Journey Towards Tiny Perceptual Super-Resolution

Royson Lee[1*][0000−0002−6716−7994], Łukasz Dudziak[1][0000−0003−4929−265X], Mohamed Abdelfattah[1][0000−0002−4568−8932], Stylianos I. Venieris[1][0000−0001−5181−6251], Hyeji Kim[1][0000−0002−2256−5729], Hongkai Wen[1,2][0000−0003−1159−090X], and Nicholas D. Lane[1,3][0000−0002−2728−8273]

[1] Samsung AI Center, Cambridge, UK
[2] University of Warwick
[3] University of Cambridge
*royson.lee@samsung.com

Abstract. Recent works in single-image perceptual super-resolution (SR) have demonstrated unprecedented performance in generating realistic textures by means of deep convolutional networks. However, these convolutional models are excessively large and expensive, hindering their effective deployment to end devices. In this work, we propose a neural architecture search (NAS) approach that integrates NAS and generative adversarial networks (GANs) with recent advances in perceptual SR and pushes the efficiency of small perceptual SR models to facilitate on-device execution. Specifically, we search over the architectures of both the generator and the discriminator sequentially, highlighting the unique challenges and key observations of searching for an SR-optimized discriminator and comparing them with existing discriminator architectures in the literature. Our tiny perceptual SR (TPSR) models outperform SRGAN and EnhanceNet on both full-reference perceptual metric (LPIPS) and distortion metric (PSNR) while being up to 26.4× more memory efficient and 33.6× more compute efficient respectively.

1 Introduction

Single-image super-resolution (SR) is a low-level vision problem that entails the upsampling of a low-resolution (LR) image to high-resolution (HR). Currently, the highest-performing solutions to this problem are dominated by the use of convolutional networks, which leave limited space for traditional approaches [6, 26]. Nevertheless, with the SR task being inherently ill-posed, *i.e.* a given LR image can correspond to many HR images, SR methods follow different approaches. In this respect, existing supervised solutions can be mainly grouped into two tracks based on the optimization target: distortion and perceptual quality.

To improve perceptual quality, Ledig *et al.* [27] first empirically showed that the use of generative adversarial networks (GANs) [15] results in upsampled images that lie closer to the natural-image manifold. This observation was later backed theoretically [5] through a proof that using GANs is a principled approach

to minimize the distance between the distribution of the upsampled image and that of natural images. Until today, there have been several works focusing on using GANs for perceptual SR, leading to prominent networks such as ESRGAN [48] and EnhanceNet [39].

Although these proposed perceptual SR solutions achieve promising results, they remain extremely resource-intensive in terms of computational and memory demands. Existing *efficient* SR solutions [22, 21, 47, 11, 10, 1, 8, 9, 44, 28], on the other hand, are mostly optimized for distortion metrics, leading to blurry results. Hence, in this work, we pose the following question: **Can we build an efficient and constrained SR model while providing high perceptual quality?**

In order to build such SR models, we apply neural architecture search (NAS). In particular, we run NAS on both the discriminator as well as the generator architecture. To the best of our knowledge, our study is the first to search for a discriminator in SR, shedding light on the role of the discriminator in GAN-based perceptual SR. Our contributions can be summarized as follows:

- We adopt neural architecture search (NAS) to find efficient GAN-based SR models, using PSNR and LPIPS [52] as the rewards for the generator and discriminator searches respectively.
- We extensively investigate the role of the discriminator in training our GAN and we show that both existing and new discriminators of various size and compute can lead to perceptually similar results on standard benchmarks.
- We present a tiny perceptual SR (TPSR) model that yields high-performance results in both full-reference perceptual and distortion metrics against much larger full-blown perceptual-driven models.

2 Background & Related Work

In SR, there is a fundamental trade-off between distortion- and perceptual-based methods [5]; higher reconstruction accuracy results in a less visually appealing image and vice versa. Distortion-based solutions [46, 29, 53] aim to improve the fidelity of the upsampled image, *i.e.* reduce the dissimilarity between the upsampled image and the ground truth, but typically yield overly smooth images.

Perceptual-based methods [27, 32, 48, 39], on the other hand, aim to improve the visual quality by reducing the distance between the distribution of natural images and that of the upsampled images, resulting in reconstructions that are usually considered more appealing. These perceptual SR models are usually commonly evaluated using full-reference methods such as LPIPS [52] or no-reference methods such as NIQE [34], BRISQUE [33], and DIIVINE [36], which are designed to quantify the deviation from natural-looking images in various domains.

Hand-crafted Super-resolution Models. Since the first CNN was proposed for SR [10] there has been a surge of novel methods, adapting successful ideas from other high- and low-level vision tasks. For instance, state-of-the-art distortion-driven models such as EDSR [29], RDN [54], and RCAN [53] use residual blocks [17], and attention mechanisms [2], respectively, to achieve competitive fidelity results. Independently, state-of-the-art perceptual-driven SR models have

been primarily dominated by GAN-based models such as SRGAN [27] (which uses a combination of perceptual loss [24] and GANs), and ESRGAN [48] (which improves on SRGAN by employing the relativistic discriminator [25]).

Towards efficiency, Dong *et al.* [11] and Shi *et al.* [42] proposed reconstructing the upsampled image at the end of a network, rather than at its beginning, to reduce the computational complexity during feature extraction. Since then, numerous architectural changes have been introduced to obtain further efficiency gains. For instance, group convolutions [17] were adopted by Ahn *et al.* [1], channel splitting [30] by Hui *et al.* [22, 21], and inverse sub-pixel convolutions by Vu *et al.* [47], all of which significantly reduced the computational cost.

Similar to one of our goals, Chen *et al.* [7] explored how the discriminator would affect performance in SR by introducing two types of attention blocks to the discriminator to boost image fidelity in both lightweight and large models. Unlike their approach, we optimize for a perceptual metric and explore a wide range of discriminators using standard popular NN operations instead.

Neural Architecture Search for Super-resolution. Recent SR works aim to build more efficient models using NAS, which has been vastly successful in a wide range of tasks such as image classification [57, 55, 38], language modeling [56], and automatic speech recognition [12]. We mainly focus on previous works that adopt NAS for SR and refer the reader to Elsken *et al.* [13] for a detailed survey on NAS. Chu *et al.* [9, 8] leveraged both reinforcement learning and evolutionary methods for exploitation and exploration respectively, considering PSNR, FLOPs and memory in a multi-objective optimization problem. Song *et al.* [44] argued that searching for arbitrary combinations of basic operations could be more time-consuming for mobile devices, a guideline that was highlighted by Ma *et al.* [30]. To alleviate that, they proposed searching using evolutionary methods for hand-crafted efficient residual blocks. Although we agree with their approach to utilize platform-specific optimizations, we decided to keep our approach platform-agnostic and only consider models that fit in the practical computational regime based on the models used in the current SoTA SR mobile framework [28]. Most importantly, our work differs from previous NAS with SR approaches as we focus on optimizing the perceptual quality rather than the fidelity of the upsampled images.

Neural Architecture Search for GANs. Recently, Gong *et al.* [14] presented a way of incorporating NAS with GANs for image generative tasks, addressing unique challenges faced by this amalgamation. Combining NAS with GANs for SR, on the other hand, presents its own set of challenges. For example, as perceptual SR only requires one visually appealing solution, mode collapse might be favorable so their proposed dynamic-resetting strategy is not desirable in our context. Another major difference is that GAN-based methods for SR usually start with a pre-trained distortion model, avoiding undesired local optima and allowing GAN training to be more stable with high-fidelity input images. Therefore, naively applying their approach is not suitable for the task. With fewer restrictions, we are able to search for a discriminator as opposed to manually tuning it to fit the generator.

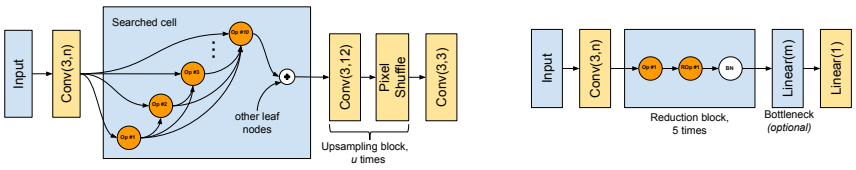

Fig. 1. Structure and search space of the generator (left) and discriminator (right). Orange nodes represent operations which are selected by the controller from the set of available candidates. In the case of the generator, the controller additionally selects only one of the incoming edges as input for each node and, after connections are selected, all leaf nodes are added together to create the cell's output. `Linear`(n) represents a linear layer with n output units. Operations in yellow blocks are fixed.

3 Searching for Tiny Perceptual SR

In the proposed scheme, we extend the original REINFORCE-based NAS framework [56] in order to search for a GAN-based super-resolution model. As a first step, we split the process into two stages. First, we search only for the best generator, using a selected distortion metric to assess different architectures. Next, we utilize the best found model and search for a matching discriminator which would maximize the generator's performance on a selected perceptual metric. Although the same backbone algorithm is used in both cases to conduct the search, the differences between distortion- and GAN-based training require us to approach the two stages with a dedicated methodology, addressing the respective challenges in critical design decisions, including defining the search space and generating reward signals.

We begin with a short introduction to REINFORCE and NAS in Section 3.1 and continue to discuss the details related to the specific use-case of perceptual SR. The skeleton models for both the generator and the discriminator are shown in Figure 1 and the search methodology for both of them is presented in Sections 3.2 and 3.3 respectively, with a summary shown in Algorithm 1.

3.1 Searching Algorithm

We can formulate our NAS problem in a generic way as:

$$\begin{gathered} \mathbb{S} = \mathbb{O}_1 \times \mathbb{O}_2 \times \cdots \times \mathbb{O}_n \\ E : \mathbb{S} \to \mathbb{R} \\ s^\star = \underset{s \in S}{\operatorname{argmax}}\ E(s) \end{gathered} \tag{1}$$

where $\mathbb{S}$ is a *search space* constructed from n independent *decisions*, $\mathbb{O}_i$ is a set of available *options* for the i-th decision, and E is a selected *evaluation function* which we aim to optimize.

Usually, E is implemented as a sequence of steps: construct a model according to the selected options s, train and evaluate it, and return its performance. In

Algorithm 1: A summary of the proposed two-stage approach to searching for a perceptually-good compact SR model

Input: search space for the generator $\mathbb{S}_G$ and discriminator $\mathbb{S}_D$, maximum number of steps when searching for generator T_G and discriminator T_D, Mult-Adds limit for the generator f
Output: trained perceptual model $\mathbf{G}^{\bullet}_{\text{best}}$

Function `search`$(\mathbb{S}, T, E)$**:**
  $s^* \leftarrow$ None
  $\theta \sim \mathcal{N}$
  for $t \leftarrow 0$ **to** T **do**
    $s_t \sim \pi_{\theta,\mathbb{S}}$
    $m_t \leftarrow E(\mathbf{s}_t)$
    if $m_t =$ None **then**
      go back to line 5
    end
    update s^* using m_t
    $\theta \leftarrow$ update θ using $\nabla_\theta \log \pi_{\theta,\mathbb{S}}(s_t) R(m_t)$
  end
  return s^*
End Function

Function $E_G(s)$**:**
  $\mathbf{G} \leftarrow$ construct model according to s and initialize its weights with the cached ones
  $f_s \leftarrow$ calc Mult-Adds required to run $\mathbf{G}$
  if $f_s > f$ **then**
    return None
  end
  $m \leftarrow$ train and evaluate $\mathbf{G}$ on the proxy distortion task
  update cached weights according to Eq. 5
  return m
End Function

$s^*_G \leftarrow$ `search`$(\mathbb{S}_G, T_G, E_G)$
$\mathbf{G}_{\text{best}} \leftarrow$ construct model using s^*_G, initialize from cache, and train on the full dist. task
Function $E_D(\mathbf{s})$**:**
  $\mathbf{D} \leftarrow$ construct discriminator according to $\mathbf{s}$
  return performance of $\mathbf{G}$ on the proxy perc. task after fine-tuning using $\mathbb{D}$
End Function

$s^*_D \leftarrow$ `search`$(\mathbb{S}_D, T_D, E_D)$
$\mathbf{D}_{\text{best}} \leftarrow$ construct discriminator according to s^*_D
$\mathbf{G}^{\bullet}_{\text{best}} \leftarrow$ fine-tune $\mathbf{G}_{\text{best}}$ with $\mathbf{D}_{\text{best}}$ on the full perceptual task

our case specifically, E represents a trained model's performance on a validation set – see the following sections for the details about training and evaluation of different models. Because training takes an excessive amount of time and it is hard to predict the performance of a model without it, brute-forcing the optimization problem in Eq. (1) quickly becomes infeasible as the number of elements in $\mathbb{S}$ increases. Therefore, a standard approach is to limit the search process to at most T models (steps), where T is usually decided based on the available time and computational resources. Given a sequence of T architectures explored during the search $\tau(T) = (s_1, s_2, \cdots, s_T)$, we can approximate the optimization problem in Eq. (1) with its equivalent over the values in $\tau(T)$:

$$s^\star \approx s^* = \underset{s \in \tau(T)}{\operatorname{argmax}} \; E(s) \tag{2}$$

We then use REINFORCE [50] to guide the search and ensure that, as T increases, s_T optimizes E thus providing us with a better approximation. More specifically, we include a probabilistic, trainable policy π_θ (a controller) which, at each search step $t = 1, 2, \cdots, T$, is first sampled in order to obtain a candidate structure s_t and then updated using $E(s_t)$. We use the following standard formulation to optimize this policy:

$$\begin{gathered} J(\theta) = \mathbb{E}_{s\sim\pi_\theta} R(E(s)) \\ \theta_{t+1} = \theta_t + \beta \nabla_{\theta_t} J(\theta_t) \\ \nabla_{\theta_t} J(\theta_t) \approx \nabla_{\theta_t} \log \pi_{\theta_t}(s_t) R(E(s_t)) \\ \underset{\theta}{\operatorname{argmax}}\, J(\theta) \approx \theta_T \end{gathered} \tag{3}$$

where $R : \mathbb{R} \to \mathbb{R}$ is a *reward function* used to make values of E more suitable for the learning process (*e.g.* via normalization), and β is a learning rate. Please refer to the supplementary material for further details.

3.2 Generator Search

When searching for the generator architecture, we adopt the micro-cell approach. Under this formulation, we focus the search on finding the best architecture of a single cell which is later placed within a fixed template to form a full model. In conventional works, the full model is constructed by stacking the found cell multiple times, forming in this way a deep architecture. In our case, since we aim to find highly compact models, we defined the full-model architecture to contain a single, relatively powerful cell. Furthermore, the single cell is instantiated by selecting 10 operations from the set of available candidates, Op, to assign to 10 nodes within the cell. The connectivity of each node is determined by configuring the input to its operation, selecting either the cell's input or the output of any previous node. Thus, our generator search space (Figure 1 (left)) is defined as:

$$\mathbb{S}_G = \mathbb{S}_{\text{CELL}} = \underbrace{Op \times \mathbb{Z}_1 \times \cdots \times Op \times \mathbb{Z}_{10}}_{\text{20 elements}} \tag{4}$$

where $\mathbb{Z}_m = \{1, 2, \cdots, m\}$ is a set of indices representing possible inputs to a node. We consider the following operations when searching for the generator:

$$\begin{aligned} Op = \{\ &\texttt{Conv}(k, n) \text{ with } k = 1, 3, 5, 7;\ \texttt{Conv}(k, n, 4), \\ &\texttt{DSep}(k, n), \text{ and } \texttt{InvBlock}(k, n, 2) \text{ with } k = 3, 5, 7; \\ &\texttt{SEBlock}(), \texttt{CABlock}(), \texttt{Identity}\ \} \end{aligned}$$

where $\texttt{Conv}(k, n, g{=}1, s{=}1)$ is a convolution with kernel $k \times k$, n output channels, g groups and stride s; $\texttt{DSep}(k, n)$ is depthwise-separable convolution [18]; `SEBlock` is Squeeze-and-Excitation block [19]; `CABlock` is channel attention block [53]; and $\texttt{InvBlock}(k, n, e)$ is inverted bottleneck block [41] with kernel size $k \times k$, n output channels and expansion of e. In the case where a cell constructed from a point in

the search space has more than one node which is not used as input to any other node (*i.e.* a leaf node), we add their outputs together and use the sum as the cell's output. Otherwise, the output of the last node is set as the cell's output.

We use weight sharing similar to [37]. That is, for each search step t, after evaluating an architecture $s_t \in \mathbb{S}_G$ we save trained weights and use them to initialize the weights when training a model at step $t+1$. Because different operations will most likely require weights of different shape, for each node i we keep track of the best weights so far for each operation from the set Op independently. Let $s(i)$ be the operation assigned to the i-th node according to the cell structure s. Further, let $\mathbb{P}_{o,i,t}$ be the set of architectures explored until step t (inclusive) in which o was assigned to the i-th node, that is: $\mathbb{P}_{o,i,t} = \{s \mid s \in \tau(t) \wedge s(i) = o\}$. Finally, let $\theta_{i,o,t}$ represent weights in the cache, at the beginning of step t, for an operation o when assigned to the i-th node, and $\hat{\theta}_{i,o,t}$ represent the same weights after evaluation of s_t (which includes training). Note that $\hat{\theta}_{o,i,t} = \theta_{o,i,t}$ if $s_t(i) \neq o$ as the weights are not subject to training. We can then formally define our weight sharing strategy as:

$$\theta_{o,i,0} \sim \mathcal{N}$$
$$\theta_{o,i,t+1} = \begin{cases} \hat{\theta}_{o,i,t} & \text{if } s_t(i) = o \text{ and} \\ & E(s_t) > \max_{s_p \in \mathbb{P}_{o,i,t-1}} E(s_p) \\ \theta_{o,i,t} & \text{otherwise} \end{cases} \tag{5}$$

As SR models require at least one order of magnitude more compute than classification tasks, we employ a variety of techniques to speed up the training process when evaluating different architectures and effectively explore a larger number of candidate architectures. First, similar to previous works [57], we use lower fidelity estimates, such as fewer epochs with higher batch sizes, instead of performing full training until convergence which can be prohibitively time consuming. Moreover, we use smaller training patch sizes as previous studies [48] have shown that the performance of the model scales according to its training patch size, preserving in this manner the relative ranking of different architectures. Lastly, we leverage the small compute and memory usage of the models in our search space and dynamically assign multiple models to be trained on each GPU. We also constrain the number of Mult-Adds and discard all proposed architectures which exceed the limit before the training stage, guaranteeing the generation of small models while, indirectly, speeding up their evaluation.

After the search has finished, we take the best found design point s^* and train it on the full task to obtain the final distortion-based generator G, before proceeding to the next stage. When performing the final training, we initialize the weights with values from the cache $\theta_{o,i,T}$, as we empirically observed that it helps the generator converge to better minima. Both the proxy and full task aim to optimize the fidelity of the upsampled image and are, thus, validated using PSNR and trained on the training set $\hat{T}$ using the L1 loss, defined as:

$$L_1 = \frac{1}{|\hat{T}|} \sum_{(I^{\mathrm{LR}}, I^{\mathrm{HR}}) \in \hat{T}} |G(I^{\mathrm{LR}}) - I^{\mathrm{HR}}| \tag{6}$$

with I^{LR} the low-resolution image and I^{HR} the high-resolution ground-truth.

3.3 Discriminator Search

After we find the distortion-based generator model G, we proceed to search for a matching discriminator D that will be used to optimize the generator towards perceptually-good solutions. The internal structure of our discriminator consists of 5 reduction blocks. Each reduction block comprises a sequence of two operations followed by a batch normalization [23] – the first operation is selected from the set of candidate operations Op which is the same as for the generator search; the second one is a reduction operation and its goal is to reduce the spatial dimensions along the x- and y-axes by a factor of 2 while increasing the number of channels by the same factor. To only choose reduction operations from the set of operations derived from Op, we only consider standard convolutions with the same hyperparameters as in Op, but with stride changed to 2 and increased number of output channels:

$$\begin{aligned} ROp = \{\ &\texttt{Conv}(k, 2n, 1, 2) \text{ with } k = 1, 3, 5, 7 \text{ and} \\ &\texttt{Conv}(k, 2n, 4, 2) \text{ with } k = 3, 5, 7\ \} \end{aligned}$$

As a result, the search space for the discriminator can be defined as:

$$\mathbb{S}_D = \underbrace{Op \times ROp \times \cdots \times Op \times ROp}_{\text{10 elements}} \tag{7}$$

After the 5 reduction blocks, the extracted features are flattened to a 1-D vector and passed to a final linear layer (preceded by an optional bottleneck with m outputs), producing a single output which is then used to discriminate between the generated upsampled image, $G(I^{\mathrm{LR}})$ and the ground truth, I^{HR}. Figure 1 (right) shows the overall structure of the discriminator architecture.

To optimize for perceptual quality (Eq. (8)), we use the perceptual loss [24], L_{vgg}, and adversarial loss [15], L_{adv}, on both the proxy and full task. The discriminator is trained on the standard loss, L_D. As observed by previous works [48, 39, 7], optimizing solely for perceptual quality may lead to undesirable artifacts. Hence, similar to Wang *et al.* [48], we incorporate L_1 into the generator loss, L_G. Additionally, we validate the training using a full-reference perceptual metric, Learned Perceptual Image Patch Similarity [52] (LPIPS), as we find no-reference metrics such as NIQE to be more unstable since they do not take into account the ground truth. Our generator loss and discriminator loss are as follows:

$$
\begin{aligned}
L_{vgg} &= \frac{1}{|\hat{T}|} \sum_{(I^{LR}, I^{HR}) \in \hat{T}} (\phi(G(I^{\mathrm{LR}})) - \phi(I^{\mathrm{HR}}))^2 \\
L_{adv} &= -\log(D(G(I^{\mathrm{LR}})) \\
L_G &= \alpha L_1 + \lambda L_{vgg} + \gamma L_{adv} \\
L_D &= -\log(D(I^{\mathrm{HR}})) - \log(1 - D(G(I^{\mathrm{LR}})))
\end{aligned}
\tag{8}
$$

Unlike the generator search we do not use weight sharing when searching for the discriminator. The reason behind this is that we do not want the discriminator to be too good at the beginning of training to avoid a potential situation where the generator is unable to learn anything because of the disproportion between its own and a discriminator's performance. Similar to the generator search stage, we used a lower patch size, fewer epochs, and a bigger batch size to speed up the training. Additionally, from empirical observations in previous works [24, 27], the perceptual quality of the upsampled image scales accordingly with the depth of the network layer used. Therefore, we use an earlier layer of the pre-trained VGG network (ϕ) as a fidelity estimate to save additional computations. In contrast to the generator search, we do not impose a Mult-Adds limit to the discriminator as its computational cost is a secondary objective for us, since it does not affect the inference latency upon deployment.

After the search finishes, we collect a set of promising discriminator architectures and use them to train the generator on the full task. At the beginning of training, we initialize the generator with the pre-trained model found in the first stage (Section 3.2) to reduce artifacts and produce better visual results.

4 Evaluation

In this section, we present the effectiveness of the proposed methodology. For all experiments, the target models were trained and validated on the DIV2K [45] dataset and tested on the commonly-used SR benchmarks, namely Set5 [3], Set14 [51], B100 [31], and Urban100 [20]. For distortion (PSNR/Structural Similarity index (SSIM) [49]) and perceptual metrics (Natural Image Quality Evaluator [34] (NIQE)/Perceptual Index [4] (PI)), we shaved the upsampled image by its scaling factor before evaluation. For LPIPS, we passed the whole upsampled image and evaluated it on version 0.1 with linear calibration on top of intermediate features in the VGG [43] network[4]. For the exhaustive list of all hyperparameters and system details, please refer to the supplementary material.

4.1 TPSR Generator

Following Algorithm 1, we began by running the first search stage for the generator architecture to obtain a distortion-driven tiny SR model. During the search,

[4] Provided by https://github.com/richzhang/PerceptualSimilarity

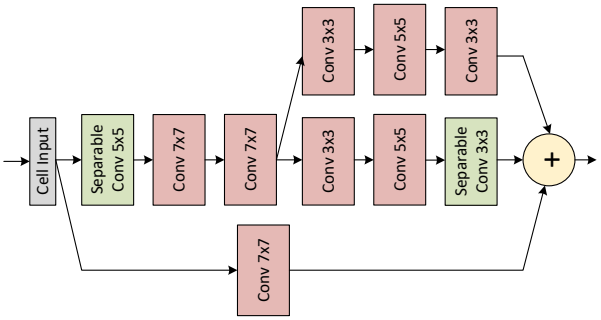

Fig. 2. Discovered cell architecture for the TPSR generator. Each operation is followed by a PReLU [16] activation.

we trained candidate models to perform $\times 2$ upscaling (*i.e.* with one upsampling block) and we set the number of feature maps (n) to 16. Each model was evaluated using PSNR as the target metric and the final reward for the controller was calculated by normalizing the average PSNR of a model.

To obtain the final generator model, we run the generator search for 2,500 steps and stored the highest-performing cell architecture as evaluated on the proxy task. Figure 2 illustrates the obtained cell structure. We refer to this model as TPSR (Tiny Perceptual Super Resolution). For the rest of this section, we use the notation TPSR-X to refer to TPSR when trained with discriminator X and TPSR-NOGAN when TPSR is distortion-driven.

After the end of the first search stage, we trained the discovered TPSR model on the full task for $\times 2$ upscaling and $\times 4$ upscaling, starting from the pre-trained $\times 2$ model, to obtain TPSR-NOGAN. Our NAS-based methodology was able to yield the most efficient architecture of only 3.6G Mult-Adds on $\times 4$ upscaling with performance that is comparable with the existing state-of-the-art distortion-driven models that lie within the same computational regime. Given that our goal was to build a perceptual-based model, we did not optimize our base model further, considering it to be a good basis for the subsequent search for a discriminator. The distortion-based results can be found in the supplementary material.

4.2 Discriminator Analysis

To obtain a discriminator architecture, we utilized the TPSR-NOGAN variant trained on the $\times 4$ upscaling task and searched for a suitable discriminator to minimize the perceptual LPIPS metric. To minimize the instability of the perceptual metric, we evaluated each model by considering the last three epochs and returning the best as the reward for the controller. We also incorporated spectral normalization [35] for the discriminator on both the proxy and the full task. We have found that discriminators of varying size and compute can lead to perceptually similar results (LPIPS). Upon further examinations, we have also found that these upsampled images look perceptually sharper than TPSR-NOGAN's.

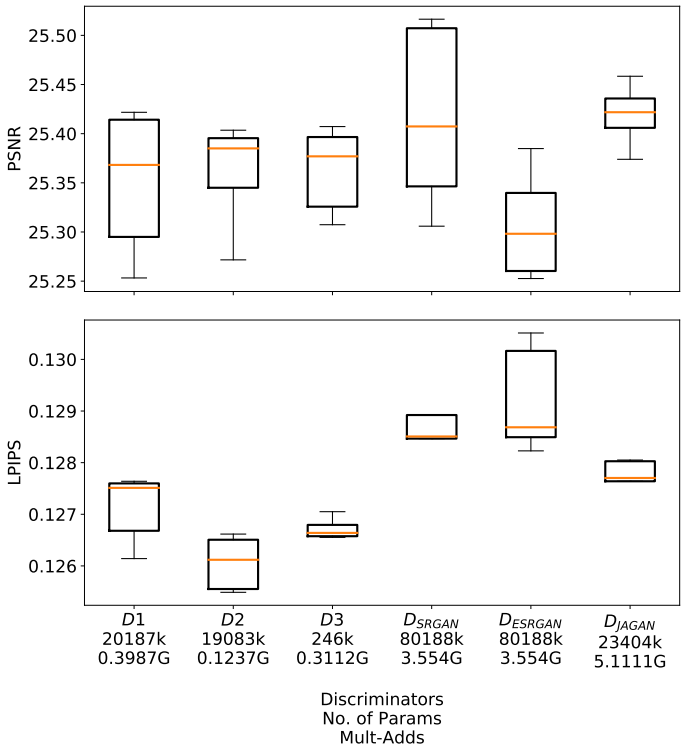

Fig. 3. Performance of TPSR after adversarial training using different discriminators found via NAS (D1, D2, D3) vs. existing discriminators designed for (SRGAN, ESRGAN, JAGAN). TPSR trained on searched discriminators, which are optimized for LPIPS, outperform fixed discriminators in the literature for the targeted metric (LPIPS). Each GAN training was performed 5 times

In order to evaluate the fidelity of the proxy task, we took the three best performing discriminator candidates based on their performance on the proxy task. We then evaluated our TPSR model when trained with these discriminators on the full task. To compare to models from the literature, we also considered the discriminators that were used in SRGAN [27], ESRGAN [48], and the recently proposed Joint-Attention GAN [7] (JAGAN). Note that the discriminator's architecture in SRGAN and ESRGAN is the same but the latter is trained using the relativistic GAN (RGAN) loss [25]. For more details on how RGAN is adopted for SR, please refer to Wang *et al.* [48].

Each GAN training was performed 5 times and the best performing model, based on the achieved LPIPS on the validation set, was evaluated on the test benchmarks. Specifically, we took the weighted average (based on the number of images) over the test benchmarks of three metrics: PSNR, NIQE, and LPIPS, and present our findings in Figure 3. Our chosen discriminators (TPSR-D1,

Table 1. We compare our ×4 upscaling TPSR models, which are optimized for LPIPS, with perceptual-driven models in the literature. Higher is better for PSNR and lower is better for LPIPS and PI. red/blue represents best/second best respectively. On the optimization target metric, LPIPS, our model (TPSR-D2) achieves the *second best* result while it is the *smallest* among all. Our model outperforms EnhanceNet and SRGAN in visual quality metrics (PSNR & LPIPS) while being 26.4× more memory efficient and 33.6× more compute efficient than SRGAN and EnhanceNet, respectively

Model	Params (K)	Mult-Adds (G)	Set5 PSNR/LPIPS/PI	Set14 PSNR/LPIPS/PI	B100 PSNR/LPIPS/PI	Urban100 PSNR/LPIPS/PI
ESRGAN	16,697	1034.1	30.40/0.0745/3.755	26.17/0.1074/2.926	25.34/0.1083/2.478	24.36/0.1082/3.770
SRGAN	1,513	113.2	29.40/0.0878/3.355	26.05/0.1168/2.881	25.19/0.1224/2.351	23.67/0.1653/3.323
EnhanceNet	852	121.0	28.51/0.1039/2.926	25.68/0.1305/3.017	24.95/0.1291/2.907	23.55/0.1513/3.471
FEQE	96	5.64	31.29/0.0912/5.935	27.98/0.1429/5.400	27.25/0.1455/5.636	25.26/0.1503/5.499
TPSR-D2	61	3.6	29.60/0.076/4.454	26.88/0.110/4.055	26.23/0.116/3.680	24.12/0.141/4.516

TPSR-D2, TPSR-D3) have led to better results as compared to the existing discriminators (TPSR-D$_{SRGAN}$, TPSR-D$_{ESRGAN}$, TPSR-D$_{JAGAN}$) in the particular perceptual metric (LPIPS) that they were optimized for. The discriminator of TPSR-D2 can be found in Figure 5.

Finally, we compared the best performing GAN-based generator (TPSR-D2) on common full-reference and no-reference perceptual metrics with various well-known perceptual models (Table 1). Considering our optimized metric, our model outperforms SRGAN and EnhanceNet while being up to 26.4× more memory efficient when compared to EnhanceNet and 33.6× more compute efficient compared to SRGAN. Additionally, our model also achieves higher performance in distortion metrics, indicating higher image fidelity and, therefore, constitutes a dominant solution for full-reference metrics (PSNR & LPIPS) especially with our tiny computational budget. Visual comparisons can be found in Figure 4.

5 Limitations and Discussion

In this paper, we have presented a NAS-driven framework for generating GAN-based SR models that combine high perceptual quality with limited resource requirements. Despite introducing the unique challenges of our target problem and showcasing the effectiveness of the proposed approach by finding high-performing tiny perceptual SR models, we are still faced with a few open challenges.

The usefulness of NAS approaches which utilize a proxy task to obtain feedback on candidate architectures naturally depends on the faithfulness of the proxy task with regards to the full task. As GANs are known to be unstable and hard to train [40], providing the search with a representative proxy task is even more challenging for them than for other workloads. We were able to partially mitigate this instability by smoothing out accuracy of a trained network, as mentioned in Section 4.2. Nevertheless, we still observed that the informativeness of results obtained on the proxy task for GAN-training is visibly worse than *e.g.* results on the proxy task when searching for a generator in the first phase

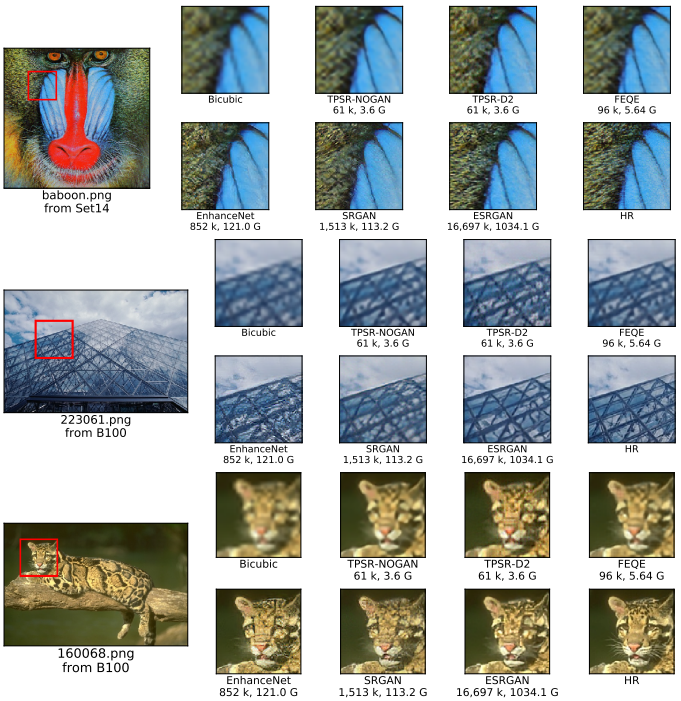

Fig. 4. Visual comparisons among SoTA perceptual-driven networks and TPSR models, with their no. of parameters (left) and mult-add operations (right). Despite the quantitative results that show that TPSR-D2 is better than eg. SRGAN (Table. 1), the qualitative results are arguably worse-off in some images, highlighting a limitation and the need for a better perceptual metric. However, TPSR-D2 still produces better reconstructions than FEQE - the current SoTA for constrained perceptual models.

of our method. This instability is reinforced even more when using no-reference perceptual metrics such as NIQE [34] and PI [4] - in which case we observed that training a single model multiple times on our proxy task can result in a set of final accuracies with variance close to the variance of all accuracies of all models explored during the search - rendering it close to useless in the context of searching. In this respect, we adopted LPIPS which, being a full-reference metric, was able to provide the search with a more robust evaluation of proposed architectures. While the strategies we used to improve the stability of the search were adequate for us to obtain decent-performing models, the challenge still remains open and we strongly suspect that overcoming it would be a main step towards improving the quality of NAS with GAN-based perceptual training.

Another important challenge comprises the selection of a metric that adequately captures perceptual quality. Identifying a metric that closely aligns with human-opinion scores across a wide range of images still constitutes an open research problem with significant invested research effort [36, 34, 52, 4]. In this respect, although we show that optimizing for LPIPS on the average leads to

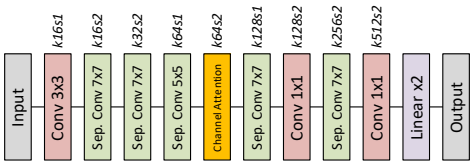

Fig. 5. Discovered architecture of the TPSR discriminator ($D2$). Convolutions are followed by batch normalization and PReLU. k = output tensor depth, s = stride

better quantitative results, the inherent limitations of the metric itself might not result to qualitatively better results on certain images.

With our work targeting highly compact models optimized for perceptual quality, it is currently challenging to find appropriate baselines that lie within the same computational and memory footprint regime, as FEQE [47] is, to the best of our knowledge, the only perceptual SR model that meets these specifications. As a result, in this paper, we also present comparisons with significantly larger models, including SRGAN and EnhanceNet, which our method outperforms in our optimized metric. We also compare with ESRGAN which is more than an order of magnitude more expensive than all examined models. Although our design did not outperform ESRGAN, we could extend our method to explore relaxed constraints to allow a slightly larger generator and employ a relativistic discriminator [25]. As our focus is on pushing the limits of building a constrained and perceptual SR model that can be deployed in a mobile SR framework [28], we leave the trade-off between model size and perceptual quality as future work.

Lastly, our method resulted in discriminators that are slightly better than existing discriminators in terms of perceptual performance. Nevertheless, even though the performance gains on LPIPS are marginal, our resulted discriminators are orders of magnitude smaller in terms of model size and computational cost and the obtained gains are consistently better across multiple runs.

6 Conclusion

In this paper, we investigated the role of the discriminator in GAN-based SR and the limits to which we can push perceptual quality when targeting extremely constrained deployment scenarios. In this context, we adopted the use of NAS to extensively explore a wide range of discriminators, making the following key observations on NAS for GAN-based SR: 1) Discriminators of drastically varying sizes and compute can lead to similar perceptually good images; possible solutions for the ill-posed super-resolution problem. 2) Due to this phenomenon and the high variance in the results of popular perceptual metrics, designing a faithful proxy task for NAS is extremely challenging. Nevertheless, we are able to find discriminators that are consistently better than existing discriminators on our chosen metric, generating a tiny perceptual model that outperforms the state-of-the-art SRGAN and EnhanceNet in both full-reference perceptual and distortion metrics with substantially lower memory and compute requirements.

References

1. Ahn, N., Kang, B., Sohn, K.A.: Fast, Accurate, and Lightweight Super-Resolution with Cascading Residual Network. In: ECCV (2018)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate (2015)
3. Bevilacqua, M., Roumy, A., Guillemot, C., line Alberi Morel, M.: Low-Complexity Single-Image Super-Resolution based on Nonnegative Neighbor Embedding. In: British Machine Vision Conference (BMVC) (2012)
4. Blau, Y., Mechrez, R., Timofte, R., Michaeli, T., Zelnik-Manor, L.: The 2018 PIRM Challenge on Perceptual Image Super-Resolution. In: ECCV Workshops (2018)
5. Blau, Y., Michaeli, T.: The Perception-Distortion Tradeoff. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
6. Chang, H., Yeung, D.Y., Xiong, Y.: Super-resolution through neighbor embedding. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) (2004)
7. Chen, R., Xie, Y., Luo, X., Qu, Y., Li, C.: Joint-attention discriminator for accurate super-resolution via adversarial training. In: Proceedings of the 27th ACM International Conference on Multimedia (ACM MM). pp. 711–719 (2019)
8. Chu, X., Zhang, B., Ma, H., Xu, R., Li, J., Li, Q.: Fast, accurate and lightweight super-resolution with neural architecture search. ArXiv **abs/1901.07261** (2019)
9. Chu, X., Zhang, B., Xu, R., Ma, H.: Multi-objective reinforced evolution in mobile neural architecture search. ArXiv **abs/1901.01074** (2019)
10. Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **38**, 295–307 (2016)
11. Dong, C., Loy, C.C., Tang, X.: Accelerating the Super-Resolution Convolutional Neural Network. In: ECCV (2016)
12. Łukasz Dudziak, Abdelfattah, M.S., Vipperla, R., Laskaridis, S., Lane, N.D.: ShrinkML: End-to-End ASR Model Compression Using Reinforcement Learning. In: Proc. Interspeech 2019. pp. 2235–2239 (2019). https://doi.org/10.21437/Interspeech.2019-2811, http://dx.doi.org/10.21437/Interspeech.2019-2811
13. Elsken, T., Metzen, J.H., Hutter, F.: Neural architecture search: A survey. J. Mach. Learn. Res. **20**, 55:1–55:21 (2018)
14. Gong, X., Chang, S., Jiang, Y., Wang, Z.: Autogan: Neural architecture search for generative adversarial networks. In: The IEEE International Conference on Computer Vision (ICCV) (Oct 2019)
15. Goodfellow, I., et al.: Generative Adversarial Nets. In: Advances in Neural Processing Systems (2014)
16. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. 2015 IEEE International Conference on Computer Vision (ICCV) pp. 1026–1034 (2015)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. pp. 770–778 (2016)
18. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. ArXiv **abs/1704.04861** (2017)
19. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 7132–7141 (2017)

20. Huang, J., Singh, A., Ahuja, N.: Single image super-resolution from transformed self-exemplars. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
21. Hui, Z., Gao, X., Yang, Y., Wang, X.: Lightweight image super-resolution with information multi-distillation network. In: Proceedings of the 27th ACM International Conference on Multimedia (ACM MM). pp. 2024–2032 (2019)
22. Hui, Z., Wang, X., Gao, X.: Fast and accurate single image super-resolution via information distillation network. In: CVPR. pp. 723–731 (2018)
23. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. ArXiv **abs/1502.03167** (2015)
24. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: ECCV (2016)
25. Jolicoeur-Martineau, A.: The relativistic discriminator: a key element missing from standard gan (2018)
26. Kim, K.I., Kwon, Y.: Single-Image Super-Resolution Using Sparse Regression and Natural Image Prior. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **32**, 1127–1133 (2010)
27. Ledig, C., et al.: Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
28. Lee, R., Venieris, S.I., Dudziak, L., Bhattacharya, S., Lane, N.D.: MobiSR: Efficient On-Device Super-Resolution Through Heterogeneous Mobile Processors. In: The 25th Annual International Conference on Mobile Computing and Networking. MobiCom '19 (2019), http://doi.acm.org/10.1145/3300061.3345455
29. Lim, B., Son, S., Kim, H., Nah, S., Lee, K.M.: Enhanced Deep Residual Networks for Single Image Super-Resolution. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2017)
30. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In: The European Conference on Computer Vision (ECCV) (2018)
31. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: IEEE International Conference on Computer Vision (ICCV) (2001)
32. Mechrez, R., Talmi, I., Shama, F., Zelnik-Manor, L.: Learning to maintain natural image statistics, [arxiv](https://arxiv.org/abs/1803.04626). arXiv preprint arXiv:1803.04626 (2018)
33. Mittal, A., Moorthy, A.K., Bovik, A.C.: No-Reference Image Quality Assessment in the Spatial Domain. IEEE Transactions on Image Processing **21**, 4695–4708 (2012)
34. Mittal, A., Soundararajan, R., Bovik, A.C.: Making a "Completely Blind" Image Quality Analyzer. IEEE Signal Processing Letters **20**, 209–212 (2013)
35. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral Normalization for Generative Adversarial Networks. ArXiv **abs/1802.05957** (2018)
36. Moorthy, A.K., Bovik, A.C.: Blind image quality assessment: From natural scene statistics to perceptual quality. IEEE Transactions on Image Processing **20**, 3350–3364 (2011)
37. Pham, H.Q., Guan, M.Y., Zoph, B., Le, Q.V., Dean, J.: Efficient neural architecture search via parameter sharing. In: International Conference on Machine Learning (2018)

38. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. CoRR **abs/1802.01548** (2018)
39. Sajjadi, M.S.M., Schölkopf, B., Hirsch, M.: Enhancenet: Single image super-resolution through automated texture synthesis. 2017 IEEE International Conference on Computer Vision (ICCV) pp. 4501–4510 (2016)
40. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., Chen, X.: Improved Techniques for Training GANs. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 2234–2242 (2016)
41. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted Residuals and Linear Bottlenecks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
42. Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z.: Real-time single image and video super-resolution using an efficient subpixel convolutional neural network. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 1874–1883 (2016)
43. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (2015)
44. Song, D., Xu, C., Jia, X., Chen, Y., Xu, C., Wang, Y.: Efficient residual dense block search for image super-resolution. In: Thirty-Fourth AAAI Conference on Artifical Intelligence (2020)
45. Timofte, R., et al.: NTIRE 2017 Challenge on Single Image Super-Resolution: Methods and Results. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) (2017)
46. Tong, T., Li, G., Liu, X., Gao, Q.: Image Super-Resolution Using Dense Skip Connections. In: IEEE International Conference on Computer Vision (ICCV) (2017)
47. Vu, T., Van Nguyen, C., Pham, T.X., Luu, T.M., Yoo, C.D.: Fast and Efficient Image Quality Enhancement via Desubpixel Convolutional Neural Networks. In: The European Conference on Computer Vision (ECCV) Workshops (September 2018)
48. Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Loy, C.C., Qiao, Y., Tang, X.: ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. In: ECCV Workshops (2018)
49. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing **13**, 600–612 (2004)
50. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning **8**, 229–256 (1992)
51. Yang, J., Wright, J., Huang, T.S., Ma, Y.: Image Super-resolution via Sparse Representation. Trans. Img. Proc. **19**(11), 2861–2873 (2010)
52. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
53. Zhang, Y., Li, K., Li, K., Wang, L., Zhong, B., Fu, Y.: Image Super-Resolution Using Very Deep Residual Channel Attention Networks. In: European Conference on Computer Vision (ECCV) (2018)
54. Zhang, Y., Tian, Y., Kong, Y., Zhong, B., Fu, Y.: Residual Dense Network for Image Super-Resolution. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
55. Zhong, Z., Yan, J., Wu, W., Shao, J., Liu, C.L.: Practical block-wise neural network architecture generation. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 2423–2432 (2018)

56. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. In: International Conference on Learning Representations (2017), https://openreview.net/forum?id=r1Ue8Hcxg
57. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 8697–8710 (2018)