Concours externe Inria 2022

Arrêté du 8 juin 2022

2022-DT-LYS-1 - Ingénieur.e développement logiciel spécialiste en communication et traitement du signal

2022-DT-LYS-2 - Ingénieur.e développement logiciel spécialiste en calcul scientifique pour l'apprentissage et le traitement du signal

Accès au corps des Ingénieurs de Recherche

Centre Inria de Lyon

Note sur 20 – Coefficient 3 – Durée 3h00

****

La notation prendra en compte la qualité des réponses, mais aussi la rédaction, la présentation, le style et l'orthographe.

Veuillez respecter l'anonymat dans les réponses.

Ne pas omettre de noter votre numéro d'ordre sur les feuilles intercalaires.

Aucun document n'est autorisé pendant cette épreuve écrite, autre que ceux fournis au démarrage de l'épreuve.

Préambule

L'épreuve écrite proposée ci-après aborde plusieurs facettes du métier d'ingénieur de recherche chez Inria : l'organisation du développement, l'analyse de documents, les capacités rédactionnelles ainsi que la maîtrise d'outils et méthodes de développement dans un contexte scientifique.

N'oubliez pas que c'est une épreuve de concours, et non un examen : il n'est pas requis que vous répondiez parfaitement à toutes les questions pour pouvoir espérer voir votre candidature retenue. La première partie en particulier teste des domaines très variés et il est donc attendu que la plupart des candidats ne répondent qu'à certaines questions.

La note de l'épreuve écrite et la note attribuée à l'issue de l'épreuve orale ont le même poids (coefficient 2 pour le dossier, 3 pour cet écrit et 3 pour l'oral) dans la note finale attribuée aux candidats.

Consignes pour la langue de rédaction :

Les réponses seront écrites en français sauf quand cela sera explicitement indiqué autrement.

Première partie (6 points)

Cette première partie compte pour 6 points, vous pouvez faire des réponses courtes en quelques lignes pour chacun des points.

Q 1.1 : Compilation

Donnez deux raisons pour lesquelles un compilateur est généralement décomposé en passes (ou phases) et indiquez un ensemble typique de passes et les structures de données passées entre elles.

Q 1.2 : Reproductibilité

Vous souhaitez reproduire des résultats publiés dans une étude récente. Quels problèmes pourriez-vous rencontrer ? Comment procéderiez-vous pour assurer un certain niveau de reproductibilité des études basées sur vos travaux/développements ?

Q 1.3 : Debug

Un logiciel embarqué que vous développez en langage C sur un processeur ARM sous Linux s'exécute correctement pendant quelques minutes puis s'arrête de façon inattendue. Quels outils proposez-vous pour trouver la raison de cet arrêt et la corriger ?

Q 1.4 : Analyse de performance

Comment identifieriez-vous les problèmes de performance d'un code Python ? Avec quels outils peut-on accélérer un code Python, par exemple en Machine Learning ?

Deuxième partie (7 points)

Exercice d’analyse de projet, proposition d’architecture logicielle et matérielle.

Vous devez prendre en main un projet de développement de taille importante dans une équipe de recherche. L'équipe travaille sur des briques logicielles modulaires de calcul scientifique et de traitement du signal avec les principaux objectifs suivants :

- Mettre à disposition un ensemble de briques logicielles robustes et performantes pouvant être utilisée dans des pipelines de traitement ;
- Construire une fondation modulaire que d'autres personnes, de l'équipe ou extérieures, peuvent étendre et développer ;
- Apprendre comment les méthodes utilisées se comportent et s'articulent entre elles, afin de pouvoir concevoir et construire de meilleurs outils.

Sur sa durée de vie, le projet compte plusieurs contributeurs permanents actifs dans l'équipe, des contributeurs ponctuels en thèse ou en postdoc et de nombreux utilisateurs externes, pouvant également proposer des améliorations. Le mode de fonctionnement est mené par apports successifs (travaux de thèse, de postdoc sous forme de preuves de concepts ou bien de réalisations autonomes) et parfois réalisés dans des langages et modèles qui peuvent être différents. Un des enjeux pour l'équipe est de consolider et d'intégrer ces apports dans une architecture commune pour constituer un socle de développement durable.

En effet, bien que le but scientifique de l'équipe de recherche soit de produire de nouvelles méthodes et algorithmes, les membres considèrent le développement comme une condition essentielle à la réussite de leurs projets, que ce soit pour servir de base à de nouvelles recherches qui s'appuient sur les idées précédentes, mais aussi comme condition nécessaire pour organiser une recherche reproductible. De plus il est important que les développements soient de niveau industriel pour assurer la crédibilité des travaux et mener à des utilisations concrètes en dehors du cercle restreint des chercheurs.

Il y a souvent une tension entre ces objectifs et la prise en main du projet doit refléter l'agilité de navigation nécessaire entre ces objectifs.

Vous allez prendre le rôle d'architecte logiciel dans ce projet. Cela implique de prendre en main les briques logicielles et guider le développement et l'intégration de celles-ci. Ce travail nécessite de réfléchir sur l'architecture logicielle utilisée pouvant s'adapter au mieux aux différentes contraintes.

Pour l'exercice, vous pouvez considérer que les pipelines logiciels visés sont programmés en utilisant un langage de type Python avec des modules de calcul pouvant être programmés dans différents langages compilés (C, C++, OCaml, etc)

Q 2.1 : Quels sont les points d'attention dans l'architecture globale des modules que vous proposeriez de mettre en avant pour maintenir et faire évoluer ce projet ?

Vous pourrez notamment indiquer sur un ou plusieurs schémas (exemples d'instances de pipelines de traitement, schéma d'organisation de codes sources et outils) les points sur lesquels il vous semble nécessaire de proposer des interfaces (API), en indiquant leurs rôles, les langages dans lesquels les éléments peuvent être réalisés, manuellement ou par génération de code, leur fonction, etc.

Q 2.2 : Proposez une mise en place de règles de bonne pratique et une méthodologie pour l'intégration d'une nouvelle brique logicielle dans le projet, que ce soit pour un module dont l'évolution est considérée comme figée (un algorithme de référence provenant de la littérature) ou bien d'un module étant le sujet d'une recherche active avec de nombreuses modifications à venir.

Q 2.3 : L'équipe souhaite également disposer, pour certains modules, de plusieurs versions (implémentations) afin de profiter des possibilités de matériels spécialisés (GPU, DSP, FPGA, etc) pour l'amélioration de performances. Comment proposeriez-vous de gérer le déport des calculs vers ces accélérateurs matériels dans votre architecture et quelles parties devrez-vous modifier ? Vous pourrez discuter en particulier les problèmes liés aux échanges de données et leurs impacts.

Troisième partie (7 points)

Exercice d'analyse de publication scientifique

Parmi les 2 sujets proposés, choisissez le sujet que vous souhaitez traiter en fonction du profil du poste sur lequel vous avez déposé votre candidature. Seul le premier sujet apparaissant sur votre copie sera noté.

Toutes les réponses de ces parties (3.1 et 3.2) sont attendues en anglais.

3.1 Profil « Communication et traitement du signal »

Cette partie s'appuie sur l'article « *Real-time Machine Learning for Symbol Detection in MIMO-OFDM Systems* » de Yibin Liang, Lianjun Li, Yang Yi et Lingjia Liu (IEEE INFOCOM 2022). Le texte de l'article est donné en annexe.

Q 3.1.1 : Describe in few sentences the OFDM principle.

Q 3.1.2 : What is a line of sight path ?

Q 3.1.3 : What is the difference between supervised and unsupervised learning ?

Q 3.1.4 : Among these problems, which ones are interesting to solve with machine learning? (with short justification)

1. Finding the optimal time to post on a web page
2. Finding the shortest path between 2 nodes in a graph
3. Predicting the number of bikes to assign to each station in a bike renting society
4. Denoising a radio signal

Q 3.1.5 : Provide a summary (around 5 lines) of the paper, depicting the goal of the work and its outcome.

Q 3.1.6 : On fig.11 of the paper, which are the best (resp. worst) scenarios, and how can you explain this ?

Q 3.1.7 : On fig.11 of the paper, justify the performances difference between LMMSE and ESN approaches.

3.2 Profil « Calcul scientifique pour l'apprentissage et le traitement du signal »

Questions préliminaires :

Q 3.2.1 : Explain what a linear regression problem is in statistics. Explain the bias-variance dilemma and comment on the interest of using regularized models in this context.

Q 3.2.2 : Describe the gradient descent algorithm and comment on the advantages and disadvantages of using it. Explain the difference with the stochastic version of this algorithm; discuss the cases where it is preferable to use one or the other method and how to choose the descent step in both cases.

La partie suivante s'appuie sur l'article « *FAST RANDOMIZED NUMERICAL RANK ESTIMATION* » de Maike Meier et Yuji Nakatsukasa (arXiv:2105.07388). Le texte est donné en annexe. Ce texte a été modifié en masquant des parties pour limiter la durée de l'exercice.

Q 3.2.3 : Provide a short summary of the article, identify limitations and ways to address them and/or prospects for further development of the methods presented. A critical look at the article will be appreciated. Put the article in perspective with a potential software development.

FAST RANDOMIZED NUMERICAL RANK ESTIMATION

MAIKE MEIER AND YUJI NAKATSUKASA*

Abstract. Matrices with low-rank structure are ubiquitous in scientific computing. Choosing an appropriate rank is a key step in many computational algorithms that exploit low-rank structure. However, estimating the rank has been done largely in an ad-hoc fashion in previous studies. In this work we develop a randomized algorithm for estimating the numerical rank of a matrix. The algorithm is based on sketching the matrix with random matrices from both left and right; the key fact is that with high probability, the sketches preserve the orders of magnitude of the leading singular values. The rank can hence be taken to be the number of singular values of the sketch that are larger than the prescribed threshold. For an $m \times n$ ($m \geq n$) matrix of numerical rank r, the algorithm runs with complexity $O(mn \log n + r^3)$, or less for structured matrices. The steps in the algorithm are required as a part of many low-rank algorithms, so the additional work required to estimate the rank can be even smaller in practice. Numerical experiments illustrate the speed and robustness of our rank estimator.

Key words. Rank estimation, numerical rank, randomized algorithm, Marčenko-Pastur rule

AMS subject classifications. Primary, 65F55; Secondary, 65F99, 68W20, 60B20

1. Introduction.

Low-rank matrices are ubiquitous in scientific computing and data science. Sometimes a matrix of interest can be shown to be of numerically low rank [4, 39], for example by showing that the singular values decay rapidly. More often, matrices that arise in applications may have a hidden low-rank structure, such as low-rank off-diagonal blocks [26, 28]. As is well known, low-rank approximation is also the basis for principal component analysis.

Numerous studies and computational algorithms exploit the (approximately) low-rank structure of these types of matrices to devise efficient algorithms. Such algorithms usually require finding a low-rank approximation of a given matrix. For large-scale matrices that cannot be treated by classical algorithms, randomized algorithms have become a powerful and reliable tool for efficiently computing a near-best low-rank approximation. Landmark references on randomized SVD are [16, 42].

A key step in low-rank algorithms, including randomized SVD, is the determination of the numerical rank. For instance, a variety of low-rank factorization techniques require the target rank of the factorization as input information. Once a low-rank approximation $\hat{A}_r$ of the specified target rank has been computed, a posteriori estimation of the error $\|A - \hat{A}_r\|$ via a small number of matrix-vector multiplications [28] is a reliable means of checking if the input rank r was sufficient. If r was too low, one would need to sample the matrix with more vectors. This clearly requires more computational work, and can be a major difficulty in the streaming model, wherein revisiting the matrix is impossible [37]. Conversely, if the input rank r was too high, the computational cost of computing $\hat{A}_r$ would be higher than it could be. Having a fast and reliable rank estimator is therefore highly desirable.

In this work we propose and analyze a fast randomized algorithm for estimating the numerical rank of a matrix $A \in \mathbb{F}^{m\times n}$. The algorithm is based on sketching AX and then forming YAX, where Y, X are random (subspace embedding) matrices. The key idea is that with high probability, the singular values of AX are good estimates of the (leading) singular values of A. Therefore the decay of $\sigma_i(A)$ can be reliably estimated by the decay of $\sigma_i(AX)$. To estimate the $\sigma_i(AX)$ we once again sketch

*Mathematical Institute, University of Oxford, Oxford, OX2 6GG, UK. (Maike.Meier@maths.ox.ac.uk, nakatsukasa@maths.ox.ac.uk)

AX to obtain YAX; it is only $\sigma_i(YAX)$ (or their estimates) that we actually need to compute.

To qualitatively understand why the singular values are roughly preserved, consider the case where A is rank-r_* where $r_* \ll n$, so that the reduced SVD is $A = U_1\Sigma_1 V_1^*$ where $V_1 \in \mathbb{C}^{n\times r_*}$. Let $X \in \mathbb{F}^{n\times r}$ be a Gaussian embedding matrix with iid elements $\mathcal{N}(0, r^{-1})$. Then $AX = U_1\Sigma_1(V_1^*X) = U_1\Sigma_1 X_1$, where $X_1 := V_1^*X$ is an $r_* \times r$ Gaussian matrix, also with $\mathcal{N}(0, r^{-1})$ elements. Now since $\sigma_i(AX) = \sigma_i(\Sigma_1 X_1)$, it follows that $\sigma_{\min}(X_1) \le \frac{\sigma_i(AX)}{\sigma_i(A)} \le \sigma_{\max}(X_1)$. By the Marcenko-Pastur law [24], assuming $r_*/r > 1$, X_1 is well conditioned with high probability, that is, the lower and upper bounds are quite close; thus giving a reliable estimate for $\sigma_i(A)$. More generally when A has higher rank but $\sigma_r(A) \le \epsilon$, the above argument will need some modification to treat the effect of the trailing singular values; fortunately, they only have an $O(\epsilon)$ effect, which does not unduly affect the rank estimator. These claims will be made precise in Section 3.

The task and development has close connections to results in the statistical literature, some of which we mention along the way.

The complexity of our algorithm is $O(mn\log n + r^3)$ for dense $m\times n$ matrices, and can be lower if A has structure that can be exploited for computing the sketches AX and YAX. This is clearly a subcubic complexity (assuming $r \ll m, n$), and it runs significantly faster than computing the full SVD. Moreover, in many cases, computing AX (which is usually the dominant part of our rank estimation algorithm) is a required part of the main algorithm (e.g. randomized SVD); and in some algorithms [10, 32] this is true even of YAX. In such cases, the additional work needed for estimating the rank is therefore significantly smaller, such as $O(r^3)$ or sometimes even $O(r)$.

Notation. Throughout, $\sigma_i(A)$ denotes the ith largest singular value of the matrix A. We use $\|\cdot\|$ to denote the spectral norm $\|A\|_2 = \sigma_1(A)$, and $\|A\|_F$ is the Frobenius norm. $\mathbb{F}$ denotes the field, in our case $\mathbb{F} = \mathbb{R}$ or $\mathbb{F} = \mathbb{C}$. Unless specified otherwise A is $m \times n$ where $m \ge n$. The numerical rank estimate will be denoted by $\hat{r}$.

Throughout the paper we use X and Y for random oblivious subspace embedding matrices, such that with high probability, for any fixed Q with orthonormal columns we have $\sigma_{\max}(Q^TX), \sigma_{\max}(YQ) \le 1+\epsilon$ and $\sigma_{\min}(Q^TX), \sigma_{\min}(YQ) \ge 1-\epsilon$ for some $\epsilon \in (0,1)$. We use r, r_1 and r_2 to refer to the size of embedding matrices, which must be at least the number of columns in Q. The matrix X is required to have more columns than Q, and the same goes for the rows of Y. For brevity we simply call such X and Y embedding matrices.

The analysis will be specified to Gaussian or subsampled randomized trigonometric transform (SRTT) embedding matrices at times. We will use $G \in \mathbb{R}^{n\times r}$, where $n > r$, to denote a matrix where each entry is iid $\mathcal{N}(0,1)$ distributed and call this matrix a (standard) Gaussian matrix. A Gaussian matrix can be scaled to an embedding matrix by defining $X = G/\sqrt{r}$; we call the scaled Gaussian matrices *Gaussian embedding matrices*. SRTT matrices are defined in Section 3.3 and as the scaling is included in the definition, they are naturally embedding matrices. We will use Θ to refer to the set of SRTT matrices.

1.1. Numerical rank and goal of a rank estimator.

So far we have been using the term "the numerical rank" informally. Let us now define the notion. This is a standard definition, see for example [4].

DEFINITION 1.1. *Let $A \in \mathbb{F}^{m\times n}$. The ϵ-rank of A, denoted by $\text{rank}_\epsilon(A)$, is the*

integer[1] *i such that* $\sigma_i(A) > \epsilon \geq \sigma_{i+1}(A)$.

What is the goal of a rank estimator? One natural answer of course is that the estimator should output $\text{rank}_\epsilon(A)$ given A and the (user-defined) threshold ϵ. However, we argue that the situation is more benign: consider, for example, a matrix with k singular values $> 10\epsilon$, five singular values in $(\epsilon, 1.1\epsilon)$, five more in $(0.9\epsilon, \epsilon)$, and the remaining $n - k - 10$ are $< 0.01\epsilon$. What should the estimator output? Is it crucial that the "correct" value $\text{rank}_\epsilon(A) = k + 5$ is identified?

In this paper we take the view that the goal of the rank estimator is to find a good ϵ-rank, not necessary *the* ϵ-rank. In the example above, any number between k and $k + 10$ would be an acceptable output. In most applications that we are aware of (including low-rank approximation, regularized linear systems, etc), there is little to no harm in choosing a rank $\hat{r} \neq \text{rank}_\epsilon$: a slight overestimate $\hat{r} > \text{rank}_\epsilon$ usually results in slightly more computational work, whereas a slight underestimate $\hat{r} < \text{rank}_\epsilon$ is fine if $\sigma_{\hat{r}+1}(A) = O(\epsilon)$, that is, a rank-$\hat{r}$ matrix can approximate A to $O(\epsilon)$-accuracy.

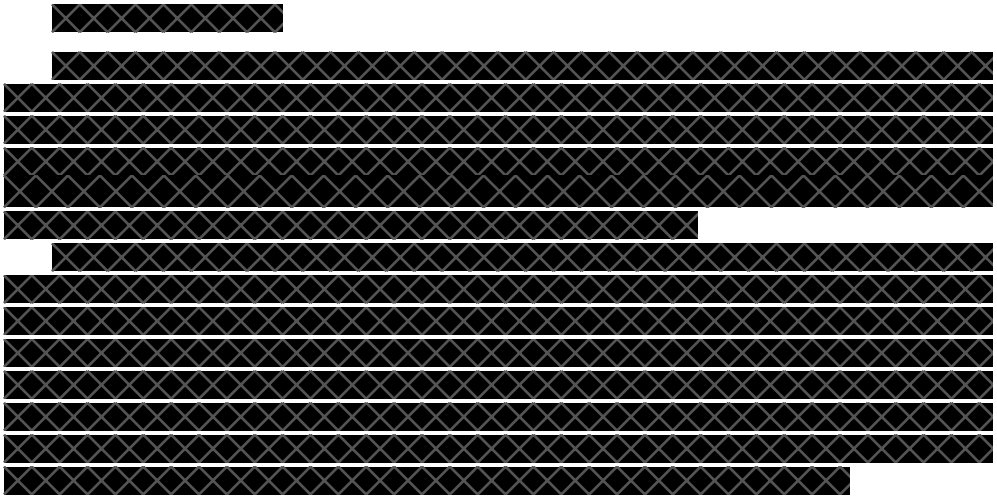
On the other hand, it is clearly *not* acceptable if the rank is unduly underestimated in that $\sigma_{\hat{r}+1}(A) \gg \epsilon$. It is also unacceptable if $\sigma_{\hat{r}}(A) \ll \epsilon$. The goal in this paper is to devise an algorithm that efficiently and reliably finds an $\hat{r}$ such that

- $\sigma_{\hat{r}+1}(A) = O(\epsilon)$ (say, $\sigma_{\hat{r}+1}(A) < 10\epsilon$): $\hat{r}$ is not a severe underestimate, and
- $\sigma_{\hat{r}}(A) = \Omega(\epsilon)$ (say, $\sigma_{\hat{r}}(A) > 0.1\epsilon$): $\hat{r}$ is not a severe overestimate.

Our rank estimate will satisfy these conditions with high probability. In particular, in situations where the numerical rank is clearly defined, i.e., if a clear gap is present in the spectrum $\sigma_r(A) \gg \epsilon \gg \sigma_{r+1}(A)$, the algorithm will reliably find the exact rank $\hat{r} = \text{rank}_\epsilon(A)$.

In addition, in situations where it is unknown whether the matrix A is low-rank approximable, our rank estimator can tell us (roughly) how well A can be approximated with a low-rank matrix.

This paper is organized as follows. Section 2 discusses related studies in the literature. In Section 3 we show that $\sigma_i(AX)$ gives useful information about $\sigma_i(A)$ for leading values of i. Then in Section 4 we show that $\sigma_i(AX)$ can be estimated via $\sigma_i(YAX)$. Section 5 summarizes the overall rank estimation algorithm. Numerical experiments are presented in Section 6.



[1] If no such i exists, we take $\text{rank}_\epsilon(A) = \min(m, n)$; however, in this paper we are never interested in this full-rank case.

3. Sketching roughly preserves singular values: $\sigma_i(AX)/\sigma_i(A) = O(1)$ for leading i. Here we show that if a matrix A has low numerical rank, then sketching preserves its leading singular values sufficiently well to provide an estimate for the numerical rank. Again drawing the connection to sample covariance matrix estimation, it is intuitive that it should be difficult, if not impossible, to obtain information about an n-dimensional distribution using only r samples, when $r \ll n$. Clearly, there are $n - r$ singular values (or signals) we cannot detect, but it is also not clear why the r singular values we estimate would be any good. One way to think about this is to see that $\mathcal{N}(0, \Sigma^2)$ lies close to a low-dimensional subspace, since A is approximately low-rank. This makes it more intuitive why a small sample size could suffice to retrieve a reasonably accurate approximation of Σ.

We make the connection between the low-rank structure of A and our ability to estimate its singular values using AX explicit with the deterministic and probabilistic error bounds on the ratio $\sigma_i(AX)/\sigma_i(A)$. The analysis will focus on Gaussian matrices. As explained before, $G \in \mathbb{R}^{n\times r}$ denotes a Gaussian matrix with iid $\mathcal{N}(0,1)$ entries, and a Gaussian embedding matrix will be of the form $X = G/\sqrt{r}$. We start

THEOREM 3.3. *Let $X \in \mathbb{R}^{n\times r}$ be a Gaussian embedding matrix, i.e., $X = G/\sqrt{r}$, and let $A \in \mathbb{R}^{m\times n}$, where $m \geq n$. Then, for $i = 1, \ldots, r$*

$$1 - \sqrt{\frac{i}{r}} \leq \mathbb{E}\frac{\sigma_i(AX)}{\sigma_i(A)} \leq 1 + \sqrt{\frac{r-i+1}{r}} + \frac{\sigma_{r+1}}{\sigma_i}\left(1 + \sqrt{\frac{n-r}{r}}\right). \tag{3.7}$$

Additionally, for each $i = 1, \ldots, r$ we have, with failure probability at most $3e^{-t^2/2}$,

$$\begin{aligned} 1 - \sqrt{\frac{i}{r}} - \frac{t}{\sqrt{r}} &\leq \frac{\sigma_i(AX)}{\sigma_i(A)} \\ &\leq 1 + \sqrt{\frac{r-i+1}{r}} + \frac{\sigma_{r+1}}{\sigma_i}\left(1 + \sqrt{\frac{n-r}{r}}\right) + \frac{t}{\sqrt{r}}\left(1 + \frac{\sigma_{r+1}}{\sigma_i}\right). \end{aligned} \tag{3.8}$$

3.3. Choice of sketch matrix. In our analysis we have mainly focused on Gaussian embedding matrices, as they are the most well-studied class of random matrices and allow us to derive sharp constants. As mentioned before, a Gaussian matrix $G \in \mathbb{R}^{n\times r}$ with iid entries $\mathcal{N}(0,1)$ can be transformed to a Gaussian embedding matrix by scaling with $1/\sqrt{r}$. The resulting matrix $X = G/\sqrt{r}$ has iid entries $\mathcal{N}(0,1/r)$. Gaussian embedding matrices are often the simplest sketch to implement. While it requires $O(mnr)$ operations to compute AX, when $r = O(1)$ the complexity is optimal as mn is a lower bound, since clearly all entries need to be read. In fact Gaussian sketches can be among the fastest to execute when $r = O(1)$.

Otherwise, if $r \gg 1$, other, more structured, class of sketch matrices have been suggested and employed to speed up the computation of AX. These include the FFT-based subsampled randomized Fourier transform (SRFT) and subsampled randomized Hadamard transform (SRHT) matrices, along with sparse matrices [10, 11, 19].

In particular, subsampled randomized trigonometric transforms (SRTT) such as the subsampled randomized Hadamard, Fourier, discrete cosine or discrete Hartley transforms are random embeddings that allow fast application to a matrix. The use of these sketches can be justified by Theorem 3.4. While they come with weaker theoretical guarantees than Gaussian embeddings [36] (in that a larger sketch is needed to ensure it is an embedding with high probability), empirical evidence strongly suggests that they usually perform just as well (we discuss pathological cases and a remedy in Section 6.4). Throughout the paper, we will denote an SRTT matrix by Θ. Assuming post-multiplication by an SRTT, such that $\Theta \in \mathbb{F}^{n\times r}$ where $n \geq r$, it takes the form

$$\Theta = DFS\sqrt{\frac{n}{r}}, \tag{3.11}$$

where $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix of independent random signs, F is an $n \times n$ orthogonal trigonometric transform, and the columns of $S \in \mathbb{R}^{n\times r}$ are a random subset of the columns of the $n \times n$ identity matrix. We will often write $\mathcal{F} = FD$

3.4. Numerical Experiments. We show how the theoretical result of this section, namely $\sigma_i(AX)/\sigma_i(A) = \mathcal{O}(1)$ for leading i, practically translates to a rank estimation technique. After sketching with an $n \times r$ embedding matrix X, we compute the singular values $\sigma_1(AX), \ldots, \sigma_r(AX)$ and estimate $\text{rank}_\epsilon(A)$ for a given tolerance ϵ to be the first $\hat{r}$ such that $\sigma_{\hat{r}}(AX) > \epsilon \geq \sigma_{\hat{r}+1}(AX)$. We applied this method to

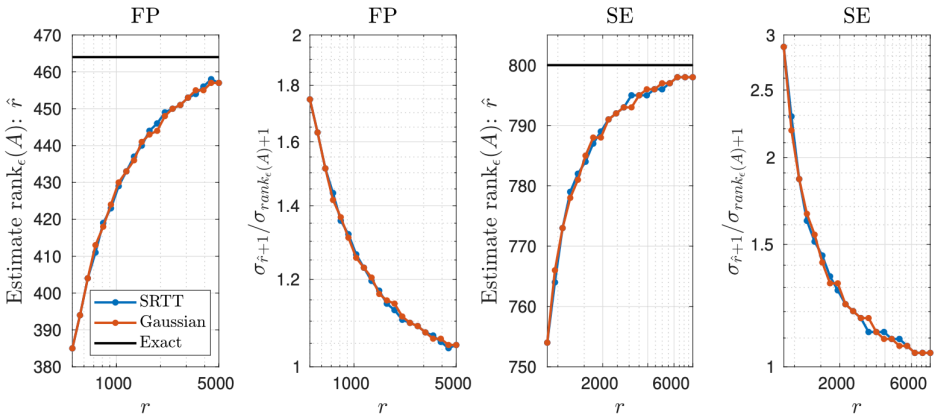


Figure 3.3: Numerical rank estimation of matrices with fast polynomially decaying singular values (FP) and slow exponentially decaying singular values (SE) where only the sketch $AX \in \mathbb{R}^{m\times r}$ is used. The matrices are real and square of dimension $n = 10^5$. The results are shown when using a subsampled randomized DCT embedding matrix (SRTT) or a Gaussian embedding matrix (Gaussian). The black lines indicate the exact numerical rank, $\text{rank}_\epsilon(A)$.

two synthetic example matrices, one with fast polynomially decaying singular values (FP), and one with slow exponentially decaying singular values (SE). The matrices are square of dimension $n = 10^5$. Exactly how the matrices are formed is described in Section 6.

Figure 3.3 shows the results of the experiment. We plot the estimates for the numerical rank $\hat{r}$ that result from varying embedding dimensions r, and the ratio $\sigma_{\hat{r}+1}(A)/\sigma_{\text{rank}_\epsilon(A)+1}$. As both matrices do not have large gaps in the spectrum, the latter is an important quantity to judge the effectiveness of the rank estimator (see Section 1.1). We see that even for small values of r, $\sigma_{\hat{r}+1}(A)$ is very close to $\sigma_{\text{rank}_\epsilon(A)+1}$ and ϵ and well within the acceptable range described in Section 1.

4. Randomized approximate orthogonalization: $\boldsymbol{\sigma_i(YAX)/\sigma_i(AX) = O(1)}$. We have seen that the numerical rank of AX is a good estimator for the numerical rank of A, provided that A is approximately low rank. Although AX is of much smaller dimension than A, it may still be prohibitively expensive to compute its exact singular values. Furthermore, one could question whether it is worth calculating the singular values accurately spending $\mathcal{O}(mr^2)$ work, considering the accuracy that was lost in the sketching step. Here we describe a method, which we call randomized approximate orthogonalization, to cheaply compute estimates of the singular values of AX in $\mathcal{O}(mr\log(m) + r^3)$ operations. The idea is similar to Section 3, in that sketches roughly preserve (the leading) singular values, but as we are working with a tall-skinny matrix AX, here *all* the singular values will be preserved up to a small multiplicative factor.

As for the choice of Y: unlike the first sketch of computing AX, in most cases the structure of A (such as sparsity, if present) is lost in $B = AX$. It is therefore usually recommended that we take Y to be an SRTT embedding, so that $YB = \Theta B$ can be computed in $O(mr\log m)$ operations. We assume this choice below and throughout the remainder of the paper, switching between the notations YAX and ΘAX when appropriate. That is,

$$\Theta = \sqrt{\frac{m}{r_2}} SFD, \tag{4.2}$$

where $S \in \mathbb{R}^{r_2\times m}$ is a sampling matrix, F a square orthogonal trigonometric transform (such as Fourier or DCT) of dimension m and D a diagonal matrix of independent random signs (see (3.11); note the order is reversed as here Θ is applied from the left).

COROLLARY 4.1. *Let* $AX \in \mathbb{R}^{m\times r_1}$, *with* $m \geq r_1$, *and let* $\Theta \in \mathbb{R}^{r_2\times m}$ *be an SRTT matrix as in* (4.2), *where the trigonometric transform* F *in* Θ *satisfies* $\eta = m\max|F_{ij}|^2$. *Let* $0 < \epsilon < 1/3$ *and* $0 < \delta < 1$. *If*

$$6\eta\epsilon^{-2}\left[\sqrt{r_1} + \sqrt{8\log(m/\delta)}\right]^2 \log(r_1/\delta) \leq r_2 \leq m, \tag{4.3}$$

then, with failure probability at most 3δ

$$\sqrt{1-\epsilon} \leq \frac{\sigma_i(\Theta AX)}{\sigma_i(AX)} \leq \sqrt{1+\epsilon}, \tag{4.4}$$

for each $i = 1, \ldots, r_1$.

4.1. Numerical Experiments. We explained how *all* singular values of a tall and skinny matrix $AX \in \mathbb{F}^{m\times r_1}$ can be estimated by the singular values of YAX, where $Y \in \mathbb{F}^{r_2\times m}$ is an embedding matrix and $r_2 \geq r_1$. In the next section we conclude how this, combined with the results of the previous section, leads to a rank estimation algorithm. In the following numerical experiment, we first illustrate the error one can expect in practice in the step from AX to YAX.

We let B be a dense matrix with fast polynomially decaying singular values, $\sigma_i(B) = i^{-3}$, of size $10^5 \times 2000$. We estimate $\sigma_i(B)$ with $\sigma_i(YB)$, where Y is either a subsampled randomized DCT embedding or a Gaussian embedding of dimension 4000. The relative error $|\sigma_i(YB) - \sigma_i(B)|/\sigma_i(B)$ is plotted in Figure 4.1. We see the relative errors are $o(1)$ for each $i = 1, \ldots, 2000$.

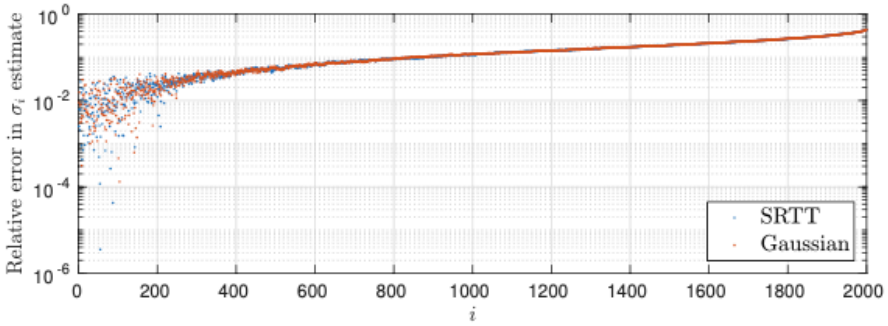


Figure 4.1: Relative error in the singular value estimates of a real matrix B with fast polynomially decaying singular values of size $10^5 \times 2000$ based on the singular values of YB, where $Y \in \mathbb{R}^{4000\times 10^5}$ is an embedding matrix (either a subsampled randomized DCT (SRTT) or a Gaussian embedding matrix).

5. A numerical rank estimation scheme. The numerical rank estimation scheme that follows from the observations

1. $\sigma_i(AX) \approx \sigma_i(A)$ for an approximately low-rank matrix and leading singular values, and
2. $\sigma_i(YAX) \approx \sigma_i(AX)$ for all singular values,

is displayed in pseudocode in Algorithm 1.

Algorithm 1: Rank Estimation with approximate orthogonalization.

Result: Given an $m \times n$ matrix A, a tolerance ϵ and an estimate for the upper bound for rank r_1, this scheme computes an estimate for the ϵ-rank of A.

1. Sketching:

Set $\tilde{r}_1 = \text{round}(1.1r_1)$ to oversample by 10%.

Draw $n \times \tilde{r}_1$ random embedding matrix X.

Form the $m \times \tilde{r}_1$ matrix AX.

2. Approximate orthogonalization:

Set $r_2 = 2\tilde{r}_1$, draw an $r_2 \times m$ SRTT embedding matrix Θ.

Form the $r_2 \times \tilde{r}_1$ matrix ΘAX.

3. Singular value estimates:

Compute the first r_1 singular values of ΘAX.

Let the numerical rank estimate be the first $\hat{r}$ s.t. $\sigma_{\hat{r}+1}(\Theta AX) \leq \epsilon$. If no such $\hat{r}$ exists, repeat the algorithm with a larger r_1, e.g. $r_1 := 2r_1$, by appending to the sketches.

Real-time Machine Learning for Symbol Detection in MIMO-OFDM Systems

Yibin Liang, Lianjun Li, Yang Yi, and Lingjia Liu
Bradley Department of Electrical and Computer Engineering
Virginia Tech, Blacksburg, VA 24060, USA

IEEE INFOCOM 2022 - IEEE Conference on Computer Communications | 978-1-6654-5822-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/INFOCOM48880.2022.9796854

***Abstract*—Recently, there have been renewed interests in applying machine learning (ML) techniques to wireless systems. Nevertheless, ML-based approaches often require a large amount of data in training, and prior ML-based MIMO symbol detectors usually adopt offline learning approaches, which are not applicable to real-time signal processing. This paper adopts echo state network (ESN), a prominent type of reservoir computing (RC), to the real-time symbol detection task in MIMO-OFDM systems. Two novel ESN training methods, namely recursive-least-square and generalized adaptive weighted recursive-least-square, are introduced to enhance the performance of ESN training. Furthermore, a decision feedback mechanism is adopted to improve training efficiency and BER performance. Simulation studies show that the proposed methods perform better than previous conventional and ML-based MIMO symbol detectors. Finally, the effectiveness of our RC-based approach is validated with a software-defined radio (SDR) transceiver and extensive field tests in various real-world scenarios. To the best of our knowledge, this is the first real-time SDR implementation for ML-based MIMO-OFDM symbol detectors. Our work strongly indicates that ML-based signal processing could be a promising and critical approach for future wireless networks.**

***Index Terms*—machine learning, reservoir computing, echo state network, MIMO-OFDM, symbol detection, wireless network, real-time signal processing, software-defined radio**

I. Introduction

Machine learning (ML) has achieved tremendous success in computer science fields, such as pattern and speech recognition [1]–[3], image processing and classification [4]–[6], and natural language processing [7], [8]. Recently, there have been renewed interests and efforts in applying machine learning and data-driven techniques to various aspects of wireless communication systems [9], ranging from modulation and signal classification [10], channel coding [11], transceiver design [12] to radio resource allocation and spectrum sharing [13]. Compared with traditional model-based approaches, ML-based approaches avoid the model mismatch issue and potentially provide better performance by implicitly learning the underlying statistical models from data. However, ML-based approaches usually require a large amount of data to train the supporting neural networks (NN), which are difficult to obtain in real-time wireless systems with limited sets of reference signals (pilots) available. Furthermore, the complexity of the underlying neural networks often introduces high computation costs in terms of both hardware resources and signal processing delay.

The use of orthogonal frequency-division multiplexing (OFDM) in multiple-input multiple-output (MIMO) wireless channels has been one of the driving forces behind the tremendous success in current wireless systems, including 4G/5G cellular networks and wireless local area networks. However, with the introduction of massive MIMO architecture and millimeter-wave bands, conventional model-based signal processing techniques face many new challenges. For example, it is challenging to model and compensate for the characteristics of millimeter-wave band antenna arrays. In addition, traditional signal processing techniques may not scale to massive MIMO scenarios due to exponentially-increasing complexity. ML-based and data-driven techniques may become promising candidates for innovative solutions to solve these issues. However, applying ML-based techniques to the physical layer signal processing is more difficult due to limited training data and real-time processing requirements.

In this paper, we focus on the problem of applying ML-based real-time signal processing to the symbol detection task in MIMO-OFDM systems. An NN-based approach for symbol detection in OFDM systems was provided in [14], where a multi-layer perceptron with three hidden layers was adopted. Convolutional neural networks (CNN) were adopted in [15] to exploit the convolutional nature of multipath channels. An extreme learning machine based method was introduced in [16] to expedite the training process. In [17], model-based and data-driven OFDM receivers were introduced. For MIMO scenarios, [18] applied deep neural networks (DNN) to conduct symbol detection based on the estimated channel inputs and received signals. [19] utilized DNN to fine-tune parameters of conventional symbol detection methods. The performance of CNN-based and DNN-based methods with and without channel state information was compared in [20]. However, all of these methods require a large amount of data to pre-train the NN in an offline manner, making them difficult to be adopted in current communications systems. Furthermore, the offline training approaches usually have difficulty with generalization performance. For example, symbol detection models trained using predefined wireless channel realizations will perform poorly in real-world scenarios with different channel statistics.

Online learning schemes using reservoir computing (RC) have been studied and developed in [21]–[24]. RC requires simpler training procedures and fewer training data than other neural networks, making it a compelling candidate for real-time signal processing in wireless systems. The efficiency of RC-based MIMO-OFDM symbol detection was verified in

[21]. Follow-up work in [22] improved the performance by adding a sliding window to the input of RC. Later, an RC-based deep neural network was introduced in [23] to further improve system performance. Unlike previous work where the NNs are trained on an OFDM frame/subframe basis, scattered pilots were utilized in [24] to update the NN weights on an OFDM symbol basis to facilitate real-time machine learning. The proposed method was implemented on a software-defined radio (SDR) platform for single-input single-output (SISO) OFDM systems. Nevertheless, the training efficiency and real-time performance in MIMO scenarios have not been thoroughly investigated.

This work extends and improves the RC-based symbol detection approach for MIMO-OFDM systems. Specifically, two recursive methods for the real-time symbol-by-symbol training of echo state network (ESN), namely recursive least square (RLS) and generalized adaptive weighted recursive least square (GAW-RLS), are adopted (also reported in [25]), in combination with a decision feedback mechanism. The main contributions of this work are outlined below:

- By adopting the RLS and GAW-RLS methods for ESN training, our symbol detection approach not only improves the BER performance in time-varying channels but also becomes more flexible to different pilot designs. These recursive methods are critical for meeting the real-time processing requirement in real-world MIMO-OFDM systems.
- The new decision feedback mechanism further improves the training process making it applicable to wireless systems with minimal training data. It also improves the BER performance, as validated by simulation studies, which further demonstrates the effectiveness of RC-based symbol detection methods compared with other offline approaches.
- The introduced ML-based scheme has been successfully implemented in a software-defined radio platform. The effectiveness and advantage of the ML-based approach are verified by extensive tests in real-world scenarios. To the best of our knowledge, this is the first real-time implementation of ML-based symbol detection for MIMO-OFDM systems, showing that real-time machine learning is a promising paradigm for future wireless systems.

The structure of the paper is as follows. Section II describes the MIMO-OFDM system model and conventional model-based receiver procedures. Section III provides background for reservoir computing and ESN and discusses our RC-based symbol detection procedures, including the RLS and GAW-RLS methods. Section IV introduces the decision feedback mechanism for ESN training and summarizes the simulation study showing that the new symbol detection methods can achieve better performance than the conventional and previous ML-based symbol detectors. Section V discusses the real-time SDR transceiver built using the GNU Radio [26] framework according to the IEEE 802.11 standard [27]. Extensive field test results for various real-world scenarios in Section VI show that the RC-based symbol detection method consistently achieves better performance than conventional methods.

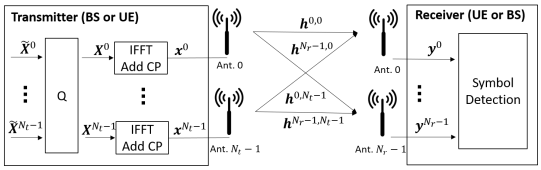


Figure 1. MIMO-OFDM system architecture

II. MIMO-OFDM SYSTEMS

A. System Model

The architecture of a MIMO-OFDM system is shown in Figure 1, where the base station (BS) and user equipment (UE) can function as both transmitter and receiver. N_t independent data streams are transmitted in parallel through N_t antennas and recovered with N_r ($N_r \geq N_t$) receiver antennas. The ith frequency-domain OFDM symbol for the tth data stream is denoted as

$$\tilde{\boldsymbol{X}}_i^t \triangleq [\tilde{X}_i^t(0), \cdots, \tilde{X}_i^t(k), \cdots, \tilde{X}_i^t(N_{sc}-1)]^T, \quad (1)$$

where $\cdot^T$ is matrix transpose operator, N_{sc} is total number of data sub-carriers per OFDM symbol, and $\tilde{X}_i^t(k)$ is the modulated QAM symbol for subcarrier k.

Before OFDM modulation, the ith frequency-domain QAM symbols at sub-carrier k of all data streams are weighted by a pre-coding matrix $\boldsymbol{Q}(k) \in \mathbb{C}^{N_t \times N_t}$ as

$$\boldsymbol{X}_i = \boldsymbol{Q}(k)\tilde{\boldsymbol{X}}_i, \quad (2)$$

where $\boldsymbol{X}_i \triangleq [X_i^0(k), \cdots, X_i^t(k), \cdots, X_i^{N_t-1}(k)]^T$ denotes the output of the pre-coding procedure. If $\boldsymbol{Q}(k)$ is an identity matrix, then $\boldsymbol{X}_i = \tilde{\boldsymbol{X}}_i$.

Next, an inverse fast Fourier transform (IFFT) is performed on $\boldsymbol{X}_i^t$ for each transmitter, and the last N_{cp} samples of the IFFT output are copied and inserted to the beginning of the time-domain signal as cyclic prefix (CP). Denote the ith time domain OFDM symbol at transmitter antenna t as

$$\boldsymbol{x}_i^t \triangleq [x_i^t(0), \cdots, x_i^t(n), \cdots, x_i^t(N_{cp}+N_{sc}-1)]^T, \quad (3)$$

where $x_i^t(n)$ is the nth sample of the ith time domain OFDM symbol. Note that $\boldsymbol{X}_i^t$ can be recovered by removing the CP of $\boldsymbol{x}_i^t$ and conducting a fast Fourier transform (FFT). The time domain OFDM frame at transmitter antenna t is a concatenation of OFDM symbols, denoted as

$$\boldsymbol{x^t} \triangleq [(\boldsymbol{x}_1^t)^T, \cdots, (\boldsymbol{x}_i^t)^T, \cdots, (\boldsymbol{x}_N^t)^T]^T, \quad (4)$$

where N is the total number of OFDM symbols in an OFDM frame. The received signal is a superposition of transmitted signals over the MIMO channel, and the received time domain OFDM frame $\boldsymbol{y^r}$ for antenna r can be expressed as

$$\boldsymbol{y}^r = \sum_{t=0}^{N_t-1} u(\boldsymbol{x}^t) \circledast \boldsymbol{h}^{r,t} + \boldsymbol{n}, \quad 0 \leq r < N_r, \quad (5)$$

where $u(\cdot)$ is a non-linear function represents the signal distortion caused by transmitter circuits, such as power amplifier

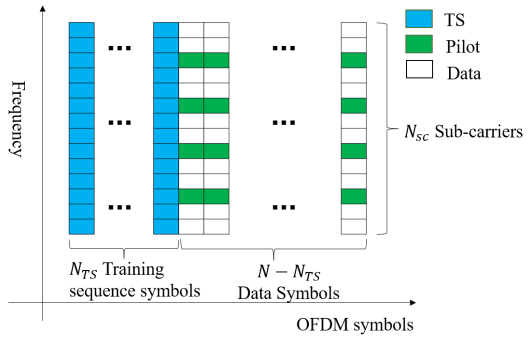


Figure 2. Wi-Fi frame structure and pilot patterns illustrating the training overhead

(PA) [28]; $\circledast$ is the convolution operator; $\boldsymbol{n}$ is additive Gaussian noise; $\boldsymbol{h}^{r,t} = [h_0^{r,t}, h_1^{r,t}, \cdots, h_{N_{cp}}^{r,t}]^T \in \mathbb{C}^{(N_{cp}+1)}$ denotes the channel impulse responses between receiver antenna r and transmitter antenna t. In general $\boldsymbol{h}^{r,t}$ is gradually changing over time due to the dynamic nature of the wireless environment. This paper adopts the quasi-static channel assumption that $\boldsymbol{h}^{r,t}$ remains constant within one OFDM symbol and changes across OFDM symbols.

At the receiver, the ith received time domain OFDM symbol of antenna r is denoted as

$$\boldsymbol{y}_i^r \triangleq [y_i^r(0), \cdots, y_i^r(n), \cdots, y_i^r(N_{cp} + N_{sc} - 1)]^T, \quad (6)$$

and its frequency domain counterpart as

$$\boldsymbol{Y}_i^r \triangleq [Y_i^r(0), \cdots, Y_i^r(k), \cdots, Y_i^r(N_{sc} - 1)]^T. \quad (7)$$

The goal of symbol detection is to recover all transmitted data streams $\tilde{\boldsymbol{X}}_i^t$ by jointly processing received signals $\boldsymbol{y}_i^r$ from all receiver antennas.

For facilitating symbol detection, MIMO-OFDM systems embed known information, a.k.a reference signals, into the OFDM symbols $\tilde{\boldsymbol{X}}_{\boldsymbol{i}}^{\boldsymbol{t}}$. For example, in Wi-Fi systems, as specified in standard 802.11 [27], the first N_{TS} OFDM symbols in a frame are designated as training sequence (TS), and scattered pilots are placed among the remaining OFDM symbols. The training overhead pattern is illustrated in Figure 2. This paper refers to the first N_{TS} OFDM symbols in a frame as TS and the remaining OFDM symbols as data symbols.

B. Conventional Methods

The conventional symbol detection method in MIMO-OFDM systems includes two steps: First, the channel is estimated using the training sequence with the linear minimum mean square error (LMMSE) method. Then the estimated channel is used to recover transmitted symbols by an LMMSE symbol detector.

1) LMMSE channel estimation: Assume the first N_{TS} transmitted OFDM symbols in an OFDM frame is designed as training sequence (TS) and is known to the receiver. Then the received TS symbols at sub-carrier k can be expressed as

$$\boldsymbol{Y}_{TS}(k) = \boldsymbol{H}(k)\boldsymbol{X}_{TS}(k) + \boldsymbol{G}, \quad (8)$$

where

$$\boldsymbol{Y}_{TS}(k) \triangleq \begin{bmatrix} Y_1^0(k) & \cdots & Y_{N_{TS}}^0(k) \\ \vdots & \vdots & \vdots \\ Y_1^{N_r-1}(k) & \cdots & Y_{N_{TS}}^{N_r-1}(k) \end{bmatrix}, \quad (9)$$

$$\boldsymbol{X}_{TS}(k) \triangleq \begin{bmatrix} X_1^0(k) & \cdots & X_{N_{TS}}^0(k) \\ \vdots & \vdots & \vdots \\ X_1^{N_t-1}(k) & \cdots & X_{N_{TS}}^{N_t-1}(k) \end{bmatrix}, \quad (10)$$

$$\boldsymbol{H}(k) \triangleq \begin{bmatrix} H^{0,0}(k) & \cdots & H^{0,N_t-1}(k) \\ \vdots & \vdots & \vdots \\ H^{N_r-1,0}(k) & \cdots & H^{N_r-1,N_t-1}(k) \end{bmatrix}, \quad (11)$$

and $\boldsymbol{G}$ is the additive Gaussian noise with variance σ^2. The wireless channel at sub-carrier k is estimated by LMMSE as

$$\hat{\boldsymbol{H}}(k) = \boldsymbol{Y}_{TS}(k)\boldsymbol{X}_{TS}^*(k)\big(\boldsymbol{X}_{TS}(k)\boldsymbol{X}_{TS}^*(k) + \sigma^2\boldsymbol{I}\big)^{-1}, \quad (12)$$

where $\cdot^*$ denotes conjugate transpose, $\boldsymbol{I}$ is the identity matrix.

2) LMMSE symbol detection: The maximum likelihood method is the optimal detector for minimizing the error probability for MIMO symbol detection. However, this optimal method has exponential complexity and thus is not used in practical wireless systems. Sphere decoding (SD) is a non-convex solver that performs optimized maximum likelihood detection under ideal assumptions but has higher computation complexity than linear receivers. LMMSE symbol detection is a linear method widely used in wireless communications systems due to its low complexity. The LMMSE detection procedure is described as follows.

Using the estimated channel $\hat{\boldsymbol{H}}$, the ith received OFDM symbol at sub-carrier k can be expressed as

$$\boldsymbol{Y}_i(k) = \hat{\boldsymbol{H}}(k)\boldsymbol{X}_i(k) + \boldsymbol{G}, \quad (13)$$

where $\boldsymbol{Y}_i(k) \triangleq [Y_i^0(k), \ldots, Y_i^{N_r-1}(k)]^T$, and $\boldsymbol{X}_i(k) \triangleq [X_i^0(k), \ldots, X_i^{N_t-1}(k)]^T$. Then, the LMMSE symbol detection is performed as

$$\hat{\boldsymbol{X}}_i(k) = \big(\hat{\boldsymbol{H}}^*(k)\hat{\boldsymbol{H}}(k) + \sigma^2\boldsymbol{I}\big)^{-1}\hat{\boldsymbol{H}}^*(k)\boldsymbol{Y}_i(k). \quad (14)$$

The LMMSE detector requires estimated channel as input, as well as prior knowledge of noise variance and channel statistics. The limitation and performance loss with imperfect channel estimation will be demonstrated in the simulation studies as well as real-world tests.

III. RC-based Symbol Detection

Reservoir computing [29]–[32] is a special kind of recurrent neural network (RNN) model that emerged from chaotic systems. An RC system consists of a reservoir for mapping inputs onto a high-dimensional space and a readout for mapping internal reservoir states to target outputs, as shown in Figure 3. The reservoir can be regarded as a randomly generated chaotic recurrent network. Once the reservoir weights are generated, the weights for input mapping and recurrent connections remain fixed, while only the readout weights need to be trained.

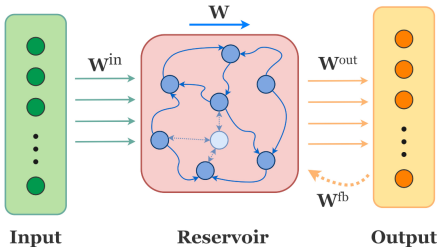


Figure 3. Basic RC structure

This simplified training procedure significantly reduces the training overhead and makes RC a promising candidate for wireless systems and applications where training information is very limited.

A. Echo State Network

Echo state network is one of the prominent candidates of reservoir computing. It has been shown that ESN can outperform traditional RNNs in non-linear system identification and temporal information processing applications [33], [34].

The basic computational model of ESN consists of three layers as illustrated in Figure 3: 1) an input layer with N_i nodes for data $\boldsymbol{x}(n) \in \mathbb{R}^{N_i}$, 2) a reservoir layer with N_s neurons for evaluating the current state $\boldsymbol{s}(n) \in \mathbb{R}^{N_s}$ using the current inputs $\boldsymbol{x}(n)$, previous state $\boldsymbol{s}(n-1)$, and outputs $\boldsymbol{y}(n-1) \in \mathbb{R}^{N_o}$, and 3) an output layer with N_o nodes. The reservoir dynamics is governed by the following state update equation:

$$\boldsymbol{s}(n) = f\big(\boldsymbol{W}\boldsymbol{s}(n-1) + \boldsymbol{W}^{in}\boldsymbol{x}(n) + \boldsymbol{W}^{fb}\boldsymbol{y}(n-1)\big), \quad (15)$$

where $\boldsymbol{W} \in \mathbb{R}^{N_s \times N_s}$ is the state transition matrix, $\boldsymbol{W}^{in} \in \mathbb{R}^{N_s \times N_i}$ is the input weight matrix, $\boldsymbol{W}^{fb} \in \mathbb{R}^{N_s \times N_o}$ is the output feedback matrix, and $f(\cdot)$ is the activation function for reservoir neurons. Define the extended system state as

$$\boldsymbol{z}(n) = [\boldsymbol{s}(n)^T, \boldsymbol{x}(n)^T]^T, \quad (16)$$

the current ESN output $\boldsymbol{y}(n) \in \mathbb{R}^{N_o}$ is obtained by

$$\boldsymbol{y}(n) = g(\boldsymbol{W}^{out}\boldsymbol{z}(n)), \quad (17)$$

where $\boldsymbol{W}^{out} \in \mathbb{R}^{N_o \times (N_s + N_i)}$ is the output weights matrix, and $g(\cdot)$ is the output activation function.

Assuming a batch of N_{train} training samples is collected, the ESN can be trained by learning the output weights $\boldsymbol{W}^{out}$ using the following steps. First, feed the ESN with the inputs $\boldsymbol{X} = \{\boldsymbol{x}(1), \cdots, \boldsymbol{x}(N_{train})\}$ to generate the reservoir states $\boldsymbol{S} = \{\boldsymbol{s}(1), \cdots, \boldsymbol{s}(N_{train})\}$, the extended states $\boldsymbol{Z} = \{\boldsymbol{z}(1), \cdots, \boldsymbol{z}(N_{train})\}$, and the corresponding outputs $\boldsymbol{Y} = \{\boldsymbol{y}(1), \cdots, \boldsymbol{y}(N_{train})\}$ according to (15) to (17). Next, obtain $\boldsymbol{W}^{out}$ by minimizing the loss between outputs $\boldsymbol{Y}$ and known training labels $\tilde{\boldsymbol{Y}} = \{\tilde{\boldsymbol{y}}(1), \cdots, \tilde{\boldsymbol{y}}(N_{train})\}$ as

$$\boldsymbol{W}^{out} = \underset{\boldsymbol{W}^{out}}{\operatorname{argmin}} \operatorname{Loss}(\boldsymbol{Y}, \tilde{\boldsymbol{Y}}). \quad (18)$$

When using the identity function as the output activation and Frobenius norm as the loss function, we can rewrite the minimization problem as

$$\boldsymbol{W}^{out} = \underset{\boldsymbol{W}^{out}}{\operatorname{argmin}} \|\tilde{\boldsymbol{Y}} - \boldsymbol{W}^{out}\boldsymbol{Z}\|_F^2, \quad (19)$$

which is a linear regression problem and can be solved using the least square (LS) estimation:

$$\boldsymbol{W}^{out} = \tilde{\boldsymbol{Y}}\boldsymbol{Z}^{\dagger}, \quad (20)$$

where $\boldsymbol{Z}^{\dagger}$ is the Moore-Penrose inverse of $\boldsymbol{Z}$.

B. Recursive Learning Methods

When using the LS estimation method, the output weights can only be obtained after all training samples are collected. In real-time applications such as wireless communications, however, recursive learning methods are required to update the output weights in a timely manner and track the dynamics of the underlying environment. This paper adopts two recursive methods for training ESN, namely recursive least square (RLS) [34], [35] and generalized adaptive weighted recursive least squares (GAW-RLS) [36].

1) Recursive least square (RLS): RLS is designed to find the optimal output weights at the current training step n such that the sum of discounted previous errors is minimized

$$\boldsymbol{W}^{out}(n) = \underset{\boldsymbol{W}^{out}}{\operatorname{argmin}} \sum_{m=1}^{n} \lambda^{n-m} \|\boldsymbol{y}(m) - \boldsymbol{W}^{out}\boldsymbol{z}(m)\|_2^2, \quad (21)$$

where $\lambda \in (0, 1]$ is known as the forgetting factor. When $\lambda < 1$, the minimization problem (21) gives more weight to errors associated with recent samples than older ones. In other words, RLS emphasizes recent observations and tends to forget the past, making it an adaptive algorithm with tracking capabilities. Such behavior is needed in wireless communication scenarios where the underlying environment gradually changes.

For finding the optimal output weights, the current weights are updated recursively with previous weights and current prediction as follows,

$$\boldsymbol{W}^{out}(n) = \boldsymbol{W}^{out}(n-1) + \boldsymbol{e}_{n-1}(n)\boldsymbol{k}^T(n), \quad (22)$$

where

$$\boldsymbol{e}_{n-1}(n) = \boldsymbol{y}(n) - \boldsymbol{W}^{out}(n-1)\boldsymbol{z}(n)$$

is the current prediction error using output weights from previous step, and

$$\boldsymbol{k}(n) = \frac{\boldsymbol{\Psi}(n-1)\boldsymbol{z}(n)}{\lambda + \boldsymbol{z}^T(n)\boldsymbol{\Psi}(n-1)\boldsymbol{z}(n)}. \quad (23)$$

$\boldsymbol{\Psi}(n)$ is inverse of the weighted extended state correlation matrix, defined as

$$\boldsymbol{\Psi}(n) = \Big(\sum_{m=0}^{n} \lambda^{n-m}\boldsymbol{z}(m)\boldsymbol{z}^T(m)\Big)^{-1}, \quad (24)$$

and it can be updated recursively by

$$\boldsymbol{\Psi}(n) = \lambda^{-1}\big(\boldsymbol{\Psi}(n-1) - \boldsymbol{k}(n)[\boldsymbol{z}^T(n)\boldsymbol{\Psi}(n-1)]\big). \quad (25)$$

RLS can be regarded as a standard backpropagation algorithm with an adaptive learning rate determined by $\boldsymbol{k}(n)$.

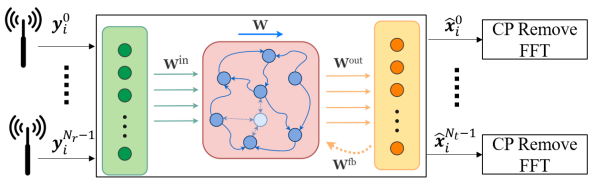


Figure 4. RC-based symbol detection architecture

2) Generalized adaptive weighted RLS (GAW-RLS): Since the input samples of NN are usually corrupted by noise, new training samples may have different levels of importance, i.e., highly corrupted samples may not help much. Therefore, it is reasonable to generalize RLS by adding a weighting factor $\omega \in [0, 1]$ to new training samples. As a result, the minimization problem in (21) becomes:

$$\boldsymbol{W}^{out}(n) = \underset{\boldsymbol{W}^{out}(n)}{\operatorname{argmin}} \sum_{m=1}^{n} \lambda^{n-m} \omega(m) \boldsymbol{D}(m, n), \tag{26}$$

where

$$\boldsymbol{D}(m, n) = \|\boldsymbol{y}(m) - \boldsymbol{W}^{out}(n)\boldsymbol{z}(m)\|_2^2.$$

This problem can be solved the same way as (22). The only modification is replacing the calculation of $\boldsymbol{k}(n)$ in (23) by

$$\boldsymbol{k}(n) = \frac{\omega(n)\boldsymbol{\Psi}(n-1)\boldsymbol{z}(n)}{\lambda + \omega(n)\boldsymbol{z}^T(n)\boldsymbol{\Psi}(n-1)\boldsymbol{z}(n)}. \tag{27}$$

According to [36], [37], the weighting factor ω is usually set inversely proportional to the prediction error as follows:

$$\omega(n) \propto 1/\|\boldsymbol{e}_{n-1}(n)\|_2 \tag{28}$$

Therefore, RLS can be regarded as a special case of GAW-RLS when $\omega = 1$. GAW-RLS adds additional flexibility in the training process to cope with the dynamic MIMO channels.

C. ESN-based Symbol Detection

In this work, we design an ESN-based symbol detection method for MIMO-OFDM systems, an online method that only utilizes existing reference signals. The system architecture is illustrated in Figure 4. The received symbols from multiple antennas are denoted as $\boldsymbol{y_i^r}$, and the predicted symbols are denoted $\hat{\boldsymbol{x}}_{\boldsymbol{i}}^{\boldsymbol{t}}$. Adopting the recursive methods for training ESN, the training procedure is divided into two steps: 1) using TS to train the initial ESN output weights, and 2) using pilots to update the weights recursively with each OFDM symbol.

1) Initial training with TS: The training data set can be expressed as an input-label tuple and prepared as

$$\begin{aligned}\Phi_{TS} &\triangleq \{\boldsymbol{I}_{TS}, \boldsymbol{L}_{TS}\} \\ &= \{[\boldsymbol{y_1}, \boldsymbol{y_2}, \cdots, \boldsymbol{y_{N_{TS}}}], [\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_{N_{TS}}}]\},\end{aligned} \tag{29}$$

where $\boldsymbol{y}_i \triangleq [\boldsymbol{y}_i^0, \cdots, \boldsymbol{y}_i^{N_r-1}]^T$, $\boldsymbol{x}_i \triangleq [\boldsymbol{x}_i^0, \cdots, \boldsymbol{x}_i^{N_t-1}]^T$. The initial ESN output weights are obtained using the least square method described by (20).

2) Recursive training using pilots: For each data symbol (the ith symbol, $i > N_{TS}$), the training tuple is prepared as:

$$\Phi_{pilot,i} \triangleq \{\boldsymbol{I}_{pilot,i}, \boldsymbol{L}_{pilot,i}\} \tag{30}$$

where

$$\boldsymbol{I}_{pilot,i} \triangleq [\boldsymbol{I}_{pilot,i}^0, \cdots, \boldsymbol{I}_{pilot,i}^{N_r-1}]^T, \tag{31}$$

$$\boldsymbol{L}_{pilot,i} \triangleq [\boldsymbol{L}_{pilot,i}^0, \cdots, \boldsymbol{L}_{pilot,i}^{N_t-1}]^T. \tag{32}$$

Training input $\boldsymbol{I}_{pilot,i}^r$ is obtained by first nulling the data sub-carriers of $\boldsymbol{Y}_i^r$, then converting it to time domain, and finally adding the CP, i.e.,

$$\boldsymbol{I}_{pilot,i}^r = f_{cp}(F^H \Sigma \boldsymbol{Y}_i^r), \tag{33}$$

where $f_{cp}(\cdot)$ stands for adding CP; F^H is the inverse Fourier transform matrix; Σ is a diagonal matrix. The diagonal entries of Σ are 0 at data sub-carrier locations and 1 on pilot sub-carrier locations. Similarly, the training label is prepared as

$$\boldsymbol{L}_{pilot,i}^t = f_{cp}(F^H \Sigma \boldsymbol{X}_i^t). \tag{34}$$

We can see that both the training input and label are only related to pilot information. After the output weights have been updated using new training samples $\Phi_{pilot,i}$, ESN will take the ith received symbol $\boldsymbol{y}_i$ as input to infer the transmitted symbol.

In terms of the training method, instead of the one-shot matrix inversion used for initial training, the real-time recursive method is more suitable because it updates the ESN weights based on new training samples in a timely manner, which, in essence, tracks the underlying channel change and achieves better performance. Nevertheless, pilots are sparsely placed in the frequency domain; therefore, ESN may not learn the complete system information using only the pilots. Therefore, the decision feedback mechanism is introduced to address this limitation.

IV. Symbol Detection with Decision Feedback

The recursive training methods address the real-time requirement for ESN training in time-varying channels. At the same time, they also limit the number of training samples for each ESN training operation. For MIMO channels, the limited number of pilots may not be enough to train the ESN accurately. In addressing these performance trade-offs, the decision feedback mechanism is combined with recursive approaches to further improve the performance of ESN-based symbol detection. Furthermore, the decision feedback method is also applicable to flexible pilot patterns in MIMO-OFDM systems. The following sections refer to the RC-based symbol detection with decision feedback as RC-DF.

A. Decision Feedback Training

After the initial and recursive training procedures, the inferred OFDM data symbols are used to construct new training data for re-training the ESN. Specifically, denote the inferred ith transmitted OFDM symbol as $\hat{\boldsymbol{x}}_i^t$, obtain its frequency counterpart $\hat{\boldsymbol{X}}_i^t$ by removing CP and performing FFT, then

map $\hat{\boldsymbol{X}}_i^t$ to the nearest QAM symbol to obtain $\bar{\boldsymbol{X}}_i^t$. Denote the decision feedback training data set as

$$\Phi_{DF,i} \triangleq \{\boldsymbol{I}_{DF,i}, \boldsymbol{L}_{DF,i}\} \tag{35}$$

where $\boldsymbol{I}_{DF,i} \triangleq \left[\boldsymbol{I}_{DF,i}^0, \boldsymbol{I}_{DF,i}^1, \ldots, \boldsymbol{I}_{DF,i}^{N_r-1}\right]^T$, and $\boldsymbol{L}_{DF,i} \triangleq \left[\boldsymbol{L}_{DF,i}^0, \boldsymbol{L}_{DF,i}^1, \ldots, \boldsymbol{L}_{DF,i}^{N_t-1}\right]^T$. The training input is prepared as $\boldsymbol{I}_{DF,i}^r = \boldsymbol{y}_i^r$, and the training label is prepared as

$$\boldsymbol{L}_{DF,i}^t = f_{cp}(F^H \bar{\boldsymbol{X}}_i^t). \tag{36}$$

After the ESN weights are updated using the GAW-RLS method for the current OFDM symbol, the symbol detection procedure moves to the next OFDM symbol. The procedure is summarized in Algorithm 1.

Algorithm 1 Symbol Detection with Decision Feedback

for Each OFDM frame **do**
  Initialize input and recurrent weights
  Collect initial training data $\boldsymbol{\Phi}_{TS}$ according to (29)
  Use $\boldsymbol{\Phi}_{TS}$ to obtain initial weights according to (20)
  for OFDM symbol $i = N_{TS}+1 : N$ **do**
    Prepare dataset $\boldsymbol{\Phi}_{pilot,i}$ according to (30)
    Use $\boldsymbol{\Phi}_{pilot,i}$ to update weights by GAW-RLS
    Infer ith transmitted symbol $\hat{\boldsymbol{x}}_i^t$ and $\bar{\boldsymbol{X}}_i^t$
    Prepare DF dataset $\boldsymbol{\Phi}_{DF,i}$ as defined in (35)
    Use $\boldsymbol{\Phi}_{DF,i}$ to update ESN weights by GAW-RLS
  end for
end for

B. Simulation and Analysis

In this section, the performance of introduced RC-DF methods is evaluated using numerical simulations. We first describe the settings of the experiments and then present the performance results and comparisons with conventional model-based approaches and other learning-based strategies.

A MIMO-OFDM system with $N_t = 4$, and $N_r = 4$ is adopted. The length of training sequence $N_{TS} = 8$, and the frame length $N = 100$. For a total number of $N_{sc} = 64$ sub-carriers, 4 sub-carriers carry pilots, 48 sub-carriers carry data, and the rest are null sub-carriers. The CP length $N_{cp} = 16$. 16-QAM modulation is used to generate information symbols $\tilde{X}_i^t(k)$. The wireless channels coefficients are generated following the 3GPP propagation channel model [38], [39], and the specific model chosen is the Extended Pedestrian A model (EPA), for which the user speed is 10 km/h.

In all the simulations, the total number of recurrent neurons in the ESN is $N_s = 32$. A sliding window of size 4 samples is added to the ESN input layer as suggested in [22]. The input weights are randomly generated from a uniform distribution, and the state transition matrix is also randomly generated with a spectrum radius of 0.2 to satisfy the echo state property [40]. Feedback weights are not used and are set to zeros. The forgetting factor λ for GAW-RLS is set to 0.9995, and the weighting factor is set as $\omega(n) = 1/(1+\exp(\alpha + \beta \log(\|\boldsymbol{e}_{n-1}(n)\|_2^2)))$, where $\alpha = 27$ and $\beta = 15$ are empirical values with good performance.

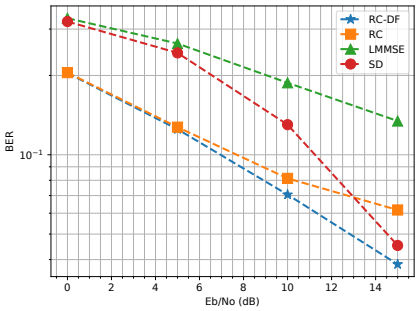


Figure 5. BER performance **without** PA distortion

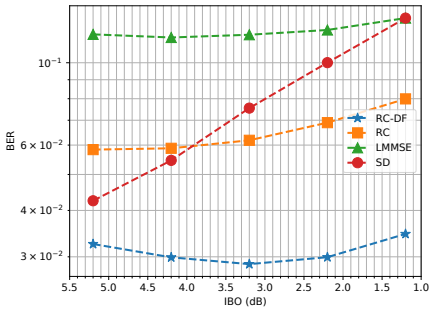


Figure 6. BER performance **with** PA distortion

At the transmitter side, the pre-coding matrix Q is set to the identity matrix. In investigating the performance of all methods with regard to the RF circuit non-linearity, the OFDM signal is passed through a power amplifier (PA) model before transmitting to the wireless channel. The PA model adopted is RAPP [41], in which the output of PA is modeled as:

$$u(x) = \frac{x}{\left[1 + \left(\frac{|x|}{x_{sat}}\right)^{2\rho}\right]^{1/2\rho}}, \tag{37}$$

where x is the input of PA, x_{sat} is the PA saturation level, and ρ is the smoothing parameter. When $|x| \ll x_{sat}$ and $u(x) \approx x$, the PA works in the linear region, and the signal has no distortion. On the other hand, when $|x| \to x_{sat}$, the PA works in the non-linear region, and the signal is highly compressed. In the simulation, $x_{sat} = 1$ and $\rho = 3$ are used.

The performance of two learning-based methods and two conventional model-based methods are compared: RC-DF is the learning-based method introduced in this paper, RC is the RC-based method introduced in [24]. Two conventional methods are LMMSE symbol detection and sphere decoding (SD) [42].

The BER performance without PA distortion is shown in Figure 5. In this case, PA is working in the linear region, in which the ratio between PA's saturation power to the input power, a.k.a input back-off (IBO), is greater than 8 dB. The results show that all learning-based methods outperform model-based methods in low to median Eb/No regime.

Conventional model-based methods need estimated channel information as input, which is inaccurate when noise is high. In contrast, the learning-based method can directly learn a mapping from received symbols to transmitted ones without intermediate channel estimation steps. When Eb/No is high, RC can achieve close to SD performance, while the introduced RC-DF outperforms SD, owing to the decision feedback that provides more informative training data.

Figure 6 shows the BER performance with PA distortion. In this case, input signal power is close to the PA saturation region, and the peak-to-average-power-ratio (PAPR) of the OFDM signal is higher than the IBO value. In the simulation, the OFDM PAPR range is between 6 dB to 9 dB, while IBO is set to smaller than 5.5 dB. The BER performance results show that learning-based methods work better than conventional methods when IBO is low, meaning they can learn and compensate for the non-linearity of the PA. It should be noted that the decrease of IBO produces two opposite effects: 1) the positive effect is SNR improves as signal power increases; 2) the adverse effect is signal distortion. The BER of RC-DF first decreases shows that it can recover the signal when the distortion is not severe; therefore, it benefits from the improved SNR. On the other hand, when the IBO is too low, RC-DF also suffers from high signal distortion as other methods do. This phenomenon demonstrates that RC-DF is more robust to signal distortion than other methods. Overall, the introduced RC-DF method achieves the best performance.

V. Real-time SDR Implementation

The RC-based symbol detection scheme can be applied to IEEE 802.11 wireless system in a plug-and-play manner by adopting the existing frame structure and transmission procedures. Simulation studies in the previous section show that the proposed RC-based symbol detection method can provide better performance than conventional model-based methods. However, only a limited number of channel models and test cases are used, which could be very different from real-world scenarios. Furthermore, the simplified RAPP model of non-linear effects is used because it is very difficult or even impossible to model all hardware-related impairment using simulation techniques, such as clock timing offset, non-stationary phase noise, and complex RF characteristics. Finally, we need to have a concrete implementation to evaluate and demonstrate the real-time capabilities of the proposed ML-based techniques. Therefore, a real-time SDR transceiver has been built using the GNU Radio framework, as described in the following sub-sections.

A. System Parameters

The IEEE 802.11 standard, which has similar MIMO-OFDM features as 4G/5G cellular networks but has better RF hardware support for SDR implementation and testing, is adopted for the real-time MIMO-OFDM transceiver implementation. The conventional symbol detection methods are also implemented and compared with the proposed RC-based symbol detection methods with decision feedback.

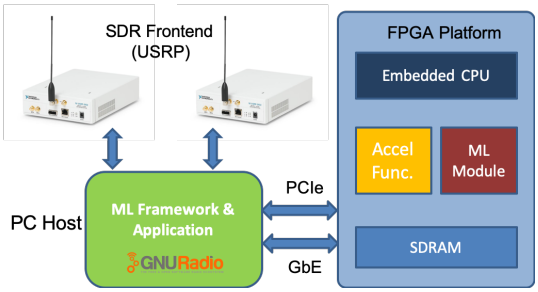


Figure 7. Real-time machine learning for SDR Platform

The high-throughput (HT) frame structure in IEEE 802.11 is used for data packet transmission. A packet frame, including the training sequence and data, contains a total of 100 OFDM symbols for each transmission antenna. FFT size of 64 is used for OFDM modulation and demodulation. Due to the speed limit of the SDR hardware, a base-band sampling rate of 10M samples/s is used, which puts more stringent requirements on the accuracy of carrier frequency synchronization and clock timing error estimation.

The MIMO-OFDM transceiver supports up to four transmit and receive antennas. However, due to the limited number of SDR equipment available, two antennas per transmitter/receiver and two simultaneous spatial streams are used in the field tests. In our SDR-based implementation, all the high-throughput (HT) format preamble signals are used as TS for RC initial training. However, these signals do not have good statistical properties in the original specification and are unsuitable for ESN training. Therefore, some of these preamble signals are replaced with new predefined sequences with better statistical properties in the actual implementation.

B. Real-time Platform

The architecture of the real-time platform is illustrated in Figure 7. The MIMO-OFDM transceiver application is implemented using GNU Radio and runs on an Ubuntu Linux PC host. Multiple SDR front-ends can be connected to the host via gigabit Ethernet (GbE) and USB ports. Two models of the universal software radio peripheral (USRP) by Ettus Research, namely USRP N210 and USRP B210, were used in field tests. Multiple USRP N210s can be synchronized as a single MIMO transceiver. Each USRP N210 is equipped with an SBX RF front-end capable of transmitting/receiving in the 400MHz-4.4GHz frequency range. The USRP B210 is capable of transmitting/receiving in the 70MHz-6GHz range with two transmit and two receive antennas.

A hardware ESN accelerator has been designed to improve signal processing speed using the field programmable gate array (FPGA) on a Xilinx VC707 FPGA evaluation board. The ESN accelerator is used for computing the reservoir state updates for both training and inference operations. The ESN accelerator communicates with host programs via PCIe or GbE interfaces, and a GNU Radio application can directly utilize its functionality using built-in Linux system interfaces.

Besides the software-hardware co-design of the machine learning for SDR platform, many optimization efforts have been spent to make the transceiver work in real-time. For example, most of the customized GNU Radio modules are written as C++ classes, and vector optimization with the VOLK library [43] is used for time-consuming operations. In addition, the ESN symbol detection algorithm is highly optimized by adopting parallel computing techniques and linear algebra libraries.

C. Receiver Implementation

Figure 8 shows the MIMO-OFDM receiver architecture and modules as a GNU Radio flow graph. The "USRP source" block takes in base-band IQ samples for multiple antennas from USRP front-ends. The "Packet Detect" block is responsible for detecting the start of an incoming packet and performing coarse and fine frequency offset estimation and compensation. For conventional symbol detection methods, the received time-domain signal is first converted to frequency-domain by the "OFDM Demod" block. The "ChanEst & CPE" block is responsible for estimating both the MIMO channel coefficients using the packet preamble and common phase errors (CPE) using continuous pilot tones. Finally, the "Equalizer and demapper" block uses the LMMSE detection method to recover the transmitted QAM symbols.

For ESN-based symbol detection, the received time-domain samples are directly fed to the "RC demod" block, which trains and predicts the transmitted signals using all received OFDM symbols in a frame. The resulting average bit error rate (BER) and estimated signal-to-noise ratio (SNR) are computed and logged when a packet is processed. The resulting QAM signal constellation is displayed in real-time for debugging and testing. In addition, the received bit streams are saved for further offline processing, such as stream de-multiplexing, de-interleaving, and channel decoding.

VI. Field Tests

Extensive field tests have been conducted in various environments, including multiple apartment buildings, several office rooms, and an RF laboratory in our department buildings. These real-world tests have helped us improve the transceiver implementation and gain much insight in addressing MIMO channel characteristics. Due to the space limit, only the summary results for one of the test campaigns are described in the following sub-sections.

A. Test Setup

The test campaign was conducted in an apartment with representative real-world channel characteristics. The apartment is closely surrounded by other apartments and tall trees, creating abundant multipath reflections. Inside the apartment, there are many metallic closet doors, which are responsible for signal blockage and strong signal reflection paths. For all the tests, the base station (BS) was fixed at the corner of the bedroom, and the user equipment (UE) was moved around to different locations, as illustrated in Figure 9.

The system performance was evaluated using different scenarios as listed in Table I, in which the scenario number corresponds to the UE location in Figure 9. Generally, the signal strength decreases as the UEs move further away from the BS. It should be noted that the system had the worst performance in location 5 due to the blockage of all direct paths and the presence of strong multipath reflection.

B. Results and Analysis

For each scenario, five tests were carried out with slightly different receiver orientation and antenna angles such that the resulting multipath profile would change randomly. At least 100 data frames were transmitted and processed in each test. The average bit error rate (BER) was measured after detecting QAM symbols without decoding convolutional or LDPC code blocks . The estimated MIMO channel coefficients for each data frame were saved for analyzing the multipath components and channel conditions. The average signal-to-noise ratio (SNR) estimation was also reported for analysis.

1) Effects of Non-linearity: For evaluating the effects of non-linearity on transceiver performance, the transmitter and receiver gains were adjusted to the non-linear region. The SBX RF front-end in USRP N210 has discrete RF amplifiers with an adjustable gain range of 0-31.5dB, but non-linear effects appear when the gain is above 28dB. The B210 has integrated LNAs within its AD9361 transceiver IC with an adjustable gain range of 0-76 dB, but the signal quality starts to degrade significantly when the gain is above 50 dB. In this test, the transceiver pair was placed at the location for scenario 1, the transmitter gain was set at 28dB, and the receiver gain was changed from 35dB to 55dB. Figure 10 summarizes the test results. The RC-based method is more robust against the transceiver non-linearity than the conventional method.

2) Average BER Results: In test scenarios 1 to 3, there were strong line-of-sight and reflection paths between the transmitter and receiver; the signal changed dramatically with minor changes in the receiver position. In scenarios 5 and 6, there were no line-of-sight paths between the transmitter and receiver, but refection paths were present. The channel conditions varied significantly from test to test. The ESN-based symbol detection worked rather well in these complex channel conditions and significantly outperformed the LMMSE symbol detector. In scenario 7, there were fewer multipath components, and the receiver performance was better than the previous scenarios, but the ESN-based detector still had better performance. Overall, the ESN-based detector had a lower average BER, as shown in Figure 11.

In summary, the performance of the conventional LMMSE symbol detection method is on par with the ESN detector when the received signal strength is high, and the channel condition is good. However, the ESN-based method works much better when the received signal strength is low (high receiver gain), or there are complex MIMO channel conditions. The ESN-based symbol detector is also more robust against non-linear effects in the transceiver signal chain.

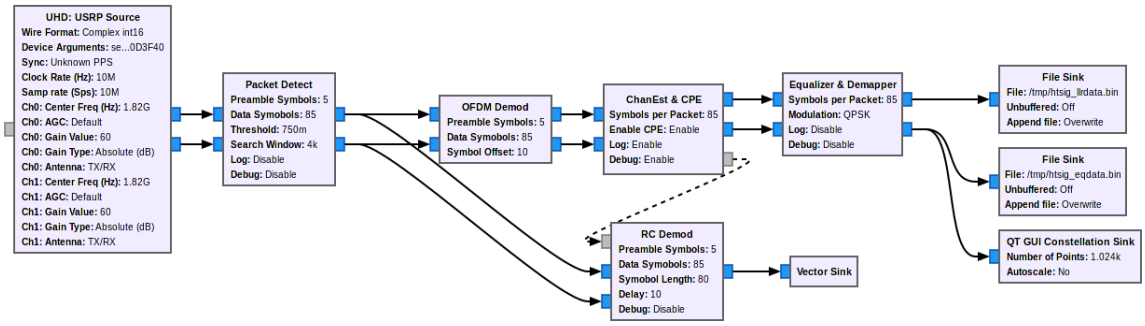


Figure 8. Top-level flow-graph for the MIMO-OFDM receiver implementation using GNU Radio

Table I
SUMMARY OF TEST SCENARIOS

Scenario	Distance	TxGain	RxGain	Description
1	2m	0-30dB	25-55dB	LOS signal present, signal strength is strong.
2	5m	23dB	48dB	LOS signal plus strong reflection paths, signal strength is strong
3	8m	25dB	58dB	No line-of-sign path, some reflection paths, signal strength is medium.
4	9m	27dB	60dB	No line-of-sign path, some reflection paths, signal strength is medium.
5	10m	28dB	60dB	Line-of-sign path blocked by walls and appliance, signal strength is weak.
6	10m	30dB	60dB	Line-of-sign path blocked by walls and appliance, signal strength is weak.
7	12m	30dB	62dB	Single reflection path, weakest signal strength.

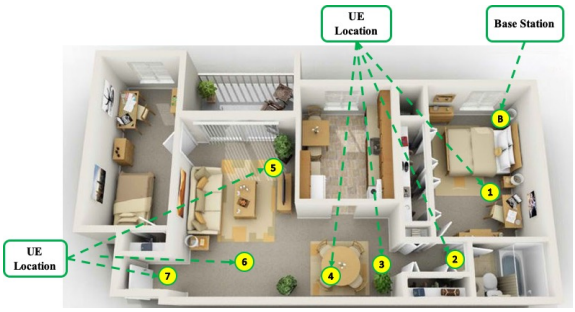


Figure 9. Illustration of field test scenarios

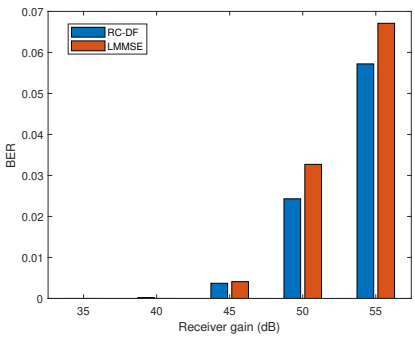


Figure 10. BER performance under non-linearity

VII. CONCLUSIONS

This paper presents the first comprehensive study and findings of a real-time implementation of an RC-based symbol detection scheme for real-world wireless systems. First, conventional algorithms and RC-based methods for detecting symbols in MIMO-OFDM systems are examined and compared. The advantages of the RC-based approach are evaluated by comprehensive simulation studies. Next, the proposed methods are implemented on an SDR platform and validated in various test scenarios.

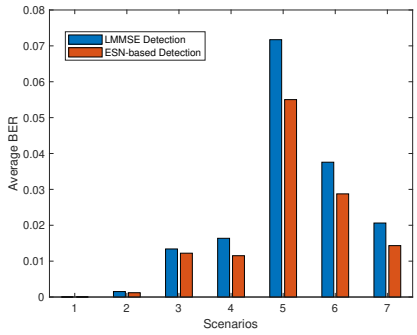


Figure 11. Summary of MIMO-OFDM test results

The real-time implementation of the RC-based symbol detection method is an illustration of the prominent features and advantages of reservoir computing:

- Only the output layer for the reservoir needs training, and the training algorithm is simple, effective, and suitable for real-time wireless systems.
- The rich dynamics of the recurrent reservoir can project inputs onto high dimensional spaces so that even complex MIMO channels can be implicitly compensated for using RC-based neural networks.

- The real-time software and hardware co-design show that RC-based signal processing techniques can be efficiently implemented and scaled to complex scenarios.

In summary, the application of ML-based real-time processing to real-world MIMO-OFDM scenarios in this paper shows that ML-based signal processing can be a promising paradigm for future wireless systems.

ACKNOWLEDGMENT

This work was supported in part by the U.S. National Science Foundation (NSF) under Grants NSF/CCF-1937487 and NSF/CNS-2003059.

REFERENCES

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, 2012.

[2] K. Yun, A. Huyen, and T. Lu, "Deep neural networks for pattern recognition," *arXiv preprint arXiv:1809.09645*, 2018.

[3] V. Passricha and R. K. Aggarwal, "Convolutional neural networks for raw speech recognition," in *From Natural to Artificial Intelligence-Algorithms and Applications*. IntechOpen, 2018.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[8] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[9] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Artificial neural networks-based machine learning for wireless networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3039–3071, 2019.

[10] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 168–179, 2018.

[11] H. Kim, Y. Jiang, R. Rana, S. Kannan, S. Oh, and P. Viswanath, "Communication algorithms via deep learning," *arXiv preprint arXiv:1805.09317*, 2018.

[12] 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.

[13] H.-H. Chang, H. Song, Y. Yi, J. Zhang, H. He, and L. Liu, "Distributive dynamic spectrum access through deep reinforcement learning: A reservoir computing-based approach," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1938–1948, 2018.

[14] H. Ye, G. Y. Li, and B.-H. Juang, "Power of deep learning for channel estimation and signal detection in OFDM systems," *IEEE Wireless Commun. Lett.*, vol. 7, no. 1, pp. 114–117, 2017.

[15] Z. Zhao, M. C. Vuran, F. Guo, and S. Scott, "Deep-waveform: A learned OFDM receiver based on deep complex convolutional networks," *arXiv preprint arXiv:1810.07181*, 2018.

[16] J. Liu, K. Mei, X. Zhang, D. Ma, and J. Wei, "Online extreme learning machine-based channel estimation and equalization for OFDM systems," *IEEE Commun. Lett.*, vol. 23, no. 7, pp. 1276–1279, 2019.

[17] P. Jiang, T. Wang, B. Han, X. Gao, J. Zhang, C.-K. Wen, S. Jin, and G. Y. Li, "Artificial intelligence-aided OFDM receiver: Design and experimental results," *arXiv preprint arXiv:1812.06638*, 2018.

[18] N. Samuel, T. Diskin, and A. Wiesel, "Learning to detect," *IEEE Trans. Signal Process.*, vol. 67, no. 10, pp. 2554–2564, 2019.

[19] M. Khani, M. Alizadeh, J. Hoydis, and P. Fleming, "Adaptive Neural Signal Detection for Massive MIMO," *IEEE Trans. Wireless Commun.*, vol. 19, no. 8, pp. 5635–5648, 2020.

[20] Q. Chen, S. Zhang, S. Xu, and S. Cao, "Efficient MIMO Detection with Imperfect Channel Knowledge-A Deep Learning Approach," in *Wireless Communications and Networking Conference*. IEEE, 2019, pp. 1–6.

[21] S. Mosleh, L. Liu, C. Sahin, Y. R. Zheng, and Y. Yi, "Brain-inspired wireless communications: Where reservoir computing meets MIMO-OFDM," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 10, pp. 4694–4708, 2018.

[22] Z. Zhou, L. Liu, and H.-H. Chang, "Learning for detection: MIMO-OFDM symbol detection through downlink pilots," *IEEE Trans. Wireless Commun.*, vol. 19, no. 6, pp. 3712–3726, 2020.

[23] Z. Zhou, L. Liu, V. Chandrasekhar, J. Zhang, and Y. Yi, "Deep Reservoir Computing Meets 5G MIMO-OFDM Systems in Symbol Detection," in *Proceedings of the AAAI Conf. Artificial Intell.*, vol. 34, no. 01, 2020, pp. 1266–1273.

[24] L. Li, L. Liu, J. C. Zhang, J. D. Ashdown, and Y. Yi, "Reservoir computing meets Wi-Fi in software radios: Neural network-based symbol detection using training sequences and pilots," in *2020 29th Wireless and Optical Communications Conference*. IEEE, 2020, pp. 1–6.

[25] L. Li, L. Liu, Z. Zhou, and Y. Yi, "Reservoir Computing Meets Extreme Learning Machine in Real-Time MIMO-OFDM Receive Processing," *IEEE Trans. Commun.*, pp. 1–1, 2022.

[26] GNU Radio. https://www.gnuradio.org/. GNU Radio project.

[27] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications (Std 802.11-2012)*, IEEE Std., 2016.

[28] J. Joung, C. K. Ho, K. Adachi, and S. Sun, "A survey on power-amplifier-centric techniques for spectrum-and energy-efficient wireless communications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 315–333, 2014.

[29] D. Verstraeten, B. Schrauwen, M. d'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural networks*, vol. 20, no. 3, pp. 391–403, 2007.

[30] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009.

[31] H. Jaeger and H. Haas, "Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, no. 5667, pp. 78–80, 2004.

[32] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, "Recent advances in physical reservoir computing: a review," *Neural Networks*, vol. 115, pp. 100–123, 2019.

[33] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks - with an erratum note," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, no. 34, p. 13, 2001.

[34] ——, "Adaptive nonlinear system identification with echo state networks," in *Advances in Neural Information Processing Systems*, 2003, pp. 609–616.

[35] B. Farhang-Boroujeny, *Adaptive filters: theory and applications*. John Wiley & Sons, 2013.

[36] Y. Naderahmadian, M. A. Tinati, and S. Beheshti, "Generalized adaptive weighted recursive least squares dictionary learning," *Signal Processing*, vol. 118, pp. 89–96, 2016.

[37] S.-C. Horng, "Compensating modeling overlay errors using the weighted least-squares estimation," *IEEE Trans. Semicond. Manuf.*, vol. 27, no. 1, pp. 60–70, 2013.

[38] *Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) Radio Transmission and Reception*, 3GPP Technical specification (TS) 36.101, 2019, version 16.0.0.

[39] *Evolved Universal Terrestrial Radio Access (E-UTRA); Base Station (BS) Radio Transmission and Reception*, 3GPP Technical specification (TS) 36.104, 2019, version 16.0.0.

[40] M. Lukoševičius, "A practical guide to applying echo state networks," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 659–686.

[41] C. Rapp, "Effects of HPA-nonlinearity on a 4-DPSK/OFDM-signal for a digital sound broadcasting signal," *ESASP*, vol. 332, pp. 179–184, 1991.

[42] A. Ghasemmehdi and E. Agrell, "Faster recursions in sphere decoding," *IEEE Trans. Inf. Theory*, vol. 57, no. 6, pp. 3530–3536, 2011.

[43] Vector-Optimized Library of Kernels. https://github.com/gnuradio/volk. GNU Radio project.