Parameterized Pseudo-Differential Operators for Graph Convolutional Neural Networks

Kevin Potter* kmpotte@sandia.gov
Steven Sleder* ssleder@sandia.gov
Matthew Smith* mdsmith@sandia.gov
Shehan Perera* srperer@sandia.gov
Alper Yilmaz† yilmaz.15@osu.edu
John Tencer* jtencer@sandia.gov

Abstract

We present a novel graph convolutional layer that is conceptually simple, fast, and provides high accuracy with reduced overfitting. Based on pseudo-differential operators, our layer operates on graphs with relative position information available for each pair of connected nodes. Our layer represents a generalization of parameterized differential operators (previously shown effective for shape correspondence, image segmentation, and dimensionality reduction tasks) to a larger class of graphs. We evaluate our method on a variety of supervised learning tasks, including 2D graph classification using the MNIST and CIFAR-100 datasets and 3D node correspondence using the FAUST dataset. We also introduce a superpixel graph version of the lesion classification task using the ISIC 2016 challenge dataset and evaluate our layer versus other state-of-the-art graph convolutional network architectures.

The new layer outperforms multiple recent architectures on graph classification tasks using the MNIST and CIFAR-100 superpixel datasets. For the ISIC dataset, we outperform all other graph neural networks examined as well as all of the submissions to the original ISIC challenge despite the best of those models having more than 200 times as many parameters as our model.

1. Introduction

Convolutional neural networks (CNNs) have performed incredibly well on tasks such as image classification, segmentation, and object detection [16]. While there have been diverse architectural design innovations leading to improved accuracy across these tasks, all share the common property that they operate on structured Euclidean domain inputs. A growing body of research on transferring these successes to non-Euclidean domains, such as manifolds and graphs, has followed [36].

We focus on unstructured graphs which represent discretizations of an underlying metric space. These data types are ubiquitous in computational physics, faceted surface meshes, and images. Previous efforts to extend CNNs to this type of data have involved parameterized function approximations on localized neighborhoods, such as MoNet [24] and SplineCNN [12]. These function approximations (Gaussian mixture models in the case of MoNet and B-spline kernels in the case of SplineCNN) are complex and expensive to calculate relative to CNN kernels.

Inspired by earlier work in shape correspondence [6], image segmentation on the unit sphere [17], and low-dimensional embeddings of computational physics data [31] we seek to utilize parameterized differential operators (PDOs) to construct convolution kernels. In contrast to MoNet and SplineCNN, parameterized differential operators are cheap to compute and involve only elementary operations. Boscaini et al. [6] used anisotropic diffusion kernels while Jiang et al. [17] included gradient operators in addition to an isotropic diffusion operator. Tencer and Potter [31] performed an ablation study of the differential operators used and demonstrated that including the gradient operators in addition to the Laplacian is broadly beneficial, but that little is gained by including additional terms.

Prior work [17, 31] used differential operators precomputed for specific meshes which has two drawbacks: (1) precomputing operators is not practical for datasets where the connectivity graph varies between samples, and (2) differential operators place restrictions on graph connectivity. Differential operators defined for mesh topologies rely on element connectivity information which is unavailable for more general graphs. In contrast to these prior works, we do not precompute any operators and we do not directly use differential operators. Instead, we formulate *pseudo-differential* operators which are cheap and easy to compute at run-time for a more general class of graphs.

*Sandia National Laboratories: Albuquerque, NM USA
†Ohio State University: Columbus, OH USA

While our approach only applies to graphs with relative position information for each node, the set of graphs with the required positional information is large, encompassing nearly all physical systems as well as a significant number of other graphs, such as graph representations derived from image data. Our method is especially well-suited for analyzing graph image representations in addition to being applicable to the datasets used by prior investigators [17, 31] to demonstrate PDO-based approaches. For regular meshes, our pseudo-differential operators closely approximate the differential operators used in those works.

1.1. Our contributions

We created a novel layer architecture inspired by PDOs.

- We improve upon the static matrix approach of previous works [17, 31] with a dynamic method that enables support for variable graph forms and eliminates the need to precompute matrices.
- Our method utilizes pseudo-differential operators in contrast to the differential operators used in prior works. Pseudo-differential operators are cheap to compute and are applicable to a broader class of graphs than differential operators.
- Our novel mixing layer is conceptually simple and easy to code (integrating painlessly with existing graph libraries). (Section 4.1)
- The new approach is accurate for both sparsely and densely connected graphs, including state-of-the-art results for the MNIST superpixel dataset even with reduced edge connection input data. (Section 5.1)
- The new approach is faster than common approaches for equivalent numbers of features owing to the simpler mathematical functions involved. (Section 5.1.2)

2. Problem

Many real world datasets may be treated as attributed unstructured graphs with positional information provided for the nodes or relative positional information provided for the edges. In physics and engineering, high-fidelity simulation tools represent continuous spatial functions using unstructured spatial discretizations. In computer vision, surface meshes and point cloud image representations are common.

Operating directly on these unstructured graphs has proven challenging. Our ideal solution, would be fast, simple, scale well, make efficient use of information, and operate effectively on sparsely connected graphs. While a number of existing approaches are applicable to this class of dataset, all have some limitation that makes them ill suited for our use cases such as requiring the evaluation of expensive patch operators [24, 12], the precomputation of many (potentially high-dimensional) operators [17, 31], or discarding either graph connectivity information [14] or positional information [18, 4]. We seek a scalable approach that supports heterogeneous graphs.

3. Related work

Early approaches showed that CNNs could be used on non-Euclidean domains by introducing new intrinsic convolutional methods [23, 6] that operate on input manifolds. More recent approaches [24] are capable of performing well on both manifolds and general graphs by creating pseudo-coordinates for either the vertices of a graph or points on a manifold, and then learning a kernel in that space.

Graph convolutional neural networks (GCNs) form the basis for other approaches that have shown great results [36]. The literature has been split among methods based on spectral graph theory [9, 18, 21, 24, 31] and methods that operate with spatial filters [12, 14, 32] . Our method falls into the latter category of spatial approaches.

4. Method

Our layer works by calculating several quantities that are analogous to the sets of differential operators used as convolutional filters by previous authors [17, 31]. For each node of a layer, each input channel has multiple values calculated (4 items for 2-dimensional and 5 for 3-dimensional graphs):

- Value of the node in the prior layer (identity)
- Average values at the 1-ring neighbors of the node
- Average gradient components (i.e. $\frac{\partial}{\partial x}$,$\frac{\partial}{\partial y}$) along each connected edge weighted by inverse edge length

These items are then mixed by a neural network yielding the desired number of output channels.

4.1. Convolutional Neural Networks Based on Parameterized Pseudo-Differential Operators

In contrast to previous methods utilizing PDOs, the method presented here generalizes to heterogeneous datasets in which the number of nodes, the connectivity, and positions vary across samples. Additionally, by relaxing the definitions of the gradient operators, our implementation is applicable to overconnected graphs (such as superpixel image representations) rather than only meshes suitable for common PDE solution methods (finite element, finite volume, etc.). A consequence of this generalization is slightly more computational overhead from dynamically generating the required operators on the fly rather than precomputing them offline. However, as seen in Figure 2, this overhead does not slow training time relative to other methods.

The key components of our network architectures are mixing and pooling layers. Our mixing layer is implemented using pytorch-geometric's `MessagePassing` base class [11]. For pooling layers, we evaluated two clustering method implementations from pytorch-geometric: `voxel_grid` [27] and `graclus` [10]. The `voxel_grid` implementation gave the best accuracy in all our trials and was used to generate all of the following results.

The `MessagePassing` class operates by calculating new values $x_i^{(k)}$ for each node i on layer $^{(k)}$ with information from their prior node values $x_i^{(k-1)}$, prior values at connected nodes $x_j^{(k-1)}$, and/or the edge attributes $e_{i,j}$.

$$\mathbf{x}_i^{(k)} = \gamma^{(k)}\left(\mathbf{x}_i^{(k-1)}, \square_{j\in\mathcal{N}(i)}\,\phi^{(k)}\left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{i,j}\right)\right) \tag{1}$$

$\square_{j\in\mathcal{N}(i)}$ is an aggregation method (we chose mean) that aggregates $\phi^{(k)}$ for each node j connected to node i in layer $(k-1)$ into a specific number of values independent of the amount of edge connections. $\phi^{(k)}$ and $\gamma^{(k)}$ are arbitrary differentiable functions.

4.1.1 Choosing $\phi^{(k)}$ and $\gamma^{(k)}$

For $\gamma^{(k)}$, we chose to use a neural network. Convenient and physically motivated choices for $\phi^{(k)}$ are derived from mesh differential operators, e.g. I, ∇, or Δ. The identity operator I simply passes the value forward. For the others, let $f:\Omega\to\mathbb{R}$ be a smooth scalar function for which only the values $f_1,\ldots,f_n$ at the nodes are known and f_i corresponds to the value of f at node i. The Laplacian operator Δ may be expressed as the difference between f_i and the average value of f_j, $j\in\mathcal{N}(i)$. For triangle meshes, the nodal gradient operator ∇ is often expressed as the average gradient in the adjacent facets which is equivalent to a weighted sum over the connected edge gradients:

$$\nabla f_i \approx \sum_{v_j\in\mathcal{N}(i)} w_{i,j}\nabla f(e_{i,j}). \tag{2}$$

with $\nabla f(e_{i,j}) = \frac{f_i - f_j}{r_{i,j}}\hat{e}_{i,j}$. We denote the Euclidean distance between the two nodes as $r_{i,j}$, and the unit vector oriented along the edge $e_{i,j}$ as $\hat{e}_{i,j}$. For a 2D mesh without intersecting edges $w_{i,j} = \frac{A_{i,j}}{\sum_{j\in\mathcal{N}(i)} A_{i,j}}$, where $A_{i,j}$ is the total area of the 2 facets connected to $e_{i,j}$ [22]. For an over-connected graph, these facet areas are not defined.

If we weight the contributions of each edge equally (to bypass the facet area problem), our choice for $\phi^{(k)}$ becomes

$$\frac{\left(\mathbf{x}_i^{(k-1)} - \mathbf{x}_j^{(k-1)}\right) r_{x,i,j}}{r_{i,j}^2}, \frac{\left(\mathbf{x}_i^{(k-1)} - \mathbf{x}_j^{(k-1)}\right) r_{y,i,j}}{r_{i,j}^2}, \mathbf{x}_j^{(k-1)} \tag{3}$$

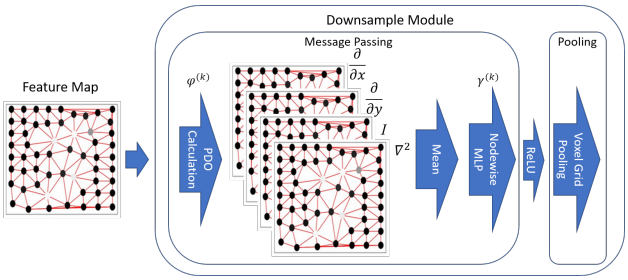

Figure 1: Depiction of downsampling module based on our graph convolutional layer and voxel grid pooling [27]. Each feature map is operated on by the 4 (5 for 3-D) operators before being mixed via a nodewise neural network. A nonlinear activation is applied and finally it pools the result.

where $r_{x,i,j}$ and $r_{y,i,j}$ are the differences in positions of nodes i and j in the x- and y-dimensions, respectively. $\phi^{(k)}$ returns a stack of these values for every node's connected nodes. The first 2 terms of (3) are the x- and y-components of the gradient. The 3^{rd} term results in the average of the neighboring nodes, a precursor to the Laplacian operator. Given that the next step after aggregation is to mix these components using our $\gamma^{(k)}$ function (a neural network) the Laplacian can be reconstructed from these terms plus the identity (if found useful by the network). The identity term $x_i^{(k-1)}$ is concatenated after aggregation of (3) by taking the mean over j. Like the Laplacian, the gradients give edge detection along the primary axes but with a linear combination can be discriminative in any direction (if beneficial).

We can easily extend these to more than 2 dimensions by adding gradient terms for each new dimension. This handles higher dimensionalities very well, as the parameter count scales linearly with the number of dimensions. 3 (or more) dimensional inputs can be handled without memory concerns, at least from exploding parameter counts.

Note that none of these values require complex calculations, which contributes to the layer's superior computational performance (Section 5.1.2). We hypothesize that by limiting the representational space for each node such that it knows only about the local gradient and itself the network is forced to find simple representations that generalize better.

5. Experiments

We evaluate our method on graph classification and node correspondence tasks from a variety of datasets. We consider 3 different classification datasets, including MNIST [24], CIFAR-100 [20], and ISIC2016 lesion classification [15]. In all cases, classification is performed based on a compressed superpixel graph representation. For MNIST we explore the effect of various hyperparameters on accuracy, convergence rate, and performance. To evaluate the performance of our method for node correspondence tasks,

we use the FAUST dataset [5].

In the absence of validation sets, we selected test results based on the best training accuracy in order to avoid biasing our model selection to the test set. For our models, this is often comparable to the overall best test accuracy (Figure 2). While there is some variation in results across our model initializations, the trend of high test performance following high training set accuracy was robust. This allowed a bias-free way of selecting performant models from multiple runs of the same hyperparameters. While we considered using k-fold cross validation, we found that our method was effective in picking out generally applicable models without biasing to the test set or introducing additional complexity to the training procedure. For all training runs, our models had training accuracies on par with test accuracies.

We compare our model performance against published results from the literature as well as implementations of other published models. We used the implementations of SplineCNN [12] and graph attention layers (GATConv) [32] available from pytorch-geometric [11] and utilized early stopping to avoid overfitting. In addition, we apply models from Gray et al. [14] and Knyazev et al.'s [19] to the MNIST and CIFAR datasets and report the results.

All models were trained with Adam optimization using a learning rate of 0.0002 and cross entropy loss for our architecture and the published options for comparison models unless otherwise noted. Learning rates were reduced by a factor of 10 on a plateau of 5 epochs without improving training loss up to a limit of $1/1000^{th}$ the original learning rate.

5.1. MNIST

We test against 3 variants of the MNIST superpixel 75 dataset [24], a compressed graph representation of MNIST in which each superpixel graph contains 75 nodes. The first variant uses all of the edges present in the original graph which we call the *raw* dataset. The second is a variant of the *hierarchical* set used by Knyazev et al. [19] with sets of approximately 75, 21, and 7 superpixels forming the feature hierarchy for each sample. The number of edges for each node was limited to its 32 nearest neighbors by Euclidean distance. The third variant, which we refer to as *pruned* is obtained by discarding all of the edge data from the original graph and applying Delaunay triangulation to the nodes, which reduces the number of edge connections by around 70% on average.

Our network architecture (Figure 3) is comprised of up to 7 layers of downsampling modules (Figure 1) followed by 2 fully connected layers with an exponential linear unit activation after the first fully connected layer and softmax on last layer. The deepest (7 of our layers) network is used for all main results because it provides the best accuracy. Each downsampling module consists of our introduced graph convolutional layer followed by a voxel grid pooling operation [27] outputting a reduced set of nodes with a selectable count of features. Pooling is done with voxel size halved at each step. Initial voxel size is set such that a 3×3 set of nodes is present after the downsampling operations prior to flattening for the fully connected layers.

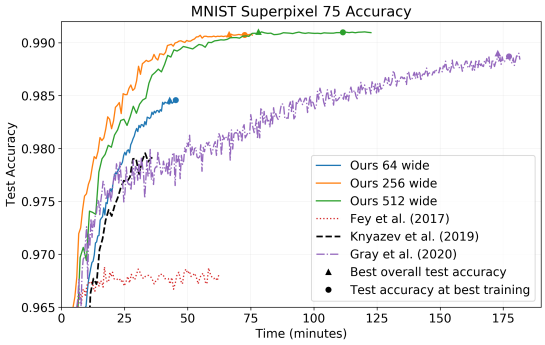

Figure 2: Accuracy compared to training time. While some of the other methods perform slightly better than our 7 layer deep model in time per epoch, we achieve higher accuracies per training minute (in addition to achieving a higher overall accuracy). Of note, using a wider network is not a penalty in accuracy vs training time. Plots show the mean test accuracy/time over 10 runs — except Gray et al. [14], which shows the median results from 10 runs due to their higher variability.

We used no normalization methods in the downsampling modules (e.g. batch normalization, dropout, edge dropout), but standard dropout of 0.5 is applied prior to each of the fully connected layers.

Our model with the best performance on all the MNIST variants used 7 downsampling layers starting with 128 features. Features doubled each layer until a maximum width of 512 — which was used for the remaining layers. Edges were dropped out on the input to prevent overfitting to the training set (Section 5.1.1).

Our architecture achieves a state-of-the-art test error of **0.80%** against the raw dataset using our best training accuracy as a selector among 32 runs. This compares to the prior best reported value of 0.95% from Gray et al. [14]. We were able to reproduce the Gray result [14] but only by selecting the best overall test accuracy out of 5 runs. The average results for ours and several competitors are shown in Table 1. Our results correspond to an input edge dropout rate of 0.45 and a learning rate of 0.002.

For the hierarchical variant, we note a slight drop in accuracy relative to the raw dataset. The hierarchical dataset adds additional nodes for various scales of abstraction (parent, grandparent, etc) with each level being progressively coarser. Knyazev et al. [19] explicitly treats each of these

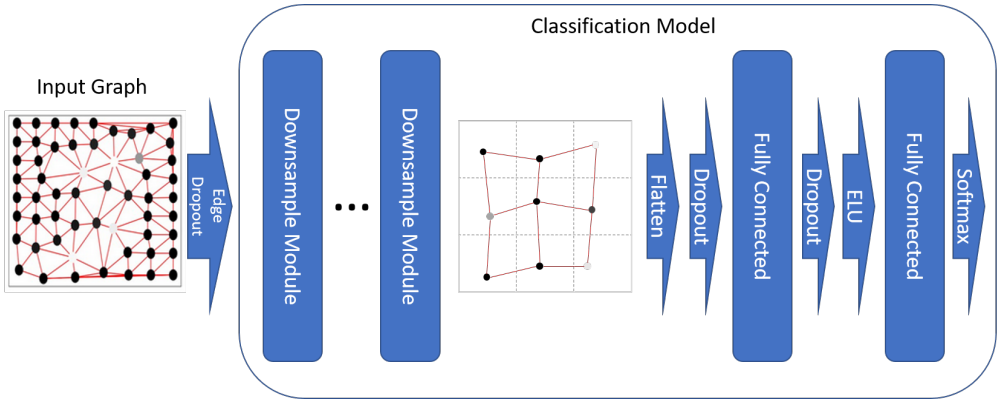

Figure 3: Depiction of classification model architecture. Edge dropout at a specified rate is performed before model input (Section 5.1.1). Multiple downsampling modules (Figure 1) are stacked. Voxel grid size is set for each module to result in a 3×3 grid prior to flattening for input to the fully connected layers. Nodes located in the same grid are collapsed into a single node during pooling.

Table 1: MNIST superpixel image classification results. Our model and Fey et al.'s [12] were trained for 100 epochs on the MNIST dataset. Knyazev et al.'s [19] was trained for 30 epochs and Gray et al.'s [14] for 400 epochs, keeping to their implementations. To avoid biasing to the test set, we selected the models by best training accuracy for a given set of hyperparameters — with the exception of Fey et al.'s for which we chose the overall best for each particular run. Average and standard deviation values taken across 32 runs for the raw and hierarchical variants, and against 10 for the pruned variant. Given that Knyazev et al.'s was highly dependent on the extra information from their hierarchical representation, we did not run their code against the pruned version. Veličković et al. [32] produced a graph attention convolutional layer but when applied to MNIST it performed poorly (and was worse without embedding positions into the input vector). Gray et al. [14] reports an accuracy of 99.05% on the raw dataset, which we were only able to replicate by taking the overall maximum test accuracy out of 5 runs, rather than the average.

	Ours	Veličković	Fey	Knyazev	Gray
Raw	**99.12**±0.06 **99.20**[1]	94.70±0.23 95.01[2]	97.05±0.22 97.45[2]	97.11±0.22 97.44[1]	98.81±0.04 98.86[1]
Hierarchical	**99.04**±0.05 **99.12**[1]		 69.75[2]	98.29±0.21 98.55[1]	98.26±0.09 98.38[1]
Pruned	98.78±0.13 **98.96**[1]		96.95±0.11 97.19[2]		**98.82**±0.07 98.89[1]

levels differently, recognizing that the edge connection is between a child-parent, siblings, etc. as part of its multi-graph convolution approach. None of the other methods differentiate between these edge relationships. The extra nodes have reduced quality information which we believe can act as a confuser to the networks when the connection type is ignored. For SplineCNN [12], the extra information causes extreme overfitting to the training set.

5.1.1 Input edge dropout

To understand how eliminating edge connections through pruning impacts accuracy, we train against the raw MNIST dataset with varied rates of edge dropout applied to the input data. As shown in Figure 4, our method beats the comparison models on the raw dataset for a range of rates.

At dropout rates above 0.6, orphan nodes, with no edges connected to them, become much more likely and above 0.8 such nodes are a significant portion of all nodes. These orphan nodes negatively impact our performance as the only information left to pass forward at each orphaned node is the identity function.

Given that position information is required by our method, an appropriate and effective edge network can be acquired using Delaunay triangulation for 2D graphs. Pruning the MNIST graphs in this way results in approximately a 70% reduction in edges without generating any orphan nodes. Using the pruned dataset, our code performs

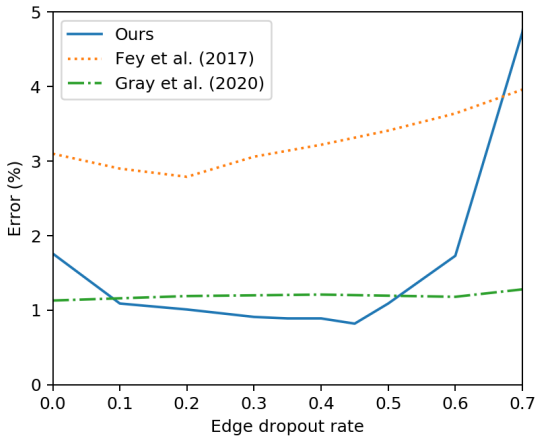

Figure 4: MNIST graph classification error rates as graph edges are dropped on input. Our model performs significantly better for a range of input edge dropout rates. Gray et al. [14] use a technique which ignores the input edges which leads to it being unaffected by edge dropout. SplineCNN [12] sees some benefit from low input edge dropout rates (probably from reduced overfitting).

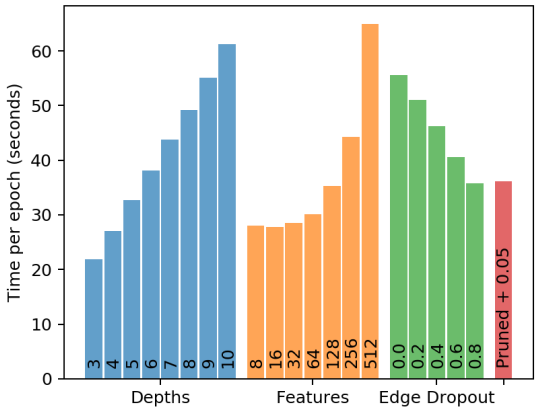

Figure 5: Effect of hyperparameters on training time per epoch performance of our architecture. We modify a default set of parameters — 7 layers deep, 256 initial & max features, no pruning, and 0.5 input edge dropout. Each was run for 5 epochs on an nVidia V100 and average times per epoch are shown.

competitively with prior reported state-of-the-art test accuracy (against models trained and evaluated using the raw dataset). We achieved an error rate of **1.04%** using a 0.1 edge dropout applied on the pruned input graph as shown in Table 1.

Moderate levels of input edge dropout seem to provide a data augmentation effect, leading to greater test accuracy and less overfitting. (Our train and test accuracies remain comparable when sufficient input edge dropout is used.) In all cases, our architectures perform better with some level of input edge dropout than with the original graph. In addition, reduced edge counts have a beneficial impact on training times (Section 5.1.2).

Gray et al. [14] use a method which ignores incoming edge information. Because of this, their accuracy is the same for all levels of input edge dropout and pruning (within their normal variance). While this eliminates any accuracy penalty for overly sparse graphs, it also eliminates the data augmentation and performance benefits of input edge dropout.

5.1.2 Performance

Our network is faster on a per epoch of training time per layer basis. After adding layers to create a deeper, wider, and more accurate network, it remains competitive in training time and superior when looking at test accuracy achieved per training time as shown in Figure 2. Surprisingly, adding extra width to the network did not worsen convergence time until a slight dip is observed going from 256 to 512 features wide. While each epoch takes longer, it converges faster.

We also show the impact of various hyperparameters on our time per epoch performance in Figure 5 . Of note, our pruned dataset running with 0.05 input edge dropout has similar performance to the raw dataset with 0.8 input edge dropout. The pruned dataset has around 70–75% fewer edge connections than in the original which accounts for the increase in speed.

Table 2: CIFAR-100 superpixel image classification test accuracy (%). We show the average and maximum test accuracies for the best training epoch over 32 runs.

Model	Superpixel	Hierarchical
Ours	**40.39**±0.27 **40.71**[3]	**41.16**±0.32 **41.41**[3]
Knyazev	30.46±1.07 31.41[3]	32.34±0.84 33.54[3]
Gray	34.28±0.44 34.88[3]	34.57±0.45 35.19[3]

5.2. CIFAR-100

[1] Test accuracy selected from best training epoch
[2] Best overall test accuracy selected
[3] Test accuracy selected from best training epoch

Table 3: Our results and previously published ISIC Skin Lesion Classification results using deep convolutional neural networks.

Model	Balanced Multiclass Accuracy	Number of Parameters
Inception-ResNet-v2 [2]	**0.692**	56M
SDL [35]	0.681	92M
Ours	0.660	**104k**
ResNet-50 [35]	0.628	23M
DRN-50 [34]	0.626^{+}	23M
Modified VGG-16 [1]	0.622^{+}	138M
GoogLeNet [30]	0.581*	6.4M
VGG-16 [28]	0.529*	138M

* Accuracy reported by Yu et al. [34]
$^{+}$ ISIC 2016 contest submission

The CIFAR-100 dataset [20] consists of 32×32 labeled color images belonging to one of 100 classes. The dataset contains 600 images per class with a 5:1 train:test ratio.

The graph versions of the dataset are obtained by applying SLIC transformations as described in Knyazev et al. [19]. Each graph is constructed with approximately 150 superpixels with node edges restricted to its 32 nearest neighbors by Euclidean distance. The nodes were constructed from levels of approximately 150, 75, 21, and 7 superpixels in the same manner as the hierarchical MNIST dataset (Section 5.1). As with MNIST, classification using the superpixel graphs represents a much more challenging learning task than classification with the original dataset. This is due to the significantly reduced quantity of information available in a superpixel graph relative to a full-resolution image.

The CIFAR-100 experiments use the same architecture as the MNIST experiments, with the exception of additional input channels for color and output channels for the larger number of classes. The CIFAR-100 results are shown in Table 2. We achieved a best accuracy of 40.71% on the raw variant and 41.41% on the hierarchical variant.

5.3. ISIC Challenge 2016

For a more practical application, we evaluate our method on the 2016 International Skin Imaging Collaboration (ISIC) 2016 challenge dataset [15]. This dataset consists of 1279 labeled images of skin lesions (with a 900/379 train/test split). The task is to predict disease state (benign or malignant) given a segmented image.

Our approach to this dataset is unique among the algorithms on the current leaderboard in that we first decompose the image into superpixels and then apply a GCN to derive classification results from the superpixel graph. This is similar to an approach that has recently been proposed for another lesion classification dataset [3]. While we do lose some information in the superpixel transformation, this allows us to significantly reduce our model size. Our network uses substantially fewer parameters than other proposed architectures without sacrificing performance (see Table 3).

Balanced accuracy [7] is used for this dataset because the training data possesses an extreme class imbalance. We use only RGB features for input consistent with the CNN approaches listed in Table 3. However, using additional features has been demonstrated to enhance performance for lesion classification tasks [3, 25] and this is an area of potential future research.

In Table 4 we compare results for this task using various graph convolution layers operating on the superpixel graphs. Our layer provides enhanced accuracy relative to other GCN approaches. The performance of our method (including the superpixel preprocessing step) on this medical dataset bodes well for the applicability of this method to larger datasets where compression of the input data is vital.

Table 4: ISIC Skin Lesion Classification results. Using various graph convolution layers operating on a SLIC superpixel representation. The graph representation allows for significantly smaller networks while our novel convolutional layer enables enhanced accuracy relative to existing alternatives.

Layer	Balanced Multiclass Accuracy
Our layer	**0.660**
GATConv [32]	0.634
TransConv [26]	0.630
ChebConv [9]	0.614

Table 5: FAUST node correspondence results. All results (except ours) as reported by authors. We did not perform an extensive hyperparameter search and our results are mostly to show that the method is generally applicable.

Model	Reported Accuracy
Boscaini et al. [6]	62.4%
Monti et al. [24]	73.8%
Sun et al. [29]	96.9%
Verma et al. [33]	98.7%
Ours	99.20%
Fey et al. [12]	99.20%
Gong et al. [13]	99.8%
Haan et al. [8]	**99.89%**

5.4. FAUST

Finally, we test our method on a shape correspondence task using the FAUST [5] dataset, which contains 100 3D meshes with $6,890$ nodes each depicting 10 scanned human bodies in 10 different poses. Each node corresponds to a particular part of each body and the task is to identify which node corresponds to what body part. We use the standard 80/20 training/test split.

As FAUST is a mesh with no values assigned to each node, it lacks an inherit meaningful field which is required for our pseudo-differential method to work, (see Equation 4.1.1). Since the identity vector is not meaningful we tried adding the position vector and/or the surface normal to each node. The position vector alone performed the best. But there is a degree of translational invariance as everything but the node's identity term has only relative information.

A modified version of the architecture used in the other experiments was used with 3D gradients, 8 layers, 16 initial features doubling each layer to a maximum of 128, and ending with 2 fully connected layers applied to each node separately (dropout applied before each). Scaled exponential linear units are used between each mixing layer. No flattening or pooling operations are used. The final output is a softmax with $6,890$ channels per node.

The model is trained using a batch size of 4, dropout of 0.3, learning rate of 0.01, and cross entropy loss. For this task, input edge dropout causes a significant performance drop and is not used. Our model achieves an accuracy performance comparable with recent results of 99.20% as shown in Table 5.

6. Conclusion

We introduce a simple graph convolutional layer that outperforms every published result we are aware of for graph classification on the MNIST and CIFAR-100 superpixel datasets with faster performance and reduced overfitting tendencies. In addition, we test our new layer on the FAUST shape correspondence and ISIC 2016 challenge lesion classification datasets.

For ISIC 2016 our new layer outperforms other graph convolution layers operating on superpixel graph representations and is significantly smaller than competitive models. We achieve this model size reduction without loss of accuracy by leveraging two innovations: (1) the use of superpixel graph representations of the image providing a significantly compressed version of the input, and (2) our novel graph convolution layers which enable enhanced accuracy on graph classification tasks.

Input edge dropout provides positive impacts on our accuracy except at extreme values. The dropout provides a significant data augmentation effect at even moderate levels as the network sees a combinatoric scale effect. For sparse graphs this had a minimal impact, but it was significant for highly connected graphs.

The performance drop observed when using hierarchical versus raw graphs suggests another area for research. Our current implementation does not take advantage of additional edge information available within the hierarchical dataset. This results in a small but meaningful reduction in accuracy. The additional information seems particularly useful for large image cases (such as CT) where it would be advantageous to inform the network of a broader region. Our convolutional layer could be modified to incorporate this information.

Acknowledgments

Supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

References

[1] Mohit Agarwal, Nandita Damaraju, and Sahbi Chaieb. Deep learning melanoma.

[2] Mohammed A. Al-masni, Dong-Hyun Kim, and Tae-Seong Kim. Multiple skin lesions diagnostics via integrated deep convolutional networks for segmentation and classification. Computer Methods and Programs in Biomedicine, 190:105351, 2020.

[3] Mahmoud H Annaby, Asmaa M Elwer, Muhammad A Rushdi, and Mohamed EM Rasmy. Melanoma detection using spatial and spectral analysis on superpixel graphs. Journal of digital imaging, pages 1–20, 2021.

[4] Rianne van den Berg, Thomas N Kipf, and Max Welling. Graph convolutional matrix completion. arXiv preprint arXiv:1706.02263, 2017.

[5] Federica Bogo, Javier Romero, Matthew Loper, and Michael J Black. Faust: Dataset and evaluation for 3d mesh registration. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3794–3801, 2014.

[6] Davide Boscaini, Jonathan Masci, Emanuele Rodoià, and Michael Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In Proceedings of the 30th International Conference on Neural Information Processing Systems, pages 3197–3205, 2016.

[7] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. The balanced accuracy and its posterior distribution. In 2010 20th international

conference on pattern recognition, pages 3121–3124. IEEE, 2010.

[8] Pim de Haan, Maurice Weiler, Taco Cohen, and Max Welling. Gauge equivariant mesh cnns: Anisotropic convolutions on geometric graphs. arXiv preprint arXiv:2003.05425, 2020.

[9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In Advances in neural information processing systems, pages 3844–3852, 2016.

[10] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. IEEE transactions on pattern analysis and machine intelligence, 29(11):1944–1957, 2007.

[11] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019.

[12] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 869–877, 2018.

[13] Shunwang Gong, Lei Chen, Michael Bronstein, and Stefanos Zafeiriou. Spiralnet++: A fast and highly efficient mesh convolution operator. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, pages 0–0, 2019.

[14] Lindsey Gray, Thomas Klijnsma, and Shamik Ghosh. A dynamic reduction network for point clouds. arXiv preprint arXiv:2003.08013, 2020.

[15] David Gutman, Noel C. F. Codella, Emre Celebi, Brian Helba, Michael Marchetti, Nabin Mishra, and Allan Halpern. Skin lesion analysis toward melanoma detection: A challenge at the international symposium on biomedical imaging (isbi) 2016, hosted by the international skin imaging collaboration (isic), 2016.

[16] Yuzhu Ji, Haijun Zhang, Zhao Zhang, and Ming Liu. Cnn-based encoder-decoder networks for salient object detection: A comprehensive review and recent advances. Information Sciences, 546:835–857, 2021.

[17] Chiyu Max Jiang, Jingwei Huang, Karthik Kashinath, Prabhat, Philip Marcus, and Matthias Niessner. Spherical CNNs on unstructured grids. In International Conference on Learning Representations, 2019.

[18] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2016.

[19] Boris Knyazev, Xiao Lin, Mohamed R Amer, and Graham W Taylor. Image classification with hierarchical multigraph networks. arXiv preprint arXiv:1907.09000, 2019.

[20] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[21] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. IEEE Transactions on Signal Processing, 67(1):97–109, 2018.

[22] Claudio Mancinelli, Marco Livesu, and Enrico Puppo. A comparison of methods for gradient field estimation on simplicial meshes. Computers and Graphics, 80, 03 2019.

[23] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In Proceedings of the IEEE international conference on computer vision workshops, pages 37–45, 2015.

[24] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

[25] RD Seeja and A Suresh. Melanoma classification employing inter neighbor statistical color and mean order pattern texture feature. Multimedia Tools and Applications, pages 1–20, 2021.

[26] Yunsheng Shi, Zhengjie Huang, Shikun Feng, and Yu Sun. Masked label prediction: Unified massage passing model for semi-supervised classification. arXiv preprint arXiv:2009.03509, 2020.

[27] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. CoRR, abs/1704.02901, 2017.

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[29] Zhiyu Sun, Ethan Rooke, Jerome Charton, Yusen He, Jia Lu, and Stephen Baek. Zernet: Convolutional neural networks on arbitrary surfaces via zernike local tangent space estimation. In Computer Graphics Forum, volume 39, pages 204–216. Wiley Online Library, 2020.

[30] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1–9, 2015.

[31] John Tencer and Kevin Potter. A tailored convolutional neural network for nonlinear manifold learning of computational physics data using unstructured spatial discretizations. SIAM Journal on Scientific Computing, 43(4):A2581–A2613, Jan 2021.

[32] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. arXiv preprint arXiv:1710.10903, 2017.

[33] Nitika Verma, Edmond Boyer, and Jakob Verbeek. Feastnet: Feature-steered graph convolutions for 3d shape analysis. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2598–2606, 2018.

[34] Lequan Yu, Hao Chen, Qi Dou, Jing Qin, and Pheng-Ann Heng. Automated melanoma recognition in dermoscopy images via very deep residual networks. IEEE transactions on medical imaging, 36(4):994–1004, 2016.

[35] Jianpeng Zhang, Yutong Xie, Qi Wu, and Yong Xia. Medical image classification using synergic deep learning. Medical image analysis, 54:10–19, 2019.

[36] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. Computational Social Networks, 2019.