Computer Vision project

Learning the Super Resolution Space

D'Urso Mattia

July 6, 2022

Contents

1 Introduction

Within the Computer Science area, one of the topics of high interest has always been Artificial Intelligence (AI). We define AI as the simulation of human intelligence computed by machines. Specific applications of AI include natural language processing, speech recognition, and computer vision (CV). The latter is one of the most-studied topics, we can define it as a subset of AI that deals in teaching machines how to understand images.

The first problem CV tried to solve was to make a robot "see" the world by connecting a video camera to a computer, around the 1960s at MIT. Today the field made several steps further, and there are many more problems that the CV aims to solve. Among these, we find image classification, the task of deciding whether class images belong, object detection, which is finding and classifying objects within an image, and semantic segmentation, which consists of grouping the parts of an image that belong to the same class of objects. With the advancement of technology, new challenges presented themselves, like the problem of the upscaling image. It is about reconstructing a meaningful high-resolution image (HR) from a low-resolution image (LR). This task is called Image Super-Resolution (SR). SR is a fundamentally ill-posed problem. In fact, for a given LR image, there exist infinitely many compatible HR predictions. Nevertheless, SR has several real-world applications such as surveillance (e.g., face recognition in upscaled images) and media sharing (e.g., sending LR images and upscaling on the fly). This issue is of interest also to large companies like NVIDIA that developed the deep learning super sampling (DLSS) [3] to use it on its GPUs when rendering video game frames. The idea is to render the game at LR (e.g., 1920×1080) and then use DLSS to upscale images to HR (e.g., 3840×2160). In this way, it is possible to play HR games with less computational power.

Regarding the SR problem, there are also some public competitions, such as the "Learning the Super-Resolution Space" challenge in 2021, held as part of the $6thth$ edition of the New Trends in Image Restoration and Enhancement (NTIRE) workshop, in conjunction with the 2021 Conference on Computer Vision and Pattern Recognition (CVPR). The goal of the "Learning the Super-Resolution Space" challenge is to develop an SR method that can upscale downscaled images sampled from the space of plausible HR images. More formally, given an HR image IMG sampled from a space of images S_{IMG} and downscaled to an LR image img, the aim is to find a method f that

$$IMG' = f(img)$$

where IMG' is an HR image and $IMG' \approx IMG$. The challenge contains two tracks, targeting 4X and 8X super-resolution respectively. The running methods shall achieve the highest possible photo-realism and be consistent with the input low-resolution image. The methods are tested with the following metrics [9].

– **Diversity score**: This metric aims to measure how well the SR image of ground truth is represented in the predicted space. From the predicted image IMG' and the original image IMG, k random patches are sampled y_k $in\mathbb{R}^{N\times N\times 3}$ and the similarity is calculated with a distance metric d. To obtain the significant diversity that the samples represent, testers calculate by how much the minimum distance from the reference patch decreases when M samples are used,

$$S_M = \frac{1}{\overline{d}_M}(\overline{d}_M - \frac{1}{K}\sum_{k=1}^{K} min\{d(y_k, \hat{y}_k^i)\}_{i=1}^{M}) \tag{1}$$

where

$$\overline{d}_M = min\{\frac{1}{K}\sum_{k=1}^{K} d(y_k, \hat{y}_k^i)\}_{i=1}^{M} \tag{2}$$

Notice that $S_M \in [0, 1]$ where $S_M = 0$ means no diversity (deterministic predictions) and $S_M = 1$ means that the ground-truth HR image was exactly captured by one of the generated samples.

– **LR-PSNR**: This metric aims to measure the amount of information retained in the SR image compared to the LR image measuring the LR-PSNR. It is calculated as the PSNR between the input LR image and the predicted sample resized with the given bicubic kernel. The goal of this challenge is to obtain an LR-PSNR of at least 45dB.

– **Mean Opinion Rank**: This metric aims to assess the degree of photorealism of an image. This task is performed by humans on three different 80×80 cutouts of the 100 images in the test set. Each task is performed by five different users. Finally, the user is asked to rank the cutouts according to their opinion.

The participating methods are ranked according to each metric. These ranks then are combined into a final score. The first method wins the challenge.

After AlexNet won the ImageNet Large Scale Visual Recognition Challenge in 2012, deep learning (DL) methods became increasingly popular in the CV field because they performed significantly better than methods used up to that point in a wide set of tasks. Following this trend, during NTIRE 2021 the most common approach was the use of DL methods. In particular, Generative Adversarial Networks (GANs), Variational AutoEncoders (VAEs), GANs trained with implicit maximum likelihood estimation (IMLE), and Flow models were used.

DA RISCRIVERE In section 2 we describe the State-of-the-art architectures for SR. In section 3 we describe in detail how the flow models for SR work. In section 4 we run some tests with the aforementioned flow models. Finally, in section 5 se we draw the conclusions.

2 State-of-the-art architectures for Super Resolution

ADD INTRO
ADD SOME CITE TO SOTA OF FOLLOWING MODELS

2.1 Generative Adversarial Networks

The Generative Adversarial Networks was introduced in [6] in 2014. Goodfellow's team proposed two networks that act as generators and discriminators. The former network aims to generate data from random noise, while the second method must decide whether the data generated is realistic or not. The two models are trained alternately, so when the generator is trained, the discriminator is kept constant and vice versa.

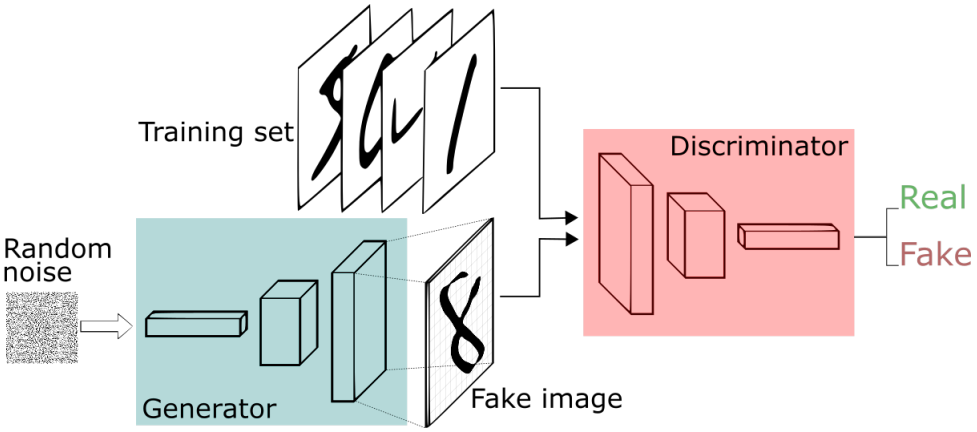


Figure 1: Generative Adversarial Networks architecture applied to images.

2.2 Variational Auto-Encoders

The Variational Auto-Encoder (VAE) architecture was introduced in [1]. VAE is an autoencoder that, unlike AutoEncoders, encodes data in two vectors that are means (μ) and standard deviations (σ). During the training, those vectors are constrained to ensure that the latent distribution they describe has some properties. Some random samples from the latent distribution are given to the decoder to generate new data.

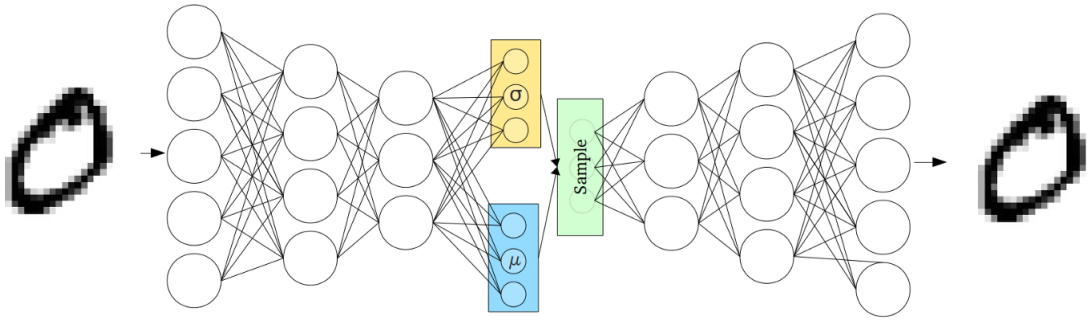


Figure 2: Variational Auto-Encoders architecture applied to an image.

2.3 Implicit Maximum Likelihood Estimator

Implicit Maximum Likelihood Estimator (IMLE) is a technique to train implicit models (e.g., GANs) introduced in [7]. Assuming that we do not know the distribution of the model and have only a finite set of i.i.d. samples, more samples are expected to be found near the data examples than elsewhere. Therefore, the aim is to minimize the distance from each data example to the nearest sample. The IMLE is defined as

$$\hat{\theta}_{IMLE} = arg\min_{\theta} \mathbb{E}_{\tilde{x}_1^\theta,\ldots,\tilde{x}_m^\theta}\left[\sum_{i=1}^{n} \min_{j\in[m]} ||\tilde{x}_j^\theta - x_i||_2^2\right] \tag{3}$$

where $x_1, ...x_n$ is the set of n samples and P_θ is some unknown parameterized probability with density p_θ. $\tilde{x}_1^\theta, ...\tilde{x}_m^\theta$ is the set of i.i.d. samples from P_θ with $m \geq n$.

2.4 Flow models

Flow models are particular architectures in which data (usually images) flows in both directions. They aim to learn the distribution of HR data $p_{y|x}(y|x,\theta)$ (where x is the LR sample, y is the HR sample and θ are the parameters of the function f), and not only an injective correspondence $x \rightarrow y$. In this way, the model can sample multiple HR samples from the learned plausible distribution given one LR sample. The model is also required to be invertible w.r.t y, which means given the latent encoding $z = f_\theta(y;x)$ we can reconstruct y as $y = f_\theta^{-1}(z;x)$.

3 Flow models for Super Resolution

ADD INTRO

Early DL approaches trained feed-forward architectures using L_1[1] or L_2[2] losses. The problem with these methods is they reduce high frequencies resulting in blurred images because of the average nature of the aforementioned loss functions. To address this problem more recent approaches introduced adversarial training and perceptual losses functions (work by summing all the squared errors between all the pixels and taking the mean). Adversarial methods obtain better results

[1] $L_1(y,\hat{y}) = \sum_{i=1}^{n} |y - \hat{y}|$
[2] $L_2(y,\hat{y}) = \sum_{i=1}^{n} (y - \hat{y})^2$

while they can compute a single prediction for an image. They also use multiple loss functions during the training phase with careful parameter tuning.

3.1 Flow models' loss function

Flow models are trained with a single loss function that is negative log-likelihood[3] (NLL), which simplifies training. The core idea of normalizing flow is to parametrize the distribution $p_{y|x}$ using a neural network f_θ. The distribution $p_{y|x}$ can be computed as

$$p_{y|x}(y|x,\theta) = p_z(f_\theta(y;x))\left|det\frac{\partial f_\theta}{\partial y}(y;x)\right| \tag{4}$$

which is derived by applying the change-of-variables formula for densities, where the second factor is the resulting volume scaling given by the absolute value of the determinant of the Jacobian $\frac{\partial f_\theta}{\partial y}$. Equation 4 allows us to train the network by minimizing the negative log-likelihood for training samples pairs (x, y) as follows.

$$\mathcal{L}(\theta;x,y) = -\log p_{y|x}(y|x,\theta) = -\log p_z(f_\theta(y;x)) - \log\left|det\frac{\partial f_\theta}{\partial y}(y;x)\right| \tag{5}$$

In order to achieve a tractable expression of the second term in equation 5, the neural network f_θ is decomposed into a sequence of N invertible layers $h^{n+1} = f_\theta^n(h^n, g_\theta(x))$, where $h^0 = y$ and $h^N = z$. We let the LR image to first be encoded by a shared deep CNN $g_\theta(x)$ that extracts a rich representation suitable for conditioning in all flow-layers. By applying the chain rule along with the multiplicative property of the determinant [4], the NLL objective in equation 4 can be expressed as

$$\mathcal{L}(\theta;x,y) = -\log p_z(z) = -\sum_{n=0}^{N-1} -\log\left|det\frac{\partial f_\theta^n}{\partial h^n}(h^n;g_\theta(x))\right| \tag{6}$$

The following flow models use equation 6 for the training.

3.2 SRFlow: Learning the Super-Resolution Space with Normalizing Flow

The Super-Resolution using Normalizing Flow model (SRFlow) [12] consists of the invertible flow network f_θ and the LR encoder g_θ. The flow network is organized into L levels, each operating at a resolution of $\frac{H}{2^l} \times \frac{W}{2^l}$, where $l \in \{1, ..., L\}$ is the level number and $H \times W$ is the HR resolution (e.g., let an input image be $3 \times 512 \times 512$, at level 2 will be $48 \times 128 \times 128$, at level 4 will be $768 \times 32 \times 32$, and so on.). Each level itself contains K number of flow steps. To achieve tractable expressions the neural network f_θ is decomposed into a sequence of N layers $h^{n+1} = f_\theta^n(h^n; g_\theta(x))$, where $h^0 = y$ and $h^N = z$. These layers can be easily inverted and derived.

The LR encoder g_θ is a Residual-in-Residual Dense Blocks (RRDB) [5] without any batch normalization layers and the final upscaling layers. To capture a richer representation of the LR image at multiple levels, the researchers additionally concatenate the activations after each RRDB block to form the final output of g_θ.

f_θ is organized as follows.

- The **squeeze** layer: this layer provides an invertible means to halving the resolution by reshaping each spatial 2×2 neighborhood into the channel dimension (e.g., $3 \times 64 \times 64$ into $12 \times 32 \times 32$). This operation is done to capture correlations and structures over larger distances. Figure 3 graphically shows the operation.
- The **Transition Step**: this block normalizes and permutes the input according to a learned method.
  - **Actnorm**: This provides a channel-wise normalization through a learned scaling and bias. This layer is ketp in its standard un-conditional form [2].

[3] $\mathcal{L}(\theta;x,y) = -\log p_{y|x}(y|x,\theta)$

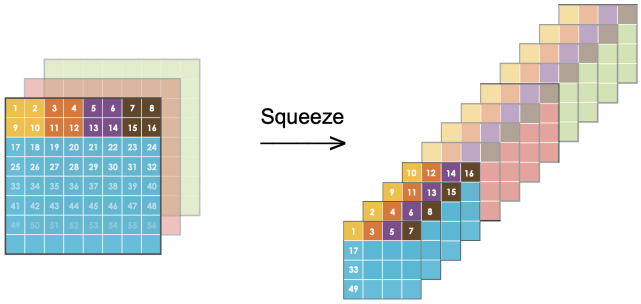


Figure 3: Visualization of the Squeeze operation.

  - Invertible **1 × 1 Convolution**: this layer is a 1 × 1 convolution with equal number of input and output channels, which is a generalization of a permutation operation [2]. According to [10], the ordering of the dimensions can be critical to the training of the model.

- The **Conditional Flow Step**: this block passes the input in another Actnorm layer and 1 × 1 convolutional layer. Then, add the conditional information in two steps.

  - **Affine Injector**: this layer directly affects all channels and spatial locations in the activation map h^n. This is achieved by predicting an element-wise scaling and bias using only the conditional encoding $u = g_\theta(x)$,

$$h^{n+1} = \exp\left(f_{\theta,s}(u)\right) \cdot h^n + f_{\theta,b}(u) \tag{7}$$

    where $f_{\theta,s}$ and $f_{\theta,b}$ can be any network, SRFlow uses a block that consists of 3 × 3 convolution, actnorm, ReLU, 1 × 1 convolution, actnorm, ReLU, and 3 × 3 conv. The exp power is used to have positive values in easy differentible way.

  - **Conditional Affine Coupling**: this layer allows applying complex unconstrained conditional learned functions that act on the normalizing flow, without harming its invertibility. The layer splits on half the input $\text{split}(h^n) = h^n_A, h^n_B$ along the channel dimension. Then the following transformation on the bypassed half h^n_A and conditional features u is computed

$$h^{n+1} = \begin{cases} h^{n+1}_A = h^n_A, \\ h^{n+1}_B = \exp\left(f_{\theta,s}(h^{n+1}_A; u)\right) \cdot h^n_B + f_{\theta,b}(h^{n+1}_A; u) \end{cases} \tag{8}$$

    The network architectures of $f_{\theta,s}$ and $f_{\theta,b}$ are similar to those of the Affine Injector, described above. The only difference is that u is resized to the resolution of h^n_A, then it is concatenated to h^{n+1}_A along the channel dimension. Note that h^{n+1} is a couple.

- **Split** layer: this layer splits the given input in two halfs along the channel dimension.

Figure 4 graphically shows a representation of the model.

3.3 SRFlow-DA: Super-Resolution using Normalizing Flow with Deep Convolutional Block

Super-Resolution using Normalizing Flow with Deep convolutional block in the Affine couplings (SRFlow-DA) [13] model is an architecture based on SRFlow that increases the SR quality while reducing the model size and the training time. The idea is to improve the perfomance enlarging receptive field[4] size in the single flow step. The SRFlow model has an equivalent RF of 5 × 5 while the SRFlow-DA model has a RF of 13 × 13, plus the number of flow steps has been reduced from 16 to 6. These two changes reduced the number of parameters from 22.8M (SRFlow) to

[4]A Receptive Field (RF) is defined as the size of the region in the input that produces the output feature. [14] shows applying a stack of smaller filters is equivalent and more efficent than a bigger one

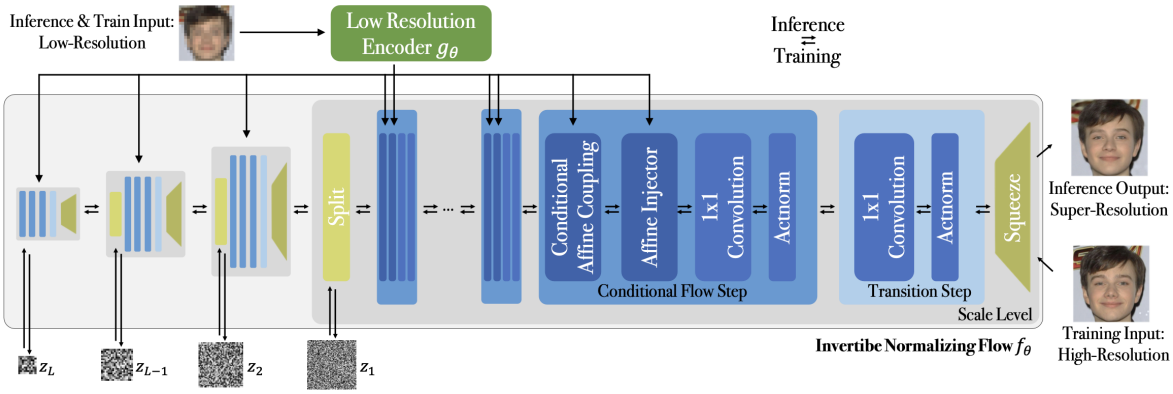
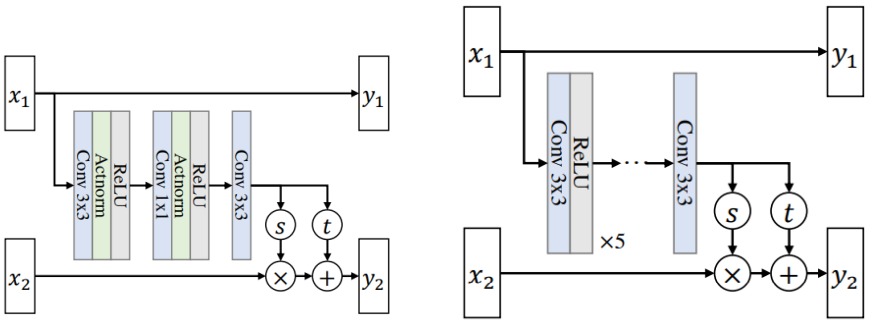


Figure 4: Super-Resolution using Normalizing Flow's architecture.

8.7M (SRFlow-DA) improving training and inference time. To achive those results, the researchers stacked 3×3 convolutional layers followed by ReLU activation except for the last convolutional layer. s and t are emprically estimated coefficents that improves the perfomances. Figure 5 highligths the differences.



(a) Affine coupling in SRFlow.

(b) Affine coupling in SRFlow-DA.

Figure 5: Comparison between the Affine Coupling layers of the two architectures.

3.4 Noise Conditional Flow Model for Learning the Super-Resolution Space

Noise Conditional Flow model for Super-Resolution (NCSR) [8] is an architecture based on SRFlow that improves the diversity of the predicted images by adding noise. In particular, the researchers add noise to the training data and through a new layer called Noise Conditional Layer. NCSR aims to solve also the dimension mismatch problem[5] typical of Flow models. The architectural difference can be found in the Conditional Flow step. NCSR adds the Noise Conditional Layer in 4^{th} position obtaining a block with five elements, as shown in figure 6.

Adding *noise* to the input data x results in a variation of the distribution of the latent data z during the inference process. To address that the *noise* used for the input x is resized and added to the LR image y as follows.

$$x^+ = x + noise$$
$$y^+ = y + noise^-$$
$$f^{-1}(x^+|y^+) = z$$

[5]This problem concerns the inability to properly train the model if the size of the data distribution does not match that of the underlying target distribution. [11]

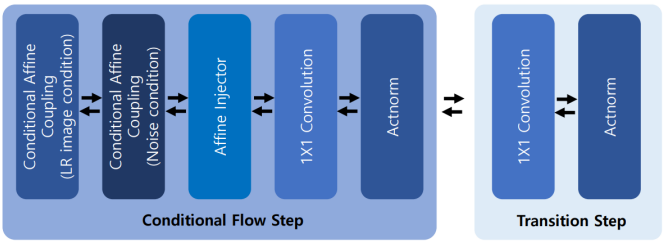


Figure 6: The Conditional Flow block.

where, x is an HR image, f is the invertible model that maps z to x and y is a LR image.

The second improvement is the introduction of the Noise Conditional Layer, which is a Conditional Affine Coupling layer that uses *noise* instead of u. The idea is to inform noise to the model structure to avoid artifacts caused by the noise injection on x. The Noise Affine Conditional layer is designed as follows.

$$h^{n+1} = \begin{cases} h_A^{n+1} = h_A^n, \; h_B^{n+1} = \exp\left(f_{\theta,s}(h_A^{n+1}; noise)\right) \cdot h_B^n + f_{\theta,b}(h_A^{n+1}; noise) \end{cases} \tag{9}$$

The third contribution is about the dimension mismatch problem. NCSR uses the same approach of [11] to overcome it by estimating a perturbed data distribution that is conditioned on noise parameters. The key is to add noise that is obtained from a randomly selected distribution and to use these distribution parameters as conditions. In particular random value c is obtained from uniform distribution $U(0, M)$, where M is 0.1, as [11] did. Next, set the to noise distribution $N(0, \Sigma)$, where $\Sigma = c^2 I$. Then, noise is sampled from $N(0, \Sigma)$.

Figure 4 graphically shows a representation of the model.

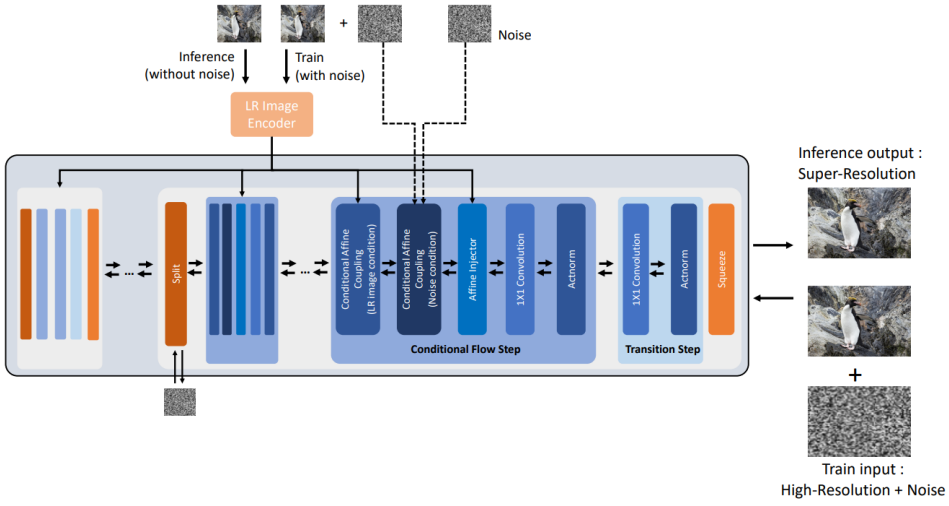


Figure 7: Noise Conditional Flow for Super-Resolution's architecture.

3.5 Metrics comparison

The table 1 and table 2 show the 4× and 8× results of the models on the 100 validation images of the DIV2K dataset reported in the papers. The arrows near the metric's name indicate the direction for a better result. Best results are highlighted in bold only if the value is reported for all the models. The * means the value is not reported in the original model's paper but in another paper for comparison purposes. Inference time is measured by [13] for generating an output image size of 1920×1080. Note that the training times and the inference times are measured on NVIDIA GeForce RTX 2080 TI.

ADD COMMENT

Model	Diversity↑	LPIPS↓	LR-PSNR↑	Params	Train/Infer. time
SRFlow	25.24*	0.120	50.64	22.8M*	120h*/1.98s*
SRFlow-DA	23.55	0.121	**50.88**	8.7M	33h/1.18s
NCSR	**26.72**	**0.119**	50.75	-	-

Table 1: 4× results on DIV2K validation set.

Model	Diversity↑	LPIPS↓	LR-PSNR↑	Params	Train/Infer. time
SRFlow	25.28*	0.272	50.09	34.1M*	120h* / 1.97s*
SRFlow-DA	23.45	**0.261**	**50.91**	13.3M	47h / 1.01s
NCSR	**26.80**	0.278	44.55	-	-

Table 2: 8× results on DIV2K validation set.

4 Experiments ?

5 Conclusions

References

[1] Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." arXiv preprint arXiv:1312.6114 (2013).

[2] Kingma, Durk P., and Prafulla Dhariwal. "Glow: Generative flow with invertible 1x1 convolutions." Advances in neural information processing systems 31 (2018).

[3] Deep Learning Super Sampling-Technology (DLSS) | NVIDIA `https://www.nvidia.com/de-at/geforce/technologies/dlss/`, last access 20 June 2022.

[4] Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using real nvp." arXiv preprint arXiv:1605.08803 (2016).

[5] Wang, Xintao, 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, pp. 0-0. 2018.

[6] Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." Advances in neural information processing systems 27 (2014).

[7] Li, Ke, and Jitendra Malik. "Implicit maximum likelihood estimation." arXiv preprint arXiv:1809.09087 (2018).

[8] Kim, Younggeun, and Donghee Son. "Noise conditional flow model for learning the super-resolution space." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 424-432. 2021..

[9] Lugmayr, Andreas, Martin Danelljan, and Radu Timofte. "NTIRE 2021 learning the super-resolution space challenge." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 596-612. 2021.

[10] Vinyals, Oriol, Samy Bengio, and Manjunath Kudlur. "Order matters: Sequence to sequence for sets." arXiv preprint arXiv:1511.06391 (2015).

[11] Kim, Hyeongju, Hyeonseung Lee, Woo Hyun Kang, Joun Yeop Lee, and Nam Soo Kim. "Softflow: Probabilistic framework for normalizing flow on manifolds." Advances in Neural Information Processing Systems 33 (2020): 16388-16397.

[12] Lugmayr, Andreas, Martin Danelljan, Luc Van Gool, and Radu Timofte. "Srflow: Learning the super-resolution space with normalizing flow." In European conference on computer vision, pp. 715-732. Springer, Cham, 2020.

[13] Jo, Younghyun, Sejong Yang, and Seon Joo Kim. "Srflow-da: Super-resolution using normalizing flow with deep convolutional block." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 364-372. 2021.

[14] Luo, Wenjie, Yujia Li, Raquel Urtasun, and Richard Zemel. "Understanding the effective receptive field in deep convolutional neural networks." Advances in neural information processing systems 29 (2016).