SAR End-to-End Compression

**Dataset**

We used National Geospatial-intelligence Agency (NGA) dataset to train, test and validate our work. The NGA dataset consists of 8 complex-valued SAR images with two instances of the same scene and four different polarizations (HH, HV, VH, and VV). Our current results are based on the HH polarization only.

The first scene is used for training our network. The second image is used for testing and validating out network.

Link to dataset: <https://umkc.box.com/s/203foyzrx2xt94w69qkujp35k7vab41d>

**Install anaconda environment**

1. Download the anaconda and install in your system.

Link to anaconda downloads: <https://www.anaconda.com/download/success>

2. Create Conda environment

|  |
| --- |
| # Create  conda create --name neurcom python=3.10  # Activate the environment  conda activate neurcom |

3. Install PyTorch (Deep learning framework)

a. For Mac

|  |
| --- |
| conda install pytorch==1.13.1 torchvision==0.14.1 torchaudio==0.13.1 -c pytorch |

b. For Linux and Wondows

|  |
| --- |
| # With GPU  pip install torch==1.13.1+cu117 torchvision==0.14.1+cu117 torchaudio==0.13.1 --extra-index-url https://download.pytorch.org/whl/cu117  # CPU only  pip install torch==1.13.1+cpu torchvision==0.14.1+cpu torchaudio==0.13.1 --extra-index-url https://download.pytorch.org/whl/cpu |

4. Install dependencies.

|  |
| --- |
| Pip install –r requirements.txt |

**Run the Python code**

Go to the folder containing Python Code

PythonDir\SARE2E-Compression

1. Compress the .NITF image

|  |
| --- |
| # Run on GPU  python compress.py --cuda --test\_image <TEST\_IMAGE\_PATH> --test\_model <MODEL\_PATH\_FOR\_DIFF\_LAMBDA> --save\_encoded <BINARY\_OUTPUT\_PATH>  # Run on CPU  python compress.py --test\_image <TEST\_IMAGE\_PATH> --test\_model <MODEL\_PATH\_FOR\_DIFF\_LAMBDA> --save\_encoded <BINARY\_OUTPUT\_PATH> |

The bitstream is saved in the location specified by <BINARY\_OUTPUT\_PATH>.

2. Decompress the bitsream

|  |
| --- |
| # Run on GPU  python decompress.py --cuda --test\_image <TEST\_IMAGE\_PATH> --test\_model <MODEL\_PATH\_FOR\_DIFF\_LAMBDA> --save\_encoded <BINARY\_OUTPUT\_PATH>  # Run on CPU  python decompress.py --test\_image <TEST\_IMAGE\_PATH> --test\_model <MODEL\_PATH\_FOR\_DIFF\_LAMBDA> --save\_encoded <BINARY\_OUTPUT\_PATH> |

NOTE:

1. <TEST\_IMAGE\_PATH> Location where your .NIFT image is saved

For HH polarization use ./sicd\_example\_2\_PFA\_RE32F\_IM32F\_HH.nitf

1. <MODEL\_PATH\_FOR\_DIFF\_LAMBDA> Location where your model is saved

For lambda = 1 use ./PythonDir/neurcom\_ubuntu/checkpoint\_best\_lambda1.pth.tar

For lambda = 4 use ./PythonDir/neurcom\_ubuntu/checkpoint\_best\_lambda4.pth.tar

And so on ...

1. <BINARY\_OUTPUT\_PATH> Location where you want to save\load the binaries

Eg:. ./output or ./PythonDir/neurcom\_ubuntu/output

**Make executable**

Go to the folder containing Python Code

PythonDir\SARE2E-Compression

Open a terminal and execute the following scripts:

1. Install PyInstaller

|  |
| --- |
| pip install pyinstaller |

2. For Ubuntu

|  |
| --- |
| # Executable for compress  pyinstaller --onefile --collect-all torch\_geometric compress.py --windowed  # Executables for decompress  pyinstaller --onefile --collect-all torch\_geometric decompress.py --windowed |

3. For CentOS

|  |
| --- |
| # Executable for compress  pyinstaller –onefile compress.py --windowed  # Executables for decompress  pyinstaller --onefile decompress.py --windowed |

**Run the Executable code**

Go to folder containing executable:

PythonDir\neurcom\_ubuntu

Open a terminal and execute the following scripts:

1. Compress the .NITF image

|  |
| --- |
| # Run on GPU  ./compress --cuda --test\_image <TEST\_IMAGE\_PATH> --test\_model <MODEL\_PATH\_FOR\_DIFF\_LAMBDA> --save\_encoded <BINARY\_OUTPUT\_PATH>  # Run on CPU  ./compress --test\_image <TEST\_IMAGE\_PATH> --test\_model <MODEL\_PATH\_FOR\_DIFF\_LAMBDA> --save\_encoded <BINARY\_OUTPUT\_PATH> |

The bitstream is saved in the location specified by <BINARY\_OUTPUT\_PATH>.

2. Decompress the bitstream

|  |
| --- |
| # Run on GPU  ./decompress --cuda --test\_image <TEST\_IMAGE\_PATH> --test\_model <MODEL\_PATH\_FOR\_DIFF\_LAMBDA> --save\_encoded <BINARY\_OUTPUT\_PATH>  # Run on CPU  ./decompress --test\_image <TEST\_IMAGE\_PATH> --test\_model <MODEL\_PATH\_FOR\_DIFF\_LAMBDA> --save\_encoded <BINARY\_OUTPUT\_PATH> |

NOTE:

1. <TEST\_IMAGE\_PATH> Location where your .NIFT image is saved

For HH polarization use ./sicd\_example\_2\_PFA\_RE32F\_IM32F\_HH.nitf

1. <MODEL\_PATH\_FOR\_DIFF\_LAMBDA> Location where your model is saved

For lambda = 1 use ./PythonDir/neurcom\_ubuntu/checkpoint\_best\_lambda1.pth.tar

For lambda = 4 use ./PythonDir/neurcom\_ubuntu/checkpoint\_best\_lambda4.pth.tar

And so on ...

1. <BINARY\_OUTPUT\_PATH> Location where you want to save\load the binaries

Eg:. ./output or ./PythonDir/neurcom\_ubuntu/output

**Run training dataset generation code**

Go to the folder containing Python Code

PythonDir\SARE2E-Compression

Open a terminal and execute the following scripts:

|  |
| --- |
| python create\_NGA\_dataset.py --data\_root <ROOT\_PATH\_FOR\_DATASET> --samples <TRAIN\_SAMPLES> <VALIDATION\_SAMPLES> <TEST\_SAMPLES> --input\_dir <PATH\_FOR\_INPUTS> --output\_dir <PATH\_FOR\_OUTPUTS> |

**Note:**<ROOT\_PATH\_FOR\_DATASET> Your root directory where dataset or code is stored. If not give it take home directory as data\_root.

<TRAIN\_SAMPLES> Number of samples you need for training

<VALIDATION\_SAMPLES> Number of samples you need for validation

<TEST\_SAMPLES> Number of samples you need for testing

<PATH\_FOR\_INPUTS> Path to your .nitf files

<PATH\_FOR\_OUTPUTS> Path to the directory you want to save your generated dataset for training

**Run training code**

Go to the folder containing Python Code

PythonDir\SARE2E-Compression

Open a terminal and execute the following scripts:

|  |
| --- |
| python train.py --mode train --primary\_pol <POL> --dataset NGA --data\_root <ROOT\_PATH\_FOR\_DATASET> --train\_dataset <PATH\_FOR\_TRAINING\_DATASET> --validation\_dataset <PATH\_FOR\_VALIDATION\_DATASET> --checkpoint <PATH\_FOR\_SAVING\_CHECKPOINT> --lambda <QP> |

<POL> Polarization for the training dataset

<ROOT\_PATH\_FOR\_DATASET> Your root directory where dataset or code is stored. If not give it take home directory as data\_root.

<PATH\_FOR\_TRAINING\_DATASET> Path to the directory where your training dataset is stored

<PATH\_FOR\_VALIDATION\_DATASET> Path to the directory where your training dataset is stored

< PATH\_FOR\_SAVING\_CHECKPOINT > Path to the directory where you want to same the models

<QP> Quality factor for your model. Higher QP means better reconstruction quality