A Flexible Convolutional Solver with Application to Photorealistic style transfer

Gilles Puy

Technicolor, Cesson-Sévigné, France

Joint work with P. Pérez

Deep Learning: From theory to applications
Cesson-Sévigné, September 06, 2018

Neural style transfer

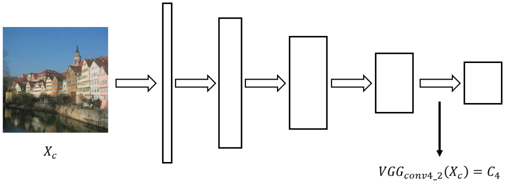
Neural style transfer

Neural style transfer [Gatys *et al.*, CVPR, 2016]

- Iterative updates to fit statistics of style image and content of target image
- Statistics: Gram matrices of feature maps at different layers of a pre-trained deep CNN.

technicolor

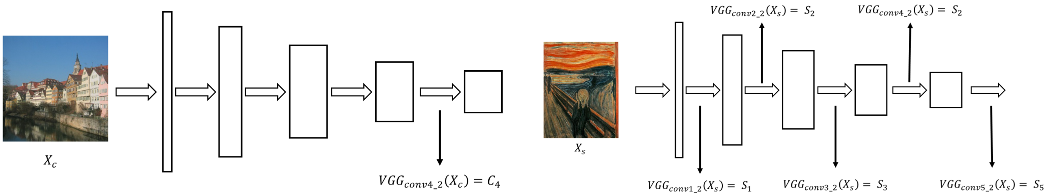
Neural style transfer



Content loss: $\mathcal{L}_c(X) = \frac{1}{n_4\, c_4} \|F_4 - C_4\|_F^2$, where $F_4 = \mathrm{VGG}_{\text{conv4\_2}}(X)$

technicolor

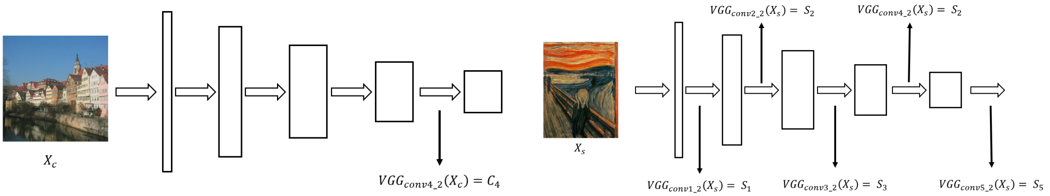
Neural style transfer



Content loss: $\mathcal{L}_c(\mathrm{X}) = \dfrac{1}{n_4\, c_4} \|\mathrm{F}_4 - \mathrm{C}_4\|_{\mathrm{F}}^2\,, \quad$ where $\mathrm{F}_4 = \mathrm{VGG}_{\mathrm{conv4\_2}}(X)$

Style loss: $\mathcal{L}_s(\mathrm{X}) = \dfrac{1}{|\mathcal{I}_s|} \displaystyle\sum_{\ell \in \mathcal{I}_s} \dfrac{1}{c_\ell^2} \left\| \dfrac{1}{n_\ell} \mathrm{F}_\ell^{\mathsf{T}} \mathrm{F}_\ell - \dfrac{1}{n'_\ell} \mathrm{S}_\ell^{\mathsf{T}} \mathrm{S}_\ell \right\|_{\mathrm{F}}^2, \quad$ where $\mathrm{F}_\ell = \mathrm{VGG}_{\mathrm{conv}\ell\_2}(X)$

Neural style transfer



Content loss: $\mathcal{L}_c(\mathrm{X}) = \dfrac{1}{n_4\, c_4} \|\mathrm{F}_4 - \mathrm{C}_4\|_{\mathrm{F}}^2$, where $\mathrm{F}_4 = \mathrm{VGG}_{\text{conv4\_2}}(X)$

Style loss: $\mathcal{L}_s(\mathrm{X}) = \dfrac{1}{|\mathcal{I}_s|} \displaystyle\sum_{\ell \in \mathcal{I}_s} \dfrac{1}{c_\ell^2} \left\| \dfrac{1}{n_\ell} \mathrm{F}_\ell^\intercal \mathrm{F}_\ell - \dfrac{1}{n'_\ell} \mathrm{S}_\ell^\intercal \mathrm{S}_\ell \right\|_{\mathrm{F}}^2$, where $\mathrm{F}_\ell = \mathrm{VGG}_{\text{conv}\ell\text{\_2}}(X)$

Complete loss: $\mathcal{L}(\mathrm{X}) = \lambda_c \mathcal{L}_c(\mathrm{X}) + \lambda_s \mathcal{L}_s(\mathrm{X})$, minimised by gradient descent.

technicolor

Neural style transfer

The method works well when the style is a painting but not when it is a photograph of a natural scene.

technicolor

Neural style transfer

The method works well when the style is a painting but not when it is a photograph of a natural scene.

For photorealistic style transfer, Luan *et al.*, CVPR, 2017 propose to regularize the problem:

$$\mathcal{L}(\mathsf{X}) = \lambda_c \, \mathcal{L}_c(\mathsf{X}) + \lambda_s \, \mathcal{L}_s(\mathsf{X}) + \lambda_L \, \mathrm{Tr}(\mathsf{X}^\intercal \mathsf{L} \mathsf{X}),$$

where L is the matting Laplacian computed on X_c [Levin *et al.*, IEEE PAMI, 2008].

Neural style transfer

technicolor

Neural style transfer

Without regularisation

Neural style transfer

Without regularisation

With regularisation $\mathrm{Tr}(X^{\intercal}LX)$

Neural style transfer

How can we estimate rapidly a solution of the photorealistic style transfer problem ?

$$\mathcal{L}_{\mathrm{ph}}(\mathsf{X}) = \lambda_c \, \mathcal{L}_c(\mathsf{X}) + \lambda_s \, \mathcal{L}_s(\mathsf{X}) + \lambda_L \, \mathrm{Tr}(\mathsf{X}^\intercal \mathsf{L} \mathsf{X}),$$

Neural style transfer

How can we estimate rapidly a solution of the photorealistic style transfer problem ?

$$\mathcal{L}_{\mathrm{ph}}(\mathsf{X}) = \lambda_c \, \mathcal{L}_c(\mathsf{X}) + \lambda_s \, \mathcal{L}_s(\mathsf{X}) + \lambda_L \, \mathrm{Tr}(\mathsf{X}^\intercal \mathsf{L} \mathsf{X}),$$

Re-use an existing network for fast artistic style transfer and retrain it.

Neural style transfer

How can we estimate rapidly a solution of the photorealistic style transfer problem ?

$$\mathcal{L}_{\mathrm{ph}}(\mathsf{X}) = \lambda_c\, \mathcal{L}_c(\mathsf{X}) + \lambda_s\, \mathcal{L}_s(\mathsf{X}) + \lambda_L \operatorname{Tr}(\mathsf{X}^\intercal \mathsf{L} \mathsf{X}),$$

Re-use an existing network for fast artistic style transfer and retrain it.

- Can the network capture the effect of $\operatorname{Tr}(\mathsf{X}^\intercal \mathsf{L} \mathsf{X})$?
- Which filters encode the effect of $\operatorname{Tr}(\mathsf{X}^\intercal \mathsf{L} \mathsf{X})$?
- If we want to change the regularisation, we need *a priori* to retrain.

technicolor

Neural style transfer

How can we estimate rapidly a solution of the photorealistic style transfer problem ?

$$\mathcal{L}_{\mathrm{ph}}(\mathsf{X}) = \lambda_c \, \mathcal{L}_c(\mathsf{X}) + \lambda_s \, \mathcal{L}_s(\mathsf{X}) + \lambda_L \, \mathrm{Tr}(\mathsf{X}^\intercal \mathsf{L} \mathsf{X}),$$

Re-use an existing network for fast artistic style transfer and retrain it.

- Can the network capture the effect of $\mathrm{Tr}(\mathsf{X}^\intercal \mathsf{L} \mathsf{X})$?
- Which filters encode the effect of $\mathrm{Tr}(\mathsf{X}^\intercal \mathsf{L} \mathsf{X})$?
- If we want to change the regularisation, we need *a priori* to retrain.

Following the idea of unrolling optimisation algorithms [Gregor *et al.*, ICML, 2010], we propose a flexible solver for which

- the network is trained to approximately minimise $\lambda_c \, \mathcal{L}_c(\mathsf{X}) + \lambda_s \, \mathcal{L}_s(\mathsf{X})$, *i.e.*, the artistic style transfer problem;
- the photorealistic prior can be added at runtime (no retraining).

A Flexible Solver

A Flexible Solver - Gradient descent and Projected gradient descent

A Flexible Solver - Gradient descent and Projected gradient descent

If $\mathcal{L}$ is differentiable, one can solve

$$\min_{\mathsf{X}} \mathcal{L}(\mathsf{X})$$

by gradient descent

$$\mathsf{X}^{(t+1)} = \mathsf{X}^{(t)} - \mu \nabla \mathcal{L}(\mathsf{X}^{(t)}).$$

A Flexible Solver - Gradient descent and Projected gradient descent

If $\mathcal{L}$ is differentiable, one can solve

$$\min_{\mathsf{X}} \mathcal{L}(\mathsf{X})$$

by gradient descent

$$\mathsf{X}^{(t+1)} = \mathsf{X}^{(t)} - \mu \nabla \mathcal{L}(\mathsf{X}^{(t)}).$$

If we add the constraint

$$\min_{\mathsf{X}} \mathcal{L}(\mathsf{X}) \quad \text{subject to} \quad \mathsf{X} \in \mathcal{C},$$

then a solution can be found using the iterations:

$$\mathsf{X}^{(t+1)} = \mathcal{P}_C \left[\mathsf{X}^{(t)} - \mu \nabla \mathcal{L}(\mathsf{X}^{(t)}) \right],$$

where $\mathcal{P}_C$ is the Euclidean projection onto $\mathcal{C}$.

A Flexible solver - Smooth graph signals

Why is it useful for solving the photorealistic style transfer of Luan *et al.* ?

$$\mathcal{L}(\mathsf{X}) = \lambda_c \, \mathcal{L}_c(\mathsf{X}) + \lambda_s \, \mathcal{L}_s(\mathsf{X}) + \lambda_L \, \mathrm{Tr}(\mathsf{X}^\intercal \mathsf{L} \mathsf{X}).$$

technicolor

A Flexible solver - Smooth graph signals

Why is it useful for solving the photorealistic style transfer of Luan *et al.* ?

$$\mathcal{L}(\mathsf{X}) = \lambda_c\, \mathcal{L}_c(\mathsf{X}) + \lambda_s\, \mathcal{L}_s(\mathsf{X}) + \lambda_L\, \mathrm{Tr}(\mathsf{X}^\intercal \mathsf{L} \mathsf{X}).$$

In graph signal processing terms, finding X for which $\mathrm{Tr}(\mathsf{X}^\intercal \mathsf{L}\mathsf{X})$ means finding an image that is is smooth on the graph with Laplacian L.

A Flexible solver - Smooth graph signals

Why is it useful for solving the photorealistic style transfer of Luan *et al.* ?

$$\mathcal{L}(\mathsf{X}) = \lambda_c \, \mathcal{L}_c(\mathsf{X}) + \lambda_s \, \mathcal{L}_s(\mathsf{X}) + \lambda_L \, \mathrm{Tr}(\mathsf{X}^\intercal \mathsf{L} \mathsf{X}).$$

In graph signal processing terms, finding X for which $\mathrm{Tr}(\mathsf{X}^\intercal \mathsf{L} \mathsf{X})$ means finding an image that is is smooth on the graph with Laplacian L.

There exists a special class of smooth graph signals: *k*-**bandlimited graph signals**.

A Flexible solver - Smooth graph signals

The eigendecomposition of L is

$$\mathsf{L} = \mathsf{U}\Lambda\mathsf{U}^{\intercal}$$

where

- $\mathsf{U} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n)$ contains the eigenvectors of L (graph Fourier basis).
- Λ is a diagonal matrix that contains the eigenvalues $\lambda_1 \leqslant \lambda_2 \leqslant \ldots \leqslant \lambda_n$ of L.

A Flexible solver - Smooth graph signals

A k-bandlimited graph signal satisfies

$$\boldsymbol{x} = \left(\begin{array}{ccc|ccc} \boldsymbol{u}_1 & \dots & \boldsymbol{u}_k & \boldsymbol{u}_{k+1} & \dots & \boldsymbol{u}_n \end{array} \right) \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_k \\ \hline 0 \\ \vdots \\ 0 \end{pmatrix} = \mathsf{U}_k \, \boldsymbol{\alpha}_k,$$

i.e., $\boldsymbol{x} \in \mathrm{span}(\mathsf{U}_k)$.

A Flexible solver - Photorealistic style transfer

Why is it useful for solving the photorealistic style transfer of Luan *et al.* ?

Why is it useful for solving the photorealistic style transfer of Luan *et al.* ?

$$\min_{\mathsf{X}} \lambda_c \, \mathcal{L}_c(\mathsf{X}) + \lambda_s \, \mathcal{L}_s(\mathsf{X}) + \lambda_L \, \mathrm{Tr}(\mathsf{X}^\intercal \mathsf{L} \mathsf{X}).$$

and

$$\min_{\mathsf{X}} \lambda_c \, \mathcal{L}_c(\mathsf{X}) + \lambda_s \, \mathcal{L}_s(\mathsf{X}) \quad \text{s.t.} \quad \boldsymbol{x}_i \in \mathrm{span}(\mathsf{U}_k), \ \forall i,$$

yields similar solutions ($\boldsymbol{x}_i$ denotes the i^{th} column of X).

technicolor

A Flexible solver

The artistic style transfer problem

$$\min_{\mathsf{X}} \mathcal{L}(\mathsf{X}) = \lambda_c \, \mathcal{L}_c(\mathsf{X}) + \lambda_s \, \mathcal{L}_s(\mathsf{X})$$

can be solved by

$$\mathsf{X}^{(t+1)} = \mathsf{X}^{(t)} - \mu \nabla \mathcal{L}(\mathsf{X}^{(t)}).$$

technicolor

A Flexible solver

The artistic style transfer problem

$$\min_{\mathsf{X}} \mathcal{L}(\mathsf{X}) = \lambda_c \, \mathcal{L}_c(\mathsf{X}) + \lambda_s \, \mathcal{L}_s(\mathsf{X})$$

can be solved by

$$\mathsf{X}^{(t+1)} = \mathsf{X}^{(t)} - \mu \nabla \mathcal{L}(\mathsf{X}^{(t)}).$$

The photorealistic style transfer problem

$$\min_{\mathsf{X}} \mathcal{L}(\mathsf{X}) \quad \text{s.t.} \quad \boldsymbol{x}_i \in \mathrm{span}(\mathsf{U}_k), \; \forall i,$$

can be solved by

$$\mathsf{X}^{(t+1)} = \mathcal{P}_{\mathrm{span}(\mathsf{U}_k)} \left[\mathsf{X}^{(t)} - \mu \nabla \mathcal{L}(\mathsf{X}^{(t)}) \right].$$

A Flexible solver

Fast solver for artistic and photorealistic style transfer

A Flexible solver

Fast solver for artistic and photorealistic style transfer

We propose to learn how to solve the **artistic** style transfer problem by replacing the true gradient with a learned update

$$\mathrm{X}^{(t+1)} = \mathrm{X}^{(t)} - \mu \nabla \mathcal{L}(\mathrm{X}^{(t)}) \quad \longrightarrow \quad \mathrm{X}^{(t+1)} = \mathrm{X}^{(t)} - g_t(\mathrm{X}^{(t)})$$

for $t = 0, \ldots, N-1$ (N=4).

A Flexible solver

Fast solver for artistic and photorealistic style transfer

We propose to learn how to solve the **artistic** style transfer problem by replacing the true gradient with a learned update

$$\mathsf{X}^{(t+1)} = \mathsf{X}^{(t)} - \mu \nabla \mathcal{L}(\mathsf{X}^{(t)}) \quad \longrightarrow \quad \mathsf{X}^{(t+1)} = \mathsf{X}^{(t)} - g_t(\mathsf{X}^{(t)})$$

for $t = 0, \ldots, N-1$ (N=4).

And propose to solve the **photorealistic** style transfer problem, **without retraining**, by adding $\mathcal{P}_{\mathrm{span}(\mathsf{U}_k)}$ after each update

$$\mathsf{X}^{(t+1)} = \mathcal{P}_{\mathrm{span}(\mathsf{U}_k)} \left[\mathsf{X}^{(t)} - g_t(\mathsf{X}^{(t)}) \right].$$

technicolor

Fast solver for artistic and photorealistic style transfer

We propose to learn how to solve the **artistic** style transfer problem by replacing the true gradient with a learned update

$$\mathsf{X}^{(t+1)} = \mathsf{X}^{(t)} - \mu \nabla \mathcal{L}(\mathsf{X}^{(t)}) \quad \longrightarrow \quad \mathsf{X}^{(t+1)} = \mathsf{X}^{(t)} - g_t(\mathsf{X}^{(t)})$$

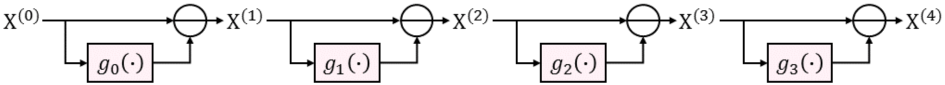
for $t = 0, \ldots, N-1$ (N=4).

And propose to solve the **photorealistic** style transfer problem, **without retraining**, by adding $\mathcal{P}_{\mathrm{span}(\mathsf{U}_k)}$ after each update

$$\mathsf{X}^{(t+1)} = \mathcal{P}_{\mathrm{span}(\mathsf{U}_k)} \left[\mathsf{X}^{(t)} - g_t(\mathsf{X}^{(t)}) \right].$$

There exist fast methods to compute $\mathcal{P}_{\mathrm{span}(\mathsf{U}_k)}$.

Architecture of g_t



Architecture of g_t

Continuing with the idea of unrolling gradient descent, we should mimic the architecture of $\nabla\mathcal{L}$ to build g_t:

$$\nabla\mathcal{L} = \lambda_c \, \nabla\mathcal{L}_c(\mathsf{X}^{(t)}) + \lambda_s \, \nabla\mathcal{L}_s(\mathsf{X}^{(t)}). \tag{1}$$

technicolor

Architecture of g_t

Continuing with the idea of unrolling gradient descent, we should mimic the architecture of $\nabla\mathcal{L}$ to build g_t:

$$\nabla\mathcal{L} = \lambda_c \nabla\mathcal{L}_c(\mathsf{X}^{(t)}) + \lambda_s \nabla\mathcal{L}_s(\mathsf{X}^{(t)}). \tag{1}$$

In practice, we do not consider $\nabla\mathcal{L}_c$ and mimic only the architecture of $\nabla\mathcal{L}_s$.

Architecture of g_t

Continuing with the idea of unrolling gradient descent, we should mimic the architecture of $\nabla\mathcal{L}$ to build g_t:

$$\nabla\mathcal{L} = \lambda_c \nabla\mathcal{L}_c(\mathsf{X}^{(t)}) + \lambda_s \nabla\mathcal{L}_s(\mathsf{X}^{(t)}). \tag{1}$$

In practice, we do not consider $\nabla\mathcal{L}_c$ and mimic only the architecture of $\nabla\mathcal{L}_s$.

To compensate, we

1. initialize $\mathsf{X}^{(0)} = \mathsf{X}_c$;
2. use the complete style transfer loss $\mathcal{L}$ for training.

Architecture of g_t

We have

$$\mathcal{L}_s(\mathsf{X}) = \frac{1}{|\mathcal{I}_s|} \sum_{\ell \in \mathcal{I}_s} \frac{1}{c_\ell^2} \left\| \frac{1}{n_\ell} \mathsf{F}_\ell^\intercal \mathsf{F}_\ell - \frac{1}{n'_\ell} \mathsf{S}_\ell^\intercal \mathsf{S}_\ell \right\|_{\mathrm{F}}^2,$$

where $\mathsf{F}_\ell = \mathrm{VGG}_\ell(\mathsf{X})$ and $\mathsf{S}_\ell = \mathrm{VGG}_\ell(\mathsf{X}_s)$.

Architecture of g_t

We have

$$\mathcal{L}_s(\mathsf{X}) = \frac{1}{|\mathcal{I}_s|} \sum_{\ell \in \mathcal{I}_s} \frac{1}{c_\ell^2} \left\| \frac{1}{n_\ell} \mathsf{F}_\ell^\intercal \mathsf{F}_\ell - \frac{1}{n'_\ell} \mathsf{S}_\ell^\intercal \mathsf{S}_\ell \right\|_{\mathrm{F}}^2 ,$$

where $\mathsf{F}_\ell = \mathrm{VGG}_\ell(\mathsf{X})$ and $\mathsf{S}_\ell = \mathrm{VGG}_\ell(\mathsf{X}_s)$.

The gradient is obtained in 3 steps

1. Compute each $\mathsf{F}_\ell = \mathrm{VGG}_\ell(\mathsf{X})$;
2. Compute each partial derivative:

$$\frac{\partial \mathcal{L}_s}{\partial \mathsf{F}_\ell} \propto \mathsf{F}_\ell \cdot \left[\frac{1}{n_\ell} \mathsf{F}_\ell^\intercal \mathsf{F}_\ell - \frac{1}{n'_\ell} \mathsf{S}_\ell^\intercal \mathsf{S}_\ell \right] ; \tag{2}$$

3. Back-propagate these partial derivatives to the input of the VGG-19.

technicolor

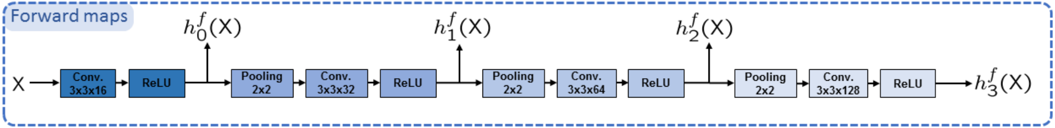
Architecture of g_t

Step 1: Compute each $\mathsf{F}_\ell = \mathrm{VGG}_\ell(\mathsf{X})$.

Architecture of g_t

Step 1: Compute each $\mathsf{F}_\ell = \mathrm{VGG}_\ell(\mathsf{X})$.

We replace the VGG-19 network with a similar but "simpler" one:



The forward maps $h_\ell^f(\mathsf{X})$ play the role of $\mathrm{VGG}_\ell(\mathsf{X})$. All the filters in $h_\ell^f(\mathsf{X})$ are trained.

Architecture of g_t

Step 2: Compute each partial derivative

$$\mathsf{F}_\ell \cdot \left[\frac{1}{n_\ell} \mathsf{F}_\ell^\intercal \mathsf{F}_\ell - \frac{1}{n'_\ell} \mathsf{S}_\ell^\intercal \mathsf{S}_\ell \right] \tag{3}$$

technicolor

Architecture of g_t

Step 2: Compute each partial derivative

$$\mathsf{F}_\ell \cdot \left[\frac{1}{n_\ell} \mathsf{F}_\ell^\intercal \mathsf{F}_\ell - \frac{1}{n'_\ell} \mathsf{S}_\ell^\intercal \mathsf{S}_\ell \right] \tag{3}$$

We directly transform the above expression to

$$h_\ell^f(\mathsf{X}) \cdot \left[\frac{1}{n_\ell} h_\ell^f(\mathsf{X})^\intercal h_\ell^f(\mathsf{X}) - \mathsf{H}_{\ell,t}^s \right], \tag{4}$$

where, therefore, $\mathsf{H}_{\ell,t}^s$ controls the style.

technicolor

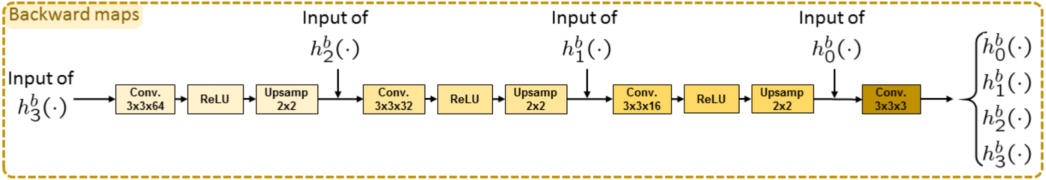
Architecture of g_t

Step 3: Backpropagation

Architecture of g_t

Step 3: Backpropagation

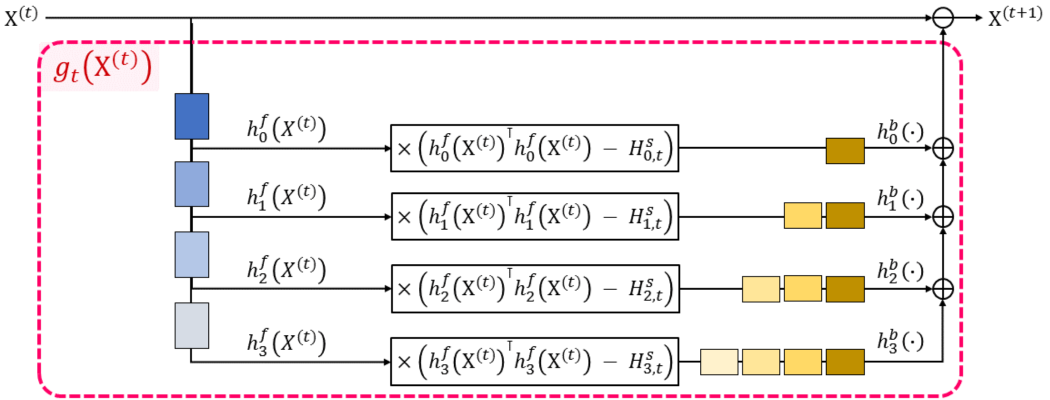
We use backwards maps h^b_ℓ symmetric to h^f_ℓ



technicolor

Architecture of g_t

Global structure

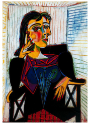


Some results

Artistic style transfer

X_c | [Gatys *et al.*] - X_c as init. | Ours

Style 1, *Portrait of Dora Maar*, P. Picasso, 1937.

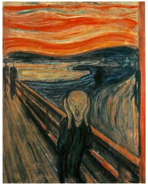


Style 2, *A Muse*, P. Picasso, 1935.

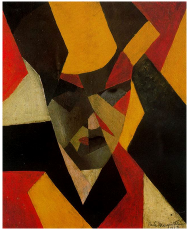
Artistic style transfer

X_c | [Gatys *et al.*] - X_c as init. | Ours

Style 3, *The scream*, E. Munch, 1893.



Style 4, *Self portrait*, R. Magritte, 1923.



technicolor

Photorealistic style transfer

X_c

Style

[Luan *et al.*]

Ours - Graph filt.

Photorealistic style transfer

X_c

Style

[Luan *et al.*]

Ours - Graph filt.

Photorealistic style transfer

X_c

Style

[Luan *et al.*]

Ours - Graph filt.

technicolor

Photorealistic style transfer

X_c

Style

Ours - No Graph filt.

Ours - Graph filt.

technicolor

Conclusion

Conclusion

- Deep networks can benefit from the flexibility of existing optimisation algorithms
- Unrolling algorithms make it easier to interpret the role of the filter in the network
- References and more details in our technical report arXiv:1806.05285
- Related work: Li *et al.*, "A Closed-form Solution to Photorealistic Image Stylization," ECCV, 2018.

technicolor