Accelerated Learning-Based MIMO Detection through Weighted Neural Network Design

Abdullahi Mohammad*, Christos Masouros* and Yiannis Andreopoulos*
*Department of Electronic and Electrical Engineering, University College London, WC1E 7JE UK
e-mail: (abdullahi.mohammad.16; c.masouros; i.andreopoulos)@ucl.ac.uk

***Abstract*—In this paper, we introduce a framework for a systematic acceleration of deep neural network (DNN) design for MIMO detection. A monotonically non-increasing function is used to scale the values of the layer weights such that only a certain fraction of the inputs is used for feedforward computation. This enables a dynamic weight scaling *across* and *within* the network layers, and it is termed as weight-scaling neural network-based MIMO detector (WeSNet). To increase the robustness against the changes in the activation patterns and additional enhancement in the detection accuracy for the same inference complexity, we introduce trainable weight-scaling functions. Experimental results show the superiority of our proposed method over the benchmark model (DetNet) and classical approaches based on semi-definite relaxation in terms of detection accuracy and computational efficiency.**

***Index Terms*—MIMO detection, multilayer perceptron, Deep Neural Networks, DetNet, WesNet, profile weight coefficients.**

I. Introduction

RECENT research work on artificial intelligence (AI) for wireless communications has shown substantial progress in applying deep learning techniques and other machine learning algorithms on the physical layer for various signal processing [1]–[4]. An ingenious auto-encoder design proposed by O'Shea and Hoydis [3] is one of the successful applications of deep learning on physical layer. It consists of module functions for constructing signal constellations based on the communication rate and error coding correction for modulation classification, and signal detection. It provides superior performance over the traditional modulation schemes such as BPSK, QAM, PSK, and Hamming code decoder. More recent works have involved MIMO detection through deep learning. One of the earliest attempts is the work of O'Shea *et al.* [5], who implemented unsupervised learning using an auto-encoder as a continuation of end-to-end learning of previous attempts [3]. The combined effects of learning and the detection capability of the conventional optimal detector are presented in the work of Samuel *et al.* [6], 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 lower inference complexity.

Generally, a deeper neural network performs better than the shallow network architectures, at the expense of a significant increase in complexity and training time [7], [8]. Therefore, designing DNN architectures with scalable complexity that can enable learning and inference on a range of low powered devices becomes necessary. While considerable number of work on accelerating DNN training and inference have been put forward in computer vision [9]–[11], to the best of our knowledge, no such systematic DNN architecture has so far been designed for physical layer communications. In this work, we attempt to address this by proposing a complexity-scalable DNN architecture for efficient MIMO detection. We introduce the concept of monotonic non-increasing profile function that scales each layer of the NN to allow the network to dynamically learn the best attenuation strategy for its weights during training. We further introduce a learnable accuracy-complexity design, where the weight profile functions themselves are made trainable in order to impede vanishing gradients caused by the variations in the values of activations. This enhances the detection accuracy of the WeSNet at the expense of additional memory due to the increase in the model parameters. This paper is part of the recent work we propose on *scalable DNN design for MIMO detection* [12].

The remainder of the paper is structured as follows: We present the system model and the review of the traditional MIMO detectors in Section II. The details of the proposed approach are presented in Section III. Experiments and results are discussed in Section IV. The paper is finally summarized and concluded in Section V.

II. System Model and Benchmarks

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

$$\bar{\mathbf{y}} = \bar{\mathbf{H}}\bar{\boldsymbol{s}} + \bar{\boldsymbol{n}} \tag{1}$$

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

and ease of implementation, the channel model is redefined over the real domain as

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

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

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

$$\mathbf{y} = \mathbf{H}\boldsymbol{s} + \boldsymbol{n} \tag{2}$$

Below, we briefly summarize some common classical and learning-based MIMO detectors that form our performance benchmarks.

A. Maximum Likelihood-detector (ML-detector)

The ML detector is known as the optimal detector [13]. 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 the number of transmit antennas). Therefore, the complexity of this detector grows exponentially with the number of transmit antennas and the modulation order, and cannot be practically realizable in real systems. ML-detector calculates the Euclidean distance between the received and the transmitted symbol vectors for a given channel and spot the one with minimum value.

$$\hat{\boldsymbol{s}} = \underset{\boldsymbol{s} \in \boldsymbol{S}}{\arg\min} \|\mathbf{y} - \mathbf{H}\boldsymbol{s}\|^2 \tag{3}$$

B. Deep MIMO Detector (DetNet)

Most pertinent to our work is the detector that uses a DNN architecture [6], [14]. The unique property of this detector is its ability to scale for higher dimension signals [14]. Learning that leads to good detection is achieved by finding a set of parameters, Υ that minimize the difference between the transmitted symbols and their estimates. By unfolding (3) using *'Stochastic Gradient Descent'* (SGD) optimization over r-th layers (i.e iterations), the architecture of the DetNet is designed as

$$\hat{\boldsymbol{s}}_{r+1} = \hat{\boldsymbol{s}}_r - 2\eta_r \mathbf{H}^T\mathbf{y} + 2\eta_r \mathbf{H}^T\mathbf{H}\hat{\boldsymbol{s}}_r \tag{4}$$

With $\mathbf{H}^T\mathbf{y}$, $\mathbf{H}^T\mathbf{H}$ and $\boldsymbol{s}$ as inputs and the application of some non-linearity, (4) is converted to a multi-layer perceptron (fully connected NN) of r-th-layers $\forall\ r = 1, \ldots, L$. Each layer consists of a three sub-layers (1r, 2r and 3r) defined by the following equations

$$\mathbf{u}_r = \Omega(\mathbf{W}_{1r}\mathbf{x}_r + \boldsymbol{b}_{1r}) \tag{5}$$

where:

$$\mathbf{x}_r = \Xi(\mathbf{H}^T\mathbf{y},\ \mathbf{H}^T\mathbf{H}\boldsymbol{s}_r,\ \boldsymbol{s}_r,\ \mathbf{a}_r) \tag{6}$$

$$\hat{\boldsymbol{s}}_{r+1} = \Theta(\mathbf{W}_{2r}\mathbf{u}_r + \boldsymbol{b}_{2r}) \tag{7}$$

$$\hat{\mathbf{a}}_{r+1} = \mathbf{W}_{3r}\mathbf{u}_r + \boldsymbol{b}_{3r} \tag{8}$$

$\Xi(\cdot)$ 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, $\Omega(\cdot)$ and $\Theta(\cdot)$ 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

$$\boldsymbol{\Upsilon} = \{\mathbf{W}_{1r},\ \mathbf{W}_{2r}\ ,\ \mathbf{W}_{3r},\ \boldsymbol{b}_{1r},\ \boldsymbol{b}_{2r},\ \boldsymbol{b}_{3r}\}_{r=1}^{L} \tag{9}$$

III. Proposed Weight-Scaling Neural-Network based MIMO Detector (WeSNet)

A. Weight Scaling Vector Coefficient (WSVC)

A WSVC is calculated 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, $\mathbf{x} = [x_1, x_2,\ \ldots, x_N]^T$ and $\mathbf{y} = [y_1, y_2,\ \ldots, y_N]^T$, if $\boldsymbol{\Psi}$ is the vector of the profile coefficients, WSVC is the truncated version of the form

$$\sum_{k=1}^{N} \Psi_k x_k^T y_k = \Psi_1 x_1^T y_1 + \Psi_2 x_2^T y_2, ... + \Psi_N x_N^T y_N \tag{10}$$

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

$$z_q = \sum_{p=1}^{N} W_{qp} x_p + b_q \tag{11}$$

where p and q are the input and output dimensions respectively; x_p is the p-th input components, W_{qp} is the channel or layer weight corresponding to the qth output and b_q is the output bias. The corresponding WSVC is derived by

$$z_q = \sum_{p=1}^{N} \Psi_p W_{qp} x_p + b_q \tag{12}$$

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 shaded region on the bottom-right side of the figure. The example shows that, via the WSVC, we can compute and use only one-fourth of the channel/layer weights out of the N layer dimension, as the remaining weights are attenuated and can be dropped.

B. Weight Coefficient Profile Formulation

In this section, we begin by presenting two non-increasing monotonic profile functions (Linear and Half-Exponential functions) for the weight coefficients [11].

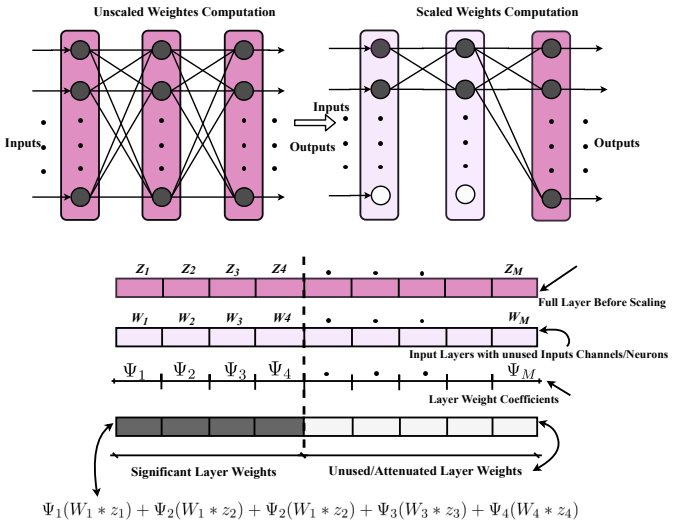

Fig. 1: WSVC in a single layer of an MLP allowing for attenuated layer weights to (optionally) be dropped [11].

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

$$\boldsymbol{\Psi}_k = 1 - \frac{k}{N} \; ; \; \forall \; k = 1, \; 2, \ldots, N \tag{13}$$

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 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.

$$\boldsymbol{\Psi}_k = \begin{cases} 1 & if \; k \; \leq \frac{N}{2} \; \forall \; k = 1, \; 2, \ldots, N \\ exp\left(\frac{N}{2} - k - 1\right) & otherwise \end{cases} \tag{14}$$

C. Architectural Framework of the WeSNet-Detector

WeSNet is a nonlinear detector designed by unfolding the ML metric using a recursive formulation of the projected gradient descent optimization. Our proposed detector applies the profile functions on the existing DetNet. Such a modification reduces the computational complexity for training the detector. We apply profile coefficients function to (5) and (8) to obtain the following non-linear WSVCs over p-th and q-th inputs of the r-th layer of the first and third sub-layers respectively.

$$u_{r(q)} = \Omega \left\{ \sum_{p=1}^{N} \Psi_{1r(p)} W_{1r(qp)} x_{r(p)} + b_{1r(q)} \right\} \tag{15}$$

$$\hat{\mathrm{a}}_{(r+1)h} = \sum_{q=1}^{M} \Psi_{3r(q)} W_{3r(hq)} u_{r(q)} + b_{3r(h)} \tag{16}$$

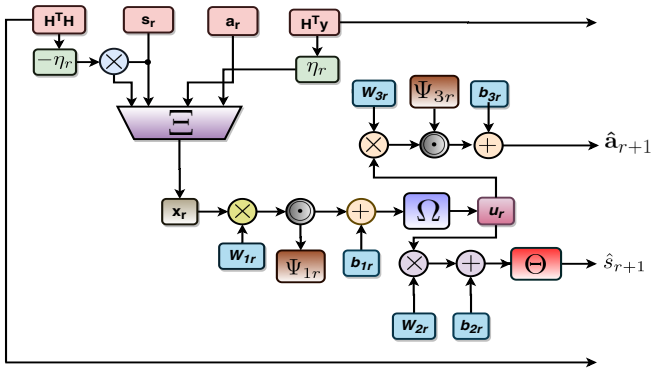

Fig. 2: Single r-th layer WeSNet-detector.

where q and h are the outputs of the first and third sub-layers respectively, while N and M are their corresponding sizes.

The WeSNet is designed with 60 layers and each layer has three sub-layers, the input layer, the auxiliary and the detection layer. The layer weights of the last sub-layer (detection layer) described by (7) are 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 (7), (15) and (16) is shown in Fig. 2. It is important to note that ML-detector does not require the knowledge of the noise variance for error estimation. Therefore, the loss function of WeSNet is derived as the weighted sum of the detector's errors normalized with the loss function of the typical linear inverse detector [6] as

$$\mathcal{L}(s; \hat{s}(H, y : \Upsilon)) = \sum_{r=1}^{L} \log(r) \frac{\left\| \boldsymbol{s} - \hat{\boldsymbol{s}}_r \right\|^2}{\left\| \boldsymbol{s} - \tilde{\boldsymbol{s}} \right\|^2} \tag{17}$$

D. 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 training process. This allows for significantly wider exploration of appropriate scaling functions than the predetermined profile functions presented earlier, albeit at the expense of computational complexity during training. To achieve this, (9) is modified to include profile weight functions as learned parameters.

$$\tilde{\boldsymbol{\Upsilon}} = \{\mathbf{W}_{1r}, \; \mathbf{W}_{2r} \; , \; \mathbf{W}_{3r}, \; \boldsymbol{b}_{1r}, \; \boldsymbol{b}_{2r}, \; \boldsymbol{b}_{3r}, \; \boldsymbol{\Psi}_r\}_{r=1}^{L} \tag{18}$$

It is important to note that the monotononicity during training and gradient update is maintained by the shape of the functions of (13) and (14). Finally, we conclude this section by describing the pseudocode for our proposed framework as shown in **Algorithms 1-3**.

IV. Simulation and Numerical Results

In this section, we present the experimental setup and the performance of the WeSNet under different profile functions.

Algorithm 1 Weight Coefficient Steps

Input: Input layer size (N number of input neurons)
Output: Vector of weight coefficient

for $k = 1$ to N **do**
  if Function is $'Linear'$ **then**
    Compute: *Linear Profile Function* (13)
  else if Function is $'Half - Exponential'$ **then**
    Compute: *Half-exponential Profile Function* (14)
  end if
end for
Initialisation :
steps $= [1,\ 1 - \frac{1}{n},\ 1 - \frac{2}{n}, \cdots, \frac{1}{n}]$ {weight masking vector}
where n is an integer, which can be chosen from 1-10
Weight Coefficient Steps $= [0, \cdots, 0]$
$m = int(\frac{N}{n})$
for $i \in$ **steps do**
  for $j = 1$ to m **do**
    Save i **to Weight coefficient Steps** for every j
  end for
end for
return *Weight Coefficient Steps*

Algorithm 2 Layer Weight Profiling

Input: Weight-Coefficients, Coefficient-ratio $= \tilde{\Psi}_{\text{cr}}$, N, in-size, out-size, begin-profile, end-profile
Output: Layer Weight-coefficients
Initialize Weight Profile :
if $\Psi_{\text{cr}} = 0$ **then**
  Weight-Coefficients $\leftarrow Weight - Coefficients$
end if
Weight-Coefficients = Weight-Coefficients $\times\ \tilde{\Psi}_{\text{cr}}$
return Weight-Coefficients
Layer Profile range
  (a) $0 \leq$ begin-profile ≤ 10
  (b) $0 \leq$ end-profile ≤ 10
Generate *Input-steps* and *Output-steps*
Initialize Profile Step :
p-step $= [0, \cdots, 0]$
p-step$\big[$begin-profile : end-profile$\big] = 1$
Let *WCS* = Weight Coefficient Steps {from **Algorithm** 1}
  (a) *Input-steps* = *WCS*(in-size, steps = p-steps)
  (b) *Output-steps* = *WCS*(out-size, steps = p-steps)
Let the Functions in **Algorithm** 1 be **P-Func** and **WC** = Weight-Coefficients
Weight Profile $\leftarrow$ **P-Func**(in-size)
Use **WC**, Weight Profile, (a), (b) in line 7 above and $\tilde{\Psi}_{\text{cr}}$ to obtain Layer Weight-coefficients
$\mathbf{W}_{LWC} \leftarrow$ Layer Weight-coefficients
return $\mathbf{W}_{LWC}$

Algorithm 3 Training via Weight-Scaling

Input: Layer Weight-coefficients($\mathbf{W}_{LWC}$), Layer Weight(W), Input-Data, Output-Grad, Loss-Function ($\mathcal{L}$)
Output: Scaled-Weight Vector ($\mathbf{W}_{WSVC}$)

Forward Propagation :
$\mathbf{x} \leftarrow$ Input-Data
out-size, input-size $\leftarrow$ shape of $\mathbf{W}$ {Weight initialization}
$\mathbf{W}_{WSVC} \leftarrow \mathbf{W}_{WSC} * W$ {$*$ is a Hadamard Product}
$\mathbf{z} \leftarrow \mathbf{x}[\mathbf{W}_{WSVC}]^T + \mathbf{b}$ {where; $\mathbf{b}$ = bias}
return $\mathbf{z}$
Backward Propagation :
out-size, input-size $\leftarrow$ shape of $\mathbf{W}$
$\mathbf{W}_{WSVC} \leftarrow \mathbf{W}_{WSC} * W$
$\frac{\partial \mathcal{L}}{\partial \mathbf{z}} \leftarrow$ Output-Grad
$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} \leftarrow \frac{\partial \mathcal{L}}{\partial \mathbf{z}} \times \mathbf{W}_{WSVC}$
$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{WSVC}} \leftarrow \big(\frac{\partial \mathcal{L}}{\partial \mathbf{z}}\big)^T \times \mathbf{x}$
$\frac{\partial \mathcal{L}}{\partial \mathbf{b}} \leftarrow \text{sum}\big(\frac{\partial \mathcal{L}}{\partial \mathbf{z}}\big)$
Parameter update :
$\mathbf{W}_{WSVC(i)} \leftarrow \mathbf{W}_{WSVC(i-1)} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}_{WSVC}}$
$\mathbf{b}_i \leftarrow \mathbf{b}_{i-1} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{b}}\ \{\forall i \in \{1,\ 2\ \ldots,\ N\}\}$

TABLE I: Simulation settings

Parameters	Values
First Sublayer Dimension	$20N_t = 320$
Second Sublayer Dimension	$N_t = 16$
Third Sublayer Dimension	$20N_t = 320$
Number of Layers	$L = 60$
Fraction of non-zero Layer Weights	$\tilde{\Psi}_{\text{cr}}$
Training Samples	100000
Batch Size	5000
Test Samples	10000
Training SNR range	5dB - 25dB
Test SNR range	6dB - 24dB
Optimizer	SGD with Adam
Learning Rate	0.001
Weight Initializer	Xavier Initializer
Number of Training Iterations	10000
Number of Monte Carlo during inference	400

WeSNet is implemented in *Tensorflow* 1.14.0 [15] and python 3.6.8. 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 a symmetric channel of 16 transmit and 16 receive antennas (16, 16). To ensure a fair comparison, we evaluate the performance of our proposed model with the benchmark model (DetNet) under the same experimental settings with the same simulation parameters as summarized in Table I.

We generate both training and test dataset stochastically drawn from BPSK constellation $s \in \{\pm 1\}^{N_t}$, a random white Gaussian noise from a uniform distribution

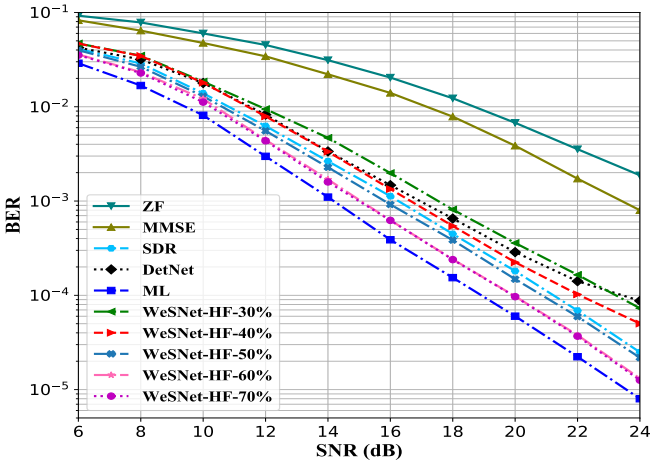

Fig. 3: BER comparison of the proposed DNN MIMO Detector: (WeSNet-HF), DetNet and other classical Detectors.

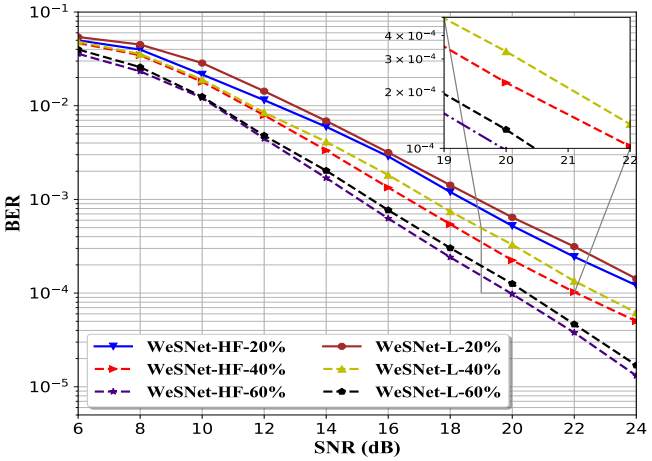

Fig. 4: BER comparison of WeSNet-HF and WeSNet-L.

over a wide range of SNR values $\mathcal{U}(5dB - 25dB)$ and the corresponding received symbols are generated from the standard wireless channel model. The model is trained for 10000 iterations with 5000 batch size. We assume an unknown noise variance (see Section II), and therefore the noise vector is generated from a random uniform distribution over the training SNR values in order to enable the parameters learning over a wide range of SNR conditions.

A. WeSNet Design with Half-Exponential and Linear Profile Functions Performance Evaluation

Fig. 3 depicts the performance of half-exponential WeSNet (WeSNet-HF)F with 40%-70% weight coefficients and other benchmark detectors. It can be seen that WeSNet-HE surpasses both ZF and MMSE by wide margin. In general, WeSNet with only 40% of the layer weights (WeSNet-HF-40%) outplays DetNet. In fact, the accuracy margin is remarkably conspicuous with profile weight coefficients $>$ 50% (WeSNet-HF-50% $-$ WeSNet-HF-70%). At 10^{-4} BER, WeSNet-HF-50% and WeSNet-HF-60% outperform DetNet with 1.34dB and 2.48dB respectively. Similarly, they also yield better accuracy of symbol detection over SDR by 0.45dB and 1.35dB margin respectively.

Fig. 4 shows the performance of WeSNet-HF and linear WeSNet (WeSNet-L) as more weight coefficients are gradually added. It can be seen that both linear and half-exponential profile WeSNet models have a comparable performance at lower SNR fora profile coefficients between 20% $-$ 40% of the layer weights. However, we observe a distinguishable difference at higher SNR as more profile weight coefficients are used. As anticipated, adding more profile coefficients increases WeSNet's accuracy, but performance saturates after 60% of the coefficients. It can also be seen that WeSNet-HF outperforms WeSNet-L with a significant margin at higher SNR. This means

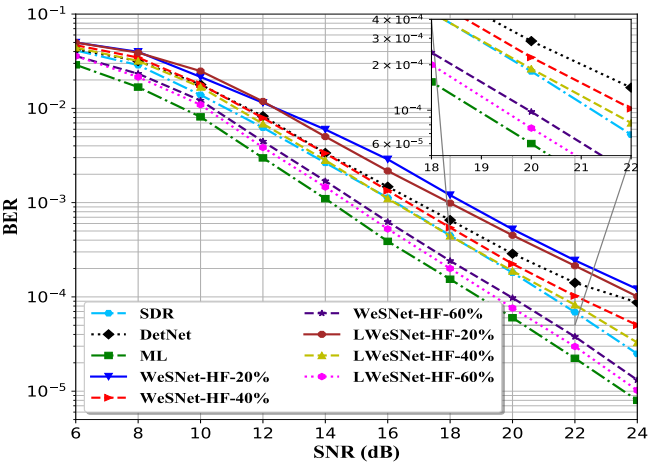

Fig. 5: BER comparison of the L-WeSNet, WeSNet-HEF, DetNet and other classical detectors.

that WeSNet-HF provides a better dynamic scalable computation over a wide range of percentages of the layer's input. Following this reason, we shall use WeSNet-HF as our reference model for the remaining analyses in the subsequent sections.

B. Performance Evaluation of L-WeSNet

In this section, the performance of our proposed approach is examined when the weight profile coefficients are made learnable (L-WeSNet).

Fig. 5 delineates the performance of WeSNet with learnable weight coefficients. It can be seen that there is a significant performance improvement as the percentage of nonzero weight coefficients are used. Our study also shows that L-WeSNet-40% produces better accuracy than DetNet and near SDR at all SNRs. Furthermore, for L-WeSNet-HF-60%, the performance is near-optimal.

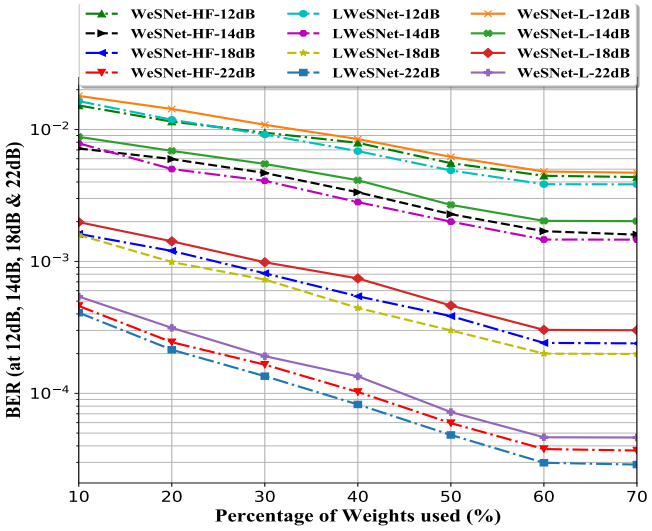

Fig. 6: BER vs Percentage Weight Profile Coefficients for L-WeSNet-HF, WeSNet-HF and WeSNet-L.

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

MIMO Detector	Number of Flops Operation
ML	$\lvert\mathbf{S}\rvert^{N_t}(8N_t^2+8N_t-2)$
SDR	$\left(13N_t^3+25N_t^2+17N_t+4\right)N_{\text{iterations}}$ [13], [16]
WeSNet	$\left[\left(\tilde{\Psi}_{\text{cr}}N_t(128N_t+5)+9N_t\right)\right]L$, $L=$ number of layers
DetNet	$[(N_t(128N_t-2))]\,L$

Fig. 6 shows the variation of BER as a function of layer weight coefficients for WeSNet-HF, WeSNet-L, and the L-WeSNet-HF. We observe that at 12dB, 14dB, 18dB, and 22dB with 50% weight coefficients, L-WeSNet-HF has been able to reduce the BER by 11.9%, 12.2%, 22.1% and 18.2% respectively as compared to non-learnable WeSNet-HF model. It can also be seen that L-WeSNet-HF has comparable accuracy with WeSNet-HF at lower SNR with $10\%-20\%$ of the weights. However, we observe a significant performance improvement over non-learnable WeSNet(s)(WeSNet-HF and WeSNet-L) at higher SNR as more weights coefficients are added.

C. Complexity Evaluation

Table II shows the computational cost of the detectors as measured by the number of floating-point operations (FLOPS). It can be seen that the complexity of ML scales exponentially with the number of transmit antennas as expected while SDR and DetNet are of $\mathcal{O}(n^4)$ and $\mathcal{O}(n^3)$ respectively. Compared to the benchmark detectors, however, the introduction of the weight profile coefficient scaling design in WeSNet has led to the significant reduction of its inference (online) detection computational cost even much lower than that of the DetNet.

V. Conclusion

In this work, we present an efficient and scalable deep neural network-based MIMO detector design, where complexity can be adjusted at inference while achieving a competitive 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. The results show that our proposed approach achieves state-of-the-art performance comparable to those optimal classical detectors at much lower complexity. Lastly, our experiments show that weight profile functions act as regularizers, i.e., in addition to exhibiting dynamic capabilities to scale over a computation range, they also help the network to generalize well by preventing overfitting when the model size grows.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] T. J. O'Shea, T. Erpek, and T. C. Clancy, "Deep learning based mimo communications," *CoRR*, vol. abs/1707.07980, 2017.

[6] 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.

[7] H. N. Mhaskar and T. Poggio, "Deep vs. shallow networks: An approximation theory perspective," *Analysis and Applications*, vol. 14, no. 06, pp. 829–848, 2016.

[8] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao, "Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review," *International Journal of Automation and Computing*, vol. 14, no. 5, pp. 503–519, 2017.

[9] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

[10] 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.

[11] 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.

[12] A. Mohammad, C. Masouros, and Y. Andreopoulos, "Complexity-scalable neural network based mimo detection with learnable weight scaling," *arXiv preprint arXiv:1909.06943*, 2019.

[13] 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.

[14] N. Samuel, A. Wiesel, and T. Diskin, "Learning to detect," *IEEE Transactions on Signal Processing*, 2019.

[15] 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.

[16] 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.