Deep learning style transfer

Andrea Battistello, Samuele Conti, Fabio Chiusano

March 2018

Contents

1 Neural Style Transfer: A Review

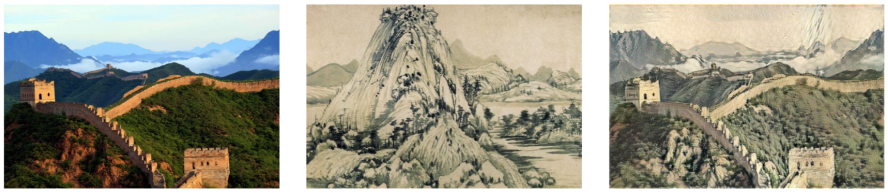
1.1 Abstract

Neural Style Transfer is the process of using CNN to migrate the semantic content of one image to different styles. This review aims to provide an overview of the current progress towards Neural Style Transfer, as well as discussing its various applications and open problems for future research.

1.2 Introduction

Recently, inspired by the power of Convolutional Neural Network (CNN), Gatys et al. first studied how to use CNN to reproduce famous painting styles on natural images. They obtained the image representations derived from CNN and found that the representations of image content and style were separable. Based on this finding, Gatys et al. proposed a Neural Style Transfer algorithm to recombine the content of a given photograph and the style of well-known artworks.

$$Image = Content + Style$$



(a) Content (b) Style (c) Content + Style

Figure 1: Image as content + style.

The key idea behind this algorithm is to start from random noise as the initial result and then change the values of pixels iteratively until the desired statistical feature distribution is satisfied.

1.3 A Taxonomy of Neural Style Transfer Methods

- **Descriptive Neural Methods Based On Image Iteration**: transfers the style by directly updating pixels in the image iteratively. The objective of image iteration is to minimize the total loss such that the stylized image simultaneously matches the content representation of the content image and the style representation of the style image. One of the keys to Neural Style Transfer is the representation of style, i.e., the pre-defined style loss function. The style loss function is optimized to match the feature statistics of the style image. Depending on the different adopted

style loss functions, Descriptive Neural Methods can be further divided into:

– **MMD-based Descriptive Neural Methods (Maximum Mean Discrepancy)**: Maximum Mean Discrepancy (MMD) is a popular metric of discrepancy between two distributions, based on the mean of features in the Hilbert space. Style transfer can be considered as a *distribution alignment process* from the content image to the style image. Therefore MMD can be used to measure the style discrepancy. MMD-based Descriptive Neural Methods refer to the neural methods that use MMD with different kernels/filters as the style loss for optimization. The algorithm of Gatys et al. is the first proposed MMD-based descriptive method. Given a content image x_c and a style image x_s, the algorithm of Gatys tries to find a stylized image x that minimizes the objective:

$$x = \arg\min_x L_{total} = \arg\min_x (\alpha L_{content}(x, x_c) + \beta L_{style}(x, x_s))$$

where:

* $L_{content}(x, x_c)$: compares content image representation of some layer l to that of the (yet unknown) stylized image.

$$L_{content}(x, x_c) = \frac{1}{2} \sum_{i,j} ||\Phi^l_{i,j}(x) - \Phi^l_{i,j}(x_c)||^2$$

where $\Phi^l_{i,j}$ is the representation of the i-th filter at position j in layer l.

* $L_{style}(x, x_c)$: compares the entries of the Gram matrices from the style image to that of the (yet unknown) stylized image.

$$L_{style}(x, x_s) = \sum_{l=0}^{L} w_l L_l$$

where L is the number of layers and w_l are weighting factors of each layer, which can be tuned manually.

$$L_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G^l_{i,j} - A^l_{i,j})^2$$

where N_l is the number of filters in layer l, M_l is the size of the feature map in layer l (i.e. height times width), $G^l_{i,j}$ is the entry in the Gram matrix of x_s in layer l corresponding to filters i and j, $A^l_{i,j}$ is the same for the image x.

$$G^l_{i,j} = \sum_k \Phi_{i,k} \Phi_{k,j}$$

We use the Gram matrix to obtain correlations between filter responses. The feature space built by these correlations is considered as a representation of the style.

  * α and β: are weighting factor for content and style reconstruction.

  This can be minimized by gradient descent with backpropagation to generate the final stylized result.

- **MRF-based Descriptive Neural Methods (Markov Random Field)**: considers the Neural Style Transfer at a local level, i.e., operating on patches to match the style. When calculating the style loss, the algorithm of Gatys et al. captures only the per-pixel feature correlations and does not constrain the spatial layout, which leads to the less visual plausibility results for photorealistic styles. So, this technique works well with photorealistic styles. We can introduce a new style loss function L_{style} which includes a patch-based MRF prior as follows:

$$L_{style}(x, x_s) = \sum_{i=1}^{m} ||\Psi_i(\Phi(x)) - \Psi_{NN(i)}(\Phi(x_s))||^2$$

  where m is the number of the local patches in stylized image x, Ψ_i denotes the i-th local patch and $\Psi_{NN(i)}$ denotes the most similar style patch in the style image x_s with the i-th local patch in the current stylized image x. The best matching $\Psi_{NN(i)}$ is calculated using normalized cross-correlation over all local style patches in x_s. It's a form of content-aware neural style transfer → it's good with photorealistic styles.

Considerations and further work:

- instabilities in the Gram matrices → add histogram losses to the objective function.
- parameter tuned manually → automatically tuned so as to prevent extreme values for gradients.
- matching the Gram matrices is equivalent to minimizing Maximum Mean Discrepancy (MMD) with the second order polynomial kernel.
- not content-aware.
- style representation in Gatys is invariant to the spatial configuration of the style image → propose a new style representation called "spatial style", which captures less style details and more spatial configurations.
- the algorithm proposed by Gatys et al. shows great results in transferring repetitive styles but often fails to transfer complex patterns → variety of small helpful modifications.

- how to control space in style transfer: by a guidance map where the desired region (getting the style) is assigned 1 and otherwise, 0.
- how to control color in style transfer...
- how to control scale in style transfer...

A problem of these algorithms is that they are slow.

- **Generative Neural Methods Based On Model Iteration**: aka Fast Neural Style transfer, since it addresses the speed and computational cost issue at the expense of losing some flexibilities.. First optimizes a generative model iteratively and produces the styled image through a single forward pass. The key idea is to train a feed-forward network over a large set of images in advance for each specific style image. The network model is optimized by updating the model iteratively using gradient descent. One of the limitations is that each generative network is tied to a single style and separate generative networks have to be trained for each specific style image.

  Considerations and further work:

  - perceptual loss function.
  - training a Markovian feed-forward network using adversarial training.
  - many style images may share some computations and it is redundant to train a separate feed- forward network for each of them → train a conditional style transfer network for multiple styles of the same type based on conditional instance normalization.
  - swap batch normalization with instance normalization, which removes instance-specific contrast information from the content image.
  - Depth-preserving Neural Style Transfer algorithm.

1.4 Extensions to Specific Types of Images

There are many studies aiming to extend state-of-the-art Neural Style Transfer algorithms to these specific types of images as well as single user-specified object stylization:

- doodles
- head portraits
- single user-specified objects
- video frames

1.5 Evaluation Methodology

There is no ground truth for the problem of Neural Style Transfer. Neural Style Transfer is the creation of art. For the same stylized result, it is common that different people have different or even opposite views. Therefore, the evaluation of visual results produced by Neural Style Transfer algorithms remains an open and important problem. There are two major types of evaluation methodologies:

- qualitative evaluation: requires participants to rank the results of different algorithms, and relies on the observation of participants (referred to as "Stylization Perceptual Studies").
- quantitative evaluation: evaluation focuses on the precise evaluation index (e.g., generating time and training time) of algorithms.

1.6 Applications

- Social communications
- User-assisted creation tools
- Production Tools for Entertainment Applications

1.7 Challenges and possible solutions

- parameter tuning for each combination of content and style images $\rightarrow$ combine some non-gradient information such as magnitude of the losses and statistics within the losses for Descriptive Neural Methods. In general, we want to break the three-way trade-off between speed, flexibility and quality.
- stroke orientation control: existing Neural Style Transfer algorithms do not consider the brush stroke orientation control in many oil painting styles $\rightarrow$ some ideas in the NPR (Non-photorealistic Rendering) field can be borrowed to solve the orientation problem of Neural Style Transfer.
- fast neural style transfer means less flexibility at the moment $\rightarrow$ boh!

Current Achievements	Paper	Description
Original Neural Style Transfer as well as its slight modifications	[14, 16]	The First Descriptive Neural Style Transfer algorithm proposed by Gatys *et al*.
	[34, 35]	Slight Modifications by varying experimental settings, *etc*.
Theoretical Explanation of Neural Style Transfer	[30]	Treat Neural Style Transfer as a Domain Adaption problem.
Solution of instabilities during iterations	[38]	Combine [14]'s style loss with a proposed histogram loss.
Automatic parameters tuning	[38]	Exploit gradient information to automatically tune the parameters.
Combining semantic information	[48]	Region Segmentation based Content-aware Neural Style Transfer.
	[10]	Masking Out based Content-aware Neural Style Transfer.
	[28]	Patch-based Markov Random Fields style loss.
	[7]	Combine [28] with the semantic map.
Preserving content image's color	[13, 17]	Color-preserving Neural Style Transfer through color transformation or operating in the luminance channel.
Brush size control in the stylized image	[17]	Coarse-to-fine procedure with down-sampling and up-sampling.
Preserving content image's depth information	[31]	Depth-preserving Neural Style Transfer through introducing an extra depth loss function.
Speeding up time-consuming Descriptive Neural Style Transfer	[24, 44, 29]	"Fast" Generative Neural Style Transfer algorithm through training a style-specific feed-forward network for every style.
	[45, 46]	Improve Generative Neural Methods through swapping Batch Normalization with Instance Normalization and learning generators that uniformly sample the Julesz ensemble.
	[12]	"Faster" Neural Style Transfer through training a conditional network for several styles.
	[9]	"Faster" Neural Style Transfer through training an inverse network for any style.
Turning sketches into artwork	[7]	Neural Doodle algorithm based on Markov Random Fields.
User-specific object stylization	[5]	Targeted Style Transfer through combining Semantic Segmentation algorithm.
Video frames stylization	[40, 3]	Neural Video Style Transfer through enforcing consistency between adjacent frames.
Head portrait stylization	[41]	Head Portrait Style Transfer through exploiting the notion of gain maps.

Figure 2: Summary of Current Achievements in the Field of Neural Style Transfer.

2 Image Style Transfer Using CNN

2.1 Abstract

- How to separate image content from style $\longrightarrow$ use a CNN

2.2 Introduction

- Transfer style can be seen as Texture Trasfer: synthesise a texture constrained to preserve semantic content of a target image.
- There exist many texture transfer algorithms. **Problem**: they only use low-level image features. They don't have a suitable image representation.
- **Idea**: use a CNN to extract features and have an image representation.

2.3 Deep Image Representation

- Use a pre-trained neural network: VGG has 16 convolutional layers and 5 pool.
- **Normalization** of the layers such that the mean activation of each convolutional filter over images and positions is equal to 1.
  This normalization does not affect result because the network uses rectifying units.
- Replaced max pooling layers with average pooling layers.

2.3.1 Content representation

Each convolutional layer gives a different non-linear representation of the image.
Given:

- N_l: Number of filters in convolutional layer l
- M_l: Total size of the filters $M_l = height_l \cdot width_l$

Then each convolutional layer l can be represented with a matrix $F^l \in \mathcal{R}^{N_l \times M_l}$ where F^l_{ij} corresponds to the activation of the i^{th} filter at position j of filter l.

To find the corresponding image $\vec{x}$ given the feature matrix at layer l, it is possible to perform gradient descent from white noise image. Let $\vec{p}$ be the original image, P^l be the original feature matrix and F^l the feature matrix generated by $\vec{x}$. Then we can minimize:

$$\mathcal{L}_{\text{content}}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F^l_{ij} - P^l_{ij}\right)^2$$

The derivative with respect to $\vec{x}$ of $\mathcal{L}_{\text{content}}(\vec{p}, \vec{x}, l)$ is:

$$\frac{\partial \mathcal{L}_{\text{content}}}{\partial F^l_{ij}} = \begin{cases} \left(F^l - P^l\right)_{ij} & \text{if } F^l_{ij} > 0 \\ 0 & \text{if } F^l_{ij} < 0 \end{cases}$$

Different layers represent the image in a different level:

- Higher layers in the network capture the high-level content in terms of objects, but does not constrain the exact pixel values
- Lower layers, instead, reproduce the exact pixel values of the original image

2.3.2 Style representation

Use the correlation between the filter responses to encode the style of an image. These feature correlations are given by the Gram matrix:

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l$$

This represent a stationary, multi-scale representation of the input image which captures the texture information but not the global arrangement.

Again we can find the image $\vec{x}$ and the corresponding Gram matrix G^l at layer l by minimizing:

$$\mathcal{L}_{\text{style}} = \sum_l w_l E_l$$

Where E_l is the contribution on the style of each layer l of the neural network:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2$$

2.4 Style transfer

Finally, to transfer the style of an image $\vec{a}$ onto a content image $\vec{p}$ we minimize the total loss given by:

$$\mathcal{L}_{\text{total}}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{\text{content}}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{\text{style}}(\vec{a}, \vec{x})$$

- Used L-BFGS to optimize $\frac{\partial \mathcal{L}_{\text{total}}}{\partial \vec{x}}$
- No regularization

2.5 Results

- Results obtained by minimizing content loss on layer **conv_4_2** and using layers **conv_1_1**, **conv_2_1**, **conv_3_1**, **conv_4_1**, **conv_5_1** with weight $w_l = \frac{1}{5}$ for style.
- ratio $\dfrac{\alpha}{\beta}$ was either 1×10^{-3}, 8×10^{-4}, 5×10^{-3} or 5×10^{-4}

3 Visual Attribute Transfer through Deep Image Analogy

3.1 Abstract

- Visual attribute transfer: transfer of visual information (such as color, tone, texture, and style) from one image to another.
- Deep image analogy: technique that adapts the notion of "image analogy" with features extracted from a Deep CNN for matching. It's able to find semantically-meaningful dense correspondences between two input images.

3.2 Introduction

The applications of color transfer, texture transfer, and style transfer share a common theme, which we characterize as *visual attribute transfer*. In this paper, we describe a new technique for visual attribute transfer for a pair of images that may be very different in appearance but semantically similar. Our technique establishes semantically-meaningful dense correspondences between the input images, which allow effective visual attribute transfer. An image analogy is codified as:

$$A : A' :: B : B'$$

where B' relates to B in the same way as A' relates to A and additionally, A and A' (also B and B') are in pixel-wise correspondences. For our scenario only a source image A and an example image B' are given, and both A' and B represent latent images to be estimated, imposing a bidirectional constraint to better match A to B'. Solving the visual attribute transfer problem is equivalent to finding both unknown images A' and B. Instead of applying image analogy to image pixels directly, we use a Deep CNN (e.g. the pre-trained VGG-19) to construct a feature space in which to form image analogies.

Considerations:

- CNN representations gradually encode image information from low-level details to high-level semantic content. This provides a good decomposition of semantic structure and visual attributes for transfer.
- The spatial correspondence between intermediate feature layers in CNN architectures is approximately maintained.

Both properties facilitate a coarse-to-fine strategy for computing the nearest-neighbor field (NNF), a core component used in reconstructing images. To speed up the required nearest-neighbor search, we adapt PatchMatch to the CNN feature domain. Our method uses the bidirectional constraint in the patch similarity metric.

This technique is not effective for images which are not semantically correlated.

3.3 Related work

- *Visual attribute transfer*: most of these approaches are, however, not general, in that they are designed to transfer a specific type of visual attribute.
  - *Color transfer*: tend to be global. They work well when the source and reference images are of similar scenes, even though the spatial layouts can be dissimilar. Local color transfer methods infer local color statistics in different color regions by establishing region correspondences (e.g. across the same semantic regions).
  - *Texture transfer*: e.g. introduce a correspondence map that includes features of the target image such as image intensity to constrain the texture synthesis procedure.
  - *Style transfer*: e.g. view style transfer as the composition of local texture transfer and global color transfer, and suggest a simple yet efficient adaptive image partition for the decomposition of style and content.
  - *Image analogy*: supervised manner, where a pair of images A and A' are manually registered, and the analog of image B (similar in style with A) is to be found (resulting in B').
- *Dense correspondence*: PatchMatch is a fast randomized algorithm for finding a dense NNF for patches.
- *Neural style transfer*: our technique transfers style in a structure-preserving manner, and this is due to semantic-based dense correspondences.

3.4 Important ideas

Given an image pair A and B', which may differ in appearance but have similar semantic structure, the goal is to find the mapping from A to B' (or from B' to A) for visual attribute transfer. It is assumed that the image pair has different visual attributes.

1. Analogy with bidirectional constraint: our solution is to formulate the mapping as a problem of image analogies $A : A' :: B : B'$, where A' and B are unknown latent variables. This analogy implies two constraints:
   (a) A and A' (also B and B') correspond at the same spatial position;
   (b) A and B (also A' and B') are similar in appearance (color, lighting, texture and etc.).

   We call:
   - $\Phi_{a\to b}$: mapping of pixels from A to B';
   - $\Phi_{b\to a}$: mapping of pixels from B' to A.

   Because of the in-place mappings $A \to A'$ and $B \to B'$, we also have that:

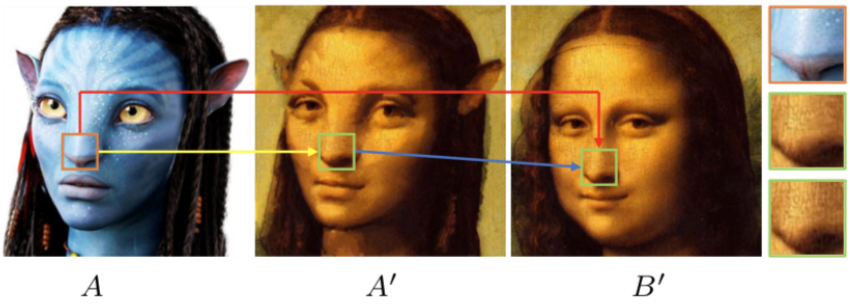


Figure 3: Our method separates the difficult mapping $A \to B'$ (red) into two tractable mappings: in-place mapping $A \to A'$ (yellow) and similar-appearance mapping $A' \to B'$ (blue)

- $\Phi_{a \to b}$: mapping of pixels from A to B, from A' to B and from A' to B';
- $\Phi_{b \to a}$: symmetrically the same.

We find the mappings by imposing:

$$A(p) = B(\Phi_{a \to b}(p)) \text{ and } A'(p) = B'(\Phi_{a \to b}(p))$$

and the bidirectional constraint

$$\Phi_{b \to a}(\Phi_{a \to b}(p)) = p \text{ and } \Phi_{a \to b}(\Phi_{b \to a}(p)) = p$$

where p is a pixel.

2. Reconstruction using CNN: The ideal A' should comprise the content structure from A and corresponding visual details from B'. We solve this problem using an image decomposition and reconstruction hierarchy. We adopt recent CNNs trained on an object recognition dataset, which compresses image information progressively from precise appearance (fine) to actual content (coarse). At the coarsest layer, A and B' may have very similar content information for better matching as indicated by yellow rectangles. As a result, we can assume A' to be A at this layer of the CNN. At other layers of the CNN, we selectively extract content structures from features of A and detail information from features of B' by a weighted mask, which helps construct a linearly weighted combination to recover features of A'. The updated latent images will carry the fusion information to the next layer for further mapping refinement.

3. Deep PatchMatch: PatchMatch is known as a randomized fast algorithm for computing approximate NNF between two images. Given latent images A' and B, inferring $\Phi_{a \to b}$ and $\Phi_{b \to a}$ is equivalent to computing a forward NNF and a reverse NNF between A and B as well as between A' and B'.

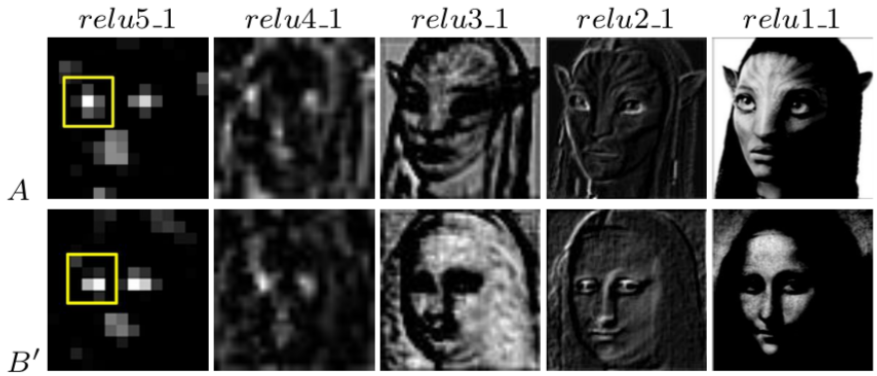


Figure 4: The input image A (or B) is encoded by the CNN (e.g. VGG-19) as a set of filtered images at each layer. Here we visualize one representative filtered image for each layer.

3.5 Deep Image Analogy

Deep Image Analogy is the process of achieving visual attribute transfer through image analogy and deep CNN features. It follows this pipeline:

1. **Preprocessing**: compute deep features for the input image pair A/B' through the pre-trained CNN. Initialize feature maps of two latent images A'/B at the coarsest CNN layer.
2. for each layer l, do the following:
   (a) **Nearest-neighbor Field Search**: compute a forward NNF (i.e. $\Phi^l_{a\to b}$) and a reverse NNF (i.e. $\Phi^l_{b\to a}$) that establish correspondences between feature maps of A and B as well as between feature maps of A' and B'.
   (b) **Latent image reconstruction**: use the NNFs and the current feature maps to reconstruct the feature maps of the next layer.
   (c) **Nearest-neighbor Field Unsampling**: unsample the NNFs to the next layer as their initializations.
3. **Output production**

Let's enter into more details for each phase:

- **Preprocessing**: we call $\{F^L_A\}$ and $\{F^L_{B'}\}$ the feature maps for $L = 1, ..., 5$. The feature map of each layer is extracted from the corresponding ReLu layer. It is a 3D tensor with size $width * heights * channel$, and its spatial resolution increases from $L = 5$ to 1. We initialize $F^5_{A'} = F^5_A$ and $F^5_B = F^5_{B'}$

- **Nearest-neighbor Field Search**: $\Phi^l_{a\to b}$ maps a point from feature map F^l_A to $F^l_{B'}$. It's computed minimizing the following energy function:

$$\Phi^l_{a\to b}(p) = \arg\min_q \sum_{x\in N(p), y\in N(q)} (||\bar{F}^l_A(x) - \bar{F}^l_B(y)||^2 + ||\bar{F}^l_{A'}(x) - \bar{F}^l_{B'}(y)||^2)$$

  where q is a pixel, $N(p)$ is the patch around p, $\bar{F}^l(x) = \frac{F^l(x)}{|F^l(x)|}$. $\Phi^l_{a\to b}$ and $\Phi^l_{b\to a}$ usually will agree. Such a unary-only energy formulation can be efficiently optimized with the PatchMatch method.

- **Latent image reconstruction**:

$$F^{L-1}_{A'} = F^{L-1}_A \circ W^{L-1}_A + R^{L-1}_{B'} \circ (1 - W^{L-1}_A)$$

  where:

  - $\circ$ is the element-wise product.
  - W^{L-1}_A is a 2D weight map obtained in this way:

$$W^{L-1}_A = \alpha_{L-1} M^{L-1}_A$$

    where:

    * M^{L-1}_A is a function that specifies the magnitudes of neuron responses at layer L-1, in order to preserve content structures from A when they are present:

$$M^{L-1}_A(x) = \frac{1}{1 + e^{-k|F^{L-1}_A(x)|^2 - \tau}}$$

      where $k = 300$ and $\tau = 0.05$.
    * $\{\alpha_L\} = \{0.8, 0.7, 0.6, 0.1\}$ control the trade-off between content and attribute such as style.
  - $R^{L-1}_{B'}$ is the modified version of $F^{L-1}_{B'}$ to fit the structure of A. Ideally, $R^{L-1}_{B'} = F^{L-1}_{B'}(\Phi^{L-1}_{a\to b})$, but $\Phi^{L-1}_{a\to b}$ is not known at layer $L-1$. So, we initialize $R^{L-1}_{B'}$ with random noise and we find it by warping at layer L and then deconvolving in the CNN, i.e. minimizing this:

$$L_{R^{L-1}_{B'}} = ||CNN^L_{L-1}(R^L_{B'}) - F^L_{B'}(\Phi^L_{a\to b})||^2$$

- **Nearest-neighbor Field Unsampling**: It serves only as initial guess, as it is further refined later with the PatchMatch algorithm.

- **Output production**:

$$A'(p) = \frac{1}{n} \sum_{x\in N(p)} B'(\Phi^1_{a\to b}(x))$$

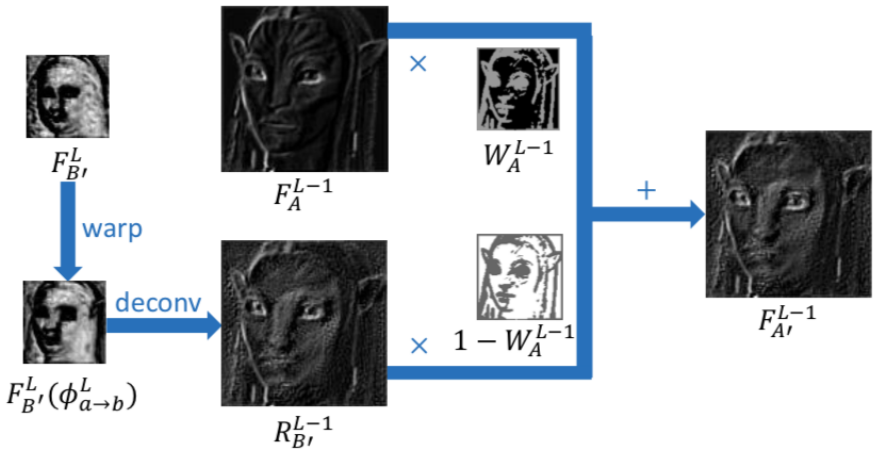


Figure 5: Recovering features of latent image A' by weighted combination of the structures from A and the visual details sampled from B'

4 Image-to-Image Translation with Conditional Adversarial Networks

4.1 Abstract

The paper investigate conditional adversarial networks as a general-purpose solution to image-to-image translation problems: these networks are not only able to learn a mapping between input and output images, but also a loss function to train this mapping. This approach, due to its generality, can be applied in the same way to many different tasks that would required very different loss formulations, avoiding the need of hand-engineering both the mapping and the loss function for the task at hand.

4.2 Introduction

Many problems in image processing can be posed as "translating" an input image into a corresponding output image; image-to-image translation can be defined as the problem of translating one possible representation of a scene into another, given sufficient training data. While convolutional neural nets (CNNs) have become the common workhorse behind a wide variety of image prediction problems, and even if the learning process is automatic, still a lot of effort goes into designing effective loss functions. In general, Generative Adversarial Networks (GANs) learn a loss that tries to classify if the output image is real or fake, while simultaneously training a generative model to minimize this loss. The paper explores GANs in a conditional setting: while GANs learn a generative model of data, conditional GANs (cGANs) learn a conditional generative model: in case of image-to-image translation tasks, we condition on an input image and generate a corresponding output image. The primary contribution of the paper is to demonstrate that on a wide variety of problems conditional GANs produce reasonable results.

4.3 Related Work

While image-to-image translation problems are often formulated as per-pixel classification or regression, considering each pixel conditionally independent from the others, conditional GANs learn a structured loss able to penalize the joint configuration of the output. Even if a body of literature already exists for losses of this kind, conditional GANs are different in that the loss is learned. This paper is not first in applying conditional GANs: for image-to-image mapping, though, several other papers have used GANs unconditionally, relying on other terms to force the output to be conditioned on the input. Again, each method was tailored for a specific task, while the approach proposed by this paper is more general and, in principle, simpler to apply. In addition, differently with respect to past works, for the generator a "U-Net"-based architecture is used, and for the discriminator a convolutional "PatchGAN" classifier is used, which only penalizes structure at the scale of image patches.

4.4 Method

GANs are generative models that learn a mapping from random noise vector z to output image y, $G : z \rightarrow y$. In contrast, conditional GANs learn a mapping from observed image x and random noise vector z, to y, $G : \{x, z\} \rightarrow y$. The generator G is trained to produce outputs that cannot be distinguished from "real" images by an adversarially trained discriminator, D, which is trained to do as well as possible at detecting the generator's "fakes".

4.4.1 Objective

The objective of a conditional GAN can be expressed as

$$L_{CGAN}(G, D) = \mathbf{E}_{x,y}[logD(x, y)] + \mathbf{E}_{x,z}[log(1 - D(x, G(x, z))]$$

where G tries to minimize this objective against an adversarial D that tries to maximize it. In this approach both the generator G and the discriminator D are conditioned by the input image x. The paper also considers a variant in which D is not conditioned by the input image x:

$$L_{CGAN}(G, D) = \mathbf{E}_{x}[logD(y)] + \mathbf{E}_{x,z}[log(1 - D(x, G(x, z))]$$

Finally the paper considers an approach in which the generator G is tasked to not only fool the discriminator but also to be near the ground truth output in an L1 sense, adding to the loss the following term:

$$L_{L1}(G) = \mathbf{E}_{x,y,z}[||y - G(x, y)||_1]$$

.

With this addition, the objective of the network is defined by:

$$G^* = \arg \min_G \max_D L_{CGAN}(G, D) + \lambda * L_{L1}(G)$$

The introduction of the noise z is needed in order to avoid deterministic results: the paper, however, emphasizes that inserting a Gaussian noise as an input to the generator G is not an effective procedure, since G would simply learn to ignore it and not exploit it. The proposed model introduces the noise z in form of dropout, applied on several layers of G at both training and test time, thus providing some stochasticity (even if small) to the output of the net.

4.4.2 Network architectures

Both generator and discriminator use modules of the form convolution-BatchNorm-ReLu (BatchNorm as Batch Normalization).

4.4.3 Generator with skips

Image-to-image translation problems define a mapping between a high resolution input grid and an output resolution grid, often considering in both the input and the output the same underlying structure. Many previous solutions to problems in this area have used an encoder-decoder network, where the input is passed through a series of layers that progressively downsample, until a bottleneck layer, at which point the process is reversed. To give the generator a means to circumvent the bottleneck for low-level information and shuttle it directly through the network, skip connections are added. Specifically, there are skip connections between each layer i and layer $n - i$, where n is the total number of layers. Each skip connection simply concatenates all channels at layer i with those at layer $n - i$.

4.4.4 Markovian Discriminator (PatchGAN)

Figure 3: Different losses induce different quality of results. Each column shows results trained under a different loss. Please see `https://phillipi.github.io/pix2pix/` for additional examples.

As shown by the figure above, both L2 and L1 loss produce blurry results on image generation problems, usually failing to encourage high-frequency crispness. Typically the L1 loss is sufficient to guarantee low-frequency correctness, so the GAN discriminator is restricted to only model high-frequency structure, focusing only on the structure of local patches of the image. The discriminator architecture, called "PatchGAN", only penalizes structure at the scale of patches. In particular, the discriminator D tries to classify if each $N \times N$ patch in an image is real or fake: the discriminator is convolutationally run across the image, averaging all responses to provide the ultimate output of D. the paper also demonstrates that N can be a small number and still provide great results, easing the computational effort. Such a discriminator effectively models the image as a Markov random field, assuming independence between pixels separated by more than a patch diameter: since this a common assumption in model of texture and style, PatchGAN can therefore be understood as a form of texture/style loss.

4.4.5 Optimization and inference

To optimize the network a minibatch SGD is used, exploiting the Adam solver: in particular, one gradient descent step on D is alternated to one gradient descent

step on G. At inference time, the generator net is run in exactly the same manner as during the training phase. This differs from the usual protocol in that dropout is applied at test time, and batch normalization is applied using the statistics of the test batch, rather than aggregated statistics of the training batch. This kind of batch normalization, called "instance normalization" since the batch set has dimension 1, has been demonstrated to be effective at image generation tasks.

4.5 Experiments

To explore the generality of GANs, the proposed method is tested on a variety of tasks and datasets, including both graphics and vision tasks.

4.5.1 Evaluation metrics

Evaluating the quality of synthesized images is an open and difficult problem: in order to more holistically evaluate the visual quality of the results, the paper employs two tactics. First "real vs fake" perceptual studies on Amazon Mechanical Turk are run, since often the plausibility to a human observer is the ultimate goal; second, in case of tasks where the network has to generate a realistic scenario, an off-the-shelf system is used to test if is capable to perform object recognition in the synthesized image: this metric is based on the intuition that if the generated images are realistic, classifiers trained on real images will be able to classify the synthesized image correctly as well. The paper uses the popular FCN-8s architecture for semantic segmentation.

4.5.2 Analysis of the objective function

The paper analyzes the different components of the proposed objective function, checking the effects of the singles components.

$$L_{CGAN}(G,D) = \underbrace{\mathbf{E}_{x,y}[logD(x,y)] + \mathbf{E}_{x,z}[log(1-D(x,G(x,z))]}_{CGan_Loss} + \lambda * \underbrace{\mathbf{E}_{x,y,z}[||y-G(x,y)||_1]}_{L1_Loss}$$

The L1 loss alone leads to reasonable but blurry results, while the cGAN loss alone (setting $\lambda = 0$ in the equation) gives much sharper results, but introduces visual artifacts on certain applications. Adding both terms together (with $\lambda = 100$) reduces these artifacts.

When considering the discriminator without the conditioning, the generator collapsed into producing nearly the exact same output regardless of input photograph, obtaining poor performances.

The paper also considers the effect of the two terms on the colourfulness of the image: while L1 encourages average, grayish colors, the cGAN, on the other hand, pushes the output distribution closer to the ground truth.

Figure 3: Different losses induce different quality of results. Each column shows results trained under a different loss. Please see `https://phillipi.github.io/pix2pix/` for additional examples.

4.5.3 Analysis of the Generator architecture

A U-Net architecture allows low-level information to shortcut across the network, and it's evaluated if the obtained results are actually better than a non-U-Net architecture.

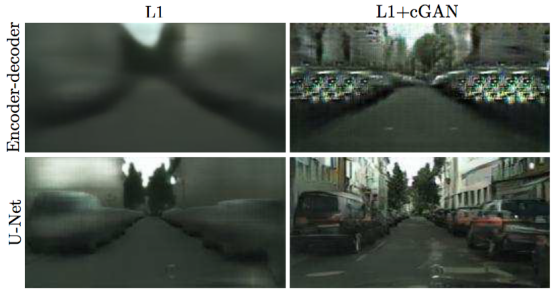


Figure 4: Adding skip connections to an encoder-decoder to create a "U-Net" results in much higher quality results.

Comparing the U-net against an encoder-decoder architecture, the encoder-decoder is unable to learn to generate realistic images in the experiments described in the paper. It also suggests that the advantages of U-Net may not be necessarily related to the specific case of cGANs.

4.6 From PixelGANs to PatchGans to ImageGANs

The paper tests the effect of varying the patch size N of the discriminator D receptive fields, from a 1×1 "PixelGAN" to a full 286×286 "ImageGAN". The PixelGAN has no effect on spatial sharpness, but does increase the colorfulness of the results; using a 16×16 PatchGAN is sufficient to promote sharp outputs, and achieves good FCN-scores, but also leads to tiling artifacts. The 70×70 PatchGAN alleviates these ar- tifacts and achieves similar scores. Scaling beyond this, to the full 286×286 ImageGAN, does not appear to improve the visual quality of the results, and in fact gets a consider- ably lower FCN-score.

An advantage of the PatchGAN is that a fixed-size patch discriminator can be applied to arbitrarily large images. The generator may also be applied convolutionally on larger images than those on which it was trained.

Figure 5: Patch size variations. Uncertainty in the output manifests itself differently for different loss functions. Uncertain regions become blurry and desaturated under L1. The 1x1 PixelGAN encourages greater color diversity but has no effect on spatial statistics. The 16x16 PatchGAN creates locally sharp results, but also leads to tiling artifacts beyond the scale it can observe. The 70×70 PatchGAN forces outputs that are sharp, even if incorrect, in both the spatial and spectral (colorfulness) dimensions. The full 286×286 ImageGAN produces results that are visually similar to the 70×70 PatchGAN, but somewhat lower quality according to our FCN-score metric (Table 2). Please see `https://phillipi.github.io/pix2pix/` for additional examples.

4.7 Perceptual validation

The authors validate the perceptual realism of their results on the tasks of map-aerial photograph and grayscale-color: as reported in the paper,results were significantly above the L1 baseline.

4.8 Semantic segmentation

Conditional GANs appear to be effective on problems where the output is highly detailed or photographic, as is common in image processing and graphics tasks: the paper test the proposed architecture on vision problems, like semantic segmentation, where the output is instead less complex than the input. Although cGANs achieve some success, they are far from the best available method for solving this problem: simply using L1 regression gets better scores than using a cGAN.

4.9 Community-driven research

The Twitter community, including computer vision and graphics practitioners as well as artists, have successfully applied the proposed framework to a variety of novel image-to-image translation tasks, far beyond the scope of this original paper.

4.10 Conclusion

The results in this paper suggest that conditional adversarial networks are a promising approach for many image- to-image translation tasks, especially those involving highly structured graphical outputs. These networks learn a loss adapted to the task and data at hand, which makes them applicable in a wide variety of settings.

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

5.1 Abstract

- Perceptual losses: we use both *per-pixel loss* and *perceptual loss*, i.e. a loss based on high-level features.
- Real time: three orders of magnitude faster than Gatys, with similar quality.
- Super-resolution: just an experiment.

5.2 Introduction

Some approaches are slow because they require solving slow optimization problems. 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.

5.3 Related work

- Feed-forward image transformation: usually per-pixel loss, CRF (Conditional Random Field) loss, gradient penalizing loss to enforce local consistency.
- Perceptual optimization: i.e. depending on high level features. E.g. maximizing class prediction scores or some individual features.
- Style Transfer: content & style.
- Image super-resolution: divided in pre-CNN and post-CNN.

5.4 Method

5.4.1 Overview

The system is made of two components:

1. the image transformation network f_W. It's a *deep residual CNN* parametrized by weights W. It transforms input image x into output image $\hat{y} = f_W(x)$.
2. the loss network ϕ, used to define several loss functions $l_1, ..., l_k$. It's pretrained and fixed. Each loss function computes a scalar value $l_i(\hat{y}, y_i)$ measuring the difference between the output image $\hat{y}$ and a target image y_i.

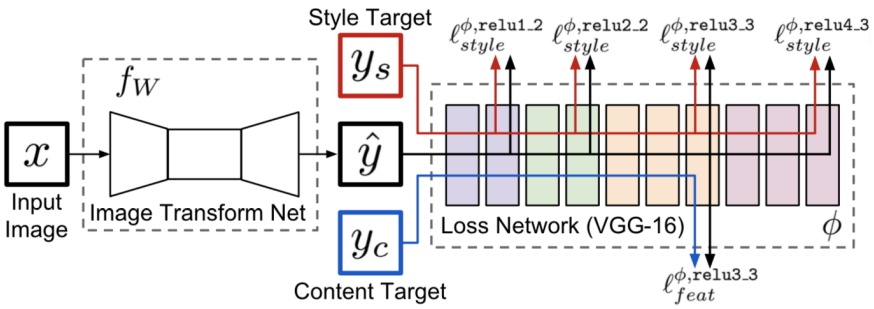


Figure 6: System overview. We train an *image transformation network* to transform input images into output images. We use a *loss network* pretrained for image classification to define *perceptual loss functions* that measure perceptual differences in content and style between images. The loss network remains fixed during the training process.

The image transformation network is trained with SGD in this way:

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

The loss network is used to define two losses:

- feature reconstruction loss l^{ϕ}_{feat}, with respect to a content target y_c.
- style reconstruction loss l^{ϕ}_{style}, with respect to a style target y_s.

For:

- style transfer: y_c is the input image x and y_s is the style target image.
- super resolution: x is the low resolution image, y_c is the ground-truth high resolution image and y_s is not used.

5.4.2 Image transformation network

- no pooling layers.
- strided and fractionally strided convolutions for in-network downsampling and upsampling.
- five residual blocks (residual connection = skip connection).
- all non-residual convolutional layers are followed by spatial batch normalization and ReLU non-linearities with the exception of the output layer, which instead uses a scaled tanh to ensure that the output image has pixels in the range.

- other than the first and last layers which use 9 * 9 kernels, all convolutional layers use 3 * 3 kernels
- we use different image transformation networks for style transfer and super resolution.

The paper says that the complete network architecture is in the supplementary materials...

5.4.3 Loss functions

The perceptual losses are:

- Feature reconstruction loss:

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

  where j is the considered convolutional layer and C_j, H_j, W_j are the channels, height and width of the j-th layer.

- style reconstruction loss:

$$l_{style}^{\phi,j}(\hat{y}, y) = ||G_j^{\phi}(\hat{y}) - G_j^{\phi}(y)||_F^2$$

  where j is the considered convolutional layer, F is the Frobenius norm and G_j^{ϕ} is the Gram matrix:

$$G_j^{\phi}(x)_{c,c'} = \frac{1}{C_j H_j W_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \phi_j(x)_{h,w,c} \phi_j(x)_{h,w,c'}$$

Other losses are:

- pixel loss:

$$l_{pixel}(\hat{y}, y) = \frac{1}{CHW} ||\hat{y} - y||_2^2$$

- total variation regularization: to encourage spatial smoothness in the output image $\hat{y}$:

$$l_{TV}(\hat{y}) = ?$$

The final loss function, using VGG-16 as loss network, is the following:

$$l^{\phi}(\hat{y}, y) = \sum_{j=1}^{4} \alpha_j l_{style}^{\phi,j}(\hat{y}, y) + \sum_{j=1}^{3} \beta_j l_{feat}^{\phi,j}(\hat{y}, y) + \gamma l_{pixel}(\hat{y}, y) + \delta l_{TV}(\hat{y})$$

l_{style} is not used in the case of super-resolution.

6 Deep Photo Style Transfer

6.1 Abstract

- **Problem**: Gatys approach is not suitable for photorealistic style transfer. When both inputs are photographs, output resembles a painting.
- **Solution**: Constraint the transformation to be locally affine in color space through a differentiable energy term.
- This approach also resolves the problem of matching the content of the two images by using semantic segmentation of the input and reference images.

6.2 Challenges and Contributions

Structure preservation We aim to have a transformation that changes the colors of an image with no geometric effects.

Our solution is restricting the transformation in colorspace.

Semantic accuracy and transfer faithfulness The transfer should respect the semantics of the scene, i.e. sky should be matched to sky, buildings to buildings and so on.

CNNMRF achieves this by matching each input neural patch with the most similar patch in the style image, to minimize the chance of inacurrate transfer. **Problem**: this matching lead to ignored regions or many regions with the same patch.

Gatys et al. uses the Gram matrix to define the "style distribution". **Problem**: only the reference image is checked for style, while there may be some 'style information' in the inpyt image. This can lead to texture mismatching (e.g. building texture matched with sky).

Solution: add semantic information so that style is transferred only between semantically equivalent regions.

6.3 Method

Introduced two new ideas:

- A photorealism regularization term, constraining the reconstructed image to be represented by locally affine color transformations of the input to prevent distortions.
- Optional guidance to the style transfer based on semantic segmentation of the inputs to avoid content-mismatch problem.

Photorealism regularization Assume input image is photorealistic. We don't want to lose this property. Hence we penalize distortions. Image transform is locally affine in color space, which means a function such that for each output patch, there is an affine function that maps the input RGB values onto their output counterparts. Each patch can have a different affine function, which allows for spatial variations.

7 Semantic Style Transfer and Turning Two-Bit Doodles into Fine Artwork

7.1 Abstract

Generative architectures based on CNNs for style transfer and image synthesis have proven highly effective, but they can sometimes lead to unpredictable results. This paper proposes to augment this type of architecture by introducing semantic annotation, in order to obtain more control over the final outcome and increase its quality.

7.2 Introduction

Thanks to the introduction of CNNs, many tasks in image processing have obtained astonishing results, including style transfer: users, however, must still select carefully the style images to avoid unexpected patterns in the final result. In general, even if CNNs are able to extract semantic information during classification tasks, this type of information is poorly exploited by generative algorithms: CNNs trained for classification, in fact, are not designed for correct synthesis and the lower layers used in generative architectures do not contain the most meaningful semantic information. To remedy this, the paper introduces an architecture that bridges the gap between generative algorithms and pixel labeling neural networks, along with the possibility of adapting already existing algorithms with the introduction of semantic annotations.

7.3 Related work

The paper analyzes different approaches used in style transfer:

- **Gram-based** style transfer, that uses the so called "Gram Matrices" to represent global statistics about the image, based on output from convolution layers.
- **Path-based** style transfer, which also operates over the output of convolutional layers, but patches are matched between the content and the style image through a nearest neighbor calculation.

Both gram-based and patch-based approach struggle to provide the user with an adequate control against possible glitches.

7.4 Model

The contribution of the paper builds on a patch-based approach, using optimization to minimize content reconstruction error Ec (weighted by α) and style remapping error Es (weight by β).

$$E = \alpha Ec + \beta Es$$

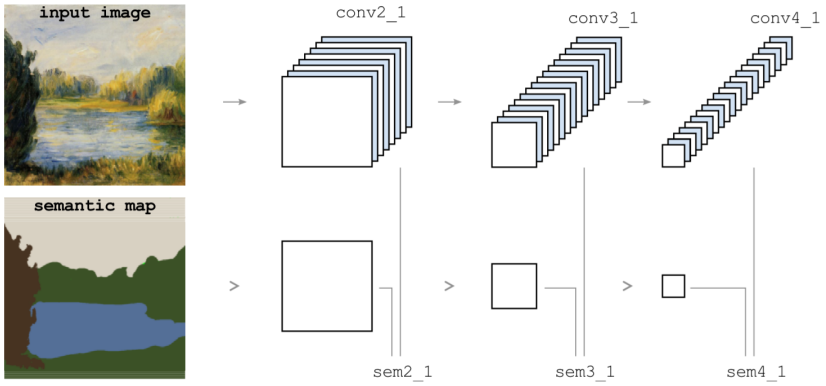


Figure 3: Our augmented CNN that uses regular filters of N channels (top), concatenated with a semantic map of M=1 channel (bottom) either output from another network capable of labeling pixels or as manual annotations.

7.4.1 Architecture

The model proposed is based on an augmentation of a VGG architecture, which combines pooling and convolution layers l with $3 * 3$ filters. Intermediate post-activation results x^l, consisting of N channels and capturing patterns from the images, are concatenated by a set of additional channels m^l of size M, all at the same resolution, computed by down-sampling a static semantic map specified as input. In addition, the new channels are weighted by a parameter γ to give more control to the user, obtaining a new output of N+M channels defined as:

$$s^l = x^l || \gamma m^l$$

The same process is applied to the style image, obtaining s_s^l

7.4.2 Representation

The input semantic map can contain an arbitrary number of channels M, but there are two main requirements:

- Each image needs to have its own semantic map of the same aspect ratio, even if the resolution can be lower.
- The number of channels for the semantic map can be arbitrary, but it must be consistent for the current style and content image.

It's relevant to notice that the semantic map can be produced as outputs by existing CNNs architectures.

7.4.3 Algorithm

Patches of $k * k$ are extracted from the semantic layers and denoted by the function ψ, respectively $\psi(s_s^l)$ for the input style patches and $\psi(s^l)$ for the current image patches. For any patch i in the current image and layer l, its nearest neighbor NN(i) is computed using normalized cross-correlation:

$$NN(i) = \arg\min_x \frac{\psi(s^l) * \psi(s_s^l)}{|\psi(s^l)| * |\psi(s_s^l)|}$$

The style error Es between all the patches i of layer l in the current image to the closest style patch is defined as the sum of the Euclidean distances:

$$E_s(s, s_s) = \sum_i ||\psi_i(s) - \psi_{NN(i)}(s_s)||^2$$

The information of the semantic map is used to compute the nearest-neighbour patches and contributes to the loss, but it's not part of the derivative of the loss. The activation x^l is compared to the corresponding style patches and optimize through the L-BFGS algorithm.

Through this method, existing patch-based implementations can follow this model to use additional semantic information without changes.

7.5 Experiments

The paper describes some experiments and the correlated results, using a VGG19 architecture with $3 * 3$ patches and semantic maps manually edited as RGB images.

7.5.1 Precise Control Via Annotations

The paper considers transferring style in faces as the most challenging task to meet expectations, particularly if the colors in the corresponding segments of the image are opposed: using semantic maps as annotations helps alleviate issues with patch- or gram-based style transfer, avoiding mixing between skin colors and background.

7.5.2 Parameter Ranges

Considering a fixed value for the parameter α, the paper tries to define some suggestions about the values of the parameters β, for the style error, and γ, for the degree of information coming from the semantic annotation. Setting $\beta = 0$ leads to a simple content reconstruction; when γ is set too high, the quality and the variety of the style tends to degenerate, while when γ decreases the algorithm progressively reverts to its semantically unaware version. The paper suggests to find a default value of γ, by equalizing the value range of the semantic channels m^l and convolution activation x^l: after that it's possible to test different values of the style weight β to obtain different meaningful interpolations.

7.6 Analysis

The paper provides different observations regarding the proposed algorithm:

- **Semantic Map Values** The semantic channels m^l affect how the normalized cross-correlation takes place, so if the channel range is large, the values from convolution x^l will be scaled very differently; in most cases it seems sensible to make sure values in m^l have similar magnitude.
- **Authored representations** Semantic embedding that compactly describe different pixel classes seems to be better suited than plain layer masks.
- **Content Accuracy vs. Style Quality** When using semantic maps, only the style patches from the appropriate segment can be used for the target image, so when the number of source patches is small, repetitive patterns can be created.
- **Blending Edges** The algorithm does a great job in painting the borders between different image segments.
- **Weight sensitivity** The algorithm is less fragile to adjustments in style weight: the semantic map helps maintain the results more consistent for a wider range of the parameter space.
- **Performance** Due to the usage of additional channels, the algorithm requires more memory and computational resources: however when the images are composed by RGB channels, the effort is still acceptable.

7.7 Conclusion

In this paper, existing style transfer techniques are improved by annotating input images with a semantic map, either manually authored or from pixel labeling algorithms. An augmented CNN architecture is introduced to leverage this information at runtime, showing that existing patch-based algorithms require minor adjustments and perform very well using this additional information. Greta results are obtained not only in terms of increased precision, but also of reduced unpredictability.

8 Layer Normalization

9 Instance Normalization: The Missing Ingredient for Fast Stylization

10 Deep Painterly Harmonization

Recap of paper [1]. They try to apply style transfer in order to uniform the style of an image onto another. They build a 2-step algorithm:

- First step will do patch match like [2].
- Second step is similar but more constrained on the style transformation
- Post processing is done also.

10.1 First step

10.2 Second step

References

[1] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep painterly harmonization. *CoRR*, abs/1804.03189, 2018.

[2] Chuan Li and Michael Wand. Combining markov random fields and convolutional neural networks for image synthesis. *CoRR*, abs/1601.04589, 2016.