Comput. Aided Geom. Des. 128 (2026) 102571

Contents lists available at ScienceDirect

Computer Aided Geometric Design

journal homepage: www.elsevier.com/locate/cagd

Geometric multigrid neural networks

Jackson Campolattaro [a,*], Ruben Wiersma [b], Klaus Hildebrandt [a]

[a] Delft University of Technology, Delft, Netherlands
[b] ETH Zurich, Zürich, Switzerland

HIGHLIGHTS

- Geometric multigrid architecture designed for point cloud learning tasks.
- Native support for multi-resolution representations of the domain.
- Multigrid convolutions enable long-range information flow.
- Delivers higher expressiveness with faster, more efficient training.

ARTICLE INFO

Keywords:
Neural networks
Shape analysis
Multigrid
Point clouds

ABSTRACT

We introduce Geometric Multigrid Neural Networks (GMNN), a novel network structure for geometric deep learning on point clouds and surfaces. Convolutional neural networks face a common challenge: how can relevant features be communicated over longer distances? Our architecture facilitates long-distance communication with Geometric Multigrid Convolution (GMC) blocks, which apply convolutions in parallel to features defined on each level of a multigrid representation of the surface, and enable communication all the way up and down the hierarchy. We observe two major structural advantages of such a network: First, because each GMC operates on all levels of the multigrid hierarchy, even early stages can make use of coarse-scale information and receptive field grows rapidly with depth. Second, networks built with this backbone have the freedom to route information between different scales, including in ways not possible for other architectures. Because of these advantages, we find that a GMNN can combine the fast convergence of a shallow network with the greater expressiveness of a deeper, larger network. We build a GMNN from the components of a state-of-the-art U-Net, and find that on real tasks it can match or exceed the accuracy of the base network while using fewer epochs and roughly half the parameter count.

1. Introduction

Advances in 3D capture and modeling technologies have made geometric data increasingly accessible, with applications spanning from computer graphics and medical imaging to engineering and manufacturing. As a result, the analysis and processing of geometric data have become key problems, with advances benefiting a wide range of use cases. Over the past decade, successful deep learning techniques from the image and language domains have been adapted to geometric data. These geometric deep learning methods have achieved breakthroughs for various challenging problems in 3D data processing and analysis.

Convolutional neural networks improve efficiency and introduce useful priors by using local operators. However, this localization comes with limitations: multiple layers are required to integrate information across distant regions of the domain and to

* Corresponding author.
Email addresses: J.R.C.Campolattaro@tudelft.nl (J. Campolattaro), rwiersma@ethz.ch (R. Wiersma), K.A.Hildebrandt@tudelft.nl (K. Hildebrandt).

https://doi.org/10.1016/j.cagd.2026.102571
Received 9 March 2026; Received in revised form 15 April 2026; Accepted 5 May 2026
Available online 12 May 2026

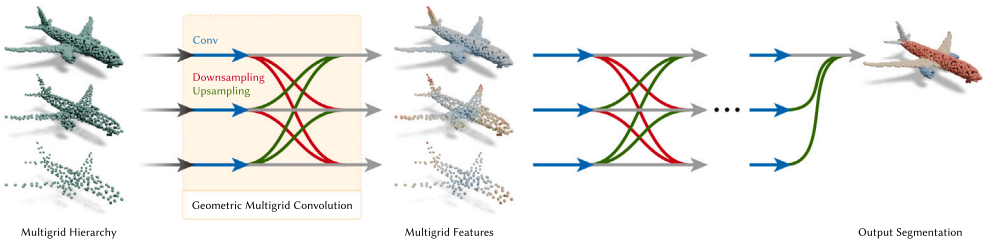


Fig. 1. Geometric multigrid convolutions operate on features defined on the levels of a multigrid. This allows Geometric Multigrid Neural Networks to extract features at a range of scales, starting at the earliest layers, and to efficiently fuse information both across scales and between spatially distant regions on the surface.

capture coarse-scale features. Several strategies have been proposed to address this, including augmenting networks with non-local, low-frequency Laplace eigenfunctions (Sharp et al., 2022), introducing additional non-local connections based on feature-space proximity (Wang et al., 2019), and applying convolutions to progressively coarser point clouds (Qi et al., 2017), like a U-Net Ronneberger et al. (2015). Despite these advances, these architectures still limit how and when features at different spatial locations and across scales are combined to a prescribed set of connections. For example, U-Net first connects nearby points and only allows far-away points to interact in deeper layers, while DiffusionNet's use of eigenfunctions is limited to low frequencies, as they must be numerically approximated.

We introduce a novel architecture that addresses this limitation: the Geometric Multigrid Neural Network (GMNN, Fig. 1). GMNNs efficiently represent information over multiple scales by retaining features on the levels of a multigrid hierarchy through all layers of the network. Levels are connected through Geometric Multigrid Convolution (GMC) blocks, which perform convolutions within individual hierarchy levels and perform pooling and unpooling across levels, enabling effective extraction and integration of multiscale features. GMNNs are constructed by chaining GMC blocks in series.

This architecture enables features at multiple scales to be extracted starting from the first layer, and to be combined across scales at every convolution. As a result, GMNNs can learn flexible and efficient strategies for integrating information across scales and spatial extents throughout the network, without being subject to the constraints imposed by prior architectures. For example, comparing GMNN with a U-Net (Ronneberger et al., 2015) highlights fundamental differences. In a U-Net, information follows a fixed trajectory: features are first extracted at fine scales and then progressively aggregated at coarser levels. In contrast, GMNN offers greater flexibility. It can follow the same fine-to-coarse pathway, but it can also exploit alternative routes, such as coarse-to-fine or paths that traverse up and down the multigrid hierarchy multiple times.

We observe that GMNN is capable of higher performance on realistic tasks. GMNN matches or surpasses state-of-the-art accuracy on benchmark segmentation datasets while using roughly half the number of parameters. The smaller parameter count is spread over many narrow convolutions, so forward and backward passes are marginally more time and memory intensive than in a comparable U-Net. In practice, GMNN converges in fewer epochs and inference is dominated by data preparation, so training time and latency remain comparable to the baseline.

To better understand possible sources of GMNN's advantage, we set up experiments to quantify the receptive field and the expressiveness of GMNN. Each GMC block links features at a given level to coarser-level features from the preceding layer, causing the receptive field to expand rapidly. We measure this effect and compare GMNN to alternative architectures to highlight the accelerated growth of its receptive field. The ability of GMNN to organize the information flow translates into greater expressive power in a synthetic function approximation task.

Beyond the performance advantages, we view GMNN as a novel approach for network design on point clouds. While the implementation of multigrid neural networks is naturally complex, our modular approach makes them straightforward to design and build, opening the door to further architectural exploration.

2. Related work

Here, we provide a summary of existing architectural solutions to enable long-range communication in convolutional neural networks, starting with other domains, and then focusing on meshes and point clouds.

Multiscale architectures for images and non-geometric graphs. The U-Net, first introduced in Ronneberger et al. (2015), enables segmentation of large images with fewer convolutions by operating on progressively coarser (lower resolution) domains before expanding to progressively finer domains. Multigrid Neural Architectures (Ke et al., 2016) demonstrated the potential of multigrid convolutions on the image domain. We adopt the same 'Multigrid' terminology in this paper. The authors augment well-known architectures with multigrid convolutional layers and demonstrate accuracy improvements on common segmentation benchmarks while retaining efficiency in parameters and compute. Feature Pyramid Networks (Lin et al., 2017) (FPN) construct features on multiple scales and apply convolutions to those scales; features from all scales are merged at the end of the network before producing labels. UNet++ (Zhou et al., 2018) extends the U-Net architecture by keeping the convolutions in higher resolutions, where the original U-Net would downsample and only continue in the lower resolution, proverbially 'filling in' the U-shape. They also connect adjacent scales in each layer. Multigrid Graph Neural Networks (Taghibakhshi et al., 2023) (MG-GNN) adapt a limited multigrid architecture to the graph learning domain, and show that a two-level multigrid structure has benefits for graph-partitioning tasks. Closest to our approach,

IM-MPNN (Finder et al., 2025) implements a more complete multigrid architecture for graph learning, with 4 scales and communication between adjacent scales. It confirms advantages on large-diameter graph problems, where communication across the original graph requires many convolutions. Each of these approaches shows that variations on the U-Net structure can benefit deep learning on other domains; our architecture incorporates some of these ideas for use on point clouds.

U-nets on point clouds. The structure of PointNet++ (Qi et al., 2017), a pioneering method for learning on point clouds, resembles that of a U-Net (Ronneberger et al., 2015). In the 'encoder,' the features in the point cloud are progressively downsampled with farthest point sampling (FPS) followed by max-pooling in a local neighborhood (KNN or ball query). The 'decoder' then upsamples the features with linear interpolation. Features from the encoder are connected to the decoder using skip-connections between corresponding scales. This base structure has been adapted and developed by many follow-up works. Among many others, PointCNN (Li et al., 2018), KPConv (Thomas et al., 2019), KPConvX (Thomas et al., 2024), PointTransformer v2 (Wu et al., 2022) and v3 (Wu et al., 2024) also employ U-Net architectures. The main contributions of these works lie in improving the convolution operator, e.g., with a graph- or point-based variant of convolution or attention (transformer). While the works also change the architecture, e.g., by adding residual connections, they do not fundamentally alter the U-Net structure. More recently, PointNeXT (Qian et al., 2022) and DeLA (Yang et al., 2024), show that adopting these architectural changes and training procedures with PointNet-style convolutions can lead to state-of-the-art results. Our work is orthogonal to the contributions on the convolution operators, as we explore different ways to connect scales in the hierarchy, regardless of the operations within the scales. Therefore, we employ simple building blocks, such as the MLPs and max-aggregation used in PointNet++ and DeLA. In practice, the insights from our work could be combined with different blocks such as graph-based, convolution-like or transformer-style layers in a neural network for point clouds.

Alternative architectures on point clouds. On point clouds, works like DGCNN (Wang et al., 2019) and DiffusionNet (Sharp et al., 2022) have explored non-architectural ways to connect information at longer ranges, but fewer works have explored alternatives to the U-Net. Multiresolution Tree Networks (Gadelha et al., 2018) (MTN) adopt a multigrid architecture (Ke et al., 2016) for point clouds. Rather than working natively in 3D, they use spatial sorting to represent point clouds as a 1D structure and apply a 1D convolution on the point clouds. They maintain 3 scales of a multigrid through much of the network, but eventually pool all scales to a global node (to enable use as an auto-encoder). PointHR (Qiu et al., 2023) resembles the augmentation of UNet++, 'filling in' convolutions in the higher resolutions throughout the network. Our architecture goes beyond both MTN and PointHR by performing convolutions on all resolutions throughout the network; at no point does it operate exclusively on coarse scales (as MTN does at the end of its encoder) or exclusively at fine scales (as PointHR does at the beginning of its encoder). It also adds the ability to communicate to non-adjacent scales with progressive upsampling and downsampling, an ability not present in MTN, PointHR, or other enhanced U-Nets.

Neural multigrid solvers. We wish to distinguish GMNN from previous works such as Margenberg et al. (2022) and Greenfeld et al. (2019), which aim to improve the performance of multigrid solvers for partial differential equations by incorporating neural components. While our design shares the use of a multigrid representation with these methods, it aims to leverage the multigrid representation for learning from geometric data, for example in segmentation tasks, as opposed to accelerating multigrid solvers.

3. Method

Our goal is to design a network (GMNN) that adapts the communication between finer and coarser scales as needed during training, rather than following a predefined path. This is achieved with a multigrid feature representation and a geometric multigrid convolution block. We define generic versions of these in this section, with sampling procedures, convolution operators, and other components left unspecified. In Section 4, we instantiate GMNNs for each task by adapting components from other networks.

3.1. Multigrid features

Throughout a GMNN, features are defined over a 'multigrid' representation of the input domain. Given an input point cloud $P^1 \in \mathbb{R}^{N\times 3}$ and a desired number of levels S, we construct a sampling hierarchy of point clouds, $\{P^1, \dots, P^s, \dots P^S\}$, with point counts N^s. Each sampling P^s represents one level of the multigrid hierarchy. The point cloud hierarchy is associated with multigrid features: a set of C features $X^s \in \mathbb{R}^{N^s\times C}$ per level P^s.

3.2. Geometric multigrid convolution blocks

A Geometric Multigrid Convolution (GMC) Block enables multigrid connectivity by composing convolutions with upsampling and downsampling stages to connect different levels of the multigrid hierarchy.

Parallel convolutions. Point cloud 'convolutions', such as the operations in PointNet++ (Qi et al., 2017), are applied to each level in parallel to produce a new set of multigrid features $X^{s'}$, as shown in Fig. 2(a). These convolutions have independent parameters and are not shared, because we expect that different types of features can be found at different levels.

Transfers between scales. We can use downsampling and upsampling to match spatial resolutions and communicate the new features $X^{s'}$ to the adjacent levels $s+1$ or $s-1$ (Fig. 2b).

We aim to design a transfer block that can communicate between each pair of multigrid levels. Connecting each level to all higher and all lower levels naively would require $S(S-1)/2$ upsampling operations and $S(S-1)/2$ downsampling operations. Instead,

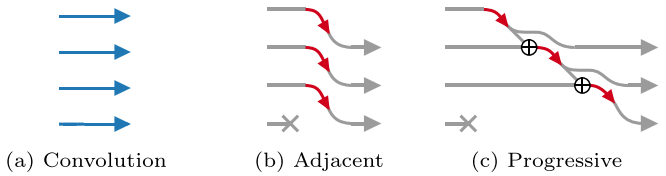


Fig. 2. Multigrid operations on 4 scales. Parallel convolution (left), adjacent and progressive downsampling (right). Upsampling operations are the inverse of downsampling.

we achieve this universal connectivity more efficiently by performing transfers progressively, rather than in parallel. This reduces our requirement to $S-1$ upsampling operations and $S-1$ downsampling operations (Fig. 2c). Progressive upsampling produces new features on all levels except for the coarsest level (S), and downsampling on all except for the finest level (1). There are many options for incorporating the level-transferred features before the next stage. In our implementations, we use a linear layer to match dimensionality and add them to the features of each level.

Transfer layouts. Given components for communicating features within each level (parallel convolutions) and between levels (progressive upsampling, downsampling), there is a degree of freedom in how we arrange the connections. We explored arrangements which alternate between convolutions and transfers, and found that a sufficiently deep GMNN is not strongly sensitive to their ordering. For most tasks, we use a GMC block that first performs progressive downsampling, then upsampling, and finally applies parallel convolutions.

3.3. Geometric multigrid neural networks

A GMNN is made up of several multigrid convolution layers, each with the same number of levels S, placed in series (Fig. 1). Because the neighborhood queries used in convolutions and pooling operations are used in every block of the network, we precompute the point hierarchy and neighborhoods once and reuse them throughout. A GMC block expects input features on all levels. For the first layer, this can be achieved by extracting local shape information independently on each level or by progressively downsampling the finest-level features X^1 from the input point cloud P^1 to generate representations for all coarser levels.

For segmentation, progressive upsampling is applied at the end of the network, ensuring features learned at all levels contribute to the final output on the finest level. After this, a conventional segmentation head (point-wise MLP) is used. Classification can be done with the reverse of this approach—ending with progressive downsampling to collect all features to one node per point cloud, and then applying an MLP-based classification head to the features of that point.

3.4. Properties

The GMNN architecture allows information to be represented at multiple scales within the multigrid hierarchy of each layer, with GMC blocks extracting and combining features across these scales. Unlike other networks, where information flow is prescribed—such as the fine-to-coarse progression in U-Nets—GMNN can flexibly organize information flow to suit the learning task. We discuss two major advantages of a GMNN over other architectures: First, its convolutions have larger receptive fields, stemming from an ability to communicate information across long distances with fewer intermediate steps (Section 3.4.1). Second, a GMNN offers a greater ability to route information flexibly between levels (Section 3.4.2).

Similar benefits could be obtained by building a deeper conventional model; however, an overly long gradient path can increase training time and exacerbate problems such as vanishing gradients. By arranging its convolutions in parallel, a GMNN obtains some advantages of deeper networks while avoiding these drawbacks.

3.4.1. Receptive field

The first stage of a U-Net allows information to travel only along the finest level of multigrid hierarchy, but the first stage of a GMNN can propagate signals much further using the coarser levels. In Section 4.1, we measure this property for several model architectures and show the advantages of a GMNN. In Section 4.2 and Appendix A.1, we find that this greater connectivity allows the network to converge quickly. This difference is intuitive because a U-Net must learn to carry useful information on the fine scales deeper into the network in order to propagate it further, whereas in a GMNN each layer after the first has direct access to information from longer distances.

3.4.2. Information routing

A U-Net has a prescribed path for how information must flow through a network: first between fine-level nodes, then progressively coarser. In contrast, a GMNN allows for the free exchange of information between levels. A sufficiently deep GMNN is a superset of a U-Net; it contains the fine-coarse-fine route, but also more complex routes, including the ability to make multiple trips from fine to coarse and back. Because of this, a GMNN can organize the communication of features between scales according to the specific characteristics of the learning target. In Section 4.4, we find that this has benefits for real segmentation tasks, on which our architecture competes with state-of-the-art models while using only half the parameters. In Section 4.3, we show how this translates to higher accuracy on a difficult synthetic function approximation task designed to measure expressiveness. We note that in our

experiments the GMNN tends to achieve much higher accuracy on the training set than the base U-Net, even when its advantage on the validation set is small. This indicates possible overfitting, which is expected of a more expressive model. This could be specific to factors like the complexity of the task or the size of the dataset. Mitigating this with standard regularization techniques could further improve validation accuracy.

4. Experiments

When comparing architectures, we consider three types of networks: a flat network, which simply chains convolutions on the input (finest) level; a U-Net, which is a typical depth-scaled design, applying blocks of convolutions to each level in series from finest to coarsest, followed by a simple MLP-based upsampling stage; and GMNN, which chains GMC blocks, described in the previous section. Unless otherwise stated, these networks all use PointNet++-style convolutions. U-Net and GMNN networks are used with identical hierarchy-construction procedures, so the message-passing is done over equivalent nodes and edges on each level.

4.1. Receptive field

We expect that early access to coarse levels provides GMNN with a structural advantage in expanding its receptive field. Consider the set of all connections between nodes on the levels of a multigrid. Each stage of a U-Net can only make use of a single subset of these connections: convolutions or downsampling connections early in the network, upsampling connections at the end. Each block of a GMNN can make use of the entire set.

This advantage is quantified by evaluating the receptive fields of different convolutions within a network. Receptive fields are estimated by tracking how information propagates from a single starting node across the domain. As a test case, we use the Stanford bunny mesh, chosen for its relatively low connectivity, with convolutions applied over the original mesh edges. Coarser levels are generated using 50% FPS sampling. Downsampling relies on edges from the fine scale, while upsampling connections are formed via nearest-neighbor clustering. Convolutions operate on the connectivity between clusters. Starting from node 0, we record the depth at which each node becomes reachable.

Fig. 3 illustrates how the receptive field grows with network depth for different architectures. In the flat network, communication is restricted to the finest scale, so the receptive field remains small. The U-Net performs better; while its early convolutions behave like those of the flat network, later stages operate on coarser meshes and therefore reach further. In contrast, the GMNN achieves rapid growth in receptive field even at shallow depths, since its early stages already include convolutions on coarse scales. The accompanying images show the receptive field of a convolution at the green-marked node after 8 stages. In the flat network, 8 fine-scale convolutions cover only a limited range. In the U-Net, 4 fine scales and 4 secondary scales moderately extend the range. In the GMNN, 8 GMC blocks—each operating on all scales—enable long-range communication.

4.2. Fast convergence

We expect that broader access to non-local information in a GMNN allows it to more readily discover useful ways in which features from different parts of the domain can be combined. In practice, this implies that a GMNN can achieve higher accuracy in fewer epochs than comparable architectures. This trend is evident in the early stages of long training runs (see Appendix A.1), and is even more pronounced under accelerated training schedules.

We build several models of different architectures, each with exactly 16 PointNet++-style convolutions (the GMNN contains 4 GMC blocks) and channel counts chosen so that all have approximately 0.9 M parameters. We use a similar training procedure to DeLA and PointNeXT, except for the learning rate schedule which is compressed to 10 epochs, a fraction of the normal training time. We find that on ShapeNet (Table 1), both the U-Net and GMNN converge quickly on common classes, but the GMNN converges to a much higher accuracy on less common classes (Cat. IoU). On the larger point clouds of S3DIS (Table 2), where information must be communicated further, this gap widens. The U-Net struggles to reach high accuracy in such a short training cycle. GMNN reliably reaches 10% higher overall accuracy (OA) than the U-Net, with even wider margins on uncommon classes (mIoU, Cls. Acc).

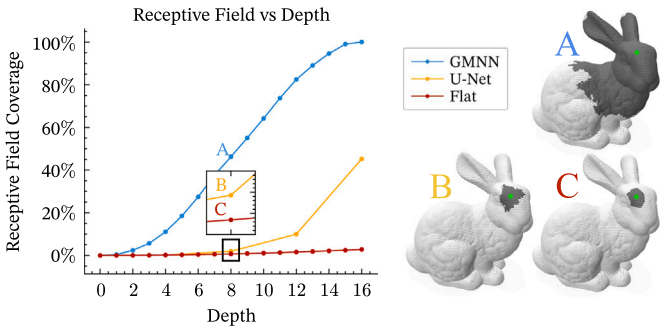


Fig. 3. Left: receptive field of models at different depths. Right: receptive fields at a depth of 8; query node marked in green, contributing nodes in black (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.).

Table 1
ShapeNet, 10 epochs.

Model	Ins. IoU	Cat. IoU
Flat	82.1(0.1)	77.8(0.1)
U-Net	84.5(0.1)	80.2(0.2)
GMNN	**84.7(0.1)**	**81.3(0.3)**

Parameter-count-matched networks with different backbones. Other hyperparameters are identical between models. Three trials were used for each run.

Table 2
S3DIS, 10 epochs.

Model	mIoU	Cls. Acc	OA
Flat	*(Omitted, high memory requirements)*		
U-Net	39.0(0.3)	46.5(0.4)	78.0(0.3)
GMNN	**63.0(0.7)**	**71.2(0.2)**	**88.0(0.3)**

Parameter-count-matched networks with different backbones. Other hyperparameters are identical between models. Three trials were used for each run.

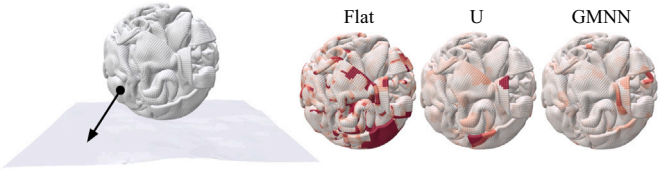


Fig. 4. Paper un-crumpling task (left). Localization of errors for different architectures (right).

Table 3
Paper Un-crumpling Task, after 10,000 training iterations.

Model	MAE	MSE
Flat	1.69(0.21)	20.4(1.6)
U-Net	0.67(0.14)	2.0(1.2)
GMNN	**0.45(0.02)**	**0.9(0.1)**

4.3. Expressiveness

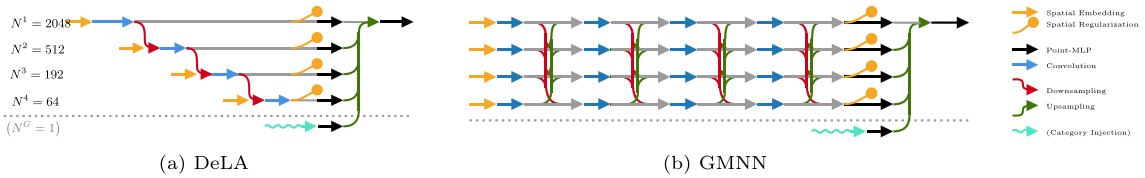
We observe that GMNN can outperform other architectures on real segmentation tasks with fewer parameters (Section 4.4). One explanation for this is increased expressiveness stemming from the variety of information paths provided by our architecture. That is, given similar parameter counts, a GMNN can express more complex functionality than a flat network or a U-Net. We can evaluate this flexibility with a difficult function approximation task. For this purpose, we simplify an expressiveness test used in Maesumi et al. (2025): Given a large mesh of a crumpled ball of paper, we train models to map points to their locations on the original flat sheet (Fig. 4). Relative point positions ($p_j - p_i$) are provided by the PointNet++-style convolutions to ensure that the network does not simply learn a mapping in the embedding space, but learns from local surface features.

We adapt the architectures from Section 4.2, configured so that each has just over 0.5 M parameters. The dataset consists of a single mesh of approximately 150,000 vertices (IndefinGaming (2021)). Convolutions are performed using neighborhoods based on edges of the original mesh, and for the U-Net and GMNN the hierarchy is produced using a similar approach to Section 4.1. In Table 3 we find that switching from a U-Net to a GMNN reduces absolute error by over 30% vs. the U-Net, and squared error by more than a factor of two.

4.4. Comparisons

Our goal is to test the effect of multigrid connections when augmenting existing networks. To simplify the analysis, we start from a PointNet-style (MLP followed by maximum aggregation) U-Net. In our comparisons against state-of-the-art, we use DeLA (Yang et al., 2024) as the base network, replicated in our own codebase, and augment it by placing its convolutions into a series of GMC layers. We denote the DeLA architecture as U-Net and our augmented variant as GMNN.

The original U-Net backbone for DeLA part-segmentation is shown in Fig. 5(a). It uses a single set of PointNet-style convolutions at the start of the network to embed spatial information (yellow arrows). This information is passed through convolutions (blue) containing efficient 'decoupled' local aggregations, downsampling (red), and a spatial regularization step (yellow dots) that computes a loss encouraging the network to preserve spatial information throughout. Finally, point-wise MLPs convert the features on all



(a) DeLA (b) GMNN

Fig. 5. Architecture comparison. The part-segmentation variant of DeLA (U-Net) and the same convolutions incorporated into a GMNN. Each arrow represents an operation with input and output features (channel counts left unspecified) on the points of the network. Where lines meet, features are added as residuals.

Table 4
Segmentation results on ShapeNet-Part.

Method	Ins. IoU	Cat. IoU	Params.
PointNet++ (Qi et al., 2017)	85.1	81.9	1.0 M
PointNeXT-S (C=64) (Qian et al., 2022)	86.9	84.8	3.7 M
PointNeXT-S (C=160) (Qian et al., 2022)	87.0	85.2	22.5 M
PTv1 (Zhao et al., 2021)	86.6	83.7	7.8 M
SPoTr (Park et al., 2023)	87.2	85.4	1.7 M
AVS-Net (Zhang et al., 2023)	87.3	85.7	–
PointHR (Qiu et al., 2023)	87.2	–	7.4 M
DeLA (Yang et al., 2024)	87.5	**86.0**	7.0 M
GMNN	87.6	85.8	3.0 M
GMNN (Scaled)	**87.7**	**86.0**	8.1 M

scales to the same channel count, so that a progressive upsampling step can add all outputs before the segmentation head. For part-segmentation, category labels are also provided as features on a global node.

ShapeNet-part segmentation. To assess part segmentation performance, we use the ShapeNet-Part dataset (Wu et al., 2015), a common benchmark that consists of 13 categories of clean object point clouds (e.g., plane, chair) divided into 50 total parts (e.g., wing, seat). As input, the model takes point coordinates, normals, and category labels (provided just before the segmentation head). As output, the model produces logits indicating the associated part for each point. We configure GMNN with the same number of channels as DeLA on the finest scale, and set the number of channels on the other scales to produce a small version of the model and a wider version with more parameters. Both models have a depth of 4 GMC layers, with half as many local aggregations per point convolution, so that the effective model depth is shallower than that of DeLA. We find that this task is prone to overfitting with deep networks. To mitigate, we make GMNN shallower without reducing parameter count by placing upsampling and downsampling blocks in parallel. For fair comparison, we use the same sampling procedures, dataset regularization, train/test split, and postprocessing techniques as used by PointNeXT and DeLA. The training cycle is reduced to 150 epochs (vs. 250) with the learning rate schedule adjusted to match. While DeLA requires the full training schedule to achieve its reported accuracy, we find that the shorter schedule is sufficient for GMNN to saturate, and additional epochs do not improve our results.

Table 4 places the performance of our adapted models in context with the original DeLA and other state-of-the-art approaches. We find that the smaller multigrid model matches DeLA with half the parameters. Scaling the model allows it to exceed DeLA on common categories. Both versions of the model also outperform state-of-the-art approaches, including modern transformers like SPoTr, AVS-Net, and PointHR.

S3DIS semantic segmentation. We measure point cloud semantic segmentation performance on S3DIS (Armeni et al., 2016), a commonly used indoor-segmentation benchmark. It consists of high-resolution scans of rooms from 6 different parts of the Stanford offices. In line with common practice, we measure accuracy on Area 5. The trained model takes positions and RGB colors for each point and produces logits assigning the points to one of 13 classes (e.g., wall, chair, door). For this task, we configure a version of GMNN with the same base channels as DeLA and six shallow GMC layers, again with half as many local aggregations to create a shallower model. We keep the same training procedures as DeLA and PointNeXT, including dataset regularization, point cloud voxelization (with interpolation), and training scene cropping. As on ShapeNet, we find that the training schedule can be shortened for GMNN, in this case from 100 epochs to 75.

Table 5 places our results in context. We see that with a narrower, shallower network for half the total parameters, we match the accuracy of DeLA (OA) and significantly outperform it on uncommon classes (mIoU, Cls. Acc). Our network's performance improves over all state-of-the-art point-transformer models, given the same training data, despite using substantially fewer parameters (1/35th) and orders of magnitude fewer neighbors per node (24 vs 1024) in the case of PointTransformer v3. We also include the results for PTv2 and PTv3 with pretraining for reference, but note that these results should not be directly compared to the other methods in the table, as they include extra training data.

Classification tasks. To assess classification performance, we use the architecture described in Section 3.3: the bulk of the network is the same as a segmentation network, but before the head we perform progressive pooling in place of unpooling. We evaluate the classification model on two commonly used datasets: ModelNet40 is a basic benchmark that consists of 40 categories of clean object

Table 5
Segmentation results on S3DIS Area 5, ++ indicates pretraining on extra data was used.

Method		mIoU	Cls. Acc	OA	Params.
PTv2 Wu et al. (2022)	++	72.7	78.0	91.6	12.8 M
PTv3 Wu et al. (2024)	++	74.3	80.1	92.0	124.8 M
PTv3 + Sonata Wu et al. (2025)	++	76.0	81.6	93.0	124.8 M
PointNet++ Qi et al. (2017)		53.5	–	83.0	**1.0 M**
PointNeXT Qian et al. (2022)		71.1	77.2	91.0	41.6 M
KPConv Thomas et al. (2019)		67.1	72.8	–	14.9 M
PTv1 Zhao et al. (2021)		70.4	76.5	90.8	4.9 M
PTv2 Wu et al. (2022)		71.6	77.9	91.1	12.8 M
PTv3 Wu et al. (2024)		73.4	78.9	91.7	124.8 M
PointHR Qiu et al. (2023)		73.2	78.7	91.8	–
DeLA Yang et al. (2024)		74.1	80.0	**92.2**	7.0 M
GMNN		**74.4**	**80.8**	92.1	3.4 M

Table 6
Classification results.

Method	Params.	ModelNet40		ScanObjectNN	
		OA (%)	mAcc (%)	OA (%)	mAcc (%)
PointNet++ Qi et al. (2017)	1.5 M	91.9	–	77.9	75.4
PointNeXT Qian et al. (2022)	1.4 M	93.2(0.1)	90.8(0.2)	87.7(0.4)	85.8(0.6)
KPConv Thomas et al. (2019)	14.3 M	92.9	–	–	–
PTv3 Wu et al. (2024)	–	–	–	86.4	83.9
PointHR Qiu et al. (2023)	–	93.9	–	–	–
PointMLP Ma et al. (2022)	13.2 M	**94.1**	91.3	85.4(1.3)	83.9(1.5)
DeLA Yang et al. (2024)	5.4 M	94.0	**92.2**	90.4	89.3
GMNN	4.7 M	93.6	90.9	**90.6**	**89.6**

Table 7
Grid-transfer connectivity ablation on S3DIS, 75 epochs.

Connectivity	mIoU
None	71.0(0.1)
Adjacent	72.3(0.1)
Progressive	**72.6(0.2)**

point clouds (e.g., plane, chair) sampled from meshes. ScanObjectNN, is a more difficult task, consisting of 15 categories of real-world object scans (e.g., bag, door). We use the most difficult variant of ScanObjectNN (PB_T50_RS), in which the objects are randomly rotated and backgrounds are included in the scan.

As with ShapeNet, we configure GMNN with the same number of channels as DeLA on the finest scale, and choose the number of channels on the other scales to produce the desired parameter count. The model chains 4 GMC layers with half as many local aggregations per point convolution, for a relatively shallow effective depth. We again use the same data preprocessing and hierarchy as DeLA; in this case this means only three multigrid levels are used.

Results are listed in Table 6: On ScanObjectNN, the results exceed DeLA and other state-of-the-art approaches, particularly on the less common classes (mAcc). On the simpler (clean, aligned) ModelNet40 dataset, the GMNN typically reaches 100% accuracy on the training set earlier than DeLA does, but fails to outperform it on the test set. Additional tuning or regularization may enable better performance on this task.

4.5. Transfer connectivity ablation

A novel aspect of GMNN is its use of progressive upsampling and downsampling stages to enable full connectivity between scales, as shown in Fig. 2(c). We can isolate the advantage of this by training networks with connections only between adjacent scales, as in Fig. 2(b). For reference, we also compare against networks that have no communication between scales, aside from the final pooling stage. This is similar to a feature pyramid network (Lin et al., 2017).

We set up an experiment on S3DIS with modified versions of a GMNN using DeLA convolutions. Table 7 shows how the connectivity between scales affects accuracy. We find that connections between scales are critical for producing an accurate network. Using progressive connections all the way up and down the network adds a consistent improvement over adjacent connections in mIoU at negligible additional computational or parameter cost.

Table 8
Training and inference times.

Dataset	Method	Training Time			Inference Latency		
		Epoch		Overall	Data Prep.	Forward	Overall
S3DIS	DeLA	173s	×100	**288m**	302ms	19ms	**321ms**
	GMNN	281s	×75	351m	302ms	50ms	352ms
ShapeNet	DeLA	59s	×250	246m	791ms	17ms	**808ms**
	GMNN	82s	×150	**205m**	791ms	43ms	834ms

4.6. Speed

A GMNN configuration may use more convolutions than competing architectures, like U-Net, especially on the finest scale. Edges can be precomputed and convolutions are performed in parallel, reducing runtime, but some additional cost remains. In Table 8 we measure the speed and latency of our best network against a re-implementation of DeLA (using the same convolutions).

Training time. On S3DIS we see that, though each epoch takes approximately 62% longer, the accelerated training schedule with fewer epochs narrows this difference to approximately 20%. On the smaller meshes of ShapeNet, the difference in epoch time is only 39%, and with the accelerated schedule, the GMNN is trained 17% faster.

Inference latency. We measure the average latency of GMNN and DeLA over all forward passes on the test set. Voting is disabled, and data preparation (including hierarchy construction and edge precomputation) is shared between models. Both DeLA and GMNN are relatively small models with well-optimized convolutions, and we find that inference passes are dominated by the data preparation stage. On S3DIS, GMNN is approximately 10% slower than DeLA and on ShapeNet the difference narrows to 3%.

5. Conclusion

In this work, we introduce Geometric Multigrid Neural Networks (GMNN), a novel architecture that addresses key limitations in existing architectures for learning on point clouds. By operating on novel multigrid features with multigrid convolutional blocks, GMNN enables the representation and integration of features across multiple spatial scales within each layer, facilitating early coarse-scale processing and efficient information exchange across the point cloud. We show that GMNN can be constructed from common components of other architectures and demonstrate its effectiveness through experiments. Our results show consistent improvements in performance, reduced parameter counts, faster convergence during training, and more efficient information propagation compared to state-of-the-art architectures such as U-Nets.

We view GMNN as a fundamentally novel framework for neural network design on point clouds. As our paper aims to isolate the advantages that come from an architectural change, several avenues for future improvement have not yet been explored. In particular, our work focuses on PointNet++-like convolutions, but the attentional convolutions of patch-transformer models like PTv3 could also be used with our backbone. Training technique changes like the use of pretraining could also be adopted from the transformer works, and may be appropriate because of the expressiveness of our model. We retain the same multigrid hierarchy as DeLA, but the GMNN design decouples depth from level count; additional performance may be possible with architectures that operate on many more levels while remaining shallow.

As our network is a superset of other architectures, we see potential for it as a tool for further design exploration, for example, in combination with pruning. Less-useful convolutions and grid-transfers could be discovered by examining the affine weights of norms or by setting these weights to zero on a trained network and observing the resulting penalty to accuracy. Connections which are found to be less useful to the network could then be pruned, leaving the necessary subset to create a smaller and faster bespoke architecture for a given task. Beyond producing a better architecture, this could give us more fundamental insights into how networks learn to route information.

Reproducibility statement. A complete implementation of our method will be made available upon publication. Model configurations and scripts for running different experiments will be included alongside the implementation.

CRediT authorship contribution statement

Jackson Campolattaro: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Ruben Wiersma:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Methodology, Investigation, Formal analysis, Conceptualization. **Klaus Hildebrandt:** Writing – review & editing, Writing – original draft, Validation, Supervision, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Klaus Hildebrandt reports financial support was provided by Nederlandse Organisatie voor Wetenschappelijk Onderzoek. Klaus Hildebrandt is a guest editor for a special issue of Computer Aided Geometric Design. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This publication is part of the project 'Geometric Multigrid CNNs and Multiscale Training for Geometric Deep Learning on Surfaces' with file number OCENW.M.21.346 of the research programme Open Competition Domain Science-M programme which is (partly) financed by the Dutch Research Council (NWO).

Appendix A

A.1. Early saturation

We find that during training, our multigrid adaptation of DeLA tends to reach high accuracy on common classes much earlier than its U-Net counterpart. This difference is clearest for instance-weighted accuracy metrics. Fig. A.6 shows relevant metrics for our implementations of GMNN and DeLA over the course of a training run, with identical training procedures. The multigrid model converges more quickly to high accuracy in both tests, and the accuracy of the U-Net only becomes competitive later in training.

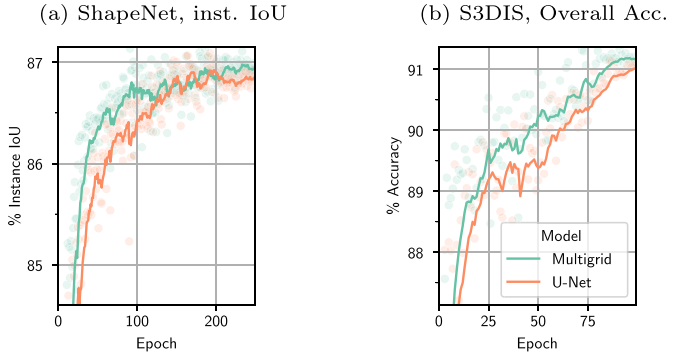


Fig. A.6. Convergence plots over the course of training on different datasets. We observe that the multigrid network converges earlier than the U-Net in the early epochs.

A.2. Robustness against edge selection

In a CNN, the primary way of increasing receptive field size is to increase the size of the kernel. In a graph convolutional network, the equivalent is to increase the number of neighbors, k. Section 4.1 indicated that GMNN has an architectural advantage in receptive field size, independent of k. Table A.9 shows how the performance of DeLA and our multigrid adaptation compare when k is modified. At $k = 20$ (the original setting), both models perform near their best. As k decreases below 16, the accuracy of DeLA falls off rapidly. The multigrid model is much more robust and maintains an accuracy closer to its best accuracy, all the way to $k = 8$. This suggests that the improved receptive field translates to an improved robustness to under-connected networks. More surprisingly, the accuracy of DeLA also falls off when k is increased. This may be due to a combination of overfitting and a missing ability to reject spurious edges. The multigrid network is similarly robust against this type of problem, possibly because it does not have the same dependency on spurious edges for longer-range communication. Isolating the exact cause of this advantage is an interesting avenue for future work; if it also applies in datasets with a mixture of different point densities, then a GMNN could be beneficial on outdoor segmentation tasks where individual scans span a wider range of density.

Table A.9
Effects of k-nearest neighbors on ShapeNet, 25 epochs, 3 trials.

Model	Accuracy % (mIoU, mean ± std)						
	$k = 8$	$k = 12$	$k = 16$	$k = 20$	$k = 24$	$k = 28$	$k = 32$
DeLA	84.1 ± 0.2	84.3 ± 0.1	84.7 ± 0.2	84.8 ± 0.0	84.7 ± 0.2	84.6 ± 0.1	84.6 ± 0.1
GMNN	84.8 ± 0.2	84.9 ± 0.1	85.0 ± 0.1	85.0 ± 0.0	85.0 ± 0.1	84.9 ± 0.3	84.8 ± 0.1

A.3. Intermediate reachability

Section 4.1 briefly discusses how the receptive field of a convolution in GMNN grows rapidly depending on its depth within the network. Fig. A.7 shows this same property in more detail:

In a flat network, the number of convolutions required to reach a node increases approximately linearly with distance, and without a very deep network much of the mesh cannot be reached at all. Unreachable nodes are shown in black, but the practical size of the receptive field is likely smaller than that shown, because the furthest nodes can only be reached using every convolution in the network. Due to the vanishing gradient problem, a network is less likely to learn to use such long communication paths.

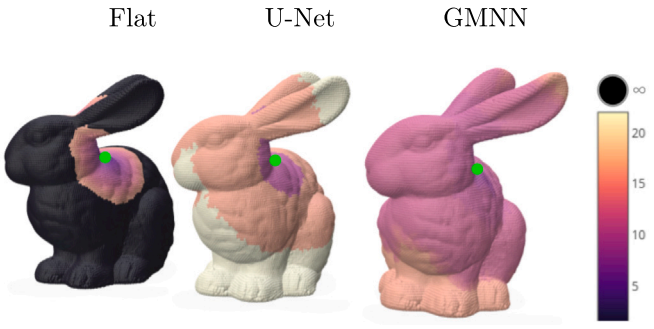


Fig. A.7. Visualization of the growth of receptive field with depth, for a flat network, a U-Net, and GMNN. Color corresponds to the number of convolutions required for a signal from the starting point (indicated in green) to reach each other point (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.).

In a U-Net, the number of convolutions instead increases in stages, as coarser levels of the hierarchy are only available after first passing through the finer levels. In a network like DeLA, multiple convolutions are performed at each stage, and at each coarser stage these convolutions grow the receptive field more rapidly. At the edge of the receptive field, we run into the same problem as the flat network—information must pass through every convolution in the network to reach the furthest points.

In a GMNN, a feature can be progressively downsampled so that coarse convolutions can carry it long distances and then progressively upsampled to reach the fine destination point, without the use of any convolutions at intermediate levels. As a result, the number of convolutions required grows slowly and approximately linearly, remaining relatively low across the entire mesh.

Data availability

The authors do not have permission to share data.

References

Armeni, I., Sener, O., Zamir, A.R., Jiang, H., Brilakis, I., Fischer, M., Savarese, S., 2016. 3d semantic parsing of Large-Scale indoor spaces. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1534–1543, https://doi.org/10.1109/CVPR.2016.170

Finder, S.E., Weber, R.S., Eliasof, M., Freifeld, O., Treister, E., 2025. Improving the Effective Receptive Field of Message-Passing Neural Networks. arXiv preprint arXiv:2505.23185.

Gadelha, M., Wang, R., Maji, S., 2018. Multiresolution tree networks for 3d point cloud processing. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (Eds.), Computer Vision – ECCV 2018. Springer International Publishing, Cham, pp. 105–122.

Greenfeld, D., Galun, M., Basri, R., Yavneh, I., Kimmel, R., 2019. Learning to Optimize Multigrid PDE Solvers. arXiv preprint arXiv:1902.10248.

IndefinGaming, 2021. Animated paper crumpling. https://www.turbosquid.com/3d-models/animated-paper-crumpling-3d-1794996.

Ke, T.W., Maire, M., Yu, S.X., 2016. Multigrid neural architectures. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4067–4075, https://api.semanticscholar.org/CorpusID:23874112.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B., 2018. PointCNN: convolution on x-transformed points. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA., pp. 828–838. Event-place: Montréal, Canada.

Lin, T.Y., Dollar, P., Girshick, R., He, K., Hariharan, B., Belongie, S., 2017. Feature pyramid networks for object detection. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE Computer Society, Los Alamitos, CA, USA, pp. 936–944, https://doi.org/10.1109/CVPR.2017.106

Ma, X., Qin, C., You, H., Ran, H., Fu, Y., 2022. Rethinking network design and local geometry in point cloud: A simple residual MLP framework. arXiv preprint arXiv:2202.07123.

Maesumi, A., Makadia, T., Groueix, T., Kim, V.G., Ritchie, D., Aigerman, N., 2025. PoissonNet: a Local-Global approach for learning on surfaces. In: ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2025).

Margenberg, N., Hartmann, D., Lessig, C., Richter, T., 2022. A neural network multigrid solver for the Navier-Stokes equations. J. Comput. Phys. 460, 110983. https://doi.org/10.1016/j.jcp.2022.110983. https://www.sciencedirect.com/science/article/pii/S0021999122000456.

Park, J., Lee, S., Kim, S., Xiong, Y., Kim, H.J., 2023. Self-Positioning Point-Based transformer for point cloud understanding. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE Computer Society, Los Alamitos, CA, USA, pp. 21814–21823, https://doi.org/10.1109/CVPR52729.2023.02089

Qi, C.R., Yi, L., Su, H., Guibas, L.J., 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. arXiv preprint arXiv:1706.02413.

Qian, G., Li, Y., Peng, H., Mai, J., Hammoud, H., Elhoseiny, M., Ghanem, B., 2022. PointNeXt: Revisiting PointNet++ with Improved Training and Scaling Strategies. Advances, In.

Qiu, H., Yu, B., Chen, Y., Tao, D., 2023. PointHR: Exploring High-Resolution Architectures for 3D Point Cloud Segmentation. arXiv preprint arXiv:2310.07743.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. Springer International Publishing, Cham, pp. 234–241.

Sharp, N., Attaiki, S., Crane, K., Ovsjanikov, M., 2022. DiffusionNet: discretization agnostic learning on surfaces. ACM Trans. Graph. 41, https://doi.org/10.1145/3507905

Taghibakhshi, A., Nytko, N., Zaman, T.U., MacLachlan, S., Olson, L.N., West, M., 2023. MG-GNN: multigrid graph neural networks for learning multilevel domain decomposition methods. In: Proceedings of the 40th International Conference on Machine Learning, JMLR.org. Event-Place, Honolulu, Hawaii, USA.

Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J., 2019. KPConv: flexible and deformable convolution for point clouds. In: Proceedings of the IEEE International Conference on Computer Vision.

Thomas, H., Tsai, Y.H.H., Barfoot, T.D., Zhang, J., 2024. KPConvX: modernizing kernel point convolution with kernel attention. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5525–5535.

Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M., 2019. Dynamic graph CNN for learning on point clouds. ACM Trans. Graph. 38, https://doi.org/10.1145/3326362

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J., 2015. 3d shapenets: a deep representation for volumetric shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1912–1920.

Wu, X., Lao, Y., Jiang, L., Liu, X., Zhao, H., 2022. Point Transformer V2: Grouped Vector Attention and Partition-Based Pooling. NeurIPS.

Wu, X., Jiang, L., Wang, P.S., Liu, Z., Liu, X., Qiao, Y., Ouyang, W., He, T., Zhao, H., 2024. Point Transformer V3: Simpler. CVPR, Faster, Stronger.

Wu, X., DeTone, D., Frost, D., Shen, T., Xie, C., Yang, N., Engel, J., Newcombe, R., Zhao, H., Straub, J., 2025. Sonata: Self-Supervised Learning of Reliable Point Representations. CVPR.

Yang, W., Lu, X., Chen, B., Lin, C., Bao, X., Liu, W., Zang, Y., Xu, J., Wang, C., 2024. DeLA: an extremely faster network with decoupled local aggregation for large scale point cloud learning. Int. J. Appl. Earth Obs. Geoinf. 135, 104255. https://doi.org/10.1016/j.jag.2024.104255. https://www.sciencedirect.com/science/article/pii/S1569843224006113.

Zhang, Y., Li, J., Wang, Z., Duan, J., Li, J., 2023. AVS-net: attention-based variable splitting network for p-MRI acceleration. In: 2023 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 1682–1689, https://doi.org/10.1109/BIBM58861.2023.10385266

Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V., 2021. Point transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 16259–16268.

Zhou, Z., Rahman Siddiquee, M.M., Tajbakhsh, N., Liang, J., 2018. UNet++: a nested U-Net architecture for medical image segmentation. In: Stoyanov, D., Taylor, Z., Carneiro, G., Syeda-Mahmood, T., Martel, A., Maier-Hein, L., Tavares, J.M.R., Bradley, A., Papa, J.P., Belagiannis, V., Nascimento, J.C., Lu, Z., Conjeti, S., Moradi, M., Greenspan, H., Madabhushi, A. (Eds.), Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support. Springer International Publishing, Cham, pp. 3–11.