# POINTCONVFORMER: REVENGE OF THE POINT-BASED CONVOLUTION

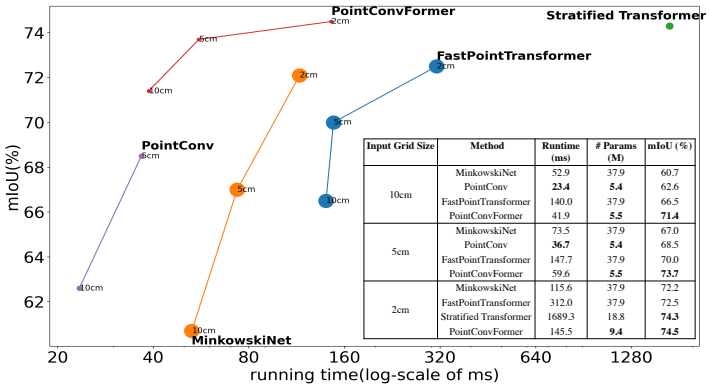
**Anonymous authors**
Paper under double-blind review

## ABSTRACT

We introduce PointConvFormer, a novel building block for point cloud based deep network architectures. Inspired by generalization theory, PointConvFormer combines ideas from point convolution, where filter weights are only based on relative position, and Transformers which utilize feature-based attention. In PointConvFormer, attention computed from feature difference between points in the neighborhood is used to modify the convolutional weights at each point. Hence, we preserved the invariances from point convolution, whereas attention helps to select relevant points in the neighborhood for convolution. We experiment on both semantic segmentation and scene flow estimation tasks on point clouds with multiple datasets including ScanNet, SemanticKitti, FlyingThings3D and KITTI. Our results show that PointConvFormer substantially outperforms classic convolutions, regular transformers, and voxelized sparse convolution approaches with much smaller and faster networks. Visualizations show that PointConvFormer performs similarly to convolution on flat areas, whereas the neighborhood selection effect is stronger on object boundaries, showing that it has got the best of both worlds. The code will be available with the final version.

## 1 INTRODUCTION

Depth sensors for indoor and outdoor 3D scanning have significantly improved in terms of both performance and affordability. Hence, their common data format, 3D point cloud, has drawn significant attention from academia and industry. Understanding the 3D real world from point clouds can be applied to many application domains, e.g. robotics, autonomous driving, CAD, and AR/VR. However, unlike image pixels arranged in regular grids, 3D points are unstructured, which makes applying grid based Convolutional Neural Networks (CNNs) difficult.

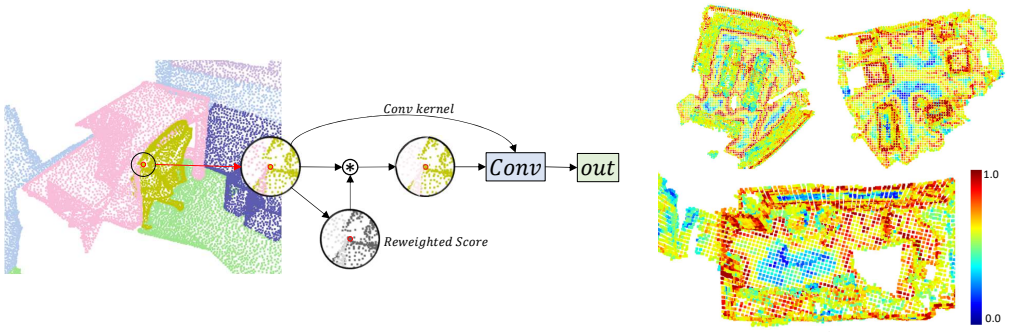



| Input Grid Size | Method | Runtime (ms) | # Params (M) | mIoU (%) |
|---|---|---|---|---|
| 10cm | MinkowskiNet | 52.9 | 37.9 | 60.7 |
| | PointConv | **23.4** | **5.4** | 62.6 |
| | FastPointTransformer | 140.0 | 37.9 | 66.5 |
| | PointConvFormer | 41.9 | **5.5** | **71.4** |
| 5cm | MinkowskiNet | 73.5 | 37.9 | 67.0 |
| | PointConv | **36.7** | **5.4** | 68.5 |
| | FastPointTransformer | 147.7 | 37.9 | 70.0 |
| | PointConvFormer | 59.6 | **5.5** | **73.7** |
| 2cm | MinkowskiNet | 115.6 | 37.9 | 72.2 |
| | FastPointTransformer | 312.0 | 37.9 | 72.5 |
| | Stratified Transformer | 1689.3 | 18.8 | **74.3** |
| | PointConvFormer | 145.5 | **9.4** | **74.5** |

Figure 1: **Performance vs. running time on ScanNet.** PointConvFormer achieves a state-of-the-art 74.5% mIoU while being efficient with faster speed and way less learnable parameters. Larger dot indicates more learnable parameters. All results are reported on a single TITAN RTX GPU

Various approaches have been proposed in response to this challenge. (Su et al., 2015; Li et al., 2016; Chen et al., 2017; Kanezaki et al., 2018; Lang et al., 2019) introduced interesting ways to project 3D point clouds back to 2D image space and apply 2D convolution. Another line of research directly voxelizes 3D space and apply 3D discrete convolution, but it induces massive computation and memory overhead (Maturana & Scherer, 2015; Song et al., 2017). Sparse convolution opera-

∗ Majority of the work done as an intern at Apple, Inc.




(a) (b)

Figure 2: (a) **PointConvFormer** can be seen as a point convolution, but modulated by a scalar attention weight for each point in the neighborhood, so that the neighborhood is selectively chosen to perform convolution; (b) Visualization of the reweighting effect in PointConvFormer. The colors are computed by the difference of the maximal attention and the minimal attention in each neighborhood. Red areas have stronger reweighting and blue areas behave similar to convolution. It can be seen that the reweighting effect is stronger at object boundaries where the neighborhoods are more likely to be problematic, whereas on smoother surfaces PointConvFormer behaves more similarly to convolution (more details in Sec. 4.4)

tions (Graham et al., 2018; Choy et al., 2019) save a significant amount of computation by computing convolution only on occupied voxels.

Some approaches directly operate on point clouds (Qi et al., 2017a;b; Su et al., 2018; Thomas et al., 2019; Wu et al., 2019b; Li et al., 2021b). Qi et al. (2017a;b) are pioneers which aggregate information on point clouds using max-pooling layers. Others proposed to reorder the input points with a learned transformation (Li et al., 2018), a flexible point kernel (Thomas et al., 2019), and a convolutional operation that directly work on point clouds (Wang et al., 2018b; Wu et al., 2019b) which utilizes a multi-layer perceptron (MLP) to learn convolution weights implicitly as a nonlinear transformation from the relative positions of the local neighbourhood.

The approach to directly work on points is appealing to us because it allows direct manipulation of the point coordinates, therefore being able to encode rotation/scale invariance/equivariance directly into the convolution weights (Zhang et al., 2019; Li et al., 2021b). These invariances can serve as priors to make the models more generalizable. Besides, point-based approaches require less parameters than voxel-based ones, which need to keep e.g. $3 \times 3 \times 3$ convolution kernels on all input and output channels. Finally, point-based approaches often utilize k-nearest neighbors (kNN) to find the local neighborhood, thus can use less layers to obtain a larger receptive field than sparse convolution, where the fixed neighborhoods are often quite empty.

However, so far the methods with the best efficiency-accuracy tradeoff have still been the sparse voxel-based approaches or a fusion between sparse voxel and point-based models. We do not believe such a fusion would be necessary, since no matter the voxel-based or point-based representation, the information from the input is exactly the same. Besides, similar convolution operations can be performed in both the voxel and point-based representations. This leads us to question the component that is indeed different between these representations: the generalization w.r.t. to the irregular local neighbourhood. The shape of the kNN neighbourhood commonly used in point-based methods varies in different parts of the point cloud. Irrelevant points from other objects, noise and the background might be included in the neighborhood, especially around object boundaries, which can be detrimental to the performance and the robustness of point-based models.

To improve the robustness of models with kNN neighborhoods, we refer back to the generalization theory of CNNs, which indicates that points with significant feature correlation should be included in the same neighborhood (Li et al., 2017). A key idea in this paper is that feature correlation can be a way to filter out irrelevant neighbors in a kNN neighborhood, which makes the subsequent convolution more generalizable. We introduce PointConvFormer, which computes attention weights based on feature differences and use that to reweight the points in the neighborhood in a point-based convolutional model. PointConvFormer addresses the puzzle of how to define a "good" (yet irregular) neighbourhood in point cloud processing for better representation and generalization.

The idea of using feature-based attention is not new, but there are important differences between PointConvFormer and the recently popular vision transformers (Dosovitskiy et al., 2020; Zhao et al.,

2021; Park et al., 2021). PointConvFormer combines features in the neighborhood with point-wise convolution, whereas Transformer attention models usually adopt softmax attention in this step. Softmax outputs a small amount of positive weights and a large amount of weights very close to zero, and is not able to generate negative coefficients in the aggregation step. In our formulation, because convolution is used for aggregation, both positive and negative weights are allowed. This is shown to be better than PointTransformer in experiments.

We evaluate PointConvFormer on two point cloud tasks, semantic segmentation and scene flow estimation. For semantic segmentation, experiment results on the indoor ScanNet (Dai et al., 2017) and the outdoor SemanticKitti (Behley et al., 2019b) demonstrate superior performances over classic convolution and transformers with a much more compact network. The performance gap is the most significant at low resolutions, e.g. on ScanNet with a 10cm resolution we achieved more than **10% improvement** over MinkowskiNet with only **15%** of its parameters (Fig. 1). We also apply PointConvFormer as the backbone of PointPWC-Net (Wu et al., 2020) for scene flow estimation, and observe significant improvements on FlyingThings3D (Mayer et al., 2016a) and KITTI scene flow 2015 (Menze et al., 2018) datasets as well. These results show that PointConvFormer could potentially replace sparse convolution as the backbone of choice for 3D point cloud tasks.

## 2 Related Work

**Voxel-based networks.** Different from 2D images, 3D point clouds are unordered and scattered in 3D space. One of the trending approaches to process 3D point clouds is to voxelize the point clouds into regular 3D voxels. However, directly applying 3D convolution (Maturana & Scherer, 2015; Song et al., 2017) onto the 3D voxels can incur massive computation and memory overhead, which limits its applications to large-scale real world scenarios. The sparse convolution (Graham et al., 2018; Choy et al., 2019) reduces the convolutional overhead by only working on the non-empty voxels. However, this kind of approaches may suffer if the quantization of the voxel grid is too coarse. The best performances are achieved with high quantization resolutions (e.g. $2cm$ per voxel), which still have high memory consumption and lead to large models.

**Point-based networks.** There are plenty of works (Qi et al., 2017a;b; Wu et al., 2019b; Li et al., 2021b; Su et al., 2018; Wang et al., 2018c) focusing on directly processing point clouds without re-projection or voxelization. Qi et al. (2017a;b) propose to use MLPs followed by max-pooling layers to encode and aggregate point cloud features. However, max-pooling could lead to the loss of critical geometric information in the point cloud. A number of works (Melekhov et al., 2019; Jia et al., 2016; Mayer et al., 2016b; Li et al., 2019; Goyal et al., 2021; Wang et al., 2018a) build a kNN graph from the point cloud and conduct message passing using graph convolution. Later on, (Wang et al., 2018b; Xu et al., 2018; Thomas et al., 2019; Wu et al., 2019b; Mao et al., 2019; Li et al., 2018; Esteves et al., 2018; Li et al., 2021b) conduct continuous convolution on point clouds. Wang et al. (2018b) represents the convolutional weights with MLPs. SpiderCNN Xu et al. (2018) uses a family of polynomial functions to approximate the convolution kernels. Su et al. (2018) projects the whole point cloud into a high-dimensional grid for rasterized convolution. Wu et al. (2019b); Thomas et al. (2019) formulate the convolutional weights to be a function of relative position in a local neighbourhood, where the weights can be constructed according to input point clouds. Li et al. (2021b) improves over (Wu et al., 2019b) by introducing hand-crafted viewpoint-invariant coordinate transforms on the relative position to increase the robustness of the network.

**Dynamic filters and Transformers.** Recently, the design of dynamic convolutional filters (Yang et al., 2019a; Zhang et al., 2020b; Chen et al., 2020; Jia et al., 2016; Wang et al., 2019; Su et al., 2019; Zamora Esquivel et al., 2019; Wang et al., 2020; Tian et al., 2020; Ma et al., 2020; Jampani et al., 2016; Zhou et al., 2021) has drawn more attentions. This line of work (Ma et al., 2020; Zhang et al., 2020b; Chen et al., 2020; Yang et al., 2019a) introduces different methods to predict convolutional filters, which are shared across the whole input. (Jia et al., 2016; Zamora Esquivel et al., 2019; Wang et al., 2020; Tian et al., 2020) propose to predict the complete convolutional filters for each pixel. However, their applications are constrained by their computational inefficiency and high memory usage. (Zhou et al., 2021) introduces decoupled dynamic filters with respect to the input features on 2D classification and upsampling tasks. (Su et al., 2019; Tabernik et al., 2020) propose to re-weight 2D convolutional kernels with a fixed Gaussian or Gaussian mixture model for pixel-adaptive convolution. Dynamic filtering share some similarities with the popular transformers, whose weights

are functions of feature correlations. However, the dynamic filters are mainly designed for images instead of point clouds and focus on regular grid-based convolutions.

With recent success in natural language processing (Devlin et al., 2018; Dai et al., 2019; Vaswani et al., 2017; Wu et al., 2019a; Yang et al., 2019c) and 2D images analysis (Hu et al., 2019; Dosovitskiy et al., 2020; Zhao et al., 2020; Ramachandran et al., 2019), transformers have drawn more attention in the field of 3D scene understanding. Some work (Lee et al., 2019; Liu et al., 2019b; Yang et al., 2019b; Xie et al., 2018) utilize global attention on the whole point cloud. However, these approaches introduce heavy computation overhead and are unable to extend to large scale real world scenes, which usually contain over $100k$ points per point cloud scan. Recently, the work (Zhao et al., 2021; Park et al., 2021) introduce point transformer with local attention to reduce the computation overhead, which could be applied to large scenes. Compared to previous convolutional approaches, our PointConvFormer computes the weights with both the relative position and the feature difference. Compared to transformers, the attention of the PointConvFormer modulates convolution kernels and use the sigmoid activation instead of softmax. Experiments showed that our design significantly improves the performance of the networks.

# 3 PointConvFormer

## 3.1 Point Convolutions and Transformers

Given a continuous input signal $x(p) \in \mathbb{R}^{c_{in}}$ where $p \in \mathbb{R}^s$ with $s$ being a small number (2 for 2D images or 3 for 3D point clouds, but could be any arbitrary low-dimensional Euclidean space), $x(\cdot)$ can be sampled as a point cloud $P = \{p_1, \dots, p_n\}$ with the corresponding values $x_P = \{x(p_1), \dots, x(p_n)\}$, where each $p_i \in \mathbb{R}^s$. The continuous convolution at point $p$ is formulated as:

$$Conv(w, x)_p = \int_{\Delta p \in \mathbb{R}^s} \langle w(\Delta p), x(p + \Delta p) \rangle d\Delta p \tag{1}$$

where $w(\Delta p) \in \mathbb{R}^{c_{in}}$ is the continuous convolution weight function. Inspired by the continuous formulation of convolution, (Simonovsky & Komodakis, 2017; Wu et al., 2019b; Wang et al., 2018c) discretize the continuous convolution on a neighbourhood of point $p$. Let $X_{p_i} \in \mathbb{R}^{c_{in}}$ be the input feature of $p_i$ the discretized convolution on point clouds is written as:

$$X'_p = \sum_{p_i \in \mathcal{N}(p)} w(p_i - p)^\top X_{p_i} \tag{2}$$

where $\mathcal{N}(p)$ is a neighborhood that is normally chosen as the $k$-nearest neighbor or $\epsilon$-ball neighborhood of the center point $p$. The function $w(p_i - p) : \mathbb{R}^s \mapsto \mathbb{R}^{c_{in}}$ can be approximated as an MLP and learned from data.Moreover, because now $p_i - p$ can be explicitly controlled, one can concatenate invariant coordinate transforms on $p_i - p$ as input to $w(\cdot)$, e.g. $\|p_i - p\|$ would be rotation invariant. Li et al. (2021b) has found that concatenating a set of invariant coordinate transforms with $p_i - p$ significantly improves the performance of point convolutions.

In PointConv (Wu et al., 2019b), an efficient formulation was derived when $w(p_i - p)$ has a linear final layer $w(p_i - p) = W_l h(p_i - p)$, where $h(p_i - p) : \mathbb{R}^3 \mapsto \mathbb{R}^{c_{mid}}$ is the output of the penultimate layer of the MLP and $W_l \in \mathbb{R}^{c_{in} \times c_{mid}}$ is the learnable parameters in the final linear layer. We can equivalently change Eq. (2) on the neighbourhood $\mathcal{N}(p)$ into,

$$X'_p = \left\langle \mathrm{vec}(W_l), \mathrm{vec}\left\{ \sum_{p_i \in \mathcal{N}(p)} h(p_i - p) X_{p_i}^\top \right\} \right\rangle. \tag{3}$$

where $\mathrm{vec}(\cdot)$ turns the matrix into a vector. Note that $W_l$ represents parameters of a linear layer and hence independent of $p_i$. Thus, when there are $c_{out}$ convolution kernels, $n$ training examples with a neighborhood size of $k$ each, there is no longer a need to store the original convolution weights $w(p_i - p)$ for each point in each neighborhood with a dimensionality of $c_{out} \times c_{in} \times k \times n$. Instead, the dimension of all the $h(p_i - p)$ vectors in this case is only $c_{mid} \times k \times n$, where $c_{mid}$ is significantly smaller(usually 8 or 16) than $c_{out} \times c_{in}$ (could go higher than $10^2 \times 10^2$). This efficient PointConv enables applications to large-scale networks on 3D point cloud processing.

Recently, transformer architectures are popular with 2D images. 3D point cloud-based transformers have also been proposed (e.g. (Zhao et al., 2021; Park et al., 2021)). Transformers compute an attention model between points (or pixels) based on the features of both points and the positional encodings of them. Relative positional encoding was the most popular which encodes $w(p_i - p)$, similar to Eq.( 2). It has been shown to outperform absolute positional encodings in many papers (Shaw et al., 2018; Chu et al., 2021; Zhao et al., 2021). Adopting similar notations to eq. 2, we can express the softmax attention model used in transformers as:

$$Attention(p) = \sum_{p_i \in \mathcal{N}(p)} \text{softmax}(\mathbf{q}(X_{p_i})\mathbf{k}(X_p) + w(p_i - p)) \cdot \mathbf{v}(X_{p_i}) \tag{4}$$

where $\mathbf{q}(\cdot), \mathbf{k}(\cdot), \mathbf{v}(\cdot)$ are transformation to the features to form the query, key and value matrices respectively, usually implemented with MLPs. One can see that there are similarities and differences between PointConv (Wu et al., 2019b) and the attention model (Vaswani et al., 2017). First, both employ $w(p_i - p)$, but in PointConv that is the sole source of the convolutional kernel which is translation-invariant. In attention models, the matching between the query transform $\mathbf{q}(X_{p_i})$ and the key transform $\mathbf{k}(X_p)$ of the features are also considered, which is no longer translation-invariant.

Another important difference to note is that in attention models the final attention value is an output from the softmax function. Note that softmax output has a range of $[0, 1]$ which is limited to **non-negative** weights at each point, which means the output of eq. 4 is a non-negative weighted average of the features of the input. To us, it is a bit curious why this is the right idea, as we tend to believe each neighborhood point could have positive and negative impacts to the features of the center point, and limiting it only to non-negative might be a dubious design choice.

Note that the $\mathbf{v}(\cdot)$ transform in eq.(4) can be seen as a $1 \times 1$ convolution on the input. It is common to insert $1 \times 1$ convolution layers between regular convolution layers in deep architectures (e.g. ResNet(He et al., 2016)), hence we can view it as an additional $1 \times 1$ convolution layer before the attention layer. Thus, we can compare the attention layer and PointConv without considering $\mathbf{v}(\cdot)$.

### 3.2 CNN Generalization Theory and The PointConvFormer Layer

We are interested in adopting the strengths of attention-based models, while still preserve some of the benefits of convolution and explore the possibility of having negative weights. To this end, we first look at theoretical insights in terms of which architecture would generalize well. We note the following bound proved in (Li et al., 2017):

$$\hat{G}_N(F) \leq C \max_{p' \in \mathcal{N}(p)} \sqrt{\mathbb{E}_{X,p}[(X_p - X_{p'})^2]} \tag{5}$$

where $\hat{G}_N(F)$ is the empirical Gaussian complexity on the function class $F$: a one-layer CNN followed by a fully-connected layer, and $C$ is a constant. A *smaller* Gaussian complexity leads to better generalization (Bartlett & Mendelson, 2002). To minimize the Gaussian complexity bound in eq. (5), it can be seen that one should select points that has high feature correlation to belong to the same neighborhood. In images, nearby pixels usually have the highest color correlation (Li et al., 2017), hence conventional CNNs achieve better generalization by choosing a small local neighborhood (e.g. $3 \times 3$). In 3D point clouds, as mentioned in the introduction, noisy points can be included in the kNN neighborhood, which reduces feature correlation and henceforth worsens the generalization. This motivates us to attempt to filter out those noisy points by explicitly checking their $X_p - X_{p'}$, hence keeping only the relevant points in the CNN neighborhood.

Inspired by the discussion above, we define a novel convolution operation, ***PointConvFormer***, which takes into account both the relative position $p_i - p$ and the feature difference $X_{p_i} - X_p$. The PointConvFormer layer of a point $p$ with its neighbourhood $\mathcal{N}(p)$ can be written as:

$$X'_p = \sum_{p_i \in \mathcal{N}(p)} w(p_i - p)^\top \psi([X_{p_i} - X_p, p_i - p]) X_{p_i} \tag{6}$$

where the function $w(p_i - p)$ is the same as defined in Eq.( 2), the scalar function $\psi([X_{p_i} - X_p, p_i - p]) : \mathbb{R}^{c_{in}} \mapsto \mathbb{R}$ is the function of both feature differences $X_{p_i} - X_p$ and position differences.

If we fix the function $\psi(\cdot) = 1$, the PointConvFormer layer is equivalent to Eq.( 3), which reduces to traditional convolution. In eq.(6), $\psi(\cdot)$ is approximated with another MLP followed by a activation layer. As a result, the function $w(p_i - p)$ learns the weights respect to the relative positions,

and the function $\psi(\cdot)$ learns to select useful points in the neighborhood, which works similarly to the attention in transformer. However, different from the transformer whose non-negative weights are directly used as a weighted average on the input, the output of $\psi(\cdot)$ only modifies the convolutional filter $w(p_i - p)$, which allows each neighborhood point to have both positive and negative contributions.

Since $\psi(\cdot)$ does not modify the convolution, we adopt the same approach in PointConv (Wu et al., 2019b) to create an efficient version of the PointConvFormer layer. Following Eq.(3), we have:

$$X'_p = W_l \sum_{p_i \in \mathcal{N}(p)} h(p_i - p)\psi(X_{p_i}, X_p)X_{p_i}^\top \tag{7}$$

where $W_l$ and $h(\cdot)$ are the same as in Eq.( 3).

**Multi-Head Mechanism** As in Eq.(7), the weight function $\psi(\cdot) : \mathbb{R}^{c_{in}} \mapsto \mathbb{R}$ learns the relationship between the center point feature $X_p \in \mathbb{R}^{c_{in}}$ and its neighbourhood features $X_{p_i} \in \mathbb{R}^{c_{in}}$, where $c_{in}$ is the number of the input feature dimension. To increase the representation power of the PointConvFormer, we use the multi-head mechanism to learn different types of neighborhood filter mechanisms. As a result, the function $\psi : \mathbb{R}^{c_{in}} \mapsto \mathbb{R}$ becomes a set of functions $\psi_i : \mathbb{R}^{c_{in}} \mapsto \mathbb{R}$ with $i \in \{1, ..., h\}$, $h$ being the number of heads.

### 3.3 PointConvFormer Block

To build deep neural network for various computer vision tasks, we construct bottleneck residual blocks with PointConvFormer layer as its main components. The detailed structures of the residual blocks are illustrated in Fig. 3. The input of the residual block is the input point features $X \in \mathbb{R}^{c_{in}}$ along with its coordinates $p \in \mathbb{R}^3$. The residual block uses a bottleneck structure, which consists of two branches. The residual branch is a linear layer, followed by PointConvFormer layer, followed by another linear layer. Following ResNet and KPConv (He et al., 2016; Thomas et al., 2019), we use one-fourth of the input channels in the first linear layer, conduct PointConvFormer with the smaller number of channels, and finally upsample to the amount of output channels. We have found this strategy to significantly reduce the model size and computational cost while maintaining high accuracy for both PointConv and PointConvFormer. The shortcut branch can be formulated in three different ways depending on the output feature size. If the output feature has the same cardinality and dimensionality, the shortcut branch is just a identity mapping. If the output feature has the same cardinality but with different dimensionality, the shortcut branch is a linear mapping. If the output feature has different cardinality, e.g. when the point cloud is downsampled, the shortcut branch uses a max-pooling layer to aggregate features.

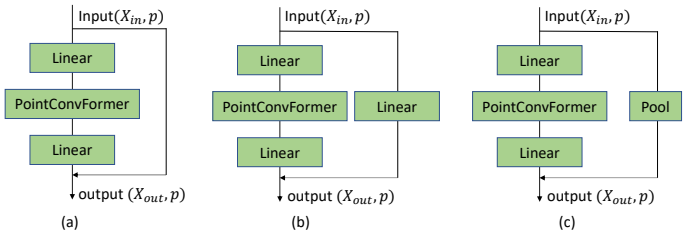



Figure 3: **The residual blocks of PointConvFormer.** We use Linear layers and pooling layers to change the dimensionality and cardinality of the shortcut to match the output of the residual branch.

**Downsampling and Deconvolution** We use the grid-subsampling method (Thomas et al., 2019) to downsample the point clouds in the same way as the 2x2 downsampling in 3D convolutions, which had been shown to outperform random and furthest point downsampling (Thomas et al., 2019). This subsampling choice makes PointConvFormer directly comparable with 3D convolution backbones at the same voxelization levels (e.g. 2cm, 5cm) as the voxelization and downsampling become both the same. For upsampling layers, we cannot apply PointConvFormer because for points that do not exist in the downsampled cloud, their features are not available. Instead, we note that in eq. 3 of PointConv, $p$ itself does not have to belong to $\mathcal{N}(p)$, thus we can just apply PointConv layers for deconvolution without features $X_p$ as long as coordinates $p$ are known. This helps us to keep the consistency of the network and avoid arbitrary interpolation layers that are not learnable.

# 4 EXPERIMENTS

In this section, we conduct experiments in a number of domains and tasks to demonstrate the effectiveness of the proposed PointConvFormer. For 3D semantic segmentation, we use the challenging ScanNet (Dai et al., 2017), a large-scale indoor scene dataset, and the SemanticKitti dataset (Behley et al., 2019b), a large-scale outdoor scene dataset. Besides, we conduct experiments on the scene flow estimation from 3D point clouds with the synthetic FlyingThings3D dataset (Mayer et al., 2016a) for training and the KITTI scene flow 2015 dataset (Menze et al., 2018). We also conduct ablation studies to explore the properties of the PointConvFormer which are shown in the appendix.

**Implementation Details.** We implement PointConvFormer in PyTorch (Paszke et al., 2019). The viewpoint-invariant coordinate transform in (Li et al., 2021b) is concatenated with the relative coordinates as the input to the $w(\cdot)$ function. We use the Adam optimizer with $(0.9, 0.999)$ betas and $0.0001$ weight decay. For the ScanNet dataset, we train the model with an initial learning rate 0.001 and dropped to 0.5x for every 60 epochs for 300 epochs. For the SemanticKitti dataset, the model is trained with an initial learning rate 0.001 and dropped to 0.5x for every 8 epochs for 40 epochs. Both semantic segmentation tasks are trained with weighted cross entropy loss. To ensure fair comparison with published approaches, we did not employ the recent Mix3D augmentations (Nekrasov et al., 2021) which would improve performance for all methods. For the scene flow estimation, we follow the exact same training pipeline as in (Wu et al., 2020) for fair comparison.

## 4.1 INDOOR SCENE SEMANTIC SEGMENTATION

We conduct indoor 3D semantic scene segmentation on the ScanNet (Dai et al., 2017) dataset. We use the official split with $1,201$ scenes for training and $312$ for validation. We compare against both voxel-based methods such as MinkowskiNet42 (Choy et al., 2019) and SparseConvNet (Graham et al., 2018), as well as point-based approaches (Qi et al., 2017a; Wu et al., 2019b; Li et al., 2021b; Yan et al., 2020; Thomas et al., 2019). Recently, there are work adopting transformer to point clouds. We chose the Point Transformer (Zhao et al., 2021) and the fast point transformer (Park et al., 2021) as representative transformer based methods. Since the Point Transformer does not report their results on the ScanNet dataset, we adopt their point transformer layer (a standard multi-head attention layer) with the same network structure as ours. Hence, it serves as a direct comparison between PointConvFormer and multi-head attention. There exists some other approaches (Chiang et al., 2019; Hu et al., 2021a;b; Kundu et al., 2020) which use additional inputs, such as 2D images, which benefit from ImageNet (Deng et al., 2009) pre-training that we do not use. Hence, we excluded these methods from comparison, accordingly but we are comparable to the best of them.

We adopt a general U-Net structure with residual blocks in the encoding layers as our backbone model, where the point clouds are gradually downsampled to coarse resolution, then gradually upsampled to its original resolution with highway connections. Through experiments we found out that the decoder can be very lightweight without sacrificing performance (Table 8). Hence, we set $c_{mid}$ to be 1 in the decoder throughout the experiments, and just have consecutive PointConv upsampling layers without any residual blocks. Please refer to the appendix for detailed network structure. Following (Park et al., 2021), we conduct experiments on different input voxel sizes, reported in Fig. 1 and Table 4 in the Appendix. Note that we still utilize kNN neighborhoods which always have $k$ neighbors whereas sparse convolution could have far fewer points in their neighborhood, hence needing more parameters to cover all neighborhood locations, and more layers than us to obtain the same receptive field. According to Fig. 1, Table. 4 and Table. 1, our PointConvFormer achieves best results regardless of the input grid size. Especially, our PointConvFormer outperforms MinkowskiNet42 (Choy et al., 2019) by a very significant **10.0% with 10cm input grid, 7.0% with 5cm input grid, and 2.3% with 2cm input grid**, while being faster than it in the first two cases. It is also significantly **(at least 3 times) faster** than all the transformer approaches while achieving similar or better results. The detailed result table (Table 4) with all the grid sizes and several ablations using the same network architecture but different layer types is shown in the appendix.

## 4.2 OUTDOOR SCENE SEMANTIC SEGMENTATION

The SemanticKitti (Behley et al., 2019b; Geiger et al., 2012) dataset is a large-scale street view point cloud dataset built upon the KITTI Vision Odometry Benchmark (Geiger et al., 2012). The dataset

Table 1: **Semantic segmentation results on ScanNet dataset.** We compare both the ScanNet (Dai et al., 2017) validation set and test set. Numbers for baselines are taken from the literature. The numbers for test set are from the official ScanNet benchmark.

| Methods | # Params(M) | Input | Runtime(ms) | Val mIoU(%) | Test mIoU(%) |
|---|---|---|---|---|---|
| PointNet++ (Qi et al., 2017b) | - | Point | - | 53.5 | 55.7 |
| PointCNN (Li et al., 2018) | - | point | - | - | 45.8 |
| PointConv (Wu et al., 2019b) | - | Point | 83.1 | 65.1 | 66.6 |
| KPConv *deform* (Thomas et al., 2019) | 14.9 | Point | - | 69.2 | 68.4 |
| PointASNL (Yan et al., 2020) | - | Point | - | 63.5 | 66.6 |
| RandLA-Net (Hu et al., 2020a) | - | point | - | - | 64.5 |
| VI-PointConv (Li et al., 2021b) | 15.5 | Point | 88.9 | 70.1 | 67.6 |
| SparseConvNet (Graham et al., 2018) | - | Voxel | - | 69.3 | 72.5 |
| MinkowskiNet42 (Choy et al., 2019) | 37.9 | Voxel | 115.6 | 72.2 | 73.6 |
| PointTransformer (Zhao et al., 2021) | - | Point | - | 70.6 | - |
| Fast Point Transformer (Park et al., 2021) | 37.9 | Voxel | 312.0 | 72.0 | - |
| Stratified Transformer (Lai et al., 2022) | 18.8 | point | 1689.3 | **74.3** | **74.7** |
| PointConvFormer(ours) | 9.4 | Point | 145.5 | **74.5** | **74.9** |

Table 2: **Semantic segmentation results on SemanticKitti validation set.**

| Method | #MACs(G) | # Param.(M) | Input | mIoU(%) |
|---|---|---|---|---|
| RandLA-Net (Hu et al., 2020b) | 66.5 | 1.2 | Point | 57.1 |
| FusionNet (Zhang et al., 2020a) | - | - | Point+Voxel | 63.7 |
| KPRNet (Kochanov et al., 2020) | - | - | Point+Range | 64.1 |
| MinkowskiNet (Choy et al., 2019) | 113.9 | 21.7 | Voxel | 61.1 |
| SPVCNN (Tang et al., 2020) | 118.6 | 21.8 | Point+Voxel | 63.8 |
| SPVNAS (Tang et al., 2020) | 64.5 | 10.8/12.5 | Point+Voxel | 64.7 |
| PointConvFormer(ours) | 91.1 | 8.1 | Point | **67.1** |

consists of $43,552$ point cloud scans sampled from 22 sequences in driving scenes. Each point cloud scan contains $10-13k$ points. We follow the training and validation split in (Behley et al., 2019b) and 19 classes are used for training and evaluation. For each 3D point, only the $(x, y, z)$ coordinates are given without any color information. It is a challenging dataset because the scanning density is uneven as faraway points are more sparse in LIDAR scans.

Table 2 reports the results on the semanticKitti dataset. Because this work mainly focus on the basic building block, PointConvFormer, which is applicable to any kind of 3D point cloud data, of deep neural network, we did not compare with work (Zhu et al., 2021; Cheng et al., 2021) whose main novelties work mostly on LiDAR datasets due to the additional assumption that there are no occlusions from the bird-eye view. From the table, one can see that our PointConvFormer outperforms both point-based methods and point+voxel fusion methods. Especially, our method obtains better results comparing with SPVNAS (Tang et al., 2020), which utilizes the network architecture search (NAS) techniques and fuses both point and voxel branches. We did not utilize any NAS in our system which would only further improve our performance.

### 4.3 Scene Flow Estimation from Point Clouds

Scene flow is the 3D displacement vector between each surface in two consecutive frames. As a fundamental tool for low-level understanding of the world, scene flow can be used in many 3D applications.Traditionally, scene flow was estimated directly from RGB/RGBD data (Huguet & Devernay, 2007; Menze & Geiger, 2015; Vogel et al., 2015). However, with the recent development of 3D sensors such as LiDAR and 3D deep learning techniques, there is increasing interest on directly estimating scene flow from 3D point clouds (Liu et al., 2019a; Gu et al., 2019; Wu et al., 2020; Puy et al., 2020; Wei et al., 2021). In this work, we adopt PointConvFormer into the PointPWC-Net (Wu et al., 2020), which utilizes a coarse-to-fine framework for scene flow estimation. PointPWC-Net (Wu et al., 2020) is a coarse-to-fine network design, which aims to iteratively refine the scene flow estimation. To adopt our PointConvFormer to the PointPWC-Net, we replace the PointConv in the feature pyramid layers with the PointConvFormer and keep the rest of the structure the same as the original version of PointPWC-Net.

We name the new network *'PCFPWC-Net'* where PCF stands for PointConvFormer. To train the PCFPWC-Net we follow the training pipeline in (Wu et al., 2020). For a fair comparison, we use the same dataset configurations as in (Wu et al., 2020). The model is first trained on FlyingThings3D (Mayer et al., 2016b), which is a large synthetic image dataset for scene flow estimation. The 3D point clouds are reconstructed from image pairs with the depth map provided in the dataset following (Gu et al., 2019). We adopt the same hyper-parameters used in (Wu et al., 2020). There are 4 pyramid levels in PCFPWC-Net. The model is trained with a starting learning rate of 0.001 and dropped by half every 80 epochs. After training on FlyingThings3D, we directly evaluate the trained model on the real world KITTI Scene Flow dataset (Menze et al., 2015; 2018) to test the generalization capabilities of our model. We follow the same preprocessing step in (Gu et al., 2019) and obtain 142 valid scenes for evaluation. For comparison, we use the same metrics as (Wu et al., 2020). Definitions of the metrics can be found in the appendix.

Table 3: **Evaluation results on Scene Flow Datasets.** All approaches are trained on FlyingThings3D with the supervised loss. On KITTI, the models are directly evaluated on KITTI without any fine-tuning.

| Dataset | Method | EPE3D(m)↓ | Acc3DS↑ | Acc3DR↑ | Outliers3D↓ | EPE2D(*px*)↓ | Acc2D↑ |
|---|---|---|---|---|---|---|---|
| Flyingthings3D | FlowNet3D (Liu et al., 2019a) | 0.1136 | 0.4125 | 0.7706 | 0.6016 | 5.9740 | 0.5692 |
| | SPLATFlowNet (Su et al., 2018) | 0.1205 | 0.4197 | 0.7180 | 0.6187 | 6.9759 | 0.5512 |
| | HPLFlowNet (Gu et al., 2019) | 0.0804 | 0.6144 | 0.8555 | 0.4287 | 4.6723 | 0.6764 |
| | HCRF-Flow (Li et al., 2021a) | 0.0488 | 0.8337 | 0.9507 | 0.2614 | 2.5652 | 0.8704 |
| | FLOT (Puy et al., 2020) | 0.052 | 0.732 | 0.927 | 0.357 | - | - |
| | PV-RAFT (Wei et al., 2021) | 0.0461 | 0.8169 | 0.9574 | 0.2924 | - | - |
| | PointPWC-Net (Wu et al., 2020) | 0.0588 | 0.7379 | 0.9276 | 0.3424 | 3.2390 | 0.7994 |
| | PCFPWC-Net(ours) | **0.0416** | **0.8645** | **0.9658** | **0.2263** | **2.2967** | **0.8871** |
| KITTI | FlowNet3D (Liu et al., 2019a) | 0.1767 | 0.3738 | 0.6677 | 0.5271 | 7.2141 | 0.5093 |
| | SPLATFlowNet (Su et al., 2018) | 0.1988 | 0.2174 | 0.5391 | 0.6575 | 8.2306 | 0.4189 |
| | HPLFlowNet (Gu et al., 2019) | 0.1169 | 0.4783 | 0.7776 | 0.4103 | 4.8055 | 0.5938 |
| | HCRF-Flow (Li et al., 2021a) | 0.0531 | 0.8631 | 0.9444 | 0.1797 | 2.0700 | 0.8656 |
| | FLOT (Puy et al., 2020) | 0.056 | 0.755 | 0.908 | 0.242 | - | - |
| | PV-RAFT (Wei et al., 2021) | 0.0560 | 0.8226 | **0.9372** | 0.2163 | - | - |
| | PointPWC-Net (Wu et al., 2020) | 0.0694 | 0.7281 | 0.8884 | 0.2648 | 3.0062 | 0.7673 |
| | PCFPWC-Net(ours) | **0.0479** | **0.8659** | 0.9332 | **0.1731** | **1.7943** | **0.8924** |

From Table 3, we can see that PCFPWC-Net outperforms previous methods in almost all the evaluation metrics. Comparing with PointPWC-Net (Wu et al., 2020), our PCFPWC-Net achieves around 10% improvement in all metrics. On the KITTI dataset, our PCFPWC-Net also shows strong result for scene flow estimation by improving the EPE3D by more than $30\%(0.0694 \mapsto 0.0479)$ over PointPWC-Net (Wu et al., 2020)).

### 4.4 Visualization of Reweighted Scores

In order to actually see what the reweighted score learnt from the dataset, we visualize the difference of the learned attention for a few example scenes in the ScanNet (Dai et al., 2017) dataset. The difference is computed by $\max_{x_i \in \mathcal{N}(x_0)} \psi(x_i) - \min_{x_i \in \mathcal{N}} \psi(x_0)$, where $\psi$ is the attention score. A larger difference indicates that some points are discarded from the neighborhood. A smaller difference indicates a nearly constant $\psi$ in the neighbourhood, where PointConvFormer would reduce to regular point convolution. We visualize the difference in Fig. 2(b). where it can be seen that larger differences happen mostly in object boundaries. For smooth surfaces and points from the same class, the difference of reweighted scores is low. This visualization further confirms that PointConvFormer is able to utilize feature differences to conduct neighborhood filtering.

## 5 Conclusion

In this work, we propose a novel point cloud layer, PointConvFormer, which can be widely used in various computer vision tasks. Unlike traditional convolution of which convolutional kernels are functions of the relative position, the convolutional weights of the PointConvFormer are modified by an attention score computed from feature differences and relative position. By taking the feature differences into account, PointConvFormer incorporates benefits of attention models, which could help the network to focus on points with high feature correlation during feature encoding. Experiments on a number of point cloud tasks showed that PointConvFormer significantly outperforms traditional point-based operations and outperforms other voxel-based or point-voxel fusion approaches, with a significantly faster and smaller model than those.

## REFERENCES


Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019a.

Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9297–9307, 2019b.

Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1907–1915, 2017.

Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11030–11039, 2020.

Ran Cheng, Ryan Razani, Ehsan Taghavi, Enxu Li, and Bingbing Liu. 2-s3net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12547–12556, 2021.

Hung-Yueh Chiang, Yen-Liang Lin, Yueh-Cheng Liu, and Winston H Hsu. A unified point-based framework for 3d segmentation. In *2019 International Conference on 3D Vision (3DV)*, pp. 155–163. IEEE, 2019.

Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3075–3084, 2019.

Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Conditional positional encodings for vision transformers. *arXiv preprint arXiv:2102.10882*, 2021.

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning so (3) equivariant representations with spherical cnns. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 52–68, 2018.

A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 3354–3361, 2012.

Nidhi Goyal, Niharika Sachdeva, Anmol Goel, Jushaan Singh Kalra, and Ponnurangam Kumaraguru. Kcnet: Kernel-based canonicalization network for entities in recruitment domain. In *International Conference on Artificial Neural Networks*, pp. 157–169. Springer, 2021.

Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. *CVPR*, 2018.

Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3254–3263, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3464–3473, 2019.

Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020a.

Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11108–11117, 2020b.

Wenbo Hu, Hengshuang Zhao, Li Jiang, Jiaya Jia, and Tien-Tsin Wong. Bidirectional projection network for cross dimension scene understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14373–14382, 2021a.

Zeyu Hu, Xuyang Bai, Jiaxiang Shang, Runze Zhang, Jiayu Dong, Xin Wang, Guangyuan Sun, Hongbo Fu, and Chiew-Lan Tai. Vmnet: Voxel-mesh network for geodesic-aware 3d semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15488–15498, 2021b.

Frédéric Huguet and Frédéric Devernay. A variational method for scene flow estimation from stereo sequences. In *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–7. IEEE, 2007.

Varun Jampani, Martin Kiefel, and Peter V Gehler. Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4452–4461, 2016.

Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. *Advances in neural information processing systems*, 29:667–675, 2016.

Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5010–5019, 2018.

Deyvid Kochanov, Fatemeh Karimi Nejadasl, and Olaf Booij. Kprnet: Improving projection-based lidar semantic segmentation. *arXiv preprint arXiv:2007.12668*, 2020.

Abhijit Kundu, Xiaoqi Yin, Alireza Fathi, David Ross, Brian Brewington, Thomas Funkhouser, and Caroline Pantofaru. Virtual multi-view fusion for 3d semantic segmentation. In *European Conference on Computer Vision*, pp. 518–535. Springer, 2020.

Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3d point cloud segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8500–8509, 2022.

Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12697–12705, 2019.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pp. 3744–3753. PMLR, 2019.

Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3d lidar using fully convolutional network. *arXiv preprint arXiv:1608.07916*, 2016.

Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9267–9276, 2019.

Ruibo Li, Guosheng Lin, Tong He, Fayao Liu, and Chunhua Shen. Hcrf-flow: Scene flow from point clouds with continuous high-order crfs and position-aware flow embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 364–373, 2021a.

Xingyi Li, Fuxin Li, Xiaoli Fern, and Raviv Raich. Filter shaping for convolutional neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Xingyi Li, Wenxuan Wu, Xiaoli Z Fern, and Li Fuxin. The devils in the point clouds: Studying the robustness of point cloud convolutions. *arXiv preprint arXiv:2101.07832*, 2021b.

Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31, 2018.

Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 529–537, 2019a.

Xinhai Liu, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Point2sequence: Learning the shape representation of 3d point clouds with an attention-based sequence to sequence network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 8778–8785, 2019b.

Ningning Ma, Xiangyu Zhang, Jiawei Huang, and Jian Sun. Weightnet: Revisiting the design space of weight networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16*, pp. 776–792. Springer, 2020.

Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3d point cloud understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1578–1587, 2019.

Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 922–928. IEEE, 2015.

N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016a. URL `http://lmb.informatik.uni-freiburg.de/Publications/2016/MIFDB16`. arXiv:1512.02134.

Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4040–4048, 2016b.

Iaroslav Melekhov, Aleksei Tiulpin, Torsten Sattler, Marc Pollefeys, Esa Rahtu, and Juho Kannala. Dgc-net: Dense geometric correspondence network. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1034–1042. IEEE, 2019.

Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3061–3070, 2015.

Moritz Menze, Christian Heipke, and Andreas Geiger. Joint 3d estimation of vehicles and scene flow. *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences*, 2: 427, 2015.

Moritz Menze, Christian Heipke, and Andreas Geiger. Object scene flow. *ISPRS Journal of Photogrammetry and Remote Sensing*, 140:60–76, 2018.

Alexey Nekrasov, Jonas Schult, Or Litany, Bastian Leibe, and Francis Engelmann. Mix3d: Out-of-context data augmentation for 3d scenes. In *2021 International Conference on 3D Vision (3DV)*, pp. 116–125. IEEE, 2021.

Chunghyun Park, Yoonwoo Jeong, Minsu Cho, and Jaesik Park. Fast point transformer. *arXiv preprint arXiv:2112.04702*, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Gilles Puy, Alexandre Boulch, and Renaud Marlet. Flot: Scene flow on point clouds guided by optimal transport. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*, pp. 527–544. Springer, 2020.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017a.

Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017b.

Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. *Advances in Neural Information Processing Systems*, 32, 2019.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3693–3702, 2017.

Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1746–1754, 2017.

Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pp. 945–953, 2015.

Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2530–2539, 2018.

Hang Su, Varun Jampani, Deqing Sun, Orazio Gallo, Erik Learned-Miller, and Jan Kautz. Pixel-adaptive convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11166–11175, 2019.

Domen Tabernik, Matej Kristan, and Aleš Leonardis. Spatially-adaptive filter units for compact and efficient deep neural networks. *International Journal of Computer Vision*, 128(8):2049–2067, 2020.

Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *European conference on computer vision*, pp. 685–702. Springer, 2020.

Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6411–6420, 2019.

Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pp. 282–298. Springer, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Christoph Vogel, Konrad Schindler, and Stefan Roth. 3d scene flow estimation with a piecewise rigid scene model. *International Journal of Computer Vision*, 115(1):1–28, 2015.

Chu Wang, Babak Samari, and Kaleem Siddiqi. Local spectral graph convolution for point set feature learning. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 52–66, 2018a.

Jiaqi Wang, Kai Chen, Rui Xu, Ziwei Liu, Chen Change Loy, and Dahua Lin. Carafe: Content-aware reassembly of features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3007–3016, 2019.

Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2589–2597, 2018b.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7794–7803, 2018c.

Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic, faster and stronger. *arXiv e-prints*, pp. arXiv–2003, 2020.

Yi Wei, Ziyi Wang, Yongming Rao, Jiwen Lu, and Jie Zhou. Pv-raft: Point-voxel correlation fields for scene flow estimation of point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6954–6963, 2021.

Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*, 2019a.

Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9621–9630, 2019b.

Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation. In *European Conference on Computer Vision*, pp. 88–107. Springer, 2020.

Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4606–4615, 2018.

Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 87–102, 2018.

Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5589–5598, 2020.

Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. *arXiv preprint arXiv:1904.04971*, 2019a.

Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3323–3332, 2019b.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019c.

Julio Zamora Esquivel, Adan Cruz Vargas, Paulo Lopez Meyer, and Omesh Tickoo. Adaptive convolutional kernels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

Feihu Zhang, Jin Fang, Benjamin Wah, and Philip Torr. Deep fusionnet for point cloud semantic segmentation. In *European Conference on Computer Vision*, pp. 644–663. Springer, 2020a.

Yikang Zhang, Jian Zhang, Qiang Wang, and Zhao Zhong. Dynet: Dynamic convolution for accelerating convolutional neural networks. *arXiv preprint arXiv:2004.10694*, 2020b.

Zhiyuan Zhang, Binh-Son Hua, David W Rosen, and Sai-Kit Yeung. Rotation invariant convolutions for 3d point clouds deep learning. In *2019 International conference on 3d vision (3DV)*, pp. 204–213. IEEE, 2019.

Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10076–10085, 2020.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16259–16268, 2021.

Jingkai Zhou, Varun Jampani, Zhixiong Pi, Qiong Liu, and Ming-Hsuan Yang. Decoupled dynamic filter networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6647–6656, 2021.

Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9939–9948, 2021.

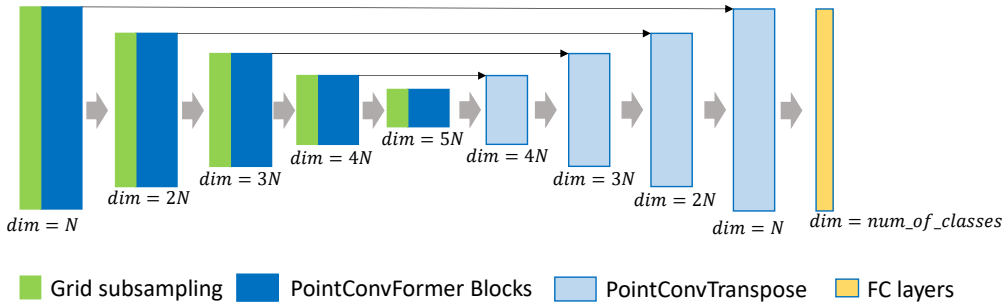



Figure 4: **The network structure of semantic segmentation.** We use a U-Net structure for semantic segmentation tasks. The U-Net contains 5 resolution levels. For each resolution level, we use grid subsampling to downsample the input point clouds, then followed by several pointconvformer residual blocks. For deconvolution, we just use PointConv as described in the main paper. We set $N = 64$ for ScanNet (Dai et al., 2017) Dataset and $N = 48$ for SemanticKitti (Behley et al., 2019a) Dataset. (Best viewed in color.)

# A Network Structure

## A.1 Network structure for semantic segmentation

As in Fig. 4, we use a U-Net structure for semantic segmentation tasks. The U-Net contains 5 resolution levels. For each resolution level, we use grid subsampling to downsample the input point clouds, then followed by several pointconvformer residual blocks. The number of layers in the blocks at each level is $[3, 2, 4, 6, 6]$ respectively. Latter blocks have more layers since they are cheaper to compute, similar to image convolutional models. For deconvolution, we just use PointConv as described in the main paper. And each block has a single PointConvTranspose layer, which is a PointConv layer that upsamples to locations without any features. For the $2cm$ grid resolution, because it is too fine to be captured by 5 downsampling levels, we utilize a sixth block which contains 2 layers.

## A.2 Network structure for scene flow estimation

Fig. 5 illustrates the network structure we used for scene flow estimation. Following the network structure of PointPWC-Net (Wu et al., 2020), which is a coarse-to-fine network design, the PCFPWC-Net also contains 5 modules, including the feature pyramid network, cost volume layers, upsampling layers, warping layers, and the scene flow predictors. We replace the PointConv in the Feature pyramid layers with the PointConvFormer and keep the rest of the structure the same as the original version of PointPWC-Net for fair comparison.

# B Evaluation Metrics for Scene flow estimation

**Evaluation Metrics.** For comparison, we use the same metrics as (Wu et al., 2020). Let $SF_\Theta$ denote the predicted scene flow, and $SF_{GT}$ be the ground truth scene flow. The evaluate metrics are computed as follows:

- *EPE3D(m)*: $\|SF_\Theta - SF_{GT}\|_2$ averaged over each point in meters.
- *Acc3DS*: the percentage of points with *EPE3D* $< 0.05m$ or relative error $< 5\%$.
- *Acc3DR*: the percentage of points with *EPE3D* $< 0.1m$ or relative error $< 10\%$.
- *Outliers3D*: the percentage of points with *EPE3D*$> 0.3m$ or relative error $> 10\%$.
- *EPE2D(px)*: 2D end point error obtained by projecting point clouds back to the image plane.
- *Acc2D*: the percentage of points whose *EPE2D* $< 3px$ or relative error $< 5\%$.

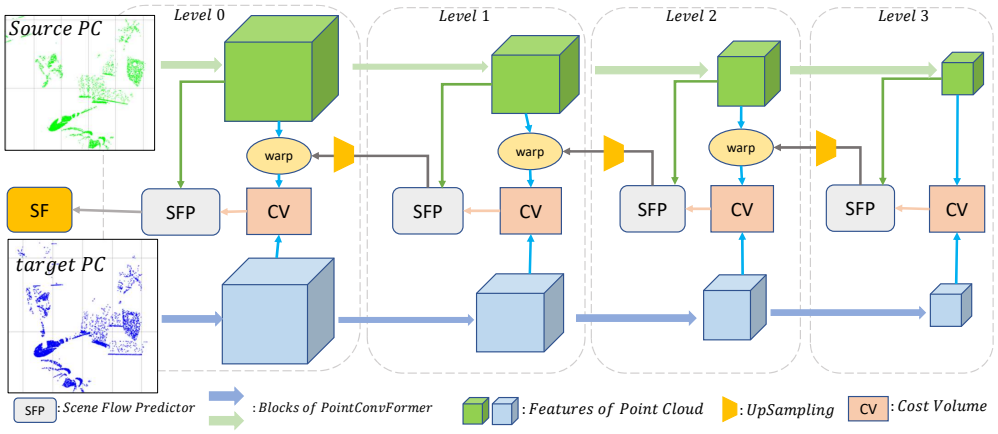



Figure 5: **The network structure of PointPWC-Net with PointConvFormer.** The feature pyramid is built with blocks of PointConvFormers. As a result, there are 4 resolution levels in the PointPWC-Net. At each level, the features of the source point cloud are warped according to the upsampled coarse flow. Then, the cost volume are computed using the warped source features and target features. Finally, the scene flow predictor predicts finer flow at the current level using a PointConv with features from the first point cloud, the cost volume, and the upsampled flow. (Best viewed in color.)

## C Detailed Results

Here we show the detailed result table corresponding to Figure 1. Our implementations of PointConv is better than the original implementation in (Wu et al., 2019b) in that we utilized the AdamW optimizer, and has more layers and channels than the original model. Overall we still achieved significant parameter savings over the original paper because of the ResNet-style architecture and the lightweight decoder we adopted.

Table 4: **Comparison with different input voxel size.** We compare the results on the ScanNet (Dai et al., 2017) validation set with different input voxel size. † means the results are reported in (Park et al., 2021). We use grid subsampling (Thomas et al., 2019) to downsample the input point clouds, which is similar to voxelization. However, we still use kNN neighborhood after downsampling which is different from the voxel neighborhood used in other approaches. * means we implemented it on the same network structure as PointConvFormer, hence it also serves as an ablation comparing regular self-attention layers and convolutional layers with PointConvFormer layers

| Methods | Voxel/grid size | # Params(M) | Input | mIoU(%) |
|---|---|---|---|---|
| MinkowskiNet42† (Choy et al., 2019) | 10cm | 37.9 | Voxel | 60.4 |
| PointConv* | 10cm | 5.4 | Point | 62.6 |
| Fast Point Transformer (Park et al., 2021) | 10cm | 37.9 | Voxel | 65.3 |
| PointConvFormer(ours) | 10cm | 5.5 | Point | **71.4** |
| MinkowskiNet42† (Choy et al., 2019) | 5cm | 37.9 | Voxel | 66.6 |
| Fast Point Transformer (Park et al., 2021) | 5cm | 37.9 | Voxel | 70.1 |
| PointConv* | 5cm | 5.4 | Point | 68.5 |
| PointConvFormer(ours) | 5cm | 5.5 | Point | **73.7** |
| MinkowskiNet42† (Choy et al., 2019) | 2cm | 37.9 | Voxel | 72.2 |
| Fast Point Transformer (Park et al., 2021) | 2cm | 37.9 | Voxel | 72.0 |
| PointConvFormer(ours) | 2cm | 9.4 | Point | **74.5** |

**Different Attention Types** We compare the PointConvFormer with a Point Transformer and no attention using the same architecture – same ResNet structure and same number of layers, just different base layer. The results in Table 5 show that the PointConvFormer attention is significantly better than Point Transformer attention as well as convolution without attention.

Table 5: **Different Attention Types** With the same model architecture and training parameters we change the attention layer of the model. The experiment is performed at the 5cm voxel grid level

| Attention Type | # Params (M) | mIoU(%) |
|---|---|---|
| PointTransformer Attention | 5.5 | 70.0 |
| No Attention (VI-PointConv only) | 5.4 | 72.8 |
| PointConvFormer | 5.5 | **73.7** |

## D Ablation Studies

In this section, we perform thorough ablation experiments to investigate our proposed PointConvFormer. The ablation studies are conducted on the ScanNet (Dai et al., 2017) dataset. For efficiency, we downsample the input point clouds with a grid-subsampling method (Thomas et al., 2019) with a grid size of $10cm$ as in (Park et al., 2021).

**Number of neighbours.** We first conduct experiments on the neighbourhood size $k$ in the PointConvFormer for feature aggregation. The results are reported in Table. 6. The best result is achieved with a neighbourhood size of 16. Larger neighbourhood sizes of $32, 48$ do not introduce significant gains on the result, and $48$ actually decreased the performance a bit, which may be caused by introducing excessive less relevant features in the neighbourhood (Zhao et al., 2021). We choose $16$ based on similar performance to 32 and significantly smaller memory footprint and faster speed.

Table 6: **Ablation Study.** Number of neighbours in each local neighbourhood.

| Nieghbourhood Size | 4 | 8 | 16 | 32 | 48 |
|---|---|---|---|---|---|
| mIoU(%) | 64.61 | 69.54 | **71.40** | 71.19 | 69.84 |

Table 7: **Ablation Study.** Number of heads.

| Number of Head | 2 | 4 | 8 | 16 |
|---|---|---|---|---|
| mIoU(%) | 70.71 | 70.58 | **71.40** | 70.97 |

**Number of heads in $\psi$.** As described in Sec. 3.2, our PointConvFormer could employ the multi-head mechanism to further improve the representation capabilities of the model. We conduct ablation experiments on the number of heads in the PointConvFormer. The results are shown in Table. 7. From Table. 7, we find that PointConvFormer achieves the best result with 8 heads.

**Decoder $c_{mid}$** PointConv and PointConvFormer implementations lead to a dimensionality expansion of the network that is $c_{mid}$ times the size of the input dimensionality, hence significantly increase the number of parameters in the subsequent linear layer $W_l$ (eqs. (3,7)). One empirical contribution in semantic segmentation we made is that we found that the decoder does not really need this dimensionality expansion, which leads to significant savings in the number of parameters. In Table 8, it is shown that adding 3 million parameters by using a $c_{mid}$ of 16 in the decoder only leads to a small improvement of $0.4\%$, hence in our model we choose to set $c_{mid} = 1$ in the decoder of segmentation models, since those parameters could be used better elsewhere. Parameter savings here and the ResNet-style blocks allow us to use more layers yet still have a smaller model than (Li et al., 2021b).

| $c_{mid}$ in decoder | 1 | 3 | 4 | 8 | 16 |
|---|---|---|---|---|---|
| mIoU (%) | 71.4 | 71.5 | 70.8 | 71.5 | **71.8** |
| # Params (M) | **5.48** | 5.90 | 6.11 | 6.96 | 8.64 |

Table 8: Different $c_{mid}$ in the decoder. $c_{mid}$ of 1 in the decoder did not significantly lower the performance, yet saves a significant amount of parameters, hence we choose it in the final model

**Different attention method.** As described in Sec. 3.2, our PointConvFormer is a combination of convolution and transformer. However, we adopted the subtractive attention $\psi(X_i - X_j)$ rather than the regular dot-product attention based on $Q$ and $K$ with regular transformers. Here we show the results comparing these formulations. The dot product version of the attention is shown in this equation:

$$X'_p = \sum_{p_i \in \mathcal{N}(p)} w(p_i - p)^\top \psi(\frac{1}{\sqrt{d}} \mathbf{q}(X_{p_i}) \mathbf{k}(X_p)) X_{p_i} \tag{8}$$

. where $d$ is the dimension of the $q$ and $k$ transforms of the input feature. Hence, the computational cost and memory usage of eq. (8) are slightly smaller. We show the results comparing these formu-

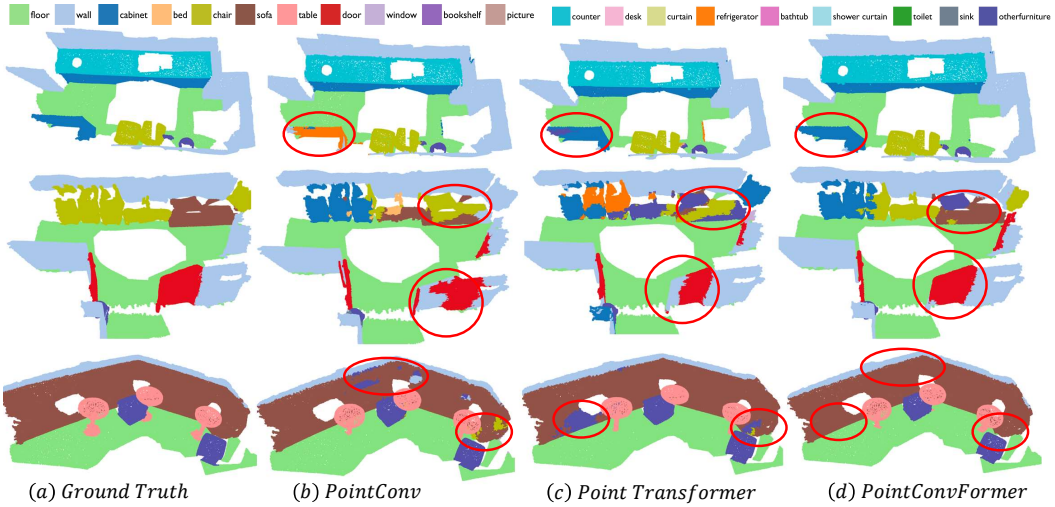
lations in Table. 9. The experiment results show that the feature difference achieves better results, which are also confirmed in (Zhao et al., 2021). Note that we were not able to make the version using softmax attention to work better than $60\%$ despite multiple trials. But the one with sigmoid as $\psi(\cdot)$ after the dot product easily eclipsed $70\%$, hence we utilize that. The QKV version has a bit more parameters due to the two MLPs for $Q$ and $K$ instead of a single one as in eq.(6). It in principle uses a bit less memory and computation during inference, but the savings is not very significant due to the small neighborhood size of $K = 16$.

Table 9: **Ablation Study.** Dot-product attention vs. additive attention.

| Method (*same backbone*) | #Params(M) | mIoU(%) |
|---|---|---|
| PointConvFormer(dot product attention) | 5.6 | 70.10 |
| PointConvFormer(subtractive attention) | 5.5 | 71.40 |

## E Result Visualizations

Fig. 6 and Fig. 7 are visualizations of the comparison among PointConv (Wu et al., 2019b), Point Transformer (Zhao et al., 2021) and PointConvFormer on the ScanNet dataset (Dai et al., 2017). We observe that PointConvFormer is able to achieve better predictions with fine details comparing with PointConv (Wu et al., 2019b) and Point Transformer (Zhao et al., 2021). Interestingly, it seems that PointConvFormer is usually able to find the better prediction out of PointConv (Wu et al., 2019b) and Point Transformer (Zhao et al., 2021), showing that its novel design brings the best out of both operations.




(a) *Ground Truth* (b) *PointConv* (c) *Point Transformer* (d) *PointConvFormer*

Figure 6: **ScanNet result visualization.** We visualize the ScanNet prediction results from our PointConvFormer, PointConv (Wu et al., 2019b) and Point Transformer (Zhao et al., 2021). The **red** ellipses indicates the improvements of our PointConvFormer over other approaches. Points with ignore labels are filtered for a better visualization. (Best viewed in color)

Fig. 8 illustrates the prediction of PointConvFormer on the SemanticKitti dataset (Behley et al., 2019a). Fig. 9, Fig. 11 and Fig.10 are the comparison between the prediction of PointPWC (Wu et al., 2020) and PCFPWC-Net on the FlyingThings3D (Mayer et al., 2016a) and the KITTI Scene Flow 2015 dataset (Menze et al., 2015). Please also refer to the video for better visualization.

Fig.  illustrates the qualitative results of PCFPWC-Net for both FlyingThings3D and KITTI dataset.

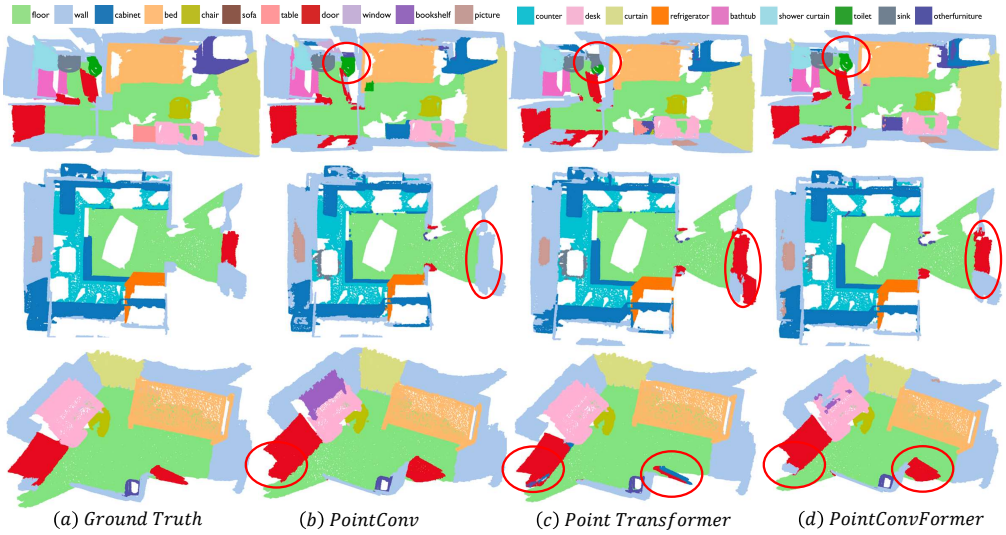



Figure 7: **ScanNet result visualization.** We visualize the ScanNet prediction results from our PointConvFormer, PointConv (Wu et al., 2019b) and Point Transformer (Zhao et al., 2021). The **red** ellipses indicates the improvements of our PointConvFormer over other approaches. Points with ignore labels are filtered for a better visualization. (Best viewed in color)

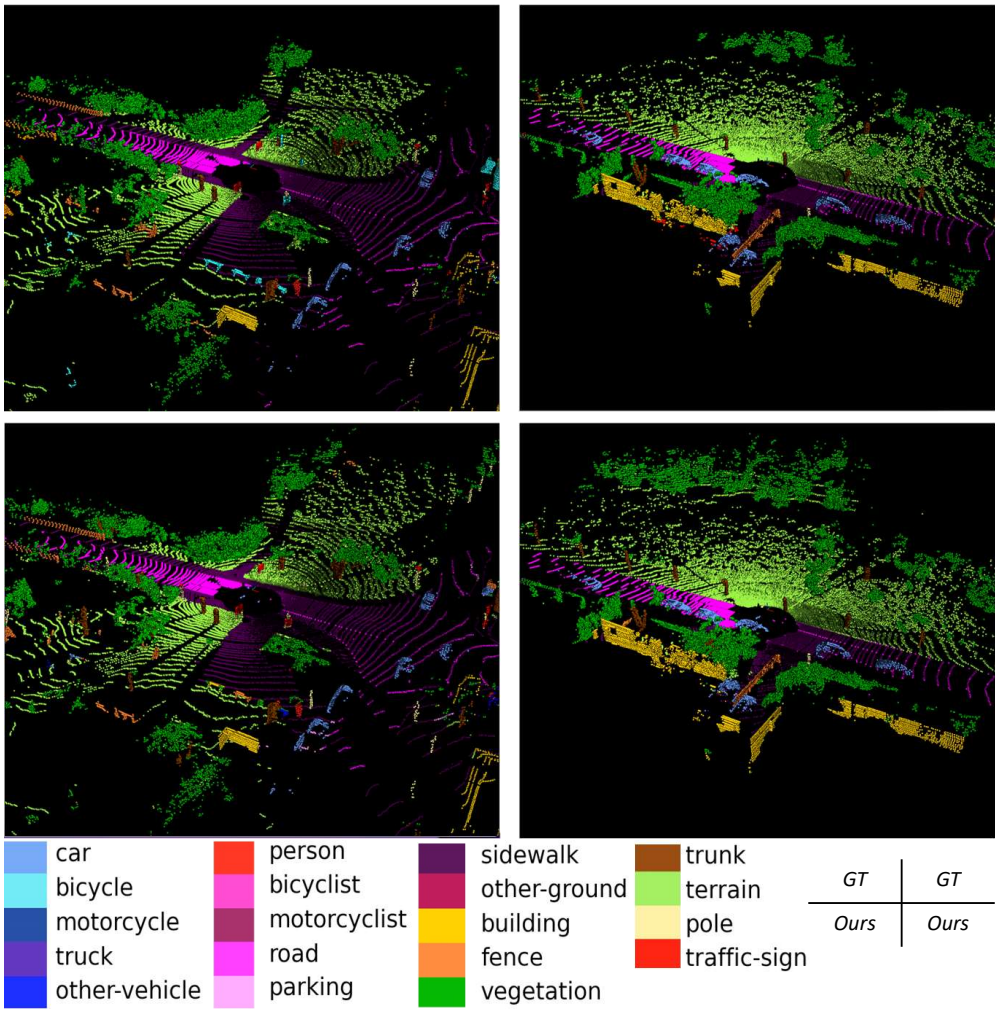



Figure 8: **SemanticKitti result visualization.** We visualize the SemanticKitti prediction results from our PointConvFormer. Each column is a scan from SemanticKitti validation set. The first row is the input, the second row is the ground truth, the third row is our prediction. (Best viewed in color.)

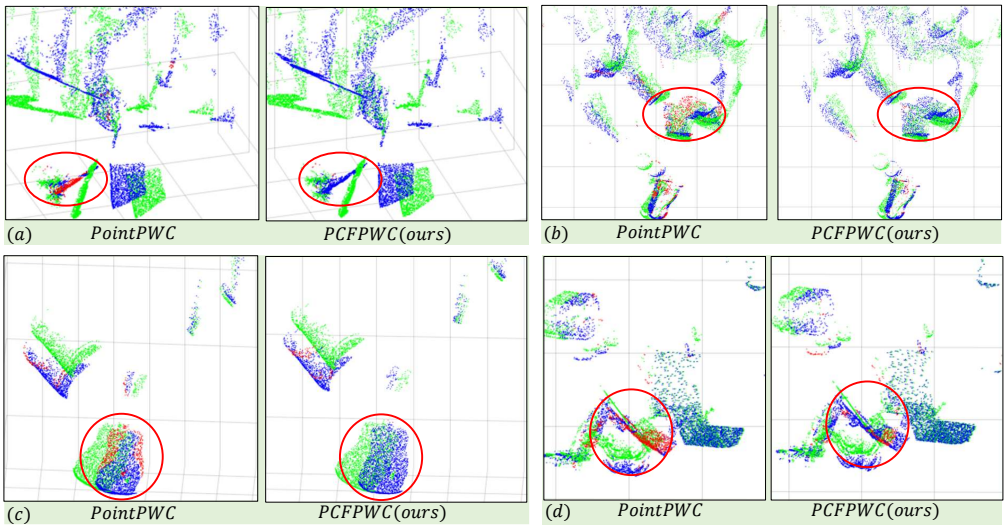



Figure 9: **Qualitative comparison between PointPWC-Net and PCFPWC-Net (FlyingThings3D (Mayer et al., 2016b)).** (a) is the visualization of the FlyingThings3D dataset. (b) is the visualization of the KITTI dataset. Green points are the source point cloud. Blue points are the points warped by the correctly predicted scene flow. The predicted scene flow belonging to Acc3DR is regarded as a correct prediction. For the points with incorrect predictions, we use the ground truth scene flow to warp them and the warped results are shown as red points. (Best viewed in color.)

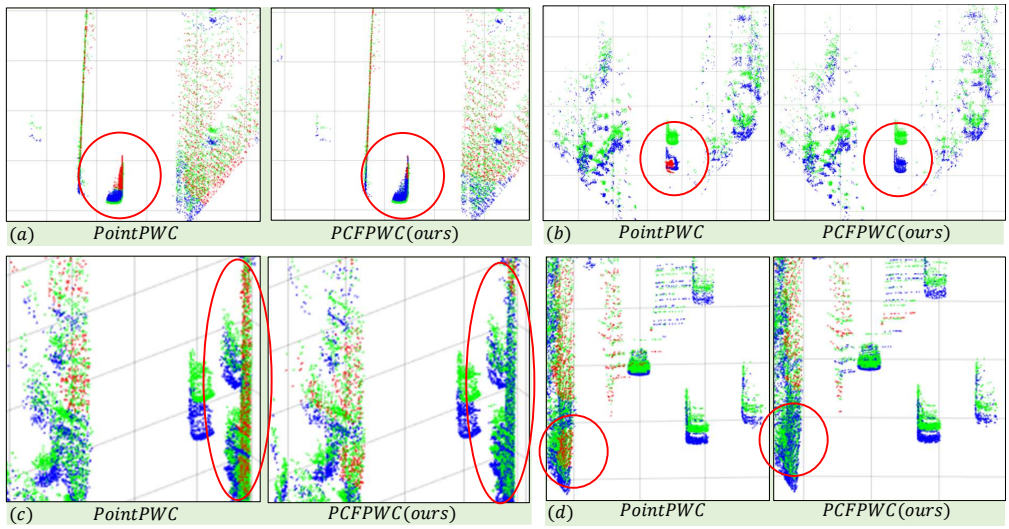



Figure 10: **Qualitative comparison between PointPWC-Net and PCFPWC-Net (KITTI (Menze et al., 2015)).** Green points are the source point cloud. Blue points are the points warped by the correctly predicted scene flow. The predicted scene flow belonging to Acc3DR is regarded as a correct prediction. For the points with incorrect predictions, we use the ground truth scene flow to warp them and the warped results are shown as red points. (d) is a failure case, where the points on the wall or ground/road are hard to find accurate correspondences for both PointPWC and PCFPWC. (Best viewed in color.)

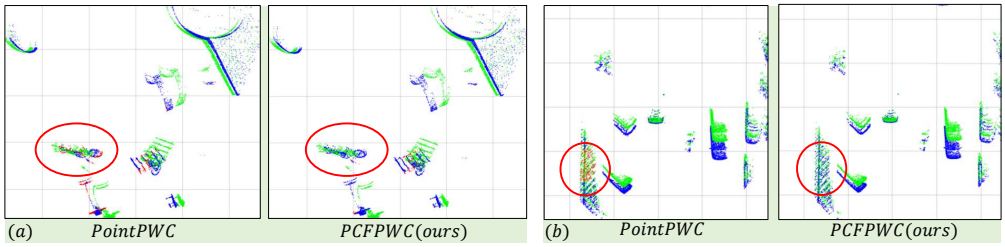



Figure 11: **Qualitative comparison between PointPWC-Net and PCFPWC-Net.** (a) is the visualization of the FlyingThings3D dataset. (b) is the visualization of the KITTI dataset. Green points are the source point cloud. Blue points are the points warped by the correctly predicted scene flow. The predicted scene flow belonging to Acc3DR is regarded as a correct prediction. For the points with incorrect predictions, we use the ground truth scene flow to warp them and the warped results are shown as red points. (Best viewed in color.)