Unifying local and non-local signal processing with graph CNNs

Gilles Puy
gilles.puy@technicolor.com
Srđan Kitić
srdan.kitic@technicolor.com
Patrick Pérez
patrick.perez@technicolor.com

Technicolor
975 Avenue des Champs Blancs
35576 Cesson-Sévigné, France

Abstract

This paper deals with the unification of local and non-local signal processing on graphs within a single convolutional neural network (CNN) framework. Building upon recent works on graph CNNs, we propose to use convolutional layers that take as inputs two variables, a signal and a graph, allowing the network to adapt to changes in the graph structure. In this article, we explain how this framework allows us to design a novel method to perform style transfer.

1 Introduction

Convolutional neural networks (CNNs) have achieved unprecedented performance in a wide variety of applications, in particular for image analysis, enhancement and editing – *e.g.*, classification [14], super-resolution [8], and colorisation [24]. Yet standard CNNs can only handle signals that live on a regular grid, and each layer of a CNN only performs a local processing. Locality has already been identified as a limitation for classical signal processing tasks where powerful non-local methods have been proposed, such as patch-based methods for inpainting [6] or denoising [9, 22]. Regular CNNs do not allow such a non-local processing. Furthermore, the growing amount of signals collected on irregular grids, such as social, transportation or biological networks, requires extending signal processing from regular to irregular graphs [20].

Any CNN consists of a composition of convolutional and pooling layers. One should thus redefine both convolution and pooling to handle "graph signals". In this work, we use convolutional layers only and hence just concentrate on the generalisation of the convolution. One major challenge in this generalisation is to take into account the possible changes of the graph structure from one signal instance to another: nodes and edges can appear, disappear, and the edge weights can vary. For instance, the connections between the users (graph's vertices) in a social network change over time. It would be cumbersome to retrain a CNN each time a connection changes. In non-local signal processing methods, the situation is even more extreme as the graph is a construct whose edges typically capture similarities between different parts of the signal itself. In this case, the CNN must not be just robust to

few variations in the graph structure but fully adapt to these variations. We propose here a solution to this challenge but passing two variables to the CNN: the signal itself, as usual, and the graph structure.

Contributions – We propose a graph CNN framework that takes as inputs two variables: a signal and a graph structure. This permits the adaptation of the CNN to changes in the structure of the graph on which the signal lives, even in the extreme case where this structure changes with the input signal itself. We also propose a unique way of defining convolutions on arbitrary graphs, in particular non-local convolutions, with application to a wide range of many different signal processing applications. Due to space constraint, we only present the use of graph CNNs for image style transfer in this article. We use a local CNN to capture and transfer local style properties of the painting to the photograph. We also use a non-local graph CNN to capture and transfer global style properties of the painting, as well as to preserve the content of the photograph. In addition, we show that this task can be done using only *two random shallow* networks, instead of a trained regular deep CNN [9].

Let us mention that additional experiments in the supplementary material [18] demonstrate the effectiveness and versatility of our framework on other kinds of signals (greyscale images, color palettes, and speech signals) and tasks (color transfer and denoising). In particular, the experiments show that it is possible to identify the optimal mixing of local and non-local signal processing techniques by learning.

2 Graph CNN

2.1 State-of-the-art methods

In this section, we review different existing solutions to generalise CNNs to signal living on graphs. The reader can refer to [2] for a detailed overview. We restrict our attention here to the solutions the most closely connected to ours.

A first approach to redefine convolution for graph signals is to work in the spectral domain, for which we need to define the graph Fourier transform. To introduce this transform, we consider an undirected weighted graph[1] $\mathcal{G}$ with graph Laplacian denoted by $\mathsf{L} \in \mathbb{R}^{n\times n}$. For example, L can be the combinatorial graph Laplacian $\mathsf{L} = \mathsf{D} - \mathsf{W}$, or the normalised one $\mathsf{L} = \mathsf{I} - \mathsf{D}^{-1/2}\mathsf{W}\mathsf{D}^{-1/2}$, where I is the identity matrix and $\mathsf{D} \in \mathbb{R}^{n\times n}$ is the diagonal degree matrix with entries $d_i = \sum_{j=1}^{n} \mathsf{W}_{ij}$ [5]. The matrix L is real symmetric and positive semi-definite. Thus, there exists a set of orthonormal eigenvectors $\mathsf{U} \in \mathbb{R}^{n\times n}$ and real eigenvalues $0 = \lambda_1 \leqslant \ldots \leqslant \lambda_n$ such that $\mathsf{L} = \mathsf{U}\Lambda\mathsf{U}^\intercal$, where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n) \in \mathbb{R}^{n\times n}$. The matrix U is viewed as the graph Fourier basis [20].

For any signal $\boldsymbol{x} \in \mathbb{R}^n$ defined on the vertices of $\mathcal{G}$, $\hat{\boldsymbol{x}} = \mathsf{U}^\intercal\boldsymbol{x}$ is its graph Fourier transform. One way to define convolution on $\mathcal{G}$ with a filter $\boldsymbol{h} \in \mathbb{R}^n$ is by filtering in the graph Fourier domain:

$$\boldsymbol{x} \star \boldsymbol{h} = \mathsf{U}\left(\hat{\boldsymbol{h}} \odot \hat{\boldsymbol{x}}\right) = \mathsf{U}\mathsf{H}\mathsf{U}^\intercal\boldsymbol{x}, \tag{1}$$

where $\odot$ denotes the entry-wise multiplication and $\mathsf{H} = \mathrm{diag}(\hat{\boldsymbol{h}}) \in \mathbb{R}^{n\times n}$. In the context of graph CNN, it is the approach chosen in [3]. This approach however has several drawbacks:

[1] A graph $\mathcal{G}$ is a set of n vertices, a set of edges $\mathcal{E}$ and a weighted adjacency matrix $\mathsf{W} = [\mathsf{W}_{ij}] \in \mathbb{R}_+^{n\times n}$, with $\mathsf{W}_{ij} > 0$ iff $(i,j) \in \mathcal{E}$. In this paper, we consider directed graphs unless explicitly stated. The matrix W is thus not symmetric in general.

Computing U is often intractable for real-size graphs; Matrix-vector multiplication with U is usually slow (there is no fast graph Fourier transform); This definition does not allow variations in the graph structure as the matrix U is impacted by any such change; The number of filter coefficients to learn is as large as the size of the input signal. The subsequent work [12] solves this last issue by imposing that the filter lives in the span of a kernel matrix $\mathsf{K}^{n\times\tilde{n}}$ with $\tilde{n} \leqslant n$.

To overcome the computational issues of the spectral approach, a known trick in the field of graph signal processing is to define a filter as a polynomial of the graph eigenvalues [10]. Let $\hat{h}\colon \mathbb{R} \to \mathbb{R}$ be a polynomial of degree $m \geqslant 0$: $\hat{h}(t) = \sum_{i=0}^{m} \alpha_i t^i$, with $\alpha_0, \ldots, \alpha_m \in \mathbb{R}$ and consider the filter $\hat{\boldsymbol{h}} = (\hat{h}(\lambda_1), \ldots, \hat{h}(\lambda_n))^\intercal$. One can easily prove that spectral filtering with $\hat{\boldsymbol{h}}$ satisfies

$$\boldsymbol{x} \star \boldsymbol{h} = \mathsf{U}\,(\hat{\boldsymbol{h}} \odot \hat{\boldsymbol{x}}) = \sum_{i=0}^{m} \alpha_i \mathsf{L}^i \boldsymbol{x}. \tag{2}$$

This expression involves only computations in the vertex domain through matrix-vector multiplications with L. As the Laplacian is usually a sparse matrix, filtering a signal with a polynomial filter is fast. This is the approach adopted by [7] and [13] in their construction of graph CNNs. Beyond the computational improvements, the number of coefficients to learn is also reduced: m instead of n. Furthermore, the localisation of the filter in the vertex domain is exactly controlled by the degree of the polynomial [7, 10]. Yet these polynomial filters are not entirely satisfying. Indeed, for, *e.g.*, a graph modelling a regular lattice, polynomial filters are isotropic unlike those in regular CNNs for images – where the underlying graph is a regular lattice. There is no equivalence between regular CNNs and graph CNNs with polynomial filters. Let us also mention the work of [1] where the convolution is defined using a diffusion process on the graph. Due to lack of space, we do not report the exact definition but this one shares similarities with (2) where the normalised transition matrix $\mathsf{P} = \mathsf{D}^{-1}\mathsf{W}$ is substituted for the Laplacian.

In our work, we built upon the work of [17] and [16] to get rid of these shortcomings. The convolutions are directly defined in the vertex domain in a way which allows ones to directly control the computational complexity and the localisation of the filters. Furthermore, these filters do not suffer from the isotropy issue of polynomial filters.

2.2 Our method

Each layer of our graph CNN implements a function

$$f\colon (\mathsf{X}, \mathcal{G}) \quad \longmapsto f(\mathsf{X}, \mathcal{G}) \tag{3}$$

where $\mathsf{X} \in \mathbb{R}^{n\times m_0}$ is the input signal, $f(\mathsf{X}, \mathcal{G}) \in \mathbb{R}^{n\times m_1}$, and $\mathcal{G}$ is a n-vertex graph on which the columns of X live and which defines how the convolution is done in this layer. The input signal X has size n in the "spatial" dimension – *e.g.*, n pixels for images – and has m_0 channels or feature maps – *e.g.*, $m_0 = 3$ for color images. The output signal has same spatial size n – we do not use any pooling layers in this work – and m_1 feature maps.

2.2.1 Convolution

The convolution we use follows principles also used in, *e.g.*, [15, 16, 17, 19], where the computation done at one vertex is a function of (at least) the values of the signal at this

vertex and neighbouring vertices as well as of labels attributed to each edge. We choose here to use the formalism of [16] for our description.

Convolutions in [16] are done in two steps: the extraction of a signal patch around each vertex and a scalar product. We assume here that all vertices have the same number of connections: $|\{j\colon (i,j)\in\mathcal{E}\}| = d$ for all $i\in\{1,\dots,n\}$. If this is not the case, one can always complete the set of edges and associate to these edges, *e.g.*, a null weight. We also assume that $(i,i)\in\mathcal{E}$ for all $i\in\{1,\dots,n\}$.

For a given graph $\mathcal{G}$ satisfying the above assumption and with adjacency matrix $\mathsf{W}\in\mathbb{R}^{n\times n}$, we model patch extraction at vertex i with a function

$$\begin{aligned} p : \{1,\dots,n\}\times\mathbb{R}^n &\longrightarrow \mathbb{R}^d \\ (i,\boldsymbol{x}) &\longmapsto (p_1(i,\boldsymbol{x}),\dots,p_d(i,\boldsymbol{x}))^\mathsf{T} \end{aligned} \tag{4}$$

where each $p_\ell(i,\boldsymbol{x})\in\mathbb{R}$, $\ell=1,\dots,d$, extracts one entry of the vector $\boldsymbol{x}$, which represents one column of the input signal X. Let $j_1,\dots,j_d$ be the d indices to which i is connected. The order in which these d entries are extracted by p is determined by "pseudo-coordinates" $u(i,j_k)\in\{1,\dots,d\}$ attributed to each connected vertex j_k [16]. The nature of these pseudo-coordinates will be given in Section 2.2.2 for local convolution and in Section 2.2.3 for non-local convolution. We define

$$p_\ell(i,\boldsymbol{x}) = g(\boldsymbol{w}_i, j_k)\ \boldsymbol{x}_{j_k} \tag{5}$$

where $u(i,j_k)=\ell$. The vector $\boldsymbol{w}_i\in\mathbb{R}^n$ is the i^{th} row of W, which contains at most d non-zero entries, and $g:\mathbb{R}^n\times\mathbb{R}\to\mathbb{R}$ is a re-weighting function that gives the possibility to account for each edge weight in the convolution. We noticed that the choice of this function is very important in the definition of the non-local convolutions to achieve good results in our signal processing applications (see its definition in Section 2.2.3). Note that g depends on $\boldsymbol{w}_i$ and j_k in our work while this function depends solely on the pseudo-coordinates in [16]. This is a simple but important modification for our applications.

Convoluting $\boldsymbol{x}$ with a filter $\boldsymbol{h}\in\mathbb{R}^d$ is then defined as in [16]:

$$(\boldsymbol{x}\star\boldsymbol{h})(i) = \boldsymbol{h}^\mathsf{T} p(i,\boldsymbol{x}), \tag{6}$$

for all $i\in\{1,\dots,n\}$. Finally, the function f in (3) satisfies

$$f(\mathsf{X},\mathcal{G}) = \left(s\Big(\sum_{j=1}^{m_0}\boldsymbol{x}_j\star\boldsymbol{h}_j^\ell, b^\ell\Big)\right)_{\ell=1,\dots,m_1} \tag{7}$$

where $s:\mathbb{R}\times\mathbb{R}\to\mathbb{R}$ is an element-wise non-linearity, *e.g.*, ReLU defined as $s(\boldsymbol{x},b) = \mathrm{ReLU}_b(\boldsymbol{x}) = \max\{0,\boldsymbol{x}+b\}$, $\boldsymbol{x}_j\in\mathbb{R}^n$ denotes the j^{th} column-vector of X, $\boldsymbol{h}_j^\ell\in\mathbb{R}^d$, $j=1,\dots,m_0$, $\ell=1,\dots,m_1$, are filters, and $b^1,\dots,b^{m_1}$ are biases.

Let us highlight that the size n of the input signal X in the spatial dimension is not fixed in (7). Hence, f can be computed for signals of different sizes using the same filters $\boldsymbol{h}_j^\ell$, exactly as with regular CNNs.

We explain in the next section how one can recover the usual local convolution for images from this definition. We will then continue with the description of the proposed non-local filtering in the Section 2.2.3.

2.2.2 Local convolution

As noticed in [16], the above definition of convolution permits us to recover easily the standard convolution for images (or, similarly, signals on regular lattices) by constructing a local graph from the Cartesian $2D$-coordinates of each pixel in the image.[2] We denote these coordinates $(\alpha(i),\beta(i))$, $i=1,\dots,n$. For a filter of size $\sqrt{d}\times\sqrt{d}$, we connect each pixel i to all its local neighbours $j_1,\dots,j_d$ that satisfies $|\alpha(j_k)-\alpha(i)|\leqslant\lfloor\sqrt{d}/2\rfloor$ and $|\beta(j_k)-\beta(i)|\leqslant\lfloor\sqrt{d}/2\rfloor$, for $k=1,\dots,d$. We then build the local adjacency matrix W that satisfies $\mathsf{W}_{ij}=1$ if $(i,j)\in\mathcal{E}$, and 0 otherwise. The pseudo-coordinates are determined using the relative position of each pixel j_k to pixel i. For any pixel of the image, the connected pixels have relative coordinates in $\{(\alpha(i)-\alpha(j_k),\beta(i)-\beta(j_k)),1\leqslant k\leqslant d\}$. We thus create a look up table $\mathrm{c}:\mathbb{R}\times\mathbb{R}\to\{1,\dots,d\}$ that associates a unique integer $\ell\in\{1,\dots,d\}$ to each of these relative coordinates. Then, we define $u(i,j_k)=c\left(\alpha(i)-\alpha(j_k),\beta(i)-\beta(j_k)\right)$.

With this procedure the pixels are always extracted in the same order, *e.g.*, lexicographically. Finally, (6) is equivalent to the usual convolution when using $g(\cdot)=1$ in (5).

2.2.3 Non-local convolution

We now describe our proposition to perform more general non-local convolutions, *i.e.*, we give the definition of the pseudo-coordinates and of the function g in (5). In our applications, these convolutions are based on a graph $\mathcal{G}$ that captures some structure that we wish to preserve in the signal X. The exact construction of $\mathcal{G}$ thus differs depending on the application. Yet, we define the pseudo-coordinates and the function g always in the same way, whatever the application. Our main contributions with respect to [16, 17] resides in these choices of pseudo-coordinates and of function g which, empirically, seems universal enough to treat several applications.

The weight W_{ij} of the edge between vertices i and j is determined based on a distance between feature vectors $\boldsymbol{f}_i$ and $\boldsymbol{f}_j$ extracted at vertex i and j, respectively. Let $\Delta(\boldsymbol{f}_i,\boldsymbol{f}_j)\geqslant 0$ denote this distance. Let again $j_1,\dots,j_d$ be the vertices to which vertex i is connected and ordered such that $\Delta(\boldsymbol{f}_i,\boldsymbol{f}_{j_1})\leqslant\ldots\leqslant\Delta(\boldsymbol{f}_i,\boldsymbol{f}_{j_d})$. We propose to define the pseudo-coordinates as $u(i,j_k)=k$, $k\in\{1,\dots,d\}$. In other words, the pseudo-coordinates re-order the distances between feature vectors in increasing order. Note that we break any tie arbitrarily.

Finally, we propose to use the following function g in (5):

$$g(\boldsymbol{w}_i,j)=\frac{\boldsymbol{w}_{ij}}{\sum_{k=1}^{n}\boldsymbol{w}_{ik}}\,d, \tag{8}$$

where $\boldsymbol{w}_{ij}=\mathsf{W}_{ij}$ is the j^{th} entry of $\boldsymbol{w}_i$.

3 Style transfer with graph CNNs

In this section, we substitute $f(\mathsf{X})$ for $f(\mathsf{X},\mathcal{G})$ in (7) to simplify notations. However, one should not forget that the convolution at each layer is defined by an underlying graph $\mathcal{G}$. This graph will always be defined explicitly in the text.

Style transfer consists in transforming a target image $\mathsf{X}_t\in\mathbb{R}^{n\times 3}$, typically a photograph, to give it the "style" of a source image $\mathsf{X}_s\in\mathbb{R}^{n'\times 3}$, typically a painting. Impressive results

[2] We consider a regular grid of equispaced pixels.

have recently been obtained using CNNs [9]. The style transfer method of Gatys *et al.* consists in solving a minimization problem of the form

$$\mathsf{X}^* \in \operatorname*{argmin}_{\mathsf{X}\in\mathbb{R}^{n\times 3}} \sum_{\ell\in\mathcal{L}_s} \|f_\ell(\mathsf{X})^\intercal f_\ell(\mathsf{X}) - f_\ell(\mathsf{X}_s)^\intercal f_\ell(\mathsf{X}_s)\|_{\mathrm{F}}^2 + \lambda \sum_{\ell\in\mathcal{L}_t} \|f_\ell(\mathsf{X}) - f_\ell(\mathsf{X}_t)\|_{\mathrm{F}}^2, \tag{9}$$

where $f_\ell(\mathsf{X})$ is a matrix with the feature maps at depth ℓ of a multi-layer CNN, $\mathcal{L}_s$ and $\mathcal{L}_t$ are two subsets of depths, and $\|\cdot\|_{\mathrm{F}}$ denotes Frobenius norm. Gatys *et al.* used the very deep VGG-19 network, pre-trained for image classification [21]. The first term encourages the solution X^* to have the style of the painting X_s by matching the Gram matrices of the feature maps, such statistics capturing texture patterns at different scales. The second term ensures that the main structures (the "content") of the original photograph X_t, as captured in feature maps, are preserved in X^*. Note that all the spatial information is lost in the first term that encodes the style, while it is still present in the second term. It was proved shortly after that similar results can be obtained using a deep neural network with all the filter coefficients chosen randomly [11]. Let us also mention that [23] showed that texture synthesis, *i.e.*, when only the first term in (9) is involved, can be done using *multiple (8) one-layer* CNNs with random filters giving each $m = 1024$ feature maps.

We show now that our graph-based CNNs allow us to revisit neural style transfer. We use only *two one-layer* graph CNNs with *random filters* giving each only 50 feature maps. This is a much "lighter" network than the ones used in the literature. The first network, denoted f_1, uses local convolutions (Section 2.2.2) and the second, denoted f_2, uses non-local convolutions (Section 2.2.3) on a graph that captures the structure of the photograph X_t to be preserved. Both f_1 and f_2 have the form (7) with $m_0 = 3$ for the three *Lab* channels of color images and $m_1 = 50$. We also choose $d = 25$ and ReLU for the non-linearity in both cases. The $25 \times 3 \times 50 \times 2 = 7500$ coefficients of the filters $\boldsymbol{h}_j^\ell$ and the $50 \times 2 = 100$ biases b^ℓ in (7) are chosen randomly using independent draws from the standard Gaussian distribution.

The graph in the second CNN f_2 is constructed as follows. For an image of interest X, we construct a feature vector $\boldsymbol{f}_i \in \mathbb{R}^{29}$ at each pixel i of the image by extracting all the pixels' *Lab* values in the neighbourhood of size 3×3 around i as well as the absolute 2D coordinates of the pixel. We then search the $d = 25$ nearest neighbours to $\boldsymbol{f}_i$ in the set $\{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n\}$ using the Euclidean distance. Let $\mathcal{D} = \{\|\boldsymbol{f}_i - \boldsymbol{f}_j\|_2\}_{ij}$ be the set of all distances between each $\boldsymbol{f}_i$ and its nearest neighbours. We have $|\mathcal{D}| = 25n$. To avoid that some pixels are too weakly connected to others, which then produces artefacts in the final images, we compute the 80^{th} percentile of the values in $\mathcal{D}$ and saturates all the distances above this percentile to this value. The weights of the adjacency matrix W then satisfy

$$\mathsf{W}_{ij} = \exp\left(-\|\boldsymbol{f}_i - \boldsymbol{f}_j\|_2^2 / \sigma^2\right), \ \forall (i,j) \in \mathcal{E}, \tag{10}$$

with σ equal to the 75^{th} percentile of $\mathcal{D}$.

We capture the style of the painting X_s by computing the Gram matrices

$$\mathsf{G}_1 = f_1(\mathsf{X}_s)^\intercal f_1(\mathsf{X}_s) \text{ and } \mathsf{G}_2 = f_2(\mathsf{X}_s)^\intercal f_2(\mathsf{X}_s), \tag{11}$$

where the non-local convolution in f_2 is computed using the graph constructed on X_s. The matrix G_1 captures local statistics while G_2 captures non-local statistics. To give the style of X_s to X_t, we now compute a new graph on X_t. We then compute an image $\mathsf{X}^* \in \mathbb{R}^{n\times 3}$ by

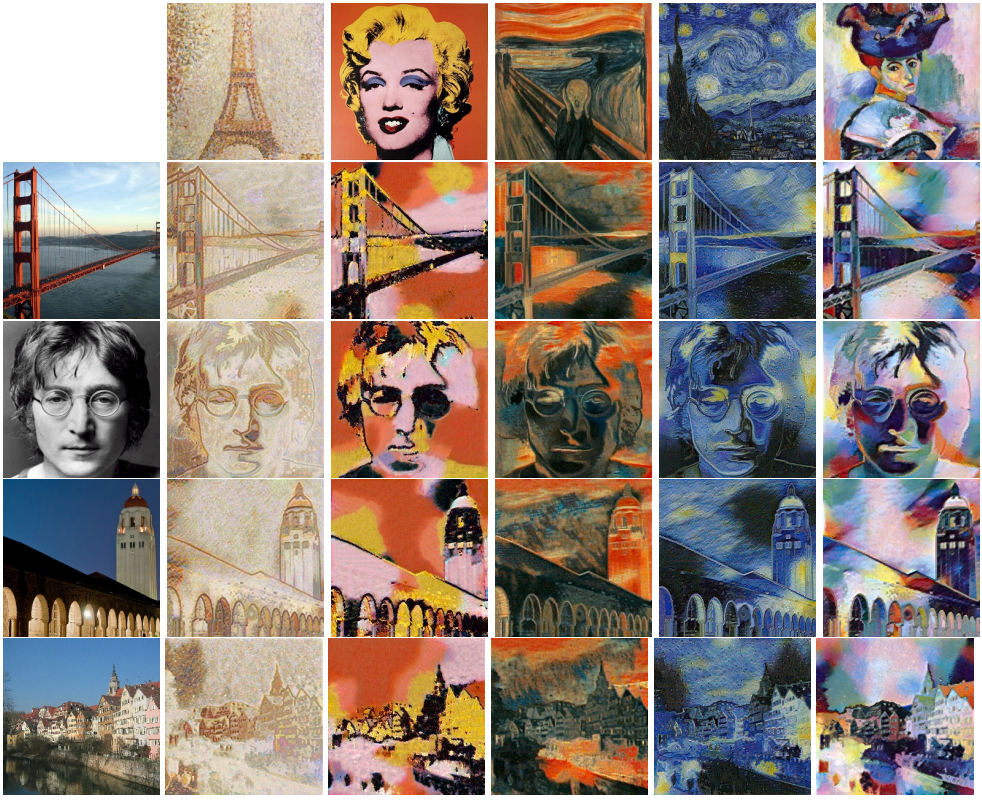

Figure 1: Examples of style transfer results obtained where the photograph (left) is transformed to have the style of a given painting (top). Using the proposed graph CNN framework, only two one-layer random CNNs are required to extract matched statistics.

solving

$$\min_{\mathsf{X}\in\mathbb{R}^{n\times 3}} \gamma_1 \left\| f_1(\mathsf{X})^\intercal f_1(\mathsf{X}) - \mathsf{G}_1 \right\|_{\mathrm{F}}^2 + \gamma_2 \left\| f_2(\mathsf{X})^\intercal f_2(\mathsf{X}) - \mathsf{G}_2 \right\|_{\mathrm{F}}^2 + \gamma_3 \left\| \mathsf{X} \right\|_{\mathrm{TV}}, \tag{12}$$

where f_2 uses, this time, *the graph constructed on* X_t for the non-local convolutions, $\|\cdot\|_{\mathrm{TV}}$ is the Total Variation norm, and $\gamma_1, \gamma_2, \gamma_3 > 0$. Note that unlike in (9), we do not try to match feature maps but only Gram matrices. Yet the final image X^* retains the structure of X_t thanks to the non-local convolution in f_2.

In practice, we minimise (12) using the L-BFGS algorithm starting from a random initialisation of X. The parameters γ_1, γ_2 are computed so that the gradient coming from the term they respectively influence has a maximum amplitude of 1 at the first iteration of the algorithm. We set $\gamma_3 = 0.01n$. All images used in the experiments have size $n = 256 \times 256$. However, we do not solve (12) directly at this resolution, but in a coarse-to-fine scheme instead: We start by downsampling all images at $n = 32 \times 32$ pixels; Solve (12) at this resolution; Upscale the solution at 64×64 pixels; Restart the same process at this new resolution using the up-scaled image as initialisation; Repeat this process until the final resolution is reached.

We present some results obtained with our graph-based method in Fig. 1. One can notice

Figure 2: Examples of style transfer results obtained with our graph CNN method using *only non-local convolutions* where the photograph (left) is transformed to have the style of a given painting (top).

that the main structure of the photograph X_t perfectly appears in X^* thanks to the presence of the structure-preserving non-local convolutions in f_2. The style is also well transferred thanks to the matching of the Gram matrices G_1 and G_2. For comparison with another method, one can refer to [9] where results for some pairs of photograph-painting in Fig.1 are also presented. It is not possible to do a quantitative comparison between both methods. Nevertheless, we notice that the method presented in [9] distorts more the edges present in the original photograph than our method does. We also remark that some large scale style characteristics are better transferred in [9] than with our method. We believe that using deeper graph CNNs can help in improving the transfer of large scale style characteristics.

To highlight the role of the graph CNN with non-local convolutions, we repeat exactly the same experiments but using only the non-local graph CNN, *i.e.*, we do not use the regular CNN with local convolutions – the TV regularisation is still present. Fig. 2 shows results for one photograph and different paintings. First, we notice that the main structures of the photograph are well preserved thanks to the graph CNN. Second, the colors of the painting and the relative arrangement of the colors are well transferred. However, we are not able to transfer finer style details like brush strokes. On the contrary, the local CNN is able to capture these finer details which appear in the results with the complete method.

Let us highlight that this experiment already shows that our graph CNN framework can adapt to many changes in the graph structure. Indeed, the graph used in f_2 was built from the painting when computing G_2 while it is built from the photograph when computing X^*.

4 Conclusion

We proposed a graph CNN framework that allows us to unify local and non-local processing of signals on graphs, and showed how to use this framework to perform style transfer. The results already suggest that the proposed convolution adapts correctly to changes in the input graph. Additional experiments in the supplementary material [18] demonstrate the versatility of our framework on other kinds of signals and tasks. Beyond signal processing, we believe that some of the tools presented here can be useful to other applications involving time-varying graph structures, such as in social networks.

References

[1] J. Atwood and D. Towsley. Diffusion-convolutional neural networks. In *NIPS*, 2016.

[2] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: Going beyond Euclidean data. *arXiv:1611.08097*, 2016.

[3] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. In *ICLR*, 2014.

[4] A. Buades, B. Coll, and J-M Morel. A non-local algorithm for image denoising. In *CVPR*, 2005.

[5] F.R.K. Chung. *Spectral graph theory*. Number 92. Amer Mathematical Society, 1997.

[6] A. Criminisi, P. Pérez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Trans. Image Process.*, 13(9):1200–1212, 2004.

[7] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 2016.

[8] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 3238(2):295–307, 2016.

[9] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.

[10] D. K. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. *Appl. Comput. Harmon. Anal.*, 30(2):129–150, 2011.

[11] K. He, Y. Wang, and J. Hopcroft. A powerful generative model using random weights for the deep image representation. In *NIPS*, 2016.

[12] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional network on graph-structured data. *arXiv:1506.05163*, 2015.

[13] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv:1609.02907*, 2016.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[15] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. *Int. Conf. on Learning Representations*, 2016.

[16] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model CNNs. *arXiv:1611.08402*, 2016.

[17] M. Niepert, M. Ahmed, and K. Kutzkov. Learning convolutional neural networks for graphs. In *ICML*, 2016.

[18] G Puy, S. Kitic, and P. Pérez. Unifying local and non-local signal processing with graph cnns. *arXiv:1702.07759*, 2017.

[19] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Trans. on Neural Networks*, 20(1):61–80, 2009.

[20] D.I. Shuman, S.K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[22] R. Talmon, I. Cohen, and S. Gannot. Transient noise reduction using nonlocal diffusion filters. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(6):1584–1599, 2011.

[23] I. Ustyuzhaninov, W. Brendel, L. A. Gatys, and M. Bethge. Texture synthesis using shallow convolutional networks with random filters. *arXiv:1606.00021*, 2016.

[24] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *ECCV*, 2016.