Effective Sparsity

This repository contains code for the experimental section of the paper "Connectivity Matters: Neural Network Pruning Through the Lens of Effective Sparsity".

Environment

To run code from this project, first clone the repository and install dependencies by executing `pip install -r requirements.txt`. The code was tested for Python-3.6 and Python-3.8. For GPU support (not available for macOS), add `tensorflow-gpu===2.5.0` to the `requirements.txt` file.

Quick demonstration

Our original full-fledged experiments involved 5 different network architectures (3 of which require a GPU) and 15 pruning algorithms combined across about 30 sparsity levels and repeated 3 times for stability of results. In total, this comes to almost 7,000 networks to be trained. For demonstration purposes, we provide a quick lightweight demonstration `demo.py` that requires no flags or arguments and takes only a few minutes to execute on a CPU-powered device. Run `python demo.py` for this demo.

Original experiments

To replicate our results, run `python main.py` with a selection of arguments (run `python main.py --help` for a description). For example, `python main.py --architecture=lenet300100 --pruner=snip --target_sparsity=0.99 --pruning_type=effective --train=1` will use SNIP to prune LeNet-300-100 to 99% effective sparsity and train this subnetwork on MNIST. Provided `--save=1` (default), all output and log files are saved to `{--out_path}/{--architecture}/{--pruner}/{--pruning_type}` in the following format: `{--sample}_{compression}_{description_of_output}.{ext}`, where `.{ext}` can be either `.log` (as in `info.log`) or `.npy` for actual output files (e.g., `0_10_accuracies.npy`). `{compression}` value (direct or effective, depending on `--pruning_type`) is calculated using `--target_sparsity` as `1/(1-target_sparsity)` or `--com_exp` as `10^(com_exp)`. The latter flag overrides `--target_sparsity` if specified. **Note:** LAMP and all magnitude-based pruners require output files (final weights) generated after training an architecture with `--pruner=dense`.

Datasets

To use ResNet-18 on TinyImageNet, download the dataset by executing the following commands:

```
wget http://cs231n.stanford.edu/tiny-imagenet-200.zip
unzip tiny-imagenet-200.zip
```

and pass the path to this folder using `--path_to_data`. MNIST and CIFAR-10/100 will be downloaded automatically when they are first needed (no need to specify `--path_to_data` for these three datasets). **Important:** avoid multiple scripts downloading the same dataset concurrently as this may result in damaged data files.

Visualizing results

The `visualization.py` file produces plots out of output files generated in the above step. Execute `python visualization.py --help` for the full list of flags and their description. To plot accuracies for LeNet-300-100 effectively pruned by SNIP, SynFlow, and IGQ (random) averaged across 3 samples, run `python visualization.py --architecture=lenet300100 --pruners_to_display snip synflow random/igq --num_samples=3 --pruning_type=effective --plot_type=accuracies`. Plots are saved to `{--out_path}/{--architecture}/figures`.