Less is More-- Group Sparsity for Parsimonious Multi Task Models

Abstract

Group sparsity in Machine Learning (ML) encourages simpler, more interpretable models with fewer active parameter groups. This work aims to incorporate structured group sparsity into the shared parameters of a Multi-Task Learning (MTL) framework, to develop parsimonious models that can effectively address multiple tasks with fewer parameters while maintaining comparable or superior performance to a dense model. Sparsifying the model during training helps decrease the model's memory footprint, computation requirements, and prediction time during inference. We use channel-wise l1/l2 group sparsity in the shared layers of the Convolutional Neural Network (CNN). This approach not only facilitates the elimination of extraneous groups (channels) but also imposes a penalty on the weights, thereby enhancing the learning of all tasks. We compare the outcomes of single-task and multi-task experiments under group sparsity on two publicly available MTL datasets, NYU-v2 and CelebAMask-HQ. We also investigate how changing the sparsification degree impacts both the performance of the model and the sparsity of groups.

Conda enviornment

Install the required python packages in a conda enviornment using the requirement.yml file, by using the following command:

conda env create -f requirement.yml

Note: wandb is used for logging the results of all the experiments. Remember to enter the correct entity name (yours) and experiment name in the main files before starting to train the models.

Dataset

In this work two publicly available datasets are used -

- NYU-v2 dataset -- the formatted multi-task version of the dataset can be downloaded from link
- CelebAMask-HQ dataset -- the formatted version can be found here.

The actual CelebAMask-HQ dataset can be found here, for the semantic segmentation task in this work the dataset was formatted for only 3 segmentation classes. The classification tasks are extracted from the attribute annotations.

Experiment configuration files

For analysing the performance of the proposed method which includes introducing group sparsity in multi-task learning setting, a wide range of experiments were conducted. Using the config files both single task and multi-task experiments (with and without sparsity) can be designed for both NYU and celebA dataset.

- For NYU dataset - config/exp_single_mtl_NYU.yaml,
- For celebA dataset - config/exp_single_multi_celebA.yaml

Remember to change the setup (singletask or multitask), group_sparsity (True or False) accordingly. The value of regularization parameter (sparsity strength) lambda can be changed as required under the 'BACKBONE OPTIMIZER PARAMETERS' i.e., {bb_optimizer_params}{lambda} The num_trials is the number of times you want to train each model. In this work every task is trained 5 times. Use the config files as required for changing the learning rates, task names, batch sizes, etc.

Training

After setting all the parameters as required in the config file, for training the models use the main_multi_trials.py files. For single task and multi-task training --

python main_multi_trials.py --config_exp config/exp_single_mtl_NYU.yaml (for NYU-v2 dataset)

python main_multi_trials.py --config_exp config/exp_single_multi_celebA.yaml (for celebA dataset)

Inference

Few of the trained models can be found here. As is it very diffucult to share all the trained models only are few of them are shared for reproducibility, rest of the models can be requested (from the suthors) when required.

To evaluate the models --

For celebA dataset : python get_out_put_celeb.py --exp_name 8_2_multi_seg_male_smile_ --num_trials 5

For NYU dataset : python get_outputs.py --exp_name 8_2_multi_seg_sn_depth_ --num_trials 5

After exp_name add the name of the experiment or the folder where the model is saved. Note - remember to change the path in the main function of the files.

All the results are saved in a csv file in the out_csv folder

Sparsity plots

To plot the sparsity plots of any trained model -

python plot_sparsity_wandb.py --exp_folder_path give_path_to_the folder_where_model_is_saved

Group sparsity

The concept of group sparsity is included in the optimizer (ADAM), this is adapted (and modified accordingly) from the article "Structured Sparsity Inducing Adaptive Optimizers for Deep Learning Tristan". The git of this work can be found here