A Comprehensive Study of CayleyNets

Alice Valença De Lorenci
Institut Polytechnique de Paris
Palaiseau, France
alice.valenca@telecom-paris.fr

Julián Alvarez de Giorgi
Institut Polytechnique de Paris
Palaiseau, France
julian.alvarez@telecom-paris.fr

Abstract

The objective of this report was to study the Graph Convolutional Neural Network with complex rational spectral filters proposed by Levie et al. [6], called CaleyNets. With that intent, we start by overviewing the underlying spectral graph theory, leading to the construction of general Spectral Graph Convolutional Neural Network models and of CayleyNets. Our objective was to achieve a clear and intuitive description of this model and its properties. Furthermore we evaluate the CayleyNets through several experiments on the CORA dataset, using our own Python implementation of the model.

CCS Concepts: • **Computing methodologies** → **Neural networks**; *Spectral methods.*

Keywords: Convolutional Graph Neural Networks; Geometrical Deep Learning; Cayley Filters; Spectral Graph Theory

1 Introduction

In recent years, Deep Learning models have enjoyed increasing success in dealing with high dimensional Euclidian data. Models such as Convolutional Neural Networks (CNNs) are able to efficiently handle images, which are inherently high dimensional, by exploring underlying structural characteristics of the data, in this case, the translational invariance of signal classes.

Going beyond Euclidean data, graphs arise as a natural representation for a variety of data, such as social networks, molecular structures and anatomical surfaces. The power of graphs can be geometrically understood by interpreting them as metric spaces that are defined locally. This geometric interpretation promptly raises the question of whether it is possible to generalize CNNs to signals defined over graphs; or, more generally, whether it is possible to generalize Deep Learning to non-Euclidean data. Indeed this is the question behind the growing field of Geometrical Deep Learning [2]. One of the current approaches of this field to dealing with signals defined over graphs is that of generalizing convolutions by means of the spectrum of the graph Laplacian [3].

In this report, we will analyze a type of Spectral Graph Convolutional Neural Networks introduced by [6], called CayleyNets. First, in Section 2, we overview the main mathematical aspects of spectral graph theory; in Section 3 we introduce CayleyNets; finally, in Section 4 we propose a series of experiments to evaluate the performance of the model and illustrate its properties.

2 Background

2.1 Spectral graph theory

Consider an undirected graph $\mathcal{G}(V, E, \mathbf{W})$, where $V = \{1, \dots, n\}$ is the set of n nodes, E is the set of edges and $\mathbf{W} = \left(w_{ij}\right)_{ij} \in \mathbb{R}^{n\times n}$ is the adjacency matrix, defined as:

$$\begin{aligned} w_{ij} &= 0, \quad \text{if } (i,j) \in E \\ w_{ij} &> 0, \quad \text{if } (i,j) \notin E \end{aligned} \tag{1}$$

Since the graph is undirected, the adjacency matrix is symmetric. From the adjacency matrix, the degree matrix $\mathbf{D} = \operatorname{diag}\left(\sum_{j\neq i} w_{ij}\right)_{i=1}^{n}$ is defined, it is a diagonal matrix whose entries correspond to the degrees of the nodes.

The Laplacian matrix of the graph is defined as $\mathbf{\Delta} = \mathbf{D} - \mathbf{W} \in \mathbb{R}^{n\times n}$, and it can be written as $\mathbf{\Delta} = \nabla^T\nabla$, ∇ being the incidence matrix of the graph, for some fixed arbitrary direction of the edges. From this construction of the Laplacian it is easily seen that it is a symmetric positive semi-definite matrix, $\mathbf{\Delta} \geq 0$. Furthermore, one can define the normalized Laplacian as $\mathbf{\Delta} = \mathbf{D}^{-1/2}\mathbf{\Delta}\mathbf{D}^{-1/2}$.

Since the Laplacian matrix is real and symmetric, it has a spectral decomposition $\mathbf{\Delta} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T$. $\mathbf{\Phi} = \left(\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_n\right)$ with $\boldsymbol{\phi}_i \in \mathbb{R}^n$, $i = 1, \dots, n$, is the matrix whose columns are the right eigenvectors of $\mathbf{\Delta}$, and $\mathbf{\Lambda} = \operatorname{diag}\left(\lambda_i\right)_{i=1}^{n}$ is the diagonal matrix of eigenvalues, such that, $0 = \lambda_1 \leq \cdots \leq \lambda_n$ [1].

The spectral decomposition of the Laplacian matrix is used to define the graph Fourier transform, in analogous fashion to the classical Fourier transform [4]. If we let $\mathbf{f} = (f_1, \dots, f_n)^T \in \mathbb{R}^n$ be a signal defined on the nodes of the graph, its graph Fourier transform is $\hat{\mathbf{f}} = \mathbf{\Phi}^T\mathbf{f} = \left(\hat{f}_1, \dots, \hat{f}_n\right)^T$. Correspondingly, the inverse graph Fourier transform is $\mathbf{f} = \mathbf{\Phi}\hat{\mathbf{f}}$, i.e., any signal on the graph can be expressed as:

$$\mathbf{f} = \hat{f}_1\boldsymbol{\phi}_1 + \cdots + \hat{f}_n\boldsymbol{\phi}_n \tag{2}$$

In fact, the eigenvalues λ_i are a measure of the smoothness of the associated eigenvector $\boldsymbol{\phi}_i$, interpreted as a signal over the graph. The smaller the eigenvalue, the smoother the corresponding eigenvector. By extension, the Fourier coefficients $\left(\hat{f}_1, \dots, \hat{f}_n\right)^T$ of the signal $\mathbf{f}$ on the $\left\{\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_n\right\}$ basis reflect the smoothness of the signal. Those considerations give rise to the interpretation of the eigenvalues of the Laplacian as graph frequencies and of $\left\{\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_n\right\}$ as

the graph Fourier basis. It is important to highlight that the graph Fourier transform is defined relative to the graph $\mathcal{G}$. For a detailed description of the connection between the spectral decomposition of the Laplacian and the smoothness of signals defined over graphs the reader may refer to [3].

Having defined the graph Fourier transform, it is possible to define the spectral convolution of two given signals $\mathbf{f}, \mathbf{g}$ on the graph:

$$\mathbf{f} * \mathbf{g} \coloneqq \mathbf{\Phi}\left(\left(\mathbf{\Phi}^T\mathbf{g}\right) \cdot \left(\mathbf{\Phi}^T\mathbf{f}\right)\right) \tag{3}$$

where $\cdot$ denotes the element-wise multiplication. The spectral convolution can be equivalently represented as:

$$\mathbf{f} * \mathbf{g} = \mathbf{\Phi}\hat{\mathbf{G}}\mathbf{\Phi}^T\mathbf{f} \tag{4}$$

$$\mathbf{f} * \mathbf{g} = \mathbf{G}\mathbf{f} \tag{5}$$

where $\hat{\mathbf{G}} \coloneqq \operatorname{diag}(\hat{g}_1, \ldots, \hat{g}_n)$ and $\mathbf{G} \coloneqq \mathbf{\Phi}\hat{\mathbf{G}}\mathbf{\Phi}^T$. Those equivalent representations will be useful in the sequence. We say that the signal $\mathbf{f}$ is filtered by $\mathbf{g}$.

In order to build intuition about spectral operations on graphs, it is interesting to analyze Eq. 4. Consider the following manipulations of Eq. 4:

$$\begin{aligned}\mathbf{f} * \mathbf{g} &= \mathbf{\Phi}\hat{\mathbf{G}}\left(\mathbf{\Phi}^T\mathbf{f}\right)\\ &= \mathbf{\Phi}\hat{\mathbf{G}}\left(\hat{f}_1, \ldots, \hat{f}_n\right)^T\\ &= \mathbf{\Phi}\left(\hat{g}_1\hat{f}_1, \ldots, \hat{g}_n\hat{f}_n\right)^T\\ &= \hat{g}_1\hat{f}_1\boldsymbol{\phi}_1 + \cdots + \hat{g}_n\hat{f}_n\boldsymbol{\phi}_n\end{aligned} \tag{6}$$

From Eq. 6 it can be seen that a diagonal operator on the Fourier space modulates the smoothness of the signal.

2.2 Spectral Graph Convolutional Neural Networks

Graph spectral convolutions were first introduced as a means to define CNNs on graphs by [3].

Consider a feature matrix $\mathbf{F}^{\text{in}} = \left(\mathbf{f}_1^{\text{in}}, \ldots, \mathbf{f}_p^{\text{in}}\right) \in \mathbb{R}^{n\times p}$, where $\mathbf{f}_j \in \mathbb{R}^n$, $j = 1, \ldots, p$, denotes a signal on the graph, and the i^{th} row of $\mathbf{F}^{\text{in}}$ represents the p-dimensional feature vector of node i, $i = 1, \ldots, n$.

Then, a convolutional layer transforms an input feature matrix $\mathbf{F}^{\text{in}} \in \mathbb{R}^{n\times p}$ into an output feature matrix $\mathbf{F}^{\text{out}} = \left(\mathbf{f}_1^{\text{out}}, \ldots, \mathbf{f}_{p'}^{\text{out}}\right) \in \mathbb{R}^{n\times p'}$ according to:

$$\mathbf{f}_{j'}^{\text{out}} = \sigma\left(\sum_{j=1}^{p} \mathbf{\Phi}_k\hat{\mathbf{G}}^{(j',j)}\mathbf{\Phi}_k^T\mathbf{f}_j^{\text{in}}\right), \quad j' = 1, \ldots, p' \tag{7}$$

On Eq. 7, σ is a non-linearity, $\mathbf{\Phi}_k = \left(\boldsymbol{\phi}_1 \ldots, \boldsymbol{\phi}_k\right) \in \mathbf{R}^{n\times k}$ is the matrix with the first k eigenvectors of the Laplacian, and $\hat{\mathbf{G}}^{(j',j)} \in \mathbb{R}^{k\times k}$, $j = 1, \ldots, p$, $j' = 1, \ldots, p'$, are diagonal matrices of learnable parameters. Therefore, such a spectral convolutional layer has $pp'k$ trainable parameters.

On Eq. 7, $\hat{\mathbf{G}}^{(j',j)}$ can be interpreted as a learnable filter in the frequency domain. Notice that this definition of a convolutional layer is based on the representation of the spectral convolution given by Eq. 4.

One drawback of the layer described by Eq. 7 is that it relies on dense matrix multiplications by $\mathbf{\Phi}_k$ and $\mathbf{\Phi}_k^T$, incurring a cost of $O(n^2)$, assuming that $k = O(n)$.

An alternative approach is to express the filtering of the signal $\mathbf{f}$ as in Eq. 5, i.e., as $\mathbf{G}\mathbf{f}$, where $\mathbf{G}$ is such that it can be written as:

$$\mathbf{G} = g(\mathbf{\Delta}) = \mathbf{\Phi}g(\mathbf{\Lambda})\mathbf{\Phi}^T = \mathbf{\Phi}\operatorname{diag}(g(\lambda_1), \ldots, g(\lambda_n))\mathbf{\Phi}^T \tag{8}$$

Depending on the choice of g, the matrix G may be sparse. This characteristic can be exploited to reduce the computational complexity of the layer. Furthermore, this approach does not require the explicit computation of the Laplacian eigenvectors.

A convolutional layer is then defined as:

$$\mathbf{f}_{j'}^{\text{out}} = \sigma\left(\sum_{j=1}^{p} \mathbf{G}^{(j',j)}\mathbf{f}_j^{\text{in}}\right), \quad j' = 1, \ldots, p' \tag{9}$$

3 CayleyNets

CayleyNets are based on the filtering approach described by Eq. 8 and consist on convolutional layers as described by Eq. 9, with $\mathbf{G}$ a Cayley filter.

In order to define such a filter, one must first consider the Cayley transform:

$$C : \mathbb{R} \to e^{i\mathbb{R}}\backslash\{1\}, \quad C(x) = \frac{x - i}{x + i} \tag{10}$$

The Cayley transform defines a smooth bijection between $\mathbb{R}$ and the complex unit circle, as illustrated on Fig. 1. This definition is extended to nonsingular real square matrices as:

$$C(\mathbf{X}) = (\mathbf{X} - i\mathbf{I})(\mathbf{X} + i\mathbf{I})^{-1} \tag{11}$$

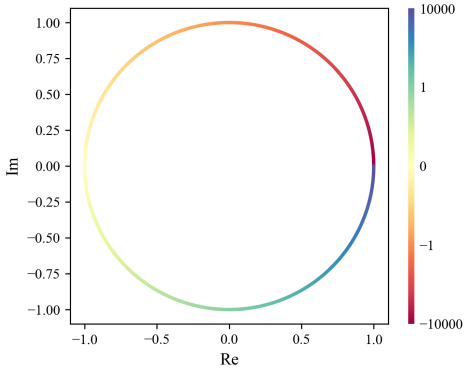


Figure 1. Plot of the Cayley transform $C(x)$, $x \in \mathbb{R}$, of the real line on the complex plane. The value of x can be identified via the color bar.

Applying the Cayley transform to the Laplacian $\mathbf{\Delta}$ scaled by a spectral zoom parameter $h > 0$ and using the spectral

decomposition of $\mathbf{\Delta}$, the following equality is obtained:

$$\begin{aligned} C(h\mathbf{\Delta}) &= (h\mathbf{\Delta} - i\mathbf{I})\,(h\mathbf{\Delta} + i\mathbf{I})^{-1} \\ &= \mathbf{\Phi}\,(h\mathbf{\Lambda} - i\mathbf{I})\,(h\mathbf{\Lambda} + i\mathbf{I})^{-1}\,\mathbf{\Phi}^T = \mathbf{\Phi} C(h\mathbf{\Lambda})\mathbf{\Phi}^T \end{aligned} \tag{12}$$

Eq. 12 will guarantee that the Cayley filter has the format of Eq. 8.

Since $\mathbf{\Lambda}$ is a diagonal matrix, $C(h\mathbf{\Lambda}) = \operatorname{diag}\left(C(h\lambda_i)\right)_{i=1}^n$. Therefore, it follows from Eq. 12 that the complex matrix $C(h\mathbf{\Delta})$ has its spectrum on $e^{i\mathbb{R}}$, therefore it is unitary. Furthermore, since $\mathbf{\Delta} \geq 0$ its eigenvalues are non-negative, thus the spectrum of $C(h\mathbf{\Delta})$ lies on the lower-half of the complex unitary circle, as can be seen from Fig. 1.

Having defined the Cayley transform, the Cayley polynomial of order r is defined as:

$$g_{\mathbf{c},h}(\lambda) = c_0 + 2\,\mathrm{Re}\left\{\sum_{k=1}^{r} c_k C(h\lambda)^k\right\} \tag{13}$$

where $\mathbf{c} = (c_0, \ldots, c_r) \in \mathbb{R} \times \mathbb{C}^r$ and $h > 0$. This definition is naturally extended to nonsingular real square matrices. In particular, the Cayley filter is the Cayley polynomial applied to the Laplacian:

$$\mathbf{G}_{\mathbf{c},h} = g_{\mathbf{c},h}(\mathbf{\Delta}) = c_0 + 2\,\mathrm{Re}\left\{\sum_{k=1}^{r} c_k C(h\mathbf{\Delta})^k\right\} \tag{14}$$

Eq. 12 guarantees that the Cayley filter is of the form of Eq. 8. A set $\left(\mathbf{G}^{(j,j')}\right)_{j,j'=1}^{p,p'}$ of Cayley filters then defines a convolutional layer as in Eq. 9, where the parameters $\mathbf{c}^{(j,j')}$, associated to each filter $\mathbf{G}^{(j,j')}$, and the parameter h, associated to the layer, are optimized during training. Such a layer has $pp'(2r+1)+1$ trainable real parameters, considering that each complex parameter corresponds to two real parameters.

3.1 Properties of Cayley filters

3.1.1 Spectral zoom. It is interesting to analyze the role of the spectral zoom parameter h in the Cayley filter.

A characteristic of the Cayley transform that will be relevant for the analysis is that the mapping of the interval $[0, 1]$ is spread out on the lower left quadrant of the unit circle. On the other hand, $[1, +\infty)$ is mapped to the lower right quadrant, resulting in the concentration close to the point $(1, 0)$ of the mapping of large values. This property can be deduced from Fig. 1.

Recalling Eq. 12, where $C(h\mathbf{\Lambda}) = \operatorname{diag}\left(C(h\lambda_i)\right)_{i=1}^n$, it is seen that h has a direct influence on the spectrum of $C(h\mathbf{\Delta})$ and amounts to "zooming" into different parts of the spectrum. Small values of h will better spread apart the larger eigenvalues of $\mathbf{\Delta}$ that are crushed by the Cayley transform, whereas large values of h will zoom into the smaller eigenvalues.

This behavior is illustrated for a synthetic 15-communities graph, represented on Fig. 2. On Fig. 3, the eigenvalues of the graph Laplacian are plotted, with the 15 smallest eigenvalues highlighted in red, the eigenvalues of the Cayley transform of the scaled graph Laplacian are plotted as well for different values of h.

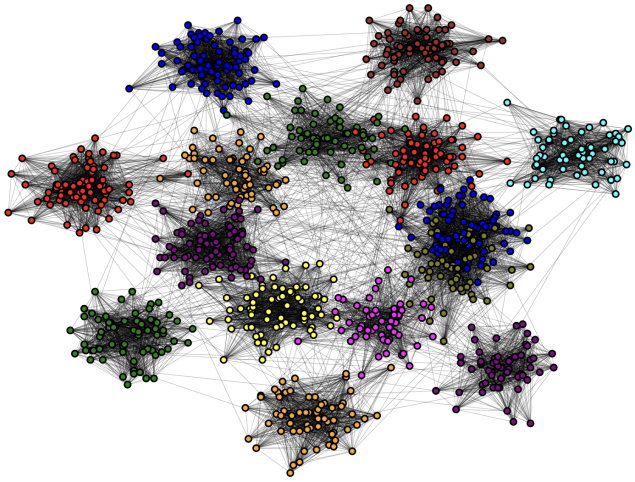
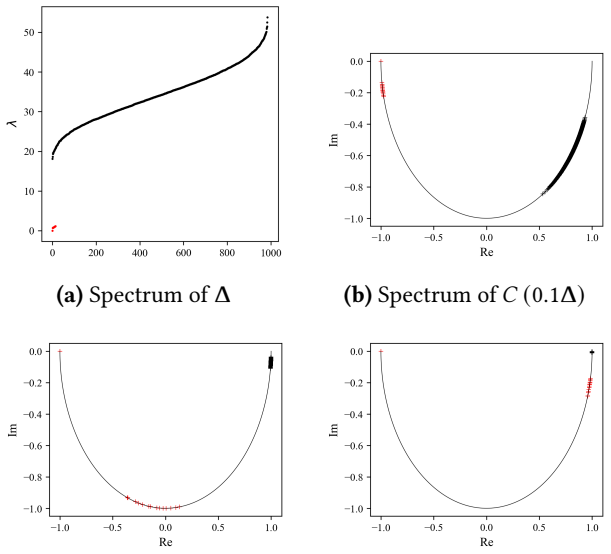


Figure 2. Synthetic 15-communities graph. The graph was created by assigning each node to a community uniformly at random; then an edge was added for each pair of nodes with probability $p = 0.5$ for nodes belonging to the same community and probability $q = 10^{-3}$ for nodes belonging to different communities.

3.1.2 Localization. Another interesting property of Cayley filters is that they are not strictly truncated to r-hop



(a) Spectrum of $\mathbf{\Delta}$ **(b)** Spectrum of $C(0.1\mathbf{\Delta})$

(c) Spectrum of $C(\mathbf{\Delta})$, $h = 1$ **(d)** Spectrum of $C(10\mathbf{\Delta})$

Figure 3. (a) Eigenvalues of the unnormalized Laplacian $\mathbf{\Delta}$ of the 15-communities graph of Fig. 2. (b,c,d) Eigenvalues of $C(h\mathbf{\Delta})$ for $h = 0.1, 1$ and 10, respectively. The 15 smallest eigenvalues are marked in red.

neighborhoods, unlike polynomial filters. This gives rise to an alternative notion of localization related to the exponential decay of $\mathbf{G}\boldsymbol{\delta}_m$, where $\boldsymbol{\delta}_m$ denotes a delta-function on the graph. In particular, the following result was proven by the authors in the article [6]:

$$\|\mathbf{G}\delta_m|_{\mathcal{N}^c_{k,m}}\|_2 \leq 4M\mathcal{K}^{k/r} \tag{15}$$

where:

- $\mathcal{N}_{k,m}$ is the k-hop neighborhood of m.
- $M = \sqrt{n}\sum_{j=1}^{r} j|c_j|$
- $\mathcal{K} = \frac{hd}{\sqrt{h^2d^2+1}}$ and $d = max_j\{d_{i,j}\}$, i.e. maximum node degree.

This way, while the support is the entire graph we can still define a notion of localization, which is particularly pertinent considering that having well-localized filters on the space domain is an important aspect of convolutional kernels.

3.1.3 Cayley filters are complete on the space of spectral filters. Another of the motivating factors for the use of Cayley filters is that any spectral filter can be formulated as one. This can be directly seen through a reformulation of Eq. 14.

We will use the fact that $2\,\mathrm{Re}\{C(h\Delta)\} = C(h\Delta)+\overline{C(h\Delta)}$, indeed this is true for any complex valued matrix. However, the spectrum of $C(h\Delta)$ lies in $e^{i\mathbb{R}}$, therefore $\overline{C(h\Delta)} = C(h\Delta)^{-1}$. Thus Eq. 14 can be reformulated as the following Laurent polynomial:

$$\begin{aligned}\mathbf{G}_{\mathbf{c},h} = g_{\mathbf{c},h}(\Delta) &= c_0 + \sum_{k=1}^{r} c_k C(h\Delta)^k + \overline{c_k}\,\overline{C(h\Delta)}^k \\ &= c_0 + \sum_{k=1}^{r} c_k C(h\Delta)^k + \overline{c_k} C(h\Delta)^{-k}\end{aligned} \tag{16}$$

Let us analyze each term in the summation of Eq. 16 in terms of the spectral decomposition of Δ:

$$\begin{aligned}p_k(h\Delta) &:= c_k C(h\Delta)^k + \overline{c_k} C(h\Delta)^{-k} \\ &= \Phi\left(c_k C(h\Lambda)^k + \overline{c_k} C(h\Lambda)^{-k}\right)\Phi^T\end{aligned} \tag{17}$$

In Eq. 17, $C(h\Lambda)^k = \mathrm{diag}\left(C(h\lambda_1)^k, \ldots, C(h\lambda_n)^k\right)$ and $C(h\lambda_p)^k \in e^{i\mathbb{R}}$, $p = 1, \ldots, n$. Therefore, the power operator can be interpreted as a multiplication by a pure harmonic in the frequency domain.

With that interpretation in mind, let $\omega_p \in [0, 2\pi)$ be the angular frequency associated to the eigenvalue λ_p by the Cayley transform, i.e, $C(h\lambda_p) = e^{i\omega_p}$. Then, from Eq. 17, we have:

$$\begin{aligned}p_k(h\lambda_p) &= c_k e^{i\omega_p k} + \overline{c_k} e^{-i\omega_p k} \\ &= 2\,\mathrm{Re}\{c_k\}cos(\omega_p k) - 2\,\mathrm{Im}\{c_k\}sin(\omega_p k)\end{aligned} \tag{18}$$

Therefore we conclude that $\mathbf{G}_{\mathbf{c},\mathbf{h}}$ is a real-valued trigonometric polynomial. This implies that any spectral filter $g(\Delta)$ can be written as a Cayley filter, since the values $g(\lambda_1)$, ..., $g(\lambda_n)$ that define it can be interpolated by a trigonometric polynomial. The implications of this property of Cayley filters will be analysed on Section 4.

3.2 Computational method

CayleyNets consist in convolutional layers that transform an input feature matrix $\mathbf{F}^{\text{in}} \in \mathbb{R}^{n\times p}$ into an output feature matrix $\mathbf{F}^{\text{out}} \in \mathbb{R}^{n\times p'}$ according to Eq. 9, where $\mathbf{G}^{(j',j)}$ is a Cayley filter, defined in Eq. 14.

Therefore, the forward pass of the network relies on computing $C(h\Delta)^k\,\mathbf{f}$, $k = 1, \ldots, r$. This can be done in an iterative fashion:

$$\begin{aligned}&\mathbf{y}_0 := \mathbf{f} \\ &\mathbf{y}_k := C(h\Delta)^k\,\mathbf{f} = C(h\Delta)\,\mathbf{y}_{k-1}, \quad k = 1, \ldots, r\end{aligned} \tag{19}$$

Racalling the definition of the Cayley transform, $C(h\Delta) = (h\Delta - i\mathbf{I})(h\Delta + i\mathbf{I})^{-1}$, therefore in order to avoid computing inverses, which has a computational complexity of $O(n^3)$, $\mathbf{y}_0, \ldots, \mathbf{y}_r$ can be expressed instead as solutions to the following systems of linear equations:

$$\begin{aligned}&\mathbf{y}_0 = \mathbf{f} \\ &(h\Delta + i\mathbf{I})\,\mathbf{y}_k = (h\Delta - i\mathbf{I})\,\mathbf{y}_{k-1}, \quad k = 1, \ldots, r\end{aligned} \tag{20}$$

Jacobi's iterative method can then be used to provide approximate solutions to the systems of linear equations of Eq. 20. Running the method with K iterations, the computational cost of computing each $\boldsymbol{y}_j$ is $O(Kn^2)$. However, if the sparsity of the Laplacian is exploited, the computational cost of the method is reduced to $O(Kn)$, assuming that each row of the Laplacian has $O(1)$ non zero entries. Therefore the overall computational cost of a layer is $O(Krn)$.

In practice, an approximate solution is sufficient because the approximation error of Jacobi's method can be compensated by the learned parameters.

4 Experiments

The code associated to the original CayleyNets article [6] is incompatible with current versions of Python frameworks for Machine Learning and was proven difficult to adapt. Therefore, we proposed our own implementation of CayleyNets using Pytorch Geometric [1]. Even though our implementation strictly follows the article's formulation of CayleyNets, our results do not exactly match those of [6]. Due to this discrepancy, considerable effort was dedicated to testing the different components of the code and implementing possible variations, however no implementation errors were identified. This difficulty highlights the importance of making available a well documented source code, in order to guarantee the reproducibility of a scientific publication.

In this section, the CayleyNets will be evaluated on a node classification task, using the CORA dataset. All experiments were conducted on a machine with a 3.00GHz Intel Xeon

[1]The code is available at https://github.com/AliceDeLorenci/CayleyNets

Gold 6136 CPU, 187GB of RAM, and a NVIDIA Tesla V100 GPU with 16GB of RAM.

4.1 CORA dataset

The CORA dataset [7] consists in a graph of citations between 2708 scientific publications, with 5429 edges. Each node is associated to a 1433 dimensional binary feature vector, that indicates the presence of the corresponding word on the scientific publication; and is classified into one of seven classes, corresponding to the category of the paper. This dataset is a common benchmark in the Graph Machine Learning community, therefore we chose to use it to evaluate the CayleyNets. An alternative formulation of Spectral Graph Neural Network known as ChebNets was used as a baseline, this model is described on Appendix A.

The problem consists in a node classification task in which the labels are given for only a subset of the nodes (training data), and the model must predict the labels for the remaining nodes (test and validation data). For the numerical experiments the CORA dataset was split into 1708 training nodes, 500 validation nodes and 500 test nodes.

4.2 Impact of the polynomial order

First of all, we analyzed the impact of the polynomial order r of the spectral filter on the performance of the network. This was done by measuring the test accuracy for $r = 1, 3, 5, 7, 9$. For that, the network architecture proposed by [6] was used, i.e., CayleyNets and ChebNets with one hidden layer with 32 features; the hidden layer uses ReLu non-linearity and the ouput layer, softmax non-linearity. For both networks the normalized Laplacian was used for the spectral filters. The models were trained for 200 epochs using the Adam optimization method, with weight decay $\lambda = 5 \cdot 10^{-4}$ and learning rate $\gamma = 5 \cdot 10^{-3}$, as well as dropout regularization with 0.5 probability. The results are summarized in Fig. 4.

Analyzing Figure 4 we observe that ChebNets consistently outperform CayleyNets, for all tested polynomial degrees. Furthermore, we observe a strong degradation of the test accuracy of CayleyNets for increasing polynomial order r. This might be an indicator of overfitting. Indeed, the number of parameters for a CayleyNet layer is $pp'(2r+1)+1 = O(pp'r)$, where p is the number of input features and p' the number of output features. Therefore the resulting total number of parameters for a CayleyNet with one 32-nodes hidden layer is of the order of 70K, 160K, 250K, 350K and 440K parameters for $r = 1, 3, 5, 7, 9$, respectively. In order to verify the overfitting hypothesis, we plotted the train and validation accuracy over the training epochs for the CayleyNet, with different polynomial orders. The results are illustrated on Fig. 5, and confirm that the network has overfitted to the trainning data.

In an attempt to avoid overfitting while preserving the expressive power of filters of higher polynomial order, we repeated the same experiment, but halving the number of hidden nodes of the CayleyNet. The results are also illustrated on Fig. 4. It can be seen that reducing the size of the hidden layer negatively impacted the performance of the network.

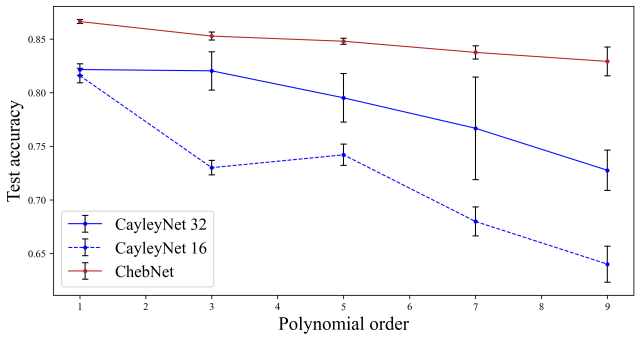


Figure 4. Evaluation of CayleyNets and ChebNets for a node classification task on the CORA dataset for different polynomial orders r ($r = 1, 3, 5, 7, 9$). Each experiment was repeated 10 times, the mean test accuracy is plotted in red for ChebNets with 32 hidden nodes, in solid blue line for CayleyNets with 32 hidden nodes and in dashed blue line for CayleNets with 16 hidden nodes. The corresponding standard deviations are displayed as error bars.

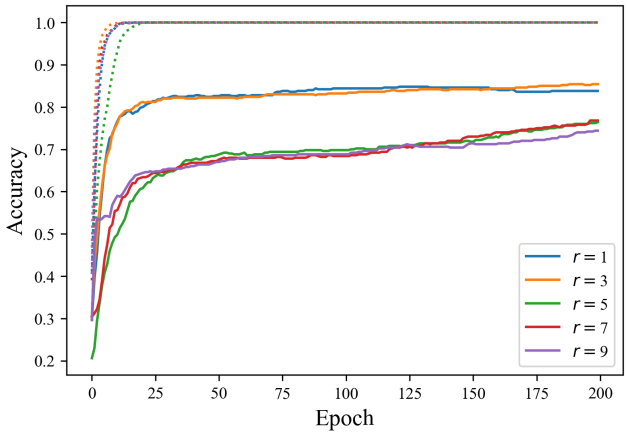


Figure 5. Train and validation accuracies over the training epochs for CayleyNets with 32 hidden nodes for different polynomial orders r ($r = 1, 3, 5, 7, 9$). The train and validation accuracies are plotted in dotted and solid lines, respectively, and the polynomial order can be identified via the color of the plots.

4.3 Comparison of different methods

On Section 4.2, CayleyNets were compared to ChebNets, an alternative Spectral Graph Convolutional Neural Network model. However, it is interesting to compare them as well to simpler Graph Neural Network (GNN) models, such as message passing GNNs.

Table 1. Test accuracy obtained with different methods on the CORA dataset.

Model	Order	Accuracy	#Parameters
CayleyNet	1	0.816 ± 0.006	138K
ChebyNet	1	0.872 ± 0.004	92K
GNN	-	0.812 ± 0.10	46K

In this context, we evaluated the performance of a standard message passing GNN with one hidden layer with 32 hidden nodes and ReLu activation function, followed by a fully connected layer with 7 output units and softmax activation, mimicking the architecture used for CayleyNets and ChebNets. The training setup is the same as that described on Section 4.2, in particular the same hyperparameters were used.

The experiment was repeated 10 times, and the mean test accuracy and standard deviation are displayed in Table 1. In this table we display as well the best mean test accuracy and standard deviation obtained on Section 4.2 for Cayleynets and ChebNets, i.e., using an architecture of 32 hidden nodes and polynomial order $r = 1$. It can be seen that the performance of CayleyNets is comparable to that of much simpler message passing networks.

4.4 Cayley filter response

Another aspect of CayleyNets that is interesting to analyze is the spectral response of the learned filters. Recalling the definition of Eq. 9 of a spectral convolutional layer and the expression of Eq. 14 for the Cayley filter, we see that each layer corresponds to pp' learned filters, where p denotes the number of input features and p' the number of output features.

In order to visualize the response of the filters, we retrieved the filter parameters from the trained weights of a CayleyNet trained on the CORA dataset. More specifically, the same experimental setup of Section 4.2 was used and the chosen architecture was a CayleyNet with a hidden layer of 32 nodes and polynomial order $r = 3$.

The spectral response of ten filters randomly chosen among all of the learned filters is plotted in Fig. 6. From the figure we observe that the network is able to learn a variety of spectral responses, spanning both narrow band and wide band filters, and focusing on different frequency bands. This variability is a result of the characteristics of Cayley filters discussed on Sections 3.1.1 and 3.1.3, i.e., the ability of focusing on different parts of the spectrum due to the spectral zoom parameter and the property that Cayley filters can approximate any spectral filter.

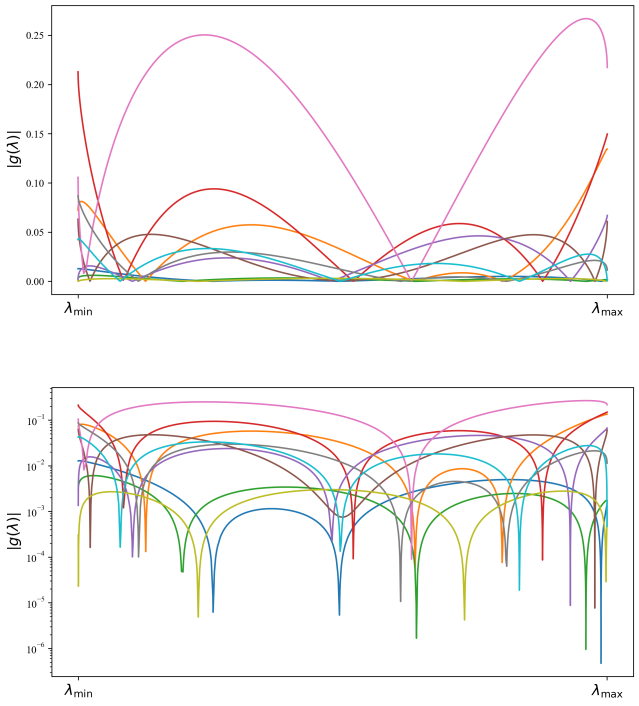


Figure 6. Responses of ten different filters learned by a CayleyNet, in linear (top) and log (bottom). The scaling of the abscissa was obtained by unrolling to the real line the Cayley transform of the Laplacian eigenvalues.

5 Conclusion

CayleyNets are a type of Spectral Convolutional Graph Neural Networks based on a sophisticated spectral filter formulation, with sound theoretical motivations. However, in spite of the theoretical properties of Cayley filters, in practice the performance of CayleyNets is inferior to that of simpler architectures, such as ChebNets, or even message passing Graph Neural Networks.

Therefore the applicability of CayleyNets is put into doubt, specially when one considers their difficult implementation and high computational cost. With respect to the first point, CayleyNets notably rely on complex parameters, which are not natively supported by Python Machine Learning frameworks, demanding ad hoc optimization of the code. Regarding the second point, the computational complexity of a CayleyNet is higher than that of an equivalent ChebNet by a factor of K, where K is the chosen number of Jacobi iterations.

However, we emphasize that our considerations regarding the performance of the networks are based on experiments carried out using the CORA dataset. A future directions of work could involve testing CayleyNets on different tasks and benchmarks, for instance, on graph classification tasks

instead of node classification ones, to determine the domain of applicability of those networks.

References

[1] Thomas Bonald. 2022–2023. Graph Mining. Course Notes. Course code: SD212.

[2] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. 2021. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges.* https://doi.org/10.48550/arXiv.2104.13478 arXiv:2104.13478 [cs.LG] Submitted on 27 Apr 2021 (v1), last revised 2 May 2021 (this version, v2).

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. *International Conference on Learning Representations (ICLR)* (April 2014).

[4] Xinye Chen. 2020. Understanding Spectral Graph Neural Network. *arXiv preprint arXiv:2012.06660* (2020). arXiv:2012.06660

[5] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. *CoRR* abs/1606.09375 (2016). arXiv:1606.09375 http://arxiv.org/abs/1606.09375

[6] Ron Levie, Federico Monti, Xavier Bresson, and Michael M. Bronstein. 2017. CayleyNets: Graph Convolutional Neural Networks with Complex Rational Spectral Filters. *CoRR* abs/1705.07664 (2017). arXiv:1705.07664 http://arxiv.org/abs/1705.07664

[7] Telecom Paris NETSET. 2023. *CORA.* https://netset.telecom-paris.fr/pages/cora.html

A ChebNets

ChebNets are a type of Spectral Graph Convolutional Neural Networks introduced by [5]. Analogously to CayleyNets, they adopt the approach of Eq. 9, however, the filter they propose is based on the Chebyshev polynomials instead of Cayley polynomials.

A Chebyshev polynomial has the form:

$$g_{\boldsymbol{\alpha}}(\tilde{\lambda}) = \sum_{k=0}^{r} \alpha_k T_k(\tilde{\lambda}) \tag{21}$$

On Eq. 21, $\boldsymbol{\alpha} \in \mathbb{R}^{r+1}$ are the trainable parameters and the functions $T_k : [-1, 1] \longrightarrow 1$ are recursively defined:

$$\begin{aligned} T_0(\tilde{\lambda}) &= 1 \\ T_1(\tilde{\lambda}) &= \tilde{\lambda} \\ T_k(\tilde{\lambda}) &= 2\tilde{\lambda}T_{k-1}(\tilde{\lambda}) - T_{k-2}(\tilde{\lambda}), \quad k > 1 \end{aligned} \tag{22}$$

Since the functions T_k are defined on the interval $[-1, 1]$, the Chebyshev polynomial must be evaluated on the rescaled Laplacian $\tilde{\boldsymbol{\Delta}} = 2\lambda_{\max}^{-1}\boldsymbol{\Delta} - \mathbf{I}$, whose eigenvalues lie on $[-1, 1]$. The resulting filter is then:

$$\mathbf{Gf} = g_{\boldsymbol{\alpha}}(\tilde{\boldsymbol{\Delta}})\mathbf{f} = \sum_{k=0}^{r} \alpha_k T_k(\tilde{\boldsymbol{\Delta}})\mathbf{f} \tag{23}$$

From the formulas for T_k, we see that $g_{\boldsymbol{\alpha}}(\tilde{\boldsymbol{\Delta}})$ is a polynomial of order r of the scaled Laplacian, which implies that the filter is an operator that affects only the r-hops neighborhood of a node.

Leveraging the recurrent formulation of T_k, the output of the filter can be computed by setting:

$$\begin{aligned} \tilde{\mathbf{f}}_0 &= \mathbf{f} \\ \tilde{\mathbf{f}}_1 &= \tilde{\Delta}\mathbf{f} \\ \tilde{\mathbf{f}}_k &= 2\tilde{\Delta}\tilde{\mathbf{f}}_{k-1} - \tilde{\mathbf{f}}_{k-2}, \quad k > 1 \end{aligned} \tag{24}$$

The computational complexity of the filter is thus $O(rn^2)$. However, exploiting the sparsity of the Laplacian the complexity can be reduced to $O(rn)$, if we assume that the nodes have $O(1)$ neighbors on average.

B Online Resources

The source code is available at https://github.com/AliceDeLorenci/CayleyNets.

Received 10 December 2023