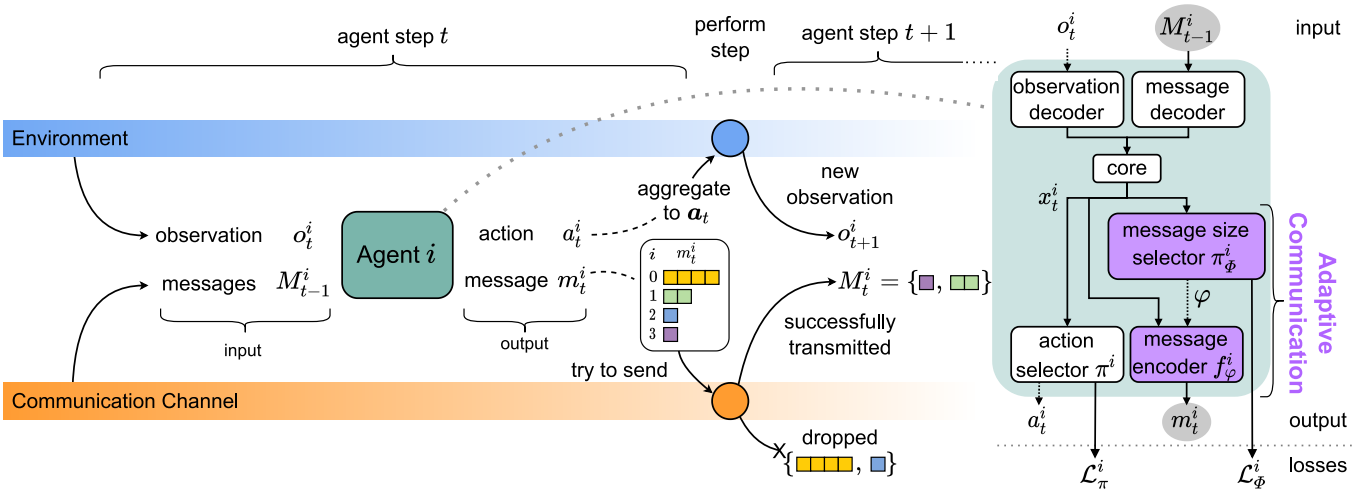
# Learning to Cooperate and Communicate Over Imperfect Channels

This is the official implementation of *Learning to Cooperate and Communicate Over Imperfect Channels*.

We investigate unreliable communication in multi-agent reinforcement learning and propose an adaptive message size selection that enables agents to use a limited (and unreliable) communication channel more efficiently.




Submitted as supplementary material to the Eleventh International Conference on Learning Representations (ICLR 2023). Do not distribute.

*The code is provided "as is", further refactoring and cleanup may be conducted before official release.*

## Requirements

This project uses Python and PyTorch. We recommend working with a virtual python environment and use miniconda in the following.

1. Create an environment for the project and activate it

```
conda create -n myenv python=3.8
conda activate myenv
```

2. [Optional] Install PyTorch with GPU support, see https://pytorch.org/get-started/locally/

3. Install the remaining requirements

```
pip install -r requirements.txt
```

4. [Windows] The traffic junction environment depends on the standard Python curses module. Windows OS users can install it as follows

```
pip install windows-curses
```

5. Set the `PYTHONPATH` environment variable to the root of this project

## Experiments

To train and evaluate the models used in the paper, we provide the following experiment configurations in `experiments/configs`:

- `paper_pomnist_c8_adaptation.yaml`: Adaptive Communication in a channel size of 8 (main results).
- `paper_pomnist_c8_random_size.yaml`: Ablation with random message size selection.
- `paper_pomnist_c8_zero_content.yaml`: Ablation with message size selection but zero content.
- `paper_pomnist_c_sizes.yaml`: Adaptive communication with different channel sizes.
- `paper_pomnist_baseline_num_agents.yaml`: POMNIST without communication and different environment splits (see appendix).
- `paper_pomnist_baseline_single_msg_sizes.yaml`: POMNIST with (1, 1) splits and different message sizes (see appendix).
- `paper_tj_adaptive.yaml`: Adaptive communication (with/without random selection) in the traffic junction environment over an imperfect channel.
- `paper_tj_baseline.yaml`: Traffic junction baseline with perfect channel.
- `paper_tj_fixed`: Traffic junction with fixed message sizes in an imperfect channel.

Selected experiments can be run with

```
python experiments/run_experiment.py --config-
file=experiments/configs/selected_config.yaml
```

This will start the runs for this configuration. All models are evaluated directly after training.

The following results will be put into a new subdirectory of `./runs`:

- Tensorboard training logs for the individual runs, updated during training
- Pickle dumps with statistics for each training step and the evaluation
- A plot of selected aggregated metrics (mean return by default)
- A `metrics_table.csv` file that contains the aggregated metrics for all runs

## Manual Training and Evaluation

Manual experiments can be performed by modifying and running `main.py`:

```
python main.py
```

Example configurations are located at the bottom of the script.

## Results

We include the aggregated evaluation results used in the paper in the `results` directory.

The evaluation figures from the paper can be recreated by running

```
python experiments/create_pomnist_paper_figs.py
```

## Contributors

Anonymous authors.