thstkdgus35 / **EDSR-PyTorch**

PyTorch version of the paper 'Enhanced Deep Residual Networks for Single Image Super-Resolution' (CVPRW 2017)

100 commits | **4** branches | **0** releases | **7** contributors | MIT

Branch: **master** | New pull request | Create new file | Upload files | Find File | Clone or download

thstkdgus35 Now support PyTorch 1.1.0 by default — Latest commit `565d6fe` on 8 May

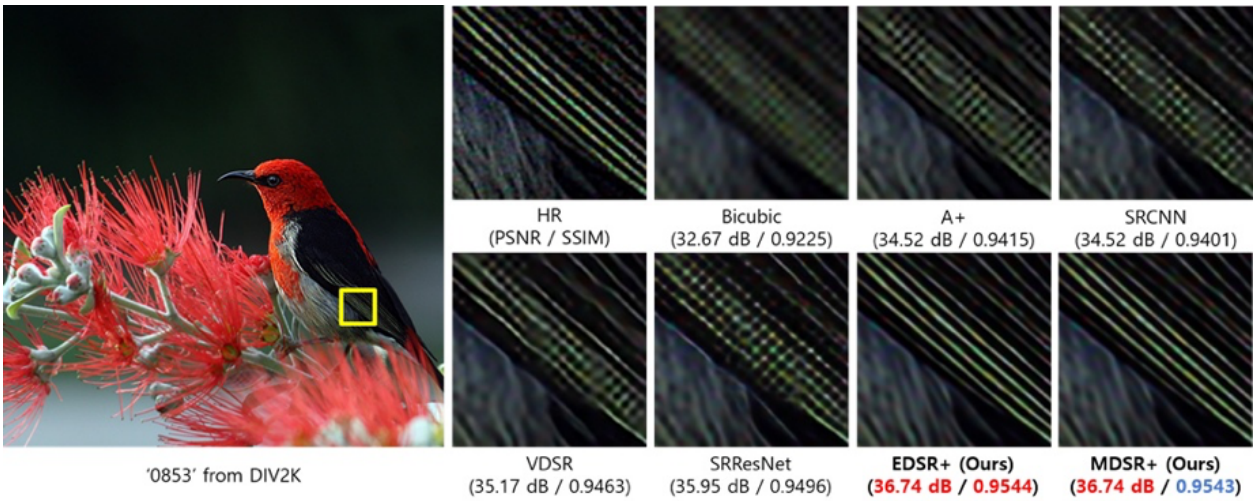
experiment	Oct 18, 2018: minor updates	9 months ago
figs	NTIRE 2019 figure	6 months ago
src	Now support PyTorch 1.1.0 by default	2 months ago
test	Jan 04, 2018 updates	2 years ago
.gitignore	Updated readme	2 years ago
LICENSE	add license	last year
README.md	Now support PyTorch 1.1.0 by default	2 months ago

README.md

EDSR-PyTorch

About PyTorch 1.1.0

- There have been minor changes with the 1.1.0 update. Now we support PyTorch 1.1.0 by default, and please use the legacy branch if you prefer older version.



This repository is an official PyTorch implementation of the paper **"Enhanced Deep Residual Networks for Single Image Super-Resolution"** from **CVPRW 2017, 2nd NTIRE**. You can find the original code and more information from here.

If you find our work useful in your research or publication, please cite our work:

[1] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee, **"Enhanced Deep Residual Networks for Single Image Super-Resolution,"** *2nd NTIRE: New Trends in Image Restoration and Enhancement workshop and challenge on image super-resolution in conjunction with **CVPR 2017***. [PDF] [arXiv] [Slide]

```
@InProceedings{Lim_2017_CVPR_Workshops,
```

```
  author = {Lim, Bee and Son, Sanghyun and Kim, Heewon and Nah, Seungjun and Lee, Kyoung Mu},
  title = {Enhanced Deep Residual Networks for Single Image Super-Resolution},
  booktitle = {The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops},
  month = {July},
  year = {2017}
}
```

We provide scripts for reproducing all the results from our paper. You can train your own model from scratch, or use pre-trained model to enlarge your images.

Differences between Torch version

- Codes are much more compact. (Removed all unnecessary parts.)
- Models are smaller. (About half.)
- Slightly better performances.
- Training and evaluation requires less memory.
- Python-based.

Dependencies

- Python 3.6
- PyTorch >= 1.0.0
- numpy
- skimage
- **imageio**
- matplotlib
- tqdm
- cv2 >= 3.xx (Only if you want to use video input/output)

Code

Clone this repository into any place you want.

```
git clone https://github.com/thstkdgus35/EDSR-PyTorch
cd EDSR-PyTorch
```

Quick start (Demo)

You can test our super-resolution algorithm with your own images. Place your images in `test` folder. (like `test/<your_image>`) We support **png** and **jpeg** files.

Run the script in `src` folder. Before you run the demo, please uncomment the appropriate line in `demo.sh` that you want to execute.

```
cd src       # You are now in */EDSR-PyTorch/src
sh demo.sh
```

You can find the result images from `experiment/test/results` folder.

Model	Scale	File name (.pt)	Parameters	**PSNR
EDSR	2	EDSR_baseline_x2	1.37 M	34.61 dB
		*EDSR_x2	40.7 M	35.03 dB
	3	EDSR_baseline_x3	1.55 M	30.92 dB

		*EDSR_x3	43.7 M	31.26 dB
	4	EDSR_baseline_x4	1.52 M	28.95 dB
		*EDSR_x4	43.1 M	29.25 dB
MDSR	2	MDSR_baseline	3.23 M	34.63 dB
		*MDSR	7.95 M	34.92 dB
	3	MDSR_baseline		30.94 dB
		*MDSR		31.22 dB
	4	MDSR_baseline		28.97 dB
		*MDSR		29.24 dB

*Baseline models are in `experiment/model`. Please download our final models from here (542MB) **We measured PSNR using DIV2K 0801 ~ 0900, RGB channels, without self-ensemble. (scale + 2) pixels from the image boundary are ignored.

You can evaluate your models with widely-used benchmark datasets:

Set5 - Bevilacqua et al. BMVC 2012,

Set14 - Zeyde et al. LNCS 2010,

B100 - Martin et al. ICCV 2001,

Urban100 - Huang et al. CVPR 2015.

For these datasets, we first convert the result images to YCbCr color space and evaluate PSNR on the Y channel only. You can download benchmark datasets (250MB). Set `--dir_data <where_benchmark_folder_located>` to evaluate the EDSR and MDSR with the benchmarks.

You can download some results from here. The link contains **EDSR+_baseline_x4** and **EDSR+_x4**. Otherwise, you can easily generate result images with `demo.sh` scripts.

How to train EDSR and MDSR

We used DIV2K dataset to train our model. Please download it from here (7.1GB).

Unpack the tar file to any place you want. Then, change the `dir_data` argument in `src/option.py` to the place where DIV2K images are located.

We recommend you to pre-process the images before training. This step will decode all **png** files and save them as binaries. Use `--ext sep_reset` argument on your first run. You can skip the decoding part and use saved binaries with `--ext sep` argument.

If you have enough RAM (>= 32GB), you can use `--ext bin` argument to pack all DIV2K images in one binary file.

You can train EDSR and MDSR by yourself. All scripts are provided in the `src/demo.sh`. Note that EDSR (x3, x4) requires pre-trained EDSR (x2). You can ignore this constraint by removing `--pre_train <x2 model>` argument.

```
cd src       # You are now in */EDSR-PyTorch/src
sh demo.sh
```

Update log

- Jan 04, 2018
    - Many parts are re-written. You cannot use previous scripts and models directly.
    - Pre-trained MDSR is temporarily disabled.
    - Training details are included.

- Jan 09, 2018
  - Missing files are included (`src/data/MyImage.py`).
  - Some links are fixed.
- Jan 16, 2018
  - Memory efficient forward function is implemented.
  - Add --chop_forward argument to your script to enable it.
  - Basically, this function first split a large image to small patches. Those images are merged after super-resolution. I checked this function with 12GB memory, 4000 x 2000 input image in scale 4. (Therefore, the output will be 16000 x 8000.)
- Feb 21, 2018
  - Fixed the problem when loading pre-trained multi-gpu model.
  - Added pre-trained scale 2 baseline model.
  - This code now only saves the best-performing model by default. For MDSR, 'the best' can be ambiguous. Use --save_models argument to save all the intermediate models.
  - PyTorch 0.3.1 changed their implementation of DataLoader function. Therefore, I also changed my implementation of MSDataLoader. You can find it on feature/dataloader branch.
- Feb 23, 2018
  - Now PyTorch 0.3.1 is default. Use legacy/0.3.0 branch if you use the old version.
  - With a new `src/data/DIV2K.py` code, one can easily create new data class for super-resolution.
  - New binary data pack. (Please remove the `DIV2K_decoded` folder from your dataset if you have.)
  - With `--ext bin`, this code will automatically generates and saves the binary data pack that corresponds to previous `DIV2K_decoded`. (This requires huge RAM (~45GB, Swap can be used.), so please be careful.)
  - If you cannot make the binary pack, just use the default setting (`--ext img`).
  - Fixed a bug that PSNR in the log and PSNR calculated from the saved images does not match.
  - Now saved images have better quality! (PSNR is ~0.1dB higher than the original code.)
  - Added performance comparison between Torch7 model and PyTorch models.
- Mar 5, 2018
  - All baseline models are uploaded.
  - Now supports half-precision at test time. Use `--precision half` to enable it. This does not degrade the output images.
- Mar 11, 2018
  - Fixed some typos in the code and script.
  - Now --ext img is default setting. Although we recommend you to use --ext bin when training, please use --ext img when you use --test_only.
  - Skip_batch operation is implemented. Use --skip_threshold argument to skip the batch that you want to ignore. Although this function is not exactly same with that of Torch7 version, it will work as you expected.
- Mar 20, 2018
  - Use `--ext sep_reset` to pre-decode large png files. Those decoded files will be saved to the same directory with DIV2K png files. After the first run, you can use `--ext sep` to save time.
  - Now supports various benchmark datasets. For example, try `--data_test Set5` to test your model on the Set5 images.
  - Changed the behavior of skip_batch.

- Mar 29, 2018
  - We now provide all models from our paper.
  - We also provide `MDSR_baseline_jpeg` model that suppresses JPEG artifacts in original low-resolution image. Please use it if you have any trouble.
  - `MyImage` dataset is changed to `Demo` dataset. Also, it works more efficient than before.
  - Some codes and script are re-written.
- Apr 9, 2018
  - VGG and Adversarial loss is implemented based on SRGAN. WGAN and gradient penalty are also implemented, but they are not tested yet.
  - Many codes are refactored. If there exists a bug, please report it.
  - D-DBPN is implemented. Default setting is D-DBPN-L.
- Apr 26, 2018
  - Compatible with PyTorch 0.4.0
  - Please use the legacy/0.3.1 branch if you are using the old version of PyTorch.
  - Minor bug fixes
- July 22, 2018
  - Thanks for recent commits that contains RDN and RCAN. Please see `code/demo.sh` to train/test those models.
  - Now the dataloader is much stable than the previous version. Please erase `DIV2K/bin` folder that is created before this commit. Also, please avoid to use `--ext bin` argument. Our code will automatically pre-decode png images before training. If you do not have enough spaces(~10GB) in your disk, we recommend `--ext img` (But SLOW!).
- Oct 18, 2018
  - with `--pre_train download`, pretrained models will be automatically downloaded from server.
  - Supports video input/output (inference only). Try with `--data_test video --dir_demo [video file directory]`.