GroupNorm

GroupNorm implementation for the DeepVision subject

Installations

- python 3.8.10
- TensorFlow 2.4.1
- jupyterlab 3.0.16
- matplotlib 3.4.2
- tensorflow-datasets
  - ipywidgets
- SGDW from TensorFlow Addons
- tqdm

With the environment yml a conda environment can be created

```
conda env create [--name envname] -f utils/environment.yml
conda activate [envname or GN]
```

Visualizing the experiments

The **Implementation.ipynb** JupyterLab notebook contains the visualization of the experiments.

Running the experiments

To run the experiments the python **run_experiment.py** file is provided with the following arguments.

```
python run_experiment.py -h
usage: run_experiment.py [-h] [--replace] [--continue] [--cont_epoch CONT_EPOCH] [--experimental_data_aug] [-

Configurations to run normalization experiments with ResNet

optional arguments:
  -h, --help                  show this help message and exit
  --replace                   overwrite previous run if it exists
  --continue                  continue training: load the latest checkpoint and continue training
  --cont_epoch CONT_EPOCH     Used together with continue, overwrites the saved epoch of the checkpoint and s
  --experimental_data_aug     Use experimental data augmentation inside the network instead of the one before
  --cpu                       train on only cpu
  --ResNet {3,5}              Defines what Resnet model to use, 3-> ResNet 20, 5-> ResNet 32
  --batch_size {32,16,8,4,2}  batch size per worker
  --epochs {30,100}           training epochs
  --norm {BN,GN}              decide if BN (batch normalization) or GN (group normalization is applied)
  --run RUN                   Differentiate multiple runs for statistical comparison
  --weight_decay              Set to use SGDW (stochastic gradient with weight decay) as optimizer and use we
```

--experimental_data_aug use experimental keras layers for data augmentation, instead of the .map() function on the tfds dataset.

For example, for the first run of batch size 2 for both Group normalization:

```
python run_experiment.py --batch_size 2 --norm BN --run 1
```

for Batch normalization:

```
python run_experiment.py --batch_size 2 --norm GN --run 1
```

This script trains a ResNet 20 network on the CIFAR-10 dataset for 30 epochs using data augmentation, lr decay and saves the weights of the best epoch, a csv log per epoch, and a tensorboard log.

For more details of the network see the Implementation JupyterLab notebook.

Folder structure

The **utils** folder contains:

- **train_utils.py** containing helper functions for the training, mostly arguments parsing and creating the folder structure
- **plot_utils.py** contains the function needed for visualizing the results.
- **ResNet_network.py** contains the classes and function to create and train the ResNet 20 network.
- **environment.yml** is the export of the conda environment used

The **train_outputs** folder contains:

- **checkpoints** contains the best checkpoints of the trained models. When training the optimizer is also saved to enable restarting the training from the last saved checkpoint (used to continue training after system crashes). The optimizer weights are not submitted since for evaluating the networks there are not needed and would exceed the 250 Mb limit.
- **logs** contains the training logs. When training the network the tensorboard callback, and a CSV logger callback are used. Submitted here are only the CSV files.

The **images** folder contains the visualization of the ResNet 20 network.