Perceptual Losses for Real-Time Style Transfer and Super-Resolution

Justin Johnson, Alexandre Alahi, and Li Fei-Fei

Department of Computer Science, Stanford University
{jcjohns, alahi, feifeili}@cs.stanford.edu

Abstract. We consider image transformation problems, where an input image is transformed into an output image. Recent methods for such problems typically train feed-forward convolutional neural networks using a *per-pixel* loss between the output and ground-truth images. Parallel work has shown that high-quality images can be generated by defining and optimizing *perceptual* loss functions based on high-level features extracted from pretrained networks. We combine the benefits of both approaches, and propose the use of perceptual loss functions for training feed-forward networks for image transformation tasks. We show results on image style transfer, where a feed-forward network is trained to solve the optimization problem proposed by Gatys *et al.* in real-time. Compared to the optimization-based method, our network gives similar qualitative results but is three orders of magnitude faster. We also experiment with single-image super-resolution, where replacing a per-pixel loss with a perceptual loss gives visually pleasing results.

Keywords: Style transfer, super-resolution, deep learning

1 Introduction

Many classic problems can be framed as *image transformation* tasks, where a system receives some input image and transforms it into an output image. Examples from image processing include denoising, super-resolution, and colorization, where the input is a degraded image (noisy, low-resolution, or grayscale) and the output is a high-quality color image. Examples from computer vision include semantic segmentation and depth estimation, where the input is a color image and the output image encodes semantic or geometric information about the scene.

One approach for solving image transformation tasks is to train a feed-forward convolutional neural network in a supervised manner, using a per-pixel loss function to measure the difference between output and ground-truth images. This approach has been used for example by Dong *et al.* for super-resolution [1], by Cheng *et al.* for colorization [2, 3], by Long *et al.* for segmentation [4], and by Eigen *et al.* for depth and surface normal prediction [5, 6]. Such approaches are efficient at test-time, requiring only a forward pass through the trained network.

However, the per-pixel losses used by these methods do not capture *perceptual* differences between output and ground-truth images. For example, consider two

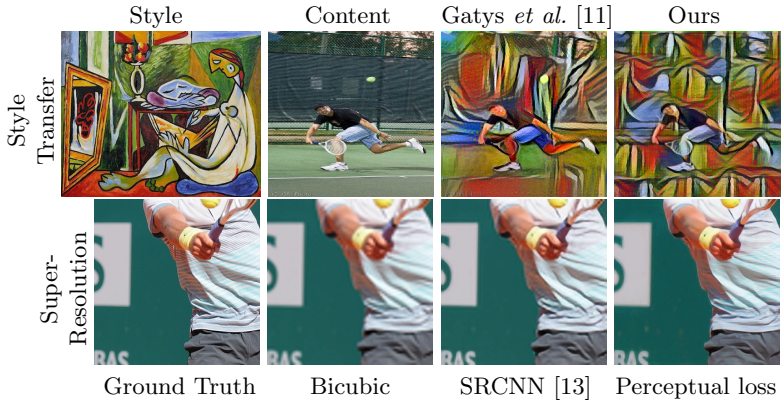

Fig. 1. Example results for style transfer (top) and $\times 4$ super-resolution (bottom). For style transfer, we achieve similar results as Gatys *et al.* [11] but are three orders of magnitude faster. For super-resolution our method trained with a perceptual loss is able to better reconstruct fine details compared to methods trained with per-pixel loss.

identical images offset from each other by one pixel; despite their perceptual similarity they would be very different as measured by per-pixel losses.

In parallel, recent work has shown that high-quality images can be generated using *perceptual loss functions* based not on differences between pixels but instead on differences between high-level image feature representations extracted from pretrained convolutional neural networks. Images are generated by minimizing a loss function. This strategy has been applied to feature inversion [7] by Mahendran *et al.*, to feature visualization by Simonyan *et al.* [8] and Yosinski *et al.* [9], and to texture synthesis and style transfer by Gatys *et al.* [10–12]. These approaches produce high-quality images, but are slow since inference requires solving an optimization problem.

In this paper we combine the benefits of these two approaches. We train feed-forward *transformation networks* for image transformation tasks, but rather than using *per-pixel* loss functions depending only on low-level pixel information, we train our networks using *perceptual loss functions* that depend on high-level features from a pretrained *loss network*. During training, perceptual losses measure image similarities more robustly than per-pixel losses, and at test-time the transformation networks run in real-time.

We experiment on two tasks: style transfer and single-image super-resolution. Both are inherently ill-posed; for style transfer there is no single correct output, and for super-resolution there are many high-resolution images that could have generated the same low-resolution input. Success in either task requires semantic reasoning about the input image. For style transfer the output must be semantically similar to the input despite drastic changes in color and texture; for super-resolution fine details must be inferred from visually ambiguous low-resolution inputs. In principle a high-capacity neural network trained for either task could implicitly learn to reason about the relevant semantics; however, in practice we

need not learn from scratch: the use of perceptual loss functions allows the transfer of semantic knowledge from the loss network to the transformation network.

For style transfer our feed-forward networks are trained to solve the optimization problem from [11]; our results are similar to [11] both qualitatively and as measured by objective function value, but are three orders of magnitude faster to generate. For super-resolution we show that replacing the per-pixel loss with a perceptual loss gives visually pleasing results for $\times 4$ and $\times 8$ super-resolution.

2 Related Work

Feed-forward image transformation. In recent years, a wide variety of image transformation tasks have been trained with per-pixel loss functions.

Semantic segmentation methods [4, 6, 14–17] produce dense scene labels by running networks in a fully-convolutional manner over input images, training with a per-pixel classification loss. Recent methods for depth [6, 5, 18] and surface normal estimation [6, 19] are similar, transforming color input images into geometrically meaningful output images using a feed-forward convolutional network trained with per-pixel regression [5, 6] or classification [19] losses. Some methods move beyond per-pixel losses by penalizing image gradients [6], framing CRF inference as a recurrent layer trained jointly with the rest of the network [17], or using a CRF loss layer [18] to enforce local consistency in the output.

The architecture of our transformation networks are inspired by [4] and [16], which use in-network downsampling to reduce the spatial extent of feature maps followed by in-network upsampling to produce the final output image.

Perceptual optimization. A number of recent papers have used optimization to generate images where the objective is perceptual, depending on high-level features extracted from a convolutional network. Images can be generated to maximize class prediction scores [8, 9] or individual features [9] in order to understand the functions encoded in trained networks. Similar optimization techniques can also be used to generate high-confidence fooling images [20, 21].

Mahendran and Vedaldi [7] invert features from convolutional networks by minimizing a feature reconstruction loss in order to understand the image information retained by different network layers; similar methods had previously been used to invert local binary descriptors [22, 23] and HOG features [24].

The work of Dosovitskiy and Brox [25] is particularly relevant to ours, as they train a feed-forward neural network to invert convolutional features, quickly approximating a solution to the optimization problem posed by [7]. However, their feed-forward network is trained with a per-pixel reconstruction loss, while our networks directly optimize the feature reconstruction loss of [7].

Style Transfer. Gatys *et al.* [11] perform artistic style transfer, combining the *content* of one image with the *style* of another by jointly minimizing the feature reconstruction loss of [7] and a *style reconstruction loss* also based on features extracted from a pretrained convolutional network; a similar method had previously been used for texture synthesis [10]. Their method produces high-quality results, but is computationally expensive since each step of the optimiza-

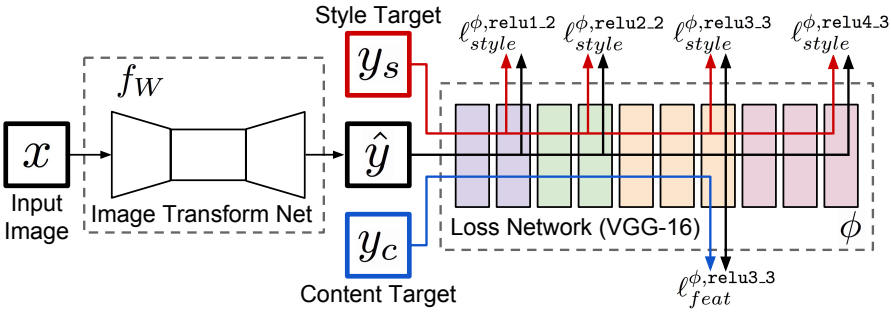

Fig. 2. System overview. We train an *image transformation network* to transform input images into output images. We use a *loss network* pretrained for image classification to define *perceptual loss functions* that measure perceptual differences in content and style between images. The loss network remains fixed during the training process.

tion problem requires a forward and backward pass through the pretrained network. To overcome this computational burden, we train a feed-forward network to quickly approximate solutions to their optimization problem. Concurrent with our work, [26, 27] also propose feed-forward approaches for fast style transfer.

Image super-resolution. Image super-resolution is a classic problem for which a variety of techniques have been developed. Yang *et al.* [28] provide an exhaustive evaluation of the prevailing techniques prior to the widespread adoption of convolutional neural networks. They group super-resolution techniques into prediction-based methods (bilinear, bicubic, Lanczos, [29]), edge-based methods [30, 31], statistical methods [32–34], patch-based methods [30, 35–41], and sparse dictionary methods [42, 43]. Recently [1] achieved excellent performance on single-image super-resolution using a three-layer convolutional neural network with a per-pixel Euclidean loss. Other recent methods include [44–46].

3 Method

As shown in Figure 2, our system consists of two components: an *image transformation network* f_W and a *loss network* ϕ that is used to define several *loss functions* $\ell_1, \ldots, \ell_k$. The image transformation network is a deep residual convolutional neural network parameterized by weights W; it transforms input images x into output images $\hat{y}$ via the mapping $\hat{y} = f_W(x)$. Each loss function computes a scalar value $\ell_i(\hat{y}, y_i)$ measuring the difference between the output image $\hat{y}$ and a *target image* y_i. The image transformation network is trained using stochastic gradient descent to minimize a weighted combination of loss functions:

$$W^* = \arg\min_W \mathbf{E}_{x,\{y_i\}} \left[\sum_{i=1} \lambda_i \ell_i(f_W(x), y_i) \right] \tag{1}$$

To address the shortcomings of per-pixel losses and allow our loss functions to better measure perceptual and semantic differences between images, we draw inspiration from recent work that generates images via optimization [7–11]. The key insight of these methods is that convolutional neural networks pretrained for image classification have already learned to encode the perceptual and semantic information we would like to measure in our loss functions. We therefore make use of a network ϕ pretrained for image classification as a fixed *loss network* in order to define our loss functions. Our deep convolutional transformation network is thus trained using loss functions that are also deep convolutional networks.

We use the loss network ϕ to define a *feature reconstruction loss* ℓ^{ϕ}_{feat} and *style reconstruction loss* ℓ^{ϕ}_{style} that measure differences in content and style between images. For each input image x we have a *content target* y_c and a *style target* y_s. For style transfer the content target y_c is the input image x and the output image $\hat{y}$ should combine the content of $x = y_c$ with the style of y_s; we train one network per style target. For super-resolution the input x is a low-resolution input, the content target y_c is the ground-truth high-resolution image, and style reconstruction loss is not used; we train one network per super-resolution factor.

3.1 Image Transformation Networks

Our image transformation networks roughly follow the architectural guidelines set forth by [47]. We eschew pooling layers, instead using strided and fractionally strided convolutions for in-network downsampling and upsampling. Our network body comprises five residual blocks [48] using the architecture of [49]. All non-residual convolutional layers are followed by batch normalization [50] and ReLU nonlinearities with the exception of the output layer, which instead uses a scaled tanh to ensure that the output has pixels in the range $[0, 255]$. The first and last layers use 9×9 kernels; all other convolutional layers use 3×3 kernels. The exact architectures of our networks can be found in the supplementary material[1].

Inputs and Outputs. For style transfer the input and output are color images of shape $3 \times 256 \times 256$. For super-resolution with upsampling factor f, the output is a high-resolution patch of shape $3 \times 288 \times 288$ and the input is a low-resolution patch of shape $3 \times 288/f \times 288/f$. Since the image transformation networks are fully-convolutional, at test-time they can be applied to images of any resolution.

Downsampling and Upsampling. For super-resolution with an upsampling factor of f, we use several residual blocks followed by $\log_2 f$ convolutional layers with stride $1/2$. This is different from [1] who use bicubic interpolation to upsample the low-resolution input before passing it to the network. Rather than relying on a fixed upsampling function, fractionally-strided convolution allows the upsampling function to be learned jointly with the rest of the network.

For style transfer our networks use two stride-2 convolutions to downsample the input followed by several residual blocks and then two convolutional layers with stride $1/2$ to upsample. Although the input and output have the same size, there are several benefits to networks that downsample and then upsample.

[1] Available at the first author's website.

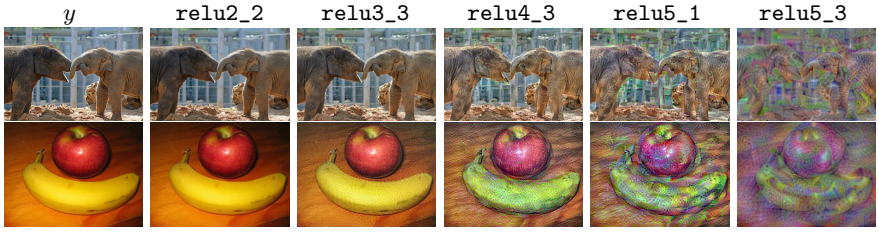

Fig. 3. Similar to [7], we use optimization to find an image $\hat{y}$ that minimizes the feature reconstruction loss $\ell_{feat}^{\phi,j}(\hat{y}, y)$ for several layers j from the pretrained VGG-16 loss network ϕ. As we reconstruct from higher layers, image content and overall spatial structure are preserved, but color, texture, and exact shape are not.

The first is computational. With a naive implementation, a 3×3 convolution with C filters on an input of size $C \times H \times W$ requires $9HWC^2$ multiply-adds, which is the same cost as a 3×3 convolution with DC filters on an input of shape $DC \times H/D \times W/D$. After downsampling, we can therefore use a larger network for the same computational cost.

The second benefit has to do with effective receptive field sizes. High-quality style transfer requires changing large parts of the image in a coherent way; therefore it is advantageous for each pixel in the output to have a large effective receptive field in the input. Without downsampling, each additional 3×3 convolution increases the effective receptive field size by 2. After downsampling by a factor of D, each 3×3 convolution instead increases effective receptive field size by $2D$, giving larger effective receptive fields with the same number of layers.

Residual Connections. He *et al.* [48] use *residual connections* to train very deep networks for image classification. They argue that residual connections make the identity function easier to learn; this is an appealing property for image transformation networks, since in most cases the output image should share structure with the input image. The body of our network thus consists of several residual blocks, each of which contains two 3×3 convolutional layers. We use the residual block design of [49], shown in the supplementary material.

3.2 Perceptual Loss Functions

We define two *perceptual loss functions* that measure high-level perceptual and semantic differences between images. They make use of a *loss network* ϕ pretrained for image classification, meaning that these perceptual loss functions are themselves deep convolutional neural networks. In all our experiments, the *loss network* ϕ is the 16-layer VGG network [51] pretrained on ImageNet [52].

Feature Reconstruction Loss. Rather than encouraging the pixels of the output image $\hat{y} = f_W(x)$ to exactly match the pixels of the target image y, we instead encourage them to have similar feature representations as computed by the loss network ϕ. Let $\phi_j(x)$ be the activations of the jth layer of the network ϕ when processing the image x; if j is a convolutional layer then $\phi_j(x)$ will be

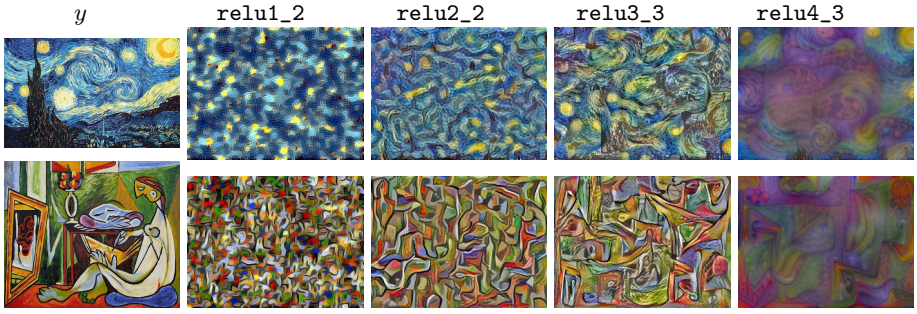

Fig. 4. Similar to [11], we use optimization to find an image $\hat{y}$ that minimizes the style reconstruction loss $\ell^{\phi,j}_{style}(\hat{y}, y)$ for several layers j from the pretrained VGG-16 loss network ϕ. The images $\hat{y}$ preserve stylistic features but not spatial structure.

a feature map of shape $C_j \times H_j \times W_j$. The *feature reconstruction loss* is the (squared, normalized) Euclidean distance between feature representations:

$$\ell^{\phi,j}_{feat}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \|\phi_j(\hat{y}) - \phi_j(y)\|_2^2 \tag{2}$$

As demonstrated in [7] and reproduced in Figure 3, finding an image $\hat{y}$ that minimizes the feature reconstruction loss for early layers tends to produce images that are visually indistinguishable from y. As we reconstruct from higher layers, image content and overall spatial structure are preserved but color, texture, and exact shape are not. Using a feature reconstruction loss for training our image transformation networks encourages the output image $\hat{y}$ to be perceptually similar to the target image y, but does not force them to match exactly.

Style Reconstruction Loss. The feature reconstruction loss penalizes the output image $\hat{y}$ when it deviates in content from the target y. We also wish to penalize differences in style: colors, textures, common patterns, etc. To achieve this effect, Gatys *et al.* [10, 11] propose the following *style reconstruction loss*.

As above, let $\phi_j(x)$ be the activations at the jth layer of the network ϕ for the input x, which is a feature map of shape $C_j \times H_j \times W_j$. Define the *Gram matrix* $G^\phi_j(x)$ to be the $C_j \times C_j$ matrix whose elements are given by

$$G^\phi_j(x)_{c,c'} = \frac{1}{C_j H_j W_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \phi_j(x)_{h,w,c} \phi_j(x)_{h,w,c'}. \tag{3}$$

If we interpret $\phi_j(x)$ as giving C_j-dimensional features for each point on a $H_j \times W_j$ grid, then $G^\phi_j(x)$ is proportional to the uncentered covariance of the C_j-dimensional features, treating each grid location as an independent sample. It thus captures information about which features tend to activate together. The Gram matrix can be computed efficiently by reshaping $\phi_j(x)$ into a matrix ψ of shape $C_j \times H_j W_j$; then $G^\phi_j(x) = \psi\psi^T / C_j H_j W_j$.

The *style reconstruction loss* is then the squared Frobenius norm of the difference between the Gram matrices of the output and target images:

$$\ell^{\phi,j}_{style}(\hat{y}, y) = \|G^{\phi}_j(\hat{y}) - G^{\phi}_j(y)\|^2_F. \tag{4}$$

The style reconstruction loss is well-defined even when $\hat{y}$ and y have different sizes, since their Gram matrices will both have the same shape.

As demonstrated in [11] and reproduced in Figure 5, generating an image $\hat{y}$ that minimizes the style reconstruction loss preserves stylistic features from the target image, but does not preserve its spatial structure. Reconstructing from higher layers transfers larger-scale structure from the target image.

To perform style reconstruction from a set of layers J rather than a single layer j, we define $\ell^{\phi,J}_{style}(\hat{y}, y)$ to be the sum of losses for each layer $j \in J$.

3.3 Simple Loss Functions

In addition to the perceptual losses defined above, we also define two simple loss functions that depend only on low-level pixel information.

Pixel Loss. The *pixel loss* is the (normalized) Euclidean distance between the output image $\hat{y}$ and the target y. If both have shape $C \times H \times W$, then the pixel loss is defined as $\ell_{pixel}(\hat{y}, y) = \|\hat{y} - y\|^2_2/CHW$. This can only be used when when we have a ground-truth target y that the network is expected to match.

Total Variation Regularization. To encourage spatial smoothness in the output image $\hat{y}$, we follow prior work on feature inversion [7, 22] and super-resolution [53, 54] and make use of *total variation regularizer* $\ell_{TV}(\hat{y})$.

4 Experiments

We perform experiments on two image transformation tasks: style transfer and single-image super-resolution. Prior work on style transfer has used optimization to generate images; our feed-forward networks give similar qualitative results but are up to three orders of magnitude faster. Prior work on single-image super-resolution with convolutional neural networks has used a per-pixel loss; we show encouraging qualitative results by using a perceptual loss instead.

4.1 Style Transfer

The goal of style transfer is to generate an image $\hat{y}$ that combines the content of a *target content image* y_c with the the *style* of a *target style image* y_s. We train one image transformation network per style target for several hand-picked style targets and compare our results with the baseline approach of Gatys *et al.* [11].

Baseline. As a baseline, we reimplement the method of Gatys *et al.* [11]. Given style and content targets y_s and y_c and layers j and J at which to perform feature and style reconstruction, an image $\hat{y}$ is generated by solving the problem

$$\hat{y} = \arg\min_y \lambda_c \ell^{\phi,j}_{feat}(y, y_c) + \lambda_s \ell^{\phi,J}_{style}(y, y_s) + \lambda_{TV} \ell_{TV}(y) \tag{5}$$

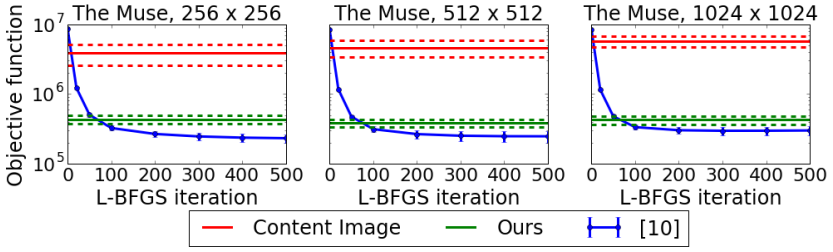

Fig. 5. Our style transfer networks and [11] minimize the same objective. We compare their objective values on 50 images; dashed lines and error bars show standard deviations. Our networks are trained on 256×256 images but generalize to larger images.

where λ_c, λ_s, and λ_{TV} are scalars, y is initialized with white noise, and optimization is performed using L-BFGS. Unconstrained optimization of Equation 5 often results in images with pixels outside the range $[0, 255]$. For a more fair comparison with our method whose output is constrained to this range, for the baseline we minimize Equation 5 using projected L-BFGS by clipping the image y to the range $[0, 255]$ at each iteration. Optimization usually converges to satisfactory results within 500 iterations. This method is slow because each iteration requires a forward and backward pass through the VGG-16 loss network ϕ.

Training Details. We train style transfer networks on the MS-COCO dataset [55]. We resize each of the 80k training images to 256×256 and train with a batch size of 4 for 40k iterations, giving roughly two epochs over the training data. We use Adam [56] with learning rate 1×10^{-3}. The output images are regularized with total variation regularization with a strength of between 1×10^{-6} and 1×10^{-4}, chosen via cross-validation per style target. We do not use weight decay or dropout, as the model does not overfit within two epochs. For all style transfer experiments we compute feature reconstruction loss at layer `relu3_3` and style reconstruction loss at layers `relu1_2`, `relu2_2`, `relu3_3`, and `relu4_3` of the VGG-16 loss network ϕ. Our implementation uses Torch [57] and cuDNN [58]; training takes roughly 4 hours on a single GTX Titan X GPU.

Qualitative Results. In Figure 6 we show qualitative examples comparing our results with the baseline for a variety of style and content images. In all cases the hyperparameters λ_c, λ_s, and λ_{TV} are exactly the same between the two methods; all content images come from the MS-COCO 2014 validation set.

Although our models are trained with 256×256 images, they can be applied in a fully-convolutional manner to images of any size at test-time. In Figure 7 we show examples of style transfer using our models on 512×512 images.

Overall our results are qualitatively similar to the baseline, but in some cases our method produces images with more repetitive patterns. For example in the Starry Night images in Figure 6, our method produces repetitive (but not identical) yellow splotches; the effect can become more obvious at higher resolutions, as seen in Figure 7.

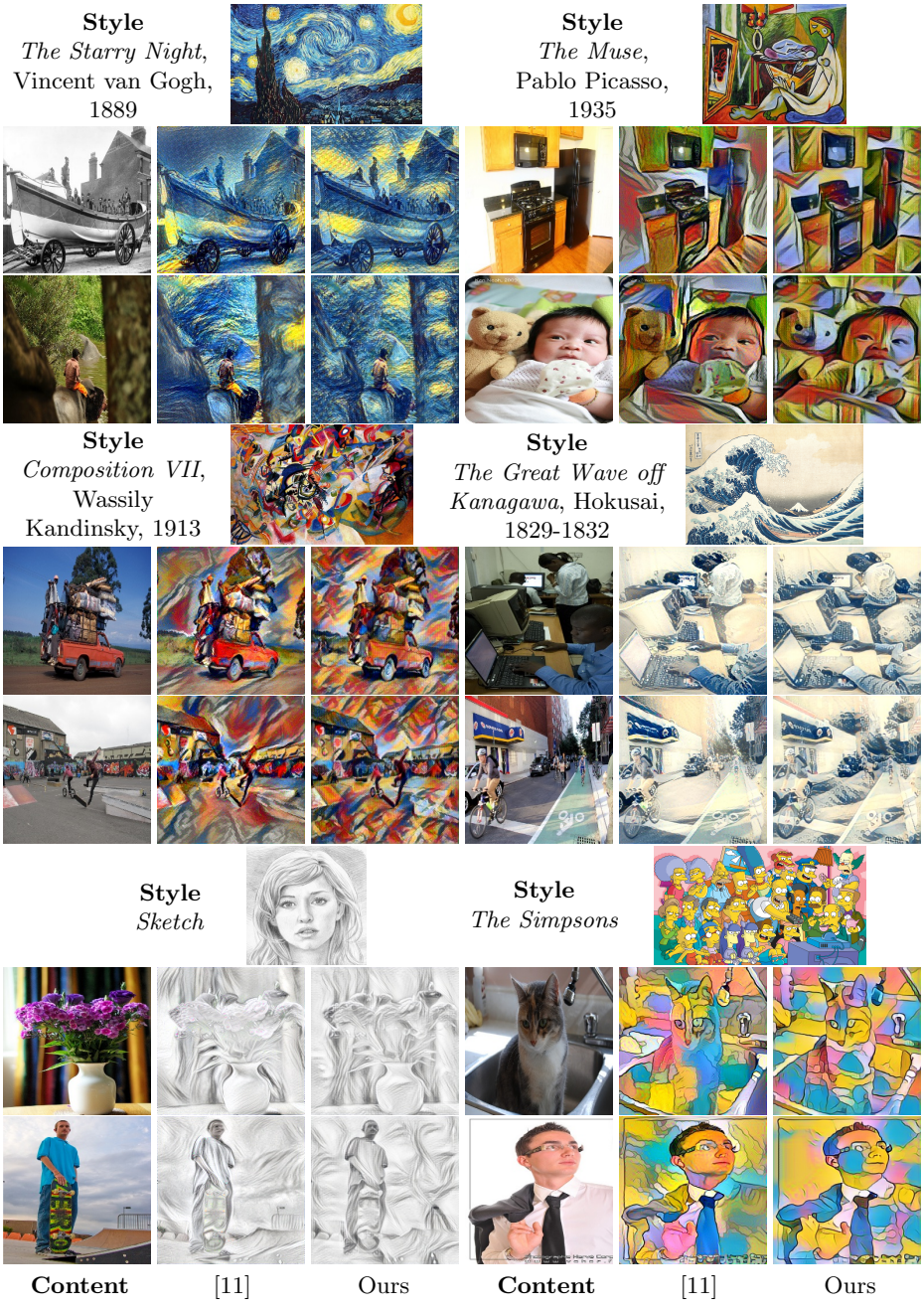

Fig. 6. Example results of style transfer using our image transformation networks. Our results are qualitatively similar to Gatys *et al.* [11] but are much faster to generate (see Table 1). All generated images are 256×256 pixels.

Fig. 7. Example results for style transfer on 512×512 images by applying models trained on 256×256 images. The style images are the same as Figure 6.

Quantitative Results. The baseline and our method both minimize Equation 5. The baseline performs explicit optimization over the output image, while our method is trained to find a solution for any content image y_c in a single forward pass. We may therefore quantitatively compare the two methods by measuring the degree to which they successfully minimize Equation 5.

We run our method and the baseline on 50 images from the MS-COCO validation set, using *The Muse* as a style image. For the baseline we record the value of the objective function at each iteration of optimization, and for our method we record the value of Equation 5 for each image; we also compute the value of Equation 5 when y is equal to the content image y_c. Results are shown in Figure 5. The content image y_c achieves a very high loss, and our method achieves a loss comparable to 50 to 100 iterations of explicit optimization.

Although our networks are trained to minimize Equation 5 for 256×256 images, they also succeed at minimizing the objective when applied to larger images. We repeat the same quantitative evaluation for 50 images at 512×512 and 1024×1024; results are shown in Figure 5. Even at higher resolutions our model achieves a loss comparable to 50 to 100 iterations of the baseline method.
Speed. Table 1 compares the runtime of our method and the baseline for several image sizes; for the baseline we report times for varying numbers of optimization iterations. Across all image sizes, our method takes about half the time of a single iteration of the baseline. Compared to 500 iterations of the baseline method, our method is three orders of magnitude faster. Our method processes 512×512 images at 20 FPS, making it feasible to run in real-time or on video.

4.2 Single-Image Super-Resolution

In single-image super-resolution, the task is to generate a high-resolution output image from a low-resolution input. This is an inherently ill-posed problem, since for each low-resolution image there exist multiple high-resolution images that could have generated it. The ambiguity becomes more extreme as the

	Gatys *et al.* [11]				Speedup		
Image Size	100	300	500	Ours	100	300	500
256×256	3.17	9.52s	15.86s	**0.015s**	212x	636x	**1060x**
512×512	10.97	32.91s	54.85s	**0.05s**	205x	615x	**1026x**
1024×1024	42.89	128.66s	214.44s	**0.21s**	208x	625x	**1042x**

Table 1. Speed (in seconds) for our style transfer networks vs the baseline for various iterations and resolutions. We achieve similar qualitative results (Figure 6) in less time than a single optimization step of the baseline. All benchmarks use a Titan X GPU.

super-resolution factor grows; for large factors ($\times 4$, $\times 8$), fine details of the high-resolution image may have little or no evidence in its low-resolution version.

To overcome this problem, we train super-resolution networks not with the per-pixel loss typically used [1] but instead with a feature reconstruction loss (see Section 3) to allow transfer of semantic knowledge from the pretrained loss network to the super-resolution network. We focus on $\times 4$ and $\times 8$ super-resolution since larger factors require more semantic reasoning about the input.

The traditional metrics used to evaluate super-resolution are PSNR and SSIM [59], both of which have been found to correlate poorly with human assessment of visual quality [60–62]. PSNR and SSIM rely on low-level differences between pixels, and PSNR operates under the assumption of additive Gaussian noise. In addition, PSNR is equivalent to the per-pixel loss ℓ_{pixel}, so as measured by PSNR a model trained to minimize per-pixel loss should always outperform a model trained to minimize feature reconstruction loss. We therefore emphasize that the goal of these experiments is not to achieve state-of-the-art PSNR or SSIM results, but instead to showcase the qualitative difference between models trained with per-pixel and feature reconstruction losses.

Model Details. We train models to perform $\times 4$ and $\times 8$ super-resolution by minimizing feature reconstruction loss at layer `relu2_2` from the VGG-16 loss network ϕ. We train with 288×288 patches from 10k images from the MS-COCO training set, and prepare low-resolution inputs by blurring with a Gaussian kernel of width $\sigma = 1.0$ and downsampling with bicubic interpolation. We train with a batch size of 4 for 200k iterations using Adam [56] with a learning rate of 1×10^{-3} without weight decay or dropout. As a post-processing step, we perform histogram matching between our network output and the low-resolution input.

Baselines. As a baseline model we use SRCNN [1] for its state-of-the-art performance. SRCNN is a three-layer convolutional network trained to minimize per-pixel loss on 33×33 patches from the ILSVRC 2013 detection dataset. SRCNN is not trained for $\times 8$ super-resolution, so we can only evaluate it on $\times 4$.

SRCNN is trained for more than 10^9 iterations, which is not computationally feasible for our models. To account for differences between SRCNN and our model in data, training, and architecture, we train image transformation networks for $\times 4$ and $\times 8$ super-resolution using ℓ_{pixel}; these networks use identical data, architecture, and training as the networks trained to minimize ℓ_{feat}.

Ground Truth	**Bicubic**	**Ours** (ℓ_{pixel})	**SRCNN** [13]	**Ours** (ℓ_{feat})
This image	31.78 / 0.8577	31.47 / 0.8573	32.99 / 0.8784	29.24 / 0.7841
Set5 mean	28.43 / 0.8114	28.40 / 0.8205	30.48 / 0.8628	27.09 / 0.7680

Ground Truth	**Bicubic**	**Ours** (ℓ_{pixel})	**SRCNN** [13]	**Ours** (ℓ_{feat})
This Image	21.69 / 0.5840	21.66 / 0.5881	22.53 / 0.6524	21.04 / 0.6116
Set14 mean	25.99 / 0.7301	25.75 / 0.6994	27.49 / 0.7503	24.99 / 0.6731
BSD100 mean	25.96 / 0.682	25.91 / 0.6680	26.90 / 0.7101	24.95 / 63.17

Fig. 8. Results for $\times 4$ super-resolution on images from Set5 (top) and Set14 (bottom). We report PSNR / SSIM for each example and the mean for each dataset. More results (including FSIM [63] and VIF [64] metrics) are shown in the supplementary material.

Evaluation. We evaluate all models on the standard Set5 [65], Set14 [66], and BSD100 [46] datasets. We report PSNR and SSIM [59], computing both only on the Y channel after converting to the YCbCr colorspace, following [1, 44].

Results. We show results for $\times 4$ super-resolution in Figure 8. Compared to the other methods, our model trained for feature reconstruction does a very good job at reconstructing sharp edges and fine details, such as the eyelashes in the first image and the individual elements of the hat in the second image.

In addition to the automated metrics shown in Figure 8, we also ran a user study on Amazon Mechanical Turk to evaluate our $\times 4$ results on the BSD100 dataset. In each trial workers were shown a nearest-neighbor upsampling of an image and results from two methods, and were asked to pick the result they preferred. All trials were randomized and five workers evaluated each image pair. Between SRCNN and ℓ_{feat}, a majority of workers preferred ℓ_{feat} on 96% of images. More details of this study can be found in the supplementary material.

Results for $\times 8$ super-resolution are shown in Figure 9. Again we see that our ℓ_{feat} model does a good job at edges and fine details compared to other models, such as the horse's legs and hooves. The ℓ_{feat} model does not sharpen edges

Ground Truth	Bicubic	Ours (ℓ_{pixel})	Ours (ℓ_{feat})
This image	22.75 / 0.5946	23.42 / 0.6168	21.90 / 0.6083
Set5 mean	23.80 / 0.6455	24.77 / 0.6864	23.26 / 0.7058
Set14 mean	22.37 / 0.5518	23.02 / 0.5787	21.64 / 0.5837
BSD100 mean	22.11 / 0.5322	22.54 / 0.5526	21.35 / 0.5474

Fig. 9. Results for $\times 8$ super-resolution results on an image from the BSD100 dataset. We report PSNR / SSIM for the example image and the mean for each dataset. More results (including FSIM [63] and VIF [64]) are shown in the supplementary material.

indiscriminately; compared to the ℓ_{pixel} model, the ℓ_{feat} model sharpens the boundary edges of the horse and rider but the background trees remain diffuse, suggesting that the ℓ_{feat} model may be more aware of image semantics.

Many of the results from our ℓ_{feat} models have grid-like artifacts at the pixel level which harm their PSNR and SSIM compared to baseline methods. Similar artifacts are visible in Figure 3 upon magnification, suggesting that they are a result of the feature reconstruction loss and not the architecture of the image transformation network. Figure 3 shows more pronounced distortions as images are reconstructed from higher-level features, motivating the use of the `relu2_2` features used for training our ℓ_{feat} super-resolution models.

Since our ℓ_{pixel} and our ℓ_{feat} models share the same architecture, data, and training procedure, all differences between them are due to the difference between the ℓ_{pixel} and ℓ_{feat} losses. The ℓ_{pixel} loss gives fewer visual artifacts and higher PSNR values but the ℓ_{feat} loss does a better job at reconstructing fine details, leading to pleasing visual results.

5 Conclusion

In this paper we have combined the benefits of feed-forward image transformation tasks and optimization-based methods for image generation by training feed-forward transformation networks with perceptual loss functions. We have applied this method to style transfer where we achieve comparable performance and drastically improved speed compared to existing methods, and to single-image super-resolution where training with a perceptual loss allows the model to better reconstruct fine details and edges. In future work we hope to explore the use of perceptual loss functions for other image transformation tasks.

Acknowledgments Our work is supported by an ONR MURI grant, Yahoo! Labs, and a hardware donation from NVIDIA.

References

1. Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks. IEEE TPAMI (2016)
2. Cheng, Z., Yang, Q., Sheng, B.: Deep colorization. In: ICCV. (2015)
3. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. ECCV (2016)
4. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR. (2015)
5. Eigen, D., Puhrsch, C., Fergus, R.: Depth map prediction from a single image using a multi-scale deep network. In: NIPS. (2014)
6. Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: ICCV. (2015)
7. Mahendran, A., Vedaldi, A.: Understanding deep image representations by inverting them. In: CVPR. (2015)
8. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. In: ICLR Workshop. (2014)
9. Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., Lipson, H.: Understanding neural networks through deep visualization. In: ICML Deep Learning Workshop. (2015)
10. Gatys, L.A., Ecker, A.S., Bethge, M.: Texture synthesis using convolutional neural networks. In: NIPS. (2015)
11. Gatys, L.A., Ecker, A.S., Bethge, M.: A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576 (2015)
12. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: CVPR. (2016)
13. Dong, C., Loy, C.C., He, K., Tang, X.: Learning a deep convolutional network for image super-resolution. In: ECCV. (2014)
14. Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Learning hierarchical features for scene labeling. IEEE TPAMI **35**(8) (2013) 1915–1929
15. Pinheiro, P.H., Collobert, R.: Recurrent convolutional neural networks for scene labeling. In: ICML. (2014)
16. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: ICCV. (2015)
17. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H.: Conditional random fields as recurrent neural networks. In: ICCV. (2015)
18. Liu, F., Shen, C., Lin, G.: Deep convolutional neural fields for depth estimation from a single image. In: CVPR. (2015)
19. Wang, X., Fouhey, D., Gupta, A.: Designing deep networks for surface normal estimation. In: CVPR. (2015)
20. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. In: ICLR. (2014)
21. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: CVPR. (2015)
22. d'Angelo, E., Alahi, A., Vandergheynst, P.: Beyond bits: Reconstructing images from local binary descriptors. In: ICPR. (2012)
23. d'Angelo, E., Jacques, L., Alahi, A., Vandergheynst, P.: From bits to images: Inversion of local binary descriptors. IEEE transactions on pattern analysis and machine intelligence **36**(5) (2014) 874–887

24. Vondrick, C., Khosla, A., Malisiewicz, T., Torralba, A.: Hoggles: Visualizing object detection features. In: ICCV. (2013)
25. Dosovitskiy, A., Brox, T.: Inverting visual representations with convolutional networks. In: CVPR. (2016)
26. Ulyanov, D., Lebadev, V., Vedaldi, A., Lempitsky, V.: Texture networks: Feed-forward synthesis of textures and stylized images. In: ICML. (2016)
27. Li, C., Wand, M.: Precomputed real-time texture synthesis with markovian generative adversarial networks. In: ECCV. (2016)
28. Yang, C.Y., Ma, C., Yang, M.H.: Single-image super-resolution: a benchmark. In: ECCV. (2014)
29. Irani, M., Peleg, S.: Improving resolution by image registration. CVGIP: Graphical models and image processing **53**(3) (1991) 231–239
30. Freedman, G., Fattal, R.: Image and video upscaling from local self-examples. ACM Transactions on Graphics (TOG) **30**(2) (2011) 12
31. Sun, J., Sun, J., Xu, Z., Shum, H.Y.: Image super-resolution using gradient profile prior. In: CVPR. (2008)
32. Shan, Q., Li, Z., Jia, J., Tang, C.K.: Fast image/video upsampling. In: ACM Transactions on Graphics (TOG). Volume 27., ACM (2008) 153
33. Kim, K.I., Kwon, Y.: Single-image super-resolution using sparse regression and natural image prior. IEEE TPAMI **32**(6) (2010) 1127–1133
34. Xiong, Z., Sun, X., Wu, F.: Robust web image/video super-resolution. Image Processing, IEEE Transactions on **19**(8) (2010) 2017–2028
35. Freeman, W.T., Jones, T.R., Pasztor, E.C.: Example-based super-resolution. Computer Graphics and Applications, IEEE **22**(2) (2002) 56–65
36. Chang, H., Yeung, D.Y., Xiong, Y.: Super-resolution through neighbor embedding. In: CVPR. (2004)
37. Glasner, D., Bagon, S., Irani, M.: Super-resolution from a single image. In: ICCV. (2009)
38. Yang, J., Lin, Z., Cohen, S.: Fast image super-resolution based on in-place example regression. In: CVPR. (2013)
39. Sun, J., Zheng, N.N., Tao, H., Shum, H.Y.: Image hallucination with primal sketch priors. In: CVPR. (2003)
40. Ni, K.S., Nguyen, T.Q.: Image superresolution using support vector regression. Image Processing, IEEE Transactions on **16**(6) (2007) 1596–1610
41. He, L., Qi, H., Zaretzki, R.: Beta process joint dictionary learning for coupled feature spaces with application to single image super-resolution. In: CVPR. (2013)
42. Yang, J., Wright, J., Huang, T., Ma, Y.: Image super-resolution as sparse representation of raw image patches. In: CVPR. (2008)
43. Yang, J., Wright, J., Huang, T.S., Ma, Y.: Image super-resolution via sparse representation. Image Processing, IEEE Transactions on **19**(11) (2010) 2861–2873
44. Timofte, R., De Smet, V., Van Gool, L.: A+: Adjusted anchored neighborhood regression for fast super-resolution. In: ACCV. (2014)
45. Schulter, S., Leistner, C., Bischof, H.: Fast and accurate image upscaling with super-resolution forests. In: CVPR. (2015)
46. Huang, J.B., Singh, A., Ahuja, N.: Single image super-resolution from transformed self-exemplars. In: CVPR. (2015)
47. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. In: ICLR. (2016)
48. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. (2016)

49. Gross, S., Wilber, M.: Training and investigating residual nets. http://torch.ch/blog/2016/02/04/resnets.html (2016)
50. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML. (2015)
51. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR. (2015)
52. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) **115**(3) (2015) 211–252
53. Aly, H.A., Dubois, E.: Image up-sampling using total-variation regularization with a new observation model. Image Processing, IEEE Transactions on **14**(10) (2005) 1647–1659
54. Zhang, H., Yang, J., Zhang, Y., Huang, T.S.: Non-local kernel regression for image and video restoration. In: ECCV. (2010)
55. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: ECCV. (2014)
56. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. In: ICLR. (2015)
57. Collobert, R., Kavukcuoglu, K., Farabet, C.: Torch7: A matlab-like environment for machine learning. In: NIPS BigLearn Workshop. (2011)
58. Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., Shelhamer, E.: cuDNN: Efficient primitives for deep learning. arXiv preprint arXiv:1410.0759 (2014)
59. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. Image Processing, IEEE Transactions on **13**(4) (2004) 600–612
60. Hanhart, P., Korshunov, P., Ebrahimi, T.: Benchmarking of quality metrics on ultra-high definition video sequences. In: Digital Signal Processing (DSP), 2013 18th International Conference on, IEEE (2013) 1–8
61. Huynh-Thu, Q., Ghanbari, M.: Scope of validity of psnr in image/video quality assessment. Electronics letters **44**(13) (2008) 800–801
62. Kundu, D., Evans, B.L.: Full-reference visual quality assessment for synthetic images: A subjective study. Proc. IEEE Int. Conf. on Image Processing (2015)
63. Zhang, L., Zhang, L., Mou, X., Zhang, D.: Fsim: a feature similarity index for image quality assessment. IEEE transactions on Image Processing **20**(8) (2011) 2378–2386
64. Sheikh, H.R., Bovik, A.C.: Image information and visual quality. IEEE Transactions on Image Processing **15**(2) (2006) 430–444
65. Bevilacqua, M., Roumy, A., Guillemot, C., Alberi-Morel, M.L.: Low-complexity single-image super-resolution based on nonnegative neighbor embedding. (2012)
66. Zeyde, R., Elad, M., Protter, M.: On single image scale-up using sparse-representations. In: Curves and Surfaces. Springer (2010) 711–730