

Supplementary Material

Hanz Cuevas-Velasquez¹

hanz.c.v@ed.ac.uk

Antonio Javier Gallego²

jgallego@dlsi.ua.es

Robert B. Fisher¹

rbf@inf.ed.ac.uk

¹ School of Informatics
University of Edinburgh
Edinburgh, UK

² Department of Software and Computing
Systems
University of Alicante,
Alicante, Spain

Abstract

This supplementary material is organized as follows: Section A demonstrates that our geometric and latent heads are a special case of a multi-head attention layer in the transformers literature. Section B shows an example of the prediction of each auxiliary output of the network. Section C describes the local aggregation step of our two-headed layer and how it relates to graph neural networks. In Section D the geometric and latent attention scores of our Ge-Latto layer are shown. Finally, Section E extends the ablation study presented in the main paper.

A Special Case of Multi-head Attention Transformer

Each head (geometric or latent) in our attention mechanism can be considered as a multi-head attention layer with feature dimension (channels) $D' = 1$ for each head or number of heads $n = D$, being D the dimensionality of the features in each layer. This is demonstrated as follows. Considering Eq. 1, the multi-head equation of a feature vector $H_i \in \mathbb{R}^D$ proposed by Vaswani *et al.* [1], where Q_k , R_k and V_k are the query, key and value vectors with size D' .

$$H_i = \text{Concat}(\text{head}_{i,1}, \text{head}_{i,2}, \text{head}_{i,3}, \dots, \text{head}_{i,n})$$

$$\text{head}_{i,n} = \sum_{k=1}^K ((\mathcal{M}\phi(\gamma(Q_k, R_k))) \odot V_k)$$

where :

\mathcal{M} = Replicates the vector D' times

ϕ = Normalization function

γ = Similarity function

$$\dim(\gamma(\cdot)) = 1$$

$$\dim(\text{head}_{i,n}) = D'$$

$$\dim(H_i) = D = nD'$$

K = Neighborhood size

(1)

Eq. 1 shows there is one weight ϕ per head n , and each ϕ multiplies D' feature channels. Therefore, **the feature $H_i \in \mathbb{R}^D$ has n weights ϕ and nD' feature channels**. In the case where the number of heads n is D , D' would have the value of 1. There would be one weight ϕ per head, and each weight would multiply one feature channel. **This would give a vector H_i with D weights ϕ and D feature channels**, which are the dimensions of our geometric and latent head features, as seen in Eq. 2 (latent feature equation), or Eq. 3 and Eq. 4 in the main paper.

$$H_i = \sum_{k=1}^K (\phi(f_{h_{att}}(h_k)) \odot h_k) \quad (2)$$

Where the dimension of $f_{h_{att}}(h_k)$, h_k and H_i is D .

B Auxiliary Loss Outputs

Each auxiliary loss optimizes the segmentation output of a sub-sampled version of the point cloud. Figure S1 shows an example of these outputs.

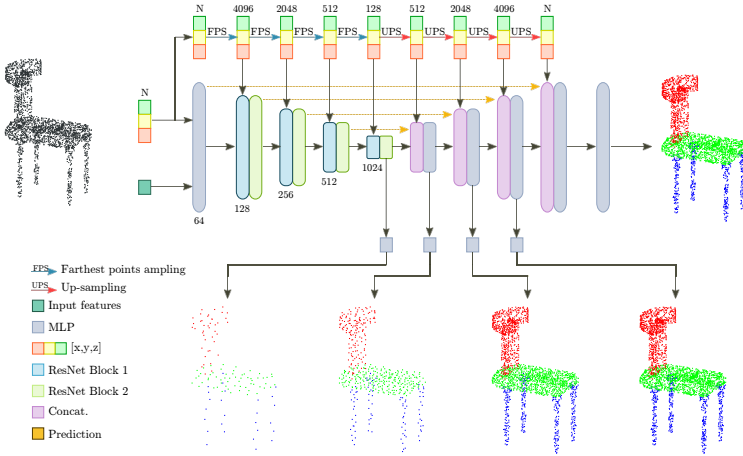


Figure S1: Auxiliary outputs. The network has 4 auxiliary outputs at multiple scales. One output comes from the last encoder layer and the other outputs plus the main output are obtained from the decoder layers.

C Gelatto as a Graph Neural Network

Our network performs local aggregation following steps similar to those that a graph neural network (GNN) uses to update the value of a node through message passing:

- Our network updates the information of each node (center point p_i) based on the information from its neighborhood q_k , preserving graph symmetries (permutation invariance).
- The weights (edges) between the center point and its neighbors are learned through our geometric and latent self-attentions.

- At every step (each new layer of the network), the information from each node is propagated to a further (or neighbor) node until almost all the global information is aggregated.

Table S1 shows this relationship graphically.

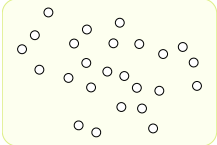
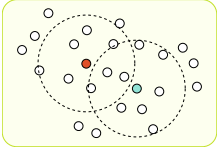
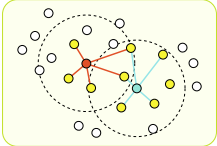
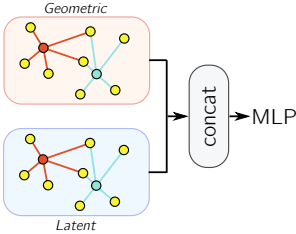
Illustration	Local aggregation (message passing) steps
	<ul style="list-style-type: none"> • Consider a set of points p_i in the point cloud P.
	<ul style="list-style-type: none"> • Our method obtains a local patch around a point by finding all the neighbor points inside a radius r. Here, we visualize the neighborhood of two points, red and cyan.
	<ul style="list-style-type: none"> • The patch is used to create a graph between the center points p_i (red and cyan) and their neighbors q_k (yellow). • To increase the robustness of our network, we randomly chose k neighbors for each centroid. It can be seen as randomly zeroing the value of an edge.
	<ul style="list-style-type: none"> • Our network uses self-attention (Eq. 3 and Eq. 4 in the main paper) to learn the edges that connect a centroid with its neighbors. • The information at a node is updated by aggregating the features from its neighbors and the learned edges. Gelatto creates two graphs, one for the geometric features, and another for the latent features. • After the <i>message passing</i> step, the updated geometric and latent features are concatenated and processed by an MLP layer.

Table S1: Local aggregation process of Gelatto.

D Geometric and Latent Attention Scores

To show the attention scores learned by our Ge-Latto layer after each encoder step, an input point, that was not discarded by the sampling process, was picked. Because the attention score of each point has a dimension D , where D is the dimensionality of a given layer, we randomly picked a value $d \in D$ per attention score to be shown in Figure S2 and Figure S3 for the geometric and latent heads, respectively.

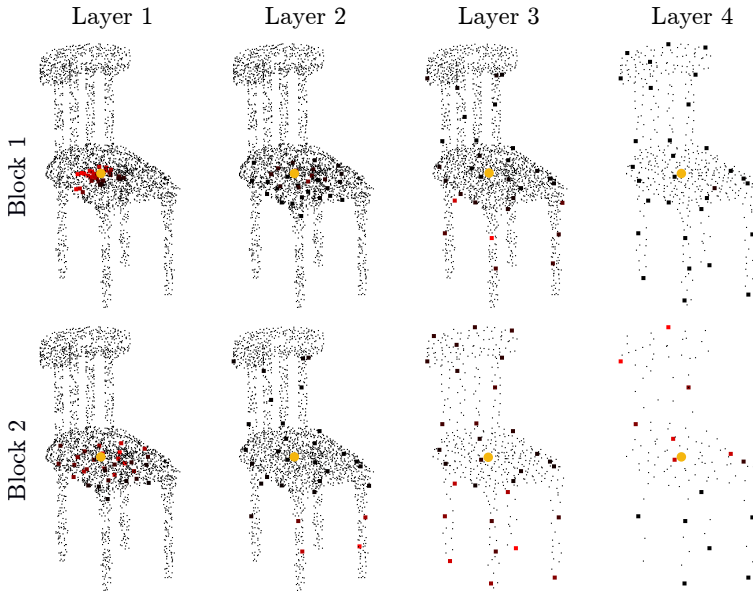


Figure S2: Learned geometric attention scores from a point (in yellow). The attention scores are represented in red, the stronger the intensity, the higher the score. The small black points are the sampled points at a given layer, the bigger points are the selected neighbor points inside a radius. The image shows the attention scores of the two ResNet Blocks at every encoder layer.

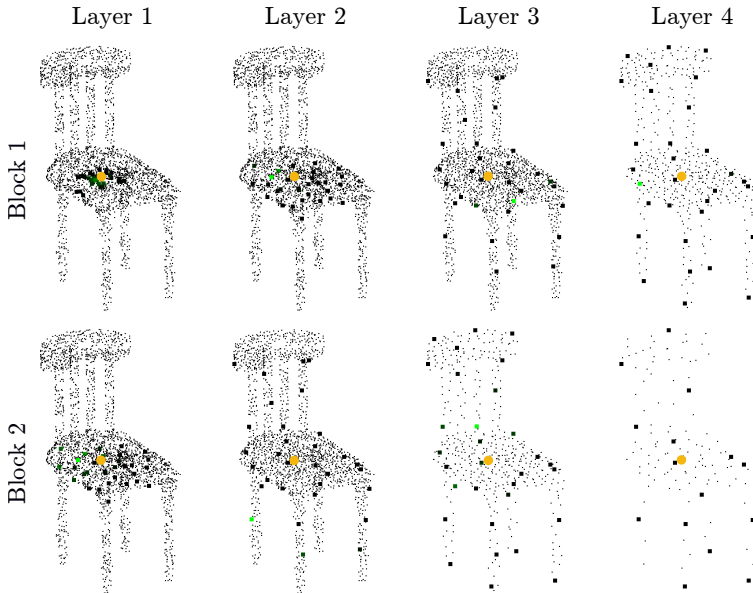


Figure S3: Learned latent attention scores from a point (in yellow). The attention scores are represented in green.

To observe how our network captures global and local relationships, all the attention scores of each head were grouped in Figure S4. The figure shows that, for the chosen point, the latent head focuses more on closer points, meanwhile, the geometric head not only pays attention to the local points but also to more distant points, such as those of the back of the chair and legs.

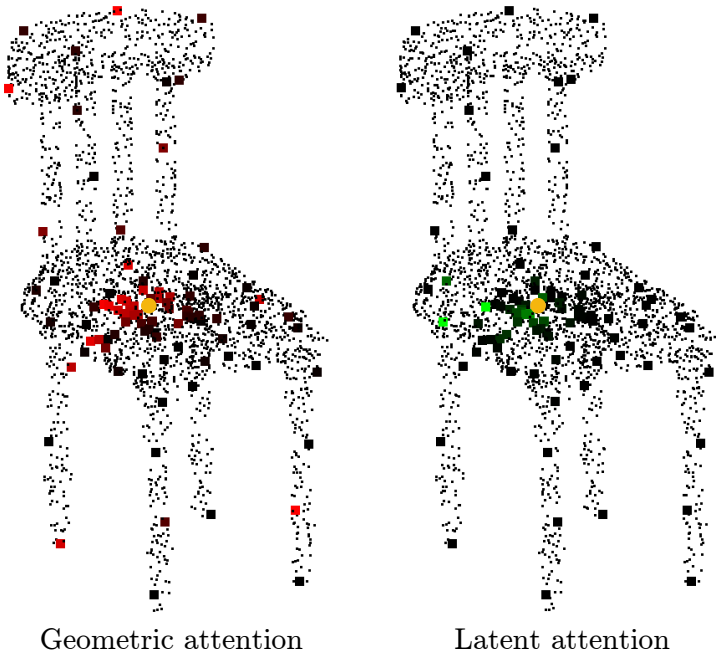


Figure S4: Geometric and Latent attention scores grouped.

E Extended Ablation Study

E.1 Auxiliary losses

In the main paper, we only considered the case when all the auxiliary weights α_i have the same value. This section explores different values for each alpha using grid search. The experiment consists of varying the alpha of one auxiliary loss, from 0 to 1 with increments of 0.2, and fixing the other alphas to 0.4 (the best value found before); the process is repeated for the 4 auxiliary losses. The values that the varying alpha can take are [0, 0.2, 0.6, 0.4, 0.8, 1], where 0 means that we do not minimize the loss for that output. We trained 24 variations of our network (4 auxiliary losses with 6 values for alpha). Each network was initialized with the weights from our best model, for the S3DIS area 5 dataset, and trained for 50 epochs. This experiment showed that there is no improvement when we vary the alphas individually and that the best value for this parameter is 0.4. However, we still observe that the use of auxiliary losses improves the performance of the network. As seen in Table 4 in the main paper, in the ablation study, when the network is trained without auxiliary losses, it obtains an IoU of 67.4%. Meanwhile, when the auxiliary losses are added, the performance increases to 69.2%.

Method	Parameters	mIoU
MinkowskiNet	21.7M	65.3
KPConv	25.8M	67.1
PCT	2.88M	61.3
FPConv	17.6M	62.7
Ge-Latto (ours)	15.3M	69.2

Table S2: Model parameters comparison table. The parameters are in millions and the metric is for the S3DIS dataset.

Number of points	6K	10K	20K	200K
Inference batch size	5	3	3	3
Inference time	100ms	200ms	210ms	300ms
Training batch size	2	-	-	-
Training time	160ms	-	-	-

Table S3: Training and inference time.

E.2 Model Size and Speed

Table S2 shows that our method achieves the state-of-the-art in the S3DIS dataset with fewer parameters than the previous methods. Our network has only 15.3M parameters, whereas KPConv has 25.8M, MinkowskiNet 21.7M, and FPConv 17.6M. The only network that has fewer parameters is PCT, 2.88M. However, we are 8% better in point cloud semantic segmentation and obtain a similar performance in ModelNet40.

The training and inference time using different numbers of points and batch sizes are shown in Table S3 reports. This table shows that it takes around 160ms to train 6144 points with a batch size of 2. At inference time, our network can analyze 6144 points with a batch size of 5 in 100ms, 20K points with a batch size of 3 in 210ms, and 200K points with a batch size of 3 in 300ms. All the tests were done using an NVIDIA RTX2080ti. These experiments show that our network is suitable for its applications that need a lighter network.

References

- [1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.