Apr. 26, 2024

AI6126 – Advanced Computer Vision Project # 2

Reinelle Jan Bugnot
G2304329L

Codalab username: *freepoul*

I. Model Architecture

For this project, I implemented both the Real-ESRGAN [1] and SwinFIR-T [2] models to perform the image super-resolution task. Both models are based on the BasicSR toolbox that implements many popular image restoration methods. Real-ESRGAN is the base model provided for the project. In this implementation, the real-esrgan model contains **1,517,571 trainable parameters**. Meanwhile, SwinFIR-T, which is the lightweight version of SwinFIR, which uses the basic building blocks of the Swin Transformer has **966,696 trainable parameters**.

Both models were trained on unpaired images for this super-resolution task. However, SwinFIR-T only provides modules for training on paired images. Hence, the provided degradation process had to be manually integrated into the base SwinFIR code, as well as modifying the training settings to accommodate this new training pipeline. The following configuration for the image degradation process is implemented identically for both model training pipelines:

```
# dataset and data loader settings
datasets:
  train:
    name: FFHQ_train
    type: FFHQsubDataset
    dataroot_gt: data/FFHQ/train/GT
    meta_info: data/FFHQ/train/meta_info_FFHQ5000sub_GT.txt
    io_backend:
      type: disk

    blur_kernel_size: 21
    kernel_list: ['iso', 'aniso', 'generalized_iso', 'generalized_aniso',
'plateau_iso', 'plateau_aniso']
    kernel_prob: [0.45, 0.25, 0.12, 0.03, 0.12, 0.03]
    sinc_prob: 0.1
    blur_sigma: [0.2, 3]
    betag_range: [0.5, 4]
    betap_range: [1, 2]

    blur_kernel_size2: 21
    kernel_list2: ['iso', 'aniso', 'generalized_iso', 'generalized_aniso',
'plateau_iso', 'plateau_aniso']
    kernel_prob2: [0.45, 0.25, 0.12, 0.03, 0.12, 0.03]
    sinc_prob2: 0.1
    blur_sigma2: [0.2, 1.5]
```

```
    betag_range2: [0.5, 4]
    betap_range2: [1, 2]

    final_sinc_prob: 0.8

    gt_size: 512
    use_hflip: True
    use_rot: False

    # data loader
    use_shuffle: true
    num_worker_per_gpu: 6
    batch_size_per_gpu: 8
    dataset_enlarge_ratio: 1000
    prefetch_mode: ~

  val:
    name: FFHQ_val
    type: PairedImageDataset
    dataroot_gt: data/FFHQ/val/GT
    dataroot_lq: data/FFHQ/val/LQ
    io_backend:
      type: disk
```

II. Loss Function

For both models, the default loss functions used in their corresponding papers were used. For RealESRGAN, **L1 Loss** was used:

```
  # losses
  pixel_opt:
    type: L1Loss
    loss_weight: 1.0
    reduction: mean
```

Meanwhile, for SwinFIR-T, the (default) **Charbonnier Loss** was used:

```
    # losses
    pixel_opt:
      type: CharbonnierLossColor
      loss_weight: 1.0
      reduction: mean
```

The Charbonnier Loss, also known as the pseudo-Huber loss, is a robust loss function commonly used in computer vision and image processing tasks. It is designed to be less sensitive to outliers and noise in the data compared to traditional loss functions like mean squared error (MSE).

III. Specs of Training Machine

Given the computational complexity of this project compared to previous ones, the main bottleneck (and differentiator across models) is training time and compute resource. In my case, I fully utilized the SCSE GPU Cluster to train my models, using the two QOS assigned to me: q_amsai and q_dmsai. The available hardware information are as follows:

Table II. Hardware Specifications

	CPU	GPU	memory	MaxWall
q_amsai	7	1	12G	8 Hrs
q_dmsai	10	1	30G	8 Hrs

My training strategy is to use the smaller q_amsai for simple parameter tuning, and then using q_dmsai for deep training with large training iterations for submission. However, the 8-hour MaxWall for the cluster GPU usage only allows the training of approximately 125,000 iterations.

IV. Implemented Strategies

One straightforward approach to enhance the performance of the base Real-ESRModel is to train for as long as possible. The default settings outlined in the provided configuration file specifies 300,000 training iterations, which requires approximately one full day given the current limitations of GPU resources.

However, this isn't a very interesting approach for me. Instead, I focused on trying to implement a different model architecture. Here, I chose SwinFIR, which claims to surpass Real-ESRGAN and other established SR models in specific SR benchmarks. Yet, the full SwinFIR model boasts around 24 million trainable parameters, exceeding the project's imposed limit. However, the lightweight version, SwinFIR-T, containing roughly 900k trainable parameters, can be a feasible option. If SwinFIR-T can rival the performance of the Real-ESRGAN implementation while requiring half the parameters, it may possess the potential to surpass other Real-ESRGAN-focused implementations.

V. Results

Fig. 1 shows the training loss curves generated after training both Real-ESRGAN and SwinFIR-T using their default hyperparameters.

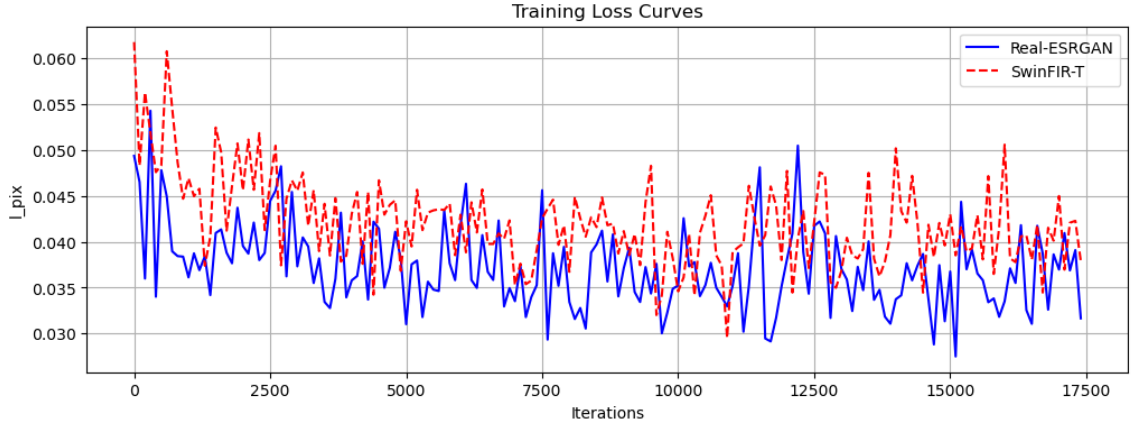

Fig. 1. Training Loss Curves for Real-ESRGAN and SwinFIR-T Implementations. Training Curve for Real-ESRGAN sliced from the full 125k iters to align with the curve from SwinFIR-T.

One major observation is that the SwinFIR-T takes significantly longer to train than Real-ESRGAN (approximately 5x longer). Within the 8-hour GPU usage limit, I was able to train for 125k iterations using Real-ESRGAN, but only 17.5k iterations when using SwinFIR-T.

From the same figure, we can see that, on average, Real-ESRGAN achieves a lower pixel loss compared to SwinFIR-T on default settings and no hyperparameter tuning. Table I shows the performance of both models on the validation set, while Fig. 2 shows the models' performance on the test set.

Table I. Validation Results for Real-ESRGAN and SwinFIR-T

	Real-ESRGAN-125k_it	**SwinFIR-T-17.5k_it**
PSNR	26.2864	26.0331

#	SCORE	FILENAME	SUBMISSION DATE	SIZE (BYTES)	STATUS	✔	
1	26.1588338549	preds.zip	04/24/2024 12:21:26	28220	Finished		+
2	26.3882211483	00000.zip	04/25/2024 05:07:02	35625	Finished	✔	+
3	---	00379_SwinFIR-T-CUSTOM.zip	04/26/2024 04:31:44	57199	Failed		+
4	26.0354832748	results.zip	04/26/2024 04:39:19	57141	Finished		+

Fig. 2. Leaderboard / Test PSNR. Entries are as follows: Real-ESRGAN-10k_it, Real-ESRGAN-125k_it, Failed entry due to filename mismatch, SwinFIR-T-17.5k_it

Both validation and inference results show that Real-ESRGAN consistently outperforms SwinFIR-T on default settings with no hyperparameter tuning. However, the fact that SwinFIR-T was able to produce results around 26 dB PSNR means that my implementation of SwinFIR-T was successful. With enough compute resource, I hypothesize that SwinFIR-T will outperform Real-ESRGAN since it operates on more modern components (i.e. Transformers). Modifying the SwinFIR-T network to maximize the parameter limit and tuning the model parameters properly are just two potential steps towards further improvement.

References

[1] https://github.com/xinntao/Real-ESRGAN

[2] SwinFIR: Revisiting the SwinIR with Fast Fourier Convolution and Improved Training for Image Super-Resolution. arXiv preprint arXiv:2208.11247, August 2022