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

2. <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 ...

3. <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:

4. <TEST_IMAGE_PATH> Location where your .NIFT image is saved
   For HH polarization use ./sicd_example_2_PFA_RE32F_IM32F_HH.nitf

5. <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 ...

6. <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