APPLIED MACHINE LEARNING SYSTEMS II (ELEC0135) 22/23 REPORT

William De Vena
University College of London

ABSTRACT

In this work, the Image Super Resolution task is investigated. In particular, two approaches are proposed: the first exploits a CNN model based on the Residual-In-Residual Dense Block (RRDB), and trained on an L1 loss function, while the second proposes an adversarial framework (GAN). The performance of the CNN model, used also in the adversarial framework as the generator, is improved by removing batch normalization, demonstrated to produce artifacts in the final images. The experimental results show that the GAN model improves both the results of the baseline models and the CNN, obtaining an average LPIPS distance of 0.238.
[1]

Index Terms— Machine learning, deep learning, computer vision, image super-resolution, GAN, CNN

1. INTRODUCTION

This work focuses on the task of image super-resolution (SR), which is the process of recovering a high-resolution image, from its low-resolution version, that is visually appealing and perceptually similar to the original one. This task is particularly important in situations where high-resolution images are crucial for critical applications, such as medical imaging [1, 2], surveillance systems [3], satellite imaging [4], and gaming [5]. In particular, SR techniques have the potential to improve the visual quality of images, allowing for better analysis, diagnosis, and decision-making. For instance, in medical imaging, SR techniques can enhance the resolution of medical images, such as CT scans and MRIs, providing more accurate diagnoses and better treatment outcomes. In surveillance systems, these techniques can improve the quality of video footage, making it easier to identify objects and individuals. In satellite imaging, improving the resolution of images allows for more accurate mapping and monitoring of environmental changes. Finally, in gaming, SR techniques can improve the resolution of the rendered scenes, providing a better experience to the user. Furthermore, because the end goal of SR techniques is to improve the quality of the images, they are often used as a preprocessing step for further CV tasks [6].

[1]The link of the GitHub project is provided: https://github.com/williamdevena/Image_Super_Resolution

2. LITERATURE SURVEY

Before the advent of Deep Learning and of CNNs, SR was tackled using interpolation-based methods. In particular, the most used methods are Nearest-Neighbor interpolation, which selects the value of the nearest pixel for each position, bilinear interpolation, which simply performs linear interpolation, and finally, bicubic interpolation, which similarly performs cubic interpolation. Since these methods are fairly simple to implement, interpretable, and with computational low costs, they are still today often used in CNN-based models [7]. One of the first works that introduced the use of Deep Learning in SR, and more specifically of CNNs, is [8]. Since then, many other works have used deep networks to improve the quality of the reconstructed images [9, 10, 11]. Subsequent works introduced more advanced convolutional blocks. For instance, [11] used residual blocks, while [12] further enhanced them by using dense residual blocks, fully exploiting the hierarchical features from all the layers, instead of only the features from the preceding layer. Furthermore, to tackle the SR task, other works have exploited more advanced NN designs. For example, [13, 14] used recursive NN to build very deep networks without having to increase the number of parameters. Adversarial learning has also been used in the SR landscape [15, 16, 17]. In particular, the architecture introduced by [15], was then further enhanced by [16]. Specifically, they built a deeper version of the generator architecture and removed batch normalization, which was demonstrated to cause the presence of artifacts in the final output. Additionally, they replaced the standard discriminator, which estimates the probability that one input image is real, with a relativistic one [18], which estimates the probability that a real image is relatively more realistic than a fake one. Finally, they improved the perceptual loss used to train the generator. In particular, in [15] the perceptual loss was calculated by taking the Euclidean distance between the two feature maps, produced by a VGG network's intermediate layers (after activation), of the output and ground truth images. On the other hand, the authors of [16] showed that by taking the feature maps after activation, there was a substantial loss of information, and demonstrated an improvement in performance by using the feature maps before activation. Furthermore, more recent works [19, 20], to tackle SR, have exploited the power of transformer-based models. Finally, by analyzing the methods used in the NTIRE 2022 super-resolution challenge [21], other categories of approaches have been introduced in the last few years. For instance, the authors of [22] showed that Denoising Diffusion Probabilistic Models [23] can achieve and improve state-of-the-art performance in SR. Furthermore, the winning team of the NTIRE 2022 SR challenge used an approach inspired by the work [24], where the authors introduced a methodology called SRFlow. The authors train a Normalizing Flow model to transform a HR image conditioned on a LR image into a latent variable, using a negative log-likelihood loss function calculated on the latent variable. Subsequently, at inference time they exploit the property of Normalizing Flows of being invertible. In particular, they sample a latent vector of Gaussian noise and pass it through the architecture, conditioning on the LR image, to produce the HR one.

3. DESCRIPTION OF MODELS

This section describes the two main approaches used in this work to tackle the SR task. In particular, the first approach tested uses a CNN with Dense Residual Blocks, while the second approach regards training the same model but in an adversarial framework in combination with a discriminator model.

3.1. CNN-based model with Dense Residual Blocks

The architecture used in the first approach takes inspiration from the one proposed in [16], where the authors proposed an enhanced version of the generator architecture introduced in [15]. In particular, in [16] the generator is composed of 23 Residual-In-Residual Dense Blocks (RRDB), which in turn are composed of 3 Dense Residual Blocks that, finally, are composed of 5 convolution layers, for a total of 345 (23*5*3) convolution layers. In this work, for computational reasons, the maximum number of RRDB blocks tested is 15. Going into the details of the architecture, the Dense Residual Block (DRB), which can be considered the building block of the model, is composed as follows (Fig. 1): 5 convolutional layers (convolution and activation function) follow each other and the input of every layer, before going into the convolution, is previously concatenated with all the previous inputs. Additionally, the output of the last block is summed with the initial input (skip connection). Furthermore, the RRDB is composed of 3 DRBs that follow each other and the output of the third DRB is summed with the initial input (Fig. 2). Finally, the architecture of the model is composed as follows (Fig. 3): an initial convolutional layer (3x3 kernel) is followed by a variable number of RRDBs (2, 10, or 15), another convolution layer, and two upsampling blocks. In particular, the upsampling blocks are composed of an upsampling operation (in this case using the Nearest-Neighbour algorithm), a convolution, and a LeakyReLu activation function. Finally, the last block

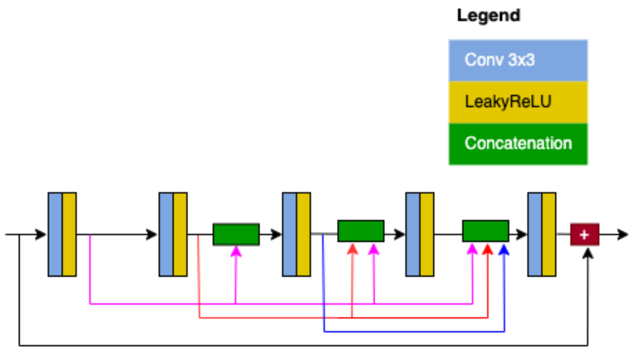


Fig. 1: Structure of a Dense Residual Block (DRB).

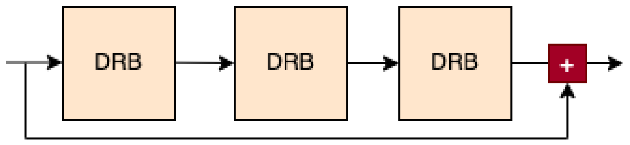


Fig. 2: Structure of a Residual-In-Residual Dense Block (RRDB).

of the architecture is composed of two convolutional layers and LeakyReLU activation function.

3.1.1. Residual and Dense Residual Blocks

The use of residual skip connections, as described by the authors of [25], is to ease the training of deep models. In particular, residual blocks ease the process of training deeper architectures by solving problems such as the vanish-gradient, thanks to skip connections that function as superhighways, allowing the gradient to flow without interference. Furthermore, the skip connection allows the block to approximate the identity function easily, hence not changing the previous layer's output. This solves the problem of degradation of neural networks, which occurs when after a certain threshold of depth, the performance of deeper models decreases. In particular, this problem is due to the difficulty of the added layers to approximate the identity function, hence by facilitating this process, skip connections solve the problem. Further-

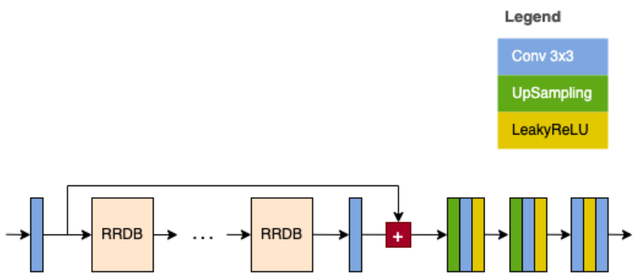


Fig. 3: Architecture of the CNN model.

more, the use of Dense Residual Blocks, as stated in [26], has several advantages: they further improve on the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and potentially reduce the number of parameters.

3.1.2. *Batch Normalization*

As in [16, 11, 27] batch normalization (BN) [28] was removed. In particular, it has been empirically proved that, despite the advantages that it brings, especially speeding up and easing the training process, removing BN in PSNR-oriented tasks, improves the overall performance of the model, and decreases the computational complexity and memory usage. In particular, it has been proved that when the distributions of the training data and testing data differ significantly, BN introduces unpleasant artifacts in the final outputs, and reduces the generalization capabilities of the model, especially with deep models and in adversarial frameworks [16].

3.2. Adversarial Learning

The second approach proposed to tackle the SR task regards exploiting the same CNN architecture but in an adversarial framework. More specifically, the architecture explained in Section 3.1 takes the role of the generator in a GAN [29]. In particular, the rationale for training the same architecture but in an adversarial framework refers to the fact that the PSNR-based approach, that is the CNN model trained on a pixel-level loss function, returns smooth images that lack high-resolution details (will see this in more details in the Section 5), while by inserting a discriminator, the generator is forced to produce images with sharper details. More specifically, in a GAN framework, the discriminator is simulating the role of a human judge, hence pushing the generator into producing more realistic images. This is the key idea behind adversarial learning: substituting an equation-form loss function with a model that distinguishes real from fake images, that in parallel trains to become better at it. In general, GANs can be formally composed of a generator G and a discriminator D, where D and G play a two-player minimax game with a value function $V(G, D)$ defined as follows:

$$E_{x\sim p_{\text{data}}(x)}[\log D(x)] + E_{z\sim p_z(z)}[1 - \log D(G(z))]. \quad (1)$$

where $p_{\text{data}}(x)$ is the real data generating distribution, while $p_z(z)$ is the prior distribution over latent vector z, hence the two-player minimax game is defined as follows

$$\min_G \max_D V(G, D). \quad (2)$$

Consequently, the discriminator's objective is to maximize $V(G, D)$, while the generator aims at minimizing it. Furthermore, to solve the problem of saturating gradients

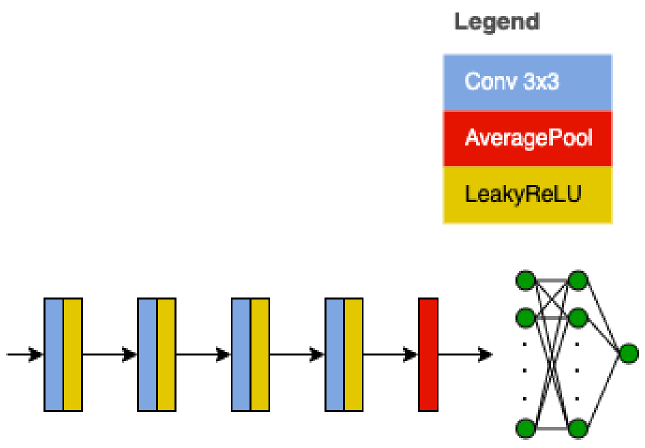


Fig. 4: Architecture of the CNN-based discriminator.

when training the generator, instead of minimizing the following value

$$E_{z\sim p_z(z)}[1 - \log D(G(z))]. \quad (3)$$

it aims at maximizing the following

$$E_{z\sim p_z(z)}[\log D(G(z))]. \quad (4)$$

In particular, the saturating gradients problem refers to the first iterations of training, where the generator has poor performances and the discriminator may reject samples with high confidence, and by using the first objective (Eq. 3) the gradients that generator receives could be very small, causing it to not train properly. Finally, for the discriminator role, a simple CNN model was used. In particular, the architecture (Fig. 4) is composed of 4 convolutional layers, which in turn are composed of a convolution layer and an activation function (LeakyReLU), an average pool layer, and a multi-layer perceptron, composed of two hidden layers, that performs the final classification.

3.3. Loss functions

Several loss functions have been used and tested in this work. In particular, caused by the fact that, compared to other CV tasks like image classification or image segmentation, SR has a definition that is more ambiguous and more complex to define mathematically, loss functions have been a major topic in the SR field [30, 31]. More specifically, the difficulty resides in expressing mathematically the concept of the quality of an image, and the reason is that, compared for example to assessing the classification of an image into a certain class, assessing the quality of an image and the similarity of two of them can have different interpretations. In particular, measuring the difference between two images can be done at a pixel level, meaning assessing the difference between the single pixels, but on the other hand, it can also be done by comparing more high-level features. In this work, several types of loss functions have been used to train the proposed models (Sec. 5 will go into more detail about the combinations of them used, comparison, and different results). In particular, the first type used to train both the CNN and the generator is pixel-level loss functions: this category of loss functions measures the differences between two images by considering only the single pixels. the two losses of this kind used in this work are the *L1 loss*, which measures the sum of all the absolute differences between the true pixel value and the predicted one, and the *L2 loss*, which on the other hand measures the sum of all the squared differences. Furthermore, in the adversarial framework, the discriminator has been trained using a simple *Binary Cross Entropy* loss, probably the most popular loss function in binary classification tasks.

3.3.1. Perceptual loss

This category of loss functions gives less weight to pixel-level differences and focuses more on high-level differences. In particular, they measure the difference between two images by comparing more high-level representations of them, hence downplaying the importance of single pixel values. The loss function used in this work is called perceptual or content loss and, rather than encouraging to match exactly the pixels, it encourages matching the feature representations. More specifically, it measures the squared and normalized euclidean distance between the feature representations of the two images. The feature representations are calculated by passing the two images as input two a CNN, in this case a VGG19, and extracting the output of the intermediate layers.

$$L_{perceptual} = \frac{1}{W_{ij}H_{ij}} \sum_{i=1}^{W_{ij}} \sum_{j=1}^{H_{ij}} (\phi_{ij}(I) - \phi_{ij}(\hat{I}))^2 \quad (5)$$

where W_{ij} and H_{ij} are the dimensions of the two feature maps and ϕ_{ij} refers to the feature map of the j-th convolution before the i-th maxpooling.

3.3.2. Adversarial loss

In the adversarial framework, in addition to the perceptual loss, an adversarial term is added to the loss function of the generator. In particular, by adding the adversarial loss, calculated on the output of the discriminator, the generator is forced to fool the discriminator by producing more natural images. The following is its mathematical representation:

$$L_{adversarial} = \sum_{i=1}^{N} -\log(D(G(I^{LR}))) \quad (6)$$

where D is the discriminator, G the generator, and I^{LR} is the low-resolution image given as input to the generator.

Fig. 5: Example of images in the NTIRE 2017 challenge dataset.

4. IMPLEMENTATION

4.1. Tools

Python was the programming language employed, and various widely-used libraries were also utilized, including Numpy, Matplotlib, Pandas, Scikit-learn, PyTorch, OpenCV, and Albumentations. To increase computational power, particularly for the training of Deep Learning models, the online platform Google Colab was utilized. Finally, both models mentioned in Sec. 3 were implemented from scratch using PyTorch, while other baseline models and evaluation metrics (described more in detail in Sec. 5) were implemented using built-in functions of the OpenCV library.

4.2. Dataset description

The dataset used in this work is the one provided by the NTIRE 2017 Super-Resolution Challenge [32]. This dataset contains images with slightly varying resolutions, with no specific themes or subject matter (Fig. 5). To cope with the limitations of computational resources, the high-resolution images in the dataset were downsampled by halving their resolution. This means that the high-resolution images used in this study have a resolution of around 1000x700, with each image having a slightly different resolution. On the other hand, the low-resolution images were produced online (during training, validation, or testing) by dividing the resolution of the original images by 4, resulting in images with a resolution of around 250x175. The dataset contains 903 PNG images, with 700 images used for training, 101 for validation, and 102 for testing, which amounts to approximately 80%, 10%, and 10% respectively.

4.3. Training strategies

The first model described in Sec. 3.1 was trained for 200 epochs using the Adam optimizer with a learning rate of 0.0001 that was halved after 100 epochs, β_1 of 0.9, β_2 of 0.999, a batch size of 16, and using early stopping technique to avoid overfitting. Furthermore, both the L1 and L2 loss functions were tested. Regarding the adversarial model, for computational reasons, it was not possible to train both the generator and the discriminator together in parallel, and to cope with this problem the approach used consisted in alternating the training of the two models. In particular, because for the generator, the pretrained CNN of the first approach was used, while the discriminator was training from scratch, the discriminator was the first to be trained for 50 epochs. After these initial 50 epochs, both models were alternated for two iterations of 50 epochs each (100 epochs each). Finally, regarding the loss functions used, for the discriminator the Binary Cross Entropy was used, while for the generator several combinations of the L1, perceptual, and adversarial loss functions were tested (Sec. 5 is going to go into the details of the different results).

5. EXPERIMENTAL RESULTS AND ANALYSIS

5.1. Evaluation metrics and baselines

To evaluate the performance of the proposed approaches three evaluation metrics were used: Peak signal-to-noise ratio (PSNR), Structural Similarity Index Measure (SSIM) [33], and the Learned Perceptual Image Patch Similarity (LPIPS) [34] metric. In particular, the first two metrics are pixel-level oriented metrics, while the latter focuses more on the comparison between high-level features. Finally, to compare the proposed approaches, two baseline algorithms were used: bilinear interpolation and nearest-neighbor upscaling.

5.2. Qualitative and quantitative results

5.2.1. CNN models

Different CNN versions, with a different number of RRDB blocks (2, 10, and 15), were tested. By training all three models for 200 epochs with the same hyperparameters (described in Sec. 4.3) it was highlighted that increasing the depth of the model, in the first non-adversarial approach, was not successful in improving the performance. In particular, as shown in Table 1, while the CNN models significantly improved the results of the baselines, the differences between the results of the shallower and deeper models are not significant enough to justify the increase in computational cost, hence in later experiments the shallower version is used.

Table 1: Results of the different CNN models.

Method	PSNR	SSIM	LPIPS
Baseline (NN)	25.32	0.829	0.379
Baseline (Bilinear Interp.)	26.06	0.836	0.391
CNN (2 RRDB blocks)	27.63	0.876	0.285
CNN (10 RRDB blocks)	27.69	0.877	0.287
CNN (15 RRDB blocks)	**27.76**	**0.879**	**0.283**

Furthermore, as already mentioned, in the SR field particular attention is given to the loss functions used. In fact, one of the several experiments that this work includes is focused on comparing the L1 and L2 loss functions, and as shown in Table 2, while the PSNR and SSIM metrics are improved by the L2 loss, the LPIPS distance is lower in the model trained with the L1 loss, and because, as already mentioned in Sec. 3.3, the LPIPS metric represents a better approximation of the comparison between two images, further experiments are performed using the L1 loss rather than the L2 loss. The results are probably caused by the fact that the L2 loss, by squaring the errors, penalizes bigger errors compared to the L1, hence it pushes the model to produce smoother images.

Table 2: Results of the CNN model trained with L1 and L2 loss function (for only 100 epcohs).

Method	PSNR	SSIM	LPIPS
CNN (L1 loss)	26.55	0.866	**0.315**
CNN (L2 loss)	**27.34**	**0.869**	0.342

5.2.2. Adversarial framework

In this section, the results of the experiments that regard the adversarial framework are shown. In particular, the first experiment regards the comparison between the non-adversarial approach (CNN trained on L1 loss) and the adversarial one. As shown in Table 3, the GAN significantly improves the LPIPS metrics, while obtaining lower performances with the PSNR and SSIM metrics. Once again, as often happens, obtaining better results on the high-level feature metric causes to worsen the performance on the pixel-level metrics.

Table 3: Comparison between adversarial framework and CNN model.

Method	PSNR	SSIM	LPIPS
CNN (2 blocks and L1 loss)	**27.63**	**0.876**	0.285
GAN	23.2	0.815	**0.238**

The second type of experiment, similar to the one described in Sec. 5.2.1, regards comparing different loss functions. In particular, two experiments of this kind were performed: the first compares two GAN models, one trained with a combination of perceptual and adversarial loss functions, referred to as dual loss, while the other trained with a combination of perceptual, adversarial and L1 losses, referred to as triad loss (Table 4), while the second, as in [16], compares two versions of the perceptual loss. More specifically, the second experiment regards comparing the version of perceptual loss calculated on the features before passing through the activation function in the VGG model, with the one calculated on the features extracted after the activation function (Table 5).

Table 4: Comparison between dual and triad loss function.

Method	PSNR	SSIM	LPIPS
GAN (dual loss)	23.2	0.815	**0.238**
GAN (triad loss)	**27.84**	**0.88**	0.28

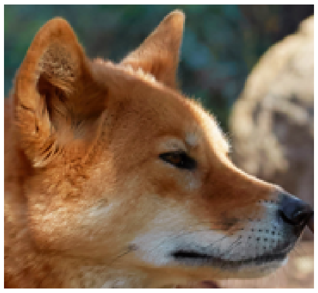
Table 5: Comparison between pre-activation and post-activation perceptual loss function.

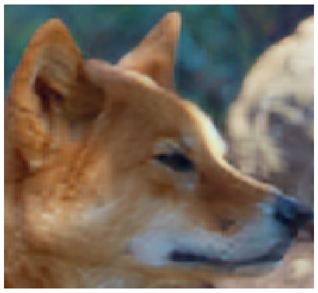
Method	PSNR	SSIM	LPIPS
GAN (pre-activation)	15.94	0.734	0.301
GAN (post-activation)	**23.2**	**0.815**	**0.238**

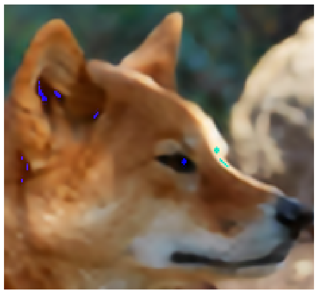
The results in Table 4 highlight how adding the L1 loss, causes the LPIPS metric to decrease and it is probably due to the fact that by adding it, the complexity of the loss function increases. Furthermore, the results in Table 5 show how using the perceptual loss calculated on the features before the activation function decreases the performance of the model. Similarly to the rationale of the latter experiment, as also mentioned in [16], this is probably caused by the fact that by passing through the activation function the features lose information, hence the resulting loss function is simpler. In fact, in [16] the results are opposite, but because, caused by computational limits, the experiments of this work are characterized by significantly shorter training times, which caused not being able to optimize more complex loss functions.

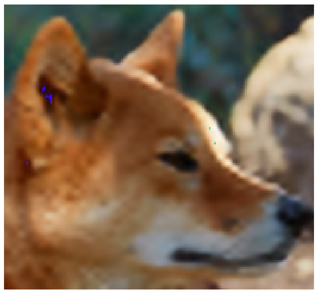
5.2.3. *Qualitative analysis*

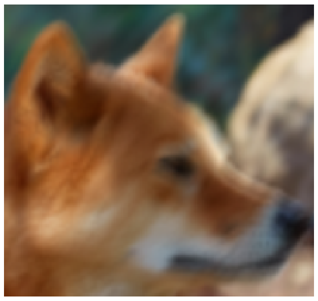
In Fig. 6 the comparison between the output of the different approaches tested in this work is shown. In particular, the comparison is between the ground truth and all the approaches tested: GAN, CNN trained on the L1 loss, CNN trained on the L2 loss, and two baseline models, bilinear interpolation and nearest-neighbor. First, it is clear that Deep learning approaches return significantly better results than non-DL ones like bilinear interpolation and nearest-neighbor. Furthermore, by comparing the outputs of the CNNs trained with L1 and L2 loss functions, there is a less significant but still clear difference. In particular, as already mentioned in Sec. 3.3, the L2 pushes the model to produce smoother and less detailed images than the L1. Similarly, pushed by the perceptual and adversarial loss to produce more realistic images (more details in Sec. 5.3), the GAN further improves the output of the CNN trained with the L1 loss, by producing even more detailed features. Finally, even if significantly better than the CNN, the GAN model is still far from producing high-resolution images that are nearly indistinguishable from the ground truth.


(a) Ground truth


(b) GAN


(c) CNN (L1 loss)


(d) CNN (L2 loss)


(e) Bilinear interpolation

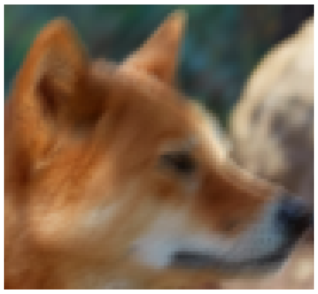

(f) Nearest-neighbor

Fig. 6: Comparison between the output of the approaches tested, baseline models, and ground truth.

5.3. GAN training

As shown in Sec. 5.2, while the GAN significantly improved the results of the CNN, on the other hand, the results are clearly still improvable, and this is mainly caused by the difficulties that occurred in the training stage. In particular, as shown in Fig. 7, while the perceptual loss behaves as expected, the adversarial loss converges nearly immediately to zero, and this is caused by an unbalance between the generator and the discriminator. More specifically, the generator learns right after the first epochs to fool the discriminator, hence zeroing the adversarial loss, and this causes the contribution of the discriminator to be canceled.

6. CONCLUSIONS

In this work, the SR task has been tackled with two main approaches: a PSNR-oriented CNN and a GAN model. The

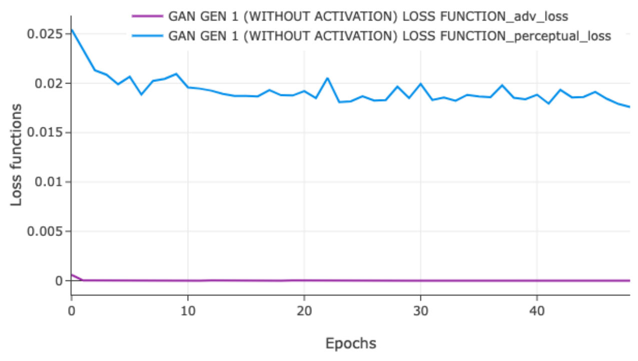


Fig. 7: Plot of the perceptual (in blue) and adversarial (in purple) loss functions values during the GAN training.

CNN significantly improves the baseline results but produces smoothed images that lack detailed features, on the other hand, the GAN improves this aspect but still produces images that are clearly distinguishable from the ground truth. In particular, while the GAN results are satisfactory, further improvements can be done: solving the unbalance problem between the generator and discriminator by training the discriminator for more epochs before training it together with the pre-trained generator, increasing the size of both the generator and discriminator, since the model used showed difficulties in optimizing more complex loss functions, and finally considering other state-of-the-art approaches, like for example Denoising Diffusion Probabilistic Models.

7. REFERENCES

[1] Hayit Greenspan, "Super-resolution in medical imaging," *The computer journal*, vol. 52, no. 1, pp. 43–63, 2009.

[2] Jithin Saji Isaac and Ramesh Kulkarni, "Super resolution techniques for medical image processing," in *2015 International Conference on Technologies for Sustainable Development (ICTSD)*. IEEE, 2015, pp. 1–6.

[3] Liangpei Zhang, Hongyan Zhang, Huanfeng Shen, and Pingxiang Li, "A super-resolution reconstruction algorithm for surveillance images," *Signal Processing*, vol. 90, no. 3, pp. 848–859, 2010.

[4] Tao Lu, Jiaming Wang, Yanduo Zhang, Zhongyuan Wang, and Junjun Jiang, "Satellite image super-resolution via multi-scale residual deep neural network," *Remote Sensing*, vol. 11, no. 13, pp. 1588, 2019.

[5] Tingxing Tim Dong, Hao Yan, Mayank Parasar, and Raun Krisch, "Rendersr: A lightweight super-resolution model for mobile gaming upscaling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3087–3095.

[6] Dengxin Dai, Yujian Wang, Yuhua Chen, and Luc Van Gool, "Is image super-resolution helpful for other vision tasks?," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016, pp. 1–9.

[7] Zhihao Wang, Jian Chen, and Steven CH Hoi, "Deep learning for image super-resolution: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 10, pp. 3365–3387, 2020.

[8] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang, "Learning a deep convolutional network for image super-resolution," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV 13*. Springer, 2014, pp. 184–199.

[9] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[10] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1646–1654.

[11] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 136–144.

[12] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu, "Residual dense network for image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2472–2481.

[13] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee, "Deeply-recursive convolutional network for image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1637–1645.

[14] Ying Tai, Jian Yang, and Xiaoming Liu, "Image super-resolution via deep recursive residual network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3147–3155.

[15] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al., "Photo-realistic single image super-resolution using a generative adversarial network," in

Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4681–4690.

[16] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy, "Esrgan: Enhanced super-resolution generative adversarial networks," in *Proceedings of the European conference on computer vision (ECCV) workshops*, 2018, pp. 0–0.

[17] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan, "Real-esrgan: Training real-world blind super-resolution with pure synthetic data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 1905–1914.

[18] Alexia Jolicoeur-Martineau, "The relativistic discriminator: a key element missing from standard gan," *arXiv preprint arXiv:1807.00734*, 2018.

[19] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte, "Swinir: Image restoration using swin transformer," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 1833–1844.

[20] Wenbo Li, Xin Lu, Jiangbo Lu, Xiangyu Zhang, and Jiaya Jia, "On efficient transformer and image pre-training for low-level vision," *arXiv preprint arXiv:2112.10175*, 2021.

[21] Andreas Lugmayr, Martin Danelljan, Radu Timofte, Kang-wook Kim, Younggeun Kim, Jae-young Lee, Zechao Li, Jinshan Pan, Dongseok Shim, Ki-Ung Song, Jinhui Tang, Cong Wang, and Zhihao Zhao, "Ntire 2022 challenge on learning the super-resolution space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2022, pp. 786–797.

[22] Hshmat Sahak, Daniel Watson, Chitwan Saharia, and David Fleet, "Denoising diffusion probabilistic models for robust image super-resolution in the wild," *arXiv preprint arXiv:2302.07864*, 2023.

[23] Jonathan Ho, Ajay Jain, and Pieter Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[24] Andreas Lugmayr, Martin Danelljan, Luc Van Gool, and Radu Timofte, "Srflow: Learning the super-resolution space with normalizing flow," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*. Springer, 2020, pp. 715–732.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[26] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[27] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee, "Deep multi-scale convolutional neural network for dynamic scene deblurring," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3883–3891.

[28] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. pmlr, 2015, pp. 448–456.

[29] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, Eds. 2014, vol. 27, Curran Associates, Inc.

[30] Justin Johnson, Alexandre Alahi, and Li Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*. Springer, 2016, pp. 694–711.

[31] Alexey Dosovitskiy and Thomas Brox, "Generating images with perceptual similarity metrics based on deep networks," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds. 2016, vol. 29, Curran Associates, Inc.

[32] Eirikur Agustsson and Radu Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 126–135.

[33] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[34] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 586–595.