Revisiting SVD for Global Covariance Pooling: Halley's Method to Overcome Over-Flattening

1 Code

Code can be available in https://anonymous.4open.science/r/HalleySVD-F355/README.md

2 Overview

This repository provides a PyTorch implementation of differentiable Singular Value Decomposition (SVD) using Halley's iteration method, specifically designed for Global Covariance Pooling (GCP) in deep learning models for visual recognition tasks.

Halley-SVD is an efficient method for computing differentiable SVD that employs Halley's iteration for matrix square root computation. This approach provides stable gradients and improved convergence compared to traditional SVD implementations in deep learning frameworks.

3 Key Features

- **Halley's Iteration**: Fast convergence for matrix square root computation
- **Differentiable SVD**: Stable gradient computation for end-to-end training
- **Global Covariance Pooling**: Enhanced feature representation for visual recognition
- **Multiple Architecture Support**: Compatible with ResNet, AlexNet, and other popular architectures

4 Installation

4.1 Clone the Repository

```
git clone <repository-url>
cd DifferentiableSVD-main
```

4.2 Install Dependencies

```
pip install -r requirements.txt
```

Required packages:

- PyTorch (with CUDA support recommended)
- Torchvision
- NumPy

- SciPy
- Matplotlib

5 Dataset Preparation

Supported datasets:

- ImageNet (ILSVRC2012)
- CUB-200-2011 (Birds)
- Stanford Dogs
- FGVC Aircraft
- Stanford Cars

Organize your datasets following the standard ImageFolder structure:

```
dataset/
|-- train/
|   |-- class1/
|   |-- class2/
|   `-- ...
`-- val/
    |-- class1/
    |-- class2/
    `-- ...
```

6 Running Halley-SVD

6.1 Basic Training

Train a ResNet-50 model with Halley-SVD on ImageNet:

```
python main.py /path/to/imagenet \
    --arch resnet50 \
    --representation SVD_Halley \
    --batch-size 256 \
    --lr 0.1 \
    --epochs 100 \
    --workers 32 \
    --modeldir ./models/resnet50_halleySVD \
    --num-classes 1000 \
    --benchmark ILSVRC2012
```

6.2 Key Parameters

- `-arch`: Model architecture (`resnet50`, `resnet101`, `alexnet`, etc.)
- `-representation`: Set to `SVD_Halley` for Halley-SVD method
- `-batch-size`: Training batch size (adjust based on GPU memory)
- `-lr`: Initial learning rate

- `-epochs`: Number of training epochs
- `-modeldir`: Directory to save checkpoints and logs
- `-num-classes`: Number of output classes
- `-benchmark`: Dataset name

6.3 Halley-SVD Specific Settings

The default Halley-SVD parameters:

- **Iterations (K)**: 8
- **Initial matrix**: $X_0 = 10^{-3}I$

These parameters are optimized for stability and performance across various tasks.

6.4 Fine-tuning for Fine-Grained Visual Classification

For fine-grained datasets, use pretrained ImageNet weights and adjust learning rates:

```
python main.py /path/to/CUB_200_2011 \
    --arch resnet50 \
    --representation SVD_Halley \
    --batch-size 64 \
    --lr 0.001 \
    --epochs 100 \
    --workers 16 \
    --modeldir ./models/resnet50_halleySVD_cub \
    --num-classes 200 \
    --benchmark CUB_200_2011 \
    --pretrained \
    --classifier-factor 10
```

Key fine-tuning parameters:

- `-pretrained`: Use ImageNet pretrained backbone
- `-classifier-factor`: Learning rate multiplier for the final classifier layer
- Lower base learning rate (e.g., 0.001) for fine-tuning

7 Evaluation

Evaluate a trained model:

```
python main.py /path/to/dataset \
    --arch resnet50 \
    --representation SVD_Halley \
    --num-classes 1000 \
    --benchmark ILSVRC2012 \
    --evaluate \
    --resume /path/to/checkpoint.pth.tar
```

8 Performance Tips

1. **GPU Memory**: Halley-SVD requires more memory than standard pooling. Adjust batch size accordingly.
2. **Learning Rate**: Use standard schedules (e.g., divide by 10 at epochs 30, 60, 90 for 100 epochs total).
3. **Warm-up**: Consider using learning rate warm-up for the first few epochs.

9 Troubleshooting

9.1 Out of Memory Errors

- Reduce batch size
- Use gradient accumulation
- Consider using mixed precision training

9.2 Convergence Issues

- Ensure proper learning rate scheduling
- Check if pretrained weights are loaded correctly
- Verify dataset normalization matches pretrained model statistics

10 Citation

If you use this code in your research, please cite the relevant papers on differentiable SVD and global covariance pooling methods.

11 License

Please refer to the LICENSE file for details on usage and distribution.