

Complexity-Scalable Neural Network Based MIMO Detection With Learnable Weight Scaling

Abdullahi Mohammad, *Student Member, IEEE*, Christos Masouros, *Senior Member, IEEE*,
and Yiannis Andreopoulos, *Senior Member, IEEE*

Abstract—This paper introduces a framework for systematic complexity scaling of deep neural network (DNN) based MIMO detectors. The model uses a fraction of the DNN inputs by scaling their values through weights that follow monotonically non-increasing functions. This allows for weight scaling *across* and *within* the different DNN layers in order to achieve scalable complexity-accuracy results. To reduce complexity further, we introduce a regularization constraint on the layer weights such that, at inference, parts (or the entirety) of network layers can be removed with minimal impact on the detection accuracy. We also introduce trainable weight-scaling functions for increased robustness to changes in the activation patterns and a further improvement in the detection accuracy at the same inference complexity. Numerical results show that our approach is 10 and 100-fold less complex than classical approaches based on semi-definite relaxation and ML detection, respectively.

Index Terms—MIMO detection, Deep Neural Networks, DetNet, profile weight coefficients.

I. INTRODUCTION

THE emergence of smart-phones, tablets, and various new applications in recent years has led to the dramatic increase in mobile data traffic, especially mobile video traffic and other multimedia applications [1]. The developments underpinning Fifth Generation (5G) networks aim to improve the network capacity, spectral efficiency, and latency in order to accommodate these bandwidth consuming applications and services. A key technology for 5G is massive MIMO systems where the base stations (BSs) of the cellular network are equipped with tens, hundreds or thousands of antennas [2]. This configuration further muddles the signal detection problem for the uplink transmission due to the computational complexity of the detector, which increases as the number of BS antennas grows [3].

MIMO detectors have been extensively studied over the last two decades [4], [5] with the view to improving their detection accuracy and decreasing their complexities. The Maximum Likelihood (ML) detector is known to be optimal, but with prohibitive complexity [6]. The sphere decoder (SD) provides near optimal performance, but its complexity still scales exponentially with the number

of antennas [6]. Optimization-based detectors such as semidefinite relaxation (SDR) and approximate message passing (AMP) have been proven to achieve near optimal performance in some scenarios, but at the expense of high-degree polynomial complexity in case of the former and divergence due to iterative characteristics for the later [7]–[10]. Finally, linear detection methods such as zero forcing (ZF), maximum ratio combining (MCR) and minimum mean squared error (MMSE) [11], [12] have the least complexity, with ZF and MMSE showing good performance characteristics for massive MIMO systems [13], albeit still having a gap in detection accuracy versus ML detection as the number of receive antennas decreases.

With the explosive growth of data, computers are empowered with some “intelligence”, i.e., the ability to learn from data inputs without being explicitly programmed. This concept is known as “machine learning”. Deep learning (DL) is the next evolution of machine learning and has already brought unprecedented performance boosting in many fields: robotics, e-commerce, computer vision, and natural language processing [14], [15]. It offers a new paradigm for data-driven learning in problems that cannot be expressed by tractable mathematical models, or are challenged by algorithmic complexity. It is against this background that some research groups in wireless communications community have started leveraging on the expressive powers of DL to find solutions to some physical layer problems where analytic models cannot be derived. Alternatively, when possible to have an analytic optimization objective, the analytic expression is highly non-convex and very-high dimensional, such that conventional numerical optimization is computationally unfeasible [16]–[18]. Some of the recent work on learning on the physical layer involve channel coding [19], [20] and end-to-end communications, and signal detection [21]–[23], where the entire communications system blocks are implemented as an auto-encoder for learning to reconstruct the transmitted symbols at the receiver end. All these are limited to a single antenna at the transmitter and receiver ends. More recent works have involved MIMO detection through deep learning. One of the earliest attempts is the work of O’Shea *et al.* [24], who implemented unsupervised learning using an auto-encoder as an extension of end-to-end learning of previous attempts [22]. Channel equalization for the nonlinear channel using a DNN was proposed by Xu *et al.* [25], where two neural networks are jointly trained: the first is a convolutional neural network (CNN), which

All authors are with the Department of Electronic and Electrical Engineering, University College London, London WC1E 7JEK, UK. (e-mail: abdullahi.mohammad.16@ucl.ac.uk; c.masouros@ucl.ac.uk; i.andreopoulos@ucl.ac.uk)

This work was supported by EPSRC under grant EP/S028455/1

is trained to recover the transmitted symbols from non-linear distortions and channel impairments. The second is a multilayer perceptron (MLP), also known as the fully connected neural network, and is used to perform the detection. Multilevel MIMO detection using coupled-neural networks structure is investigated by Corlay *et al.* [26]. These DNN detectors, while being able to yield performance improvements, they are still not capable to utilize the combined effects of both learning and the detection capability of the conventional optimal detector. This is addressed in the work of Samuel *et al.* [27], where a novel deep learning method based on ML projected gradient optimization known as DetNet is proposed. This approach is significant as it derives a learnable signal detection architecture for multiple channels on a single training shot with near-optimal performance and with lower inference complexity. The work has been further extended to handle higher digital constellations [28], where the authors investigated the complexity-accuracy trade-off as more layers are added. A similar approach by unfolding belief propagation (BP) based on modified BP algorithms (damped BP and maximum BP) presented recently by Tan *et al.* [29] and Liu and Li [30]. Finally, beyond DNN approaches, iterative data-driven massive MIMO detector based on the projected gradient descent approach was proposed recently by Imanishi *et al.* [31].

Overall, it is now well understood that making neural network architectures deeper provides for better detection accuracy than shallow learning approaches, at the expense of significantly higher complexity and training time. This creates the imperative for systematic approaches to design DNN architectures with scalable complexity that can enable learning and inference on a range of devices such as mobile devices, filters, multiplexers, etc. It is against this background that DNN acceleration at both training and inference becomes an active research area within the deep learning community and has recently received significant attention [32]–[34]. While many proposals have been put forward for accelerated DNN training and inference in computer vision [35]–[39], to the best of our knowledge, no systematic DNN acceleration has so far been designed for physical layer communications. In our work, we attempt to cover this gap by proposing a complexity-scalable DNN model for efficient MIMO detection.

A key challenge of most deep learning based MIMO decoders is that they struggle to decode a message with code length having a number of bits greater than 50 ($M = 2^n$; $n > 50$). This tends to be addressed by substantially increasing model capacity, i.e., making DNNs substantially deeper. However, this increases training and inference complexity and makes such designs prone to over-fitting, thus making them extremely challenging to train [34], [35], [40]. For example, it takes nearly 49 hours (approximately 2 days) to train DetNet on a standard Intel i7-6700 CPU @ 3.40 GHz processor and about 26 hours on GPU.

In this work, we introduce the concept of monotonic non-increasing profile function that scale each layer of the

NN in order to allow the network to dynamically learn the best attenuation strategy for its own weights during training. By doing so, we introduce sparsity in the DNN, which results in a significant complexity saving at inference. Our focus is on DNN designs that unfold iterative projected gradient descent unconstrained optimization for massive MIMO ML-based detection. While methods of artificial “suppression” of neurons during training are known to create sparsity and can be detrimental to inference accuracy [34], [39], we show that, by tuning these profile function appropriately, we can provide a control mechanism that trades off DNN complexity for detection accuracy in a scalable manner. Our contributions are summarized below:

- We introduce a weight scaling framework for DNN-based MIMO detection. Our approach is realized by adjusting layer weights through monotonic profile functions. The original DNN design is based on DetNet, i.e., unfolding a projected gradient descent scheme [27]. We term our proposal the weight-scaling neural-network based MIMO detector (WeSNet).
- In order to allow for entire layers to be abrogated in a controllable manner during inference, we introduce a regularization approach that imposes constraints on the layer weights. This allows for reduction of the model size and the resulting computational complexity with graceful degradation in the detection accuracy.
- To improve the performance of WeSNet, we introduce a learnable accuracy-complexity design, where the weight profile functions themselves are made trainable in order to prevent vanishing gradient due to changes in the values of activations. This improves the detection accuracy of the WeSNet at the cost of increased memory due to increase in the model parameters.
- Finally, we present a comprehensive complexity analysis of WeSNet inference in relation to both DNN-based MIMO detection as well as traditional detectors. Our study and results show that under the same experimental conditions, WeSNet with 50% of the layer weights outperforms the detection accuracy of DetNet while offering 51.43% reduction in complexity and close to 50% reduction in model size. Furthermore, its detection accuracy is similar to SDR with nearly 10-fold reduction of computational complexity.

The remainder of the paper is structured as follows: System model and the review of traditional MIMO detectors are presented in Section II. The proposed approach is introduced in Section III and extensions are proposed in Section IV. Simulations and complexity results are presented in Section VI. Finally, Section VII summarizes and concludes the paper.

II. SYSTEM MODEL AND REVIEW OF MIMO DETECTORS

Consider a communications system with N_t transmit and N_r receive antennas. The received signal is modelled

using a standard MIMO channels equation as

$$\bar{\mathbf{y}} = \bar{\mathbf{H}}\bar{\mathbf{s}} + \bar{\mathbf{n}} \quad (1)$$

where the received complex symbol vector is: $\bar{\mathbf{y}} \in \mathbb{C}^{N_r \times 1}$, and the corresponding transmitted symbol vector, is $\bar{\mathbf{s}} \in \mathbb{C}^{N_t \times 1}$. $\bar{\mathbf{H}} \in \mathbb{C}^{N_r \times N_t}$ is Rayleigh fading channel matrix and the additive white Gaussian noise (AWGN) vector $\bar{\mathbf{n}} \in \mathbb{C}^{N_r \times 1}$ with zero mean and variance σ^2 . For convenience and ease of implementation, the channel model is redefined over the real domain as

$$\mathbf{y} \equiv \begin{bmatrix} \Re\{\bar{\mathbf{y}}\} \\ \Im\{\bar{\mathbf{y}}\} \end{bmatrix} \in \mathbb{R}^{2N_r \times 1}, \quad \mathbf{s} \equiv \begin{bmatrix} \Re\{\bar{\mathbf{s}}\} \\ \Im\{\bar{\mathbf{s}}\} \end{bmatrix} \in \mathbb{R}^{2N_t \times 1}$$

$$\mathbf{H} \equiv \begin{bmatrix} \Re\{\bar{\mathbf{H}}\} & -\Im\{\bar{\mathbf{H}}\} \\ \Im\{\bar{\mathbf{H}}\} & \Re\{\bar{\mathbf{H}}\} \end{bmatrix} \in \mathbb{R}^{2N_r \times 2N_t} \quad (2)$$

With this representation, (1) can be expressed in terms of real-valued vectors and matrix as

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n} \quad (3)$$

Below, we briefly summarize the four MIMO detectors that form our performance benchmarks.

A. Maxim Likelihood-detector (ML-detector)

The ML detector is known as the optimal detector [7]. However, finding the estimated symbols involves searching over all the possible transmitted digital symbols of $|\mathbb{M}|^{N_t}$ vectors (where \mathbb{M} is the size of the modulated symbol and N_t is number of transmit antennas). Therefore, the complexity of this detector grows exponentially with the number of transmit antennas and the modulation order (constellation size), and quickly becomes impractical in real systems. ML-detector minimizes the Euclidean distance between the received and the transmitted symbols

$$\hat{\mathbf{s}} = \arg \min_{\mathbf{s} \in \mathbf{S}} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2 \quad (4)$$

B. Linear Detectors

Linear Detectors [12] can be derived via the generalized pseudo-inverse equation

$$\hat{\mathbf{s}}_{\text{linear}} = (\mathbf{H}^H \mathbf{H} + \alpha \mathbf{I})^{-1} \mathbf{H}^H \mathbf{y} \quad (5)$$

where \mathbf{I} is the identity matrix, $\alpha = \frac{\sigma_n^2}{\sigma_s^2}$ for MMSE and $\alpha = 0$ for ZF. The transmitted symbol in ZF is affected by the presence of colored noise and leads to performance degradation. On the other hand, MMSE suppresses the noise enhancement as shown in (5), but assumes knowledge of the noise variance.

C. Optimization Based Detector

Typical optimization based detectors are based on quadratically constrained quadratic programming (QCQP) to provide detection at lower computational cost than an ML detector [41]. Equation (4) can be written as

$$\min_{\mathbf{s} \in \mathbf{S}, \mathbf{s} \in \mathbb{R}^{2N_t \times 1}} (\mathbf{H}^T \mathbf{H} \mathbf{s} - 2\mathbf{s}^T \mathbf{H}^T \mathbf{y} + \|\mathbf{y}\|^2) \quad (6)$$

s.t. $\mathbf{s} \in \mathbf{S}, \forall s_i \in \{1, \dots, 2N_t\}$

This can be made convex via semidefinite relaxation (SDR), by removing the above rank constraints

$$\min_{\mathbf{X}} \text{trace}(\mathbf{L}\mathbf{X})$$

s.t. $\text{diag}(\mathbf{X}) = \mathbf{I}$
 $\mathbf{X}(2N_t + 1, 2N_t + 1) = 1$
 $\mathbf{X} \succeq 0$
 $\text{rank}(\mathbf{X}) = 1$ (7)

where:

$$\mathbf{L} = \begin{bmatrix} \mathbf{H}^T \mathbf{H} & -\mathbf{H}^T \mathbf{y} \\ -\mathbf{y}^T \mathbf{H} & \|\mathbf{y}\|^2 \end{bmatrix}; \quad \mathbf{X} = \mathbf{s}^T \mathbf{s}$$

The difficulty of solving (7) lies with the rank-1 constraint (nonconvex constraint). To solve this, the rank-1 constraint is dropped, thus reducing the problem to a formulation that can be addressed by semidefinite programming:

$$\min_{\mathbf{X}} \text{trace}(\mathbf{L}\mathbf{X})$$

s.t. $\text{diag}(\mathbf{X}) = \mathbf{I}$
 $\mathbf{X}(2N_t + 1, 2N_t + 1) = 1$
 $\mathbf{X} \succeq 0$ (8)

To make sure the resulting solution obeys the rank constraints of the original problem, the above optimization is typically followed by the randomization iterations [41].

D. Deep MIMO Detector (DetNet)

Most relevant to our work is the detector that uses a DNN architecture [27], [28]. The unique property of this detector is its ability to scale for higher dimension signals [28]. Learning that leads to good detection is achieved by finding a set of parameters Φ that minimize the difference between the transmitted symbols and their estimates, and is given by

$$\min_{\Phi} \mathbf{E}\{\mathcal{L}(\mathbf{s}; \hat{\mathbf{s}}(\mathbf{H}, \mathbf{y}) : \Phi)\} \quad (9)$$

where $\mathcal{L}(\cdot)$ is the error loss function defined over \mathbf{s} and \mathbf{y} . Equation (9) is the solution to ML detector. Based on it, the architecture of the DetNet is designed such that the estimated symbols parametrized by α are

$$\{\hat{\mathbf{s}}(\mathbf{H}, \mathbf{y}) : \Phi\} = \begin{cases} \text{Loss function} \\ \mathbf{y}\mathbf{H} \mapsto \{\text{Constellation Vector}\}^{N_t} \end{cases} \quad (10)$$

The estimate of the received symbols can be obtained from a trained neural network by an update rule using an iterative projected gradient descent formulation [28]. For a function defined by $f(x, y)$, the estimate of \mathbf{x} and \mathbf{y} over r -th layers can be found from gradient descent using the following update rule:

$$\mathbf{x}_{r+1} = \mathbf{x}_r - \eta \frac{\partial f(x, y)}{\partial x} \quad (11a)$$

$$\mathbf{y}_{r+1} = \mathbf{y}_r - \eta \frac{\partial f(x, y)}{\partial y} \quad (11b)$$

where η is the learning rate. By the same argument, the gradient descent optimization of (4) can be expressed as

$$\hat{\mathbf{s}}_{r+1} = \hat{\mathbf{s}}_r - \eta_r \left. \frac{\partial \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2}{\partial \mathbf{s}} \right|_{\mathbf{s}=\hat{\mathbf{s}}_r} \quad (12)$$

Expanding (12), we obtain:

$$\hat{\mathbf{s}}_{r+1} = \hat{\mathbf{s}}_r - 2\eta_r \mathbf{H}^T \mathbf{y} + 2\eta_r \mathbf{H}^T \mathbf{H} \hat{\mathbf{s}}_r \quad (13)$$

With $\mathbf{H}^T \mathbf{y}$, $\mathbf{H}^T \mathbf{H}$ and \mathbf{s} as inputs and the application of some nonlinearity, (13) is converted to a multi-layer perceptron also known as fully connected NN defined by the following equations

$$\mathbf{v} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

$$\mathbf{z} = \Omega(\mathbf{v}) \quad (14)$$

where \mathbf{x} is the input, \mathbf{W} is the layer/channel weight, \mathbf{b} the bias, \mathbf{y} is the layer output and $\Omega(\cdot)$ is some non-linearity. We can write the NN equivalent model of (13) as

$$\mathbf{u}_r = \Theta(\mathbf{W}_{1r}\mathbf{x} + \mathbf{b}_{1r}) \quad (15)$$

where:

$$\mathbf{x}_r = \Pi(\mathbf{H}^T \mathbf{y}, \mathbf{H}^T \mathbf{H} \mathbf{s}_r, \mathbf{s}_r, \mathbf{a}_r) \quad (16)$$

$$\hat{\mathbf{s}}_{r+1} = \Psi(\mathbf{W}_{2r}\mathbf{u}_r + \mathbf{b}_{2r}) \quad (17)$$

$$\hat{\mathbf{a}}_{r+1} = \mathbf{W}_{3r}\mathbf{u}_r + \mathbf{b}_{3r} \quad (18)$$

$\Pi(\cdot)$ is the concatenation function, \mathbf{W}_r , \mathbf{W}_{2r} and \mathbf{W}_{3r} are the weights of the input, detection and auxiliary layers, \mathbf{a}_r is auxiliary input, $\Theta(\cdot)$ and $\Psi(\cdot)$ are nonlinear and piece-wise linear sign functions, respectively. These equations represent a single layer of DetNet. The trainable parameters that are optimized during training are defined by

$$\Phi = (\mathbf{W}_{1r}, \mathbf{W}_{2r}, \mathbf{W}_{3r}, \mathbf{b}_{1r}, \mathbf{b}_{2r}, \mathbf{b}_{3r})_{r=1}^L \quad (19)$$

III. PROPOSED WEIGH-SCALING NEURAL-NETWORK BASED MIMO DETECTOR (WeSNet)

A. Weight Scaling Vector Coefficient (WSVC)

A WSVC is computed by applying monotonically non-increasing coefficients (known as profile function coefficients) to the layer weights during the forward propagation. This results in prioritizing the selection of the layer weights in decreasing fashion from the most significant to least significant. Mathematically, for two given vectors, $\mathbf{x} = [x_1, x_2, \dots, x_N]^T$ and $\mathbf{y} = [y_1, y_2, \dots, y_N]^T$, if β is the vector of the profile coefficients, WSVC is the pruned version of the form

$$\sum_{i=1}^N \beta_i x_i y_i = \beta_1 x_1 y_1 + \beta_2 x_2 y_2, \dots + \beta_N x_N y_N \quad (20)$$

In a standard fully connected NN, the output of the feed forward pass is given by

$$z_j = \sum_{i=1}^N W_{ji} x_i + b_j \quad (21)$$

where i and j are the input and output dimensions (size of the neurons) respectively; x_i is the i -th input components, W_{ji} is the channel or layer weight corresponding to the j th output and b_j is the output bias. The corresponding WSVC is derived by

$$z_j = \sum_{i=1}^N \beta_i W_{ji} x_i + b_j \quad (22)$$

Fig. 1 shows the difference between the feed forward computations of a layer of an MLP and the MLP augmented by WSVC. The part of the WSVC corresponding to significant layer weights is indicated by the light colored shaded region on the bottom-right side of the figure. The example shows that, via the WSVC, we can compute and use only one-third of the channel/layer weights out of the N layer dimension, as the remaining two-thirds of the weights are attenuated and can be dropped.

B. Weight Coefficient Profile Function

We begin by introducing two non-increasing monotonic profile functions (Linear and Half-Exponential functions) for the weight coefficients [39].

1) *Linear Profile Function*: The profile function coefficients comprise samples of the linear function:

$$\beta_i = 1 - \frac{i}{N}; \forall i = 1, 2, \dots, N \quad (23)$$

where N is the layer size.

2) *Half-Exponential Profile Function*: This function attenuates coefficients corresponding to half of the channel via an exponential decay function. The implication of this is that it allows the network training to adjust the gradient

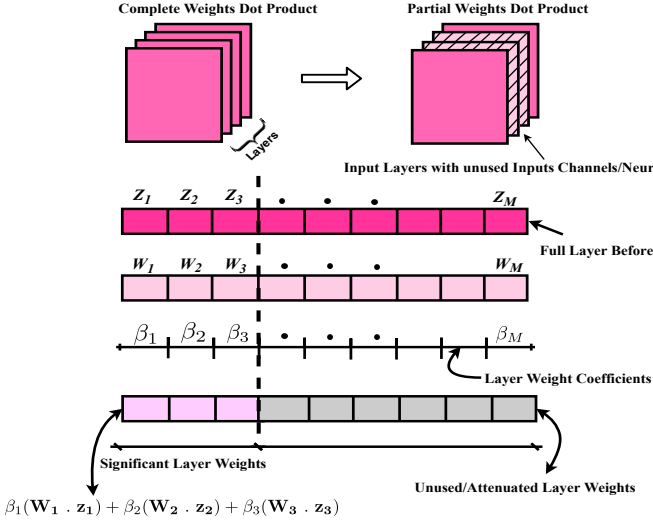


Fig. 1: WSVC in a single layer of an MLP allowing for attenuated layer weights to (optionally) be dropped.

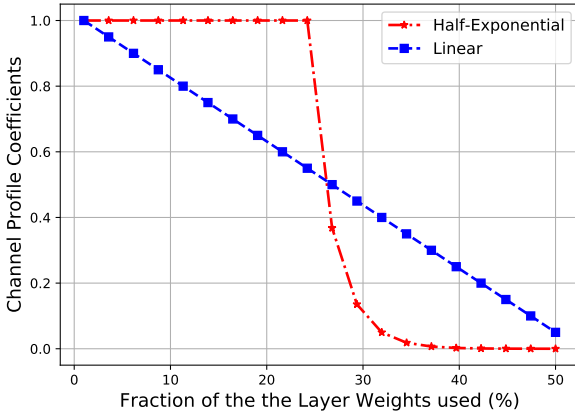


Fig. 2: Profile Coefficients vs Layer Weights used.

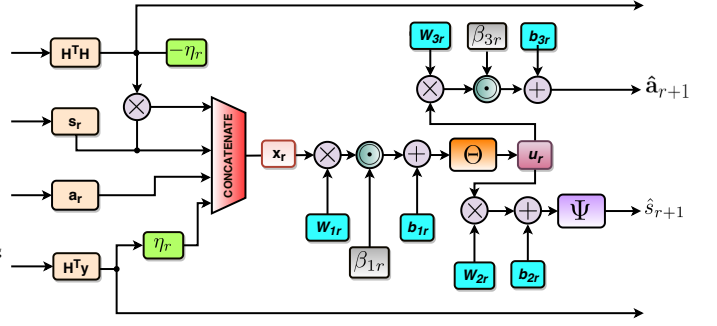
flow such that important weights are retained in the non-attenuated half of each layer and the less important ones in the exponentially-attenuated half.

$$\beta_i = \begin{cases} 1 & \text{if } i \leq \frac{N}{2} \forall i = 1, 2, \dots, N \\ \exp\left(\frac{N}{2} - i - 1\right) & \text{otherwise} \end{cases} \quad (24)$$

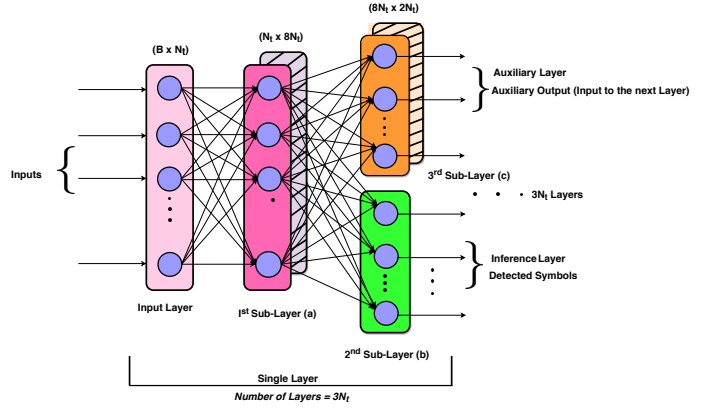
The profile functions are shown in Fig. 2.

C. Structure of the WeSNet-Detector

WeSNet is a nonlinear estimator designed by unfolding the ML metric using a recursive formulation of the projected gradient descent optimization. Our proposed detector applies the profile coefficients on the existing DetNet. Such a modification reduces the computational complexity for training the detector. We apply profile coefficients to (15) and (18) to obtain the following non-



(a) Single r -th layer WeSNet-detector.



(b) Single Layer WeSNet Architecture.

Fig. 3: WeSNet Model Architecture

linear WSVCs over i -th and j -th inputs of the r -th layer of the first and third sublayers respectively

$$u_{r(j)} = \Theta \left\{ \sum_{i=1}^N \beta_{1r(i)} W_{1r(j)} x_{r(i)} + b_{1r(j)} \right\} \quad (25)$$

$$\hat{a}_{(r+1)k} = \sum_{j=1}^M \beta_{3r(j)} W_{3r(kj)} u_{r(j)} + b_{3r(k)} \quad (26)$$

where j and k are the outputs of the first and third sublayers respectively, while N and M are their corresponding sizes.

WeSNet has $3N_t$ layers with each layer having three sublayers, the input layer, the auxiliary and the detection layer. The layer weights of the last sub-layer (detection layer) described by (17) is not scaled in order to maintain the full dimension of the detected symbols as originally transmitted. The flowchart of a single layer WeSNet based on the (17), (25) and (26) is shown in Fig. 3(a). The complete architecture of the WeSNet is shown in Fig. 3(b). Since the error estimation of the ML-detector does not require the knowledge of the noise variance, the loss function of WeSNet is derived as the weighted sum of the detector's errors normalized with the loss function of the standard linear inverse detector (ZF) [27] as

$$\mathcal{L}(s; \hat{s}(H, y : \Phi)) = \sum_{r=1}^L \log(r) \frac{\|s - \hat{s}_r\|^2}{\|s - \tilde{s}\|^2} \quad (27)$$

IV. INTRODUCING ROBUSTNESS THROUGH REGULARIZED WeSNet (R-WeSNet)

Regularization in machine learning is the technique of imposing some penalty on the weights of a model in order to avoid overfitting. Common regularizing operators are the L_1 and L_2 norms [42]. Both are used to increase the robustness against collinearity between the prediction and the ground truth by adding some additional constraints to the cost function being minimized. The choice of regularization is specific to the type of task, but for most deep learning architectures, the L_2 norm is the preferred candidate.

A. Motivation

In this work, in addition to providing stability to the learning algorithm, regularization also allows for scalable model size reduction during inference. Given that our aim is to introduce sparsity so that layers (and parts of layers) with few non-zero coefficients can be removed to scale complexity, we propose the use of the L_1 norm. The choice of L_1 is explained by the geometry depicted in Fig. 4. For simplicity, suppose two parameters W_1 and W_2 are defined by some contour S . Then, L_1 and L_2 can be expressed as

$$\begin{aligned} L_1 &: \lambda|W_1| + \lambda|W_2| \leq S \\ L_2 &: \lambda W_1^2 + \lambda W_2^2 \leq S \end{aligned}$$

The system in Fig. 4 is hypothesized by problem space \mathcal{H} , whose solution is the set of points where the curve meets the constraints. The solution corresponding to the L_2 norm is represented by the circle, whose solution lies tangential to any point on its circumference (see Fig. 4b). This means that L_2 has a closed form solution. On the other hand, for L_1 , the only feasible solutions are constrained to the axes (i.e corners), as shown by the diamond shape in Fig. 4a. For example, the value of W_1 is obtained when $W_2 = 0$. This means that the influence of W_2 in the solution is limited, which forces some of the coefficients to be zero and leads to sparsity. For high dimensional space, many coefficients will be exactly zero. This unique property of L_1 makes it more appealing and robust than L_2 , as well as a better candidate for feature selection [43].

B. Proposed Loss Function

Following this motivation, the loss function of (27) is modified such that an L_1 penalty term is imposed on the weights:

$$\mathcal{L}(s; \hat{s}(H, y : \Phi)) = \sum_{r=1}^L \log(r) \frac{\|s - \hat{s}_r\|^2}{\|s - \tilde{s}\|^2} + \lambda f(\beta_r, \tilde{W}_r) \quad (28)$$

where λ is the regularization parameter that controls the importance of sparsity in the layers weights and $f(\beta_r, \tilde{W}_r)$

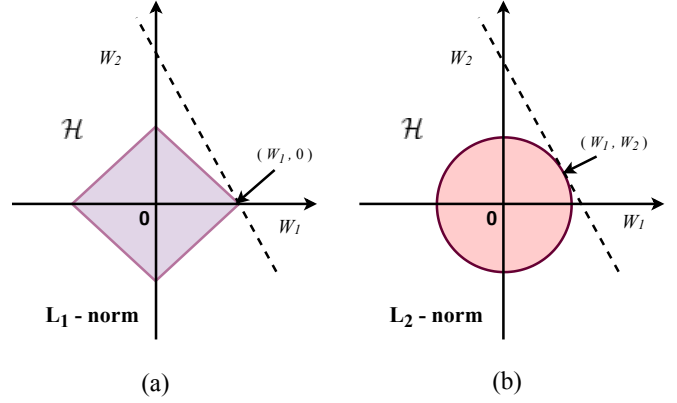


Fig. 4: L_1 and L_2 Regularizations for two Parameters. [43]

is the function of layer weights with respect to the neuron connections between adjacent layers, and is given by

$$f(\beta_r, \tilde{W}_r) = \sum_{r=l}^L \log(1 + (r-1)|\beta_r \tilde{W}_r|) \quad \forall r = l, \dots, L \quad (29)$$

where:

$$(\beta \tilde{W})(r, k) = \sum_{k=1}^{l_{\text{sublayers}}} \beta_{kr} W_{kr} \quad \forall k = 1, \dots, l_{\text{sublayers}} \quad (30)$$

$r = l$ is the initial layer from which the penalty is imposed

k is the number of sub-layers

$l_{\text{sublayers}}$ is the number of sub-layer in each layer block and is one of the profile functions of (23) and (24).

In the proposed loss function of (29), we opt for the logarithm function in order to: (i) avoid the β profile functions converging into the constant unity function and (ii) prevent gradient explosion, i.e., having the logarithmic decay act as a regularizer [44], [45]. Together with the use of the L_1 norm, these two aspects enforce sparsity in the network weights corresponding to the lowest part of the β profile functions when the regularization parameter (λ) is adequately large [35]. In this way, the model size can be scaled down by expunging some layers deterministically during inference, which reduces memory and computational requirements during model deployment with graceful degradation in detection accuracy.

C. WeSNet with Learnable Weight Profile Coefficients (L-WeSNet)

To improve the robustness of the WeSNet against vanishing gradients and possible gradient explosion, the weight profile functions themselves are made trainable parameters, whose values are optimized during the network training process. This allows for significantly wider exploration of appropriate scaling functions than the pre-determined profile functions presented earlier, albeit at the expense of computational complexity during training. To achieve this, (19) is modified to include profile weight functions as learned parameters.

TABLE I: Matrix-vector floating point operations [46]

Expression	Description	Multiplications	Summations	Total Flops
$\alpha \mathbf{a}$	Vector Scaling	N		N
$\alpha \mathbf{A}$	Matrix Scaling	MN		MN
$\mathbf{A}\mathbf{b}$	Matrix-Vector Prod.	MN	$M(N-1)$	$2MN - M$
$\mathbf{A}\mathbf{B}$	Matrix-Matrix Prod.	MNL	$ML(N-1)$	$2MNL - ML$
$\mathbf{A}\mathbf{D}$	Matrix-Diagonal Prod.	MN		MN
$\mathbf{a}^H \mathbf{b}$	Inner Prod.	N	$N-1$	$2N-1$
$\mathbf{a}\mathbf{c}^H$	Outer Prod.	MN		MN
$\mathbf{A}^H \mathbf{A}$	Gram	$\frac{MN(N+1)}{2}$	$\frac{N(M-1)(N+1)}{2}$	$MN^2 + N(M - \frac{N}{2}) - \frac{N}{2}$
$\ \mathbf{A}\ ^2$	Euclidean norm	MN	$MN-1$	$2MN-1$
\mathbf{Q}^{-1}	Inverse of Pos. Definite	$\frac{N^3}{2} + \frac{3N^2}{2}$	$\frac{N^3}{2} - \frac{N^2}{2}$	$N^3 + N^2 + N$ Including N roots

TABLE II: MIMO detectors' complexity per symbol slot time.

MIMO Detector	Number of Flops Operation
ZF	$\left(\frac{56}{3}\right) N_t^3 + 38N_t^2 + \left(\frac{28}{3}\right) N_t$
MMSE	$\left(\frac{56}{3}\right) N_t^3 + 40N_t^2 + \left(\frac{34}{3}\right) N_t + 1$
ML	$ \mathbf{S} ^{N_t} (8N_t^2 + 8N_t - 2)$
SDR	$\left(13N_t^3 + 25N_t^2 + 17N_t + 4\right) N_{\text{iterations}}$ [7], [47]
WeSNet	$[(\beta N_t (128N_t + 5) + 9N_t)] L$, L = number of layers
DetNet	$[(N_t (128N_t - 2))] L$

TABLE III: Simulation settings

Parameters	Values
First Sublayer Dimension	$8N_t = 240$
Second Sublayer Dimension	$N_t = 30$
Third Sublayer Dimension	$2N_t = 60$
Number of Layers	$L = 3N_t = 90$
Fraction of the Layer Weight used	β
Training Samples	500000
Batch Size	5000
Test Samples	50000
Training SNR range	8dB - 14dB
Test SNR range	0dB - 15dB
Optimizer	SGD with Adam
Learning Rate	0.001
Weight Initializer	Xavier Initializer
Number of Training Iterations	25000
Number of Monte Carlo during inference	200

$$\tilde{\Phi} = (\mathbf{W}_{1r}, \mathbf{W}_{2r}, \mathbf{W}_{3r}, \mathbf{b}_{1r}, \mathbf{b}_{2r}, \mathbf{b}_{3r}, \beta_r)_{r=1}^L \quad (31)$$

It is important to note that the monotonicity during training and gradient update is maintained by the shape of the functions of (23) and (24).

V. COMPLEXITY ANALYSIS

WeSNet is a truncated version of DetNet, and the detection is performed at the inference layer (see Fig. 3(b)) by feed forward computation and subsequent application of the soft sign activation function. The computational cost of WeSNet inference is derived based on the cost of operations of an MLP (please see Appendix A for the details). Our proposed model has 90 layers formed by stacking block of layers, each consisting of three layers DNN. The propagation error is found by computing the derivative of the cost function with respect to the parameters in each block. The computational complexity is specifically measured by the number of operations based on the detector's model. Suppose $\mathbf{A} \in \mathbb{C}^{M \times N}$ and $\mathbf{B} \in \mathbb{C}^{N \times L}$ are arbitrary matrices. $\mathbf{D} \in \mathbb{C}^{M \times N}$ is a diagonal matrix, $\mathbf{a}, \mathbf{b} \in \mathbb{C}^{N \times 1}$ and $\mathbf{c} \in \mathbb{C}^{M \times 1}$ are arbitrary vectors and $\mathbf{Q} \in \mathbb{C}^{N \times N}$ is positive definite. The required number of FLOPs operations of the standard algebraic expressions of interest to this work are summarized in Table I.

We use the previous equations and the complexity of the feed-forward inference formulation as detailed in Appendix A to compute the number of floating point operations of each MIMO detector. Our results are summarized in Table II, and correspond to the following standard assumptions:

- 1) One addition, subtraction of a real number is equal to one computational operation.

- 2) One multiplication of a complex number is equivalent to four real number multiplications and two real number addition.
- 3) One addition or subtraction of a complex number is equivalent to two real number additions.
- 4) One division of a complex number is is equivalent to eight real number multiplications and four additions.

Since only a certain fraction of the inputs are used to compute the layer weights of WeSNet and R-WeSNet, most of the operations involved in the feed-forward computations are either sparse vector-matrix multiplication and/or sparse matrix-matrix multiplication. We can evaluate the asymptotic complexity as follows; $\beta_{1r} \mathbf{W}_{1r}$, $\beta_{2r} \mathbf{W}_{2r}$ and $\beta_{3r} \mathbf{W}_{3r}$ for detecting a single received symbol are computed by matrix-vector and matrix-matrix multiplications

as $\mathcal{O}\left(\sum_{r=1}^L \mathbf{u}_{1r} + \mathbf{s}_{2r} + \mathbf{a}_{3r}\right) = \mathcal{O}(n^3 + n^2) = \mathcal{O}(n^3)$.

VI. SIMULATION AND NUMERICAL RESULTS

In this section, we present the experimental setup and the performance of the WeSNet under different profile functions and their trainable versions. Amongst deep learning based MIMO detectors, DetNet achieves the best complexity-accuracy performance and also forms the basis of our proposal. Therefore, we deploy and benchmark WeSNet against DetNet, but also present performance comparisons against other classical detectors.

A. Simulation Setup

WeSNet is implemented in *Tensorflow* 1.12.0 [48] with python 3.6.7. Since deep learning libraries only support real number computations, we use real-valued representation of the random signals and fading channel to generate the training and test datasets. The detector is evaluated under the condition of 30 transmit and 60 receive antennas (60, 30). To ensure a fair comparison with the benchmark model, we use the same simulation settings, which are summarized in Table III. The classical detectors against which we further juxtapose the performance of our proposed scheme are:

- 1) Linear detectors (**ZF** and **MMSE**) implemented based on [10].
- 2) The optimal detector (**ML**) and optimization based detector (**SDR**) based on relaxed semidefinite programming as proposed in [8] and [47] respectively.
- 3) The deep learning based MIMO detector as proposed by Samuel *et al.* [27]

B. Training

The training dataset is stochastically generated from random normal distribution drawn from BPSK constellation $s \in \{\pm 1\}^{N_T}$, a random white Gaussian noise from a uniform distribution over a wide range of SNR values $\mathcal{U}(8\text{dB} - 14\text{dB})$ and the corresponding received symbols are generated from the standard wireless channel model. Similarly, the test dataset is generated using a uniform random white Gaussian noise over $\mathcal{U}(0\text{dB} - 15\text{dB})$. We train the model for 25000 iterations with 5000 batch size for each iteration on a standard Intel i7-6700 CPU @ 3.40 GHz processor and use Adam Optimizer [15] for gradient descent optimization. It takes between 17-19 hours to train WeSNet with 20% and 50% profile weight coefficients respectively. As described in Section II, we assume an unknown noise variance and therefore generate the noise vector from a random uniform distribution over the training SNR values $\mathcal{U}(SNR_{\min}, SNR_{\max})$. This allows the network to learn over a wide range of SNR conditions.

C. Performance of a WeSNet Realization with Half-Exponential and Linear Profile Functions

Fig. 5 shows the performance of WeSNet with the half-exponential (WeSNet-HF) and linear (WeSNet-L) profile

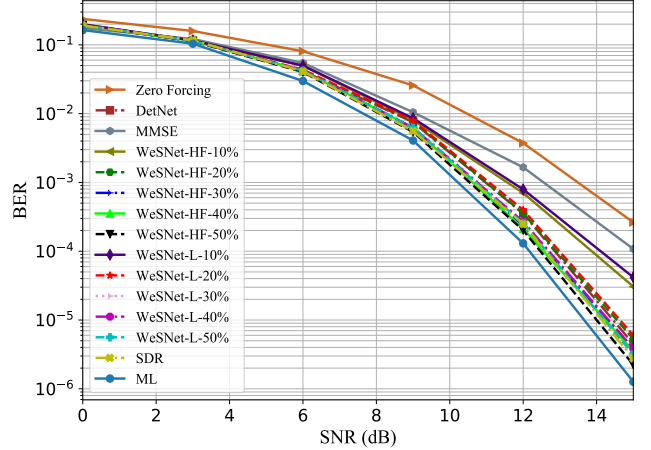


Fig. 5: BER comparison of the proposed DNN MIMO Detectors (WeSNet-HF, WeSNet-L), DetNet, ZF, MMSE, SDR and ML.

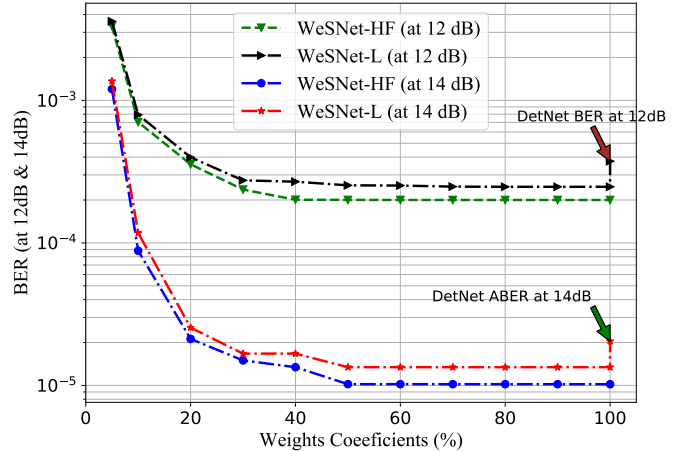


Fig. 6: BER vs Percentage Weight Profile Coefficients for WeSNet.

functions of (23) and (24) when retaining increased percentage of inference layers (as marked in the corresponding legends). The benchmarks comprise DetNet, ZF, MMSE, SDR and ML detectors. Both linear and half-exponential profile WeSNet have comparable performance at lower SNR and profile coefficients between 20% - 30% of the layer weights. As expected, the addition of more profile coefficients increases WeSNet's detection accuracy, but performance saturates after 60% of the coefficients. However, we observe an appreciable difference at higher SNR as more profile weight coefficients are added. At 10^{-3} BER, WeSNet can be trained with only 10% of the layer weights and still outperforms ZF and MMSE by 1.68dB and 0.79dB respectively. Overall, WeSNet with only 20% of the layer weights (WeSNet-HF-20%) achieves virtually the same performance as our benchmark model (DetNet). In fact, with 50% profile weight coefficients (WeSNet-HF-50%), WeSNet outperforms DetNet, producing the accuracy of symbol detection equivalent to SDR. This

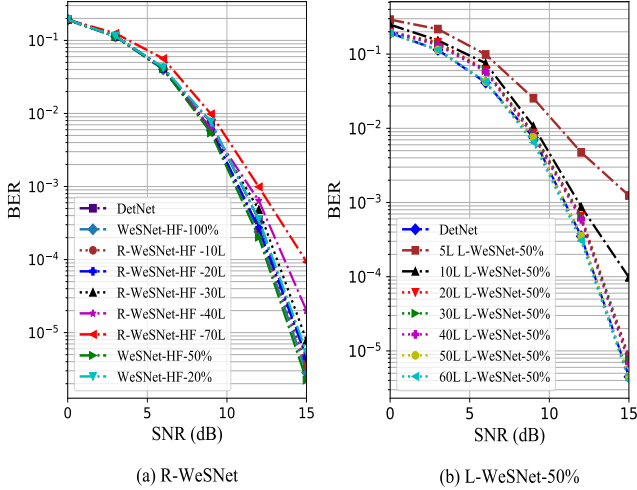


Fig. 7: Performance comparison of R-WeSNet, L-WeSNet trained with 50% profile weight coefficients as a function of layers, WeSNet-HF-50%, WeSNet-HF-20% and DetNet detectors.

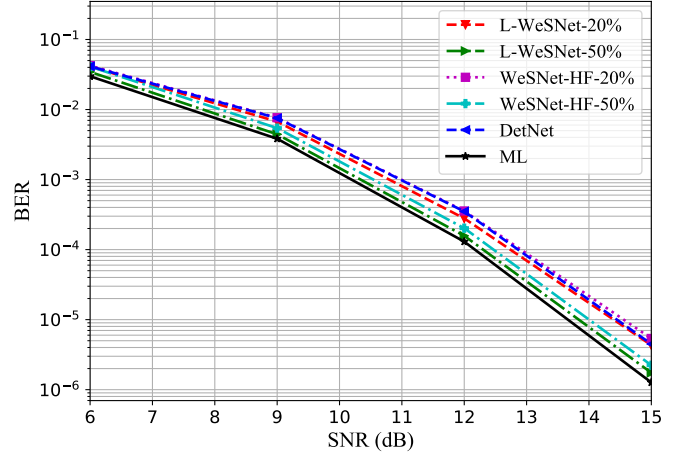


Fig. 9: Performance comparison of L-WeSNet, WeSNet, DetNet and ML detectors.

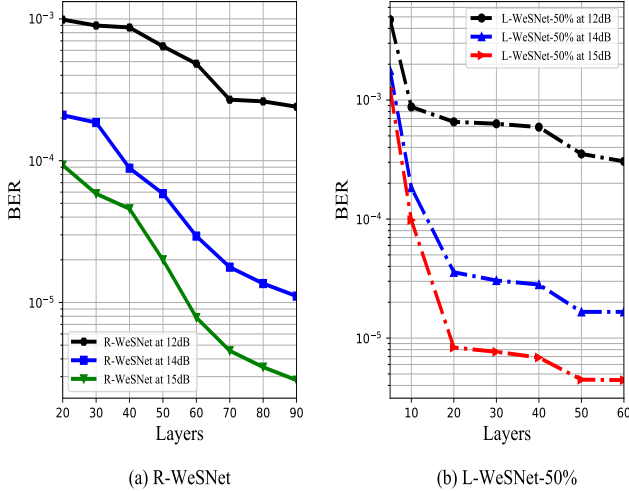


Fig. 8: BER for R-WeSNet-HF and L-WeSNet vs number of layers.

gain is an experimental validation that weight profile functions also act as regularizers, i.e., beyond their sparsity-enforcing property, they also avoid overfitting when the model size grows.

In Fig. 6, we show the performance of the WeSNet-HF and WeSNet-L parametric to the utilized layer weight profile coefficients at 12dB and 14dB SNRs. Both outperform DetNet and WeSNet-HF surpasses the WeSNet-L at high SNR. For example, at 14dB and 10^{-5} BER, it has gain margin of 0.312dB over the WeSNet-L. We also observe that the BER at 12dB and 15dB SNR improves as more profile coefficients are added, but saturates at 50% due to weight saturation. This illustrates that, with the addition of profile weight coefficients, at higher SNR the size of the WeSNet can be scaled down during training by almost 40%-50% and still achieve almost identical detection accuracy to the full architecture that retains 100% of the weights during training.

D. Performance Evaluation of R-WeSNet

To examine in more detail the performance of our approach against the “direct” approach of removing entire layers by enforcing penalty on the weights through L_1 regularization, Fig. 7a shows BER-SNR performance curves of the full WeSNet (WeSNet-HF-100%), DetNet and WeSNet when removing entire layers. The figure shows that removing 70 - 40 layers (R-WeSNet-HF-70L and R-WeSNet-HF-40L) results in considerable loss of accuracy as compared to the corresponding WeSNet-HF models (WeSNet-HF-20% and WeSNet-HF-50%). Nevertheless, 20 - 30 layers (R-WeSNet-HF-20L and R-WeSNet-HF-30L) can be removed while still achieving BER-SNR performance slightly greater than the DetNet’s. It can be noticed that a R-WeSNet-HF-10L (with 10 layers shortfall) outperforms both WeSNet-HF-50% and DetNet.

E. BER Performance of L-WeSNet

In order to examine the performance of our approach when the weight profile coefficients are made learnable (L-WeSNet), Fig. 7b presents the performance of with 50% learnable weight coefficients (L-WeSNet-HF-50%) over different number of layers. It can be seen that there is a remarkable performance improvement as the size of the network grows from 5 layers to 60 layers. For instance, at 7.2×10^{-2} BER, we observe margin of 2.8dB between 5 to 30 layers. On the other hand, accuracy remains fairly consistent when going from 20 to 40 layers. Our study also shows that L-WeSNet-50% produces the same accuracy as DetNet trained with full 90 layers. This means that, for the studied problem, an efficient deep MIMO detector can be designed with 50% trainable weight coefficients and 50 layers.

Fig. 8 shows the average BER for both R-WeSNet and L-WeSNet against the number of layers at 12dB, 14dB and 15dB SNRs respectively. At 12dB SNR (Fig. 8a), removing 10 - 30 layers during feed forward inference does

not significantly change the performance as compared to at 14dB and 15dB. However, at 12dB and 15dB, we observe a sharp decrease in performance from 70 - 20 layers. The BER is about 10^{-4} at 15dB and less than 10^{-3} at 14dB with 40 layers removed. We also discern that, at higher SNR, model size can be reduced significantly by removing up to 50 layers during the inference with slightly loss of accuracy.

No rules or analysis exists to precisely determine the size of a neural network (i.e., number of neurons, layers, or layer parameters) for a specific task. Therefore, we train WeSNet-HF with trainable weight coefficients over the different number of layers to determine the conditions under which we can obtain the minimum BER. The average BER is evaluated at 12dB, 14dB and 15dB for each layer configuration as shown in Fig. 8b. It can be seen that the BER falls off quickly from 5-60 layers. The BER at 14dB is approximately 3×10^{-5} . This value is reasonably constant from 20-30 layers and goes down as more layers are added. It can also be seen that at 15dB, L-WeSNet-50% produces nearly 10^{-5} BER with only 20 layers.

In Fig. 9, we compare the BER-SNR performance of WeSNet and L-WeSNet both trained over the entire layers with 20% and 50% of the profile weight coefficients (L-WeSNet-20% and L-WeSNet-50%) against other benchmark models. Our study shows that WeSNet with trainable weight profile functions outperforms the one with non-trainable functions. This comes at the expense of slightly increased training cost due to the additional number of training parameters. This additional training overhead, however, does not increase the inference complexity of the L-WeSNet over WeSNet's, as the inference architectures are the same, except of the difference in the values of the trained weight scaling values. It can be seen that L-WeSNet-20% at 10^{-3} BER outperforms both DetNet and WeSNet-HF-20% by 0.19dB. Similarly, L-WeSNet-50% yields better detection accuracy over WeSNet-HF-50% and DetNet by 0.22dB and 0.69dB, respectively.

F. Complexity Evaluation of the Proposed Scheme

To associate layer sizes with complexity and number of antennas in the MIMO configuration, Fig. 10a shows the complexity evaluated as the number of FLOPs for WeSNet-HF-100%, WeSNet-HF-50%, WeSNet-HF-20%, DetNet, ZF, MMSE, SDR and ML detectors against the number of transmit antennas. As expected, as the number of antennas increases, the complexity of ML grows exponentially. On the other hand, WeSNet-HF-20% has the lowest computational cost. As far as the model configuration is concerned, equal number of matrix-matrix and matrix-vector floating point operations are performed by both WeSNet-HF-100% and DetNet during the feed forward inference. However, WeSNet-HF-50% and WeSNet-HF-20% are computationally more efficient than DetNet. For example, with 20% - 80% profile weights coefficients, the training of WeSNet-HF is less complex than that of

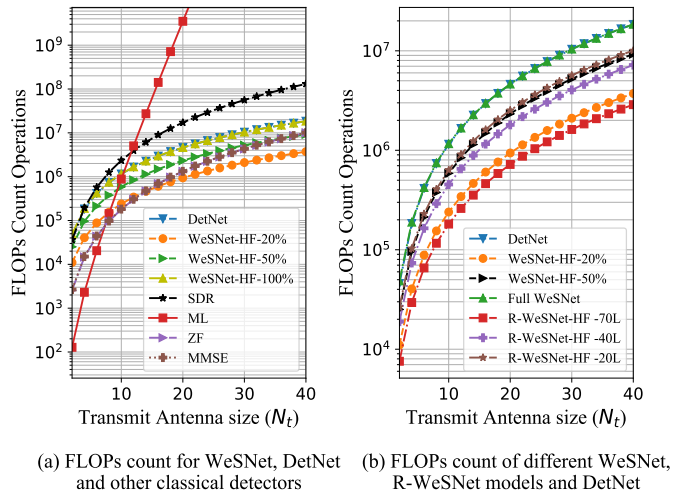


Fig. 10: Computational complexity comparison of the detectors against transmit antenna size.

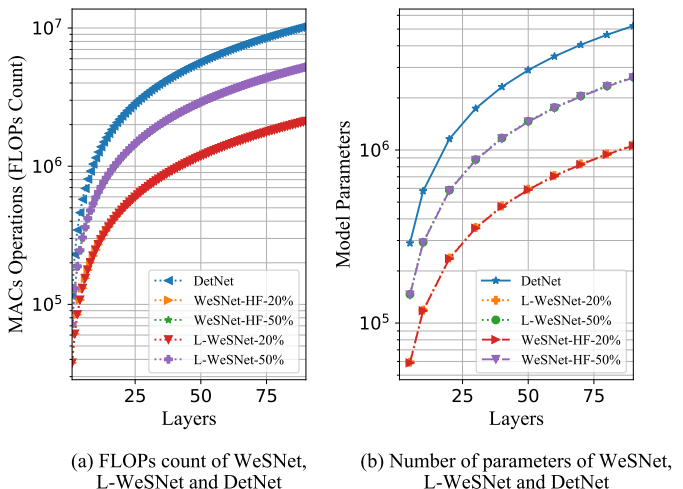


Fig. 11: Complexity comparison of WeSNet, L-WeSNet and DetNet in terms of FLOPs count and model parameters as a function of network layers.

DetNet under the same operating conditions. When a regularized WeSNet-HF-100% is trained and layers are removed deterministically at inference, Fig. 10b shows that the complexity drops, with graceful degradation in performance. Importantly, as expected from prior experiments, the first 30 layers can be abrogated without any significant compromise on the performance.

Fig. 11a depicts the complexity as function of network layers. The computational requirement grows linearly as more layers are added. It can be observed that the WeSNet-HF-50% and WeSNet-HF-20% are less complex than DetNet over the entire range of layers. Our study shows that, at the inference, the complexity of L-WeSNet is not affected by the presence of learnable weight profile functions. Therefore, WeSNet-HF-50% and WeSNet-HF-20% and their corresponding learnable versions (L-WeSNet-50% and L-WeSNet-20%) have the same compu-

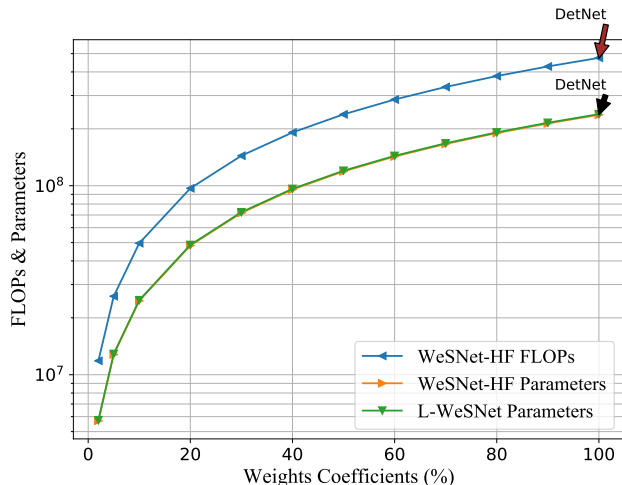


Fig. 12: Total FLOPs and model parameters vs weight profile coefficients.

tational complexity at inference.

Fig. 11b shows the variation of the model size in terms of number of learnable parameters as a function of network layers. For a given layer dimension (number of neurons), the size of the model is determined by the number of layers and the number of trainable parameters. It can be seen that WeSNet, in addition to having better detection accuracy, it is substantially more memory efficient than DetNet and requires less training time under the same experimental conditions.

As more weights profile coefficients are added, the number of FLOPs increases. Fig. 12 shows how the computational cost and model parameters change with the profile weight coefficients. As shown by earlier experiments, WeSNet-HF achieves performance close to DetNet with only 20% to 30% of the layer weights. Therefore, such weight scaling leads to a significant decrease in the model size by 79.82% and 68.73% respectively. Similarly, we observe a reduction of 51.43% computational cost and 49.78% decrease in model size with 50% profile weight coefficients.

VII. CONCLUSION

In this work, we present an efficient and scalable deep neural network based MIMO detector, where complexity can be adjusted at inference with graceful degradation in the detection accuracy. We introduce a weight scaling framework using monotonically non-increasing profile functions to dynamically prioritize a fraction of the layer weights during training. In order to allow for the neural network architecture to self-adjust to the detection complexity, we also allow for the profile functions themselves to be trainable parameters in the proposed architecture. From our simulation results, we find that the model with trainable coefficients outperforms the one with non-trainable coefficients, but at the cost of complexity. In addition, our proposal shows that adding weight scaling

via monotonic profile functions maintains the detection accuracy when dropping layer weights. This is achieved in part by introducing an L_1 -based regularization function on the layer weights and their profile function coefficients so that the model size can be scaled down by nearly 40% during the feed-forward inference with marginal impact in the detection accuracy.

APPENDIX A

FEED-FORWARD COMPUTATIONAL COST OF AN MLP

Consider an input, $\mathbf{X} \in \mathbb{R}^{(j,k)}$ and weight $\mathbf{W} \in \mathbb{R}^{(i,j)}$, the linear combination of \mathbf{X} and \mathbf{W} is given by

$$\mathbf{Z}_{ik} = \mathbf{W}_{ij}\mathbf{X}_{j,k} + \mathbf{b}_i \quad (32)$$

Applying non linear activation to equation (32), gives:

$$\mathbf{a}_{ik} = g(\mathbf{Z}_{ik}) \quad (33)$$

where $g(\cdot)$ is the nonlinear activation function. The matrix multiplication has an asymptotic computational complexity $\mathcal{O}(n^3)$ and the activation function has $\mathcal{O}(n)$ complexity.

A. Feed-Forward Inference

For $N^{[L]}$ number of neurons including bias unit in the r -th layer, the total complexity can be calculated as a sum of the total number of matrix multiplication and the applied activation over the entire layers as

$$N_{\text{matmul}} = \sum_{r=2}^L (N^{[r]}N^{[r-1]}N^{[r-2]}) + N^{[1]}N^{[0]} \quad (34)$$

$$N_g = \sum_{r=1}^L (N)^{[r]} \quad (35)$$

$$\begin{aligned} \text{Complexity} &= N_{\text{matmul}} + N_g \\ &= N_L \cdot N^3 \end{aligned} \quad (36)$$

The complexity for r -th layers:

$$\begin{aligned} N_{\text{matmul}} &= \mathcal{O}(n \cdot n^3) \\ &= \mathcal{O}(n^4) \end{aligned} \quad (37)$$

Similarly, the complexity N_g for the activation function with L layers is:

$$\begin{aligned} N_g &= N_L \cdot N \\ &= \mathcal{O}(n^2) \end{aligned} \quad (38)$$

Therefore, the total complexity of the forward propagation is

$$\begin{aligned} \text{Total complexity} &= \mathcal{O}(n^4 + n^2) \\ &\approx \mathcal{O}(n^4) \end{aligned} \quad (39)$$

REFERENCES

- [1] C.-X. Wang, F. Haider, X. Gao, X.-H. You, Y. Yang, D. Yuan, H. M. Aggoune, H. Haas, S. Fletcher, and E. Hepsaydir, "Cellular architecture and key technologies for 5g wireless communication networks," *IEEE communications magazine*, vol. 52, no. 2, pp. 122–130, 2014.
- [2] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5g wireless networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1617–1655, 2016.
- [3] L. Lu, G. Y. Li, A. L. Swindlehurst, A. Ashikhmin, and R. Zhang, "An overview of massive mimo: Benefits and challenges," *IEEE journal of selected topics in signal processing*, vol. 8, no. 5, pp. 742–758, 2014.
- [4] H. Yao and G. W. Wornell, "Lattice-reduction-aided detectors for mimo communication systems," in *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE*, vol. 1. IEEE, 2002, pp. 424–428.
- [5] C. Windpassinger, L. Lampe, R. F. Fischer, and T. Hehn, "A performance study of mimo detectors," *IEEE Transactions on Wireless Communications*, vol. 5, no. 8, pp. 2004–2008, 2006.
- [6] T. Kailath, H. Vikalo, and B. Hassibi, "Mimo receive algorithms," *Space-Time Wireless Systems: From Array Processing to MIMO Communications*, vol. 3, p. 2, 2005.
- [7] W.-K. Ma, P.-C. Ching, and Z. Ding, "Semidefinite relaxation based multiuser detection for m-ary psk multiuser systems," *IEEE Transactions on Signal Processing*, vol. 52, no. 10, pp. 2862–2872, 2004.
- [8] A. Wiesel, Y. C. Eldar, and S. Shamai, "Semidefinite relaxation for detection of 16-qam signaling in mimo channels," *IEEE Signal Processing Letters*, vol. 12, no. 9, pp. 653–656, 2005.
- [9] N. D. Sidiropoulos and Z.-Q. Luo, "A semidefinite relaxation approach to mimo detection for high-order qam constellations," *IEEE signal processing letters*, vol. 13, no. 9, pp. 525–528, 2006.
- [10] C.-Y. Hung and W.-H. Chung, "An improved mmse-based mimo detection using low-complexity constellation search," in *2010 IEEE Globecom Workshops*. IEEE, 2010, pp. 746–750.
- [11] D. Wubben, R. Bohnke, V. Kuhn, and K.-D. Kammeyer, "Near-maximum-likelihood detection of mimo systems using mmse-based lattice-reduction," in *2004 IEEE International Conference on Communications (IEEE Cat. No. 04CH37577)*, vol. 2. IEEE, 2004, pp. 798–802.
- [12] R. T. Kobayashi, F. Ciriaco, and T. Abrão, "Performance and complexity analysis of sub-optimum mimo detectors under correlated channel," in *2014 International Telecommunications Symposium (ITS)*. IEEE, 2014, pp. 1–5.
- [13] C. Zhang, Y. Jing, Y. Huang, and L. Yang, "Performance analysis for massive mimo downlink with low complexity approximate zero-forcing precoding," *IEEE Transactions on Communications*, vol. 66, no. 9, pp. 3848–3864, 2018.
- [14] Y. Bengio *et al.*, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.
- [15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [16] T. Wang, C.-K. Wen, H. Wang, F. Gao, T. Jiang, and S. Jin, "Deep learning for wireless physical layer: Opportunities and challenges," *China Communications*, vol. 14, no. 11, pp. 92–111, 2017.
- [17] O. Simeone, "A very brief introduction to machine learning with applications to communication systems," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 4, pp. 648–664, 2018.
- [18] Z. Qin, H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep learning in physical layer communications," *arXiv preprint arXiv:1807.11713*, 2018.
- [19] T. Gruber, S. Cammerer, J. Hoydis, and S. ten Brink, "On deep learning-based channel decoding," in *2017 51st Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2017, pp. 1–6.
- [20] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2016, pp. 341–346.
- [21] T. J. O'Shea, K. Karra, and T. C. Clancy, "Learning to communicate: Channel auto-encoders, domain specific regularizers, and attention," in *2016 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. IEEE, 2016, pp. 223–228.
- [22] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017.
- [23] S. Dörner, S. Cammerer, J. Hoydis, and S. ten Brink, "Deep learning based communication over the air," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 132–143, 2018.
- [24] T. J. O'Shea, T. Erpek, and T. C. Clancy, "Deep learning based mimo communications," *CoRR*, vol. abs/1707.07980, 2017.
- [25] W. Xu, Z. Zhong, Y. Be'ery, X. You, and C. Zhang, "Joint neural network equalizer and decoder," in *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, 2018, pp. 1–5.
- [26] V. Corlay, J. J. Boutros, P. Ciblat, and L. Brunel, "Multilevel mimo detection with deep learning," *arXiv preprint arXiv:1812.01571*, 2018.
- [27] N. Samuel, T. Diskin, and A. Wiesel, "Deep mimo detection," in *Signal Processing Advances in Wireless Communications (SPAWC), 2017 IEEE 18th International Workshop on*. IEEE, 2017, pp. 1–5.
- [28] N. Samuel, A. Wiesel, and T. Diskin, "Learning to detect," *IEEE Transactions on Signal Processing*, 2019.
- [29] X. Tan, W. Xu, Y. Be'ery, Z. Zhang, X. You, and C. Zhang, "Improving massive mimo belief propagation detector with deep neural network," *CoRR*, vol. abs/1804.01002, 2018.
- [30] X. Liu and Y. Li, "Deep mimo detection based on belief propagation," in *2018 IEEE Information Theory Workshop (ITW)*. IEEE, 2018, pp. 1–5.
- [31] M. Imanishi, S. Takabe, and T. Wadayama, "Deep learning-aided iterative detector for massive overloaded mimo channels," *CoRR*, vol. abs/1806.10827, 2018.
- [32] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefanet: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.
- [33] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.
- [34] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1389–1397.
- [35] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *International conference on machine learning*, 2013, pp. 1058–1066.
- [36] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5918–5926.
- [37] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnet: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.
- [38] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in neural information processing systems*, 2016, pp. 4107–4115.
- [39] B. McDanel, S. Teerapittayanon, and H. Kung, "Incomplete dot products for dynamic computation scaling in neural network inference," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2017, pp. 186–193.
- [40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [41] Z. Luo, W. Ma, A. M. So, Y. Ye, and S. Zhang, "Semidefinite relaxation of quadratic optimization problems," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 20–34, May 2010.
- [42] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1, no. 10.
- [43] K. Van Den Doel, U. Ascher, and E. Haber, "The lost honour of l2-based regularization," *Large Scale Inverse Problems, Radon Ser. Comput. Appl. Math*, vol. 13, pp. 181–203, 2012.
- [44] A. S. Ross and F. Doshi-Velez, "Improving the adversarial robustness and interpretability of deep neural networks by regular-

- izing their input gradients,” in *Thirty-second AAAI conference on artificial intelligence*, 2018.
- [45] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.
- [46] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2012, vol. 3.
- [47] W.-K. Ma, C.-C. Su, J. Jaldén, and C.-Y. Chi, “Some results on 16-qam mimo detection using semidefinite relaxation,” in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2008, pp. 2673–2676.
- [48] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, “Tensorflow: A system for large-scale machine learning,” in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.