AI-Enabled Robust SVD Operator

for Wireless Communication

Task Description

| Prepared by | Sweden Research Center, Algorithm Lab | Date | 2025-07 |
| --- | --- | --- | --- |



华为技术有限公司

Huawei Technologies Co., Ltd.

版权所有 侵权必究

All rights reserved

修订记录Revision record

| 日期  Date | 修订版本Revision version | 修改描述  change Description |
| --- | --- | --- |
| 2025-07-09 | V1.0 | First version |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Introduction

This competition focuses on developing an efficient and robust AI-based SVD operator tailored for wireless channels. For additional information, a supplementary document named as ‘Tech Arena 2025\_Task Background.doc’ is provided. In the following, we outline the organization of the competition and participant expectations in Section 2. In Section 3, we provide a summary of the competition's objectives, along with guidance on how to read the provided data and details on the evaluation criteria.

# Competition Description

The competition consists of two rounds, **both are held online**.

* In the first round, all participating teams will have ***10*** days to develop/submit their solutions. Based on the ranking, the top 15 teams will advance to the second round.
* In the second round, the 15 qualifying teams from the first round will be given ***4*** *days* to compete on a new dataset. Based on the ranking, the top 6 teams will be selected to present in the Swedish Final, where they will present their solutions to the jury. Final rankings will be determined based on the quality of the presentations and the solutions provided.

## Submission requirements

In both rounds, all teams are required to submit:

* **Algorithm output** (with the extension ‘.npz’) to our evaluation platform
* **Neural model definition file** (with the extension ‘.py’)

*which contains the (torch.nn.Module) class which instantiates the submissions neural network. The source file should be standalone and should not depend on anything else other than pytorch and/or numpy.*

* **Trained model parameter file** (with the extension ‘.pth’)

## Important Notes

* **This competition primarily explores the capability of AI algorithms, with the restriction that only neural network-based algorithms are allowed.** The channel dataset incorporates non-ideal factors such as complex Gaussian white noise and timing advance, under which traditional SVD methods perform poorly. Neural networks are expected to learn and achieve more accurate and robust reconstructions.
* **The development language for this competition is limited to Python**, and the design of the network models is restricted to using standard modules, such as conventional architectures defined in torch.nn. The use of complex composite operators or highly correlated operators such as SVD and EVD is strictly prohibited. If you have special requirements, please contact us.
* **To improve the efficiency of data validation in this challenge, all phases will adopt a submission mechanism where participants generate and submit output results independently.** Teams that advance to each phase must also submit their corresponding code, which will be inspected and checked for duplication by competition organizer. If the submitted output results differ from the results reproduced by us or if significant similarities between codes are detected, the behavior will be considered cheating and the team will be disqualified.
* **Each team has 300 submission opportunities in each round, and the best-performing submission will be used for ranking.** Please ensure proper local version control of your code to guarantee that the code corresponding to the best result is saved for future submission to the organizer.
* The competition organizer **reserves the right to the final interpretation of all aspects of this challenge.** Any deficiencies in this document will be updated and corrected in future versions.

# Task Description

This task challenges the accuracy and robustness of SVD-based algorithms for wireless channels. The competition organizer provides non-ideal MIMO channel data collected from multiple sampling locations across various scenarios within a single cell. For some of the data, the corresponding ideal channel labels are also provided. Participating teams are required to design neural network models based on the given data to establish a functional module that maps the input (non-ideal channels at fixed sampling points) to the output (approximated SVD results of the channel).

## Task Summary

As shown in Figure 1, the core objective of this competition is to design a robust SVD model algorithm tailored for wireless channels.

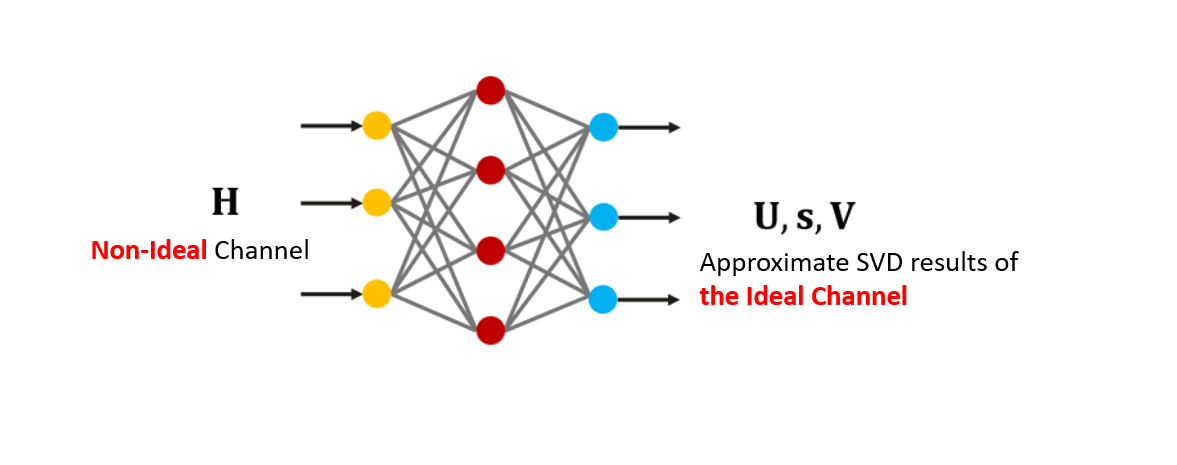


Fig.1 Neural Network-based SVD operator

The task details are as follows:

* **Algorithm Input**

A batch of non-ideal channel matrices, denoted as:

Where and denotes the rows and columns of the channel matrix

**Algorithm Output**

The top largest singular values and their corresponding left and right singular vectors (), including:

* Left singular matrix:
* Singular value vector: , which denotes the diagonal entries of
* Right singular matrix:
* **Algorithm Design**

Participants must design neural network structures and training strategies based on the dataset provided, establishing a functional mapping from input to output.

* **Design Objectives**

The model output should **accurately reconstruct** the ideal channel matrix corresponding to each input, denoted by ,such that:

We can also rewrite this expression as the following

Where , is a diagonal matrix of positive real elements where the diagonal elements are the singular values. The singular values are sorted , and are the columns of (the left vectors) and are the columns of (the right vectors). The output left and right singular matrices must satisfy **orthonormality constraints**, ,

* **Evaluation Metrics**

To comprehensively evaluate both the SVD reconstruction accuracy and orthogonality of singular vectors, we define the **Approximation Error (AE)** as:

Where the denotes the frobenius norm of a matrix. The average AE across all sampling points in a scenario gives the **overall approximation error** for that scenario. Averaging over all scenarios and all points yields the **final algorithm error**. In addition to the SVD approximation error, the **model complexity** is evaluated using the **number of multiply-accumulate operations (MACs)** during model inference, this can be measured by pytorchs built in profiler.

## Data Description

### Data Overview

We will provide 2 sets of CompetitionData, as shown in Table 1:

* **CompetitionData:** the official competition datasets and will be provided in stages.

Table 1: Data Overview List

| Name | Purpose | Remarks |
| --- | --- | --- |
| CompetitionData1 | Used in the first round (to select the top 15). The following documents are provided:  Configuration File:  Round1CfgDataX.txt  Model training file:  Round1TrainDataX.npy  Round1TrainLabelX.npy  Model test file:  Round1TestDataX.npy | For details about the data format, see section 3.2.2 |
| CompetitionData2 | Used in the second round (to select the final 6). The following documents are provided:  Configuration File:  Round2CfgDataX.txt  Model training file:  Round2TrainDataX.npy  Round2TrainLabelX.npy  Model test file:  Round2TestDataX.npy |  |

### Data Format

This section describes the data file formats provided by the organizing committee and the format for saving the algorithm output results. At the same time, we will provide a set of sample code, including functional modules such as data reading and result storage. Read this section and sample code carefully to ensure that the submitted result format is correct. Otherwise, the evaluation may be invalid due to incorrect file format.

#### Provided data format

* RoundYCfgDataX.txt: configuration parameter file for the training dataset for scenario X in round Y. The file contains 5 lines. The following table lists the parameters defined in each line.

Table 2: Content of the RoundYCfgDataX.txt File

| Parameters | Meaning | Example Value (The actual value is subject to the file.) |
| --- | --- | --- |
|  | Total number of sampling points in this scenario | 20000 |
|  | Number of rows in the channel matrix | 64 |
|  | Number of Channel Matrix Columns | 64 |
|  | real and virtual part | 2 |
|  | The goal is to compute the first largest singular value. | 32 |

* RoundYTrainDataX.npy: non-ideal channel training data in the Xth scenario in round Y, which is used as the model training input. The channel data structure is a 4-dimensional complex tensor, and a dimension is . A definition of each dimension is shown in Table 2
* RoundYTrainLabelX.npy: ideal channel training label data corresponding to the Xth scenario in the Yth round. The data structure is the same as that of RoundYTrainDataX.npy.
* RoundYTestDataX.npy: non-ideal channel test data in the Xth scenario in round Y, used as the input for model test. The data structure is the same as that of RoundYTrainDataX.npy (but might be smaller than that of the training data)

#### Submission format

The competing team needs to upload a .zip file which contains the solutions which are named {1-N}.npz, aswell as a python file(extension .py) and the trained weights stored in a .pth file. The content of source file(.py) should contain the code used to both define and train the submitted model. The content of the .pth file should contain the torch.nn.Module.state\_dict(), such that it can be loaded with torch.nn.Module.load\_state\_dict() method. The content of the .npz file should be the following

Table 3: Content of the X.npz File

| Keyword parameter | Meaning | Dimension |
| --- | --- | --- |
| U | Left Singular Matrix Corresponding to Channel Matrix of All Sampling Points |  |
| S | Singular values corresponding to channel matrices at all sampling points, all elements must positive |  |
| V | Right Singular Matrix Corresponding to Channel Matrix of All Sampling Points |  |
| C | A single float denoting the **Mega MACs** of the model, this can be measured by pytorch profiler |  |

The submissions for each found should look like the following

| **Competition Stage** | **Result Name** |
| --- | --- |
| Round 1 | 1.npz  2.npz  3.npz  <anything>.py  <anything>.pth |
| Round 2 | 1.npz  2.npz  3.npz  <anything>.py  <anything>.pth |

# Scoring Metrics

The overall score will be calculated as the sum of average (Approximation Error) and Mega Macs of the model.

For instance, if one team gets 0.9856 for the AE, and 1.4177M macs for complexity, then their Score would be 100\*0.9856 + 1.4177 = 99.9777.

The teams are ranked from lowest to highest.