Siren: A toolbox to generate adversarial examples and evaluate defense methods for Automatic Speech Recognition (ASR) Models

Overview of the toolbox

Supported Tasks

1. Speech to Text (STT) via DNN-based ASR Models
2. Generate adversarial audio examples with state-of-the-art attack methods
3. Generate **robust** and **transferable** adversarial audio examples
4. Evaluate the robustness of ASR models under different input transformation methods
5. Evaluate the robustness of different input transformation-based defense methods

Supported ASRs

ASR	Framework	White-Box	Supported	Repo Link	Checkpoints	Datasets	WER
DeepSpeechTF	Tensorflow	yes	yes	Link	DeepSpeech 0.6.1 DeepSpeech 0.5.1	LibriSpeech Clean	7.5% 8.22%
DeepSpeechPT	PyTorch	yes	yes	Link	AN4 Librispeech TED	AN4 Librispeech TEDLIUM	10.349% 6.654% (clean), 19.889% (other) 30.886%

ASR	Framework	White-Box	Supported	Repo Link	Checkpoints	Datasets	WER
Jasper	PyTorch	yes	yes	Link	Jasper10x5dr	Librispeech	3.61%
					Jasper10x5dr	Librispeech, Mozilla Common Voice, WSJ, Fisher, Switchboard	3.46%(dev-clean),10.40%(dev-other),3.69%(test-clean), 10.49%(test-other)
					Jasper10x5SEP	Aishell2	-N/A
					QuartzNet15x5	Librispeech, Mozilla Common Voice	4.19%(test-clean), 10.98%(test-other)
					QuartzNet15x5	Aishell2	-N/A
Wav2letter++	C++	no	Not yet	Link			
PyKaldi	Python	no	Not yet	Link			

Supported Attack Methods

- CarliniWagnerAttack
  Carlini, Nicholas, and David Wagner. "Audio adversarial examples: Targeted attacks on speech-to-text." SPW'18.

- YaoCarliniAttack
  Qin, Yao, et al. "Imperceptible, robust, and targeted adversarial examples for automatic speech recognition." ICML'19

Supported Criteria

- TargetSequence
- SimilarSequence

Supported distance measure

- Mean Square Error (MSE)
- Mean Absolute Error (MAE)
- L Infinity (Linf)
- L0
- Masking Threshold Distance (MTD)

Supported augmentation methods

- Time domain augmentaion methods
  - Add random noise
  - Filter

    - Dynamic filter
    - Quantization
    - Random crop
    - Random insert
    - Random mask
    - Resampling
- Spectrogram domain augmentation methods
    - Add random noise
    - Frequency masking (TODO)
    - Time masking (TODO)
    - Time wrap (TODO)

Speech to Text (STT)

Speech to Text (STT) within Docker

Set up docker environment

- Step 1: Choose the corresponding docker file for the ASRs. We list the docker filename for different ASRs

ASR	Docker Filename
DeepSpeechTF	Dockerfile_deepspeech_tf
DeepSpeechPT	Dockerfile_deepspeech_pt
Jasper	Dockerfile_nemo

All the docker file could be found in the dir 'asr_services/dockerfiles'

- Step 2: Create the docker image from the docker file. We use model `Jasper` as an example. The docker file for `Jasper` is `Dockerfile_nemo`. We create a docker image for `Jasper`

```
docker image build -t [name]:[tag] - <
asr_services/dockerfiles/Dockerfile_nemo
```

  Remarks
    - You can set `[name]:[tag]` as `jasper:v1`.
    - For different ASR models, we need to create different docker image (unless they do not require conflict environments)
- Step 3: Run your own container in detached mode

```
nvidia-docker run --name [your_container_name] --shm-size=1g --ulimit
memlock=-1 -p [port_id]:[port_id] --ulimit stack=67109964 -t -d -v
/home/[your_user_name]:/workspace -it [name]:[tag]
```

  Remarks
    - You can set `your_container_name` as `nemo_container`.
    - The container also map your home directory `/home/[your_user_name]` to the dir `workspace` within the docker.

- If you do not want to set up the ASR model as web service, the option `-p [port_id]:[port_id]` can be omitted. Suppose the port `8888` is not occupied by some other dockers, you can set the port id as `8888`.

Conduct Speech to Text (STT) tasks

Firstly, use exec command to get the shell of your container

```
docker exec -ti [your_container_name] bash
```

Secondly, go the dir /workspace/siren/examples, and trasncribe audio files

- Transcribe a single audio file `sample.wav`

  ```
  python audio_prediction.py
      -in sample.wav
      -asr DeepSpeechTF
      -v 5
      --ckpt_path corresponding_checkpoint_files
      --decoder_name greedy
  ```

  - The option `-in` should be following by a single audio file or a file containing list of audio files.
  - The option `-asr` should be following by the name of the ASR model
  - The options `-v`, `--ckpt_path`, `--decoder_name` are parameters for ASR models. To find out which parameters shuld be provided for corresponding ASR models, please refer to the config file `configs/model_config.yaml`.

  For more details of the audio prediction task, please refer to the file 'audio_prediction.py'.

- Transcribe batch of audio files, you only need to replace `sample.wav` with `audio_list_example.csv`. Here `audio_list_example.csv` is a `csv` file which has at east one column listing all the audio filenames.

Set up Online ASR web service

A more recommened way is to set up the ASR as web service on the docker. In this way, you can query multiple ASR models and do not need to care about their requirement conflicts. Follow below instructions to set up your asr services.

- Make sure your docker container has enabled port id. For example, when setting the docker container, you can add option `-p 6006:6006`.
- Start ASR web service on the docker container. For example, the following command line starts the Jasper model as web service

  ```
  sudo docker exec -ti [your_container_name] python
  /workspace/siren/asr_services/server.py --port 6006 --asr_name Jasper --
  model_config /workspace/siren/checkpoints/quartznet15x5/quartznet15x5.yaml --
  ckpt_dir /workspace/siren/checkpoints/quartznet15x5
  ```

  Remarks

    - To figure out which model parameters should be provided, please refer to as the file `configs/model_config.yaml`
    - To know more details on what services are provided, please refer to the file `asr_services/server.py`
- Inspect the ip addreess of your docker container

```
docker inspect your_container_name |grep IPAddres
```

- Transcribe audios
    - Transcribe from commandline: (Suppose the ip address you get from previous step is 172.17.0.8, and `sample.wav` is in current dir)

```
curl -X POST -F file=@"sample.wav" 172.17.0.8:6006/transcribe
```

    - Transcribe through python scripts

```
python audio_prediction.py
    -in sample.wav
    -asr WebService
    --host 172.17.0.8
    --port 6006
```

      Remarks
        - To conduct STT for a batch of audio files, you only need to replace `sample.wav` with `audio_list_example.csv`. Here `audio_list_example.csv` is a `csv` file which has at east one column listing all the audio filenames.
        - It is recommended to run this python script within a docker container which can be built from the docker file 'asr_services/dockerfiles/Dockerfile_core'

Generate Adversarial Examples

Environment Set Up

- **ASR Models.** We recommend to run all asr models as web service. To set up ASR model as web service, please refer to previous section.
- **Attack tasks.** Run all the attack tasks in the docker container built from the docker file `asr_services/dockerfiles/Dockerfile_core`
    - Build docker images

```
docker image build -t siren:v2 - <
asr_services/dockerfiles/Dockerfile_core
```

    - Run container in detached mode

```
nvidia-docker run --name siren_container -p 127.0.0.1:6008:6008 --shm-
size=1g --ulimit memlock=-1 --ulimit stack=67109964 -t -d -v
/home/xiaowei:/workspace -it siren:v2
```

- Use exec command to get the shell of the container

```
docker exec -ti siren_container zsh
```

- Go the examples dir

```
cd /workspace/siren/examples/
```

Introduction to the command line tool

We provide a python script for user to generate the adversarial examples through the command line easily.

```
usage: adversarial_example_generation.py [-h] -in INPUT_FILE
                                         (-range RANGE [RANGE ...] | -tgt TARGET
[TARGET ...])
                                         [--attack
{CarliniWagnerAttack,YaoCarliniAttack}]
                                        -cfg CONFIG_FILE
                                         [--distance {MSE,MAE,Linf,L0,MTD}]
                                         [--store_dir STORE_DIR]

optional arguments:
  -h, --help            show this help message and exit
  -in INPUT_FILE, --input_file INPUT_FILE
                        Single audio filename (.wav) or the file which
                        contains the list of audios we need to transcribe,
                        their original transcript, and the target transcripts
                        (.csv)
  -range RANGE [RANGE ...]
                        The index range of the audio files we want to generate
                        adv examples
  -tgt TARGET [TARGET ...], --target TARGET [TARGET ...]
                        Target phrase
  --attack {CarliniWagnerAttack,YaoCarliniAttack}
                        Attack methods
  -cfg CONFIG_FILE, --config_file CONFIG_FILE
                        Configuration files of models and augmentation methods
  --distance {MSE,MAE,Linf,L0,MTD}
                        Distance measure between original audio and the
                        manipulated audio
  --store_dir STORE_DIR
                        The dir to store the adv examples
```

The key argument is `-cfg`, which specifies details of ASR models and augmentation methods used for attacks.

Generate adversarial example for a specific ASR model

- Attack config file example

```
models:
        - DeepSpeechTF:
                ckpt_path: '../test/checkpoints/deepspeech-0.6.0-
checkpoint/best_dev-233784'
                v: 6
                decoder_name: 'greedy'

augments:
        - null
```

  - This config file says that the attacker will attack the ASR model `DeepSpeechTF`, and the audio is not augmented
  - The docker container `siren_container` supports loading `DeepSpeechTF` directly

- Attack through command line

```
 python adversarial_example_generation.py \
    -in sample.wav  \
    -tgt ok google \
    -cfg ./attack_cfg_examples/naive_attack_config.yaml \
    --distance MTD \
    --attack YaoCarliniAttack \
    --store_dir ./
```

  - An adversarial example with the target phrase `ok google` would be generated.
  - The adversarial example would be saved in current directory with name `sample_adv.wav`.
  - Feel free to try different combinations of `distance` and `attack`

- Generate adversarial examples for batch of audios

```
python adversarial_example_generation.py \
    -in audio_list_examples.csv  \
    -range 0 10\
    -cfg ./attack_cfg_examples/naive_attack_config.yaml \
    --distance MTD \
    --attack YaoCarliniAttack \
    --store_dir ./
```

  - This command generates adversarial examples for the 0-th to the 10th audios in the file `audio_list_examples.csv` simulatenously.
  - The `csv` files should have fields `audio_name`, `audio_len`, and `target`

Generate Transferable Adversarial Examples

To generate transferable AEs, we need to attack multiple ASR models simutaneously. To run the attack, we just need to provide a parameter for `-cfg`.

```
models:
        - WebService:
                host: '172.17.0.8'
                port: 6006
```

```
        - DeepSpeechTF:
                ckpt_path: '../test/checkpoints/deepspeech-0.6.0-
checkpoint/best_dev-233784'
                v: 6
                decoder_name: 'greedy'

augments:
        - null
        - filter:
                filter_name: 'moving_average'
                window_size: 11
```

- This config file says that the attacker will attack the ASR model `DeepSpeechTF` and another ASR hold as web service (ip: 172.17.0.8, port:6006), and the audio is not augmented

Generate Robust Adversarial Examples

Some augmentation methods simulate the real-world environment. For example, the `phone effect` filter simulates the scenario where people speak over the phone. The `room reverberation` filter simulates the scenario where people talk in a room.

To generate robust adversarial examples, we augment the audios with different augmentation methods to simulate the distortion caused by real-world environments, and the AEs generated by the attack should be able to fool the ASR w/o the augmentation methods. An example attack config file is as follows.

```
models:
        - WebService:
                host: '172.17.0.8'
                port: 6006
augments:
        - null
        - filter:
                filter_name: 'moving_average'
                window_size: 11
```

- The config file says that the attack will try to fool the ASR hosted by web service w/o augment method `moving average filter`

Generate Robust and Transferable Adversarial Examples

An example config file is as follows.

```
models:
        - WebService:
                host: '172.17.0.8'
                port: 6006
        - DeepSpeechTF:
                ckpt_path: '../test/checkpoints/deepspeech-0.6.0-
checkpoint/best_dev-233784'
                v: 6
                decoder_name: 'greedy'
```

```
augments:
        - null
        - filter:
                filter_name: 'moving_average'
                window_size: 11
```

- Attack two ASR models: Web service hosted model, and DeepSpeechTF model
- Fool the models w/o filter `moving average`

Contributors

- Xiaowei Chen