Graph Based Convolutional Neural Network

Michael Edwards
Swansea University
Swansea, UK

Xianghua Xie
http://www.csvision.swan.ac.uk

Abstract

The benefit of localized features within the regular domain has given rise to the use of Convolutional Neural Networks (CNNs) in machine learning, with great proficiency in the image classification. The use of CNNs becomes problematic within the irregular spatial domain due to design and convolution of a kernel filter being non-trivial. One solution to this problem is to utilize graph signal processing techniques and the convolution theorem to perform convolutions on the graph of the irregular domain to obtain feature map responses to learnt filters. We propose graph convolution and pooling operators analogous to those in the regular domain. We also provide gradient calculations on the input data and spectral filters, which allow for the deep learning of an irregular spatial domain problem. Signal filters take the form of spectral multipliers, applying convolution in the graph spectral domain. Applying smooth multipliers results in localized convolutions in the spatial domain, with smoother multipliers providing sharper feature maps. Algebraic Multigrid is presented as a graph pooling method, reducing the resolution of the graph through agglomeration of nodes between layers of the network. Evaluation of performance on the MNIST digit classification problem in both the regular and irregular domain is presented, with comparison drawn to standard CNN. The proposed graph CNN provides a deep learning method for the irregular domains present in the machine learning community, obtaining 94.23% on the regular grid, and 94.96% on a spatially irregular subsampled MNIST.

1 Introduction

In recent years, the machine learning and pattern recognition community has seen a resurgence in the use of neural network and deep learning architecture for the understanding of classification problems. Standard fully connected neural networks have been utilized for domain problems within the feature space with great effect, from text document analysis to genome characterization [22]. The introduction of the CNN provided a method for identifying locally aggregated features by utilizing kernel filter convolutions across the spatial dimensions of the input to extract feature maps [10]. Applications of CNNs have shown strong levels of recognition in problems from face detection [11], digit classification [4], and classification on a large number of classes [15].

The core CNN concept introduces the hidden convolution and pooling layers to identify spatially localized features via a set of receptive fields in kernel form. The convolution operator takes an input and convolves kernel filters across the spatial domain of the data

Pages 114.1-114.11
DOI: https://dx.doi.org/10.5244/C.30.114

provided some stride and padding parameters, returning feature maps that represent response to the filters. Given a multi-channel input, a feature map is the summation of the convolutions with separate kernels for each input channel. In CNN architecture, the pooling operator is utilized to compress the resolution of each feature map in the spatial dimensions, leaving the number of feature maps unchanged. Applying a pooling operator across a feature map enables the algorithm to handle a growing number of feature maps and generalizes the feature maps by resolution reduction. Common pooling operations are that of taking the average and max of receptive cells over the input map [1].

Due to the usage of convolutions for the extraction of partitioning features, CNNs require an assumption that the topology of the input dimensions provides some spatially regular sense of locality. Convolution on the regular grid is well documented and present in a variety of CNN implementations [9, 20], however when moving to domains that are not supported by the regular low-dimensional grid, convolution becomes an issue. Many application domains utilize irregular feature spaces [13], and in such domains it may not be possible to define a spatial kernel filter or identify a method of translating such a kernel across spatial domain. Methods of handling such an irregular space as an input include using standard neural networks, embedding the feature space onto a grid to allow convolution [8], identifying local patches on the irregular manifold to perform geodesic convolutions [14], or graph signal processing based convolutions on graph signal data [7]. The potential applications of a convolutional network in the spatially irregular domain are expansive, however the graph convolution and pooling is not trivial, with graph representations of data being the topic of ongoing research [5, 21]. The use of graph representation of data for deep learning is introduced by [3], utilizing the Laplacian spectrum for feature mining from the irregular domain. This is further expanded upon in [7], providing derivative calculations for the back-propagation of errors during gradient descent. We formulate novel gradient equations that show more stable calculations in relation to both the input data and the tracked weights in the network.

In this methodology-focused study, we explore the use of graph based signal-processing techniques for convolutional networks on irregular domain problems. We evaluate two methods of graph pooling operators and report the effects of using interpolation in the spectral domain for identifying localized filters. We evaluate the use of Algebraic Multigrid agglomeration for graph pooling. We have also identified an alternative to the gradient calculations of [7] by formulating the gradients in regards to the input data as the spectral convolution of the gradients of the output with the filters (Equation 2), and the gradients for the weights as the spectral convolution of the input and output gradients (Equation 3). These proposed gradient calculations show consistent stability over previous methods [7], which in turn benefits the gradient based training of the network. Results are reported on the MNIST dataset and the subsampled MNIST on an irregular grid.

The rest of the paper is outlined as follows. Section 2 describes the generation of a graph based CNN architecture, providing the convolution and pooling layers in the graph domain by use of signal-processing on the graph. Section 3 details the experimental evaluation of the proposed methods and a comparison against the current state of the art, with Section 4 reporting the results found and conclusions drawn in Section 5.

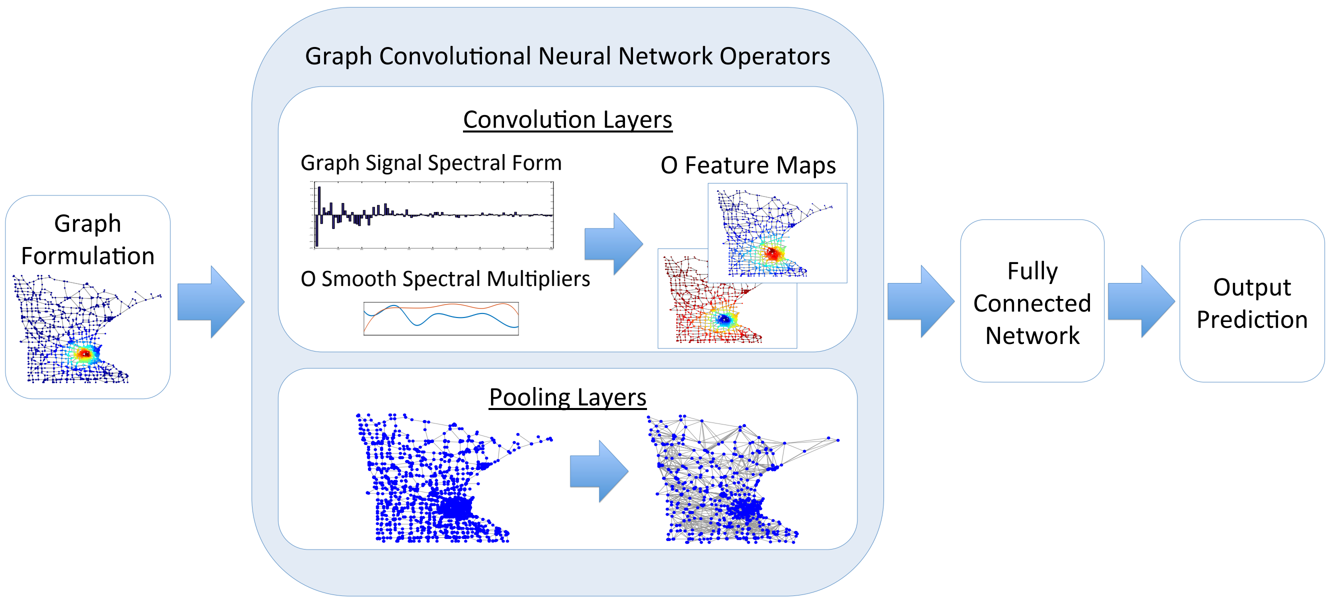

Figure 1: Graph based Convolutional Neural Network components. The GCNN is designed from an architecture of graph convolution and pooling operator layers. Convolution layers generate O output feature maps dependent on the selected O for that layer. Graph pooling layers will coarsen the current graph and graph signal based on the selected vertex reduction method.

2 Methods

The familiar CNN architecture pipeline consists of an input layer, a collection of convolution and/or pooling layers followed by a fully connected neural network and an output prediction layer. One issue with CNNs is that the convolution of a filter across the spatial domain is non-trivial when considering domains in which there is no regular structure. One solution is to utilize the multiplication in the spectral graph domain to perform convolution in the spatial domain, obtaining the feature maps via graph signal processing techniques. The graph based CNN follows a similar architecture to standard CNNs; with randomly initialized spectral multiplier based convolution learnt in the spectral domain of the graph signal and graph coarsening based pooling layers, see Figure 1 for a pipeline. Training is compromised of a feed-forward pass through the network to obtain outputs, with loss propagated backwards through the network to update the randomly initialized weights.

The topic of utilizing graphs for the processing of signals is a recently emerging area in which the graph G forms a carrier for the data signal f [19]. The graph holds an underlying knowledge about the spatial relationship between the vertices and allows many common signal processing operators to be performed upon f via G, such as wavelet filtering, convolution, and Fourier Transform [6, 19]. By representing the observed domain as a graph it is possible to perform the signal processing operators on the observed data as graph signals. Coupling these graph signal processing techniques with deep learning it is possible to learn within irregularly spaced domains, upon which conventional CNNs would be unable to convolve a regular kernel across. The proposed technique will therefore open the door for deep learning to be utilized by a wider collection of machine learning and pattern recognition domains with irregular, yet spatially related features.

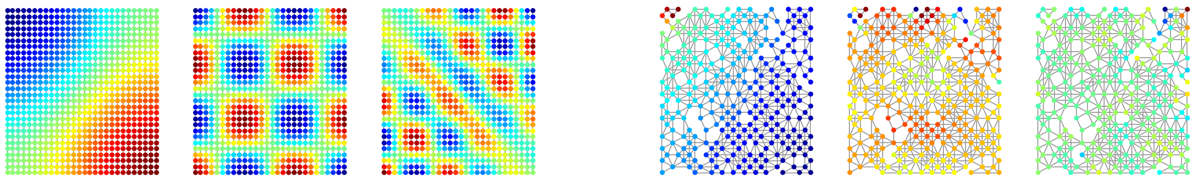

Figure 2: Eigenvectors $u_{i=\{2,20,40\}}$ of the full 28×28 regular gird (left) and the subsampled irregular grid (right).

2.1 Convolution on Graph

A graph $G = \{V, W\}$ consists of N vertices V and the weights W of the undirected, non-negative, non-selflooping edges between two vertices v_i and v_j. The unnormalized graph Laplacian matrix L is defined as $L = D - W$, where $d_{i,i} = \sum_{i=1}^{N} a_i$ forms a diagonal matrix containing the sum of all adjacencies for a vertex. Given G, an observed data sample is a signal $f \in \mathbb{R}^N$ that resides on G, where f_i corresponds to the signal amplitude at vertex v_i.

Convolution is one of the two key operations in the CNN architecture, allowing for locally receptive features to be highlighted in the input image [10]. A similar operator is presented in graph based CNN, however due to the potentially irregular domain graph convolution makes use of the convolution theorem of convolution in the spatial domain being a multiplication in the frequency domain [2].
To project the graph signal into the frequency domain, the Laplacian L is decomposed into a full matrix of orthonormal eigenvectors $U = \{u_{i=1...N}\}$, where u_i is a column of the matrix U, and the vector of associated eigenvalues $\lambda_{i=1...N}$ [19], Figure 2. The forward Graph Fourier Transform is therefore given for a given signal as $\tilde{f}_i = \sum_{l=1}^{N} \lambda_l f_i^T u_i$, and its corresponding inverse $f_i = \sum_{l=1}^{N} \lambda_l \tilde{f}_i u_i$. Using the matrix U the Fourier transform is defined as $\tilde{f} = U^T f$, and the inverse as $f = U\tilde{f}$, where U^T is the transpose of the eigenvector matrix.

For forward convolution, a convolutional operator in the vertex domain can be composed as a multiplication in the Fourier space of the Laplacian operator [2]. Given the spectral form of our graph signal $\tilde{f} \in \mathbb{R}^N$ and the spectral multiplier $k \in \mathbb{R}^N$, the convolved output signal in the original spatial domain is the spectral multipication, i.e. $y = U\tilde{f}k$. It is possible to expand this for multiple input channels and multiple output feature maps:

$$y_{s,o} = U \sum_{i=1}^{I} U^T f_{s,i} \odot k_{i,o} \ , \tag{1}$$

where I is the number of input channels for f, s is a given batch sample, and o indexes an output feature map from O output maps.

Localized regions in the spatial domain are defined by the kernel receptive field in CNNs, and for graph based CNNs the spatial vertex domain localization is given by a smoothness within the spectral domain. Therefore to identify local features within the spatial domain the spectral multipliers used for spectral convolution are identified by tracking a subsampled set of filter weights $\hat{k}_{i,o}k \in \mathbb{R}^{<N}$ which are interpolated up to a full filter via a smoothing kernel Φ such as cubic splines: $k_{i,o} = \Phi\hat{k}_{i,o}$. This has the added benefit of reducing the number of tracked weights, however leads to an extra pair of operations in interpolating the weights to the full $k \in \mathbb{R}^N$ for multiplication. Reducing the number of tracked weights increases the smoothness of the final interpolated filter, and lowering the tuning parameter of the number of tracked weights learns sharper features.

2.2 Backpropagation on Graph

Backpropagation of errors is a pivotal component of deep learning, providing updates of weights and bias for the networks towards the target function with gradient descent. This requires obtaining derivatives in regards to the input and weights used to generate the output, in the case of graph based CNN convolution the gradients are formulated in regards to the graph signal f and the spectral multipliers k. The gradients for an input feature map channel $f_{s,i}$ is given as the convolution of the gradients for the output ∇y and the spectral multipliers in the spectral domain via

$$\nabla f_{s,i} = U \sum_{o=1}^{O} U^T \nabla y_{s,o} \odot k_{i,o} \tag{2}$$

for a provided batch of S graph signals. Gradients for the full set of interpolated spectral multipliers is formulated as the convolution of the gradients for the output ∇y with the input $f_{s,i}$ via

$$\nabla k_{i,o} = \sum_{s=1}^{N} U^T \nabla y_{s,o} \odot U^T f_{s,i}. \tag{3}$$

As the filters are spectral domain multipliers, we do not project this spectral convolution back through the graph Fourier transform. The smooth multiplier weights ∇k can then be projected back to the subsampled set of tracked weights by the multiplication with the inversed smoothing kernel $\nabla \hat{k}_{i,o} = \Phi^T \nabla k_{i,o}$.

2.3 Pooling on Graph

The pooling layer is the second component in conventional CNN, reducing the resolution of the input feature map in both an attempt to generalize the features identified and to manage the memory complexity when using numerous filters [1]. During graph based convolutions there is no reduction in size between the input signal and the output feature map due to the multiplication of the $\mathbb{R}^N$ filter with the $\mathbb{R}^N$ spectral signal. As such, each layer of a deep graph CNN would possess a graph with $\mathbb{R}^N$ vertices. Such a construction could be beneficial, as this would allow the algorithm to store a single instance of the graph and the associated N^2 eigenvector matrix U. If pooling is utilized, there is benefit gained from the feature map generalization and the reduction in complexity of the graph Fourier transforms as each layer's vertex count N is lowered. To pool local features together on the graph, it is required to perform graph coarsening and project the input feature maps through to the new, reduced size graph. Coarsening $G = \{V,W\}$ to $\hat{G} = \{\hat{V},\hat{W}\}$ not only requires the reduction of vertex counts, but also a handling of edges between the remaining $\hat{N}$ vertices. Common methods of generating $\hat{V}$ are to either select a subset of V to carry forward to $\hat{G}$ [18] or to form completely new set of nodes $\hat{V}$ from some aggregation of related nodes within V [16]. One method for selection of $\hat{V}$ can be achieved by selecting the largest eigenvalue λ_N and splitting V into two subsets based on the polarity of the associated eigenvector U_N [12]. We can therefore define $\hat{V} = \{u_N, i\}; u_N, i >= 0$ and its complement $\hat{V}^c = \{u_N, i\}; u_N, i < 0$. $\hat{V}$ is then utilized to construct the graph $\hat{G}$, although by reversing the selection for the polarity to keep it is just as understandable to choose $\hat{V}^c$ for construction of $\hat{G}$.

In this study we utilize Algebraic Multigrid (AMG) for graph coarsening, a method of projecting signals to a coarser graph representation obtained via greedy selection of vertices [16]. Aggregation takes a subset of vertices on V and generates a singular vertex in the new

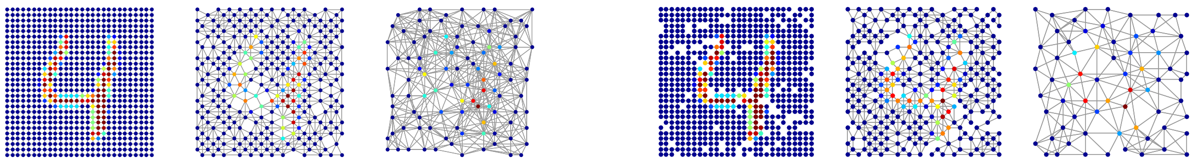

Figure 3: Two levels of graph pooling operation on regular and irregular grid with MNIST signal. From left: Regular grid, AMG level 1, AMG level 2, Irregular grid, AMG level 1, AMG level 2.

set of coarsened nodes $\hat{V}$ in the output graph. Graph coarsening is by no means a trivial task, with extensive literature exploring the subject [12, 16, 17].

With a coarser graph structure $\hat{G}$ it is required to then downsample the graph signal $f_{1:N}$ into a new signal $\hat{f}_{1:n}$ that is able to reside on $\hat{G}$. AMG provides a set of matrices for the interpolation of the input signal f; the restriction matrix R and the projection matrix P. Downsampling $f \in \mathbb{R}^N$ on G to $\hat{f} \in \mathbb{R}^{\hat{N}}$ on $\hat{G}$ is achieved by the multiplication of the signal with the restriction matrix, $\hat{f}_{s,i} = Rf_{s,i}$, whilst the reverse pooling required for backpropagation is achieved via multiplication with the projection matrix, $f_{s,i} = P\hat{f}_{s,i}$.

3 Implementation

Although we utilize forms of the 2D grid, the graph CNN is generalizable to more irregular domain problems; such as sensor networks, mesh signals, text corpora, human skeleton graphs and more. These domains quite often contain irregular spatial geometries, upon which it is non-trivial to define a filter kernel for convolution. In this study we evaluate the performance of the proposed graph CNN derivative calculations with an implementation on both the standard regular 28 grid and the irregular subsampled 2D grid.

The 2D grid is the graph representation of the Von Neumann neighborhood of vertices in a regular domain, most commonly applied to that of pixel relationships in images. For an image, each pixel is represented by a vertex on G, with the pixel intensities for each vertex forming the graph signal f. The edge weights are taken as the euclidean distance between the nodes in the Von Neumann neighborhood. To evaluate the performance of graph CNN on the 2D grid we utilize the MNIST dataset, consisting of 60,000 examples of handwritten numerical digits in 28×28 grayscale pixel images. The edge weights for G are the binary presence of an edge between v_i and v_j on the 4-way adjacency, with $V \in \mathbb{R}^{784}$.

To obtain an irregular spatial geometry domain upon which a conventional CNN cannot convolve, we subsampled the 28×28 grid by selecting 84 random vertices to exclude from the grid. Upon removing the selected vertices and their corresponding edges from the graph, we then subsample the MNIST dataset with the respective signals such that $f \in \mathbb{R}^{700}$. This irregular spatial domain now requires the graph-based CNN operators above to form a convolved output feature map.

The architecture of the graph CNN was set to $C^{20}PC^{50}PRF$; where C^{κ} defines a convolutional layer with 60 tracked weights and κ output feature maps, P defines an AMG pooling with a coarsening factor of $\beta = 0.05$ and 2 levels, R defines a rectified linear unit layer, and finally F describes fully connected layers providing output class predictions. Networks were trained for 500 epochs, with the full 10,000 test samples being classified at each epoch to track the predictive performance of the network.

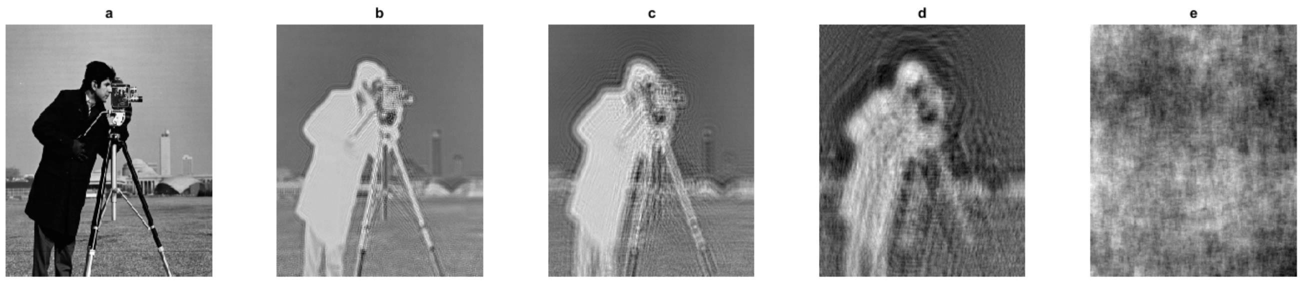

Figure 4: Effect of spline interpolation of tracked weights on spectral filter smoothness during graph-based convolution. a) Original image, b) $\hat{k} = \mathbb{R}^{\text{ceil}(\sqrt[4]{N})}$, c) $\hat{k} = \mathbb{R}^{\text{ceil}(\sqrt[3]{N})}$, d) $\hat{k} = \mathbb{R}^{\text{ceil}(\sqrt[2]{N})}$, e) $\hat{k} = \mathbb{R}^{N}$

To perform derivative checking, the calculation of the gradients for ∇f, ∇k and $\nabla \hat{k}$ were evaluated using random perturbations of errors on the scale of 10^{-4}. Derivatives for $\nabla \hat{k}$ were checked for interpolation over varying numbers of tracked weights in the network, including the full set $\hat{N} \in \mathbb{R}^{N}$. The experiment was repeated 100 times and the average percentage error of the calculated gradient versus the empirically obtained gradient is reported in Figure 7.

The graph-based CNN architecture was implemented within MATLAB using GPU enabled operators.

4 Results

The graph CNN method was evaluated on both the regular 28×28 grid and irregular randomly subsampled grid. We report the predictive accuracy of the network at each epoch of training using both the proposed graph CNN method and the method proposed by [7]. We also show the effects of smoothed spectral multiplier filters on the convolution output and the derivative errors we obtained for gradient calculations. In summary we found that by increasing the smoothness of the spectra filters we were able to increase the local relationship of features in the spatial domain, however this also resulted in higher error being introduced by the interpolation when calculating the gradients of the tracked weights $\hat{k}$. Overall we found that the proposed calculations for derivatives in respect to $\hat{k}$ introduced little error during backpropagation. The accuracy observed when testing unobserved samples is very promising, exceeding 94% on both the regular and irregular geometry domains.

4.1 Convolution and filter smoothness

Reducing the number of tracked filter weights produces smoother spectral multipliers after interpolation up to $k_{i,o} = \Phi \hat{k}_{i,o}$. Figure 4 shows the effect of interpolating weights from various lengths of $\hat{k}$ as applied to the 2D graph with the Cameraman.tif graph signal residing on it. As the number of tracked weights is reduced the spatial locality of the features learnt is reduced, providing sharper features, whilst as the number of tracked weights approaches N the spatial localization of the feature map is lost.

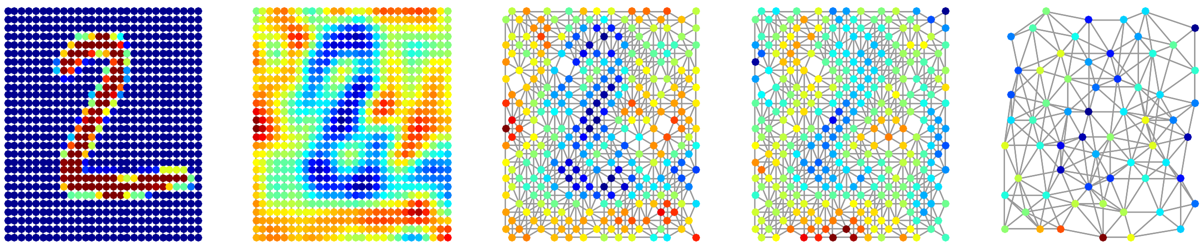

Figure 5: Feature maps formed by a feed-forward pass of the regular domain. From left: Original image, Convolution round 1, Pooling round 1, Convolution round 2, Pooling round 2.

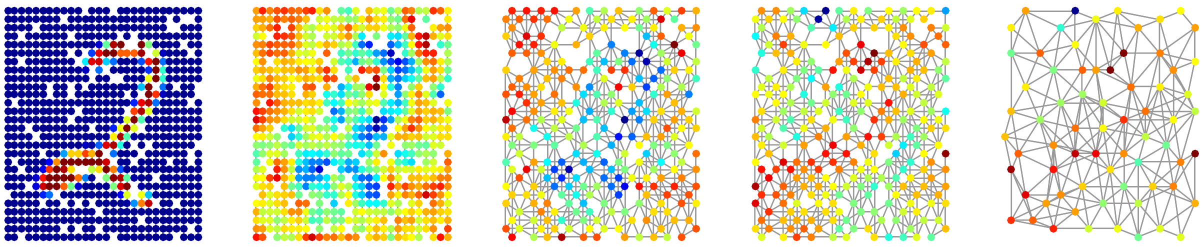

Figure 6: Feature maps formed by a feed-forward pass of the irregular domain. From left: Original image, Convolution round 1, Pooling round 1, Convolution round 2, Pooling round 2.

4.2 Localized feature maps

By interpolating smooth spectral multipliers from the 60 tracked weights we were able to convolve over the irregular domain to produce feature maps in the spatial domain with spatially localized features. Figure 6 visualizes output for each layer of the Graph CNN convolution and pooling layers for both the regular and irregular domain graphs.

4.3 Backpropagation derivative checks

The proposed method gave an average of $1.41\%(\pm 4.00\%)$ error in the calculation of the gradients for the input feature map. In comparison, by not first applying a graph Fourier transform to $\nabla y_{s,o}$ in the calculation for $\nabla f_{s,i}$, as in [7], we obtain errors of $376.50\%(\pm 1020.79\%)$. Similarly the proposed method of obtaining the spectral forms of $\nabla y_{s,o}$ and $f_{s,i}$ in the calculation of $\nabla k_{i,o}$ gave errors of $3.81\%(\pm 16.11\%)$. By not projecting to the spectral forms of these inputs, errors of $826.08\%(\pm 4153.32\%)$ are obtained. Figure 7 shows the average percentage derivative calculation error for $\nabla \hat{k}$ of varying numbers of tracked weights over 100 runs. The proposed method of gradient calculation shows lower errors than the compared method gradient calculation of ∇k when $\hat{k} \in \mathbb{R}^N$ and all but the lowest number of tracked weights of $\hat{k} \in \mathbb{R}^{100}$. The introduction of interpolation leads to a higher introduction of error into the calculated gradient errors, especially in the presence of a low number of tracked weights.

4.4 Testing performance

Classification performance on the MNIST dataset is reported in Table 1, with progression of testing accuracy over epochs given in Figure 8 comparing between the proposed gradient calculations and those of [7]. The proposed graph CNN method does not obtain the 99.77% accuracy rates of the state of the art CNN architecture presented by [4] on the full 28×28

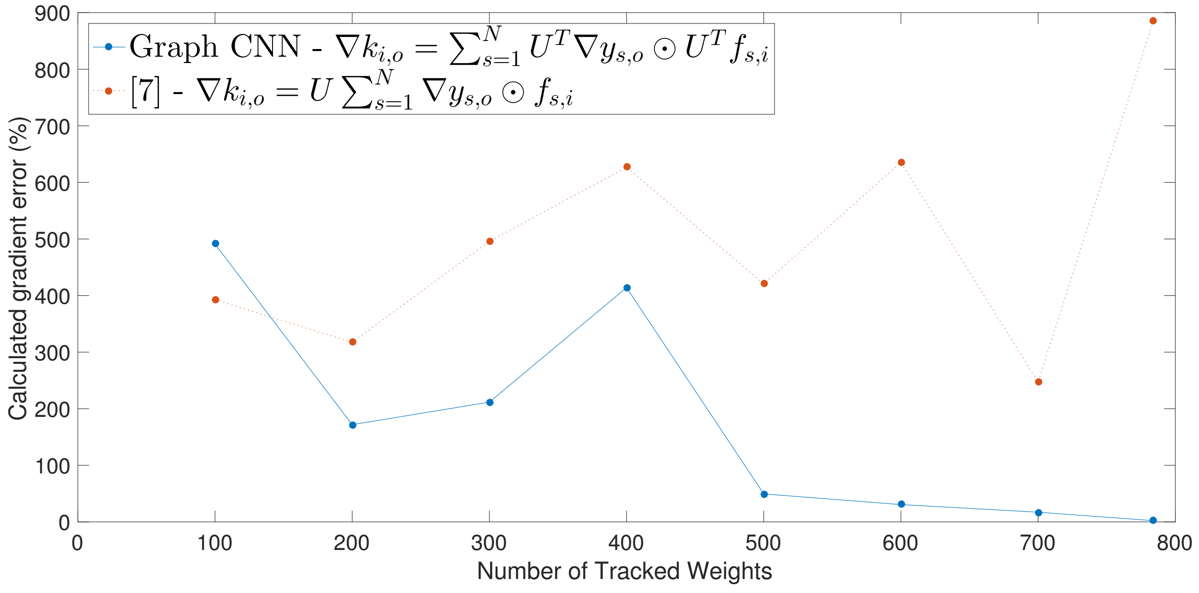

Figure 7: Gradient calculation errors for interpolation of various numbers of tracked weights.

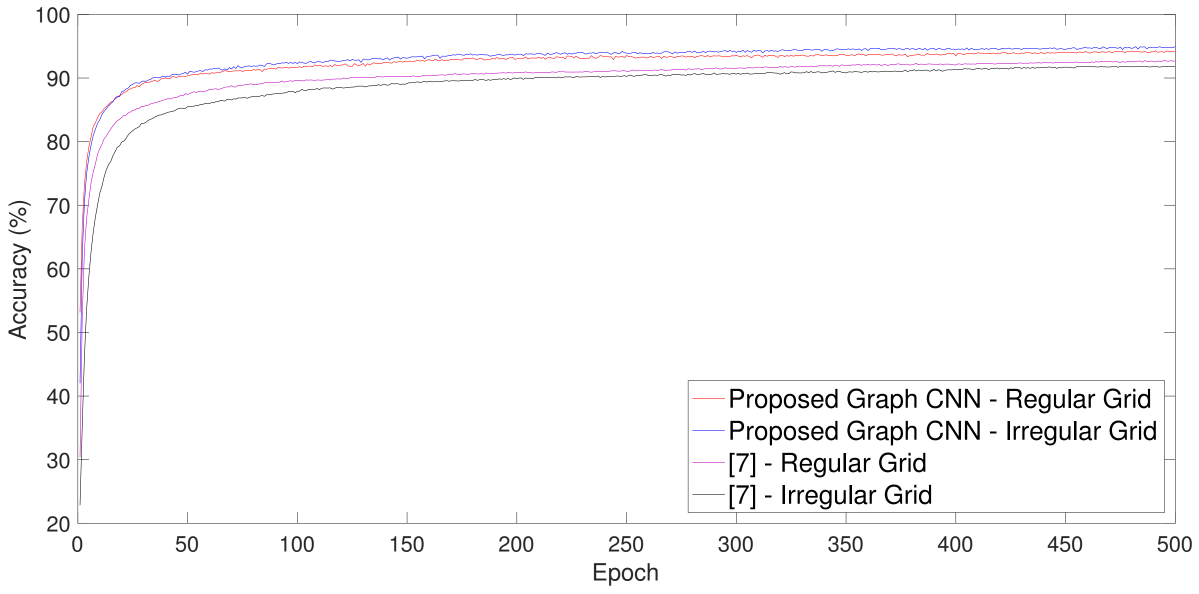

Figure 8: Test Set accuracy on the MNIST dataset on the regular and irregular 2D grid. An increasing in testing accuracy is observed when utilizing the proposed gradient calculations from equations 2 and 3.

grid. This is understandable, as standard CNNs are designed to operate in the regular Cartesian space, giving it a strong performance in the image classification problem. The main benefit of the graph CNN is in it's ability to handle the irregular spatial domain presented by the subsampled MNIST grid by use of convolution in the graph spectral domain.

5 Conclusion

This study proposes a novel method of performing deep convolutional learning on the irregular graph by coupling standard graph signal processing techniques and backpropagation based neural network design. Convolutions are performed in the spectral domain of the graph

Table 1: Testing set accuracy of network (%)

Dataset	[7]	Proposed Graph CNN
Regular grid MNIST	92.69	**94.23**
Subsampled irregular grid MNIST	91.84	**94.96**

Laplacian and allow for the learning of spatially localized features whilst handling the non-trivial irregular kernel design. Results are provided on both a regular and irregular domain classification problem and show the ability to learn localized feature maps across multiple layers of a network. A graph pooling method is provided that agglomerates vertices in the spatial domain to reduce complexity and generalize the features learnt. GPU performance of the algorithm improves upon training and testing speed, however further optimization is needed. Although the results on the regular grid are outperformed by standard CNN architecture this is understandable due to the direct use of a local kernel in the spatial domain. The major contribution over standard CNNs is the ability to function on the irregular graph is not to be underestimated. Graph based CNN requires costly forward and inverse graph Fourier transforms, and this requires some work to enhance usability in the community. Ongoing study into graph construction and reduction techniques is required to encourage uptake by a wider range of problem domains.

References

[1] Y-Lan Boureau, Jean Ponce, and Yann Lecun. A theoretical analysis of feature pooling in visual recognition. In *Proc. Int. Conf. Mach. Learning*, 2010.

[2] Ronald Bracewell. *The Fourier Transform & Its Applications*. McGraw, 1999.

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *CoRR*, abs/1312.6203, 2013.

[4] Dan C. Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. *CoRR*, abs/1202.2745, 2012.

[5] Leo Grady and Jonathan R. Polimeni. *Discrete Calculus - Applied Analysis on Graphs for Computational Science*. Springer, 2010.

[6] David K. Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129 – 150, 2011.

[7] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *CoRR*, abs/1506.05163, 2015.

[8] E. P. Ijjina and C. K. Mohan. Human action recognition based on mocap information using convolution neural networks. In *Proc. Int. Conf. Mach. Learning and Applications*, pages 159–164, Dec 2014. doi: 10.1109/ICMLA.2014.30.

[9] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093*, 2014.

[10] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[11] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *Proc. IEEE Conf. on Comp. Vis. and Pat. Rec.*, pages 5325–5334, 2015.

[12] P. Liu, X. Wang, and Y. Gu. Graph signal coarsening: Dimensionality reduction in irregular domain. In *IEEE GLobal Conf. Signal and Information Processing*, pages 798–802, 2014.

[13] D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. Int. Conf. on Comp. Vis.*, volume 2, pages 1150–1157, 1999.

[14] Jonathan Masci, Davide Boscaini, Michael M. Bronstein, and Pierre Vandergheynst. Shapenet: Convolutional neural networks on non-euclidean manifolds. *CoRR*, abs/1501.06297, 2015.

[15] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proc. IEEE Conf. on Comp. Vis. and Pat. Rec.*, pages 1717–1724, 2014.

[16] Ilya Safro. Comparison of coarsening schemes for multilevel graph partitioning. In *Int. Conf. Learning and Intelligent Optimization*, pages 191–205. Springer-Verlag, 2009.

[17] Ilya Safro, Peter Sanders, and Christian Schulz. *Proc. Int. Symposium Experimental Algorithms*, chapter Advanced Coarsening Schemes for Graph Partitioning, pages 369–380. Springer Berlin Heidelberg, 2012.

[18] D. I. Shuman, M. J. Faraji, and P. Vandergheynst. A multiscale pyramid transform for graph signals. *IEEE Trans. Signal Process.*, 64(8):2119–2134, 2016.

[19] 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.

[20] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In *Proceeding of the ACM Int. Conf. on Multimedia*, 2015.

[21] Cha Zhang, Dinei Florencio, and Philip Chou. Graph signal processing - a probabilistic framework. Technical Report MSR-TR-2015-31, 2015.

[22] Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.