

RePaint

This repository contains code for reproducing the repaint paper:

Repaint: Improving the Generalization Performance of Down-Stream Visual Tasks by Generating Multiple Instances of Training Examples

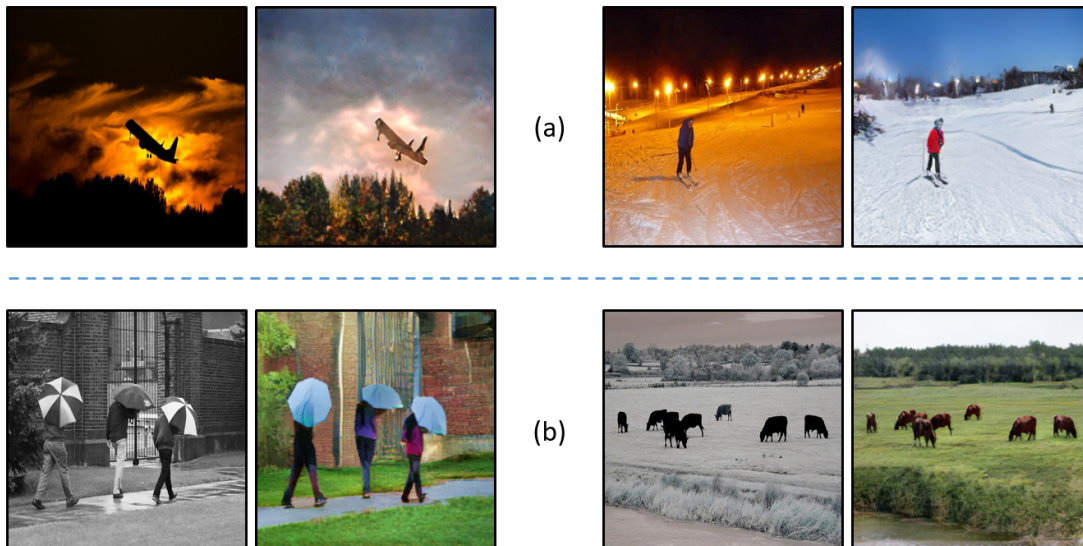
- Repaint is a method for diversifying the texture/color while preserving shapes/structures, and feeding these examples to down-stream tasks.
- The main idea in this method is to regenerate multiple repainted examples from each input image, and use these diversified examples in training, to improve the generalization of down-stream tasks.
- Examples - Diversified Texture: in each case, the top-left corner shows an original image, and the other three are repainted versions.



- Examples - Paired Snapshots: top row are original images, and bottom row are repainted ones. Notice the texture changes such as: tower lights, boat reflection and design, beach waves pattern, animal bodies or land coverage, and mountain snow.



- Examples - Uncovering Texture: a: uncovering from the dark things such as trees texture, mountain trees, or a building far away. b: colorization. For each pair, the left-side image is the original, and the right-side one is a repainted version.



Requirements

```
torch>=1.6
torchvision
opencv-python
cython
pycocotools
albumentations
pillow>=6.1
dill
dominate
pandas
psutil
requests
tqdm
tensorboardX
pyyaml
numpy
scikit-image
pyyaml
omegaconf
```

Code directory structure

```
.
├── README.md
├── src
│   ├── classification
│   ├── detection
│   ├── segmentation
│   ├── data_calibration
│   │   └── Repaint
│   │       └── train.py
├── docs
└── readme.pdf
```

- The algorithm is mainly hosted in the `Repaint` folder, with the main entry point being `train.py` file.
- The `classification` and `detection` folders contain code for stand-alone down-stream tasks of image classification and object detection. The `segmentation` folder contains a copy of DeepLabv2 to generate masks (in the paper we also tried the rough masks of the FH method).
- The `docs` folder contains a PDF copy of the instructions and example snapshots.

How to run the code

Training for the image classification down-stream task

```
cd code
export PYTHONPATH=$PWD
python3 src/data_calibration/Repaint/train.py
```

- Note that for the results obtained in the paper, we ran jobs on cloud cluster nodes with 8-V100 GPUs (32 GB memory) and various configurations.
- Also, you would need to ensure appropriate options and arguments are selected by reviewing the default values in:
 - a) `src/data_calibration/Repaint/options` and check specially `base_options.py` and `train_options.py`.
 - b) `src/data_calibration/Repaint/tasks/classification/options.py` for adjusting the classification options.
- More info on the GAN modules arguments and data folder setup can be found [here](#).

Training for the object detection down-stream task

This part of the code will be released when the paper is published.