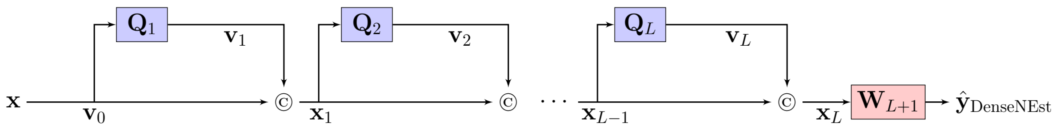


Figure 2.3. A generic vector-valued DenseNEst that has a chain of L dense blocks (or units). The symbol "©" represents the concatenation operation. We intentionally draw a DenseNEst in such a form to emphasize its relationship to a ResNEst (see Proposition 5).

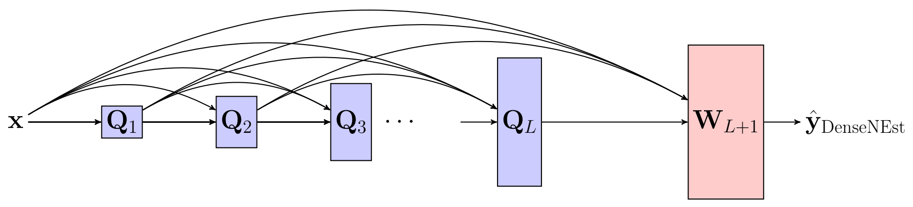


Figure 2.4. An equivalence to Figure 2.3 emphasizing the growth of the input dimension at each block.

Although ($\mathrm{P}_{\boldsymbol{\phi}}$) is a non-convex optimization problem according to Proposition 2 (a), Theorem 5 (b) suggests a desirable property for saddle points in the loss landscape. Because there exists at least one direction with strictly negative curvature at every saddle point that satisfies the bottleneck condition, the second-order optimization methods can rapidly escape from saddle points [Dauphin et al., 2014]. If the first-order methods are used, the randomness in stochastic gradient helps the first-order methods to escape from the saddle points [Ge et al., 2015]. Again, we require the bottleneck condition to be satisfied in order to guarantee such a nice property about saddle points. Note that Theorem 5 is not limited to fixing any weights during training; and it applies to both normal training and blockwise training procedures due to Remark 1.

2.4 DenseNEsts are wide ResNEsts with bottleneck residual blocks equipped with orthogonalities

Instead of adding one nonlinear feature in each block and remaining in same space $\mathbb{R}^M$, the DenseNEst model shown in Figure 2.3 preserves each of features in their own subspaces by a sequential concatenation at each block. For an L-block DenseNEst, we define the i-th dense block as a function $\mathbb{R}^{M_{i-1}} \mapsto \mathbb{R}^{M_i}$ of the form

$$\mathbf{x}_i = \mathbf{x}_{i-1} © \mathbf{Q}_i\left(\mathbf{x}_{i-1};\boldsymbol{\theta}_i\right) \tag{2.8}$$

for $i = 1,2,\cdots,L$ where the dense function $\mathbf{Q}_i$ is a general nonlinear function; and $\mathbf{x}_i$ is the output of the i-th dense block. The symbol © concatenates vector $\mathbf{x}_{i-1}$ and vector $\mathbf{Q}_i\left(\mathbf{x}_{i-1};\boldsymbol{\theta}_i\right)$ and produces a higher-dimensional vector $\begin{bmatrix}\mathbf{x}_{i-1}^T & \mathbf{Q}_i\left(\mathbf{x}_{i-1};\boldsymbol{\theta}_i\right)^T\end{bmatrix}^T$. We define $\mathbf{x}_0 = \mathbf{x}$ where $\mathbf{x} \in \mathbb{R}^{N_{in}}$ is the input to the DenseNEst. For all $i \in \{1,2,\cdots,L\}$, $\mathbf{Q}_i(\mathbf{x}_{i-1};\boldsymbol{\theta}_i) : \mathbb{R}^{M_{i-1}} \mapsto \mathbb{R}^{D_i}$ is a function implemented by a neural network with parameters $\boldsymbol{\theta}_i$ where $D_i = M_i - M_{i-1} \geq 1$ with $M_0 = N_{in} = D_0$. The output of a DenseNEst is defined as $\hat{\mathbf{y}}_{\text{DenseNEst}} = \mathbf{W}_{L+1}\mathbf{x}_L$ for $\mathbf{W}_{L+1} \in \mathbb{R}^{N_o \times M_L}$, which can be written as

$$\mathbf{W}_{L+1}\left(\mathbf{x}_0 © \mathbf{Q}_1\left(\mathbf{x}_0;\boldsymbol{\theta}_1\right) © \cdots © \mathbf{Q}_L\left(\mathbf{x}_{L-1};\boldsymbol{\theta}_L\right)\right) = \sum_{i=0}^{L} \mathbf{W}_{L+1,i}\mathbf{v}_i\left(\mathbf{x}\right) \tag{2.9}$$

where $\mathbf{v}_i\left(\mathbf{x}\right) = \mathbf{Q}_i(\mathbf{x}_{i-1};\boldsymbol{\theta}_i) = \mathbf{Q}_i(\mathbf{x}_0 © \mathbf{v}_1 © \mathbf{v}_2 © \cdots © \mathbf{v}_{i-1};\boldsymbol{\theta}_i)$ for $i = 1,2,\cdots,L$ are regarded as nonlinear features of the input $\mathbf{x}$. We define $\mathbf{v}_0 = \mathbf{x}$ as the linear feature.

$$\mathbf{W}_{L+1} = \begin{bmatrix}\mathbf{W}_{L+1,0} & \mathbf{W}_{L+1,1} & \cdots & \mathbf{W}_{L+1,L}\end{bmatrix}$$

is the prediction weight matrix in the DenseNEst as all the weights which are responsible for the prediction is in this single matrix from the viewpoint of basis function modeling. The ERM problem (PD) for the DenseNEst is defined on $(\mathbf{W}_{L+1},\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_L)$. To fix the features, the set of parameters $\boldsymbol{\phi} = \{\boldsymbol{\theta}_i\}_{i=1}^{L}$ needs to be fixed. Therefore, the DenseNEst ERM problem for any fixed features, denoted as (PD$_{\boldsymbol{\phi}}$), is fairly straightforward as it only requires to optimize over a single weight matrix, i.e.,

$$(\text{PD}_{\boldsymbol{\phi}}) \min_{\mathbf{W}_{L+1}} \mathscr{D}\left(\mathbf{W}_{L+1};\boldsymbol{\phi}\right) \tag{2.10}$$

where $\mathscr{D}(\mathbf{W}_{L+1};\boldsymbol{\phi}) = \frac{1}{N}\sum_{n=1}^{N}\ell\left(\hat{\mathbf{y}}^{\boldsymbol{\phi}}_{L\text{-DenseNEst}}(\mathbf{x}^n), \mathbf{y}^n\right)$. Unlike ResNEsts, there is no such coupling between the feature finding and linear prediction in DenseNEsts. Compared to ResNEsts or A-ResNEsts, the way the features are generated in DenseNEsts generally makes the linear predictability even more unlikely. To see that, note that the $\mathbf{Q}_i$ directly applies on the concatenation of all previous features; however, the $\mathbf{G}_i$ applies on the sum of all previous features.

Different from a ResNEst which requires Assumption 3, 4 and 5 to guarantee its superiority with respect to the best linear predictor (Corollary 2), the corresponding guarantee in a DenseNEst shown in Proposition 4 requires weaker assumptions.

Proposition 4. *If Assumption 3 is satisfied, then any local minimum of (PD) is smaller than or equal to the minimum empirical risk given by any linear predictor of the input.*

The proof of Proposition 4 is deferred to Appendix 2.7.1 in the supplementary material. Notice that no special architectural design in a DenseNEst is required to make sure it always outperforms the best linear predictor. Any DenseNEst is always better than any linear predictor when the loss function is differentiable and convex (Assumption 3). Such an advantage can be explained by the $\mathbf{W}_{L+1}$ in the DenseNEst. Because $\mathbf{W}_{L+1}$ is the only prediction weight matrix which is directly applied onto the concatenation of all the features, ($\text{PD}_{\boldsymbol{\phi}}$) is a convex optimization problem. We point out the difference of $\mathbf{W}_{L+1}$ between the ResNEst and DenseNEst. In the ResNEst, $\mathbf{W}_{L+1}$ needs to interpret the features from the residual representation; while the $\mathbf{W}_{L+1}$ in the DenseNEst directly accesses the features. That is why we require Assumption 5 in the ResNEst to eliminate any ambiguity on the feature interpretation.

Can a ResNEst and a DenseNEst be equivalent? Yes, Proposition 5 establishes a link between them.

Proposition 5. *Given any DenseNEst $\hat{\mathbf{y}}_{L\text{-DenseNEst}}$, there exists a wide ResNEst with bottleneck residual blocks $\hat{\mathbf{y}}^{\boldsymbol{\phi}}_{L\text{-ResNEst}}$ such that $\hat{\mathbf{y}}^{\boldsymbol{\phi}}_{L\text{-ResNEst}}(\mathbf{x}) = \hat{\mathbf{y}}_{L\text{-DenseNEst}}(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^{N_{in}}$. If, in addition, Assumption 3 and 4 are satisfied, then $\varepsilon = 0$ for every local minimizer of ($P_{\boldsymbol{\phi}}$).*

The proof of Proposition 5 is deferred to Appendix 2.7.1 in the supplementary material. Because the concatenation of two given vectors can be represented by an addition over two vectors projected onto a higher dimensional space with disjoint supports, one straightforward construction for an equivalent ResNEst is to sufficiently expand the input space and enforce the orthogonality of all the column vectors in $\mathbf{W}_0, \mathbf{W}_1, \cdots, \mathbf{W}_L$. As a result, any DenseNEst can be viewed as a ResNEst that always satisfies Assumption 5 and of course the bottleneck condition no matter how we train the DenseNEst or select its hyperparameters, leading to the desirable guarantee, i.e., any local minimum obtained in optimizing the prediction weights of the resulting ResNEst from any DenseNEst always attains the lower bound. Thus, DenseNEsts are certified as being advantageous over ResNEsts by Proposition 5. For example, a small M may be chosen and then the guarantee in Theorem 4 can no longer exist, i.e., $\varepsilon > 0$. However, the corresponding ResNEst induced by a DenseNEst always achieves $\varepsilon = 0$. Hence, Proposition 5 can be regarded as a theoretical support for why standard DenseNets [Huang et al., 2017] are in general better than standard ResNets [He et al., 2016b].

2.5 Related work

In this section, we discuss ResNet works that investigate on properties of local minima and give more details for our important references that appear in the introduction. We focus on highlighting their results and assumptions used so as to compare to our theoretical results derived from practical assumptions. The earliest theoretical work for ResNets can be dated back to [Hardt and Ma, 2017] which proved a vector-valued ResNet-like model using a linear residual function in each residual block has no spurious local minima (local minima that give larger objective values than the global minima) under squared loss and near-identity region assumptions. There are results [Li and Yuan, 2017, Liu et al., 2019] proved that stochastic gradient descent can converge to the global minimum in scalar-valued two-layer ResNet-like models; however, such a desirable property relies on strong assumptions including single residual block and Gaussian input distribution. Li et al. [2018] visualized the loss landscapes of a ResNet and its plain counterpart (without skip connections); and they showed that the skip connections promote flat minimizers and prevent the transition to chaotic behavior. Liang et al. [2018] showed that scalar-valued and single residual block ResNet-like models can have zero training error at all local minima by making strong assumptions in the data distribution and loss function for a binary classification problem. In stead of pursuing local minima are global in the empirical risk landscape using strong assumptions, Shamir [2018] first took a different route and proved that a scalar-valued ResNet-like model with a direct skip connection from input to output layer (single residual block) is better than any linear predictor under mild assumptions. To be more specific, he showed that every local minimum obtained in his model is no worse than the global minimum in any linear predictor under more generalized residual functions and no assumptions on the data distribution. He also pointed out that the analysis for the vector-valued case is nontrivial. Kawaguchi and Bengio [2019] overcame such a difficulty and proved that vector-valued models with single residual block is better than any linear predictor under weaker assumptions. Yun et al. [2019] extended the prior work by Shamir [2018] to multiple residual blocks. Although the model considered is closer to a standard ResNet compared to previous works, the model output is assumed to be scalar-valued. All above-mentioned works do not take the first layer that appears before the first residual block in standard ResNets into account. As a result, the dimensionality of the residual representation in their simplified ResNet models is constrained to be the same size as the input.

2.6 Broader impact

One of the mysteries in ResNets and DenseNets is that learning better DNN models seems to be as easy as stacking more blocks. In this paper, we define three generalized and analyzable DNN architectures, i.e., ResNEsts, A-ResNEsts, and DenseNEsts, to answer this question. Our results not only establish guarantees for monotonically improved representations over blocks, but also assure that all linear (affine) estimators can be replaced by our architectures without harming performance. We anticipate these models can be friendly options for researchers or engineers who value or mostly rely on linear estimators or performance guarantees in their problems. In fact, these models should yield much better performance as they can be viewed as basis function models with data-driven bases that guarantee to be always better than the best linear estimator. Our contributions advance the fundamental understanding of ResNets and DenseNets, and promote their use cases through a certificate of attractive guarantees.

2.7 Appendix

2.7.1 Proofs

Proof of Proposition 2

Proof. Let

$$\mathbf{V}_i = \begin{bmatrix} \mathbf{v}_i(\mathbf{x}^1) & \mathbf{v}_i(\mathbf{x}^2) & \cdots & \mathbf{v}_i(\mathbf{x}^N) \end{bmatrix} \tag{2.11}$$

for $i = 0, 1, \cdots, L$ and

$$\mathbf{\Delta} = \left(\mathbf{W}_{L+1} \sum_{i=0}^{L} \mathbf{W}_i \mathbf{V}_i - \mathbf{Y} \right)^T = \left(\hat{\mathbf{Y}} - \mathbf{Y} \right)^T = \begin{bmatrix} \boldsymbol{\delta}_1 & \boldsymbol{\delta}_2 & \cdots & \boldsymbol{\delta}_{N_o} \end{bmatrix} \tag{2.12}$$

where $\mathbf{Y} = \begin{bmatrix} \mathbf{y}^1 & \mathbf{y}^2 & \cdots & \mathbf{y}^N \end{bmatrix}$. The Hessian of $\mathscr{R}(\mathbf{W}_L, \mathbf{W}_{L+1}; \boldsymbol{\phi})$ in $(\mathrm{P}_{\boldsymbol{\phi}})$ is given by

$$\begin{aligned} \nabla^2 \mathscr{R} &= \begin{bmatrix} \frac{\partial^2 \mathscr{R}}{\partial \mathrm{vec}\left(\mathbf{W}_L^T\right)^2} & \frac{\partial^2 \mathscr{R}}{\partial \mathrm{vec}\left(\mathbf{W}_{L+1}^T\right) \partial \mathrm{vec}\left(\mathbf{W}_L^T\right)} \\ \frac{\partial^2 \mathscr{R}}{\partial \mathrm{vec}\left(\mathbf{W}_L^T\right) \partial \mathrm{vec}\left(\mathbf{W}_{L+1}^T\right)} & \frac{\partial^2 \mathscr{R}}{\partial \mathrm{vec}\left(\mathbf{W}_{L+1}^T\right)^2} \end{bmatrix} \\ &= \frac{2}{N} \begin{bmatrix} \mathbf{W}_{L+1}^T \mathbf{W}_{L+1} \otimes \mathbf{V}_L \mathbf{V}_L^T & \mathbf{W}_{L+1}^T \otimes \mathbf{V}_L \sum_{i=0}^{L} \mathbf{V}_i^T \mathbf{W}_i^T + \mathbf{E} \\ \mathbf{W}_{L+1} \otimes \sum_{i=0}^{L} \mathbf{W}_i \mathbf{V}_i \mathbf{V}_L^T + \mathbf{E}^T & \mathbf{I}_{N_o} \otimes \sum_{i=0}^{L} \mathbf{W}_i \mathbf{V}_i \left(\sum_{i=0}^{L} \mathbf{W}_i \mathbf{V}_i \right)^T \end{bmatrix} \end{aligned} \tag{2.13}$$

where

$$\mathbf{E} = \begin{bmatrix} \mathbf{I}_M \otimes \mathbf{V}_L \boldsymbol{\delta}_1 & \cdots & \mathbf{I}_M \otimes \mathbf{V}_L \boldsymbol{\delta}_{N_o} \end{bmatrix}. \tag{2.14}$$

We have used $\otimes$ to denote the Kronecker product. See Appendix 2.7.1 for the derivation of the Hessian. By the generalized Schur complement,

$$\nabla^2 \mathscr{R} \succeq \mathbf{0} \implies \text{range}\left(\frac{\partial^2 \mathscr{R}}{\partial \text{vec}\left(\mathbf{W}_{L+1}^T\right) \partial \text{vec}\left(\mathbf{W}_L^T\right)} \right) \subseteq \text{range}\left(\frac{\partial^2 \mathscr{R}}{\partial \text{vec}\left(\mathbf{W}_L^T\right)^2} \right) \tag{2.15}$$

which implies the projection of $\frac{\partial^2 \mathscr{R}}{\partial \text{vec}\left(\mathbf{W}_{L+1}^T\right) \partial \text{vec}\left(\mathbf{W}_L^T\right)}$ onto the range of $\frac{\partial^2 \mathscr{R}}{\partial \text{vec}\left(\mathbf{W}_L^T\right)^2}$ is itself. As a result,

$$\left(\mathbf{I}_{MK_L} - \frac{\partial^2 \mathscr{R}}{\partial^2 \text{vec}\left(\mathbf{W}_L^T\right)} \left(\frac{\partial^2 \mathscr{R}}{\partial^2 \text{vec}\left(\mathbf{W}_L^T\right)} \right)^{\dagger} \right) \frac{\partial^2 \mathscr{R}}{\partial \text{vec}\left(\mathbf{W}_{L+1}^T\right) \partial \text{vec}\left(\mathbf{W}_L^T\right)} = \mathbf{0} \tag{2.16}$$

where $\dagger$ denotes the Moore-Penrose pseudoinverse. Substituting the submatrices in (2.13) to the above equation, we obtain

$$\frac{2}{N} \begin{bmatrix} \left(\mathbf{I}_M - \mathbf{W}_{L+1}^T \mathbf{W}_{L+1} \left(\mathbf{W}_{L+1}^T \mathbf{W}_{L+1} \right)^{\dagger} \right)^T \otimes \boldsymbol{\delta}_1^T \mathbf{V}_L^T \\ \vdots \\ \left(\mathbf{I}_M - \mathbf{W}_{L+1}^T \mathbf{W}_{L+1} \left(\mathbf{W}_{L+1}^T \mathbf{W}_{L+1} \right)^{\dagger} \right)^T \otimes \boldsymbol{\delta}_{N_o}^T \mathbf{V}_L^T \end{bmatrix}^T = \mathbf{0} \tag{2.17}$$

which implies

$$\mathbf{W}_{L+1}^T \mathbf{W}_{L+1} \left(\mathbf{W}_{L+1}^T \mathbf{W}_{L+1} \right)^{\dagger} = \mathbf{I}_M \quad \text{or} \quad \mathbf{V}_L \boldsymbol{\Delta} = \mathbf{0}. \tag{2.18}$$

On the other hand, the above condition is also necessary for the Hessian to be negative semidefinite because $\nabla^2 \mathscr{R} \preceq \mathbf{0} \implies -\nabla^2 \mathscr{R} \succeq \mathbf{0}$ which implies (2.16).

Now, using the assumption $\sum_{n=1}^{N} \mathbf{v}_L\left(\mathbf{x}^n\right) \mathbf{y}^{nT} \neq \mathbf{0}$, notice that the condition in (2.18) is not satisfied for any point in the set

$$\mathscr{S} = \left\{ (\mathbf{W}_L, \mathbf{W}_{L+1}) \middle| \mathbf{W}_L \in \mathbb{R}^{M \times K_L}, \mathbf{W}_{L+1} = \mathbf{0} \right\}. \tag{2.19}$$

Hence, there exist some points in the domain at which the Hessian is indefinite. The objective function $\mathscr{R}(\mathbf{W}_L, \mathbf{W}_{L+1}; \boldsymbol{\phi})$ in (P$_{\boldsymbol{\phi}}$) is non-convex and non-concave. We have proved the statement (a).

By the generalized Schur complement and the assumption that $\sum_{n=1}^{N} \mathbf{v}_L(\mathbf{x}^n) \mathbf{v}_L(\mathbf{x}^n)^T$ is full rank, we have

$$\nabla^2 \mathscr{R} \preceq \mathbf{0} \implies \frac{\partial^2 \mathscr{R}}{\partial \mathrm{vec}\left(\mathbf{W}_L^T\right)^2} \preceq \mathbf{0} \implies \mathbf{W}_{L+1} = \mathbf{0} \tag{2.20}$$

where we have used the spectrum property of the Kronecker product and the positive definiteness of $\mathbf{V}_L \mathbf{V}_L^T$. Notice that this is a contradiction because any point with $\mathbf{W}_{L+1} = \mathbf{0}$ is in the set $\mathscr{S}$. Hence, there exists no point at which the Hessian is negative semidefinite. Because the negative semidefiniteness is a necessary condition for a local maximum, every critical point is then either a local minimum or a saddle point. We have proved the statement (b). □

Proof of Proposition 3

Proof. $\mathscr{A}(\mathbf{H}_0, \cdots, \mathbf{H}_L; \boldsymbol{\phi})$ is convex in $\begin{bmatrix} \mathbf{H}_0 & \mathbf{H}_1 & \cdots & \mathbf{H}_L \end{bmatrix}$ because it is a nonnegative weighted sum of convex functions composited with affine mappings. Thus, (PA$_{\boldsymbol{\phi}}$) is a convex optimization problem and $\left(\mathbf{H}_0^*, \cdots, \mathbf{H}_L^*\right)$ is the best linear fit using $\boldsymbol{\phi}$. That is, for any local minimizer $\left(\mathbf{H}_0^*, \cdots, \mathbf{H}_L^*\right)$, it is always true that

$$\frac{1}{N} \sum_{n=1}^{N} \ell\left(\sum_{i=0}^{L} \mathbf{H}_i^* \mathbf{v}_i(\mathbf{x}^n), \mathbf{y}^n \right) \leq \frac{1}{N} \sum_{n=1}^{N} \ell\left(\sum_{i=0}^{L} \mathbf{A}_i \mathbf{v}_i(\mathbf{x}^n), \mathbf{y}^n \right) \tag{2.21}$$

for arbitrary $\mathbf{A}_i \in \mathbb{R}^{N_o \times K_i}, i = 0, 1, \cdots, L$. □

Proof of Theorem 4

Proof. By the convexity in Proposition 3, every critical point in (PA$_{\boldsymbol{\phi}}$) is a global minimizer. Since the objective function of (PA$_{\boldsymbol{\phi}}$) is differentiable, the first-order derivative is a zero row vector at any critical point, i.e.,

$$\begin{aligned}
\frac{\partial \mathscr{A}}{\partial \mathrm{vec}\left(\mathbf{H}_i\right)} &= \frac{1}{N}\sum_{n=1}^{N} \left.\frac{\partial \ell\left(\hat{\mathbf{y}},\mathbf{y}^n\right)}{\partial \mathrm{vec}\left(\mathbf{H}_i\right)}\right|_{\hat{\mathbf{y}}=\sum_{i=0}^{L}\mathbf{H}_i\mathbf{v}_i(\mathbf{x}^n)} \\
&= \frac{1}{N}\sum_{n=1}^{N} \left.\frac{\partial \ell\left(\hat{\mathbf{y}},\mathbf{y}^n\right)}{\partial \hat{\mathbf{y}}}\frac{\partial \hat{\mathbf{y}}}{\partial \mathrm{vec}\left(\mathbf{H}_i\right)}\right|_{\hat{\mathbf{y}}=\sum_{i=0}^{L}\mathbf{H}_i\mathbf{v}_i(\mathbf{x}^n)} \\
&= \frac{1}{N}\sum_{n=1}^{N} \left.\frac{\partial \ell\left(\hat{\mathbf{y}},\mathbf{y}^n\right)}{\partial \hat{\mathbf{y}}}\right|_{\hat{\mathbf{y}}=\sum_{i=0}^{L}\mathbf{H}_i\mathbf{v}_i(\mathbf{x}^n)} \left(\mathbf{v}_i\left(\mathbf{x}^n\right)^T \otimes \mathbf{I}_{N_o}\right) \\
&= \frac{1}{N}\sum_{n=1}^{N} \left(\left(\mathbf{v}_i\left(\mathbf{x}^n\right)\otimes \mathbf{I}_{N_o}\right) \underbrace{\left.\frac{\partial \ell\left(\hat{\mathbf{y}},\mathbf{y}^n\right)^T}{\partial \hat{\mathbf{y}}}\right|_{\hat{\mathbf{y}}=\sum_{i=0}^{L}\mathbf{H}_i\mathbf{v}_i(\mathbf{x}^n)}}_{\mathbf{g}_a(\mathbf{x}^n)} \right)^T \qquad (2.22) \\
&= \frac{1}{N}\sum_{n=1}^{N} \mathrm{vec}\left(\mathbf{g}_a\left(\mathbf{x}^n\right)\mathbf{v}_i\left(\mathbf{x}^n\right)^T\right)^T \\
&= \mathbf{0}
\end{aligned}$$

for $i = 0, 1, \cdots, L$. Again, we have used $\otimes$ to denote the Kronecker product. According to (2.22), the point $\left(\mathbf{H}_0^*, \cdots, \mathbf{H}_L^*\right)$ is a global minimizer in (PA$_{\boldsymbol{\phi}}$) if and only if the sum of rank one matrices is a zero matrix for $i = 0, 1, \cdots, L$, i.e.,

$$\sum_{n=1}^{N} \mathbf{v}_i\left(\mathbf{x}^n\right)\mathbf{g}_a\left(\mathbf{x}^n\right)^T = \mathbf{0}, \quad i = 0, 1, \cdots, L. \qquad (2.23)$$

Next, we show that every local minimizer $\left(\mathbf{W}_L^*, \mathbf{W}_{L+1}^*\right)$ of (P$_{\boldsymbol{\phi}}$) establishes a corresponding global minimizer $\left(\mathbf{H}_0^*, \cdots, \mathbf{H}_L^*\right)$ in (PA$_{\boldsymbol{\phi}}$) such that $\mathbf{H}_i^* = \mathbf{W}_{L+1}^*\mathbf{W}_i$ for $i = 0, 1, \cdots, L$.

At any local minimizer of (P$_{\boldsymbol{\phi}}$), the first-order necessary condition with respect to $\mathbf{W}_L$ is given by

$$\begin{aligned}
\frac{\partial \mathscr{R}}{\partial \mathrm{vec}\left(\mathbf{W}_L\right)} &= \frac{1}{N}\sum_{n=1}^{N} \left.\frac{\partial \ell\left(\hat{\mathbf{y}},\mathbf{y}^n\right)}{\partial \mathrm{vec}\left(\mathbf{W}_L\right)}\right|_{\hat{\mathbf{y}}=\mathbf{W}_{L+1}\sum_{i=0}^{L}\mathbf{W}_i\mathbf{v}_i(\mathbf{x}^n)} \\
&= \frac{1}{N}\sum_{n=1}^{N} \left.\frac{\partial \ell\left(\hat{\mathbf{y}},\mathbf{y}^n\right)}{\partial \hat{\mathbf{y}}}\frac{\partial \hat{\mathbf{y}}}{\partial \mathrm{vec}\left(\mathbf{W}_L\right)}\right|_{\hat{\mathbf{y}}=\sum_{i=0}^{L}\mathbf{W}_{L+1}\mathbf{W}_i\mathbf{v}_i(\mathbf{x}^n)} \\
&= \frac{1}{N}\sum_{n=1}^{N} \left.\frac{\partial \ell\left(\hat{\mathbf{y}},\mathbf{y}^n\right)}{\partial \hat{\mathbf{y}}}\right|_{\hat{\mathbf{y}}=\mathbf{W}_{L+1}\sum_{i=0}^{L}\mathbf{W}_i\mathbf{v}_i(\mathbf{x}^n)} \left(\mathbf{v}_L\left(\mathbf{x}^n\right)^T \otimes \mathbf{W}_{L+1}\right) \\
&= \frac{1}{N}\sum_{n=1}^{N} \left(\left(\mathbf{v}_L\left(\mathbf{x}^n\right) \otimes \mathbf{W}_{L+1}^T\right) \underbrace{\left.\frac{\partial \ell\left(\hat{\mathbf{y}},\mathbf{y}^n\right)^T}{\partial \hat{\mathbf{y}}}\right|_{\hat{\mathbf{y}}=\mathbf{W}_{L+1}\sum_{i=0}^{L}\mathbf{W}_i\mathbf{v}_i(\mathbf{x}^n)}}_{\mathbf{g}_r(\mathbf{x}^n)}\right)^T \qquad (2.24) \\
&= \frac{1}{N}\sum_{n=1}^{N} \mathrm{vec}\left(\mathbf{W}_{L+1}^T\mathbf{g}_r\left(\mathbf{x}^n\right)\mathbf{v}_L\left(\mathbf{x}^n\right)^T\right)^T \\
&= \mathbf{0}.
\end{aligned}$$

Equivalently, we can write the above first-order necessary condition into a matrix form

$$\sum_{n=1}^{N} \mathbf{v}_L\left(\mathbf{x}^n\right)\mathbf{g}_r\left(\mathbf{x}^n\right)^T\mathbf{W}_{L+1} = \mathbf{0}. \qquad (2.25)$$

On the other hand, for the first-order necessary condition with respect to $\mathbf{W}_{L+1}$, we obtain

$$\begin{aligned}
\frac{\partial \mathscr{R}}{\partial \mathrm{vec}\left(\mathbf{W}_{L+1}\right)} &= \frac{1}{N}\sum_{n=1}^{N} \left.\frac{\partial \ell\left(\hat{\mathbf{y}},\mathbf{y}^n\right)}{\partial \mathrm{vec}\left(\mathbf{W}_{L+1}\right)}\right|_{\hat{\mathbf{y}}=\mathbf{W}_{L+1}\sum_{i=0}^{L}\mathbf{W}_i\mathbf{v}_i(\mathbf{x}^n)} \\
&= \frac{1}{N}\sum_{n=1}^{N} \left.\frac{\partial \ell\left(\hat{\mathbf{y}},\mathbf{y}^n\right)}{\partial \hat{\mathbf{y}}}\frac{\partial \hat{\mathbf{y}}}{\partial \mathrm{vec}\left(\mathbf{W}_{L+1}\right)}\right|_{\hat{\mathbf{y}}=\sum_{i=0}^{L}\mathbf{W}_{L+1}\mathbf{W}_i\mathbf{v}_i(\mathbf{x}^n)} \\
&= \frac{1}{N}\sum_{n=1}^{N} \mathbf{g}_r\left(\mathbf{x}^n\right)^T \left(\left(\sum_{i=0}^{L}\mathbf{W}_i\mathbf{v}_i\left(\mathbf{x}^n\right)\right)^T \otimes \mathbf{I}_{N_o}\right) \\
&= \frac{1}{N}\sum_{n=1}^{N} \left(\left(\left(\sum_{i=0}^{L}\mathbf{W}_i\mathbf{v}_i\left(\mathbf{x}^n\right)\right)\otimes \mathbf{I}_{N_o}\right)\mathbf{g}_r\left(\mathbf{x}^n\right)\right)^T \\
&= \frac{1}{N}\sum_{n=1}^{N} \mathrm{vec}\left(\mathbf{g}_r\left(\mathbf{x}^n\right)\sum_{i=0}^{L}\mathbf{v}_i\left(\mathbf{x}^n\right)^T\mathbf{W}_i^T\right)^T \\
&= \mathbf{0}.
\end{aligned} \tag{2.26}$$

The corresponding matrix form of the above condition is given by

$$\sum_{i=0}^{L}\mathbf{W}_i\sum_{n=1}^{N}\mathbf{v}_i\left(\mathbf{x}^n\right)\mathbf{g}_r\left(\mathbf{x}^n\right)^T = \mathbf{0}. \tag{2.27}$$

When $\mathbf{W}_{L+1}$ is full rank at a critical point, (2.25) implies $\sum_{n=1}^{N}\mathbf{v}_L\left(\mathbf{x}^n\right)\mathbf{g}_r\left(\mathbf{x}^n\right)^T = \mathbf{0}$ because the null space of $\mathbf{W}_{L+1}^T$ is degenerate according to Assumption 4. Then, applying such an implication to (2.27) along with Assumption 5, we obtain

$$\sum_{i=0}^{L-1}\mathbf{W}_i\sum_{n=1}^{N}\mathbf{v}_i\left(\mathbf{x}^n\right)\mathbf{g}_r\left(\mathbf{x}^n\right)^T = \mathbf{0} \implies \sum_{n=1}^{N}\mathbf{v}_i\left(\mathbf{x}^n\right)\mathbf{g}_r\left(\mathbf{x}^n\right)^T = \mathbf{0}, \quad i = 0, 1, \cdots, L-1. \tag{2.28}$$

Note that all the column vectors in $\begin{bmatrix}\mathbf{W}_0 & \mathbf{W}_1 & \cdots & \mathbf{W}_{L-1}\end{bmatrix}$ are linearly independent if and only if the linear inverse problem $\sum_{i=0}^{L-1}\mathbf{x}_i = \sum_{i=0}^{L-1}\mathbf{W}_i\mathbf{v}_i$ has a unique solution for $\mathbf{v}_0, \cdots, \mathbf{v}_{L-1}$. We have proved the statement (a).

On the other hand, when $\mathbf{W}_{L+1}$ is not full rank at a local minimizer, then there exists a perturbation on $\mathbf{W}_L$ such that the new point is still a local minimizer which has the same objective value. Let $(\mathbf{W}_L, \mathbf{W}_{L+1})$ be any local minimizer of ($\mathrm{P}_{\boldsymbol{\phi}}$) for which $\mathbf{W}_{L+1}$ is not full row rank. By the definition of a local minimizer, there exists some $\gamma > 0$ such that

$$\mathscr{R}\left(\mathbf{W}'_L, \mathbf{W}'_{L+1}; \boldsymbol{\phi}\right) \geq \mathscr{R}\left(\mathbf{W}_L, \mathbf{W}_{L+1}; \boldsymbol{\phi}\right), \forall \left(\mathbf{W}'_L, \mathbf{W}'_{L+1}\right) \in B\left(\left(\mathbf{W}_L, \mathbf{W}_{L+1}\right), \gamma\right) \tag{2.29}$$

where B is an open ball centered at $(\mathbf{W}_L, \mathbf{W}_{L+1})$ with the radius γ. Then $(\mathbf{W}_L + \mathbf{a}\mathbf{b}^T, \mathbf{W}_{L+1})$ must also be a local minimizer for any nonzero $\mathbf{a} \in \mathscr{N}(\mathbf{W}_{L+1})$ and any sufficiently small nonzero $\mathbf{b} \in \mathbb{R}^{K_L}$ such that $(\mathbf{W}_L + \mathbf{a}\mathbf{b}^T, \mathbf{W}_{L+1}) \in B\left((\mathbf{W}_L, \mathbf{W}_{L+1}), \gamma/2\right)$. Substituting the minimizer $(\mathbf{W}_L + \mathbf{a}\mathbf{b}^T, \mathbf{W}_{L+1})$ in (2.27) yields

$$\sum_{i=0}^{L-1} \mathbf{W}_i \sum_{n=1}^{N} \mathbf{v}_i\left(\mathbf{x}^n\right) \mathbf{g}_r\left(\mathbf{x}^n\right)^T + \left(\mathbf{W}_L + \mathbf{a}\mathbf{b}^T\right) \sum_{n=1}^{N} \mathbf{v}_L\left(\mathbf{x}^n\right) \mathbf{g}_r\left(\mathbf{x}^n\right)^T = \mathbf{0}. \tag{2.30}$$

Subtracting (2.27) from the above equation, we obtain

$$\mathbf{a}\mathbf{b}^T \sum_{n=1}^{N} \mathbf{v}_L\left(\mathbf{x}^n\right) \mathbf{g}_r\left(\mathbf{x}^n\right)^T = \mathbf{0}. \tag{2.31}$$

Multiplying both sides by $\mathbf{a}^T / \|\mathbf{a}\|_2^2$, we have

$$\mathbf{b}^T \sum_{n=1}^{N} \mathbf{v}_L\left(\mathbf{x}^n\right) \mathbf{g}_r\left(\mathbf{x}^n\right)^T = \mathbf{0} \implies \sum_{n=1}^{N} \mathbf{v}_L\left(\mathbf{x}^n\right) \mathbf{g}_r\left(\mathbf{x}^n\right)^T = \mathbf{0} \tag{2.32}$$

because $\mathbf{b} \neq \mathbf{0}$ can be arbitrary as long as it is sufficiently small. As a result, (2.28) is also true when $\mathbf{W}_{L+1}$ is not full row rank. We have proved the statement (b). □

Proof of Corollary 1

Proof. The proof of Theorem 4 has shown that every local minimizer $\left(\mathbf{W}_L^*, \mathbf{W}_{L+1}^*\right)$ of ($\mathrm{P}_{\boldsymbol{\phi}}$) establishes a corresponding global minimizer $\left(\mathbf{H}_0^*, \cdots, \mathbf{H}_L^*\right)$ in ($\mathrm{PA}_{\boldsymbol{\phi}}$) such that $\mathbf{H}_i^* = \mathbf{W}_{L+1}^* \mathbf{W}_i$

for $i = 0, 1, \cdots, L$. Therefore, it must be true that

$$\mathscr{R}\left(\mathbf{W}_{L_\alpha}^*, \mathbf{W}_{L_\alpha+1}^*, \boldsymbol{\alpha}\right) = \mathscr{A}\left(\mathbf{H}_0^*, \cdots, \mathbf{H}_{L_\alpha}^*, \mathbf{0}, \cdots, \mathbf{0}, \boldsymbol{\phi}\right). \tag{2.33}$$

Next, by the convexity in Proposition 3, we have

$$\begin{aligned} \mathscr{R}\left(\mathbf{W}_{L_\alpha}^*, \mathbf{W}_{L_\alpha+1}^*, \boldsymbol{\alpha}\right) &= \mathscr{A}\left(\mathbf{H}_0^*, \cdots, \mathbf{H}_{L_\alpha-1}^*, \mathbf{H}_{L_\alpha}^*, \mathbf{0}, \cdots, \mathbf{0}, \boldsymbol{\phi}\right) \\ &\overset{a}{\leq} \mathscr{A}\left(\mathbf{H}_0^*, \cdots, \mathbf{H}_{L_\beta}^*, \mathbf{0}, \cdots, \mathbf{0}, \boldsymbol{\phi}\right) \\ &= \mathscr{R}\left(\mathbf{W}_{L_\beta}^*, \mathbf{W}_{L_\beta+1}^*, \boldsymbol{\beta}\right). \end{aligned} \tag{2.34}$$

The equality a in (2.34) holds true by the relation $L_\beta < L_\alpha$. □

Proof of Corollary 2

Proof. By Theorem 4 (b),

$$\mathscr{R}\left(\mathbf{W}_0^*, \cdots, \mathbf{W}_{L+1}^*, \boldsymbol{\theta}_1^*, \cdots, \boldsymbol{\theta}_L^*\right) = \mathscr{R}\left(\mathbf{W}_L^*, \mathbf{W}_{L+1}^*, \boldsymbol{\phi}^*\right) = \mathscr{A}\left(\mathbf{H}_0^*, \cdots, \mathbf{H}_L^*, \boldsymbol{\phi}^*\right) \tag{2.35}$$

for any local minimizer $\left(\mathbf{H}_0^*, \cdots, \mathbf{H}_L^*\right)$ of (PA$_{\boldsymbol{\phi}}$) using feature finding parameters $\boldsymbol{\phi}^*$. Then, by the convexity in Proposition 3, every local minimizer $\left(\mathbf{H}_0^*, \cdots, \mathbf{H}_L^*, \boldsymbol{\phi}^*\right)$ is a global minimzer of (PA$_{\boldsymbol{\phi}}$) using $\boldsymbol{\phi}^*$. Hence, it must be true that

$$\begin{aligned} \mathscr{R}\left(\mathbf{W}_0^*, \cdots, \mathbf{W}_{L+1}^*, \boldsymbol{\theta}_1^*, \cdots, \boldsymbol{\theta}_L^*\right) &= \mathscr{A}\left(\mathbf{H}_0^*, \cdots, \mathbf{H}_L^*, \boldsymbol{\phi}^*\right) \\ &\leq \mathscr{A}\left(\mathbf{H}_0^*, \mathbf{0}, \cdots, \mathbf{0}, \boldsymbol{\phi}^*\right) \\ &= \mathscr{A}\left(\mathbf{H}_0^*, \mathbf{0}, \cdots, \mathbf{0}, \boldsymbol{\psi}\right) \\ &= \min_{\mathbf{A} \in \mathbb{R}^{N_o \times N_{in}}} \frac{1}{N} \sum_{n=1}^{N} \ell\left(\mathbf{A}\mathbf{x}^n, \mathbf{y}^n\right) \end{aligned} \tag{2.36}$$

for arbitrary $\boldsymbol{\psi}$ due to the zero prediction weights for $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_L$. We have proved the statement (a). If the inequality in (2.36) is strict, i.e.,

$$\mathscr{A}\left(\mathbf{H}_0^*, \cdots, \mathbf{H}_L^*, \boldsymbol{\phi}^*\right) < \mathscr{A}\left(\mathbf{H}_0^*, \mathbf{0}, \cdots, \mathbf{0}, \boldsymbol{\phi}^*\right), \tag{2.37}$$

then (2.36) implies

$$\mathscr{R}\left(\mathbf{W}_0^*, \cdots, \mathbf{W}_{L+1}^*, \boldsymbol{\theta}_1^*, \cdots, \boldsymbol{\theta}_L^*\right) < \min_{\mathbf{A} \in \mathbb{R}^{N_o \times N_{in}}} \frac{1}{N} \sum_{n=1}^{N} \ell\left(\mathbf{A}\mathbf{x}^n, \mathbf{y}^n\right). \tag{2.38}$$

We have proved the statement (b). $\square$

Proof of Theorem 5

Proof. By Theorem 4 (a), every critical point with full rank $\mathbf{W}_{L+1}$ is a global minimizer of (P$_{\boldsymbol{\phi}}$). Therefore, $\mathbf{W}_{L+1}$ must be rank-deficient at every saddle point. We have proved the statement (a).

We argue that the Hessian is neither positive semidefinite nor negative semidefinite at every saddle point. According to the proof of Proposition 2, there exists no point in the domain of the objective function of (P$_{\boldsymbol{\phi}}$) at which the Hessian is negative semidefinite. If $\mathbf{W}_{L+1}$ is not full rank, then the positive semidefiniteness of the Hessian at every critical point becomes a sufficient condition for a local minimizer. This can be easily seen by replacing the convex loss with the squared loss in the proof for Theorem 4 and applying (2.18). We conclude that the Hessian must be indefinite at every saddle point under the assumptions; in other words, the Hessian has at least one strictly negative eigenvalue. We have proved the statement (b). $\square$

Proof of Proposition 4

Proof. Note that (PD$_{\boldsymbol{\phi}}$) is a convex optimization problem because its objective function is a nonnegative weighted sum of convex functions composited with affine mappings. Since (PD$_{\boldsymbol{\phi}}$)

is a convex optimization problem, it is true that

$$\mathscr{D}\left(\mathbf{W}_{L+1}^{*};\boldsymbol{\phi}\right) \leq \min_{\mathbf{A}\in\mathbb{R}^{N_o\times N_{in}}} \mathscr{D}\left(\begin{bmatrix}\mathbf{A} & \mathbf{0} & \cdots & \mathbf{0}\end{bmatrix};\boldsymbol{\phi}\right) = \min_{\mathbf{A}\in\mathbb{R}^{N_o\times N_{in}}} \frac{1}{N}\sum_{n=1}^{N}\ell\left(\mathbf{A}\mathbf{x}^n,\mathbf{y}^n\right) \tag{2.39}$$

for any local minimizer $\mathbf{W}_{L+1}^{*}$ of (PD$_{\boldsymbol{\phi}}$) and arbitrary feature finding parameters $\boldsymbol{\phi}$. □

Proof of Proposition 5

Proof. Let $\mathbf{0}_{m\times n}$ be an m-by-n zero matrix and $\mathbf{I}_{m\times n}$ be an m-by-n matrix with ones on diagonal entries and zero elsewhere, i.e.,

$$\left[\mathbf{I}_{m\times n}\right]_{ij} = \begin{cases} 1, & i=j \\ 0, & i\neq j \end{cases}. \tag{2.40}$$

The superscript of every hyperparameter in this proof indicates the network type. We define $M^{\text{ResNEst}} = \sum_{i=0}^{L} K_i^{\text{ResNEst}} = M_L^{\text{DenseNEst}}$ and $K_i^{\text{ResNEst}} = D_i^{\text{DenseNEst}}$ for $i=1,2,\cdots,L$. Let

$$\boldsymbol{\Pi}_i = \begin{bmatrix} \mathbf{0}_{\left(\sum_{j=0}^{i-1} K_j^{\text{ResNEst}}\right)\times K_i^{\text{ResNEst}}} \\ \mathbf{I}_{K_i^{\text{ResNEst}}\times K_i^{\text{ResNEst}}} \\ \mathbf{0}_{\left(M^{\text{ResNEst}}-\sum_{j=0}^{i} K_j^{\text{ResNEst}}\right)\times K_i^{\text{ResNEst}}} \end{bmatrix} \tag{2.41}$$

for $i=0,1,\cdots,L$. Let $\mathbf{W}_{L+1}^{\text{ResNEst}} = \mathbf{W}_{L+1}^{\text{DenseNEst}}$ and $\mathbf{W}_i^{\text{ResNEst}} = \boldsymbol{\Pi}_i$ for $i=0,1,\cdots,L$. We define the function $\mathbf{G}_i$ in the ResNEst as

$$\mathbf{G}_i\left(\mathbf{x}_{i-1}\right) = \mathbf{Q}_i\left(\begin{bmatrix}\boldsymbol{\Pi}_0 & \boldsymbol{\Pi}_1 & \cdots & \boldsymbol{\Pi}_{i-1}\end{bmatrix}^T \mathbf{x}_{i-1}\right) \tag{2.42}$$

for $i=1,\cdots,L$ where $\mathbf{x}_i \in \mathbb{R}^{M^{\text{ResNEst}}}$ is the residual representation in the ResNEst.

Based on such a construction, the feature finding weights $\boldsymbol{\phi}$ in the ResNEst satisfies Assumption 5. Therefore, by Theorem 4 (b), the excess minimum empirical risk is zero or $\varepsilon = 0$, i.e., the minimum value at every local minimizer of (P$_{\boldsymbol{\phi}}$) is equivalent to the global minimum value in (PA$_{\boldsymbol{\phi}}$). $\square$

Important first- and second-order derivatives

We derive the Hessian in the proof of Proposition 2. Let $\mathbf{X} = \begin{bmatrix} \mathbf{x}^1 & \mathbf{x}^2 & \cdots & \mathbf{x}^N \end{bmatrix}$. Let $\hat{\mathbf{Y}}(\mathbf{X}) = \begin{bmatrix} \hat{\mathbf{y}}(\mathbf{x}^1) & \hat{\mathbf{y}}(\mathbf{x}^2) & \cdots & \hat{\mathbf{y}}(\mathbf{x}^N) \end{bmatrix}$ where each of column vectors is the ResNEst output given by the function $\hat{\mathbf{y}}(\mathbf{x}) = \mathbf{W}_{L+1} \sum_{i=0}^{L} \mathbf{W}_i \mathbf{v}_i(\mathbf{x})$. The empirical risk using the squared loss (up to a scaling factor) is defined as

$$\mathscr{R}\left(\mathbf{W}_L, \mathbf{W}_{L+1}; \boldsymbol{\phi}\right) = \frac{1}{2}\frac{1}{N}\sum_{n=1}^{N} \left\|\hat{\mathbf{y}}\left(\mathbf{x}^n\right) - \mathbf{y}^n\right\|_2^2. \tag{2.43}$$

The Jacobian of $\mathscr{R}$ with respect to $\mathbf{W}_L$ is given by

$$\begin{aligned}
\frac{\partial \mathscr{R}}{\partial \text{vec}(\mathbf{W}_L^T)} &= \frac{1}{2}\frac{1}{N}\frac{\partial}{\partial \text{vec}(\mathbf{W}_L^T)}\sum_{n=1}^{N} \left\|\hat{\mathbf{y}}\left(\mathbf{x}^n\right) - \mathbf{y}^n\right\|_2^2 \\
&= \frac{1}{2}\frac{1}{N}\frac{\partial}{\partial \text{vec}(\mathbf{W}_L^T)}\left\|\hat{\mathbf{Y}}(\mathbf{X}) - \mathbf{Y}\right\|_F^2 \\
&= \frac{1}{2}\frac{1}{N}\frac{\partial}{\partial \text{vec}(\mathbf{W}_L^T)}\text{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right)^T \text{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right) \\
&= \frac{1}{N}\text{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right)^T \frac{\partial}{\partial \text{vec}(\mathbf{W}_L^T)}\text{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right) \\
&= \frac{1}{N}\text{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right)^T \frac{\partial}{\partial \text{vec}(\mathbf{W}_L^T)}\text{vec}\left(\sum_{i=0}^{L}\mathbf{V}_i^T\mathbf{W}_i^T\mathbf{W}_{L+1}^T - \mathbf{Y}^T\right) \\
&= \frac{1}{N}\text{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right)^T \frac{\partial}{\partial \text{vec}(\mathbf{W}_L^T)}\text{vec}\left(\mathbf{V}_L^T\mathbf{W}_L^T\mathbf{W}_{L+1}^T\right) \\
&= \frac{1}{N}\text{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right)^T \frac{\partial}{\partial \text{vec}(\mathbf{W}_L^T)}\left(\mathbf{W}_{L+1}\otimes\mathbf{V}_L^T\right)\text{vec}\left(\mathbf{W}_L^T\right) \\
&= \frac{1}{N}\text{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right)^T \left(\mathbf{W}_{L+1}\otimes\mathbf{V}_L^T\right).
\end{aligned} \tag{2.44}$$

The Jacobian of $\mathscr{R}$ with respect to $\mathbf{W}_{L+1}$ is given by

$$\begin{aligned}
\frac{\partial \mathscr{R}}{\partial \operatorname{vec}(\mathbf{W}_{L+1}^T)} &= \frac{1}{2}\frac{1}{N}\frac{\partial}{\partial \operatorname{vec}(\mathbf{W}_{L+1}^T)}\operatorname{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right)^T \operatorname{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right) \\
&= \mathbf{e}^T \frac{\partial}{\partial \operatorname{vec}(\mathbf{W}_{L+1}^T)}\operatorname{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right) \\
&= \mathbf{e}^T \frac{\partial}{\partial \operatorname{vec}(\mathbf{W}_{L+1}^T)}\operatorname{vec}\left(\sum_{i=0}^{L} \mathbf{V}_i^T \mathbf{W}_i^T \mathbf{W}_{L+1}^T - \mathbf{Y}^T\right) \\
&= \mathbf{e}^T \frac{\partial}{\partial \operatorname{vec}(\mathbf{W}_{L+1}^T)}\operatorname{vec}\left(\sum_{i=0}^{L} \mathbf{V}_i^T \mathbf{W}_i^T \mathbf{W}_{L+1}^T \mathbf{I}_{N_o}\right) \\
&= \mathbf{e}^T \frac{\partial}{\partial \operatorname{vec}(\mathbf{W}_{L+1}^T)}\left(\mathbf{I}_{N_o} \otimes \sum_{i=0}^{L} \mathbf{V}_i^T \mathbf{W}_i^T\right)\operatorname{vec}\left(\mathbf{W}_{L+1}^T\right) \\
&= \mathbf{e}^T \left(\mathbf{I}_{N_o} \otimes \sum_{i=0}^{L} \mathbf{V}_i^T \mathbf{W}_i^T\right)
\end{aligned} \tag{2.45}$$

where we have used

$$\mathbf{e} = \frac{1}{N}\operatorname{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right). \tag{2.46}$$

Now, we find each of the block matrices in the Hessian.

$$\begin{aligned}
\frac{\partial^2 \mathscr{R}}{\partial \operatorname{vec}\left(\mathbf{W}_L^T\right)^2} &= \frac{\partial}{\partial \operatorname{vec}\left(\mathbf{W}_L^T\right)}\left(\frac{\partial \mathscr{R}}{\partial \operatorname{vec}(\mathbf{W}_L^T)}\right)^T \\
&= \frac{\partial}{\partial \operatorname{vec}\left(\mathbf{W}_L^T\right)}\frac{1}{N}\left(\mathbf{W}_{L+1}^T \otimes \mathbf{V}_L\right)\operatorname{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right) \\
&= \frac{1}{N}\left(\mathbf{W}_{L+1}^T \otimes \mathbf{V}_L\right)\frac{\partial}{\partial \operatorname{vec}\left(\mathbf{W}_L^T\right)}\operatorname{vec}\left(\sum_{i=0}^{L} \mathbf{V}_i^T \mathbf{W}_i^T \mathbf{W}_{L+1}^T - \mathbf{Y}^T\right) \\
&= \frac{1}{N}\left(\mathbf{W}_{L+1}^T \otimes \mathbf{V}_L\right)\frac{\partial}{\partial \operatorname{vec}\left(\mathbf{W}_L^T\right)}\operatorname{vec}\left(\mathbf{V}_L^T \mathbf{W}_L^T \mathbf{W}_{L+1}^T\right) \\
&= \frac{1}{N}\left(\mathbf{W}_{L+1}^T \otimes \mathbf{V}_L\right)\frac{\partial}{\partial \operatorname{vec}\left(\mathbf{W}_L^T\right)}\operatorname{vec}\left(\mathbf{V}_L^T \mathbf{W}_L^T \mathbf{W}_{L+1}^T\right) \\
&= \frac{1}{N}\left(\mathbf{W}_{L+1}^T \otimes \mathbf{V}_L\right)\left(\mathbf{W}_{L+1} \otimes \mathbf{V}_L^T\right) \\
&= \frac{1}{N}\left(\mathbf{W}_{L+1}^T \mathbf{W}_{L+1} \otimes \mathbf{V}_L \mathbf{V}_L^T\right).
\end{aligned} \tag{2.47}$$

$$
\begin{aligned}
\frac{\partial^2 \mathscr{R}}{\partial \operatorname{vec}\left(\mathbf{W}_{L+1}^T\right)^2} &= \frac{\partial}{\partial \operatorname{vec}\left(\mathbf{W}_{L+1}^T\right)} \left(\frac{\partial \mathscr{R}}{\partial \operatorname{vec}\left(\mathbf{W}_{L+1}^T\right)} \right)^T \\
&= \frac{1}{N} \frac{\partial}{\partial \operatorname{vec}\left(\mathbf{W}_{L+1}^T\right)} \left(\mathbf{I}_{N_o} \otimes \sum_{i=0}^{L} \mathbf{W}_i \mathbf{V}_i \right) \operatorname{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T \right) \\
&= \mathbf{F} \frac{\partial}{\partial \operatorname{vec}\left(\mathbf{W}_{L+1}^T\right)} \operatorname{vec}\left(\sum_{i=0}^{L} \mathbf{V}_i^T \mathbf{W}_i^T \mathbf{W}_{L+1}^T - \mathbf{Y}^T \right) \\
&= \mathbf{F} \frac{\partial}{\partial \operatorname{vec}\left(\mathbf{W}_{L+1}^T\right)} \operatorname{vec}\left(\sum_{i=0}^{L} \mathbf{V}_i^T \mathbf{W}_i^T \mathbf{W}_{L+1}^T \mathbf{I}_{N_o} \right) \\
&= \mathbf{F} \left(\mathbf{I}_{N_o} \otimes \sum_{i=0}^{L} \mathbf{V}_i^T \mathbf{W}_i^T \right) \\
&= \frac{1}{N} \left(\mathbf{I}_{N_o} \otimes \sum_{i=0}^{L} \mathbf{W}_i \mathbf{V}_i \left(\sum_{i=0}^{L} \mathbf{W}_i \mathbf{V}_i \right)^T \right)
\end{aligned}
\tag{2.48}
$$

where we have used

$$
\mathbf{F} = \frac{1}{N} \left(\mathbf{I}_{N_o} \otimes \sum_{i=0}^{L} \mathbf{W}_i \mathbf{V}_i \right). \tag{2.49}
$$

$$
\begin{aligned}
&\frac{\partial^2 \mathscr{R}}{\partial \operatorname{vec}\left(\mathbf{W}_{L+1}^T\right) \partial \operatorname{vec}\left(\mathbf{W}_L^T\right)} \\
&= \frac{\partial}{\partial \operatorname{vec}\left(\mathbf{W}_{L+1}^T\right)} \left(\frac{\partial \mathscr{R}}{\partial \operatorname{vec}\left(\mathbf{W}_L^T\right)} \right)^T \\
&= \frac{1}{N} \frac{\partial}{\partial \operatorname{vec}\left(\mathbf{W}_{L+1}^T\right)} \left(\mathbf{W}_{L+1}^T \otimes \mathbf{V}_L\right) \operatorname{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right) \\
&= \frac{1}{N} \left(\frac{\partial}{\partial \operatorname{vec}\left(\mathbf{W}_{L+1}^T\right)} \left(\mathbf{W}_{L+1}^T \otimes \mathbf{V}_L\right) \right) \operatorname{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right) \qquad (2.50) \\
&+ \frac{1}{N} \left(\mathbf{W}_{L+1}^T \otimes \mathbf{V}_L\right) \frac{\partial}{\partial \operatorname{vec}\left(\mathbf{W}_{L+1}^T\right)} \operatorname{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right) \quad \text{(see (2.52))} \\
&= \frac{1}{N} \begin{bmatrix} \mathbf{I}_M \otimes \mathbf{V}_L \boldsymbol{\delta}_1 & \cdots & \mathbf{I}_M \otimes \mathbf{V}_L \boldsymbol{\delta}_{N_o} \end{bmatrix} + \frac{1}{N} \left(\mathbf{W}_{L+1}^T \otimes \mathbf{V}_L\right) \left(\mathbf{I}_{N_o} \otimes \sum_{i=0}^{L} \mathbf{V}_i^T \mathbf{W}_i^T \right) \\
&= \frac{1}{N} \begin{bmatrix} \mathbf{I}_M \otimes \mathbf{V}_L \boldsymbol{\delta}_1 & \cdots & \mathbf{I}_M \otimes \mathbf{V}_L \boldsymbol{\delta}_{N_o} \end{bmatrix} + \frac{1}{N} \left(\mathbf{W}_{L+1}^T \otimes \mathbf{V}_L \sum_{i=0}^{L} \mathbf{V}_i^T \mathbf{W}_i^T \right).
\end{aligned}
$$

$$
\begin{aligned}
&\frac{\partial^2 \mathscr{R}}{\partial \operatorname{vec}\left(\mathbf{W}_L^T\right) \partial \operatorname{vec}\left(\mathbf{W}_{L+1}^T\right)} \\
&= \frac{\partial}{\partial \operatorname{vec}\left(\mathbf{W}_L^T\right)} \left(\frac{\partial \mathscr{R}}{\partial \operatorname{vec}\left(\mathbf{W}_{L+1}^T\right)} \right)^T \\
&= \frac{1}{N} \frac{\partial}{\partial \operatorname{vec}\left(\mathbf{W}_L^T\right)} \left(\mathbf{I}_{N_o} \otimes \sum_{i=0}^{L} \mathbf{W}_i \mathbf{V}_i \right) \operatorname{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right) \\
&= \frac{1}{N} \left(\frac{\partial}{\partial \operatorname{vec}\left(\mathbf{W}_L^T\right)} \left(\mathbf{I}_{N_o} \otimes \sum_{i=0}^{L} \mathbf{W}_i \mathbf{V}_i \right) \right) \operatorname{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right) \qquad (2.51) \\
&+ \frac{1}{N} \left(\mathbf{I}_{N_o} \otimes \sum_{i=0}^{L} \mathbf{W}_i \mathbf{V}_i \right) \frac{\partial}{\partial \operatorname{vec}\left(\mathbf{W}_L^T\right)} \operatorname{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right) \quad \text{(see (2.53) and (2.54))} \\
&= \frac{1}{N} \begin{bmatrix} \mathbf{I}_M \otimes \boldsymbol{\delta}_1^T \mathbf{V}_L^T \\ \mathbf{I}_M \otimes \boldsymbol{\delta}_2^T \mathbf{V}_L^T \\ \vdots \\ \mathbf{I}_M \otimes \boldsymbol{\delta}_{N_o}^T \mathbf{V}_L^T \end{bmatrix} + \frac{1}{N} \mathbf{W}_{L+1} \otimes \sum_{i=0}^{L} \mathbf{W}_i \mathbf{V}_i \mathbf{V}_L^T.
\end{aligned}
$$

Notice that we have used the following identities in (2.50) and (2.51).

$$
\begin{aligned}
&\left(\frac{\partial}{\partial \mathrm{vec}\left(\mathbf{W}_{L+1}^T\right)}\left(\mathbf{W}_{L+1}^T \otimes \mathbf{V}_L\right)\right)\mathrm{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right) \\
&= \left(\frac{\partial}{\partial \mathrm{vec}\left(\mathbf{W}_{L+1}^T\right)}\left(\mathbf{W}_{L+1}^T \otimes \mathbf{V}_L\right)\right)\mathrm{vec}\left(\mathbf{\Delta}\right) \\
&= \sum_{j=1}^{N_o}\sum_{k=1}^{N}\left(\frac{\partial}{\partial \mathrm{vec}\left(\mathbf{W}_{L+1}^T\right)}\left(\left(\mathbf{W}_{L+1}^T\right)_j \otimes (\mathbf{V}_L)_k\right)\right)\delta_{k,j} \\
&= \left[\sum_{j=1}^{N_o}\sum_{k=1}^{N}\delta_{k,j}\frac{\partial\left(\mathbf{W}_{L+1}^T\right)_j \otimes (\mathbf{V}_L)_k}{\partial\left(\mathbf{W}_{L+1}^T\right)_1} \quad \cdots \quad \sum_{j=1}^{N_o}\sum_{k=1}^{N}\delta_{k,j}\frac{\partial\left(\mathbf{W}_{L+1}^T\right)_j \otimes (\mathbf{V}_L)_k}{\partial\left(\mathbf{W}_{L+1}^T\right)_{N_o}}\right] \\
&= \left[\sum_{k=1}^{N}\delta_{k,1}\frac{\partial \mathrm{vec}\left((\mathbf{V}_L)_k\left(\mathbf{W}_{L+1}^T\right)_1^T\right)}{\partial\left(\mathbf{W}_{L+1}^T\right)_1} \quad \cdots \quad \sum_{k=1}^{N}\delta_{k,N_o}\frac{\partial \mathrm{vec}\left((\mathbf{V}_L)_k\left(\mathbf{W}_{L+1}^T\right)_{N_o}^T\right)}{\partial\left(\mathbf{W}_{L+1}^T\right)_{N_o}}\right] \\
&= \left[\sum_{k=1}^{N}\delta_{k,1}\mathbf{I}_M \otimes (\mathbf{V}_L)_k \quad \cdots \quad \sum_{k=1}^{N}\delta_{k,N_o}\mathbf{I}_M \otimes (\mathbf{V}_L)_k\right] \\
&= \left[\mathbf{I}_M \otimes \mathbf{V}_L\boldsymbol{\delta}_1 \quad \cdots \quad \mathbf{I}_M \otimes \mathbf{V}_L\boldsymbol{\delta}_{N_o}\right].
\end{aligned}
\tag{2.52}
$$

$$\left(\frac{\partial}{\partial \text{vec}\left(\mathbf{W}_L^T\right)}\left(\mathbf{I}_{N_o} \otimes \sum_{i=0}^{L} \mathbf{W}_i \mathbf{V}_i\right)\right) \text{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right)$$

$$= \begin{bmatrix} \boldsymbol{\delta}_1^T \frac{\partial}{\partial \text{vec}\left(\mathbf{W}_L^T\right)}\left(\sum_{i=0}^{L} \mathbf{V}_i^T \mathbf{W}_i^T\right)_1 \\ \vdots \\ \boldsymbol{\delta}_1^T \frac{\partial}{\partial \text{vec}\left(\mathbf{W}_L^T\right)}\left(\sum_{i=0}^{L} \mathbf{V}_i^T \mathbf{W}_i^T\right)_M \\ \boldsymbol{\delta}_2^T \frac{\partial}{\partial \text{vec}\left(\mathbf{W}_L^T\right)}\left(\sum_{i=0}^{L} \mathbf{V}_i^T \mathbf{W}_i^T\right)_1 \\ \vdots \\ \boldsymbol{\delta}_2^T \frac{\partial}{\partial \text{vec}\left(\mathbf{W}_L^T\right)}\left(\sum_{i=0}^{L} \mathbf{V}_i^T \mathbf{W}_i^T\right)_M \\ \vdots \\ \boldsymbol{\delta}_{N_o}^T \frac{\partial}{\partial \text{vec}\left(\mathbf{W}_L^T\right)}\left(\sum_{i=0}^{L} \mathbf{V}_i^T \mathbf{W}_i^T\right)_1 \\ \vdots \\ \boldsymbol{\delta}_{N_o}^T \frac{\partial}{\partial \text{vec}\left(\mathbf{W}_L^T\right)}\left(\sum_{i=0}^{L} \mathbf{V}_i^T \mathbf{W}_i^T\right)_M \end{bmatrix} = \begin{bmatrix} \boldsymbol{\delta}_1^T \mathbf{V}_L^T & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\delta}_1^T \mathbf{V}_L^T & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\delta}_1^T \mathbf{V}_L^T & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\delta}_1^T \mathbf{V}_L^T \\ \boldsymbol{\delta}_2^T \mathbf{V}_L^T & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\delta}_2^T \mathbf{V}_L^T & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\delta}_2^T \mathbf{V}_L^T & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\delta}_2^T \mathbf{V}_L^T \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{\delta}_{N_o}^T \mathbf{V}_L^T & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\delta}_{N_o}^T \mathbf{V}_L^T \end{bmatrix} \tag{2.53}$$

$$= \begin{bmatrix} \mathbf{I}_M \otimes \boldsymbol{\delta}_1^T \mathbf{V}_L^T \\ \mathbf{I}_M \otimes \boldsymbol{\delta}_2^T \mathbf{V}_L^T \\ \vdots \\ \mathbf{I}_M \otimes \boldsymbol{\delta}_{N_o}^T \mathbf{V}_L^T \end{bmatrix}.$$

$$\begin{aligned}
&\left(\mathbf{I}_{N_o} \otimes \sum_{i=0}^{L} \mathbf{W}_i \mathbf{V}_i\right) \frac{\partial}{\partial \mathrm{vec}\left(\mathbf{W}_L^T\right)} \mathrm{vec}\left(\hat{\mathbf{Y}}(\mathbf{X})^T - \mathbf{Y}^T\right) \\
&= \left(\mathbf{I}_{N_o} \otimes \sum_{i=0}^{L} \mathbf{W}_i \mathbf{V}_i\right) \left(\mathbf{W}_{L+1} \otimes \mathbf{V}_L^T\right) \\
&= \mathbf{W}_{L+1} \otimes \sum_{i=0}^{L} \mathbf{W}_i \mathbf{V}_i \mathbf{V}_L^T.
\end{aligned} \tag{2.54}$$

2.7.2 Empirical results

In addition to the theoretical results, we also provide empirical results on image classification tasks to further understand our new principle-guided models. The goal of this empirical study is to answer the following question: *How do ResNEsts and A-ResNEsts perform compared to standard ResNets?*

Datasets

The image classification tasks chosen in our empirical study are CIFAR-10 and CIFAR-100. The CIFAR-10 dataset [Krizhevsky, 2009] consists of 60000 32×32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The CIFAR-100 dataset [Krizhevsky, 2009] is just like the CIFAR-10, except it has 100 classes containing 600 images each. The CIFAR-10 and CIFAR-100 datasets were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

Models and architectures

Every ResNEst was a standard ResNet without the batch normalization and Rectified Linear Unit (ReLU) at the final residual representation, i.e., their architectures are exactly the same before the final residual representation. Every BN-ResNEst was a standard ResNet without the ReLU at the final residual representation. In other words, a BN-ResNEst is a modified ResNEst because it adds a batch normalization layer at the final residual representation in the

ResNEst. Such a modification can avoid gradient explosion during training and allow larger learning rates to be used. For A-ResNEsts, we applied 2-dimensional average pooling on each $\mathbf{v}_i$ going into each $\mathbf{H}_i$.

The standard ResNets used in this empirical study are wide ResNet 16-8 (WRN-16-8), WRN-40-4 [Zagoruyko and Komodakis, 2016], ResNet-110, and ResNet-20 [He et al., 2016b]. All these models use pre-activation residual blocks, i.e., they are in the pre-activation form [He et al., 2016b].

Implementation details

The training procedure is exactly the same as the wide ResNet paper by Zagoruyko and Komodakis [2016]. The loss function was a cross-entropy loss. The batchsize was 128. All networks were trained for 200 epochs in total. The optimizer was stochastic gradient descent (SGD) with Nesterov momentum. The momentum was set to 0.9. The weight decay was 0.0005. The learning rate was initially set to 0.1 (0.01 for ResNEsts to avoid gradient explosion) and decreased by a factor of 5 after training 60, 120, and 160 epochs. Learning rates 0.1 and 0.05 both led to gradient explosion in training ResNEsts, so we used 0.01 for ResNEsts to avoid divergence. A-ResNEsts and BN-ResNEsts do not have such an issue.

In addition, we followed the same moderate data augmentation and preprocessing techniques in the wide ResNet paper by Zagoruyko and Komodakis [2016]. For the moderate data augmentation, a random horizontal flip and a random crop from a image padded by 4 pixels on each side are applied on the training set. For preprocessing, standardization is applied to every image including the training set and the test set. The mean and the standard deviation are computed from the training set.

Code is available at `https://github.com/kjason/ResNEst`.

Comparison

Our empirical results are summarized in the two tables below in terms of classification accuracy and number of parameters. The classification accuracy is an average of 7 trials with different initializations. The number of parameters is shown in the unit of million. "1.0M" means one million parameters.

Table 2.1. CIFAR-10.

Model / Archit.	Standard	ResNEst	BN-ResNEst	A-ResNEst
WRN-16-8	95.56% (11M)	94.39% (11M)	95.48% (11M)	95.29% (8.7M)
WRN-40-4	95.45% (9.0M)	94.58% (9.0M)	95.61% (9.0M)	95.48% (8.4M)
ResNet-110	94.46% (1.7M)	92.77% (1.7M)	94.52% (1.7M)	93.97% (1.7M)
ResNet-20	92.60% (0.27M)	91.02% (0.27M)	92.56% (0.27M)	92.47% (0.24M)

Table 2.2. CIFAR-100.

Model / Archit.	Standard	ResNEst	BN-ResNEst	A-ResNEst
WRN-16-8	79.14% (11M)	75.43% (11M)	78.99% (11M)	78.74% (8.9M)
WRN-40-4	79.08% (9.0M)	75.16% (9.0M)	78.97% (9.0M)	78.62% (8.7M)
ResNet-110	74.08% (1.7M)	69.08% (1.7M)	73.95% (1.7M)	72.53% (1.9M)
ResNet-20	68.56% (0.28M)	64.73% (0.28M)	68.47% (0.28M)	68.16% (0.27M)

A-ResNEsts empirically exhibit competitive performance to standard ResNets

Empirical results in Section 2.7.2 show that A-ResNEsts in general exhibit competitive classification accuracy with fewer parameters compared to standard ResNets; and ResNEsts are not as good as A-ResNEsts. The A-ResNets in most cases have fewer parameters than the ResNEsts and standard ResNets because they do not have the layers $\mathbf{W}_L$ and $\mathbf{W}_{L+1}$; and the number of prediction weights in $\mathbf{H}_0, \mathbf{H}_1, \cdots, \mathbf{H}_L$ is usually not larger than the number of weights in $\mathbf{W}_L$ and $\mathbf{W}_{L+1}$ (see Figure 2.1 and Figure 2.2). Note that A-ResNEsts can have more parameters than standard ResNets when the depth and the output dimension are very large, e.g., the A-ResNEst model under the architecture ResNet-110 for CIFAR-100 in Table 2.2.

A BN-ResNEst slightly outperforms a standard ResNet when the network is very deep on the CIFAR-10 dataset

Empirical results in Table 2.1 show that BN-ResNEsts slightly outperform standard ResNets and A-ResNEsts in the architectures WRN-40-4 and ResNet-110 on the CIFAR-10 dataset. For architectures WRN-16-8 and ResNet-20, BN-ResNEsts remain competitive performance compared to standard ResNets. Notice that WRN-40-4 and ResNet-110 are much deeper than WRN-16-8 and ResNet-20. Therefore, these empirical results suggest that keeping the batch normalization and simply dropping the ReLU at the final residual representation in standard pre-activation ResNets can improve the test accuracy on CIFAR-10 when the network is very deep. However, if the batch normalization at the final residual representation is also dropped, then the test accuracy is noticeably lower.

2.8 Acknowledgements

We would like to thank the anonymous reviewers for their constructive comments. This work was supported in part by NSF under Grant CCF-2124929 and Grant IIS-1838830, in part by NIH/NIDCD under Grant R01DC015436, Grant R21DC015046, and Grant R33DC015046, in part by Halıcıoğlu Data Science Institute, and in part by Wrethinking, the Foundation.

Chapter 2, in full, is a reprint of the material as it appears in K.-L. Chen, C.-H. Lee, H. Garudadri, and B. D. Rao, “ResNEsts and DenseNEsts: Block-based DNN models with improved representation guarantees,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. The dissertation author was the primary investigator and author of this material.

Chapter 3

Subspace Representation Learning for Sparse Linear Arrays to Localize More Sources than Sensors: A Deep Learning Methodology

Localizing more sources than sensors with a sparse linear array (SLA) has long relied on minimizing a distance between two covariance matrices and recent algorithms often utilize semidefinite programming (SDP). Although deep neural network (DNN)-based methods offer new alternatives, they still depend on covariance matrix fitting. In this paper, we develop a novel methodology that estimates the co-array subspaces from a sample covariance for SLAs. Our methodology trains a DNN to learn signal and noise subspace representations that are invariant to the selection of bases. To learn such representations, we propose loss functions that gauge the separation between the desired and the estimated subspace. In particular, we propose losses that measure the length of the shortest path between subspaces viewed on a union of Grassmannians, and prove that it is possible for a DNN to approximate signal subspaces. The computation of learning subspaces of different dimensions is accelerated by a new batch sampling strategy called consistent rank sampling. The methodology is robust to array imperfections due to its geometry-agnostic and data-driven nature. In addition, we propose a fully end-to-end gridless approach that directly learns angles to study the possibility of bypassing subspace methods. Numerical results show that learning such subspace representations is more beneficial than learning covariances or angles. It outperforms conventional SDP-based methods such as the sparse and parametric approach (SPA) and existing DNN-based covariance reconstruction methods for a wide range of signal-to-noise ratios (SNRs), snapshots, and source numbers for both perfect and imperfect arrays.

3.1 Introduction

Direction-of-arrival (DoA) estimation is one of the fundamental problems in array processing, providing the direction information of sources to many applications such as hearing aids [Pisha et al., 2019], wireless communications [Sant and Rao, 2020], and sonar systems [Liu et al., 2021]. When a sufficiently large number of array measurements or *snapshots* are available, most approaches estimate a spatial covariance matrix (SCM) and apply subspace methods like MUtiple SIgnal Classification (MUSIC) [Schmidt, 1986] to find the DoAs. Because the noise subspace is required to be nontrivial, an M-element uniform linear array (ULA) can only resolve up to $M-1$ sources. To remove such a limit and reduce the cost of sensors, one can choose an N-element sparse linear array (SLA) with the same aperture but no "holes" in its co-array [Van Trees, 2002]. In this case, the M-by-M SCM of the original ULA can be reconstructed from the N-by-N SCM of the SLA. Taking a 5-element minimum redundancy array (MRA) for example, it can recover the SCM of a 10-element ULA and thus resolve up to 9 sources with only 5 sensors. Although such an exploitation on the co-array structure can deliver more degrees of freedom, an extra step of covariance matrix estimation is required [Sarangi et al., 2023].

The earliest approach to this problem dates back to the work by Pillai et al. in 1985, which completes a Toeplitz matrix via redundancy averaging and direct augmentation. Since the SCM of a ULA is positive semidefinite and possibly Toeplitz, the matrix estimation problem can be formulated as constrained optimization problems under the well-known maximum likelihood (ML) principle. However, these problems are nontrivial due to being highly nonconvex, and one often needs to relax them into convex optimization problems. For example, the problem of the coarray ML-MUSIC (Co-MLM) [Qiao and Pal, 2017] is usually relaxed into the SDP problem of SPA [Yang et al., 2014] according to the extended invariance principle [Stoica and Söderström, 1989, Ottersten et al., 1998], with its global minimizer approximating the ML estimator as the number of snapshots approaches infinity. Besides convex relaxation, another strategy to tackle nonconvex optimization is majorization-minimization. For instance, the recently proposed StructCovMLE approach by Pote and Rao [2023] majorizes the concave component by a supporting hyperplane and then solves a sequence of SDP problems to arrive at a solution. There are also many other approaches such as regularized algorithms based on nuclear norm [Li and Chi, 2015] or atomic norm [Tang et al., 2014] minimization, Wasserstein distance minimization [Wang et al., 2019], and proxy covariance estimation [Sarangi et al., 2021]. Literature on DoA estimation that primarily relies on optimization techniques is vast [Wu et al., 2017, Zhou et al., 2018], so we focus on gridless and regularizer-free approaches in this paper. For grid-based DoA estimation, we refer the reader to other references such as [Stoica et al., 2010b] and [Stoica and Babu, 2012].

In the past decade, the advent of deep learning has opened up a new paradigm for DoA estimation [Liu et al., 2018, Papageorgiou et al., 2021, Barthelme and Utschick, 2021b, Chen et al., 2023a]. As the most intuitive and earliest learning-based approach, one can discretize the angle domain into a grid and then learn a classifier [Papageorgiou et al., 2021]. However, the performance of this approach is limited by the grid size and often the performance quickly saturates as the SNR increases. On the gridless side, it was not until a recent work by Wu et al. [2022] that the potential of deep learning for the matrix estimation problem was shown. Based on enforcing the Toeplitz structure of the matrix, they showed that DNNs can be trained to retrieve the noiseless SCM of a ULA from the sample SCM of an MRA, and numerical results show that such an approach outperforms the SPA in most cases. However, it was reported that its performance is worse than MUSIC when the source number is small at high SNRs. Another feature that makes the approach slightly less appealing is that a separate DNN is required for each individual source number. It is unknown whether training one DNN for all source numbers can still provide good performance. In contrast to using the Toeplitz structure, the framework proposed by Barthelme and Utschick [2021a] enforces the structure of positive definiteness of the matrix. Although in [Barthelme and Utschick, 2021a] the task of interest is subarray sampling, which is different from the present paper, the method can be applied seamlessly to the matrix estimation problem here. These two approaches are probably the most relevant related work to this paper.

In this paper, we propose a new methodology that exploits the fundamental property that a subspace is invariant of the choice of the spanning basis, and answer the following question: *Is it possible for a neural network to learn the signal or noise subspace?* In particular, we formulate the DoA estimation problem as a subspace representation learning problem, and propose new empirical risk minimization problems and loss functions to train a DNN to learn subspace representations. Our approach first constructs a DNN to output a square matrix and performs eigenvalue decomposition on the Gram matrix of the square matrix to obtain unitary bases for the signal and noise subspace, which we refer to as subspace representations. The DNN is then trained by minimizing loss functions of different dimensions based on principal angles that calculate the average degree of separation between the desired subspace and the subspace representation. In fact, with this new methodology, one can argue that learning subspaces is simpler than learning covariance matrices. Because our loss functions are invariant to the selection of bases, they create a larger solution space and thus make it easier for a DNN to learn subspace structures. Furthermore, we prove that it is possible for a neural network to approximate signal subspaces. To parallelize the computation of learning subspaces of different dimensions, we propose a new batch sampling strategy called consistent rank sampling, which greatly accelerates the training process. In addition, we propose a new gridless end-to-end approach learning DoAs directly to study the benefit of bypassing the root-MUSIC algorithm. Our methodology does not require knowledge of the sensor array positions, making it geometry-agnostic and robust to array imperfections. Under the standard assumptions of DoA estimation, numerical results show that our approach outperforms existing SDP-based and DNN-based methods across a wide range of SNRs, snapshots, and numbers of sources.

3.2 Preliminaries

Notations, assumptions, definitions, and the problem of interest are set up in this section. The set $\{1,2,\cdots,n\}$ is denoted by $[n]$. The zero-mean circularly symmetric complex Gaussian distribution with covariance $\mathbf{\Sigma}$ is denoted by $\mathscr{C}\mathscr{N}(\mathbf{0},\mathbf{\Sigma})$. The Frobenius norm of a matrix $\mathbf{A}$ is denoted by $\|\mathbf{A}\|_F$. The trace of a matrix $\mathbf{A}$ is denoted by $\operatorname{tr}(\mathbf{A})$. The set of n-by-n Hermitian matrices is denoted by $\mathbb{H}^n$. Given $\mathbf{A} \in \mathbb{H}^n$, $\mathbf{A} \succeq 0$ (resp., $\mathbf{A} \succ 0$) means that $\mathbf{A}$ is positive semidefinite (resp., positive definite). The set of n-by-n Toeplitz matrices is denoted by $\mathbb{T}^n$. For every $\mathbf{A} \in \mathbb{H}^n \cap \mathbb{T}^n$ whose first row is represented by a vector $\mathbf{u}$, $\mathbf{A}$ is denoted as $\operatorname{Toep}(\mathbf{u})$. The minimum eigenvalue of a matrix $\mathbf{A} \succeq 0$ is denoted by $\lambda_{\min}(\mathbf{A})$. The matrix logarithm of $\mathbf{A}$ is denoted by $\log(\mathbf{A})$ [Higham, 2008]. The set of all k-by-k permutation matrices is denoted by $\mathscr{P}_k$. The orthogonal projector onto a subspace $\mathscr{U}$ and the range of a matrix $\mathbf{A}$ are denoted by $P_{\mathscr{U}}$ and $P_{\mathbf{A}}$, respectively.

3.2.1 Assumptions

Let us consider an M-element ULA with spacing $d = \frac{\lambda}{2}$ centered at the origin. Assume that there are k narrowband and far-field source signals $\{s_i\}_{i=1}^k$ with a carrier wavelength λ impinging on the array from DoAs

$$\boldsymbol{\theta} = \{\theta_1, \theta_2, \cdots, \theta_k\} \subset [0,\pi]. \tag{3.1}$$

Under the plane wave assumption [Van Trees, 2002], the received array measurement vector or snapshot $\mathbf{y}(t) \in \mathbb{C}^M$ at time $t \in [T]$ can be modeled as

$$\mathbf{y}(t) = \sum_{i=1}^{k} s_i(t)\mathbf{a}(\theta_i) + \mathbf{n}(t) = \mathbf{A}(\boldsymbol{\theta})\mathbf{s}(t) + \mathbf{n}(t) \tag{3.2}$$

where $\mathbf{a}(\theta) : [0,\pi] \to \mathbb{C}^M$ is the array manifold of the M-element ULA whose i-th element is given by

$$[\mathbf{a}(\theta)]_i = e^{j2\pi\left(i-1-\frac{(M-1)}{2}\right)\frac{d}{\lambda}\cos\theta}, i \in [M] \tag{3.3}$$

and

$$\mathbf{A}(\boldsymbol{\theta}) = \begin{bmatrix} \mathbf{a}(\theta_1) & \mathbf{a}(\theta_2) & \cdots & \mathbf{a}(\theta_k) \end{bmatrix}. \tag{3.4}$$

The source signal vectors are given by

$$\mathbf{s}(t) = \begin{bmatrix} s_1(t) & s_2(t) & \cdots & s_k(t) \end{bmatrix}^{\mathsf{T}} \tag{3.5}$$

for all $t \in [T]$ and are independent and identically distributed (i.i.d.) with $\mathbf{s}(t) \sim \mathscr{CN}(\mathbf{0},\mathbf{P})$ where $\mathbf{P} = \mathrm{diag}(p_1, p_2, \cdots, p_k)$ and $p_i > 0$ is the power of the i-th source signal for all $t \in [T]$. The additive noises follow $\mathbf{n}(t) \sim \mathscr{CN}(\mathbf{0}, \eta\mathbf{I}_M)$ for all $t \in [T]$ which are i.i.d. and uncorrelated with $\mathbf{s}(t)$ for all $t \in [T]$. We further assume that $T \geq M$.

Let $N \leq M$ and $\mathscr{S} = \{s_1, s_2, \cdots, s_N\} \subset [M]$ such that $s_1 < s_2 < \cdots < s_N$. Then a physical N-element linear array can be created by removing the i-th sensor from a virtual M-element ULA if $i \notin \mathscr{S}$ for all $i \in [M]$. As a result, the snapshot $\mathbf{y}_{\mathscr{S}}(t) \in \mathbb{C}^N$ received on this physical N-element linear array at time $t \in [T]$ is given by

$$\mathbf{y}_{\mathscr{S}}(t) = \boldsymbol{\Gamma}\mathbf{y}(t) \tag{3.6}$$

where $\mathbf{y}(t)$ is the snapshot received on the virtual M-element ULA and $\boldsymbol{\Gamma} \in \mathbb{R}^{N\times M}$ is a row selection matrix given by

$$[\boldsymbol{\Gamma}]_{nm} = \begin{cases} 1, & \text{if } s_n = m, \\ 0, & \text{otherwise,} \end{cases}, n \in [N], m \in [M]. \tag{3.7}$$

In this paper, we are interested in $\mathscr{S}$ that gives rise to an SLA with the same aperture as the ULA and with no holes in its co-array such as an MRA [Van Trees, 2002] or a nested array [Pal and Vaidyanathan, 2010].

3.2.2 SCMs and the DoA estimation problem

With the above assumptions, it follows that the noiseless SCM of the ULA at every $t \in [T]$ can be written as

$$\mathbf{R}_0 = \mathbf{A}(\boldsymbol{\theta})\mathbf{P}\mathbf{A}^{\mathsf{H}}(\boldsymbol{\theta}) \tag{3.8}$$

and the noiseless SCM of the SLA is

$$\mathbf{R}_{\mathscr{S}} = \boldsymbol{\Gamma}\mathbf{R}_0\boldsymbol{\Gamma}^{\mathsf{T}}. \tag{3.9}$$

The sample SCM of the ULA and the SLA are denoted by

$$\hat{\mathbf{R}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{y}(t)\mathbf{y}^{\mathsf{H}}(t) \tag{3.10}$$

and

$$\hat{\mathbf{R}}_{\mathscr{S}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{y}_{\mathscr{S}}(t)\mathbf{y}_{\mathscr{S}}^{\mathsf{H}}(t), \tag{3.11}$$

respectively. Given M, k, $\mathscr{S}$, and $\hat{\mathbf{R}}_{\mathscr{S}}$, the goal of the DoA estimation problem is to recover $\boldsymbol{\theta}$. Note that it is possible that $k > N$ because $k \in [M-1]$. In this paper, we focus on gridless methods recovering an M-by-M matrix and use the root-MUSIC algorithm [Barabell, 1983, Rao and Hari, 1989] to find $\boldsymbol{\theta}$.

3.2.3 Neural network models

The rectified linear unit (ReLU) activation function is defined as $x \mapsto \max(0, x)$. A ReLU network can be expressed as a series of compositions of affine functions and the ReLU activation function. We follow Definition 4 for the definition of ReLU networks.

3.3 Prior art

According to the assumptions and settings in Section 3.2, we will briefly review several popular or insightful approaches in the literature including the widely used SDP-based methods and recently proposed DNN-based approaches. Despite their differences, notice that most of them fall into the category of minimizing some distance between two covariance matrices in an appropriate space. The materials covered in this section will serve as important background and contrast with our main contributions detailed in Section 3.4.

3.3.1 The maximum likelihood problem

Based on the assumptions in Section 3.2.1, it follows that $\mathbf{R}_0 + \eta \mathbf{I}_M$ is positive semidefinite and Toeplitz. Because $\mathbf{y}_{\mathscr{S}}(t) \sim \mathscr{CN}(\mathbf{0}, \mathbf{R}_{\mathscr{S}} + \eta \mathbf{I}_N)$, one can formulate the following constrained optimization problem according to the maximum likelihood principle:

$$\begin{aligned} \min_{\mathbf{R} \in \mathbb{H}^M} \quad & \log\det\left(\mathbf{\Gamma}\mathbf{R}\mathbf{\Gamma}^\mathsf{T}\right) + \operatorname{tr}\left(\left(\mathbf{\Gamma}\mathbf{R}\mathbf{\Gamma}^\mathsf{T}\right)^{-1} \hat{\mathbf{R}}_{\mathscr{S}}\right) \\ \text{subject to} \quad & \mathbf{R} \succeq 0, \quad \mathbf{R} \in \mathbb{T}^M. \end{aligned} \tag{3.12}$$

By minimizing the Kullback–Leibler divergence between $\mathscr{CN}\left(\mathbf{0}, \hat{\mathbf{R}}_{\mathscr{S}}\right)$ and $\mathscr{CN}\left(\mathbf{0}, \mathbf{\Gamma}\mathbf{R}\mathbf{\Gamma}^\mathsf{T}\right)$, one can also derive the above problem. Due to the nonconvex objective, solving (3.12) is nontrivial; and thus relaxing or refomulating (3.12) into a tractable problem is often necessary to arrive at an accepted solution. Section 3.3.2 and 3.3.3 below describe tractable optimization problems that are widely used in this context.

3.3.2 Redundancy averaging and direct augmentation

Because an SLA can generate all of the autocorrelation lags of the corresponding ULA, Pillai et al. [1985] proposed the earliest approach of recovering $\mathbf{R}_0 + \eta \mathbf{I}_M$ from $\hat{\mathbf{R}}_{\mathscr{S}}$, i.e., the so-called redundancy averaging and direct augmentation approach. This approach is identical to solving the following matrix augmentation problem [Wang et al., 2019]:

$$\min_{\mathbf{R}\in\mathbb{C}^{M\times M}} \quad \left\|\mathbf{\Gamma R \Gamma}^{\mathsf{T}} - \hat{\mathbf{R}}_{\mathscr{S}}\right\|_F \quad \text{subject to} \quad \mathbf{R}\in\mathbb{T}^M \tag{3.13}$$

which has a closed-form solution that is Hermitian and Toeplitz but not necessarily positive semidefinite. Spatial smoothing [Pal and Vaidyanathan, 2010] can be applied to fix this issue via

$$\frac{1}{M}\mathbf{R}\mathbf{R}^{\mathsf{H}} \tag{3.14}$$

if $\mathbf{R}$ is the solution of (3.13).

3.3.3 Direct SDP-based methods

Based on the covariance fitting criterion [Stoica et al., 2010a], Yang et al. [2014] formulated the SPA involving the optimization problem:

$$\begin{aligned} \min_{\mathbf{X}\in\mathbb{H}^N,\mathbf{R}\in\mathbb{H}^M} \quad & \operatorname{tr}(\mathbf{X}) + \operatorname{tr}\left(\hat{\mathbf{R}}_{\mathscr{S}}^{-1}\mathbf{\Gamma R \Gamma}^{\mathsf{T}}\right) \\ \text{subject to} \quad & \begin{bmatrix} \mathbf{X} & \hat{\mathbf{R}}_{\mathscr{S}}^{\frac{1}{2}} & \\ \hat{\mathbf{R}}_{\mathscr{S}}^{\frac{1}{2}} & \mathbf{\Gamma R \Gamma}^{\mathsf{T}} & \\ & & \mathbf{R} \end{bmatrix} \succeq 0, \quad \mathbf{R}\in\mathbb{T}^M. \end{aligned} \tag{3.15}$$

The noiseless SCM is then estimated by $\mathbf{R} - \lambda_{\min}(\mathbf{R})\,\mathbf{I}$ where $\mathbf{R}$ is the solution of (3.15). Another interesting approach based on the Bures-Wasserstein distance [Bhatia et al., 2019] was developed by Wang et al. [2019]. The optimization problem is given by

$$\begin{aligned} \min_{\mathbf{X}\in\mathbb{C}^{N\times N},\mathbf{R}\in\mathbb{H}^M} \quad & \operatorname{tr}\left(\hat{\mathbf{R}}_{\mathscr{S}} + \mathbf{\Gamma R \Gamma}^{\mathsf{T}} - \mathbf{X} - \mathbf{X}^{\mathsf{H}}\right) \\ \text{subject to} \quad & \begin{bmatrix} \mathbf{\Gamma R \Gamma}^{\mathsf{T}} & \mathbf{X} \\ \mathbf{X}^{\mathsf{H}} & \hat{\mathbf{R}}_{\mathscr{S}} \end{bmatrix} \succeq 0, \quad \mathbf{R}\succeq 0, \quad \mathbf{R}\in\mathbb{T}^M. \end{aligned} \tag{3.16}$$

Both optimization problems in (3.15) and (3.16) are SDPs that can be solved by off-the-shelf solvers such as the SDPT3 [Toh et al., 1999].

3.3.4 Majorization-Minimization

Since the term $\log\det(\cdot)$ in (3.12) is concave on the positive semidefinite cone and the trace term can be written as an SDP via the Schur complement lemma, majorization-minimization algorithms can be used to tackle (3.12). Using a supporting hyperplane to majorize the term $\log\det$, one can derive the so-called "StructCovMLE" approach [Pote and Rao, 2023]. Let $\mathbf{R}^{(0)}$ be initialized to $\mathbf{I}_M$. For $i = 0, 1, 2, \cdots$, StructCovMLE calculates the iterate $\mathbf{R}^{(i+1)}$ by solving the optimal $\mathbf{R}$ in the following SDP:

$$\begin{aligned} \min_{\mathbf{R}\in\mathbb{H}^M, \mathbf{X}\in\mathbb{H}^N} \; & \operatorname{tr}\left(\left(\mathbf{\Gamma}\mathbf{R}^{(i)}\mathbf{\Gamma}^\mathsf{T}\right)^{-1}\mathbf{\Gamma}\mathbf{R}\mathbf{\Gamma}^\mathsf{T}\right) + \operatorname{tr}\left(\mathbf{X}\hat{\mathbf{R}}_{\mathscr{S}}\right) \\ \text{subject to} \; & \begin{bmatrix} \mathbf{X} & \mathbf{I}_N & \\ \mathbf{I}_N & \mathbf{\Gamma}\mathbf{R}\mathbf{\Gamma}^\mathsf{T} & \\ & & \mathbf{R} \end{bmatrix} \succeq 0, \quad \mathbf{R} \in \mathbb{T}^M. \end{aligned} \tag{3.17}$$

The final solution is then obtained through running a number of iterations until a stopping criterion is satisfied. For example, the relative change between $\mathbf{R}^{(i)}$ and $\mathbf{R}^{(i+1)}$ being sufficiently small. As there is a sequence of SDPs to be solved, the complexity of this approach is greater than the complexity of the above direct SDP-based methods in Section 3.3.3.

3.3.5 Proxy covariance matrix estimation

Instead of estimating the covariance matrix, Sarangi et al. [2021] proposed a "proxy covariance matrix" approach (Prox-Cov) that jointly calculates a positive definite weighting matrix $\mathbf{W}$ and a proxy covariance $\mathbf{R}$ such that the weighted covariance matrix from the data best fits the proxy covariance. Based on this rationale, they formulated the following SDP:

$$\begin{aligned} \min_{\mathbf{R}\in\mathbb{H}^M,\mathbf{W}\in\mathbb{H}^T} \quad & \left\|\mathbf{Y}\mathbf{W}\mathbf{Y}^{\mathsf{H}} - \boldsymbol{\Gamma}\mathbf{R}\boldsymbol{\Gamma}^{\mathsf{T}}\right\|_F^2 \\ \text{subject to} \quad & \mathbf{R}\succeq 0, \quad \mathbf{R}\in\mathbb{T}^M, \quad \mathbf{W}\succeq \varepsilon\mathbf{I}_T \end{aligned} \tag{3.18}$$

where $\mathbf{Y} = \begin{bmatrix}\mathbf{y}(1) & \mathbf{y}(2) & \cdots & \mathbf{y}(T)\end{bmatrix}$ is a matrix whose columns are all of the received snapshots and ε is a hyperparameter which is strictly positive. An interesting property of (3.18) is that it can exactly recover the signal subspace, overcoming the shortcoming of (3.13), under appropriate assumptions [Sarangi et al., 2021]. Unlike the aforementioned methods that estimate a covariance matrix from a sample SCM, Prox-Cov considers all snapshots and attempts to estimate the signal and noise subspaces by introducing a weighting matrix $\mathbf{W}$, which allows for arbitrary signal powers while maintaining the same range space from the snapshots.

3.3.6 DNN-based covariance matrix reconstruction

Let $\mathscr{D} = \left\{\hat{\mathbf{R}}_{\mathscr{S}}^{(l)}, \mathbf{R}_0^{(l)}\right\}_{l=1}^{L}$ be a dataset containing L pairs of matrices where every $\hat{\mathbf{R}}_{\mathscr{S}}^{(l)} \in \mathbb{H}^N$ is a sample SCM of the N-element SLA and $\mathbf{R}_0^{(l)} \in \mathbb{H}^M$ is the corresponding noiseless SCM of the M-element ULA. According to the work by Barthelme and Utschick [Barthelme and Utschick, 2021a], one can formulate the matrix estimation problem as a learning problem whose goal is to find optimal parameters W^* of a DNN model $f_W : \mathbb{C}^{N\times N} \to \mathbb{C}^{M\times M}$ such that $f_{W^*}\left(\hat{\mathbf{R}}_{\mathscr{S}}\right) f_{W^*}^{\mathsf{H}}\left(\hat{\mathbf{R}}_{\mathscr{S}}\right) \approx \mathbf{R}_0$ for every possible pair $\left(\hat{\mathbf{R}}_{\mathscr{S}}, \mathbf{R}_0\right)$ of interest. The Gram matrix here is to ensure the positive semidefiniteness. The search of W^* is done through the training of the DNN. After training, the function f_{W^*} is evaluated at an N-by-N sample SCM to obtain an M-by-M SCM estimate. The model f_W is trained by solving the empirical risk minimization problem

$$\min_{W} \quad \frac{1}{L}\sum_{l=1}^{L} d\left(f_W\left(\hat{\mathbf{R}}_{\mathscr{S}}^{(l)}\right) f_W^{\mathsf{H}}\left(\hat{\mathbf{R}}_{\mathscr{S}}^{(l)}\right), \mathbf{R}_0^{(l)}\right) \tag{3.19}$$

where $d : \mathbb{C}^{M\times M} \times \mathbb{C}^{M\times M} \to [0,\infty)$ is a metric or distance. For example, the well-known Frobenius norm

$$d_{\text{Fro}}(\mathbf{E},\mathbf{F}) = \|\mathbf{E}-\mathbf{F}\|_F \tag{3.20}$$

and the *affine invariant distance* [Bhatia, 2009]

$$d_{\text{Aff}}(\mathbf{E},\mathbf{F}) = \left\|\log\left(\mathbf{F}^{-\frac{1}{2}}\mathbf{E}\mathbf{F}^{-\frac{1}{2}}\right)\right\|_F \tag{3.21}$$

that gives the length of the shortest curve(s) between the two points in the convex cone of all positive definite matrices $\{\mathbf{E} \in \mathbb{H}^M \mid \mathbf{E} \succ 0\}$. If (3.21) is used in (3.19), $\mathbf{R}_0^{(l)}$ is replaced by $\mathbf{R}_0^{(l)} + \delta\mathbf{I}_M$ for some $\delta > 0$ as $\mathbf{R}_0^{(l)}$ can be singular. Although this method by Barthelme and Utschick [2021a] was originally developed for the subarray sampling problem, we find that it is suitable for the matrix estimation problem in this paper.

An early study in the literature that tackles the matrix estimation problem using a DNN is perhaps the work by Wu et al. [2022]. Let $\mathbf{u} \in \mathbb{C}^M$ be the vector representing the first row of $\mathbf{A}(\boldsymbol{\theta})\mathbf{A}^{\mathsf{H}}(\boldsymbol{\theta})$. Instead of using the Gram matrix to generate a positive semidefinite matrix output, Wu et al. constructed a DNN $f_{W_k} : \mathbb{C}^{N\times N} \to \mathbb{C}^M$ to estimate $\mathbf{u}$ and then recovered the matrix by $\text{Toep}\left(f_{W_k}\left(\hat{\mathbf{R}}_{\mathscr{S}}\right)\right)$ for a given $\hat{\mathbf{R}}_{\mathscr{S}}$ and source number k. The models $\{f_{W_k}\}_{k=1}^{M-1}$ were trained individually by the squared loss function

$$d_{\text{squ}}(\mathbf{u},\mathbf{v}) = \frac{1}{2M}\|\mathbf{u}-\mathbf{v}\|_2^2. \tag{3.22}$$

Though $M-1$ DNNs are used in [Wu et al., 2022], note that this method is not limited by the number of DNNs used. The Toeplitz prior and d_{squ} can be used to train a single network if desired.

3.4 Subspace representation learning

A weakness of the above DNN-based methods is that their loss functions are not invariant to a different matrix representation of the signal or noise subspace. To elaborate, let $\mathbf{\Sigma} \in \mathbb{H}^K$ be any positive definite matrix such that $\mathbf{\Sigma} \neq \mathbf{P}$. Then, $\mathbf{A}(\boldsymbol{\theta})\mathbf{\Sigma}\mathbf{A}^{\mathsf{H}}(\boldsymbol{\theta})$ and $\mathbf{R}_0$ have exactly the same signal subspace $\left\{\mathbf{A}(\boldsymbol{\theta})\mathbf{x} \middle| \mathbf{x} \in \mathbb{C}^K\right\}$ that leads to the same DoAs via the root-MUSIC algorithm. However, $\mathbf{A}(\boldsymbol{\theta})\mathbf{\Sigma}\mathbf{A}^{\mathsf{H}}(\boldsymbol{\theta}) \neq \mathbf{R}_0$ which implies $d\left(\mathbf{A}(\boldsymbol{\theta})\mathbf{\Sigma}\mathbf{A}^{\mathsf{H}}(\boldsymbol{\theta}), \mathbf{R}_0\right) > 0$ for any metric or distance d on $\mathbb{C}^{M \times M}$. If $\mathbf{\Sigma} = \rho \mathbf{I}_K$, it can be easily seen that $d \to \infty$ as $\rho \to \infty$ for most of the common distances such as d_{Fro} and d_{squ} mentioned above even though the signal subspace induced by $\mathbf{A}(\boldsymbol{\theta})\mathbf{\Sigma}\mathbf{A}^{\mathsf{H}}(\boldsymbol{\theta})$ is always the same as the one induced by $\mathbf{R}_0$. This is not a desirable property for a loss function because it significantly reduces the solution space and makes it much more difficult to find and approximate the signal or noise subspace. It is worth noting that many existing methods (e.g., most of the methods in Section 3.3) measure the goodness of fit via some distance between two covariance matrices, effectively solving a harder problem than needed. Because the root-MUSIC algorithm only requires the knowledge of the signal or noise subspace, the problem of covariance estimation is actually harder than DoA estimation.

To address the above-mentioned issue, we propose a new methodology which we call *subspace representation learning*. In the subsections below, we will first introduce a new output representation for DNN models to establish the invariance to the choice of $\mathbf{\Sigma}$. Next, we construct a novel family of loss functions to train these DNN models based on the goodness of subspace fitting and show that it is possible for a DNN to approximate signal subspaces. The Root-MUSIC algorithm is then applied on the learned signal subspace to obtain the DoAs. We then discuss the use case for imperfect arrays. Finally, we propose a new batch sampling approach to parallelize the computation involved during training.

3.4.1 Subspace representations of different dimensions

Because every k-dimensional subspace $\mathscr{U}_k$ of $\mathbb{C}^M$ is a point in the *Grassmann manifold* or *Grassmannian* $\mathrm{Gr}(k,M)$, we construct a DNN model f_W such that

$$f_W : \mathbb{C}^{N\times N} \times [M-1] \to \bigcup_{k=1}^{M-1} \mathrm{Gr}(k,M) \tag{3.23}$$

whose codomain is a union of $M-1$ Grassmannians. To represent points of this union numerically, we can pick any matrix $\mathbf{U} \in \mathbb{C}^{M\times k}$ whose columns represent a unitary basis of $\mathscr{U}_k \in \mathrm{Gr}(k,M)$ for all $k \in [M-1]$. Based on this perspective, the model f_W is instructed to generate a matrix $\mathbf{X} \in \mathbb{C}^{M\times M}$ whose Gram matrix is factorized by eigenvalue decomposition:

$$\mathbf{X}\mathbf{X}^{\mathsf{H}} = \begin{bmatrix} \tilde{\mathbf{U}} & \tilde{\mathbf{V}} \end{bmatrix} \begin{bmatrix} \mathbf{\Lambda}_k & \\ & \mathbf{\Lambda}_{M-k} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{U}}^{\mathsf{H}} \\ \tilde{\mathbf{V}}^{\mathsf{H}} \end{bmatrix} \tag{3.24}$$

where $\mathbf{\Lambda}_k$ and $\mathbf{\Lambda}_{M-k}$ are diagonal matrices representing the k largest eigenvalues and $M-k$ smallest eigenvalues, respectively; and the columns of $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ are their corresponding orthonormal eigenvectors, respectively. Since the columns of $\tilde{\mathbf{U}} \in \mathbb{C}^{M\times k}$ forms a unitary basis, a subspace $\tilde{\mathscr{U}_k} \in \mathrm{Gr}(k,M)$ can then be identified by the range space of $\tilde{\mathbf{U}}$ and thus the function f_W can generate points in the union of the Grassmannians. As long as $\mathbf{X}$ maintains the same signal subspace, the subspace $\tilde{\mathscr{U}}$ generated by f_W is invariant to the change of $\mathbf{X}$. One simple invariance can be easily seen by changing the eigenvalues while maintaining the order of $\mathbf{\Lambda}_k$ and $\mathbf{\Lambda}_{M-k}$. The subspace $\tilde{\mathscr{U}}$ is also invariant to the equivalence class of its unitary bases.

Given a dataset $\mathscr{D} = \left\{\hat{\mathbf{R}}_{\mathscr{S}}^{(l)}, \mathbf{R}_0^{(l)}\right\}_{l=1}^{L}$, we extract the signal subspace $\mathscr{U}^{(l)}$ of $\mathbf{R}_0^{(l)}$ for every $l \in [L]$ via eigenvalue decomposition to create target subspace representations. Note that $\mathscr{U}^{(l)}$ can also be identified from $\mathbf{A}\left(\boldsymbol{\theta}^{(l)}\right)\mathbf{A}^{\mathsf{H}}\left(\boldsymbol{\theta}^{(l)}\right)$ if only a dataset of $\left\{\hat{\mathbf{R}}_{\mathscr{S}}^{(l)}, \boldsymbol{\theta}^{(l)}\right\}$ is given.

3.4.2 Distances between subspace representations

To learn the target subspace representations in $\mathscr{D}$, we find the parameters W by solving the following empirical risk minimization problem

$$\min_{W} \quad \frac{1}{L}\sum_{l=1}^{L} d_{k=k^{(l)}}\left(f_W\left(\hat{\mathbf{R}}_{\mathscr{S}}^{(l)}, k^{(l)}\right), \mathscr{U}^{(l)}\right) \tag{3.25}$$

where $d_k : \mathrm{Gr}(k,M) \times \mathrm{Gr}(k,M) \to [0,\infty)$ is some distance on the Grassmannian $\mathrm{Gr}(k,M)$. We propose to construct d_k as a function of the vector of *principal angles* between two given subspaces because it is a necessary condition if d_k is invariant to any rotation in the unitary group $\mathbb{U}(M)$ of M-by-M unitary matrices [Wong, 1967], i.e.,

$$d_k\left(\mathbf{Q}\cdot\mathscr{U}, \mathbf{Q}\cdot\tilde{\mathscr{U}}\right) = d_k\left(\mathscr{U}, \tilde{\mathscr{U}}\right) \tag{3.26}$$

for every $\mathscr{U}, \tilde{\mathscr{U}} \in \mathrm{Gr}(k,M)$ and every $\mathbf{Q} \in \mathbb{U}(M)$. The *left action* of $\mathbb{U}(M)$ on $\mathrm{Gr}(k,M)$ in (3.26) is defined by $\mathbf{Q}\cdot\mathscr{U} := \mathrm{span}\left(\mathbf{QB}\right)$ where the columns of $\mathbf{B} \in \mathbb{C}^{M\times k}$ form a basis of $\mathscr{U}$. According to Theorem 1 of [Björck and Golub, 1973], the principal angles $\boldsymbol{\phi}_k = \begin{bmatrix}\phi_1 & \phi_2 & \cdots & \phi_k\end{bmatrix}^{\mathsf{T}}$ between $\mathscr{U} \in \mathrm{Gr}(k,M)$ and $\tilde{\mathscr{U}} \in \mathrm{Gr}(k,M)$ can be calculated by

$$\phi_i\left(\mathscr{U}, \tilde{\mathscr{U}}\right) = \cos^{-1}\left(\sigma_i\left(\mathbf{U}^{\mathsf{H}}\tilde{\mathbf{U}}\right)\right) \tag{3.27}$$

for $i \in [k]$ where $\mathbf{U} \in \mathbb{C}^{M\times k}$ and $\tilde{\mathbf{U}} \in \mathbb{C}^{M\times k}$ are matrices whose columns form unitary bases of $\mathscr{U}$ and $\tilde{\mathscr{U}}$, respectively, and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k$ are the singular values of the singular value decomposition of $\mathbf{U}^{\mathsf{H}}\tilde{\mathbf{U}}$. Therefore, one is able to define many distances based on $\boldsymbol{\phi}_k$ [Edelman et al., 1998, Barg and Nogin, 2002, Hamm and Lee, 2008, Ye and Lim, 2016]. Among them, the most natural choice of d_k is the *geodesic distance* [Wong, 1967]

$$d_k^{\mathrm{Geo}}\left(\mathscr{U}, \tilde{\mathscr{U}}\right) = \left\|\boldsymbol{\phi}_k\left(\mathscr{U}, \tilde{\mathscr{U}}\right)\right\|_2 = \left(\sum_{i=1}^{k}\phi_i^2\left(\mathscr{U}, \tilde{\mathscr{U}}\right)\right)^{\frac{1}{2}} \tag{3.28}$$

which defines the length of the shortest curve(s) between the two points $\mathscr{U}$ and $\tilde{\mathscr{U}}$ on the Grassmannian $\mathrm{Gr}(k,M)$. The geodesic distance of any two points on $\mathrm{Gr}(k,M)$ is bounded from above by $\sqrt{k}\frac{\pi}{2}$ [Wong, 1967]; and one can easily construct different loss functions which are bounded.

3.4.3 Approximation

In this subsection, we attempt to enhance the feasibility of subspace representation learning from an approximation viewpoint. In particular, we present a guarantee for a neural network model to approximate the signal subspace.

Theorem 6. *For every $k \in [M-1]$ and every $\varepsilon > 0$, there exists a ReLU network $f : \mathbb{C}^{N\times N} \to Gr(k,M)$ such that*

$$\int_{[0,\pi]^k} d_k^{Geo}\left(f\left(\mathbf{R}_{\mathscr{S}}\right), P_{\mathbf{A}(\boldsymbol{\theta})}\right) d\boldsymbol{\theta} < \varepsilon. \tag{3.29}$$

The proof of Theorem 6 is contained in Appendix 3.8.1. Here, subspaces are represented by their orthogonal projectors to ensure every $\mathscr{U} \in \mathrm{Gr}(k,M)$ has a unique representation. In other words, $\mathrm{Gr}(k,M)$ is equivalent to

$$\left\{P \in \mathbb{C}^{M\times M} \mid P^{\mathsf{H}} = P, P^2 = P, \mathrm{rank}(P) = k\right\}. \tag{3.30}$$

If the ideal covariance matrices are used, Theorem 6 shows that the average geodesic distance between the predicted subspaces and the desirable signal subspaces can be made arbitrarily small when a suitable ReLU network is picked. From an array processing point of view, it is trivial that the signal subspace can always be extracted from $\mathbf{R}_{\mathscr{S}}$. However, Theorem 6 illustrates that this process can be achieved up to a small error by evaluating a continuous piecewise linear function [Chen et al., 2022]. In order to sketch the proof, notice that a simple distance on $\mathrm{Gr}(k,M)$ can be constructed by $(\mathscr{U}_1,\mathscr{U}_2) \mapsto \left\|P_{\mathscr{U}_1} - P_{\mathscr{U}_2}\right\|_F$. Lemma 14 below shows that the geodesic distance can be bounded from above by the composition of a strictly increasing function and the simple distance, allowing us to leverage the continuity of the orthogonal projection operator in an appropriate manner to prove Theorem 6. It may be possible to extend Theorem 6 to a more realistic case using $\hat{\mathbf{R}}_{\mathscr{S}}$ with a probabilistic guarantee.

Lemma 14. *For every $\mathscr{U}_1, \mathscr{U}_2 \in Gr(k, M)$ where $k \in [M-1]$,*

$$d_k^{Geo}(\mathscr{U}_1, \mathscr{U}_2) \leq \sqrt{k} \sin^{-1}\left(\frac{\left\|P_{\mathscr{U}_1} - P_{\mathscr{U}_2}\right\|_F}{\sqrt{2}}\right). \tag{3.31}$$

The proof of Lemma 14 is contained in Appendix 3.8.2. Lemma 14 ensures that

$$\left\|P_{\mathscr{U}_1} - P_{\mathscr{U}_2}\right\|_F \to 0 \quad \text{implies} \quad d_k^{\mathrm{Geo}}(\mathscr{U}_1, \mathscr{U}_2) \to 0. \tag{3.32}$$

3.4.4 Learning with imperfect arrays

Sensor arrays are not perfect in reality. For example, the array manifold may be corrupted by several imperfections including the gain bias, phase bias, sensor position error, and the intersensor mutual coupling [Liu et al., 2018]. Because model-based methods such as SDP-based approaches in (3.15) and (3.16) often rely on prior knowledge of the sensor positions $\mathscr{S}$ to create $\mathbf{\Gamma}$ in their optimization problems, they are not robust to sensor position errors; and fixing such a model mismatch is nontrivial. In contrast, our methodology does not suffer from this model mismatch issue due to its geometry-agnostic or imperfection-agnostic nature. The empirical risk minimization problem we solve in the imperfect array case is still (3.25). As described in Section 3.4.1, $\mathscr{U}^{(l)}$ can be identified from the ground truth $\boldsymbol{\theta}^{(l)}$; and $\hat{\mathbf{R}}_{\mathscr{S}}$ is the sample SCM from the imperfect array. Hence, both the problem formulation in (3.25) and the model (3.23) do not depend on the sensor positions. In addition, our method does not need to know the array is imperfect and the degree of imperfections. The information is already embedded in the dataset and solving (3.25) will enforce the DNN model learn the subspace representations of a perfect virtual ULA from the imperfect array.

3.4.5 Consistent rank sampling

To learn subspaces of different dimensions in one DNN model, the empirical risk minimization problem (3.25) requires $M-1$ loss functions $d_1, d_2, \cdots, d_{M-1}$ that calculate unitary bases of different dimensions from 1 to $M-1$. Although (3.25) can be solved by the well-known minibatch stochastic gradient descent (SGD) algorithm, it is hard for the computation of different dimensions to be parallelized on a graphics processing unit (GPU). To fix this issue, we propose *consistent rank sampling*, a new batch sampling strategy for learning subspaces of different dimensions in one DNN model. Instead of randomly sampling from $\mathscr{D}$, we propose randomly sampling a batch of data points whose source number k is consistent from $\mathscr{D}$. This way, only one d_k needs to be evaluated in every gradient step, streamlining the computation of unitary bases in the same dimension k. It is important to note that consistent rank sampling is a crucial strategy to make training efficient. Without this strategy, training DNNs on large datasets is almost intractable due to slow training speed. Although the strategy is developed for subspace representation learning, it is generally applicable to empirical risk minimization problems that involve loss functions of different dimensions.

3.5 A gridless end-to-end approach

The subspace representation learning approach utilizes the root-MUSIC algorithm on the obtained subspaces to estimate the DoAs. A natural question to study here is the following: *Is it possible to bypass the root-MUSIC algorithm and directly learn a model to output the DoAs in a gridless manner?* The best-known end-to-end approach is probably the work by Papageorgiou et al. [2021]; however, it relies on a grid. The approach of mean cyclic error (MCE) network or MCENet by Barthelme and Utschick [2021b] is a gridless end-to-end approach but it was designed for subarray sampling and not for more sources than sensors. Below, we propose a new gridless end-to-end approach that tailored to localization of more sources than sensors using an SLA.

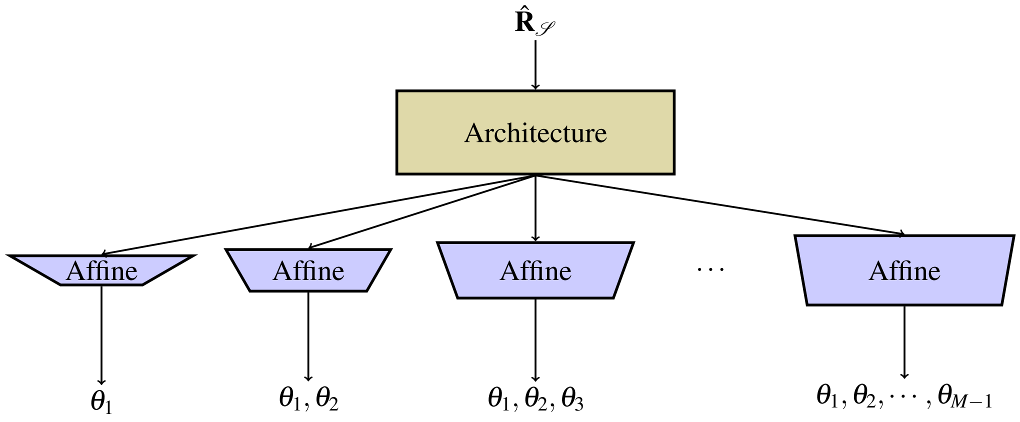

$\hat{\mathbf{R}}_{\mathscr{S}}$
Architecture
Affine
Affine
Affine
$\cdots$
Affine
θ_1
θ_1, θ_2
$\theta_1, \theta_2, \theta_3$
$\theta_1, \theta_2, \cdots, \theta_{M-1}$

Figure 3.1. An illustration of the gridless end-to-end model, which consists of an architecture and several output layers. The model simultaneously generates DoAs for every possible number of sources so there are $M-1$ heads (affine functions) at the output. The k-th head is picked when there are k sources.

As illustrated in Figure 3.1, we propose to construct a DNN model g_W such that

$$g_W : \mathbb{C}^{N \times N} \times [M-1] \to \mathbb{R}^1 \times \mathbb{R}^2 \times \cdots \mathbb{R}^{M-1} \tag{3.33}$$

whose codomain is the $(M-1)$-ary Cartesian product of Euclidean spaces $\mathbb{R}^1, \mathbb{R}^2, \cdots, \mathbb{R}^{M-1}$. These Euclidean spaces are viewed as different "heads" at the output of the model where the k-dimensional Euclidean space represents the k-th head. The k-th head will be picked when there are k sources such that an element from $\mathbb{R}^k$ can represent k angles. Let $r(i) = \frac{i(i-1)}{2}$ for $i = 1, 2, \cdots, M$ and denote $h_k : \mathbb{R}^{r(M)} \to \mathbb{R}^k$ the projection

$$\left(x_1, \cdots, x_{r(M)}\right) \mapsto \left(x_{r(k)+1}, x_{r(k)+2}, \cdots, x_{r(k)+k}\right). \tag{3.34}$$

The empirical risk minimization problem of the gridless end-to-end model g_W is then formulated as follows

$$\min_{W} \quad \frac{1}{L} \sum_{l=1}^{L} d_{k=k^{(l)}} \left(h_k \circ g_W \left(\hat{\mathbf{R}}_{\mathscr{S}}^{(l)}, k^{(l)} \right), \boldsymbol{\theta}^{(l)} \right) \tag{3.35}$$

where $d_1, d_2, \cdots, d_{M-1}$ are loss functions of different dimensions that calculate some minimum distances among all permutations. Taking the squared loss for example,

$$d_k\left(\hat{\boldsymbol{\theta}},\boldsymbol{\theta}\right) = \min_{\boldsymbol{\Pi}\in\mathscr{P}_k} \left\|\boldsymbol{\Pi}\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\right\|_2^2 \tag{3.36}$$

for $k = 1, 2, \cdots, M-1$. The minimum in (3.36) is equivalent to the squared loss applied to the corresponding sorted arguments according to the rearrangement inequality [Hardy et al., 1934]. Because a loss function of different dimensions is adopted, the consistent rank sampling strategy detailed in Section 3.4.5 can be applied to accelerate training on GPUs.

3.6 Numerical results

In this section, we will compare our new methodology with existing methods including the SPA [Yang et al., 2014], the Wasserstein distance based approach (WDA) [Wang et al., 2019], the DNN-based covariance reconstruction (DCR) approach based on the Toeplitz prior [Wu et al., 2022] (DCR-T), and the DCR approach based on the Gram matrix [Barthelme and Utschick, 2021a] (DCR-G). In particular, we use both the Frobenius norm and the affine invariant distance for DCR-G, leading to two methods termed DCR-G-Fro and DCR-G-Aff. We do not include StructCovMLE [Pote and Rao, 2023] and Prox-Cov [Sarangi et al., 2021] because the performance of StructCovMLE was similar to SPA, and Prox-Cov did not yield better performance than the SPA in our preliminary experiments. Below, we will first set up the scenarios for the DoA estimation problem. Next, we will describe the DNN architectures and the training procedures for the DNN-based approaches. Finally, for a given SNR and number of snapshots T, we will compare performance of different approaches in terms of the mean squared error (MSE)

$$\frac{1}{L_{\text{test}}}\sum_{l=1}^{L_{\text{test}}}\frac{1}{k}\min_{\boldsymbol{\Pi}\in\mathscr{P}_k}\left\|\boldsymbol{\Pi}\hat{\boldsymbol{\theta}}_l - \boldsymbol{\theta}_l\right\|_2^2 \tag{3.37}$$

for different source numbers $k \in [M-1]$ where L_{test} is the total number of random trials, $\boldsymbol{\theta}_l$ is the vector of DoAs of the ground truth at the l-th trial, and $\hat{\boldsymbol{\theta}}_l$ is the corresponding estimate given by a method of interest. Code is available at `https://github.com/kjason/SubspaceRepresentationLearning`.

3.6.1 Settings

The physical array is an N-element MRA with $N = 5$ and $\mathscr{S} = \{1,2,5,8,10\}$, leading to a 10-element virtual ULA or $M = 10$. A study for different MRAs is deferred to Section 3.6.2. Below we describe the test or evaluation conditions. The number of snapshots T is set to 50, if not explicitly specified. We assume equal source powers $p_1 = p_2 = \cdots = p_k$ and the SNR is defined as $10\log_{10}\left(\frac{p_i}{\eta}\right)$. The SNR is set to 20 dB if not explicitly stated. The finite set of SNRs $\{-10,-8,-6,\cdots,16,18,20\}$ is picked when a range of SNRs is required for evaluation. The number of sources k can be any number in the set $[M-1]$. For any $k \in [M-1]$, the DoAs $\theta_1,\theta_2,\cdots,\theta_k$ are uniformly selected at random in the range $\left[\frac{1}{6}\pi,\frac{5}{6}\pi\right]$ with a minimum separation constraint $\min_{i\neq j}|\theta_i-\theta_j| \geq \frac{1}{45}\pi$. For every given SNR, T, and $k \in [M-1]$, there are 100 trials of random source signals and noises for a given $\boldsymbol{\theta}$, and there are in total 100 random $\boldsymbol{\theta}$, leading to a total number of trials $L_{\text{test}} = 10^4$ for each case. All SDP problems are solved by the SDPT3 [Toh et al., 1999] solver in CVX [Grant and Boyd, 2014, 2008].

DNN models

As illustrated in Figure 3.2, we use WRN-16-8 [Zagoruyko and Komodakis, 2016] without the batch normalization. The pair of numbers 16-8 implies that the total number of layers is 16 and the widening factor is 8. The ReLU activation function is adopted by all of the nonlinearities in the network. All of the residual blocks are in the pre-activation form [He et al., 2016b]. Note that wide ResNets avoid the degradation problem and enjoy certain optimization guarantees under mild assumptions [Chen et al., 2021]. The network takes an input tensor in $\mathbb{R}^{2\times N\times N}$ and generates an output tensor in $\mathbb{R}^{2\times M\times M}$ ($\mathbb{R}^{2\times M}$ for DCR-T). Given an N-by-N complex matrix, it is represented by its real and imaginary parts as inputs to the network. The first and second planes of the output tensor represent the real and imaginary parts of an M-by-M

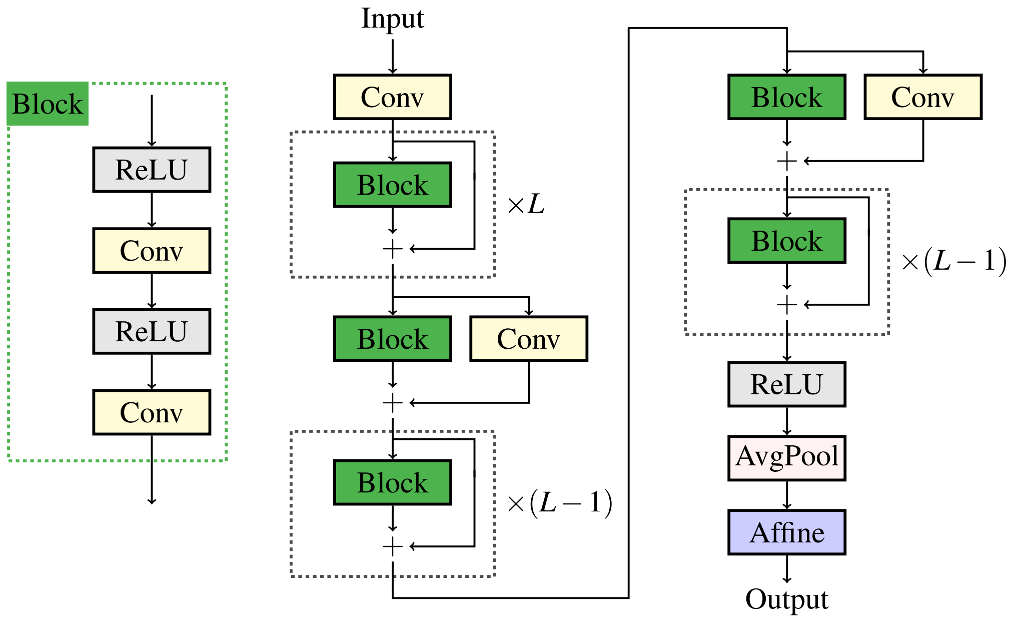


Figure 3.2. An illustration of a 3-stage L-block ResNet model [He et al., 2016a]. In the wide ResNet 16-8 (WRN-16-8) [Zagoruyko and Komodakis, 2016], there are $L = 2$ blocks per stage, leading to 16 layers in total. The widening factor is 8, meaning that WRN-16-8 is 8 times wider than the original ResNet. See Section 3.6.1 for more details.

complex matrix, respectively. The number of parameters is approximately 11 million. All DNN-based methods use the same architecture. The output layer is an affine function whose output dimension is tailored to each approach.

Training

The minibatch SGD algorithm with Nesterov momentum is used to train all of the DNN models. The momentum is set to 0.5 and the batch size is 4096. The weight decay is set to 0. All of the models are trained for 50 epochs with the one-cycle learning rate scheduler [Smith and Topin, 2019]. The best maximum learning rate of the scheduler for each approach is found through a grid search whose description is deferred to Appendix 3.8.3. The learning rates for DCR-T, DCR-G-Fro, DCR-G-Aff, and our approach are 0.05, 0.01, 0.005, and 0.1, respectively. The weights in all models are initialized using normal distributions [He et al., 2015]. The value of δ is set to 10^{-4} in the DCR-G-Aff approach. For each $k \in [M-1]$, there are 2×10^6 and 6×10^5 random data points for training and validation, respectively, leading to a training dataset of size $L_{\text{train}} = 9 \times 2 \times 10^6$ and a validation dataset of size $L_{\text{val}} = 9 \times 6 \times 10^5$. For each data point, the source signals and noises are generated randomly according to the assumptions in Section 3.2.1. The SNR in decibels is uniformly picked at random in the finite set $\{-11, -9, -7, \cdots, 17, 19, 21\}$. The DoAs in the vector $\boldsymbol{\theta}$ are uniformly selected at random in the range $\left[\frac{1}{6}\pi, \frac{5}{6}\pi\right]$ with a minimum separation constraint $\min_{i \neq j} |\theta_i - \theta_j| \geq \frac{1}{60}\pi$. The number of snapshots is set to 50. PyTorch is used to train all the DNN models [Paszke et al., 2019].

3.6.2 Results

Superior performance over a wide range of SNRs

Figure 3.3 compares the proposed method with the five baseline approaches in terms of MSE over a wide range of SNRs and number of sources. For $k = 1$, all of the methods have almost the same performance. For $k = 2$, the proposed method is significantly better than SPA and WDA from -10 to 6 dB SNR. In fact, it is uniformly better than WDA from -10 to 20 dB SNR. However, once the SNR goes beyond 14 dB, SPA starts to outperform the proposed method and the gap seems to become larger as the SNR increases. DCR-T is slightly worse than the proposed method in the high SNR region but the gap of MSE gets larger as SNR increases. With regard to DCR-G-Fro and DCR-G-Aff, their performance is similar to the proposed method. For $k = 3$, the proposed method is better than SPA, WDA, DCR-T, and DCR-G-Fro across almost the entire evaluation range and even superior by orders of magnitude from 0 to 15 dB SNR with respect to SPA, WDA, and DCR-T. DCR-G-Aff is slightly better than the proposed method in the high SNR region but is slightly worse in the low SNR region. Then, for $k \in \{4, 5, 6, 7, 8, 9\}$, the proposed method consistently and significantly outperforms SPA and WDA. As for DCR-T and DCR-G-Fro, they are noticeably better than SPA and WDA but significantly inferior than the proposed method. In particular, DCR-G-Aff is the most competitive approach to the proposed method. However, it is still much inferior than our approach. Overall, the proposed method is significantly better than all of the baseline approaches.

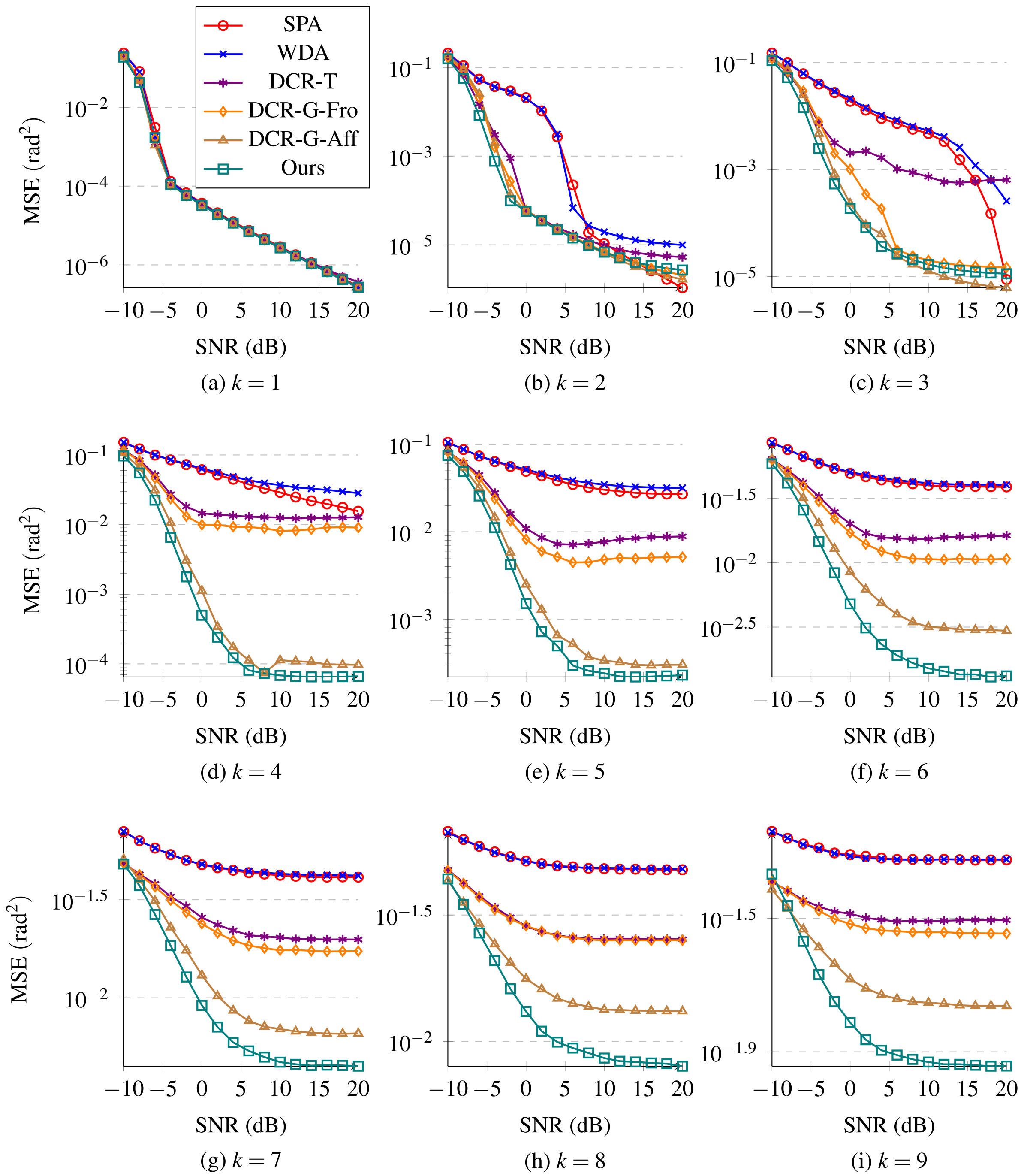


Figure 3.3. MSE vs. SNR. Our approach is in general superior to all of the baselines. In most cases, it is significantly better than SPA, WDA, DCR-T, and DCR-G-Fro. DCR-G-Aff is the most competitive baseline. For $k > 3$, our approach outperforms DCR-G-Aff. In comparison to DCR-G-Aff at $k = 2$ or $k = 3$, our approach is slightly better at low SNRs but worse at high SNRs.

Generalization to unseen numbers of snapshots

Figure 3.4 evaluates SPA, WDA, DCR-G-Aff, and the proposed method in terms of MSE in a wide range of numbers of snapshots and sources. We do not include the other baselines here because DCR-G-Aff is significantly better than them according to Figure 3.3. For $k = 1$, all of the methods have similar performance. For $k = 6$ and $k = 9$, the proposed method is consistently and significantly better than SPA, WDA, and DCR-G-Aff. More importantly, Figure 3.4 also implies that a DNN model trained by the subspace representation learning approach on a specific number of snapshots can generalize well to a wide range of unseen numbers of snapshots.

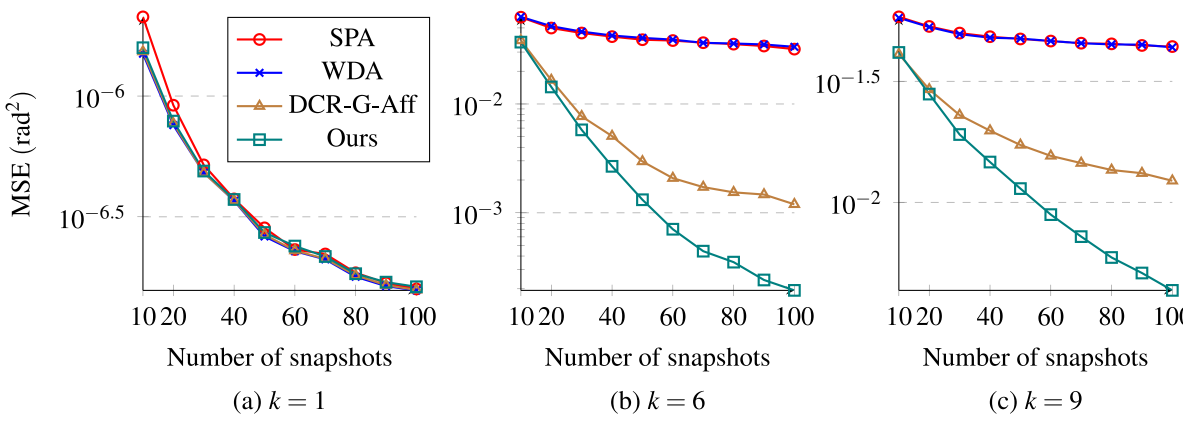


Figure 3.4. MSE vs. number of snapshots. Although the DNN models are only trained on a single number of snapshots $T = 50$, they are able to generalize to a wide range of unseen scenarios from $T = 10$ to $T = 100$. Our approach is consistently better than SPA, WDA, and DCR-G-Aff.

On different SLAs

It is desirable to show that the main conclusions drawn from the 5-element MRA experiments in Section 3.6.2 in general hold true for an arbitrary N-element MRA. Here, we demonstrate that this is true for 4-element and 6-element MRAs. Most of the hyperparameters stay the same as the setting in Section 3.6. We find that $\delta = 10^{-4}$ leads to unstable training in the DCR-G-Aff approach for the case of the 6-element MRA so we increase δ to 10^{-3} in this particular case. Results for the 4-element MRA $\mathscr{S} = \{1, 2, 5, 7\}$ are shown in Figure 3.5.

Results for the 6-element MRA $\mathscr{S} = \{1, 2, 5, 6, 12, 14\}$ are shown in Figure 3.6. All of these results in Figure 3.3, 3.5, and 3.6 demonstrate that the proposed method outperforms all of the baseline approaches. Furthermore, they seem to suggest our approach is consistently better than all baselines if $k \geq N$. Figure 3.7 and 3.8 show the results on the number of snapshots for the 4-element and 6-element MRAs, they imply conclusions that are similar to ones drawn from Figure 3.4.

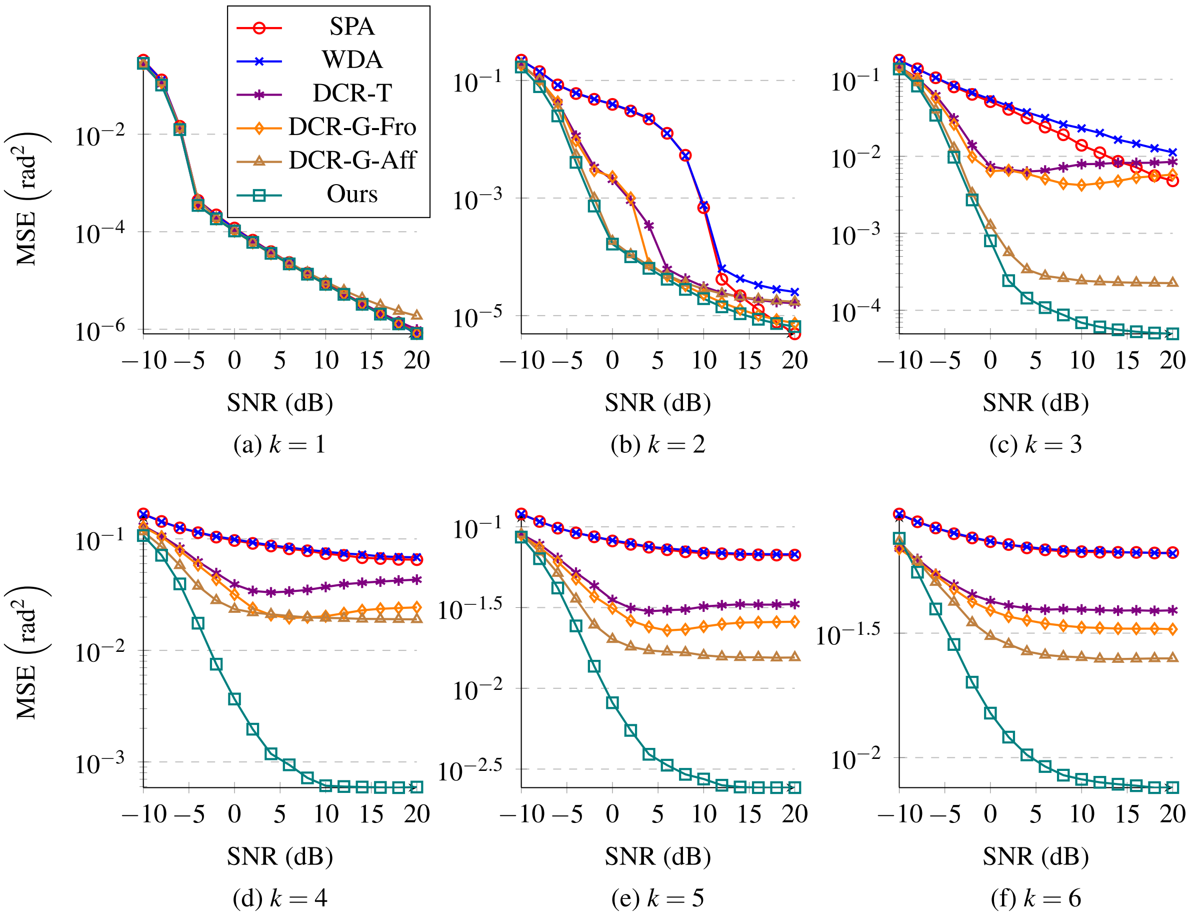


Figure 3.5. MSE vs. SNR. $N = 4$. $M = 7$. Our approach is significantly better than all of the baselines when $k > 2$. For $k = 2$, it is better than all of the DNN-based baselines but slightly worse than the SPA at 20 dB SNR. The main results obtained for the 5-element MRA are similar to the 4-element MRA.

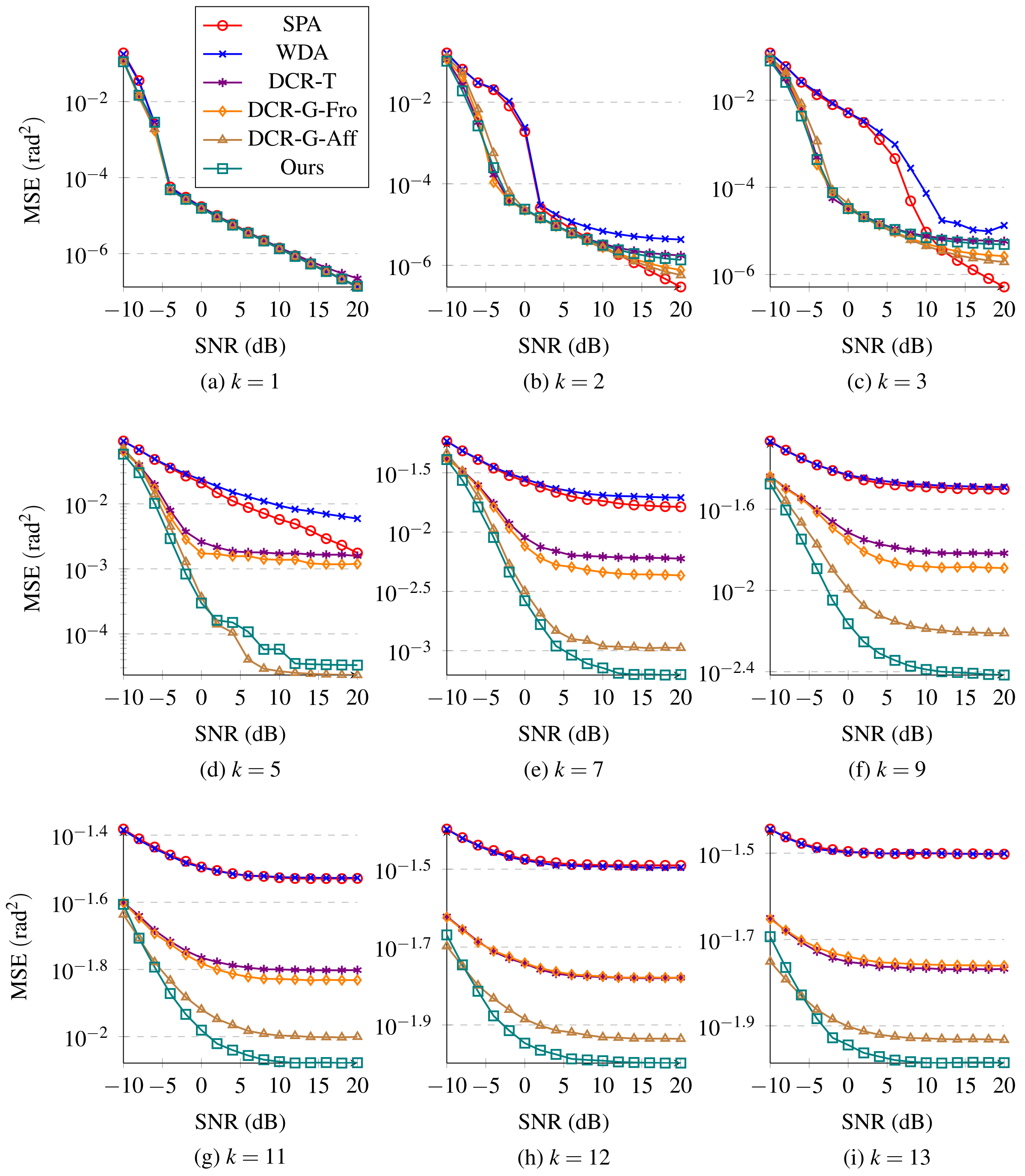


Figure 3.6. MSE vs. SNR. $N = 6$. $M = 14$. These results, along with Figure 3.3 and 3.5, imply that the proposed method consistently outperforms all of the baselines if $k \geq N$. The proposed method is slightly inferior than DCR-G-Aff at high SNRs when $k < N$.

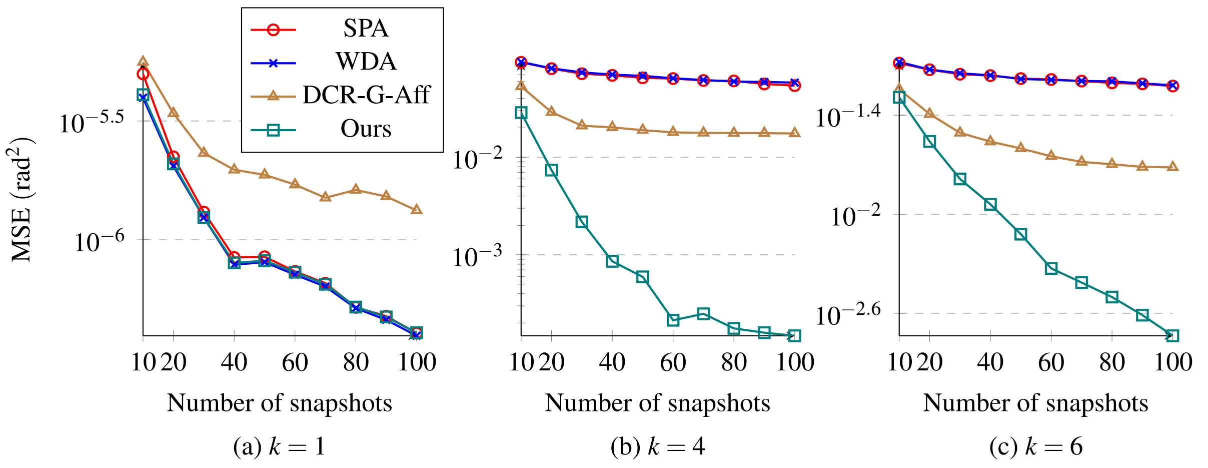


Figure 3.7. MSE vs. number of snapshots. $N = 4$. $M = 7$.

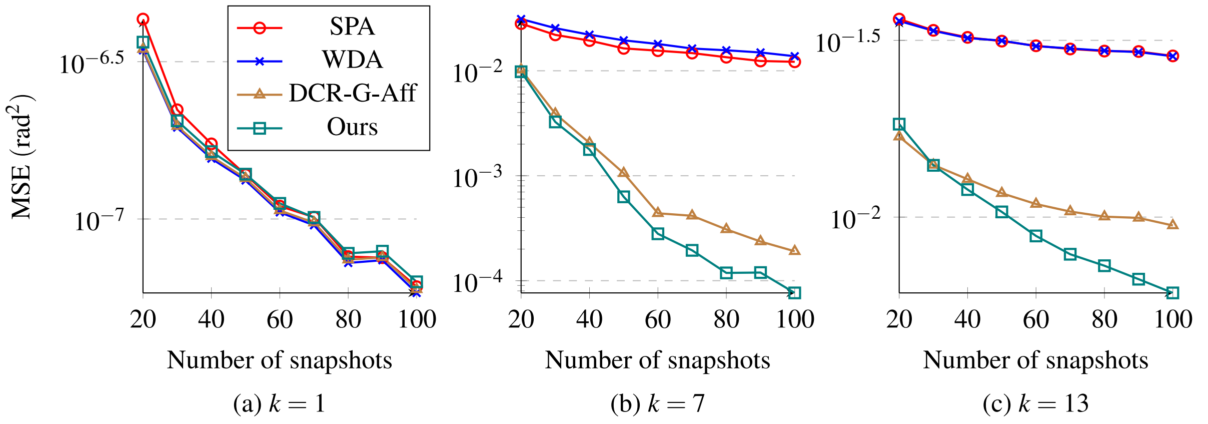


Figure 3.8. MSE vs. number of snapshots. $N = 6$. $M = 14$.

3.6.3 Comparison to the proposed gridless end-to-end approach

To answer the question posed in Section 3.5, we use the same WRN-16-8 but replace the final affine layer by $M-1$ affine heads whose number of output neurons are $1,2,3,\cdots,M-1$ real numbers, as illustrated in Figure 3.1. The squared loss functions of different dimensions are adopted as shown in (3.36). All of the settings here are the same as the ones described in Section 3.6.1 and 3.6.1. The best learning rate is 0.2 according to a simple grid search in Section 3.8.3. Figure 3.9 shows that the gridless end-to-end approach tends to saturate its performance earlier than the subspace representation learning approach for $k < 8$ as the SNR increases. As a result, the subspace representation learning approach shows significantly better performance at

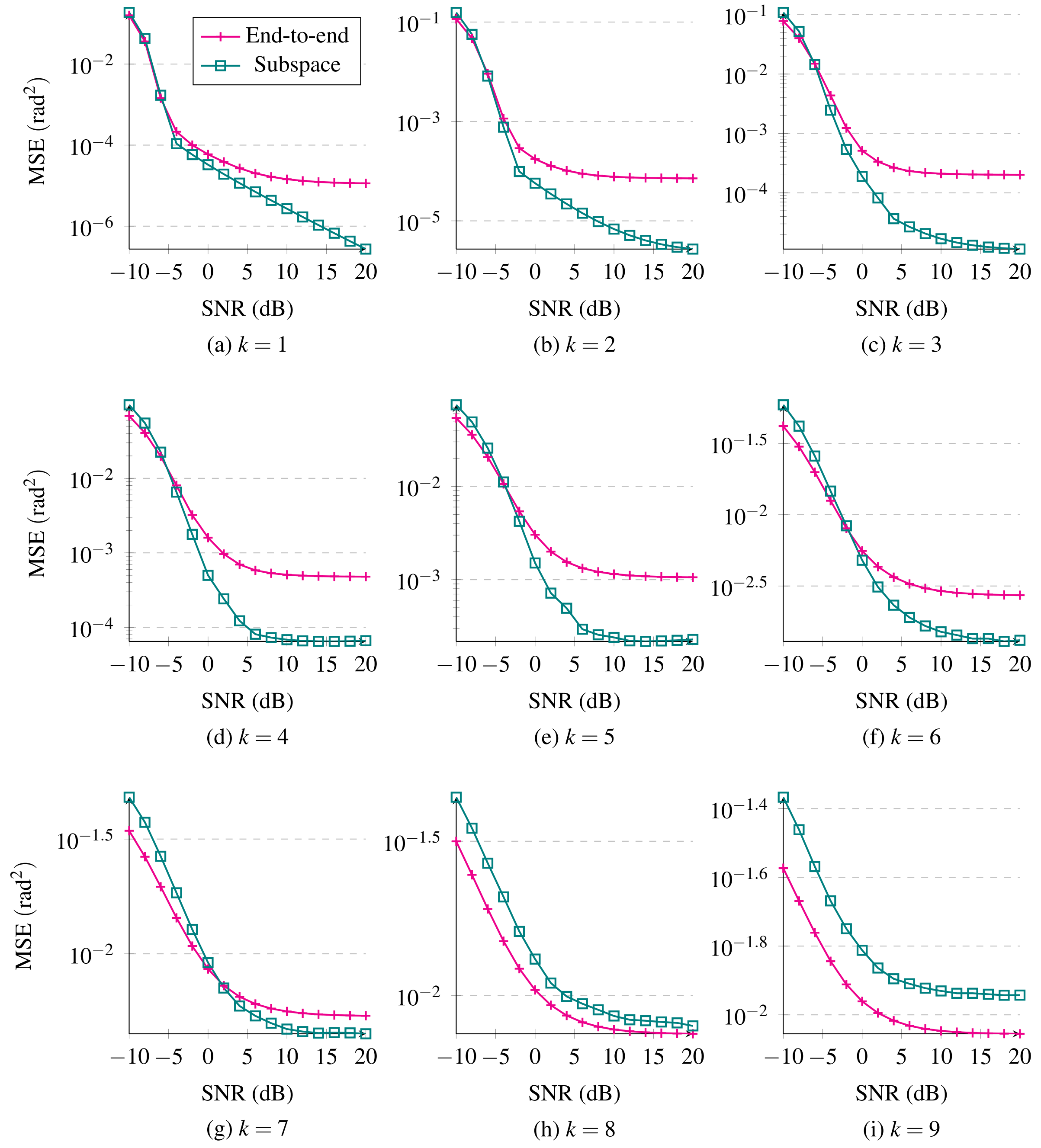


Figure 3.9. MSE vs. SNR. $N = 5$. $M = 10$. For $k < 8$, the performance of the gridless end-to-end approach saturates at a higher MSE than the subspace representation learning method as the SNR increases. For $k = 8$ and $k = 9$, the gridless end-to-end approach shows consistently better performance.

high SNRs. However, for $k = 8$ and $k = 9$, it is consistently worse than the gridless end-to-end approach. Although the gridless end-to-end approach does not have a grid at the output layer, its behavior of hitting an early plateau seems to be similar to grid-based methods that are limited by their grid resolution. Overall, subspace representation learning gives better performance than the gridless end-to-end approach and we can deduce that learning subspace representations is more beneficial than learning angles directly.

3.6.4 Robustness to array imperfections

With regard to array imperfections, we use the imperfect array manifold introduced by Liu et al. [2018]. The exact formulation is given below. Define the following real hyperparameters

$$e_1, \cdots, e_M, g_1, \cdots, g_M, h_1, \cdots, h_M \tag{3.38}$$

and a complex hyperparameter γ. The array manifold with sensor position errors is given by $\mathbf{a}_\rho(\theta) : [0, \pi] \to \mathbb{C}^M$ such that

$$[\mathbf{a}_\rho(\theta)]_i = e^{j2\pi\left(i-1-\frac{(M-1)}{2}+\rho e_i\right)\frac{d}{\lambda}\cos\theta} \tag{3.39}$$

for $i \in [M]$. Then, an imperfect array manifold $\tilde{\mathbf{a}}_\rho(\theta)$ of an M-element ULA can be defined by

$$\tilde{\mathbf{a}}_\rho(\theta) = \mathbf{C}_\rho \mathbf{G}_\rho \mathbf{H}_\rho \mathbf{a}_\rho(\theta) \tag{3.40}$$

where the gain bias is modeled by

$$\mathbf{G}_\rho = \mathbf{I} + \rho \operatorname{diag}(g_1, g_2, \cdots, g_M), \tag{3.41}$$

the phase bias is modeled by

$$\mathbf{H}_\rho = \operatorname{diag}\left(e^{j\rho h_1}, e^{j\rho h_2}, \cdots, e^{j\rho h_M}\right), \tag{3.42}$$

and the intersensor mutual coupling is modeled by

$$\mathbf{C}_\rho = \mathbf{I} + \rho \mathrm{Toep}\left(\begin{bmatrix} 0 & \gamma & \gamma^2 & \cdots & \gamma^{M-1} \end{bmatrix}^\mathsf{T}\right). \tag{3.43}$$

For the hyperparameters in the case of the 5-element MRA, we use $e_1 = 0, e_2 = \cdots = e_6 = -0.2, e_7 = \cdots = e_{10} = 0.2$, $g_1 = 0, g_2 = \cdots = g_6 = 0.2, g_7 = \cdots = g_{10} = -0.2$, $h_1 = 0, h_2 = \cdots = h_6 = -\frac{1}{6}\pi, h_7 = \cdots = h_{10} = \frac{1}{6}\pi$, and $\gamma = 0.3e^{\frac{\pi}{3}}$. For the 4-element and 6-element MRAs, the hyperparameters can be found in the source code. The degree of imperfections is controlled by a scalar $\rho \in [0,1]$. A larger ρ makes the imperfections more severe and $\rho = 0$ means the array is perfect. To train a model for imperfect arrays, we uniformly select ρ at random on the unit interval $[0,1]$. To train a model for a perfect array, we use $\rho = 0$.

Figure 3.10 shows MSE in terms of the array imperfection parameter ρ for different numbers of sources. Indeed, a different ρ represents a different array; thus, one can collect a new dataset and then specifically train a new model. However, here, we train a single model on a joint dataset collected from different arrays. The MSE of SPA and WDA both get worse as ρ increases, verifying that both methods suffer from model mismatch and are not robust to array imperfections. Even though no attempts have been made to SPA and WDA to contend with array imperfections, such corrections are nontrivial and require the knowledge of imperfections. On the other hand, the MSE of the proposed method stays at the same level despite the increasing degree of imperfections, implying that the subspace representation learning approach is robust to array imperfections. Figure 3.11 and 3.12 show the results for the 4-element and 6-element MRAs, one can also deduce conclusions that are similar to ones from Figure 3.10.

3.7 Conclusion

A new methodology learning subspace representations is proposed for robust estimation of more sources than sensors. To learn subspace representations, the codomain of a DNN model, is defined as a union of Grassmannians reflecting signal subspaces of different dimensions. Then,

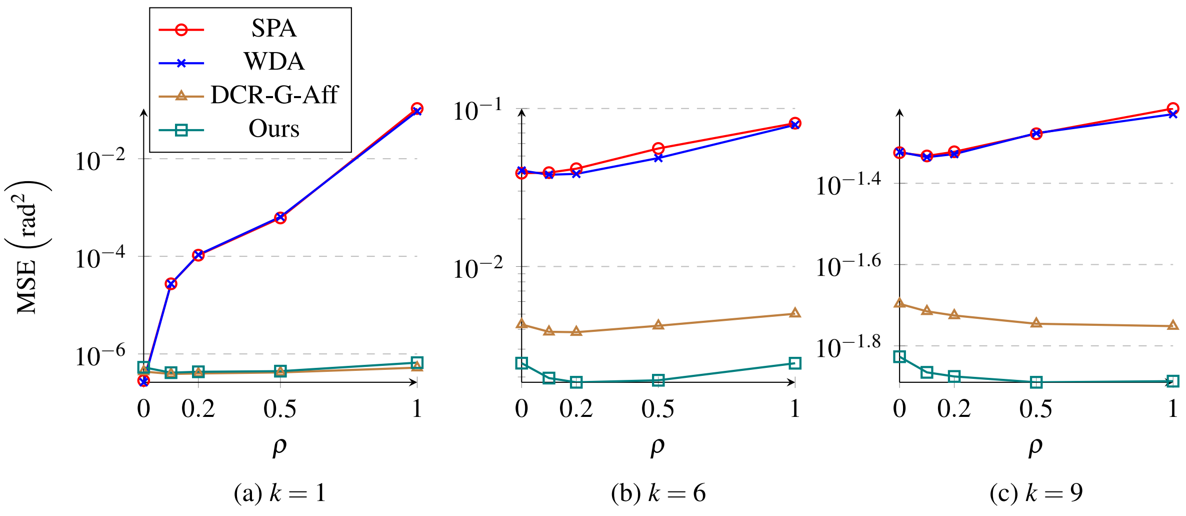


Figure 3.10. MSE vs. the array imperfection parameter ρ. Note that only one DNN model is trained for our approach. Unlike model-based methods that give significantly worse MSE as ρ increases, our approach is robust to array imperfections without being given ρ or the knowledge about the degree of imperfections.

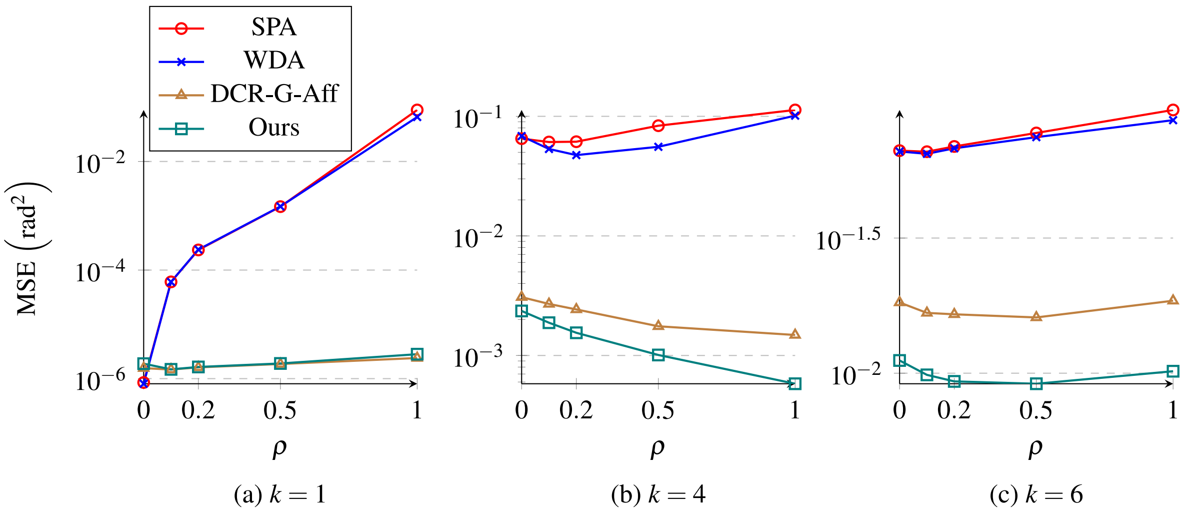


Figure 3.11. MSE vs. the array imperfection parameter ρ. $N = 4$. $M = 7$.

a family of loss functions is proposed as functions of the principal angles between subspaces to ensure rational invariance. In particular, we use geodesic distances on Grassmannians to train a DNN model and prove that it is possible for a ReLU network to approximate signal subspaces. Because a subspace is invariant to the selection of the basis, our methodology enlarges the solution space of a DNN model compared to existing approaches that learn covariance matrices.

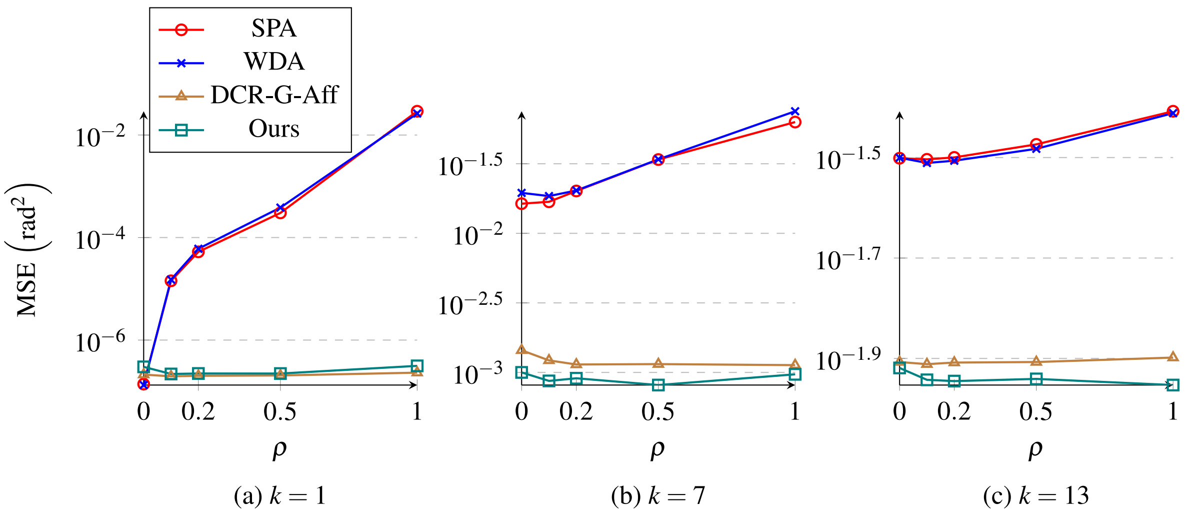


Figure 3.12. MSE vs. the array imperfection parameter ρ. $N = 6$. $M = 14$.

In addition, due to its geometry-agnostic nature, our methodology is robust to array imperfections. To study the possibility of bypassing the root-MUSIC algorithm, we propose a gridless end-to-end approach that directly learns a mapping from sample SCMs to DoAs. Numerical results show that subspace representation learning outperforms existing SDP-based approaches including the SPA and WDA, DNN-based covariance matrix reconstruction methods, and the gridless end-to-end approach under the standard assumptions. These results imply that learning subspace representations is more beneficial than learning covariance matrices or angles directly.

3.8 Appendix

3.8.1 Proof of Theorem 6

Proof. Let $\mathscr{I} = \left\{(i,j) \in [k] \times [k] \mid i \neq j\right\}$. For every $(i,j) \in \mathscr{I}$, define

$$\mathscr{F}_{i,j} = \left\{\boldsymbol{\theta} \in [0,\pi]^k \mid \theta_i = \theta_j\right\}. \tag{3.44}$$

Pick $\delta > 0$ and let μ denote the Lebesgue measure. Because $\mathscr{F}_{i,j}$ is closed and $\mu(\mathscr{F}_{i,j}) = 0$ for every $(i,j) \in \mathscr{I}$, there exists a open set

$$\mathscr{F}_\delta \supset \bigcup_{(i,n)\in\mathscr{I}} \mathscr{F}_{i,j} \tag{3.45}$$

such that

$$\mu(\mathscr{F}_\delta) < \delta. \tag{3.46}$$

Therefore, $\mathscr{E}_\delta = [0,\pi]^k \setminus \mathscr{F}_\delta$ is compact. Now, note that $\mathbf{A}(\boldsymbol{\theta})$ is a rank-k matrix for every $\boldsymbol{\theta} \in \mathscr{E}_\delta$ due to the Vandermonde structure. It follows that the function $\mathbf{X} \mapsto P_{\mathbf{X}}$ is continuous on $\mathbf{A}(\mathscr{E}_\delta)$. On the other hand, the mapping $\mathbf{R}_{\mathscr{S}} \mapsto \mathbf{R}_0$ is affine on $\mathbf{A}(\mathscr{E}_\delta)$ since the SLA has no holes in its co-array. As $\mathbf{R}_0 = \mathbf{A}(\boldsymbol{\theta})\mathbf{P}\mathbf{A}^{\mathsf{H}}(\boldsymbol{\theta})$, we have $P_{\mathbf{R}_0} = P_{\mathbf{A}(\boldsymbol{\theta})}$, implying that $\mathbf{R}_{\mathscr{S}} \mapsto P_{\mathbf{A}(\boldsymbol{\theta})}$ is continuous on $\mathbf{A}(\mathscr{E}_\delta)$. By Theorem 1 of [Chen et al., 2022], any continuous piecewise linear function can be represented by a ReLU network. Because the set of continuous piecewise linear functions is dense in the set of continuous functions on any compact subset of $\mathbb{C}^{N\times N}$, it follows that, for every ε, there is a ReLU network f such that

$$\sup_{\boldsymbol{\theta}\in\mathscr{E}_\delta} \left\| f(\mathbf{R}_{\mathscr{S}}) - P_{\mathbf{A}(\boldsymbol{\theta})} \right\|_F < \varepsilon. \tag{3.47}$$

Note that $\mathbf{R}_{\mathscr{S}}(\mathscr{E}_\delta)$ is still compact since $\boldsymbol{\theta} \mapsto \mathbf{R}_{\mathscr{S}}$ is continuous. By Lemma 14,

$$\int_{\mathscr{E}_\delta} d_k^{\text{Geo}}\left(f(\mathbf{R}_{\mathscr{S}}), P_{\mathbf{A}(\boldsymbol{\theta})}\right) d\boldsymbol{\theta} < \pi^k \sqrt{k} \sin^{-1}\left(\frac{\varepsilon}{\sqrt{2}}\right). \tag{3.48}$$

As f is continuous and every nonzero orthogonal projection is bounded, there exists $L > 0$ such that $\left\| f(\mathbf{R}_{\mathscr{S}}) - P_{\mathbf{A}(\boldsymbol{\theta})} \right\|_F < L$ for every $\boldsymbol{\theta} \in \mathscr{F}_\delta$, which implies

$$\int_{\mathscr{F}_\delta} d_k^{\text{Geo}}\left(f(\mathbf{R}_{\mathscr{S}}), P_{\mathbf{A}(\boldsymbol{\theta})}\right) d\boldsymbol{\theta} < \delta \sqrt{k} \sin^{-1}\left(\frac{L}{\sqrt{2}}\right). \tag{3.49}$$

The claim is proved because both $\delta > 0$ and $\varepsilon > 0$ can be arbitrarily small, and $\sin^{-1}(x) \to 0$ as $x \to 0^{+}$. $\square$

3.8.2 Proof of Lemma 14

Proof. Because there is a one-to-one correspondance between the set of linear subspaces and the set of orthogonal projectors, a distance $d : \mathrm{Gr}(k,M) \times \mathrm{Gr}(k,M) \to [0,\infty)$ between $\mathscr{U}_1 \in \mathrm{Gr}(k,M)$ and $\mathscr{U}_2 \in \mathrm{Gr}(k,M)$ can be defined as

$$d(\mathscr{U}_1, \mathscr{U}_2) = \left\| P_{\mathscr{U}_1} - P_{\mathscr{U}_2} \right\|_F \tag{3.50}$$

where $P_{\mathscr{U}_1}$ and $P_{\mathscr{U}_2}$ are the orthogonal projectors onto $\mathscr{U}_1$ and $\mathscr{U}_2$, respectively. Then, it follows that

$$\begin{aligned} d^2(\mathscr{U}_1, \mathscr{U}_2) &= 2k - 2\mathrm{tr}\left(P_{\mathscr{U}_1} P_{\mathscr{U}_2}\right) \\ &= 2k - 2\sum_{i=1}^{k} \sigma_i^2\left(P_{\mathscr{U}_1} P_{\mathscr{U}_2}\right) \\ &= 2k - 2\sum_{i=1}^{k} \sigma_i^2\left(\mathbf{U}_1^{\mathsf{H}} \mathbf{U}_2\right) \\ &= 2k - 2\sum_{i=1}^{k} \cos^2 \phi_i \end{aligned} \tag{3.51}$$

where $\mathbf{U}_1$ and $\mathbf{U}_2$ are matrices whose columns form unitary bases of $\mathscr{U}_1$ and $\mathscr{U}_2$. Therefore, we have

$$\left\| P_{\mathscr{U}_1} - P_{\mathscr{U}_2} \right\|_F = \sqrt{2}\left(\sum_{i=1}^{k} \sin^2 \phi_i\right)^{\frac{1}{2}} \tag{3.52}$$

which was shown in [Stewart, 1991]. Finally, (3.52) implies that

$$\phi_i \leq \sin^{-1}\left(\frac{\left\| P_{\mathscr{U}_1} - P_{\mathscr{U}_2} \right\|_F}{\sqrt{2}}\right) \tag{3.53}$$

for every $i \in [k]$. $\square$

3.8.3 Learning rates

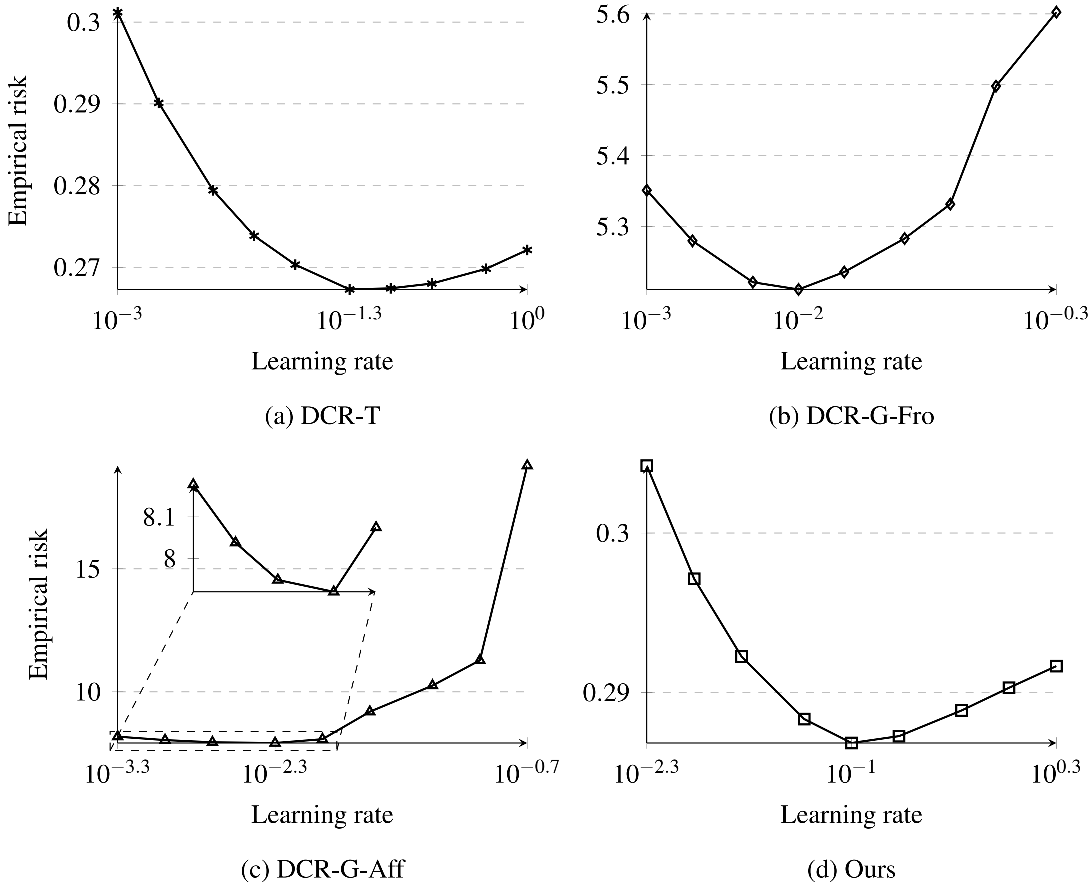


Figure 3.13. Search of the best learning rates. Empirical risk on the validation set vs. the maximum learning rate in the one-cycle learning rate scheduler.

To determine the best learning rates to use in Section 3.6, a grid search of the best maximum learning rate in the one-cycle learning rate scheduler [Smith and Topin, 2019] is performed for each approach. For each $k \in [M-1]$, there are 2×10^6 and 6×10^5 random data points for training and validation, respectively, leading to a training dataset of size $L_{\text{train}} = 9 \times 2 \times 10^6$ and a validation dataset of size $L_{\text{val}} = 9 \times 6 \times 10^5$. Figure 3.13 shows that the best learning rates for DCR-T, DCR-G-Fro, DCR-G-Aff, and the proposed approach are 0.05, 0.01, 0.005, and 0.1, respectively. These learning rates are also used to train all the corresponding models in Section 3.6.2 and 3.6.3. For the gridless end-to-end approach, the best learning is 0.2 according to Figure 3.14.

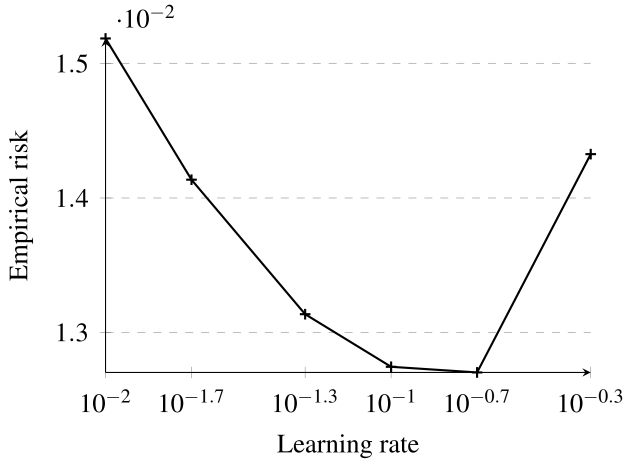


Figure 3.14. Search of the best learning rate for the gridless end-to-end approach.

3.9 Acknowledgements

This work was supported in part by IEEE Signal Processing Society Scholarship, and in part by NSF under Grant CCF-2225617 and Grant CCF-2124929. The authors would like to thank Dr. Ching-Hua Lee and Mr. Rushabha Balaji for their constructive comments on the gridless end-to-end training and consistent rank sampling, respectively.

Chapter 3, in part, is a reprint of the material as it appears in K.-L. Chen and B. D. Rao, "Subspace representation learning for sparse linear arrays to localize more sources than sensors: A deep learning methodology," which was submitted to *IEEE Transactions on Signal Processing* in 2024, with its preprint available at arXiv (`https://arxiv.org/abs/2408.16605`). The dissertation author was the primary investigator and author of this material.

Chapter 4

Conclusions and Future Work

In this chapter, we will summarize the key contributions of this dissertation. These contributions advance the fundamental theory of deep learning and provide new deep learning-based methodologies for signal processing. Lastly, we will point out their future research directions.

4.1 Neural complexity and dimension-independent bounds

We established tighter upper bounds for the number of computational units required for a ReLU neural network to represent or compute any given CPWL function. Specifically, we prove that any n-dimensional CPWL function with q pieces can be represented by a feedforward network using ReLUs with at most $\mathscr{O}(q^2)$ hidden neurons. This quadratic bound is independent of the input dimension and outperforms previous bounds exponentially. As far as we know, this upper bound currently holds the state-of-the-art in literature. Besides ensuring the existence of such a low-complexity network, we have also developed the first polynomial-time complexity algorithm to identify a neural network that satisfies these tighter bounds. In addition, a tighter upper bound on the number of pieces, in terms of the number of linear components, has been proved. Our findings advance the fundamental understanding of neural complexity in computing a CPWL function. With these new results, any CPWL function can be realized by a low-complexity DNN. Since every ReLU neural network is CPWL, these upper bounds directly serve as worst-case guarantees for many DNN pruning algorithms. Furthermore, the network found by my algorithm allows most of the weights to be represented by one bit, making it possible to design new low-energy and low-complexity neural engines. We also show that one can reverse-engineer a ReLU network in polynomial time provided the partitions of the input space are available. All of these findings have potentially strong implications for modeling complexity of neural networks using other types of activation functions since the family of CPWL functions is dense in the family of continuous functions.

4.2 Interpretable neural building blocks with optimization guarantees

We proved that the most popular building block in deep learning, “the skip connection,” can guarantee to improve representations over residual blocks when the expansion layer is sufficiently large under mild assumptions. Our results are significant as it explains (a) why the ResNet can avoid the “degradation problem” and be scaled to thousands of layers, (b) why the wide ResNet is superior than the ResNet, and (c) why the bottleneck blocks are more economical than the basic block. To the best of my knowledge, our work is the first to theoretically establish the benefits of input space expansion in ResNets. To prove these results, we designed a slightly different ResNet architecture called ResNEst and found that a ResNEst is a basis function model that is limited by a coupling problem in basis learning and linear prediction. By constructing an augmented ResNEst that decouples prediction weights from basis learning, we showed that the minimum empirical risk of the augmented ResNEst serves as an empirical risk lower bound for a ResNEst using the same basis. Then, It follows that every local minimizer of such a ResNEst is a global minimizer despite the optimization landscape being nonconvex. More importantly, it implies that any local minimizer of a ResNEst model is better than any best linear predictor in training performance. This guarantee allows one to confidently replace linear models with ResNEsts without compromising training performance. Furthermore, our results demonstrate that bottleneck blocks are more economical than basic blocks when increasing depth, which supports empirical findings and provides principles to select hyperparameters for ResNet-like networks.

4.3 Subspace representation learning

We developed novel deep learning-based methodologies for SLAs to localize more sources than sensors in DoA estimation. In contrast to previous SDP-based or DNN-based methods reconstructing covariance matrices, we propose a methodology called *subspace representation learning*, which constructs subspace representations suitable for DNNs and utilizes a DNN to estimate the signal and noise subspaces. We view the output representations of the DNN as a union of Grassmannians and construct loss functions of different dimensions reflecting different numbers of sources via principal angles between the desired subspace and the predicted subspace. In particular, we develop empirical risk minimization problems and losses using the geodesic distances that measure the length of the shortest curve(s) within the union of Grassmannians. We also proved that it is possible for a ReLU network to approximate signal subspaces. The proposed methodology is robust to array imperfections due to its array- or geometry-agnostic nature. As training requires the evaluation of loss functions of different dimensions, we propose a new batch sampling strategy called *consistent rank sampling* to parallelize computation on a GPU, greatly accelerating the training procedure. To study the effectiveness of subspace representations, we also propose a gridless end-to-end methodology that directly learns the DoAs. Numerical results show that subspace representation learning outperforms SDP-based methods, including the widely used SPA, recently proposed DNN-based covariance reconstruction approaches, and the proposed gridless end-to-end approach, for a wide range of SNRs, snapshots, and numbers of sources, under the standard assumptions. These results imply that learning subspaces is more beneficial than learning or estimating covariance matrices.

4.4 Future work

The complexity results presented in Chapter 1 raise the following new questions. First, determining whether the upper bounds are tight remains an open problem. Second, the network architecture generated by Algorithm 1 implicitly imposes a constraint on the total number of pieces of a CPWL function, potentially serving as a regularizer to prevent overfitting. Third, for certain families of functions, it may be possible to derive some interesting and insightful bounds in terms of q, k, and an accuracy parameter ε, reducing the dependency on q or k by tolerating small errors.

The insights in Chapter 2 may have some interesting extensions by noting that every ReLU network partitions the input space into polyhedra. Since every skip connection can be viewed as a ReLU network, every ResNEst also divides the input space into polyhedra. However, the skip connections may cause the approximation scheme of ResNEst to differ significantly from that of ReLU networks. For instance, is it possible to show that different residual blocks employ multi-resolution approximators? On the other hand, the bounds derived in Chapter 1 are based on ReLU networks. Is it possible to extend them to ResNEsts or A-ResNEsts?

Although the methodology of subspace representation learning in Chapter 3 was developed for narrowband signals, it is possible to extend the methodology to wideband scenarios. Because subspace representation learning can be applied to every frequency component in the frequency band, perhaps the simplest extension is to construct a DNN to approximate a collection of signal or noise subspaces, which can be achieved by adding an extra dimension to the subspace representations.

Bibliography

R. Arora, A. Basu, P. Mianjy, and A. Mukherjee. Understanding deep neural networks with rectified linear units. In *International Conference on Learning Representations*, 2018.

P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, 1989.

A. Barabell. Improving the resolution performance of eigenstructure-based direction-finding algorithms. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 336–339. IEEE, 1983.

A. Barg and D. Y. Nogin. Bounds on packings of spheres in the grassmann manifold. *IEEE Transactions on Information Theory*, 48(9):2450–2454, 2002.

A. Barthelme and W. Utschick. DoA estimation using neural network-based covariance matrix reconstruction. *IEEE Signal Processing Letters*, 28:783–787, 2021a.

A. Barthelme and W. Utschick. A machine learning approach to DoA estimation and model order selection for antenna arrays with subarray sampling. *IEEE Transactions on Signal Processing*, 69:3075–3087, 2021b.

R. Bhatia. Positive definite matrices. In *Positive Definite Matrices*. Princeton University Press, 2009.

R. Bhatia, T. Jain, and Y. Lim. On the Bures–Wasserstein distance between positive definite matrices. *Expositiones Mathematicae*, 37(2):165–191, 2019.

C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

Å. Björck and G. H. Golub. Numerical methods for computing angles between linear subspaces. *Mathematics of Computation*, 27(123):579–594, 1973.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

K.-L. Chen and B. D. Rao. Subspace representation learning for sparse linear arrays to localize more sources than sensors: A deep learning methodology. *arXiv preprint arXiv:2408.16605*, 2024.

K.-L. Chen, C.-H. Lee, H. Garudadri, and B. D. Rao. ResNEsts and DenseNEsts: Block-based DNN models with improved representation guarantees. In *Advances in Neural Information Processing Systems*, pages 3413–3424, 2021.

K.-L. Chen, H. Garudadri, and B. D. Rao. Improved bounds on neural complexity for representing piecewise linear functions. In *Advances in Neural Information Processing Systems*, volume 35, pages 7167–7180, 2022.

K.-L. Chen, C.-H. Lee, B. D. Rao, and H. Garudadri. A DNN based normalized time-frequency weighted criterion for robust wideband DoA estimation. In *International Conference on Acoustics, Speech and Signal Processing*, pages 1–5. IEEE, 2023a.

K.-L. Chen, D. D. Wong, K. Tan, B. Xu, A. Kumar, and V. K. Ithapu. Leveraging heteroscedastic uncertainty in learning complex spectral mapping for single-channel speech enhancement. In *International Conference on Acoustics, Speech and Signal Processing*, pages 1–5. IEEE, 2023b.

L. O. Chua and A.-C. Deng. Canonical piecewise-linear representation. *IEEE Transactions on Circuits and Systems*, 35(1):101–111, 1988.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems*, pages 2933–2941, 2014.

A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

R. Eldan and O. Shamir. The power of depth for feedforward neural networks. In *Conference on Learning Theory*, pages 907–940. PMLR, 2016.

K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.

J. Gao, C. Zhong, G. Y. Li, J. B. Soriaga, and A. Behboodi. Deep learning-based channel estimation for wideband hybrid mmwave massive MIMO. *IEEE Transactions on Communications*, 71(6):3679–3693, 2023.

R. Ge, F. Huang, C. Jin, and Y. Yuan. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Conference on Learning Theory*, pages 797–842, 2015.

I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *International Conference on Machine Learning*, pages 1319–1327. PMLR, 2013.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008.

M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. https://cvxr.com/cvx, Mar. 2014.

K. Gregor, F. Besse, D. Jimenez Rezende, I. Danihelka, and D. Wierstra. Towards conceptual compression. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

J. Hamm and D. D. Lee. Grassmann discriminant analysis: a unifying view on subspace-based learning. In *International Conference on Machine Learning*, pages 376–383, 2008.

B. Hanin and D. Rolnick. Complexity of linear regions in deep networks. In *International Conference on Machine Learning*, pages 2596–2604. PMLR, 2019a.

B. Hanin and D. Rolnick. Deep ReLU networks have surprisingly few activation patterns. In *Advances in Neural Information Processing Systems*, 2019b.

M. Hardt and T. Ma. Identity matters in deep learning. In *International Conference on Learning Representations*, 2017.

G. H. Hardy, J. E. Littlewood, and G. Pólya. *Inequalities*. Cambridge University Press, 1934.

H. He, B. Xin, S. Ikehata, and D. Wipf. From bayesian sparsity to gated recurrent nets. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

J. He, L. Li, J. Xu, and C. Zheng. ReLU deep neural networks and linear finite elements. *Journal of Computational Mathematics*, 38(3):502–527, 2020.

K. He and J. Sun. Convolutional neural networks at constrained time cost. In *Conference on Computer Vision and Pattern Recognition*, pages 5353–5360. IEEE, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *International Conference on Computer Vision*, pages 1026–1034. IEEE, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 770–778. IEEE, 2016a.

K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016b.

C. Hertrich, A. Basu, M. Di Summa, and M. Skutella. Towards lower bounds on the depth of ReLU neural networks. In *Advances in Neural Information Processing Systems*, pages 3336–3348, 2021.

N. J. Higham. *Functions of Matrices : Theory and Computation*. Society for Industrial and Applied Mathematics, 2008.

P. Hinz. *An analysis of the piece-wise affine structure of ReLU feed-forward neural networks*. PhD thesis, ETH Zurich, 2021.

P. Hinz and S. van de Geer. A framework for the construction of upper bounds on the number of affine linear regions of relu feed-forward neural networks. *IEEE Transactions on Information Theory*, 65(11):7304–7324, 2019.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *Conference on Computer Vision and Pattern Recognition*, pages 7132–7141. IEEE, 2018.

G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Conference on Computer Vision and Pattern Recognition*, pages 4700–4708. IEEE, 2017.

W.-L. Hwang and A. Heinecke. Un-rectifying non-linear networks for signal representation. *IEEE Transactions on Signal Processing*, 68:196–210, 2019.

N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, pages 302–311, 1984. Revised version: *Combinatorica* 4:373–395, 1984.

K. Kawaguchi. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, pages 586–594, 2016.

K. Kawaguchi and Y. Bengio. Depth with nonlinearity creates no bad local minima in resnets. *Neural Networks*, 118:167–174, 2019.

L. G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk*, 244 (5):1093–1096, 1979. Translated in *Soviet Mathematics Doklady* 20(1):191–194, 1979.

J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Conference on Computer Vision and Pattern Recognition*, pages 1646–1654. IEEE, 2016.

J. Kim, M. El-Khamy, and J. Lee. Residual LSTM: Design of a deep recurrent architecture for distant speech recognition. *arXiv preprint arXiv:1701.03360*, 2017.

A. Krizhevsky. Learning multiple layers of features from tiny images. *Tech Report*, 2009.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pages 6389–6399, 2018.

Y. Li and Y. Chi. Off-the-grid line spectrum denoising and estimation with multiple measurement vectors. *IEEE Transactions on Signal Processing*, 64(5):1257–1269, 2015.

Y. Li and Y. Yuan. Convergence analysis of two-layer neural networks with relu activation. In *Advances in Neural Information Processing Systems*, volume 30, pages 597–607, 2017.

S. Liang, R. Sun, Y. Li, and R. Srikant. Understanding the loss surface of neural networks for binary classification. In *International Conference on Machine Learning*, pages 2835–2843, 2018.

T. L. Liu, M. Chen, M. Zhou, S. Du, E. Zhou, and T. Zhao. Towards understanding the importance of shortcut connections in residual networks. In *Advances in Neural Information Processing Systems*, 2019.

Y. Liu, H. Chen, and B. Wang. DOA estimation based on CNN for underwater acoustic array. *Applied Acoustics*, 172:107594, 2021.

Z.-M. Liu, C. Zhang, and S. Y. Philip. Direction-of-arrival estimation based on deep neural networks with robustness to array imperfections. *IEEE Transactions on Antennas and Propagation*, 66(12):7315–7327, 2018.

X. Lu, Y. Tsao, S. Matsuda, and C. Hori. Speech enhancement based on deep denoising autoencoder. In *Interspeech*, volume 2013, pages 436–440, 2013.

Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang. The expressive power of neural networks: A view from the width. In *Advances in Neural Information Processing Systems*, 2017.

A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning*, 2013.

A. Magnani and S. P. Boyd. Convex piecewise-linear fitting. *Optimization and Engineering*, 10 (1):1–17, 2009.

W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

F. Milletari, N. Navab, and S.-A. Ahmadi. V-Net: Fully convolutional neural networks for volumetric medical image segmentation. In *International Conference on 3D Vision*, pages 565–571. IEEE, 2016.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

G. Montúfar. Notes on the number of linear regions of deep neural networks. In *International Conference on Sampling Theory and Applications*, 2017.

G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2924–2932, 2014.

V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, pages 807–814, 2010.

B. Ottersten, P. Stoica, and R. Roy. Covariance matching estimation techniques for array signal processing applications. *Digital Signal Processing*, 8(3):185–210, 1998.

S. Ovchinnikov. Max-min representation of piecewise linear functions. *Contributions to Algebra and Geometry*, 43(1):297–302, 2002.

P. Pal and P. P. Vaidyanathan. Nested arrays: A novel approach to array processing with enhanced degrees of freedom. *IEEE Transactions on Signal Processing*, 58(8):4167–4181, 2010.

G. K. Papageorgiou, M. Sellathurai, and Y. C. Eldar. Deep networks for direction-of-arrival estimation in low SNR. *IEEE Transactions on Signal Processing*, 69:3714–3729, 2021.

R. Pascanu, G. Montúfar, and Y. Bengio. On the number of response regions of deep feed forward networks with piece-wise linear activations. *International Conference on Learning Representations*, 2014.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, , A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

S. U. Pillai, Y. Bar-Ness, and F. Haber. A new approach to array geometry for improved spatial spectrum estimation. *Proceedings of the IEEE*, 73(10):1522–1524, 1985.

L. Pisha, J. Warchall, T. Zubatiy, S. Hamilton, C.-H. Lee, G. Chockalingam, P. P. Mercier, R. Gupta, B. D. Rao, and H. Garudadri. A wearable, extensible, open-source platform for hearing healthcare research. *IEEE Access*, 7:162083–162101, 2019.

R. R. Pote and B. D. Rao. Maximum likelihood-based gridless DoA estimation using structured covariance matrix recovery and SBL with grid refinement. *IEEE Transactions on Signal Processing*, 71:802–815, 2023.

H. Qiao and P. Pal. On maximum-likelihood methods for localizing more sources than sensors. *IEEE Signal Processing Letters*, 24(5):703–706, 2017.

X. Qin, Z. Zhang, C. Huang, M. Dehghan, O. R. Zaiane, and M. Jagersand. U^2-Net: Going deeper with nested U-structure for salient object detection. *Pattern Recognition*, 106:107404, 2020.

M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein. On the expressive power of deep neural networks. In *International Conference on Machine Learning*, pages 2847–2854. PMLR, 2017.

B. D. Rao and K. S. Hari. Performance analysis of root-music. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(12):1939–1949, 1989.

J. Renegar. A polynomial-time algorithm, based on Newton's method, for linear programming. *Mathematical Programming*, 40(1):59–93, 1988.

D. Rolnick and K. Kording. Reverse-engineering deep ReLU networks. In *International Conference on Machine Learning*, pages 8178–8187. PMLR, 2020.

O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, pages 234–241. Springer, 2015.

F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

A. Sant and B. D. Rao. DOA estimation in systems with nonlinearities for mmwave communications. In *International Conference on Acoustics, Speech and Signal Processing*, pages 4537–4541. IEEE, 2020.

P. Sarangi, M. C. Hücümenoğlu, and P. Pal. Beyond coarray music: Harnessing the difference sets of nested arrays with limited snapshots. *IEEE Signal Processing Letters*, 28:2172–2176, 2021.

P. Sarangi, M. C. Hücümenoğlu, R. Rajamäki, and P. Pal. Super-resolution with sparse arrays: A nonasymptotic analysis of spatiotemporal trade-offs. *IEEE Transactions on Signal Processing*, 71:4288–4302, 2023.

R. Schmidt. Multiple emitter location and signal parameter estimation. *IEEE Transactions on Antennas and Propagation*, 34(3):276–280, 1986.

T. Serra, C. Tjandraatmadja, and S. Ramalingam. Bounding and counting linear regions of deep neural networks. In *International Conference on Machine Learning*, pages 4558–4566. PMLR, 2018.

O. Shamir. Are ResNets provably better than linear predictors? In *Advances in Neural Information Processing Systems*, pages 507–516, 2018.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587): 484–489, 2016.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015a.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015b.

S. Smale. Mathematical problems for the next century. *The Mathematical Intelligencer*, 20(2): 7–15, 1998.

L. N. Smith and N. Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, pages 369–386. SPIE, 2019.

R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

G. W. Stewart. Perturbation theory for the singular value decomposition. In *SVD and Signal Processing II, Algorithms, Analysis and Applications*, pages 99–109. Elsevier Science Publishers, 1991.

P. Stoica and P. Babu. SPICE and LIKES: Two hyperparameter-free methods for sparse-parameter estimation. *Signal Processing*, 92(7):1580–1590, 2012.

P. Stoica and T. Söderström. On reparametrization of loss functions used in estimation and the invariance principle. *Signal processing*, 17(4):383–387, 1989.

P. Stoica, P. Babu, and J. Li. New method of sparse parameter estimation in separable models and its use for spectral analysis of irregularly sampled data. *IEEE Transactions on Signal Processing*, 59(1):35–47, 2010a.

P. Stoica, P. Babu, and J. Li. SPICE: A sparse covariance-based estimation method for array processing. *IEEE Transactions on Signal Processing*, 59(2):629–638, 2010b.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Conference on Computer Vision and Pattern Recognition*, pages 1–9. IEEE, 2015.

G. Tang, B. N. Bhaskar, and B. Recht. Near minimax line spectral estimation. *IEEE Transactions on Information Theory*, 61(1):499–512, 2014.

J. M. Tarela and M. V. Martínez. Region configurations for realizability of lattice piecewise-linear models. *Mathematical and Computer Modelling*, 30(11-12):17–27, 1999.

J. M. Tarela, E. Alonso, and M. V. Martínez. A representation method for PWL functions oriented to parallel processing. *Mathematical and Computer Modelling*, 13(10):75–83, 1990.

M. Telgarsky. Benefits of depth in neural networks. In *Conference on Learning Theory*, pages 1517–1539. PMLR, 2016.

M. E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of machine* *learning research*, 1(Jun):211–244, 2001.

K.-C. Toh, M. J. Todd, and R. H. Tütüncü. SDPT3—A MATLAB software package for semidefinite programming, version 1.3. *Optimization Methods and Software*, 11(1-4):545–581, 1999.

H. L. Van Trees. *Optimum array processing: Part IV of detection, estimation, and modulation theory*. John Wiley & Sons, 2002.

G. Vardi, D. Reichman, T. Pitassi, and O. Shamir. Size and depth separation in approximating benign functions with neural networks. In *Conference on Learning Theory*, pages 4195–4223. PMLR, 2021.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

S. A. Vavasis and Y. Ye. A primal-dual interior point method whose running time depends only on the constraint matrix. *Mathematical Programming*, 74(1):79–120, 1996.

A. Veit, M. Wilber, and S. Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems*, pages 550–558, 2016.

M. Wang, Z. Zhang, and A. Nehorai. Grid-less DOA estimation using sparse linear arrays based on wasserstein distance. *IEEE Signal Processing Letters*, 26(6):838–842, 2019.

S. Wang. General constructive representations for continuous piecewise-linear functions. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 51(9):1889–1896, 2004.

S. Wang and X. Sun. Generalization of hinging hyperplanes. *IEEE Transactions on Information Theory*, 51(12):4425–4431, 2005.

D. P. Wipf and B. D. Rao. Sparse bayesian learning for basis selection. *IEEE Transactions on Signal processing*, 52(8):2153–2164, 2004.

Y.-C. Wong. Differential geometry of grassmann manifolds. *Proceedings of the National Academy of Sciences*, 57(3):589–594, 1967.

X. Wu, W.-P. Zhu, and J. Yan. A Toeplitz covariance matrix reconstruction approach for direction-of-arrival estimation. *IEEE Transactions on Vehicular Technology*, 66(9):8223–8237, 2017.

X. Wu, X. Yang, X. Jia, and F. Tian. A gridless DOA estimation method based on convolutional neural network with Toeplitz prior. *IEEE Signal Processing Letters*, 29:1247–1251, 2022.

S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Conference on Computer Vision and Pattern Recognition*, pages 1492–1500. IEEE, 2017.

W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke. The Microsoft 2017 conversational speech recognition system. In *International Conference on Acoustics, Speech and Signal Processing*, pages 5934–5938. IEEE, 2018.

Z. Yang, L. Xie, and C. Zhang. A discretization-free sparse and parametric approach for linear array signal processing. *IEEE Transactions on Signal Processing*, 62(19):4959–4973, 2014.

D. Yarotsky. Optimal approximation of continuous functions by very deep relu networks. In *Conference on learning theory*, pages 639–649. PMLR, 2018.

K. Ye and L.-H. Lim. Schubert varieties and distances between subspaces of different dimensions. *SIAM Journal on Matrix Analysis and Applications*, 37(3):1176–1197, 2016.

C. Yun, S. Sra, and A. Jadbabaie. Are deep ResNets provably better than linear predictors? In *Advances in Neural Information Processing Systems*, pages 15686–15695, 2019.

S. Zagoruyko and N. Komodakis. Wide residual networks. In *British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, 2016.

T. Zaslavsky. *Facing up to arrangements: Face-count formulas for partitions of space by hyperplanes*, volume 154. American Mathematical Society, 1975.

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. E. Hinton. On rectified linear units for speech processing. In *International Conference on Acoustics, Speech and Signal Processing*, pages 3517–3521. IEEE, 2013.

C. Zhou, Y. Gu, X. Fan, Z. Shi, G. Mao, and Y. D. Zhang. Direction-of-arrival estimation for coprime array via virtual array interpolation. *IEEE Transactions on Signal Processing*, 66(22): 5956–5971, 2018.