调用方式

可以在外部指定某个参数，来覆盖原来的参数，从而避免修改原参数

For example, if you want to run robustness verification on the ResNet-2B model, changing epsilon from 2/255 (specified in config file) to 1/255:

python robustness\_verifier.py --config exp\_configs/cifar\_resnet\_2b.yaml --epsilon 0.00392156862745098

general:

device: cuda # Select device to run verifier, cpu or cuda (GPU).

seed: 100 # Random seed.

conv\_mode: patches # Convolution mode during bound propagation: "patches" mode (default) is very efficient, but may not support all architecture; "matrix" mode is slow but supports all architectures.

deterministic: false # Run code in CUDA deterministic mode, which has slower performance but better reproducibility.

double\_fp: false # Use double precision floating point. GPUs with good double precision support are preferable (NVIDIA P100, V100, A100; AMD Radeon Instinc MI50, MI100).

loss\_reduction\_func: sum # When batch size is not 1, this reduction function is applied to reduce the bounds into a single number (options are "sum" and "min").

mode: verified-acc # Verify against all labels ("verified-acc" mode), or just the runnerup labels ("runnerup" mode), or using a specified label in dataset ("speicify-target" mode, only used for oval20). Mode can also be set as "crown-only-verified-acc" or "alpha-crown-only-verified-acc", which quickly computes the verified accuracy over the entire dataset via CROWN or alpha-CROWN.

complete\_verifier: bab # Complete verification verifier. "bab": branch and bound with beta-CROWN; "mip": mixed integer programming (MIP) formulation; "bab-refine": branch and bound with intermediate layer bounds computed by MIP.

enable\_incomplete\_verification: true # Enable/Disable initial alpha-CROWN incomplete verification (this can save GPU memory when disabled).

model:

path: null # Load pretrained model from this specified path.

name: please\_specify\_model\_name # Name of model. Model must be defined in the load\_verification\_dataset() function in utils.py.

data:

start: 0 # Start from the i-th property in specified dataset.

end: 10000 # End with the (i-1)-th property in the dataset.

num\_outputs: 10 # Number of classes for classification problem.

mean: 0.0 # Mean vector used in data preprocessing.

std: 1.0 # Std vector used in data preprocessing.

pkl\_path: null # Load properties to verify from a .pkl file (only used for oval20 dataset).

dataset: CIFAR # Dataset name. Dataset must be defined in utils.py.

data\_idx\_file: null # A text file with a list of example IDs to run.

specification:

type: lp # Type of verification specification. "lp" = L\_p norm, "bounds" = element-wise lower and upper bound provided by dataloader.

norm: .inf # Lp-norm for epsilon perturbation in robustness verification (1, 2, inf).

epsilon: null # Set perturbation size (Lp norm). If not set, a default value may be used based on dataset loader.

solver:

alpha-crown:

lr\_alpha: 0.1 # Learning rate for the optimizable parameter alpha in alpha-CROWN bound.

iteration: 100 # Number of iterations for alpha-CROWN incomplete verifier.

share\_slopes: false # Share some alpha variables to save memory at the cost of slightly looser bounds.

no\_joint\_opt: false # Run alpha-CROWN bounds without joint optimization (only optimize alpha for the last layer bound).

beta-crown:

batch\_size: 64 # Batch size in beta-CROWN (number of parallel splits).

lr\_alpha: 0.01 # Learning rate for optimizing alpha during branch and bound.

lr\_beta: 0.05 # Learning rate for optimizing beta during branch and bound.

lr\_decay: 0.98 # Learning rate decay factor during optimization. Need to use a larger value like 0.99 or 0.995 when you increase the number of iterations.

optimizer: adam # Optimizer used for alpha and beta optimization.

iteration: 50 # Number of iteration for optimizing alpha and beta during branch and bound.

mip:

parallel\_solvers: null # Number of multi-processes for mip solver. Each process computes a mip bound for an intermediate neuron. Default (None) is to auto detect the number of CPU cores (note that each process may use multiple threads, see the next option).

solver\_threads: 1 # Number of threads for echo mip solver process (default is to use 1 thread for each solver process).

refine\_neuron\_timeout: 15 # MIP timeout threshold for improving each intermediate layer bound (in seconds).

refine\_neuron\_time\_percentage: 0.8 # Percentage (x100%) of time used for improving all intermediate layer bounds using mip. Default to be 0.8\*timeout.

bab:

max\_domains: 200000 # Max number of subproblems in branch and bound.

decision\_thresh: 0 # Decision threshold of lower bounds. When lower bounds are greater than this value, verification is successful. Set to 0 for robustness verification.

timeout: 360 # Timeout (in second) for verifying one image/property.

branching:

method: kfsb # Branching heuristic. babsr is fast but less accurate; fsb is slow but most accurate; kfsb is usualy a balance.

candidates: 3 # Number of candidates to consider when using fsb or kfsb. More leads to slower but better branching.

reduceop: min # Reduction operation to compute branching scores from two sides of a branch (min or max). max can work better on some models.

attack:

pgd\_order: before # Run PGD before/after incomplete verification, or skip it.

enable\_mip\_attack: false # Use MIP (Gurobi) based attack if PGD cannot find a successful adversarial example.

pgd\_steps: 100 # Steps of PGD attack.

pgd\_restarts: 30 # Number of random PGD restarts.

pgd\_early\_stop: true # Early stop PGD when an adversarial example is found.

pgd\_lr\_decay: 0.99 # Learning rate decay factor used in PGD attack.

pgd\_alpha: auto # Step size of PGD attack. Default (auto) is epsilon/4.