PyTorch X Shape Bias

An implementation of adding biologically inspired shape bias to popular data sets, exploring the effects on common CNN architectures.

Instructions

The code is intended to run in Google Colab (Colab), although can be configured if you have your own GPU.

Open a Colab notebook (or alternatively see the link for a set up - open in playground to run, may need to connect to GPU [step 1]: https://colab.research.google.com/drive/1UWgWIGu0RCuGbANeQPRXZ_Ta4wYKlmZq)

There are a few key steps to get up and running in the Colab environment.

1. Connect to hosted GPU. (runtime -> change runtime type -> Hardware -> GPU)
2. Download the repository/source code (using git clone preferably)
3. Install requirements in the cloned repository
4. Choose dataset, architecture, transformations and other hyperparameters you wish to run & run command

Example of set up in Colab:

```
# Note: '!' in Colab is to run command line
# Clone Repository
!git clone https://github.com/Mikcl/ShapeBias.git
# Change Directory
%cd ./ShapeBias/
# Install Requirements
!pip install -r requirements.txt
# Run Experimenent - train a VGG-16 model on dataset 0 (CIFAR-10)
!python main.py -a vgg-16 --dataset 0
```

Usage

Ensure all requirements are installed before running experiments: `pip install -r requirements.txt`

```
usage: main.py [-h] [-a ARCH] [-j N] [--epochs N] [--start-epoch N] [-b N]
               [--lr LR] [--momentum M] [--wd W] [-p N] [-e] [--pretrained]
               [--gpu GPU] [--own PATH] [-f] [-l LF] [-c CHANNELS] [--decay
D]
               [-d DATASET] [--data DATA] [--concat] [--same] [--DOG]
               [-o O [O ...]] [--gabor] [-s S [S ...]] [-u U [U ...]]
               [--savecsv] [--savemodel]

Shape Bias Training
```

```
optional arguments:
  -h, --help            show this help message and exit
  -a ARCH, --arch ARCH  model architecture: alexnet | densenet121 |
                        densenet161 | densenet169 | densenet201 | googlenet
|
                        inception_v3 | mnasnet0_5 | mnasnet0_75 |
mnasnet1_0 |
                        mnasnet1_3 | mobilenet_v2 | resnet101 | resnet152 |
                        resnet18 | resnet34 | resnet50 | resnext101_32x8d |
                        resnext50_32x4d | shufflenet_v2_x0_5 |
                        shufflenet_v2_x1_0 | shufflenet_v2_x1_5 |
                        shufflenet_v2_x2_0 | squeezenet1_0 | squeezenet1_1
|
                        vgg11 | vgg11_bn | vgg13 | vgg13_bn | vgg16 |
vgg16_bn
                        | vgg19 | vgg19_bn | wide_resnet101_2 |
                        wide_resnet50_2 (default: resnet18)
  -j N, --workers N     number of data loading workers (default: 4)
  --epochs N            number of total epochs to run
  --start-epoch N       manual epoch number (useful on restarts)
  -b N, --batch-size N  mini-batch size (default: 256), this is the total
                        batch size of all GPUs on the current node when
using
                        Data Parallel or Distributed Data Parallel
  --lr LR, --learning-rate LR
                        initial learning rate
  --momentum M          momentum
  --wd W, --weight-decay W
                        weight decay (default: 1e-4)
  -p N, --print-freq N  print frequency (default: 10)
  -e, --evaluate        evaluate model on validation set
  --pretrained          use pre-trained model
  --gpu GPU             GPU id to use.
  --own PATH            path to your own given model state dict (note arch
of
                        this model must match -a)
  -f, --finetune        fine-tune model fc layer on training set
  -l LF, --loadfinetuned LF
                        use fintuned own model with defined number of
output
                        classes
  -c CHANNELS, --channels CHANNELS
                        number of channels (default: 3)
  --decay D             decay learning rate every D epochs by factor of 10
  -d DATASET, --dataset DATASET
                        which dataset to train on, default- None (define
                        custom path to directory), 0 - CIFAR10, 1 -
CIFAR100
  --data DATA           path to custom dataset and where output folder is
  --concat              concat transformed data with original data
  --same                train and validated on same (type of
transformation)
                        dataset, primarily used from custom transformations
```

```
  --DOG                 DOG transformation, use --options for non default
                        hyper parameters, [sigma k]
  -o O [O ...], --options O [O ...]
                        options (list) for the transformation, pass as: -o
s k
  --gabor               gabor 2D CWT
  -s S [S ...], --scales S [S ...]
                        scales (list) for the 2D Gabor Wavelet: -s 2 2.5
  -u U [U ...], --orientations U [U ...]
                        orientations (list) for the 2D Gabor Wavelet: -u 1
2 3
  --savecsv             save the csv to google drive [only in collab]
  --savemodel           save the model to google drive [only in collab]
```

Datasets

Can perform experiments on various datasets.

Custom

To utilise a custom dataset, use `--data ./path/to/dataset`, where `.../dataset/` contains the directories `/dataset/train` and `/dataset/val` where images are grouped in directories (representing their class name).

Example:

```
...dataset/

...dataset/train
...dataset/train/dog/
...dataset/train/dog/bulldog.jpeg
...dataset/train/dog/poodle.jpeg
...dataset/train/car/
...dataset/train/car/hatchback.jpeg
...dataset/train/car/jeep.jpeg
...

...dataset/val
...dataset/val/dog/
...dataset/val/dog/boxer.jpeg
...dataset/val/car/
...dataset/val/car/limousine.jpeg
...
```

To download and format the Tiny ImageNet dataset, run `./scripts/prep.sh`

Torch Vision

Support for CIFAR-10 and CIFAR-100 have been added thus far, which can be utilised by `--dataset N`

Data set	N
CIFAR-10	0
CIFAR-100	1

Examples

- **Train** CNN vgg-16 (learning rate=0.01, weight_decay=0.0005) on CIFAR-10 and validate on CIFAR-10

  ```
  python main.py -a vgg-16 --lr 0.01 --wd 0.0005 --dataset 0
  ```

- Train for 75 **epochs** and reduce the **learning rate** by a factor of 10 every 30 epochs

  ```
  python main.py --dataset 0 --decay 30 --epochs 75
  ```

- Train on CIFAR-10 **and** CIFAR-10 **transformed** by a Difference Of Gaussian and validate on CIFAR-10, and **save** the model into Google Drive with `--savemodel`

  ```
  python main.py --dataset 0 --concat --DOG --savemodel
  ```

- Train on **only** CIFAR-10 **transformed** by a 2D Gabor continuous wavelet transform (2D-CWT) and validate **only** on the same transform applied to CIFAR-10 validation, hyperparameter of orientation `-u 2` is used for the wavelet.

  (note: `-u` and `-s` can take multiple orientations and scales respectively for the `--gabor` 2D Wavelet)

  ```
  python main.py --dataset 0 --same --gabor -u 2
  ```

- **Evaluate** a model `./path/to/model.pth.tar` on a CIFAR-100 dataset.

  (note: should be trained to have 100 classes as output)

  ```
  python main.py --own ./path/to/model.pth.tar --dataset 1 -e
  ```

- **Finetune** a model `./path/to/model.pth.tar` on a CIFAR-100 dataset

  ```
  python main.py --own ./path/to/model.pth.tar --dataset 1 -f
  ```

- **Evaluate** a **finetuned** model `./path/finetuned/model.pth.tar` on a CIFAR-100 dataset. Pass 100 because 100 classes.

  ```
  python main.py --own ./path/finetuned/model.pth.tar --dataset 1 -l 100 -e
  ```

- **Train** on a **custom** dataset `./path/to/dataset/`

  ```
  python main.py --data ./path/to/dataset/
  ```

Inspired by PyTorch Image Net Example

Please see Image Net Example for details on Multi-processing Distributed Data Parallel Training if you wish to extend functionality

https://github.com/pytorch/examples/

Additional Notes

When attempting to fine tune models, you may find that you will need import the model saved into the colab environment. With large model files this will take considerable time to 'Upload' the file directly to the Colab environment. A more efficient way to do this is (in Colab) to Mount your google drive, from the files section. Then run the script below inside `./ShapeBias/` to retrieve `model.pth.tar` from your Drive.

```
!cp ../drive/My\ Drive/model.pth.tar ./
```

Which will copy the model into your working directory very quickly. Files inside your drive can be found at drive.google.com