MOESR: MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM FOR IMAGE SUPER-RESOLUTION

Anonymous authors
Paper under double-blind review

A OVERVIEW

The supplementary materials included in this paper provide an exhaustive quantitative and qualitative assessment of the proposed method. In Section B, we delineate the process of decomposing the multi-objective optimization into multiple subproblems and conduct an ablation study concerning the relevant parameters. Section C contains a detailed description of our method's training process. In addition, a more in-depth qualitative analysis is performed in Section D. Besides, the details of the definition of the multi-objective evaluation metrics in Section E. Finally, we expand upon related works in Section F.

B DECOMPOSITION METHOD

At first, to avoid the effect of the magnitudes of SSIM and $L1$, we normalize them separately.

$$\begin{aligned} ssim_{norm} &= \frac{ssim - ssim_{min}}{ssim_{min} - ssim_{max}} \\ L1_{norm} &= \frac{L1 - L1_{min}}{L1_{max} - L1_{min}} \end{aligned} \tag{1}$$

Noting that the closer $ssim_{norm}$ and $L1_{norm}$ are to 0, the higher the quality of the super-resolution image. Then, we calculate

$$\begin{aligned} &ssim_{dif} = ssim_{norm} - ssim_{mean} \\ &L1_{dif} = L1_{norm} - L1_{mean} \end{aligned} \tag{2}$$

where $ssim_{mean}$ and $L1_{mean}$ represent the average of $ssim_{norm}$ and $L1_{norm}$ over all populations, respectively.

Based on this, we carried out the calculation of the decomposition ratio of the two objectives.

$$\begin{aligned} p_{ssim} &= \frac{e^{L1_{dif}}}{e^{ssim_{dif}} + e^{L1_{dif}}} \\ p_{L1} &= \frac{e^{ssim_{dif}}}{e^{ssim_{dif}} + e^{L1_{dif}}} \end{aligned} \tag{3}$$

Thus, our score function is as follows:

$$score = p_{ssim} * ssim_{norm} + p_{L1} * L1_{norm} \tag{4}$$

However, experimental findings revealed that training the model using SSIM Loss during the parent preparation phase resulted in parent solutions with more substantial variation in SSIM values. This, in turn, predisposed the model towards optimizing for SSIM. As such, we incorporated parameters to regulate its proportion.

Accordingly, the final score function assumes the following form:

$$score = (1 - \lambda) p_{ssim} * ssim_{norm} + \lambda * p_{psnr} * psnr_{norm} \tag{5}$$

To thoroughly evaluate the influence of varying λ values on the MOESR decomposition problem, we selected the EDSR baseline as a representative model. The impact of distinct λ values on the final image quality was assessed across five datasets: Set5, Set14, BSD100, Urban100, and Manga109.

As Table 1 indicates, a clear pattern emerges: as λ increases, the model shows a tendency towards the optimization of PSNR values. Conversely, SSIM values demonstrate a steady rise as λ decreases. This observed correlation aligns with our initial hypotheses and further validates the idea that the trajectory of multi-objective optimization can be modulated through adjustments to the scoring function. This offers the potential to more effectively guide the direction of multi-objective optimization.

Table 1: The decomposition ration influence of the performance. Average PSNR/SSIM for scale factor x2, on benchmark datasets Set5, Set14, BSD100, Urban100, and Manga109.

λ	Scale	Set5		Set14		BSD100		Urban100		Manga109	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
0.1	x2	37.67	0.9664	33.48	0.9220	31.87	0.9082	31.48	0.9325	37.96	0.9815
0.2	x2	37.72	0.9621	33.52	0.9205	37.99	0.9056	31.67	0.9309	38.04	0.9793
0.4	x2	37.83	0.9608	33.55	0.9190	32.08	0.9030	31.90	0.9290	38.22	0.9789
0.6	x2	37.99	0.9604	33.59	0.9175	32.25	0.9014	32.16	0.9272	38.35	0.9769
0.8	x2	38.13	0.9599	33.71	0.9152	32.38	0.8995	32.18	0.9248	38.54	0.9757
1.0	x2	38.31	0.9587	33.96	0.9138	33.46	0.8978	32.34	0.9231	38.96	0.9741

C ADDITIONAL IMPLEMENTATION DETAILS

Parents Preparation. Existing methods predominantly employ L1 loss for training, with a primary focus on optimizing the PSNR. However, to accomplish multi-objective optimization, we require solutions that yield better SSIM values. As such, we opted to utilize SSIM loss to train our model for one epoch, selecting the results from 20 equally distributed batches as parent solutions. To further detail our methodology, we set the learning rate to 1e-6 and the batch size to 32. All other configuration settings are maintained as the corresponding paper.

MOESR training details. Based on the experiments in Section B, we set the decomposition weight λ to 0.4 and the number of parent generations Popsize to 20. the number of iterations to 60, and the Batchsize to 160. And we use SHADE as our optimization method.

D QUALITATIVE ANALYSIS

Generally speaking, training super-resolution tasks using supervised deep learning with an L1 loss function often results in models that make images appear darker and blurrier, while an SSIM loss function focuses more on the brightness and structural information of the images. Our experiments have revealed that the models optimized using our multi-objective approach exhibit significantly improved overall brightness, more in line with high-resolution reference images. Therefore, our method provides a better visual effect while maintaining the same MSE error. Since visual effects related to brightness are not easily discernible in small patches, we recommend comparing the visual effects of different methods using the entire image.

E EVALUATION METRICS

For a fair evaluation of MO-based super-resolution methods, we apply the following metrics to MOSR tasks: generational distance (GD), and hypervolume indicator (HV). S represents the set of approximate Pareto solutions obtained by the optimization algorithm, while P denotes the set of solutions obtained through uniform sampling on the true Pareto front.

$$GD(S,P) = \frac{\sum_{x \in S} d(x,P)}{|S|}$$

$d(x,P)$ represents the Euclidean distance between solution x in S and its nearest solution in P. The number of $|S|$ said set S. A smaller value of GD means that the solution set S is closer to the true

Pareto solutions.

$$HV(S) = \mathrm{VOL}\left(\bigcup_{x\in S} [f_1(x), r_1^*] \times \cdots \times [f_m(x), r_m^*]\right)$$

Where $r = (r_1, \ldots, r_m)^T$ represents a predefined reference point, VOL signifies the Lebesgue metric, and HV denotes the spatial convolution sum between the solution set S and r. A higher HV value signifies a higher-quality solution set.

In the context of this paper, the set P is not initially available. We generate both P and r^* using the following approach, which enables us to compute IGD, HV, and metrics.

Initially, we normalize the current set of solutions to be evaluated using the maximum and minimum values from the initial population. We define the point (1.1, 1.1) as our reference point and draw a circle with its center at the origin (0,0).

For each normalized solution, we create a set of reference point sets by intersecting the circle with a line that connects that solution point to the origin, as illustrated in Figure 1. We then de-normalize all these points to obtain the set of reference solutions P required for GD and IGD evaluation, as well as the reference point r^* necessary for HV computation, as depicted in Figure 2. Consequently, the evaluation of the current solution set depends not only on the set itself but also on the characteristics of the initial population.

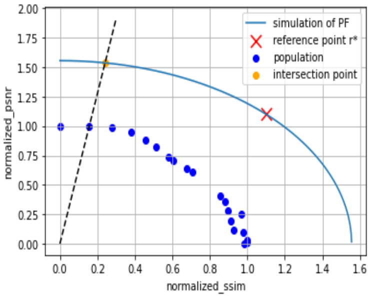

Figure 1: Normalization and generate reference points

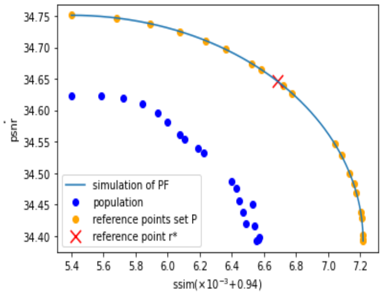

Figure 2: The reference point r^* and the set of reference points P

F ADDITIONAL RELATED WORK

Evolutionary Algorithm. EA methods have proven effective in the design and training of reinforcement learning models, as demonstrated in several studies Montana et al. (1989); Stanley et al. (2019); Suganuma et al. (2017); Zhang & Li (2007). Neuro-evolution, for instance, NEAT Stanley & Miikkulainen (2002), has been successfully deployed in creating more efficient neural architectures. Further, certain neuro-evolution techniques, such as evolution strategy Salimans et al. (2017), have been shown to outperform the deep Q-learning, as well as the policy gradient algorithm A3C Mnih et al. (2016), among others, in reinforcement learning tasks. Nevertheless, these EA-based methods have reportedly been more efficacious with smaller datasets and smaller Deep Neural Networks (DNNs)Piotrowski (2014). When applied to optimize DNN weights in large-scale datasets, EA-based methods often struggle with slow convergence or even failure to converge, given the overwhelming number of model parameters and the complexity of the search space required to achieve deep representation. Piotrowski documented the stagnation issues of several adaptive Differential Evolution (DE) variants, including SADE, JADE, and DEGL, when optimizing network weights for regression problemsPiotrowski (2014). To address this, Sun et al. proposed an efficient gene encoding approach that leverages the concept of null spaces for DNNs in unsupervised learning tasks on the MNIST and CIFAR-10 scales Sun et al. (2018). Yet, existing EA-based weight optimization methods have demonstrated limited scalability when applied to larger datasets.

REFERENCES

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

David J Montana, Lawrence Davis, et al. Training feedforward neural networks using genetic algorithms. In *IJCAI*, volume 89, pp. 762–767, 1989.

Adam P Piotrowski. Differential evolution algorithms applied to neural network training suffer from stagnation. *Applied Soft Computing*, 21:382–406, 2014.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

Kenneth O Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1(1):24–35, 2019.

Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. A genetic programming approach to designing convolutional neural network architectures. In *Proceedings of the genetic and evolutionary computation conference*, pp. 497–504, 2017.

Yanan Sun, Gary G Yen, and Zhang Yi. Evolving unsupervised deep neural networks for learning meaningful representations. *IEEE Transactions on Evolutionary Computation*, 23(1):89–103, 2018.

Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.