Exploring Simple and Transferable Recognition-Aware Image Processing

Zhuang Liu, Hungju Wang, Tinghui Zhou, Zhiqiang Shen, Bingyi Kang, Evan Shelhamer, and Trevor Darrell

Abstract—Recent progress in image recognition has stimulated the deployment of vision systems at an unprecedented scale. As a result, visual data are now often consumed not only by humans but also by machines. Existing image processing methods only optimize for better human perception, yet the resulting images may not be accurately recognized by machines. This can be undesirable, e.g., the images can be improperly handled by search engines or recommendation systems. In this work, we examine simple approaches to improve machine recognition of processed images: optimizing the recognition loss directly on the image processing network or through an intermediate input transformation model. Interestingly, the processing model’s ability to enhance recognition quality can transfer when evaluated on models of different architectures, recognized categories, tasks, and training datasets. This makes the methods applicable even when we do not have the knowledge of future recognition models, e.g., when uploading processed images to the Internet. We conduct experiments on multiple image processing tasks paired with ImageNet classification and PASCAL VOC detection as recognition tasks. With these simple yet effective methods, substantial accuracy gain can be achieved with strong transferability and minimal image quality loss. Through a user study we further show that the accuracy gain can transfer to a black-box cloud model. Finally, we try to explain this transferability phenomenon by demonstrating the similarities of different models’ decision boundaries. Code is available at https://github.com/liuzhuang13/Transferable_RA.

1 INTRODUCTION

Unlike in image recognition where a deep network maps an image to a category label, a deep network used for image processing maps an input image to an output image with some desired properties. Examples include super-resolution [1], denoising [2], deblurring [3], colorization [4].

The goal of such systems is to produce images of high perceptual quality to a human observer. For example, in image denoising, we aim to remove noise that is not useful to an observer and restore the image to its original “clean” form. Metrics like PSNR/SSIM [5] are often used [1], [6] to approximate human-perceived similarity between the processed and original images, and direct human assessment on the fidelity of the output is often considered the “gold-standard” [7], [8]. Therefore, techniques have been proposed to make outputs look perceptually pleasing to humans [7], [9], [10].

However, while looking good to humans, image processing outputs may not be accurately recognized by image recognition systems. As shown in Fig. 1, the output image of a denoising model could easily be recognized by a human as a bird, but a recognition model classifies it as a kite. One could specifically train a recognition model only on these output images produced by the denoising model to achieve better performance on such images, or could leverage domain adaptation approaches to adapt the recognition model to this domain, but the performance on natural images can be harmed. This retraining/adaptation scheme might also be impractical considering the significant overhead induced by catering to various image processing tasks and models.

With the fast-growing size of image data, images are often “viewed” and analyzed more by machines than by humans. Nowadays, any image uploaded to the Internet is likely to be analyzed by certain vision systems. Therefore, it is of great importance for the processed images to be recognizable not only by humans, but also by machines. In other words, recognition systems (e.g., image classifier) should be able to accurately explain the underlying category meaning of the image content. In this way, we make them easier to search, recommended to more interested audience, and so on, as these procedures are mostly executed by machines based on their understanding of the images. Therefore, we
argue that image processing systems should also aim at machine recognizability. We call this problem “Recognition-Aware Image Processing”.

It is also important that the enhanced recognizability is not specific to any concrete recognition model, i.e., the improvement is only achieved when the output images are evaluated on one particular model. Instead, the improvement should ideally be transferable when evaluated with different downstream models/tasks, to support its usage without access to possible future recognition systems. On reason for this is what model will be used for recognizing the processed image could be out of our control, for example if we upload it to the Internet or share it on social media. We may not know what network architectures (e.g., ResNet or VGG) will be used for inference, what object categories the model recognizes (e.g., animals or scenes), or even what task will be performed (e.g., classification or detection). Without these specifications, could it be hard for us to enhance output images’ machine recognition accuracy in an unknown context?

In this work, we explore simple yet highly effective approaches to make image processing outputs more accurately recognized by downstream recognition systems, and demonstrate that these approaches generate transferable accuracy gain among different recognition architectures, categories, tasks and training datasets. The approaches we study add a recognition loss optimized jointly with the image processing loss. The recognition loss is computed using a fixed recognition model that is pretrained on natural images, and can be done without further supervision from class labels for training images. It can be optimized either directly by the original image processing network, but we also have the option of resorting to an intermediate transformation network or training in an unsupervised manner, depending on different use cases. Interestingly, we find the accuracy gain from optimizing one recognition model’s loss transfers favorably among different recognition model architectures, object categories, and recognition tasks, which renders our simple solutions effective even when we do not know what the downstream recognition model is.

We conduct extensive experiments, on multiple image processing (super-resolution, denoising, JPEG-deblocking) and downstream recognition (classification, detection) tasks. The results demonstrate our methods can substantially boost the recognition accuracy (e.g., up to 10%, or 20% relative gain), with minimal loss in image quality. Results are also compared with alternative approaches in Section 7. We demonstrate in detail that these methods generate transferable accuracy boost, when the downstream model is either with a different architecture, recognizes different classes, performs different tasks, or even is a cloud-based, black-box model. We conduct decision boundary analysis and show different models’ decision boundaries exhibit high similarities, to give an explanation for this transferability phenomenon.

We would like to emphasize that in studying and analyzing these approaches, our contribution does not lie in proposing novel network architectures, training procedures or loss functions, but in demonstrating that simple methods are already effective at making the processed images more accurately recognized, and the improved machine recognizability can be transferred favorably to different contexts. The simplicity of these methods also leads to easy deployment in practice, and they could serve as strong baselines in this relatively under-explored problem. The transferability also facilitates their practical usage in various and potentially changing deployment environments.

Our contributions can be summarized as:

- We propose to study the problem of enhancing the machine recognition of image processing outputs, a desired property considering the amount of images analyzed by machines nowadays.
- We study simple yet effective methods towards this goal, suitable for different use cases. Extensive experiments are conducted on multiple image processing and recognition tasks.
- We show that with simple approaches, the recognition accuracy improvement is transferable among recognition architectures, categories, tasks and datasets, a desirable behavior making the proposed methods applicable without access to downstream recognition models.
- We provide decision boundary analysis of recognition models and show their similarities to gain a better understanding of the transferability phenomenon.

2 RELATED WORK

Image processing/enhancement problems such as super-resolution and denoising have a long history [11], [12], [13], [14]. Since the initial success of deep networks on these problems [11], [2], [15], a large body of works try to investigate better model architecture design and training techniques [16], [17], [18], [19], [20], [21]. These works focus on generating high visual quality images under conventional metrics (PSNR/SSIM) or human evaluation, without considering recognition performance on the output.

There are also a number of works that relate image recognition with image processing. Some works [4], [22], [23], [24], [25] use image recognition accuracy as an evaluation metric for image colorization/super-resolution/denoising, but without optimizing for it in training. [26], [27], [28], [29], [30] investigate how to achieve more accurate recognition on low-resolution or corrupted/noisy images. [31] propose a method to make denoised images more accurately segmented. [32] introduced a theoretical framework for classification-distortion-perception tradeoff and conducted experiments with simulated or toy datasets, while our work develops practical approaches for real-world datasets. Most existing works only consider one image processing task or image domain, and develop specific techniques, while our simpler approach is task-agnostic, more widely applicable, and is first to be shown transferable. Our work is related but different from those aiming for robustness of the recognition model [33], [34], [35], as we focus on training the processing models. Our method also shares some similarity with [36] which tries to differentiate input signals by optimizing recognition
accuracy using auto-encoders. [37], [37], [38], [39], [40], [41], [42], [43] jointly train a processing model (e.g., dehazing, resizing, face reconstruction) together with a recognition model (e.g., object recognition or face recognition) to achieve better image processing and/or recognition quality. Our problem setting is different from these works, in that we assume we do not have the control of the recognition model, as it might be on the cloud or to be decided in the future, thus we adapt the image processing model only. This has the advantage of making the performance gain transferable to different downstream environments, and also ensures the recognition of natural images is not harmed as the recognition model is fixed. Section 7 includes our comparison with training recognition model jointly.

3 METHODS

We first formally define the problem setting of recognition-aware image processing, and then introduce the multiple approaches we examined, each suited for different use cases. We finally introduce different transferring scenarios we explored. Our proposed methodology, although only introduced in a vision context, can be extended to other domains (e.g., speech) as well.

3.1 Problem Setting

In a typical image processing problem, given a set of training input images \{I^k_m\} and corresponding target images \{I^k_target\}, we aim to train a network that maps an input to its target. Denoting this network as P (processing), parameterized by \(W_P\), our optimization objective is

$$\min_{W_P} L_{\text{proc}} = \frac{1}{N} \sum_{k=1}^{N} l_{\text{proc}}(P(I^k_m), I^k_{\text{target}}),$$

where \(l_{\text{proc}}\) is the loss function for each sample (e.g., \(L_2\)). The performance is typically evaluated by similarity (e.g., PSNR/SSIM) between \(I^k_{\text{target}}\) and \(P(I^k_m)\), or human assessment. In recognition-aware processing, we are interested in a recognition task, with a trained recognition model \(R\) (recognition). We assume each target image \(I^k_{\text{target}}\) is associated with a category label \(S^k\) for the recognition task. Our goal is to train a processing model \(P\) such that the recognition performance on the output images \(P(I^k_m)\) is high, when evaluated using \(R\) with the category labels \(\{S^k\}\). In practice, \(R\) might not be available (e.g., on the cloud), in which case we could resort to other models if the performance improvement transfers among models.

3.2 Optimizing Recognition Loss

Given our goal is to make the output images by \(P\) more recognizable by \(R\), it is natural to add a recognition loss on top of the objective of the image processing task (Eqn. (1)) during training

$$\min_{W_P} L_{\text{recog}} = \frac{1}{N} \sum_{k=1}^{N} l_{\text{recog}}(R(P(I^k_m)), S^k),$$

(2)

where \(l_{\text{recog}}\) is the per-example recognition loss defined by the downstream recognition task. For example, for image classification, \(l_{\text{recog}}\) could be the cross-entropy (CE) loss. Adding the image processing loss (Eqn. (1)) and recognition loss (Eqn. (2)) together, our total training objective becomes

$$\min_{W_P} L_{\text{proc}} + \lambda L_{\text{recog}},$$

(3)

where \(\lambda\) is the coefficient controlling the weights of \(L_{\text{recog}}\) relative to \(L_{\text{proc}}\). We denote this simple solution as “RA (Recognition-Aware) processing”, which is visualized in Fig. 2 left. Note that once the training is finished, the recognition model used as loss is not needed anymore, and during inference, we only need the processing model \(P\), thus no overhead is introduced in deployment. A potential shortcoming of directly optimizing \(L_{\text{recog}}\) is that it might deviate \(P\) from optimizing the original loss \(L_{\text{proc}}\), and the trained \(P\) will generate images that are not as good as if we only optimize \(L_{\text{proc}}\). We will show that, however, with proper choice of \(\lambda\), we could substantially boost the recognition performance with nearly no sacrifice on image quality.

3.3 Unsupervised Optimization of Recognition Loss

The solution above requires category labels for training images, which however, may not be always available. In this case, we could regress the recognition model’s output of the target image \(R(I^k_{\text{target}})\). The recognition objective changes to

$$\min_{W_P} L_{\text{recog}} = \frac{1}{N} \sum_{k=1}^{N} l_{\text{dis}}(R(P(I^k_m)), R(I^k_{\text{target}})),$$

(4)
where \(d_{ps}\) is a distance measure between two of \(R\)'s outputs (e.g., \(L_2\) distance, KL divergence). We call this approach "unsupervised RA". Note that it is only unsupervised for training model \(P\), but not necessarily for the model \(R\). The (pre)training of the model \(R\) is not our concern since in our problem setting \(R\) is a given trained model, and it can be trained either with or without full supervision. Unsupervised RA is related to the “knowledge distillation” paradigm [44] used for network model compression, where the output of a large model is used to guide a small model, given the same input images. Instead we use the same recognition model \(R\), but guide the upstream processing model to generate desirable images. It is also related to the perceptual loss/feature loss used in [7], [9], where the feature distance is minimized instead of output distance. We provide a comparison in Section 7.

3.4 Using an Intermediate Transformer

Sometimes we want to prevent the added recognition loss \(L_{\text{recog}}\) from causing \(P\) to deviate from optimizing its original loss. We can achieve this by introducing an intermediate transformation model \(T\): \(P\)'s output is first fed to the \(T\), and \(T\)'s output serves as the input for \(R\) (Fig. 2 right). \(T\)'s parameters \(W_T\) are optimized for the recognition loss

\[
\min_{W_T} L_{\text{recog}} = \frac{1}{N} \sum_{k=1}^{N} l_{\text{recog}}(R(T(P(t_{in}))), S^k).
\]

With the help of \(T\) on optimizing the recognition loss, the model \(P\) can now "focus" on its original image processing loss \(L_{\text{proc}}\). The optimization objective becomes

\[
\min_{W_P} L_{\text{proc}} + \min_{W_T} \lambda L_{\text{recog}}.
\]

In Eqn. (6), \(P\) is solely optimizing \(L_{\text{proc}}\) as in the original image processing problem (Eqn. (1)). \(P\) is learned as if there is no recognition loss, and therefore the image processing quality of its output will not be affected. This could be implemented by “detaching” the gradient generated by \(L_{\text{recog}}\) between the model \(T\) and \(P\) (Fig. 2 right). We term this solution as “RA with transformer”. Its downside compared with directly optimization using \(P\) is that there are two instances for each image (the output of model \(P\) and \(T\)), one is “for human” and the other is “for machines”. Therefore, the transformer is best suited when we want to guarantee the image processing quality not affected at all, at the expense of maintaining another image. For example, in classifying images, we can have the higher-quality image presented to users for better experience and the other image passed to the backend for accurate machine classification.

3.5 Transferring Scenarios

If using \(R\) as a fixed loss can only boost the accuracy on \(R\) itself, the use of the method could be limited. Sometimes we do not have the knowledge about the future downstream recognition model or even task. Thus, we explore several scenarios to see whether processing models trained with the loss of one recognition model \(R_1\), can also boost the performance when evaluated using another model \(R_2\). Here \(R_2\) is not only possibly a different architecture but also can perform a different vision task, etc. If the improvement is transferable, then we can train our image processing model \(P\) with a loss from a generic recognition model, such as those trained on ImageNet. In this case, even if in the future the output images from \(P\) is tested on another model, we still have accuracy gains compared with a vanilla processing model. Specifically, we examine the following transfer scenarios:

- **Transferring to a different model architecture.** \(R_1\) and \(R_2\) perform the same vision task, are trained on the same datasets, and only differ in model architecture, e.g., ResNet-50 and VGG-16.
- **Transferring to another set of categories.** \(R_1\) and \(R_2\) are of the same architecture, perform the same task, but are trained to recognize disjoint subsets of categories from the same dataset.
- **Transferring to another task and dataset.** \(R_1\) and \(R_2\) perform different tasks, for instance image classification and object detection. In most cases, this also means \(R_1\) and \(R_2\) are trained on different datasets with different set of categories. They can have different architectures as well.
- **Transferring to a black-box model.** The model \(R_2\) could be a proprietary online model that provides recognition service but does not allow users access to its structures, weights, what dataset(s) it was trained on, or even its set of output categories.

Interestingly, we find that in each of the above cases, the accuracy boost gained on \(R_1\) also transfers to \(R_2\). This makes our method effective even when we cannot access the target downstream model, where we could use another trained model as the loss function. This phenomenon also implies that the “recognizability” of a processed image can be more general than just the extent it fits to a specific model. More details are presented in the experiments.

4 Experiments

4.1 Experimental Details

**General Setup.** We evaluate our proposed methods on three image processing tasks: image super-resolution, denoising, and JPEG-deblocking. In those tasks, the target images are all the original images from the datasets. To obtain the input images, for super-resolution, we use a downsampling scale factor of \(4 \times\); for denoising, we add Gaussian noise on the images with a standard deviation of 0.1 to obtain the noisy images; for JPEG deblocking, a quality factor of 10 is used to compress the image to JPEG format. The image processing loss used is the mean squared error (MSE, or \(L_2\) loss). For the recognition tasks, we consider image classification and object detection, two common tasks in computer vision. In total, we have 6 (3 \(\times\) 2) task pairs to evaluate. Training is performed with the training set and results on the validation set are reported.

We adopt the SRResNet [7] as the architecture of the image processing model \(P\) (unless otherwise specified, e.g., in Section 4.7), which is simple yet effective in optimizing the MSE loss. Even though SRResNet is originally designed for super-resolution, we find it also performs well on denoising and JPEG deblocking when its upscale parameter set to 1 for the same input-output sizes. For the transformer
model $T$, we use the ResNet-like architecture in CycleGAN [45]. The recognition models are ResNet, VGG and DenseNet. Please refer to Section 7 for comparison with alternative approaches.

Throughout the experiments, on both the image processing network and the transformer, we use the Adam optimizer [46] with an initial learning rate of $10^{-4}$, following the original SRResNet [7]. Our implementation is in PyTorch [47]. The experiments are run on 1-4 NVIDIA TITAN Xp GPUs. The training process finishes in 2-24 hours depending on the model sizes/variants of methods/recognition tasks, and the maximum GPU memory taken is 30GB (multi-GPU) with batch size of 20.

**Image Classification.** For image classification, we evaluate our method on the large-scale ImageNet benchmark [48], which can be downloaded at http://image-net.org/download. It consists of $\sim$1.2 million training images and 50,000 validation images. We use five pre-trained image classification models, ResNet-18/50/101 [49], DenseNet-121 [50] and VGG-16 [51] with BN [52] (denoted as R18/50/101, D121, V16 in Table 1), on which the top-1 accuracy (%) of the original validation images is 69.8, 76.2, 77.4, 74.7, and 73.4 respectively. We train the processing models for 6 epochs on the training set, with a learning rate decay of $10^{-4}$ at epoch 5 and 6, and a batch size of 20. In evaluation, we feed unprocessed validation images to the image processing model, and report the accuracy of the output images evaluated on the pre-trained classification networks. For unsupervised RA, we use $L_2$ distance as the function $l_{\lambda_\alpha}$ in Eqn. (4). The hyperparameter $\lambda$ is chosen using super-resolution with the ResNet-18 recognition model, on two small subsets for training/validation from the original large training set, from a grid search from $10^{-4}$ to 100. The $\lambda$ chosen for RA processing, RA with transformer, and unsupervised RA is $10^{-3}$, $10^{-2}$ and 10 respectively.

**Object Detection.** For object detection, we evaluate on PASCAL VOC 2007 and 2012 detection dataset (https://pjreddie.com/projects/pascal-voc-dataset-mirror/), using Faster-RCNN [53] as the recognition model. Our implementation is based on the code from [54]. Following common practice [53], [55], [56], we use VOC 07 and 12 trainable data as the training set, and evaluate on VOC 07 test data. The Faster-RCNN training uses the same hyperparameters in [54]. For the recognition model’s backbone architecture, we evaluate ResNet-18/50/101 and VGG-16 (without BN [52]), obtaining mAP of 74.2, 76.8, 77.9, 72.2 on the test set respectively. Given those trained detectors as recognition loss functions, we train the models on the training set for 7 epochs, with a learning rate decay of $10^{-4}$ at epoch 6 and 7, and a batch size of 1. We report the mean Average Precision (mAP) of processed images in the test set. As in image classification, we use $\lambda = 10^{-3}$ for RA processing, and $\lambda = 10^{-2}$ for RA with transformer.

### 4.2 Evaluation on the Same Recognition Model

We first present results when the $R$ used for evaluation is the same as the $R$ we use as the recognition loss in training. Table 1 shows our results on ImageNet classification. ImageNet-pretrained classification models ResNet-18/50/101, DenseNet-121 and VGG-16 are denoted as R18/50/101, D121, V16. “No Processing” denotes the accuracy on input images (low-resolution/noisy/JPEG-compressed); “Plain Processing” denotes using image processing models trained without recognition loss (Eqn. (1)). We observe that plain processing can boost the accuracy over unprocessed images. These two are considered as baselines.

From Table 1, using RA processing can significantly boost the accuracy of output images over plainly processed ones, for all image processing tasks and recognition models. This is more prominent when the accuracy of plain processing is lower, e.g., in SR and JPEG-deblocking, where we mostly obtain $\sim$10% accuracy gain (close to 20% in relative terms). Even without category labels, our unsupervised RA can still in most cases outperform baseline methods, despite achieving lower accuracy than its supervised counterpart. Also in SR and JPEG-deblocking, using an intermediate transformer $T$ can bring additional improvement over RA processing.

The results for PASCAL VOC object detection, when evaluated on the same architecture, are shown in Table 2. We observe similar trend as in classification: using recognition loss can consistently improve the mAP over plain image processing by a notable margin. On super-resolution, RA processing mostly performs on par with RA with transformer, but on the other two tasks using a transformer is slightly better. The model with transformer performs better more often possibly because with this extra network in the middle, the capacity of the whole system is increased.

For all result tables, each reported accuracy number is based on one run due to the relatively stable performance (almost all within 1%) we noticed and the large amount of tasks combinations/architectures to be evaluated. For example, on super-resolution with ResNet-18 as $R$ on ImageNet, five runs on plain processing gives accuracies (%): 53.9, 53.8, 54.0, 53.8, 53.9 (53.88 $\pm$ 0.07); RA processing: 61.7, 61.9, 61.6, 61.8, 61.7 (61.74 $\pm$ 0.10); unsupervised RA: 61.2, 61.3, 61.3,

### Table 1

| Task | Super-Resolution | Denoising | JPEG-Deblocking |
|------|------------------|-----------|-----------------|
|      | R18 | R50 | R101 | D121 | V16 | R18 | R50 | R101 | D121 | V16 | R18 | R50 | R101 | D121 | V16 |
| No Processing | 46.3 | 50.4 | 55.5 | 51.6 | 42.1 | 46.8 | 55.8 | 61.3 | 59.7 | 46.7 |
| Plain Processing | 52.6 | 58.8 | 61.9 | 57.7 | 50.2 | 61.9 | 68.0 | 69.1 | 66.4 | 60.9 | 48.2 | 53.8 | 56.0 | 50.2 | 52.9 |
| RA Processing | 61.8 | 67.3 | 69.6 | 66.0 | 61.9 | 65.1 | 71.2 | 72.7 | 69.8 | 66.5 | 57.7 | 63.6 | 65.8 | 62.3 | 56.7 |
| RA, Unsupervised | 61.3 | 66.9 | 69.4 | 65.3 | 61.0 | 61.7 | 68.6 | 70.8 | 67.1 | 63.6 | 53.8 | 60.4 | 63.4 | 59.7 | 53.1 |
| RA w/ Transformer | 63.0 | 68.2 | 70.1 | 66.5 | 63.0 | 65.2 | 70.9 | 72.3 | 69.6 | 65.9 | 59.8 | 65.1 | 66.7 | 63.9 | 58.7 |

$R18$ means ResNet-18, etc. The five models achieve 69.8, 76.2, 77.4, 74.7, 73.4 on original images. RA processing techniques substantially boost recognition accuracy.
4.3 Transfer Between Recognition Architectures

In reality, sometimes the R we want to eventually evaluate the output images on might not be available for us to use as a loss for training, e.g., it could be on the cloud, kept confidential or decided later. In this case, we could train a processing model P using recognition model RA(source) that is accessible to us, and after we obtain the trained model P, evaluate its output images’ accuracy using another unseen RB(target). We evaluate model architecture pairs on ImageNet in Table 3, for RA Processing, where row is the source model (RA), and column is the target model (RB). In Table 3’s each column, training with any model RA produces substantially higher accuracy than plainly processed images on RB, indicating that the improvement is transferable among recognition architectures. This phenomenon enables us to use RA processing without the knowledge of the downstream recognition architecture.

We provide the model transferability results of RA processing on object detection in Table 4. Rows indicate the models trained as recognition loss and columns indicate the evaluation models. We see similar trend as in classification (Table 1): using other architectures as loss can also improve recognition performance over plain processing; the loss model that achieves the highest performance is mostly the model itself, as can be seen from the fact that most boldface numbers are on the diagonals.

In Table 5, we present the results when transferring between recognition architectures, using unsupervised RA. We note that for super-resolution and JPEG-deblocking, similar trend holds as in (supervised) RA processing, as using any architecture in training will improve over plain processing. But for denoising, this is not always the case. Some models P trained with unsupervised RA are slightly worse than the plain processing counterpart. A possible reason for this is the noise level in our experiments is not large enough and plain processing achieve very high accuracy already.

In Table 6, we present the results of transferring between architectures when we use a transformer T. We use the processing model P and transformer T trained with RA together when evaluating on RB. From Table 6, in most cases improvement is still transferable but there are a few exceptions. For example, when RA is ResNet or DenseNet...
and when $R_B$ is VGG-16, in most cases the accuracy falls behind plain processing by a large margin. This weaker transferability is possibly caused by the fact that there is no constraint imposed by the image processing loss on $T$’s output, thus it “overfits” more to the specific $R$ it is trained with.

### 4.4 Transfer Between Object Categories

What if the $R_A$ and $R_B$ recognize different categories of objects? We divide the 1000 classes from ImageNet into two splits, denoted as Cat (category) $A/B$, each with 500 classes, and train two 500-way classifiers ($R_{18}$) on both splits, obtaining $R_A$ and $R_B$. Next, we train two image processing models $P_{A/B}$ with the $R_A/R_B$ as recognition loss, using images from Cat $A/B$ respectively. Note that neither $P$ nor $R$ has seen images/categories from the other split. We evaluate obtained processing models on both splits in Table 8.

We observe that RA still benefits the accuracy even when transferring across categories (e.g., in SR, 60.1% to 66.5% transferring from Cat $A$ to Cat $B$). The improvement is only marginally lower than directly training on the same categories (e.g., 60.2% to 67.8% on Cat $B$). This suggests RA processing models do not impose category-specific signals to the images, but signals that enable wider sets of classes to be better recognized.

### 4.5 Transfer Between Recognition Tasks and Datasets

We evaluate task transferability when task $A$ is classification and task $B$ is object detection in Table 7, where rows are classification models and columns are detection models, so even the same name in row and column (e.g., “R18”) indicates different models trained on different tasks and datasets.

Note that rows are classification models and columns are detection models, so even the same name in row and column (e.g., “R18”) indicates different models trained on different tasks and datasets.

### Table 5

| Task | Super-Resolution | Denoising | JPEG-Deblocking |
|------|------------------|-----------|-----------------|
| Plain Processing | 52.6 58.8 61.9 57.7 50.2 | 61.9 68.0 69.1 66.4 60.9 | 48.2 53.8 56.0 52.9 42.4 |
| Unsup. RA w/R18 | 61.3 66.3 68.6 64.5 57.3 | 61.7 67.9 69.7 66.4 60.5 | 53.8 59.1 62.0 57.5 50.0 |
| Unsup. RA w/R50 | 58.9 66.9 68.6 64.1 58.2 | 61.2 68.6 70.3 66.6 61.3 | 52.8 60.4 62.5 58.3 50.3 |
| Unsup. RA w/R101 | 57.8 64.9 69.0 62.9 56.9 | 60.6 68.0 70.7 66.3 60.7 | 52.3 58.7 63.4 57.9 49.0 |
| Unsup. RA w/D121 | 58.0 64.7 67.2 65.3 56.0 | 60.7 67.8 69.7 67.1 60.3 | 52.2 59.2 62.2 59.7 49.9 |
| Unsup. RA w/V16 | 57.7 64.6 67.3 63.2 61.0 | 60.4 67.1 69.6 65.9 63.6 | 52.0 58.4 61.5 57.4 53.1 |

### Table 6

| Task | Super-Resolution | Denoising | JPEG-Deblocking |
|------|------------------|-----------|-----------------|
| Plain Processing | 52.6 58.8 61.9 57.7 50.2 | 61.9 68.0 69.1 66.4 60.9 | 48.2 53.8 56.0 52.9 42.4 |
| RA w/T w/R18 | 63.0 59.2 67.0 63.9 27.0 | 65.2 69.4 71.6 68.4 40.3 | 59.8 58.7 62.6 60.3 19.9 |
| RA w/T w/R50 | 60.5 68.2 68.9 65.8 40.4 | 63.1 70.9 71.5 68.6 48.7 | 55.0 65.1 63.9 61.9 31.5 |
| RA w/T w/R101 | 59.6 66.2 70.1 65.1 35.6 | 62.4 68.8 72.3 67.6 52.3 | 54.8 61.3 66.7 24.8 60.5 |
| RA w/T w/D121 | 58.5 64.2 66.9 66.5 27.3 | 58.0 66.8 67.3 69.6 46.7 | 46.6 57.2 59.0 63.9 9.0 |
| RA w/T w/V16 | 59.2 64.7 67.8 65.0 63.0 | 57.6 64.0 67.1 55.7 63.1 | 56.1 61.2 63.4 58.7 60.1 |

### Table 7

| Task | Super-Resolution | Denoising | JPEG-Deblocking |
|------|------------------|-----------|-----------------|
| Plain Processing | 68.5 69.7 73.1 63.2 | 68.1 71.6 74.1 65.7 | 62.4 65.6 69.5 58.3 |
| RA w/R18 | 71.3 73.5 75.6 67.8 | 70.6 73.1 75.5 64.1 | 67.7 70.3 73.2 62.4 |
| RA w/R50 | 70.8 73.2 74.8 67.8 | 70.4 73.1 75.8 66.2 | 67.8 70.2 73.1 62.8 |
| RA w/R101 | 70.7 73.2 75.3 67.0 | 70.5 73.5 75.7 66.9 | 68.1 70.2 72.8 63.2 |
| RA w/D121 | 71.2 73.6 75.3 67.2 | 70.5 73.2 75.7 65.7 | 68.1 70.5 73.1 62.6 |
| RA w/V16 | 70.4 72.4 74.6 67.5 | 70.6 73.0 75.7 67.7 | 67.8 70.3 73.2 63.7 |

Note that rows are classification models and columns are detection models, so even the same name in row and column (e.g., “R18”) indicates different models trained on different tasks and datasets.

Refer to text in Section 4.3 for details.
classification models used for RA loss and columns are detection models for evaluation. There is also a dataset shift, since model $P$ and $R$ are both trained on ImageNet; during evaluation, $P$ is fed with VOC images and we use a VOC-trained detection model $R$. We observe that using classification loss on model $A$ (row) gives accuracy gain on model $B$ (column) over plain processing in most cases. Such task transferability suggests the “machine semantics” of the image could be a task-agnostic property.

We also evaluate the opposite direction, from detection to classification. The results are shown in Table 9. Here, using RA processing can still consistently improve over plain processing for any pair of models, but we note that the improvement is not as significant as directly training using classification models as loss (Tables 1 and 3).

Additionally, the results when we transfer the model $P$ trained with unsupervised RA with image classification to object detection are shown in Table 10. In most cases, it improves over plain processing, but for image denoising, this is not always the case. Similar to results in Table 5, this could be because the noise level is relatively low in our experiments.

### 4.6 Transfer to a Black-Box, Third-Party Cloud Model

We compare the images generated from plain processing and RA models using the “General” model at clarifai.com, a company providing state-of-the-art image classification cloud services. We do not have knowledge of the model’s architecture or what datasets it was trained on, except we can access the service using APIs. The model also recognizes over 11000 + concepts that are different from the 1000-class ImageNet categories. For this experiment, we only take the output category with the maximum probability as the prediction. We use the SR processing model trained with R18/ImageNet as the RA model. We randomly sample image indices from ImageNet validation set, and ask clarifai.com for predictions of both images generated from plain and RA processing models.

From the results, we then randomly select 100 instances where clarifai.com gives different predictions on plain and RA super-resolved images, to compose a survey for user study. For each of the 100 instances, the survey presents the user with the target image, and both prediction labels generated from plain/RA images, in randomized left/right order. The survey asks the user to indicate in his/her opinion which label(s) describe the image to a satisfactory level. The user has the options to choose none, either or both labels. The survey and instructions can be found at https://tinyurl.com/y698779q. 10 volunteers participated in our survey. The resulting average satisfaction rates for plain and RA super-resolved images are 40.1% and 55.3% respectively. We achieve 15.2% absolute gain or 37.9% relative gain on recognition satisfaction rate, indicating the strong transferability our method provides without knowledge of the black-box cloud model.

### Table 9

Transfer From PASCAL VOC Object Detection to ImageNet Classification (Accuracy %)

| Task                     | Super-Resolution | Denoising | JPEG-Deblocking |
|--------------------------|------------------|-----------|-----------------|
|                         | R18  | R50  | R101 | D21 | V16 | R18 | R50  | R101 | D21 | V16 | R18 | R50  | R101 | D21 | V16 |
| Plain Processing         | 53.0 | 58.9 | 62.0 | 57.3 | 50.9 | 59.7 | 65.1 | 67.3 | 63.9 | 59.2 | 48.8 | 54.6 | 56.8 | 53.1 | 44.7 |
| RA w/R18                 | 54.6 | 60.2 | 63.4 | 58.8 | 52.7 | 60.8 | 66.7 | 68.8 | 65.2 | 61.1 | 50.8 | 57.2 | 59.6 | 55.4 | 48.5 |
| RA w/R50                 | 54.0 | 59.7 | 63.0 | 58.7 | 52.0 | 60.5 | 66.6 | 68.5 | 64.9 | 60.8 | 50.7 | 56.9 | 59.2 | 55.3 | 48.3 |
| RA w/R101                | 54.1 | 59.8 | 63.3 | 58.7 | 52.5 | 60.2 | 66.1 | 68.3 | 64.6 | 60.6 | 51.3 | 57.2 | 59.5 | 55.5 | 48.3 |
| RA w/V16                 | 54.5 | 60.4 | 63.6 | 59.1 | 52.7 | 60.4 | 66.6 | 68.4 | 64.7 | 60.6 | 50.6 | 56.5 | 58.7 | 54.9 | 47.9 |

A image processing model $P$ trained with detection model $A$ (row) as recognition loss can improve the performance on classification model $B$ (column) over plain processing.

### Table 10

Transfer From ImageNet Classification to PASCAL VOC Object Detection (mAP), Using Unsupervised RA

| Evaluation on | Super-Resolution | Denoising | JPEG-Deblocking |
|---------------|------------------|-----------|-----------------|
|               | R18  | R50  | R101 | V16 | R18 | R50  | R101 | V16 | R18 | R50  | R101 | V16 |
| Plain Processing | 68.5 | 69.7 | 73.1 | 63.2 | 68.1 | 71.6 | 74.1 | 65.7 | 62.4 | 65.6 | 69.5 | 58.3 |
| Unsup. RA w/R18 | 71.3 | 73.4 | 75.3 | 66.8 | 69.0 | 71.3 | 74.3 | 61.1 | 65.2 | 68.1 | 71.3 | 59.8 |
| Unsup. RA w/R50 | 70.7 | 73.3 | 75.0 | 66.6 | 68.9 | 71.7 | 74.4 | 63.1 | 65.4 | 68.5 | 71.2 | 60.0 |
| Unsup. RA w/R101 | 70.7 | 73.2 | 75.0 | 66.2 | 68.9 | 71.3 | 73.9 | 63.3 | 65.2 | 67.9 | 71.1 | 59.6 |
| Unsup. RA w/D21  | 71.0 | 73.2 | 75.1 | 66.6 | 68.7 | 70.3 | 73.0 | 63.8 | 65.9 | 68.6 | 71.4 | 61.1 |
| Unsup. RA w/V16  | 70.3 | 72.3 | 74.3 | 67.0 | 68.5 | 70.7 | 74.0 | 63.6 | 65.9 | 68.2 | 71.5 | 61.1 |

### Table 11

Accuracy (%) on ResNet-18 ImageNet Classification, With More Processing Architectures

| $P$ Architecture | SRRResNet | SRDNet | RDN | DBPN |
|------------------|-----------|--------|-----|------|
| No Processing    | 46.3      | 46.3   | 46.3 | 46.3 |
| Plain Processing | 52.6      | 54.7   | 55.5 | 54.8 |
| RA Processing    | 61.8      | 62.4   | 63.8 | 62.6 |
| RA, Unsupervised | 61.3      | 61.7   | 63.6 | 62.4 |
| RA w/ Transformer| 63.0      | 63.2   | 64.1 | 63.5 |

Processing task is super-resolution.
4.7 Experiments on More Architectures

In previous sections, we used SRResNet [7] as our processing model \( P \). Here we provide more results with other more recent processing models, including SRDenseNet (SRDNet) [6], Residual Dense Network (RDN) [57], and Deep Back-Projection Networks (DBPN) [58]. We present results at Table 11, with super-resolution as the processing task, ImageNet classification as recognition task, and \( R \) being ResNet-18. The general trend we observed before holds for various architectures.

4.8 Experiments on ImageNet-C

We evaluate our methods on the ImageNet-C benchmark [33]. It imposes 17 different types of corruptions on the ImageNet [48] validation set. Despite ImageNet-C benchmark is originally designed for evaluating recognition models that are robust to corruptions, it is a good testbed for our methods in a broader range of processing tasks. We use the corrupted image as the input image to the processing model and the original clean image as the target image. Since only corrupted images from the validation set are released, we divide it evenly for each class into two halves and train/test on its first/second half. The recognition model used in this experiment is an ImageNet-pretrained ResNet-18.

In Table 12, we evaluate RA Processing on all 17 types of corruptions, with “corruption level” set to 5 [33]. We observe that RA Processing brings consistent improvement over plain processing, sometimes by an even larger margin than the tasks considered in Section 4.

In Table 13, we experiment with different levels of corruptions using two corruption types: “speckle noise” and “snow”. We also evaluate with our variants – Unsupervised RA and RA with Transformer. We observe that when the corruption level is higher, our methods tend to bring more recognition accuracy gain. In this case, we note that using a Transformer could sometimes hurt the accuracy compared with plain processing. This is possibly because the insufficient training data in ImageNet-C dataset (half of validation set) caused the transformer to hurt the accuracy, since more parameters typically require more training data. In the majority of other cases, it improves slightly over RA processing.

In Table 14, we examine the transferability of RA Processing between recognition architectures, using the same two tasks “speckle noise” and “snow”, with corruption level 5. Note the recognition loss used during training is from a ResNet-18, and we evaluate the improvement over plain processing on ResNet-50/101, DenseNet-121 and VGG-16. We observe that the improvement over plain processing is transferable among different architectures.
TABLE 15
Randomized and Compound Corruption Experiment
(ImageNet Classification Accuracy %)

|                      | Plain Processing | RA Processing |
|----------------------|------------------|---------------|
| Normal Super-resolution | 52.6             | 61.8          |
| Random Super-resolution | 53.1             | 60.7          |
| Normal Denoising      | 61.9             | 65.1          |
| Random Denoising      | 59.9             | 62.1          |
| Normal JPEG-deblocking | 48.2             | 57.7          |
| Random JPEG-deblocking | 53.6             | 59.1          |
| Random Compound       | 31.5             | 41.2          |

4.9 Experiments on Randomized Image Corruption Tasks

Real-world image corruptions are usually not with a fixed type or degree. To account for this fact, we develop a randomized version of image corruption for each of the image processing tasks and evaluate our methods. For super resolution, we use a Gaussian kernel of $4 \times 4$ with random standard deviations sampled from $[0.5, 1, 1.5, 2, 2.5]$; for denoising, we choose one from the three options uniformly: 1) additive Gaussian noise with standard deviation sampled from $[0.08, 0.10, 0.12, 0.14, 0.16]$, 2) additive uniform noise with range sampled from $[0.1, 0.2, 0.3, 0.4, 0.5]$, 3) multiplicative uniform noise with range sampled from $[0.1, 0.2, 0.3, 0.4, 0.5]$; for JPEG-deblocking, the quality factor is sampled from $[5, 10, 20, 30, 40]$. We further compound these three randomized corruptions sequentially to approximate real world image distortions. We conduct the experiments using ResNet-18 on ImageNet as the recognition model, and the results are shown in Table 15. In all cases, RA processing can boost the recognition accuracy. In the randomized compound experiment, the relative accuracy gain is even more significant ($31.5\% \rightarrow 41.2\%$, a $30.8\%$ relative improvement).

5 IMAGE PROCESSING QUALITY ASSESSMENT

We compare the output image quality using conventional metrics (PSNR/SSIM). When using RA with transformer, the output quality of $P$ is guaranteed unaffected, therefore here we evaluate RA processing. We use R18 as loss on ImageNet, and report results with different $\lambda$ (Eqn. (3)) in Table 16. $\lambda = 0$ corresponds to plain processing.

When $\lambda = 10^{-4}$, PSNR/SSIM are only marginally worse. However, the accuracy obtained is significantly higher. This suggests that the added recognition loss is not harmful when $\lambda$ is chosen properly. When $\lambda$ is excessively large ($10^{-2}$), image quality is hurt more, and interestingly even the recognition accuracy start to decrease, which could be due to the change of actual learning rate. A proper balance between processing and recognition loss is needed for both image quality and accuracy. We also measure the image quality using the PieAPP metric [59], which emphasizes more on perceptual difference: on SR, when $\lambda = 0/10^{-4}/10^{-3}$, PieAPP (lower is better) = 1.329/1.313/1.323. Interestingly, RA processing can slightly improve perceptual quality measured with PieAPP. In Fig. 3, we visualize some examples where the output image is incorrectly classified with a plain processing model, but correctly recognized with RA processing. With smaller $\lambda$ ($10^{-2}$ and $10^{-3}$), the image is nearly the same as the plainly processed images. When $\lambda$ is too large ($10^{-2}$), we could see some extra textures when zooming in.

6 DECISION BOUNDARIES ANALYSIS AND TRANSFERABILITY

Inspired by prior works’ analysis on adversarial example transferability [60], [61], we conduct decision boundary analysis to gain insights on RA processing’s transferability. The task used is SR with ImageNet. We restrict our analysis to a single direction at a time due to image’s high dimension: given a input image $x$ and a direction $d$ (unit vector, same dimension as $x$), we analyze how the output of the recognition model $R$ changes when $x$ moves along $d$ by $\delta$ amount, i.e., when input is $x + \delta s$. We define the boundary distance (BD) of model $R$, with respect to input $x$ and direction $s$, as the minimum amount of movement along $d$ required for $x$ to produce a different label at $R$, or more formally: $BD(R, x, d) = \arg\min_{\delta > 0} R(x + \delta \cdot d) \neq R(x)$.

Consider a two-model scenario, with a source model $R_s$ and a target model $R_t$ sharing the same output categories. We define their inter-boundary distance (IBD): $IBD(R_s, R_t, x, d) = |BD(R_s, x, d) - BD(R_t, x, d)|$. Intuitively, if $R_s(x) = R_t(x)$ (same prediction within boundary), a small IBD between $R_s$ and $R_t$ means they have close boundaries along the $s$ direction, since $x$ does not need to move beyond one’s boundary too far to reach the other’s. In this case, changes made to $x$ along $d$ likely has a transferring effect from source to target model due to their close boundaries.

We take the image $x$ to be a plain processing output, and consider two types of directions: 1. random direction $d_r$. 2. The direction generated by subtracting the plain processing output $x$ from RA processing output $x_s$, i.e., $(x_s - x)/||x_s - x||_2$. The RA processing model here is trained with the source model $R_s$, thus $x_s$ is specific to $R_s$. We call this “RA direction” ($d_{RA}$), since it points to the RA output $x_s$ from the plain output $x$. We take all validation images such that the plain processing output $x$ generates the same wrong prediction when fed to $R_s$ and $R_t$, i.e., $R_s(x) = R_t(x) \neq \text{Ground Truth}$. For each image, we compute $BD(R_s, x, d), BD(R_t, x, d)$ and $IBD(R_s, R_t, x, d)$ with $d$ being random direction and RA direction. In this experiment we present results with $d_r$ being R18 and $d_r$ being R50, as we observe other model pairs produce similar trends.

The average results are shown in Table 17. We first observe that BDs are much smaller alongside the RA direction than the random direction. This indicates moving along the RA direction will change the model’s wrong prediction at $x$ faster, possibly to a correct prediction. More importantly, under either random or RA direction, IBD is always smaller than source/target BDs, which indicates $R_s$ and $R_t$’s boundaries are relatively close, leading to a transferring effect. This
result in RA direction can explain why RA processing can lead to transferable accuracy gains, since the RA loss brings this direction as the effect on the processing output $x$.

We further visualize decision boundaries in Fig. 4 with examples. We use R18 as source and each of the other models as target. Here we plot $d_{RA}$ as the horizontal and $d_r$ as the vertical axis. The origin represents the plain processing output $x$, and the color of point $(u, v)$ represents the predicted class of the image $x + u \cdot d_{RA} + v \cdot d_r$. From the plot, we can see that different models share similar decision boundaries, and also tend to change to the same prediction once we move from the origin along a direction far enough. In both examples, we do confirm that when we move towards RA direction (towards right at horizontal axis), the first color we encounter (green...
for top, purple for bottom) represents the image’s correct label. This suggests the signal from RA loss (RA direction) can correct the wrong prediction with plain processing output \(x\) at origin, and such correction is transferable given the similar decision boundaries among models.

7 Comparison With Alternatives

We analyze some alternatives to our approaches. Unless otherwise specified, experiments in this section are conducted using RA processing on super-resolution, with ResNet-18 trained on ImageNet as the recognition model, and \(\lambda = 10^{-3}\) if used. Under this setting, we achieve 61.8% classification accuracy on the output images.

7.1 Training/Fine-Tuning the Recognition Model

Instead of fixing the recognition model \(R\), we could train/fine-tune it together with the training of image processing model \(P\), to optimize the recognition loss. Many prior works \([37], [38], [40]\) do train/fine-tune the recognition model jointly with the image processing model. We use SGD with momentum as \(R\)'s optimizer, and the final accuracy reaches 63.0%. However, since we do not fix \(R\), it becomes a model that specifically recognizes super-resolved images, and we found its performance on original target images drops from 69.8% to 60.5%. Moreover, when transferring the trained \(P\) on ResNet-50, the accuracy is 62.4%, worse than 66.7% when we train with a fixed ResNet-18.

### Table 17

| Direction | Random  | RA       |
|-----------|---------|----------|
| \(BD(R_s)\) | 34.7    | 14.1     |
| \(BD(R_t)\) | 34.3    | 16.5     |
| \(IBD(R_s, R_t)\) | 25.2    | 10.5     |

**Fig. 4.** Different models’ decision boundaries are similar, especially along the RA direction (horizontal axis).
This suggests we lose some transferability if we do not fix the recognition model $R$.

### 7.2 Training Recognition Models From Scratch

We could first train a super-resolution model, and then train $R$ from scratch on the output images. Doing this, we achieve 66.1% accuracy, higher than 61.8% in RA processing. However, $R$’s accuracy on original clean images drops from 69.8% to 66.1%. Alternatively, we could train $R$ from scratch on interpolated low-resolution images, in which case we achieve 66.0% on interpolated validation data but only 50.2% on the original data. In summary, training/fine-tuning $R$ to cater the need of super-resolved or interpolated images can harm its performance on original images, and causes additional overhead in storing models. In contrast, RA processing could boost the accuracy of output images with the performance on original images intact.

### 7.3 Training Without the Image Processing Loss

It is possible to train the processing model on the recognition loss $L_{recog}$ without even keeping the original image processing loss $L_{proc}$ (Eqn. (3)). This may presumably lead to better recognition performance since the model $P$ can now “focus on” optimizing the recognition loss. However, we found removing the original image processing loss hurts the recognition performance: the accuracy drops from 61.8% to 60.9%; even worse, the SSIM/PSNR metrics drop from 26.69/0.804 to 16.92/0.263, which is reasonable since the image processing loss is not optimized during training. This suggests the original image processing loss is helpful for the recognition accuracy, since it helps the corrupted image to restore to its original form.

### 7.4 Perceptual/Feature Loss

Our unsupervised RA method optimizes the recognition model’s output probability distance between processed and target images. This is related to the perceptual loss (also called feature loss) used in [7], [9]. Perceptual loss optimizes processed and target images’ distance in VGG feature space. Note that the perceptual loss was originally proposed to improve output’s quality from a human observer’s perspective. To compare both methods, we follow [7] to optimize the perceptual loss from VGG-16. We find perceptual loss yields lower accuracy than unsupervised RA (56.7% versus 61.0% on the VGG-16 recognition model). This could be because using final probabilities provides more category supervision, while intermediate features improve the outputs from a perceptual perspective.

### 8 Conclusion

We investigated an important yet largely overlooked problem: enhancing the machine recognition of image processing outputs. We found that a set of simple approaches—that optimize an additional recognition loss—can significantly boost the recognition accuracy with little to no loss in image quality. Moreover, the gain in accuracy can transfer across architectures, categories, and vision tasks unseen during training, or even transfer to a black-box cloud model. This indicates that the enhanced interpretability is not specific to one particular model but generalizable to others. This makes the approaches applicable even when the future downstream recognition models are unknown. Finally we analyzed the reason of such transferability phenomenon from the perspective of decision boundary similarities between recognition models.

### References

[1] C. Dong, C. C. Loy, K. He, and X. Tang, “Learning a deep convolutional network for image super-resolution,” in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 184–199.

[2] J. Xie, L. Xu, and E. Chen, “Image denoising and inpainting with deep neural networks,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 341–349.

[3] D. Eigen, D. Krishnan, and R. Fergus, “Restoring an image taken through a window covered with dirt or rain,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 633–640.

[4] R. Zhang, P. Isola, and A. A. Efros, “Colorful image colorization,” in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 649–666.

[5] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[6] T. Tong, G. Li, X. Liu, and Q. Gao, “Image super-resolution using dense skip connections,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2018, pp. 4809–4817.

[7] C. Ledig et al., “Photo-realistic single image super-resolution using a generative adversarial network,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 105–114.

[8] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 588–595.

[9] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 694–711.

[10] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5967–5976.

[11] R. Tsai, “Multiframe image restoration and registration,” *Adv. Comput. Vis. Image Process.*, vol. 1, pp. 317–339, 1984.

[12] S. C. Park, M. K. Park, and M. G. Kang, “Super-resolution image reconstruction: A technical overview,” *IEEE Signal Process. Mag.*, vol. 20, no. 3, pp. 21–36, May 2003.

[13] L. I. Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms,” *Physica D, Nonlinear Phenom.*, vol. 60, no. 1–4, pp. 259–268, 1992.

[14] E. J. Candès, J. Romberg, and T. Tao, “Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information,” *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[15] Z. Wang, D. Liu, S. Chang, Q. Ling, Y. Yang, and T. S. Huang, “D3: Deep dual-domain based fast restoration of JPEG-compressed images,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2764–2772.

[16] C. Dong, C. C. Loy, and X. Tang, “Accelerating the super-resolution convolutional neural network,” in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 391–407.

[17] J. Kim, J. K. Lee, and K. M. Lee, “Accurate image super-resolution using very deep convolutional networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1646–1654.

[18] W. Shi et al., “Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1874–1883.

[19] J. Kim, J. K. Lee, and K. M. Lee, “Deeply-recursive convolutional network for image super-resolution,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1637–1645.

[20] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, “Deep laplacian pyramid networks for fast and accurate super-resolution,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5835–5843.

[21] J. Chen, J. Chen, H. Chao, and M. Yang, “Image blind denoising with generative adversarial network based noise modeling,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3155–3164.

[22] G. Larsson, M. Maire, and G. Shakhnarovich, “Learning representations for automatic colorization,” in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 577–593.
[23] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, “Image super-resolution using very deep residual channel attention networks,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 294–310.

[24] M. S. M. Sajjadi, B. Scholkopf, and M. Hirsch, “EnhanceNet: Single image super-resolution through automated texture synthesis,” in Proc. IEEE Int. Conf. Comput. Vis., 2017, pp. 4501–4510.

[25] Y. Lee, J. Lee, H. Ahn, and M. Jeon, “SNIDER: Single noisy image denoising and rectification for improving license plate recognition,” in Proc. IEEE Int. Conf. Comput. Vis. Workshops, 2019, pp. 1017–1026.

[26] Z. Wang, S. Chang, Y. Yang, D. Liu, and T. S. Huang, “Studying very low resolution recognition using deep networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 4792–4800.

[27] L. Qin, G. Shakhnarovich, and N. Utkin, “Task-driven super-resolution: Object detection in low-resolution images,” 2018, arXiv:1803.11136.

[28] R. G. VidalMate et al., “Bridging the gap between computational photography and visual recognition,” 2019, arXiv:1901.09482.

[29] S. Banerjee, R. G. VidalMata, Z. Wang, and W. J. Scheirer, “Report on UG+2 challenge task 1: Assessing algorithms to improve video object detection and classification from unconstrained mobility platforms,” 2019, arXiv:1907.11529.

[30] C. Zhao and B. Ghanem, “ThumbNet: One thumbnail contains all you need for recognition,” in Proc. 28th ACM Int. Conf. Multimedia, 2020, pp. 1506–1514.

[31] S. Wang, B. Wen, J. Wu, D. Tao, and Z. Wang, “Segmentation-aware image denoising without knowing true segmentation,” 2019, arXiv:1907.00013.

[32] D. Liu, H. Zhang, and Z. Xiong, “On the classification-distortion-perception tradeoff,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2019, pp. 1206–1215.

[33] D. Hendrycks and T. Dietterich, “Benchmarking neural network robustness to common corruptions and perturbations,” 2019, arXiv:1903.12261.

[34] D. Li, J. Zhang, Y. Yang, C. Liu, Y.-Z. Song, and T. M. Hospedales, “Episodic training for domain generalization,” 2019, arXiv:1902.00113.

[35] S. Shankar, V. Piratla, S. Chakrabarti, S. Chaudhuri, P. Jyothi, and S. Sarawagi, “Generalizing across domains via cross-gradient training,” 2018, arXiv:1804.10745.

[36] S. Palacio, J. Folz, J. Hees, F. Raue, D. Borth, and A. Dengel, “What do deep networks like to see?,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 3108–3117.

[37] Y. Bai, Y. Zhang, M. Ding, and B. Ghanem, “Finding tiny faces in the wild with generative adversarial network,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 21–30.

[38] K. Zhang et al., “Super-identity convolutional neural network for face hallucination,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 196–211.

[39] F. Liu, R. Zhu, D. Zeng, Q. Zhao, and X. Liu, “Disentangling features in 3D face shapes for joint face reconstruction and recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 5216–5225.

[40] V. Sharma, A. Diba, D. Neven, M. S. Brown, L. V. Gool, and R. Stiefelhagen, “Classification-driven dynamic image enhancement,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 4033–4041.

[41] D. Liu, B. Wen, X. Liu, Z. Wang, and T. S. Huang, "When image denoising meets high-level vision tasks: A deep learning approach," 2017, arXiv:1706.04384.

[42] B. Li, X. Peng, Z. Wang, J. Xu, and D. Feng, "End-to-end united video dehazing and detection," in Proc. 32nd AAAI Conf. Artif. Intell., 2018, Art. no. 859.

[43] H. Talebi and P. Milanfar, “Learning to resize images for computer vision tasks,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 487–496.

[44] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” 2014, arXiv:1503.02531.

[45] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in Proc. IEEE Int. Conf. Comput. Vis., 2017, pp. 2242–2251.

[46] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, arXiv:1412.6980.

[47] M. Hocke et al., “Automatic differentiation in pytorch,” in Proc. Int. Conf. Neural Inf. Process. Syst. Workshop, 2017. [Online]. Available: https://papers.nips.cc/paper/2019/hash/bdbca288ee7e792f2fafa9f7012727740-Abstract.html.

[48] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2009, pp. 248–255.

[49] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.

[50] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 2261–2269.

[51] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2015, arXiv:1409.1556.

[52] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” 2015, arXiv:1502.03167.

[53] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2015, pp. 91–99.

[54] J. Yang, J. Lu, D. Batra, and D. Parikh, “A faster pytorch implementation of faster R-CNN.” 2017. [Online]. Available: https://github.com/jwyang/faster-rcnn.pytorch.

[55] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 779–788.

[56] J. Dai, Y. Li, K. He, and J. Sun, “R-FCN: Object detection via region-based fully convolutional networks,” in Proc. Int. Conf. Neural Inf. Process. Syst., 2016, pp. 379–387.

[57] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, “Residual dense network for image super-resolution,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 2472–2481.

[58] M. Haris, G. Shakhnarovich, and N. Utkin, “Deep-back-projection networks for super-resolution,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 1664–1673.

[59] E. Prashnani, H. Cai, Y. Mostofi, and P. Sen, “PieAPP: Perceptual image-error assessment through pairwise preference,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 1808–1817.

[60] Y. Liu, X. Chen, C. Liu, and D. Song, “Delving into transferable adversarial examples and black-box attacks,” 2016, arXiv:1611.02770.

[61] F. Tramer, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, “The space of transferable adversarial examples,” 2017, arXiv:1704.03453.
Zhiqiang Shen received the PhD degree from the School of Computer Science, Fudan University, China. He is currently a postdoctoral researcher with Carnegie Mellon University. He was also a joint-training PhD student with IFP Group, UIUC. His research interests include computer vision, deep learning, and machine learning.

Bingyi Kang received the BE degree in automation from Zhejiang University, Hangzhou, Zhejiang, in 2016, and the PhD degree in electronic and computer engineering from the National University of Singapore. His current research interest includes sample efficient learning and reinforcement learning.

Evan Shelhamer received the PhD degree from UC Berkeley, in 2019, as a member of BAIR advised by Trevor Darrell. During that time, he brewed a lot of code and coffee as the lead developer of the Caffe deep learning framework from version 0.1 to 1.0. Before that, he studied computer science and psychology with UMass Amherst advised by Erik Learned-Miller. His research interests include reconciling visual structure with end-to-end learning and equipping models to adapt to new and different data.

Trevor Darrell received the BSE degree from the University of Pennsylvania, in 1988, and the SM and PhD degrees from MIT, in 1992 and 1996, respectively. He is on the faculty of the CS and EE Divisions, EECS Department, UC Berkeley. He founded and co-leads Berkeley’s Berkeley Artificial Intelligence Research (BAIR) Lab and the Berkeley DeepDrive (BDD) Industrial Consortia. He also is faculty director of the PATH Research Center, UC Berkeley, and previously led the Vision Group, UC-affiliated International Computer Science Institute in Berkeley. Prior to that, he was on the faculty of the MIT EECS Department from 1999-2008, where he directed the Vision Interface Group.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.