Artistic Style Transfer

Saifuddin Syed

MLRG Fall 2016

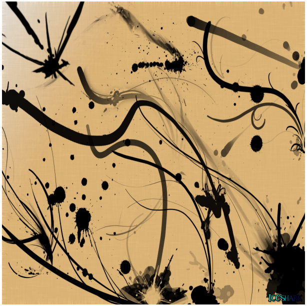
Outline

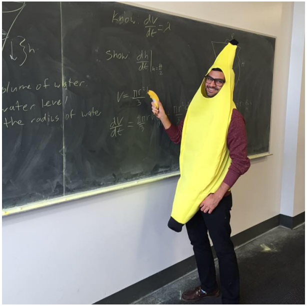
Introduction

Suppose I give you a piece of art and a photo. I ask you to recreate the photo in the style of the art. How would one go about doing it?

Introduction

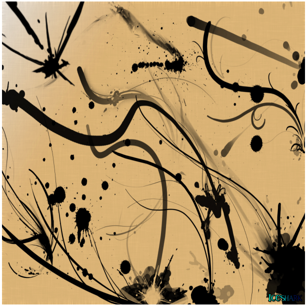
Suppose I give you a piece of art and a photo. I ask you to recreate the photo in the style of the art. How would one go about doing it?

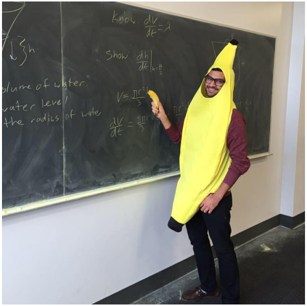




Introduction

Suppose I give you a piece of art and a photo. I ask you to recreate the photo in the style of the art. How would one go about doing it?





This is a very difficult task for humans, even talented ones.

Our goal is to teach a computer to do exactly this.

Outline

CNN Overview

Convolutional neural networks (CNN) are a type of neural network which have been widely used for image recognition tasks.

CNN Overview

Convolutional neural networks (CNN) are a type of neural network which have been widely used for image recognition tasks.

We input an image and each layer applies a set of filters that identify local features in the network.

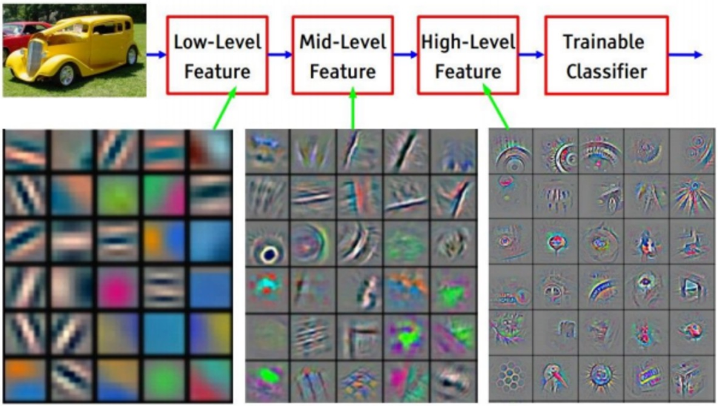
CNN Overview

Convolutional neural networks (CNN) are a type of neural network which have been widely used for image recognition tasks.

We input an image and each layer applies a set of filters that identify local features in the network.

Typically the deeper we go in the network, high level content is identified as opposed to just pixel values.

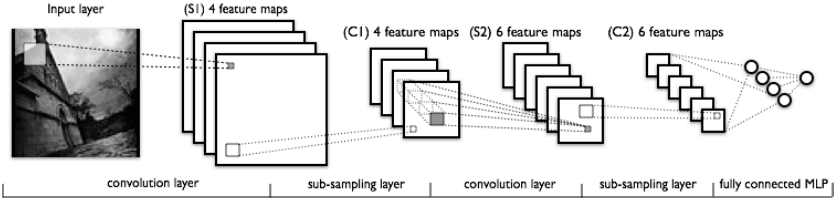
CNN Overview



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Feature Maps

Suppose layer l of the network has N_l filters, we will refer the collection of filtered images the **feature maps** at layer l.



VGG Network

The content generated in this talk was done using the VGG-19 network without the fully connected layer. Implementations in Lenet and Resnet also exist.

VGG Network

The content generated in this talk was done using the VGG-19 network without the fully connected layer. Implementations in Lenet and Resnet also exist.

Properties of VGG.

- Won ImageNet with a 7.3% top 5 error rate.
- Only 3x3 Conv stride 1, pad 1
- 2x2 MAX POOL stride 2
- 140 Million parameters

Outline

Artistic Style Transfer

We will now outline the procedure of Gatys, Ecker, Bethge. (Sept 2015).

Artistic Style Transfer

We will now outline the procedure of Gatys, Ecker, Bethge. (Sept 2015).

The key finding of this paper is that the representations of "content" and "style" in the Convolutional Neural Network are separable.

Artistic Style Transfer

We will now outline the procedure of Gatys, Ecker, Bethge. (Sept 2015).

The key finding of this paper is that the representations of "content" and "style" in the Convolutional Neural Network are separable.

We will make precise what we mean by style and content, but first, let us set up the problem formally.

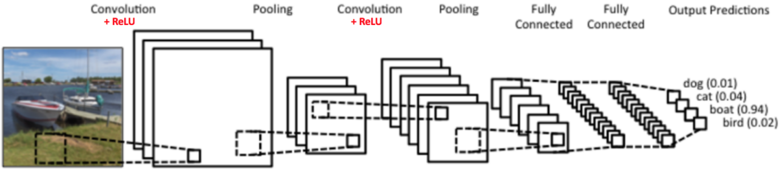
Notation

Our aim to is to construct an image **x** with the content of image **p** in the style of image **a**.

Notation

Our aim to is to construct an image $\mathbf{x}$ with the content of image $\mathbf{p}$ in the style of image $\mathbf{a}$.

We suppose that in our network, layer l has N_l filters, each with spatial dimension M_l (the product of its width and height).



Notation

Let $\Phi^l(\cdot)$ the function implemented by the part of the convolutional network from input up to the layer l.

Notation

Let $\Phi^l(\cdot)$ the function implemented by the part of the convolutional network from input up to the layer l.

The feature maps extracted by the network from the original image $\mathbf{p}$, the style image $\mathbf{a}$ and the stylized image $\mathbf{x}$ we denote by $\mathbf{P}^l$, $\mathbf{S}^l$, and $\mathbf{F}^l$ respectively.

Notation

Let $\Phi^l(\cdot)$ the function implemented by the part of the convolutional network from input up to the layer l.

The feature maps extracted by the network from the original image $\mathbf{p}$, the style image $\mathbf{a}$ and the stylized image $\mathbf{x}$ we denote by $\mathbf{P}^l$, $\mathbf{S}^l$, and $\mathbf{F}^l$ respectively.

$$\mathbf{P}^l = \Phi^l(\mathbf{p}), \quad \mathbf{S}^l = \Phi^l(\mathbf{a}), \quad \mathbf{F}^l = \Phi^l(\mathbf{x})$$

Notation

Let $\Phi^l(\cdot)$ the function implemented by the part of the convolutional network from input up to the layer l.

The feature maps extracted by the network from the original image $\mathbf{p}$, the style image $\mathbf{a}$ and the stylized image $\mathbf{x}$ we denote by $\mathbf{P}^l$, $\mathbf{S}^l$, and $\mathbf{F}^l$ respectively.

$$\mathbf{P}^l = \Phi^l(\mathbf{p}), \quad \mathbf{S}^l = \Phi^l(\mathbf{a}), \quad \mathbf{F}^l = \Phi^l(\mathbf{x})$$

The dimensionality of these feature maps is $N_l \times M_l$.

Content Representation

Each layer aims to learn a different aspect of the image content. It is reasonable to assume that two images with similar content should have similar feature maps at each layer.

Content Representation

Each layer aims to learn a different aspect of the image content. It is reasonable to assume that two images with similar content should have similar feature maps at each layer.

We will say $\mathbf{x}$ matches the content of $\mathbf{p}$ at layer l, if their feature responses at layer l of the network are the same.

Content Loss

Let F^l_{ij} and P^l_{ij} be the j^{th} position of filter i in layer l of the network.

Content Loss

Let F^l_{ij} and P^l_{ij} be the j^{th} position of filter i in layer l of the network.

We define the content loss at layer l to be,

$$\begin{aligned}\mathcal{L}^l_c(\mathbf{x}, \mathbf{p}) &= \frac{1}{2N_l M_l} \|\Phi^l(\mathbf{x}) - \Phi^l(\mathbf{p})\|_2^2 \\ &= \frac{1}{2N_l M_l} \sum_{i,j} |F^l_{ij} - P^l_{ij}|^2\end{aligned}$$

Content Loss

Let F^l_{ij} and P^l_{ij} be the j^{th} position of filter i in layer l of the network.

We define the content loss at layer l to be,

$$\mathcal{L}^l_c(\mathbf{x}, \mathbf{p}) = \frac{1}{2N_l M_l} \|\Phi^l(\mathbf{x}) - \Phi^l(\mathbf{p})\|_2^2 \\ = \frac{1}{2N_l M_l} \sum_{i,j} |F^l_{ij} - P^l_{ij}|^2$$

We define our content reconstruction $\mathbf{x}^l_c$ to be

$$\mathbf{x}^l_c = \mathrm{argmin}_{\mathbf{x}} \mathcal{L}^l_c(\mathbf{x}, \mathbf{p})$$

Content Loss

We have $\mathcal{L}_c^l$ satisfies

$$\frac{\partial \mathcal{L}_c^l}{\partial F_{ij}^l} = \begin{cases} (\mathbf{F}^l - \mathbf{P}^l)_{ij} & F_{ij}^l > 0 \\ 0 & F_{ij}^l < 0 \end{cases}$$

Content Loss

We have $\mathcal{L}_c^l$ satisfies

$$\frac{\partial \mathcal{L}_c^l}{\partial F_{ij}^l} = \begin{cases} (\mathbf{F}^l - \mathbf{P}^l)_{ij} & F_{ij}^l > 0 \\ 0 & F_{ij}^l < 0 \end{cases}$$

We can use back propagation and descent methods to iteratively minimize $\mathcal{L}_c^l$ and learn $\mathbf{x}_c^l$.

Content Loss

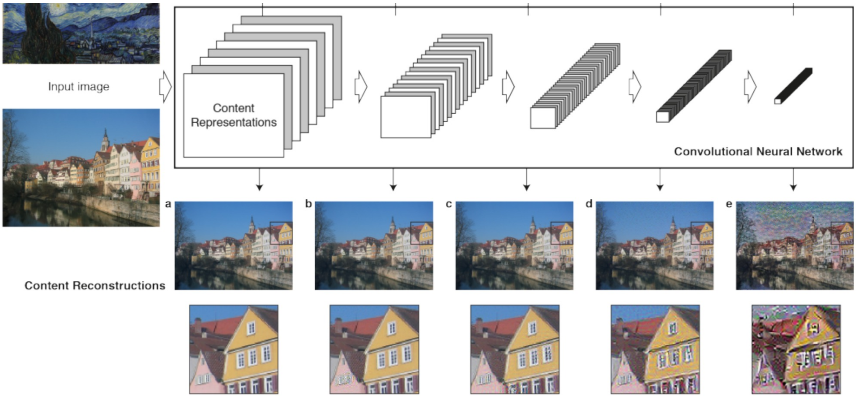
We have $\mathcal{L}_c^l$ satisfies

$$\frac{\partial \mathcal{L}_c^l}{\partial F_{ij}^l} = \begin{cases} (\mathbf{F}^l - \mathbf{P}^l)_{ij} & F_{ij}^l > 0 \\ 0 & F_{ij}^l < 0 \end{cases}$$

We can use back propagation and descent methods to iteratively minimize $\mathcal{L}_c^l$ and learn $\mathbf{x}_c^l$.

Normally $\mathbf{x}$ is initialized as a Gaussian white noise.

Content Reconstruction



Content Reconstruction

Higher layers in the network capture the high-level content in terms of objects and their arrangement in the input image but do not constrain the exact pixel values of the reconstruction.

Content Reconstruction

Higher layers in the network capture the high-level content in terms of objects and their arrangement in the input image but do not constrain the exact pixel values of the reconstruction.

In contrast, reconstructions from the lower layers simply reproduce the exact pixel values of the original image.

Content Reconstruction

Higher layers in the network capture the high-level content in terms of objects and their arrangement in the input image but do not constrain the exact pixel values of the reconstruction.

In contrast, reconstructions from the lower layers simply reproduce the exact pixel values of the original image.

The feature responses in higher layers better encode the content of the image.

Style Representation

The feature responses of an image **a** at layer l encode the content, however to determine style we are less interested in any individual feature of our image but rather how they all relate to each other.

Style Representation

The feature responses of an image **a** at layer l encode the content, however to determine style we are less interested in any individual feature of our image but rather how they all relate to each other.

The style consists of the **correlations** between the different feature responses.

Style Representation

The feature responses of an image **a** at layer l encode the content, however to determine style we are less interested in any individual feature of our image but rather how they all relate to each other.

The style consists of the **correlations** between the different feature responses.

We will say **x** matches the style of **a** at layer l, if the correlations between their feature maps at layer l of the network are the same.

Style Representation

The feature responses of an image **a** at layer l encode the content, however to determine style we are less interested in any individual feature of our image but rather how they all relate to each other.

The style consists of the **correlations** between the different feature responses.

We will say **x** matches the style of **a** at layer l, if the correlations between their feature maps at layer l of the network are the same.

This was the main insight of Gatys, et al.

Style Representation

We will encode the correlations of the feature maps into the Graham Matrices,

$$A_{ij}^{l} = \mathbf{S}_{i\bullet}^{l} \cdot \mathbf{S}_{j\bullet}^{l} = \sum_{k=1}^{M_l} S_{ik}^{l} S_{jk}^{l}$$

$$G_{ij}^{l} = \mathbf{F}_{i\bullet}^{l} \cdot \mathbf{F}_{j\bullet}^{l} = \sum_{k=1}^{M_l} F_{ik}^{l} F_{jk}^{l}$$

$\mathbf{A}^l$ and $\mathbf{G}^l$ define a $N_l \times N_l$ dimension matrix, where N_l is the number of filters in layer l.

Style Loss

We define the style loss at layer l to be,

$$\begin{aligned}\mathcal{L}_s^l(\mathbf{x}, \mathbf{a}) &= \frac{1}{4N_l^2 M_l^2} \|\mathbf{G}^l - \mathbf{A}^l\|_F^2 \\ &= \frac{1}{4N_l^2 M_l^2} \sum_{i,j} |G_{ij}^l - A_{ij}^l|^2\end{aligned}$$

Style Loss

We define the style loss at layer l to be,

$$\begin{aligned}\mathcal{L}_s^l(\mathbf{x}, \mathbf{a}) &= \frac{1}{4N_l^2 M_l^2} \|\mathbf{G}^l - \mathbf{A}^l\|_F^2 \\ &= \frac{1}{4N_l^2 M_l^2} \sum_{i,j} |G_{ij}^l - A_{ij}^l|^2\end{aligned}$$

We define our style reconstruction $\mathbf{x}_s^l$ to be

$$\mathbf{x}_s^l = \mathrm{argmin}_{\mathbf{x}} \mathcal{L}_s^l(\mathbf{x}, \mathbf{a})$$

Style Loss

Similar to the content loss, we have $\mathcal{L}_s^l$ satisfies

$$\frac{\partial \mathcal{L}_s^l}{\partial F_{ij}^l} = \begin{cases} \frac{1}{N_l^2 M_l^2}((\mathbf{F}^l)^T(\mathbf{G}^l - \mathbf{A}^l))_{ij} & \mathbf{F}_{ij}^l > 0 \\ 0 & \mathbf{F}_{ij}^l < 0 \end{cases}$$

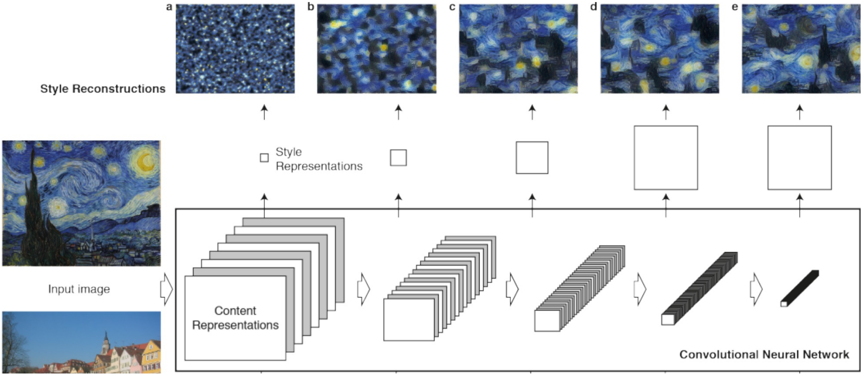
Style Loss

Similar to the content loss, we have $\mathcal{L}_s^l$ satisfies

$$\frac{\partial \mathcal{L}_s^l}{\partial F_{ij}^l} = \begin{cases} \frac{1}{N_l^2 M_l^2}((\mathbf{F}^l)^T(\mathbf{G}^l - \mathbf{A}^l))_{ij} & \mathbf{F}_{ij}^l > 0 \\ 0 & \mathbf{F}_{ij}^l < 0 \end{cases}$$

We can use back propagation and descent methods to iteratively minimize $\mathcal{L}_s^l$ and learn $\mathbf{x}_s^l$

Style Reconstruction



Style Reconstruction

Note the size and complexity of local image structures from the input image increases along the hierarchy.

Style Reconstruction

Note the size and complexity of local image structures from the input image increases along the hierarchy.

Heuristically, the higher layers learn more complex features than lower layers, and produce a more detailed style representation.

Image Construction

We now want to combine the content and style constructions outlined previously to develop an image $\mathbf{x}$ which simultaneously tries to match the content of $\mathbf{p}$ with the style of $\mathbf{a}$.

Image Construction

We now want to combine the content and style constructions outlined previously to develop an image $\mathbf{x}$ which simultaneously tries to match the content of $\mathbf{p}$ with the style of $\mathbf{a}$.

We will define our content (style) loss as the weighted average of the style (content) loss at each layer.

$$\mathcal{L}_c(\mathbf{x}, \mathbf{p}) = \sum_l \alpha^l \mathcal{L}_c^l(\mathbf{x}, \mathbf{p})$$

$$\mathcal{L}_s(\mathbf{x}, \mathbf{a}) = \sum_l \beta^l \mathcal{L}_s^l(\mathbf{x}, \mathbf{p})$$

Image Construction

We now want to combine the content and style constructions outlined previously to develop an image $\mathbf{x}$ which simultaneously tries to match the content of $\mathbf{p}$ with the style of $\mathbf{a}$.

We will define our content (style) loss as the weighted average of the style (content) loss at each layer.

$$\mathcal{L}_c(\mathbf{x}, \mathbf{p}) = \sum_l \alpha^l \mathcal{L}_c^l(\mathbf{x}, \mathbf{p})$$
$$\mathcal{L}_s(\mathbf{x}, \mathbf{a}) = \sum_l \beta^l \mathcal{L}_s^l(\mathbf{x}, \mathbf{p})$$

Often we take the $\alpha^l = 0$ for low l, and $\beta^l = 1$.

Image Construction

To match the content we need to minimize $\mathcal{L}_c$ and to match the style we need to minimize $\mathcal{L}_s$. Therefore we will minimize both simultaneously by minimizing

$$\mathcal{L}(\mathbf{x}, \mathbf{p}, \mathbf{a}) = \alpha \mathcal{L}_c(\mathbf{x}, \mathbf{p}) + \beta \mathcal{L}_s(\mathbf{x}, \mathbf{a}).$$

Image Construction

To match the content we need to minimize $\mathcal{L}_c$ and to match the style we need to minimize $\mathcal{L}_s$. Therefore we will minimize both simultaneously by minimizing

$$\mathcal{L}(\mathbf{x}, \mathbf{p}, \mathbf{a}) = \alpha \mathcal{L}_c(\mathbf{x}, \mathbf{p}) + \beta \mathcal{L}_s(\mathbf{x}, \mathbf{a}).$$

and we define the image $\mathbf{x}^*$ as,

$$\mathbf{x}^* = \mathrm{argmin}_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mathbf{p}, \mathbf{a}).$$

Image Construction

$$\mathcal{L}(\mathbf{x}, \mathbf{p}, \mathbf{a}) = \alpha \mathcal{L}_c(\mathbf{x}, \mathbf{p}) + \beta \mathcal{L}_s(\mathbf{x}, \mathbf{a}).$$

The constants α and β dictate how much preference we give to content matching vs style matching.

Image Construction

$$\mathcal{L}(\mathbf{x}, \mathbf{p}, \mathbf{a}) = \alpha \mathcal{L}_c(\mathbf{x}, \mathbf{p}) + \beta \mathcal{L}_s(\mathbf{x}, \mathbf{a}).$$

The constants α and β dictate how much preference we give to content matching vs style matching.

If $\frac{\alpha}{\beta}$ is high, we favour matching the content more than the style.

Image Construction

$$\mathcal{L}(\mathbf{x}, \mathbf{p}, \mathbf{a}) = \alpha \mathcal{L}_c(\mathbf{x}, \mathbf{p}) + \beta \mathcal{L}_s(\mathbf{x}, \mathbf{a}).$$

The constants α and β dictate how much preference we give to content matching vs style matching.

If $\frac{\alpha}{\beta}$ is high, we favour matching the content more than the style.

If $\frac{\alpha}{\beta}$ is low, we favour matching the style more than the content.

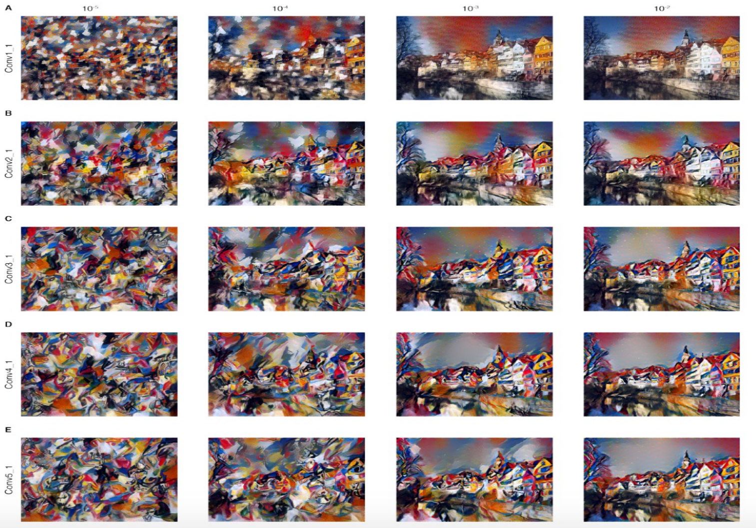


Figure: Columns: $\frac{\alpha}{\beta}$. Row: Layer of network

Examples

B

Figure: **Left:** Neckarfront in Tubingen, Germany, **B:** *The Shipwreck of the Minotaur* by J.M.W. Turner, 1805

Examples

c

Figure: **Left:** Neckarfront in Tubingen, Germany, **C:** *The Starry Night* by Vincent van Gogh, 1889

Examples

D

Figure: **Left:** Neckarfront in Tubingen, Germany, **D:** *Der Schrei* by Edvard Munch, 1893

Examples

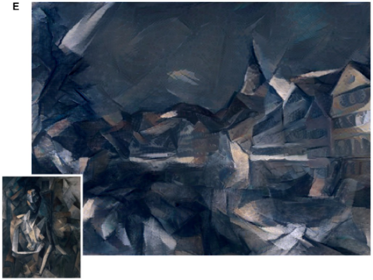


Figure: **Left:** Neckarfront in Tubingen, Germany, **E:** *Femme nue assise* by Pablo Picasso, 1910

Examples

F

Figure: **Left:** Neckarfront in Tubingen, Germany, **F:** *Composition VII* by Wassily Kandinsky, 1913

Examples

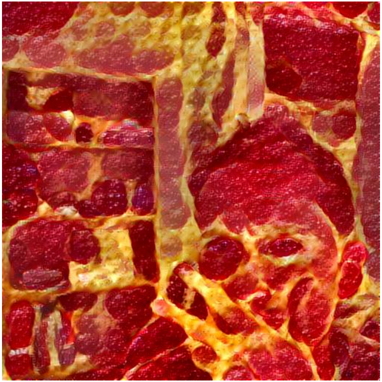


Figure: **Left:** My friend Grant, **Right:** Grant as a pizza

Outline

Alternative Methods

Following the success of Gatys et. al., many people began to refine and improve upon their method.

Alternative Methods

Following the success of Gatys et. al., many people began to refine and improve upon their method.

Some examples include:

- “Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis” (Li and Wand, Jan 2016)
- “Perceptual Losses for Real-Time Style Transfer and Super-Resolution” (Johnson, et al, Mar 2016)
- “Semantic Style Transfer and Turning Two-Bit Doodles into Fine Artwork”(Champandard, Mar 2016)

Alternative Methods

Following the success of Gatys et. al., many people began to refine and improve upon their method.

Some examples include:

- "Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis" (Li and Wand, Jan 2016)
- "Perceptual Losses for Real-Time Style Transfer and Super-Resolution" (Johnson, et al, Mar 2016)
- "Semantic Style Transfer and Turning Two-Bit Doodles into Fine Artwork"(Champandard, Mar 2016)

We will outline the work of Li and Wand.

MRF construction of Li and Wand

Their method was analogous to that of Gatys with a different style representation.

MRF construction of Li and Wand

Their method was analogous to that of Gatys with a different style representation.

Following Gatys, et al. Li and Wand, tried to match the content and style simultaneously by trying to minimize a linear combination of content and style loss function in addition to a regularizer.

$$\mathcal{L}^{MRF}(\mathbf{x}, \mathbf{p}, \mathbf{a}) = \alpha \mathcal{L}_c(\mathbf{x}, \mathbf{p}) + \beta \tilde{\mathcal{L}}_s(\mathbf{x}, \mathbf{a}) + \lambda R(\mathbf{x})$$

MRF construction of Li and Wand

Their method was analogous to that of Gatys with a different style representation.

Following Gatys, et al. Li and Wand, tried to match the content and style simultaneously by trying to minimize a linear combination of content and style loss function in addition to a regularizer.

$$\mathcal{L}^{MRF}(\mathbf{x}, \mathbf{p}, \mathbf{a}) = \alpha \mathcal{L}_c(\mathbf{x}, \mathbf{p}) + \beta \tilde{\mathcal{L}}_s(\mathbf{x}, \mathbf{a}) + \lambda R(\mathbf{x})$$

Where $\mathcal{L}_c$ is the same content loss function used in the Gatys construction.

Style Loss

Li and Wand also noted that the style of an image is encoded in the feature maps in a given layer of the network. Where their approach is differs is in how they extract it.

Style Loss

Li and Wand also noted that the style of an image is encoded in the feature maps in a given layer of the network. Where their approach is differs is in how they extract it.

Unlike using correlations and Graham matrices, they noted that the “style” can encoded locally via patches in the feature maps in a given layer l.

Style Loss

Li and Wand also noted that the style of an image is encoded in the feature maps in a given layer of the network. Where their approach is differs is in how they extract it.

Unlike using correlations and Graham matrices, they noted that the "style" can encoded locally via patches in the feature maps in a given layer l.

The patches are of dimension $k \times k \times N_l$, where k is the width and height of the patch (typically k is small) and N_l is the number of filters in layer l.

Style Loss

Li and Wand also noted that the style of an image is encoded in the feature maps in a given layer of the network. Where their approach is differs is in how they extract it.

Unlike using correlations and Graham matrices, they noted that the "style" can encoded locally via patches in the feature maps in a given layer l.

The patches are of dimension $k \times k \times N_l$, where k is the width and height of the patch (typically k is small) and N_l is the number of filters in layer l.

Our goal will be to match patches of $\Phi^l(\mathbf{x})$ to $\Phi^l(\mathbf{a})$ in some layer l.

Style Loss

Let $\Psi(\Phi^l(\mathbf{x})) = \{\Psi_i(\Phi^l(\mathbf{x}))\}_{i=1}^{m_l}$ be an ordered list of all local patches extracted from $\Phi^l(\mathbf{x})$.

Style Loss

Let $\Psi(\Phi^l(\mathbf{x})) = \{\Psi_i(\Phi^l(\mathbf{x}))\}_{i=1}^{m_l}$ be an ordered list of all local patches extracted from $\Phi^l(\mathbf{x})$.

Given i, we define $NN(i)$ to be the position of the patch in $\Psi(\Phi^l(\mathbf{a}))$ that deviates the most from from $\Psi_i(\Phi^l(\mathbf{x}))$. I.e.

Style Loss

Let $\Psi(\Phi^l(\mathbf{x})) = \{\Psi_i(\Phi^l(\mathbf{x}))\}_{i=1}^{m_l}$ be an ordered list of all local patches extracted from $\Phi^l(\mathbf{x})$.

Given i, we define $NN(i)$ to be the position of the patch in $\Psi(\Phi^l(\mathbf{a}))$ that deviates the most from from $\Psi_i(\Phi^l(\mathbf{x}))$. I.e.

$$NN(i) = \mathrm{argmin}_j \frac{\Psi_i(\Phi^l(\mathbf{x})) \cdot \Psi_j(\Phi^l(\mathbf{a}))}{|\Psi_i(\Phi^l(\mathbf{x}))| \cdot |\Psi_j(\Phi^l(\mathbf{a}))|}$$

So we define $\tilde{\mathcal{L}}_s$ to be

$$\tilde{\mathcal{L}}_s(\mathbf{x}, \mathbf{a}) = \sum_{i=1}^{m_l} \|\Psi_i(\Phi^l(\mathbf{x})) - \Psi_{NN(i)}(\Phi^l(\mathbf{a}))\|^2$$

Regularizer

There is significant amount of low-level image information discarded during the training of the network. In consequence, reconstructing an image can often be noisy and unnatural.

Regularizer

There is significant amount of low-level image information discarded during the training of the network. In consequence, reconstructing an image can often be noisy and unnatural.

To correct this we add a regularizer that encourages smoothness in the final image.

Regularizer

There is significant amount of low-level image information discarded during the training of the network. In consequence, reconstructing an image can often be noisy and unnatural.

To correct this we add a regularizer that encourages smoothness in the final image.

We defining the discrete gradient of **x** as

$$\Delta \mathbf{x}_{i,j} = (x_{i+1,j} - x_{i,j}, x_{i,j+1} - x_{i,j}).$$

Regularizer

There is significant amount of low-level image information discarded during the training of the network. In consequence, reconstructing an image can often be noisy and unnatural.

To correct this we add a regularizer that encourages smoothness in the final image.

We defining the discrete gradient of $\mathbf{x}$ as

$$\Delta \mathbf{x}_{i,j} = (x_{i+1,j} - x_{i,j}, x_{i,j+1} - x_{i,j}).$$

There is smoothness in the image when $\|\Delta \mathbf{x}\|_2^2$ is small, so we let

$$R(\mathbf{x}) = \|\Delta \mathbf{x}\|_2^2 = \sum_{i,j} (x_{i+1,j} - x_{i,j})^2 + (x_{i,j+1} - x_{i,j})^2$$

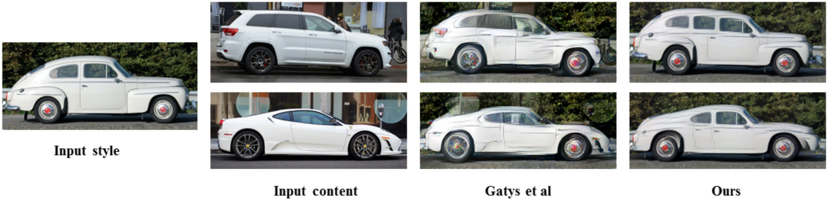
Examples

Input A   Input B   Content A + Style B   Content B + Style A

Examples

Content Image Gatys et al Ours

Examples



Input style — Input content — Gatys et al — Ours

Examples

Examples

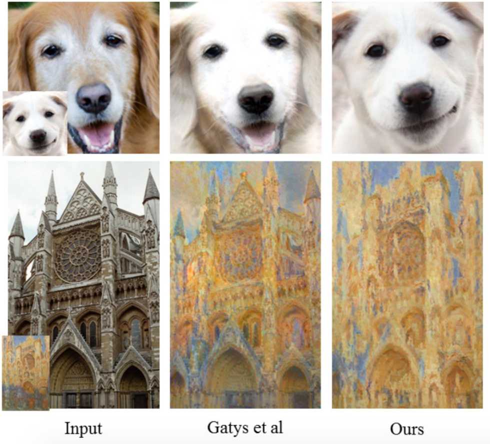


Figure: Example of Gatys, et al performing better

Outline

Artistic Style Transfer for Video

The natural extension to doing artistic style transfer for an image is to do artistic style transfer for a video.

Artistic Style Transfer for Video

The natural extension to doing artistic style transfer for an image is to do artistic style transfer for a video.

Ruder, Dosovitskiy, and Brox in April 2016 did exactly this. We will now outline their construction.

Artistic Style Transfer for Video

The natural extension to doing artistic style transfer for an image is to do artistic style transfer for a video.

Ruder, Dosovitskiy, and Brox in April 2016 did exactly this. We will now outline their construction.

The two main issues we will have to deal with is the initialization of the optimization procedure and the temporal consistency between frames.

Notation

We use the following notation: Let $\mathbf{p}$ be the content video with frames $\mathbf{p}^i$, and $\mathbf{a}$ be the style image. We want to create a video $\mathbf{x}$ such that each frame $\mathbf{x}^i$ has the content of $\mathbf{p}^i$ and style $\mathbf{a}$.

Notation

We use the following notation: Let $\mathbf{p}$ be the content video with frames $\mathbf{p}^i$, and $\mathbf{a}$ be the style image. We want to create a video $\mathbf{x}$ such that each frame $\mathbf{x}^i$ has the content of $\mathbf{p}^i$ and style $\mathbf{a}$.

Our goal will be to determine $\mathbf{x}^i$ in chronological order. We will also denote $\mathbf{x}_0^i$ to be the initialization of in the optimization procedure to determine $\mathbf{x}^i$.

Naive Method

The first natural thing to attempt is to apply your favourite artistic style routine to each frame individually.

Naive Method

The first natural thing to attempt is to apply your favourite artistic style routine to each frame individually.

However, the optimization procedure is not perfect and due to the random initialization, different frames fall into different local minima. This results in flickering and lack of continuity between frames.

Naive Method

The first natural thing to attempt is to apply your favourite artistic style routine to each frame individually.

However, the optimization procedure is not perfect and due to the random initialization, different frames fall into different local minima. This results in flickering and lack of continuity between frames.

The next natural step would be to initialize the optimization procedure by $\mathbf{x}_0^i = \mathbf{x}^{i-1}$.

If there is motion in the scene, this simple approach does not perform well since moving objects are initialized incorrectly.

Optical Flow to the rescue

The **optical flow** in a the content video $\mathbf{p}$ between frame j to i (denoted by w_j^i) is a function that warps a given image $\mathbf{p}^j$ using the optical flow field that was estimated between frame $\mathbf{p}^j$ and $\mathbf{p}^i$.

Optical Flow to the rescue

The **optical flow** in a the content video $\mathbf{p}$ between frame j to i (denoted by w_j^i) is a function that warps a given image $\mathbf{p}^j$ using the optical flow field that was estimated between frame $\mathbf{p}^j$ and $\mathbf{p}^i$.

Intuitively, it can be thought of as a function that predicts frame i in $\mathbf{p}$ given frame j. I.e.

$$w_j^i(\mathbf{p}^j) \approx \mathbf{p}^i.$$

Optical Flow to the rescue

The **optical flow** in a the content video $\mathbf{p}$ between frame j to i (denoted by w_j^i) is a function that warps a given image $\mathbf{p}^j$ using the optical flow field that was estimated between frame $\mathbf{p}^j$ and $\mathbf{p}^i$.

Intuitively, it can be thought of as a function that predicts frame i in $\mathbf{p}$ given frame j. I.e.

$$w_j^i(\mathbf{p}^j) \approx \mathbf{p}^i.$$

Optical flow can be computed via two state-of-the-art optical flow estimation algorithms: DeepFlow and EpicFlow.

Optical Flow to the rescue

The **optical flow** in a the content video $\mathbf{p}$ between frame j to i (denoted by w_j^i) is a function that warps a given image $\mathbf{p}^j$ using the optical flow field that was estimated between frame $\mathbf{p}^j$ and $\mathbf{p}^i$.

Intuitively, it can be thought of as a function that predicts frame i in $\mathbf{p}$ given frame j. I.e.

$$w_j^i(\mathbf{p}^j) \approx \mathbf{p}^i.$$

Optical flow can be computed via two state-of-the-art optical flow estimation algorithms: DeepFlow and EpicFlow.

Given the optical flow of w_{i-1}^i between frame $\mathbf{p}^{i-1}$ and $\mathbf{p}^i$ of the content video, we can initialize $\mathbf{x}_0^i$ via

$$\mathbf{x}_0^i = w_{i-1}^i(\mathbf{x}^{i-1}).$$

Temporal consistency

To force extra temporal continuity between frames, we will add a regularizer that rewards consecutive frames that are consistent with each other.

Temporal consistency

To force extra temporal continuity between frames, we will add a regularizer that rewards consecutive frames that are consistent with each other.

We define the temporal loss to be

$$\mathcal{L}_{temp}(\mathbf{x}, \mathbf{w}, \mathbf{c}) = \frac{1}{D} \sum_{k=1}^{D} c_k (x_k - w_k)^2$$

Where D is the dimension of the image. And $c_k = 0$ if the motion at pixel w_k is a boundary point, and 1 otherwise. $\mathbf{c}^i$ can be approximated using optical flow. For details of this procedure see Arxiv 1604.08610.

Temporal Consistency

To force some consistency between consecutive frames, we can minimize

$$\begin{aligned}\mathcal{L}_{short}(\mathbf{x}^i, \mathbf{p}^i, \mathbf{a}) = \; & \alpha \mathcal{L}_c(\mathbf{x}^i, \mathbf{p}^i) + \beta \mathcal{L}_s(\mathbf{x}^i, \mathbf{a}) \\ & + \gamma \mathcal{L}_{temp}(\mathbf{x}^i, w_{i-1}^i(\mathbf{x}^{i-1}), \mathbf{c}^i)\end{aligned}$$

Temporal Consistency

To force some consistency between consecutive frames, we can minimize

$$
\begin{aligned}
\mathcal{L}_{short}(\mathbf{x}^i, \mathbf{p}^i, \mathbf{a}) = \ & \alpha \mathcal{L}_c(\mathbf{x}^i, \mathbf{p}^i) + \beta \mathcal{L}_s(\mathbf{x}^i, \mathbf{a}) \\
& + \gamma \mathcal{L}_{temp}(\mathbf{x}^i, w_{i-1}^i(\mathbf{x}^{i-1}), \mathbf{c}^i)
\end{aligned}
$$

To get a smoother result, it is better to achieve some long term cosistency between not just the previous frame, but rather the previous J frames (typically $J = 1, 2, 4$).

$$
\begin{aligned}
\mathcal{L}_{long}(\mathbf{x}^i, \mathbf{p}^i, \mathbf{a}) = \ & \alpha \mathcal{L}_c(\mathbf{x}^i, \mathbf{p}^i) + \beta \mathcal{L}_s(\mathbf{x}^i, \mathbf{a}) \\
& + \gamma \sum_{j=1}^{J} \mathcal{L}_{temp}(\mathbf{x}^i, w_{i-j}^i(\mathbf{x}^{i-j}), \mathbf{c}^{i-j})
\end{aligned}
$$

Example

See https://www.youtube.com/watch?v=Khuj4ASldmU