E(3)-Equivariant Mesh Neural Networks

Thuan Anh Trang *,1, Ngo Nhat Khang *,1, Daniel Levy *,2, Thieu N. Vo 3, Siamak Ravanbakhsh 2, and Truong Son Hy 1,4,†

1 FPT Software AI Center, 2 McGill University / MILA, 3 Ton Duc Thang University, 4 Indiana State University

* Co-first authors † Correspondence to TruongSon.Hy@indstate.edu

Abstract

Although recent works have addressed the need for geometric deep learning on 3D meshes, we observe that the complexities in many of these architectures do not translate to practical performance, and simple deep models for geometric graphs are competitive in practice. Motivated by this observation, we minimally extend the update equations of E(n)-Equivariant Graph Neural Networks (EGNNs) [1] to incorporate mesh face information, and further improve it to account for long-range interactions through hierarchy. The resulting architecture, Equivariant Mesh Neural Network (EMNN), outperforms other, more complicated equivariant methods on mesh tasks, with a fast run-time and no expensive preprocessing.

Background

Geometric Feature on Meshes When processing a mesh, our method accounts for its points, edges and faces. Each point $p \in \mathcal{P}$ is associated with its 3D coordinate vector $x_p \in \mathbb{R}^3$. Meanwhile, a face $f \in \mathcal{F}$ can have scalar and vector attributes corresponding to the area and normal vector, denoted by a_f and n_f respectively. Considering a triangle face $f = (p_1, p_2, p_3)$, the normal vector and the area are given by $n_f = (x_{p_2} - x_{p_1}) \times (x_{p_3} - x_{p_1})$ and $a_f = \frac{\|n_f\|}{2}$ respectively. By convention, we can generally choose the normal vector to point outwards.

Equivariant Graph Neural Networks A geometric graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with two main components: vertices $v_i \in \mathcal{V}$ and edges $e_{ij} \in \mathcal{E}$. Each vertex v_i is associated with scalar invariant features $h_i \in \mathbb{R}^d$ and n-dimensional equivariant coordinates $x_i \in \mathbb{R}^n$. To make the messages invariant to E(n) transformations, the formula for EGNN's layer is proposed as:

$$m_{ij} = \phi_e(h_i^l, h_j^l, \|x_i^l - x_j^l\|, e_{ij}), \tag{1}$$

where the superscript l denotes the layer number. Then, the scalar and vector features at the layer $l+1$ are updated by the following equations:

$$h_i^{l+1} = \phi_h(h_i^l, \sum_{(i,j) \in \mathcal{E}} m_{ij}) \tag{2}$$

$$x_i^{l+1} = x_i^l + \sum_{j \in \epsilon(i)} (x_i^l - x_j^l) \phi_x(m_{ij}), \tag{3}$$

Here, ϕ_h and ϕ_e are multi-layer perceptrons (MLPs) and $\epsilon(i) = \{j \mid (i,j) \in \mathcal{E}\}$ denotes the set of neighbours of node i.

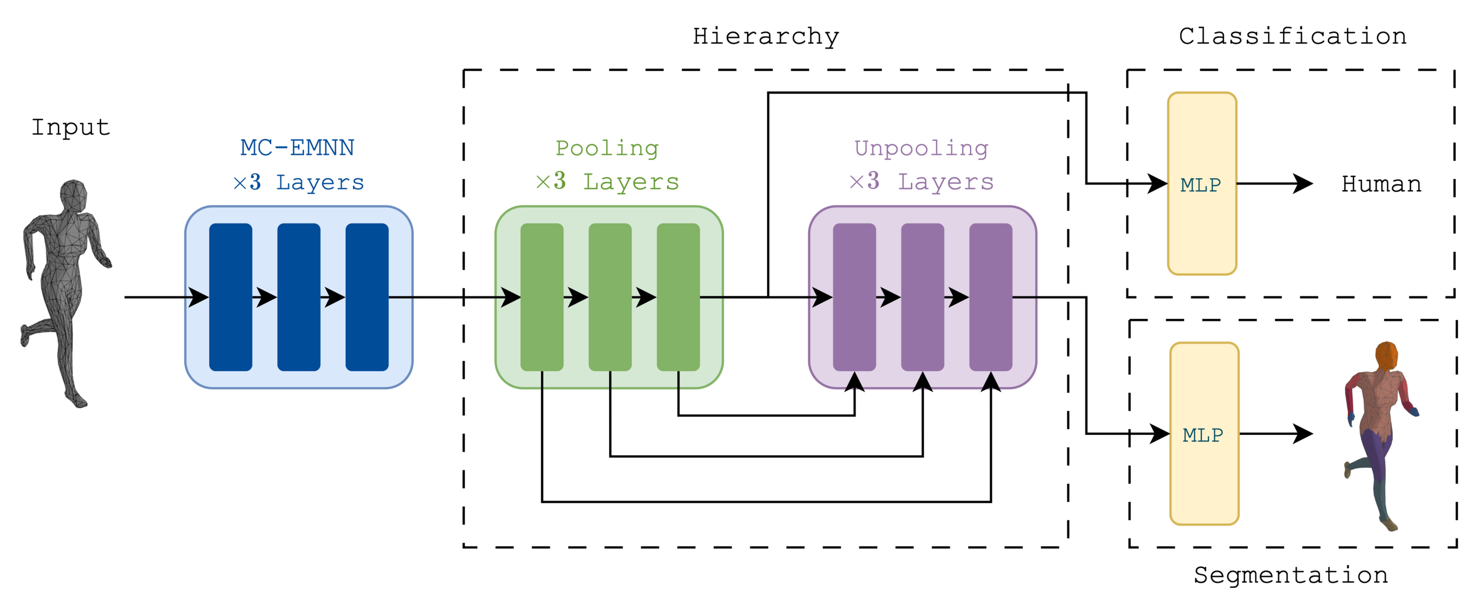

Figure 1: EMNN Architecture.

References

[1] Satorras, V. G., Hoogeboom, E., and Welling, M. (2021). E(n) equivariant graph neural networks. In M. Meila and T. Zhang, editors, Proceedings of the 38th International Conference on Machine Learn- ing, volume 139 of Proceedings of Machine Learning Research, pages 9323–9332. PMLR.

Method

Equivariant Graph Neural Networks Considering a triangle face (i, j, k), we define a surface-aware message from this face to node i as:

$$m_{ijk} = \phi_s\left(h_i^l, h_j^l + h_k^l, \|(x_j^l - x_i^l) \times (x_k^l - x_i^l)\|\right), \tag{4}$$

where ϕ_s is an MLP. The invariant feature for node i is created by aggregating all such messages from neighbouring faces $\tau(i) = \{(j,k) | (i,j,k) \in \mathcal{F}\}$, and neighbouring edges $\epsilon(i) = \{j | (i,j) \in \mathcal{E}\}$:

$$h_i^{l+1} = \phi_h\left(h_i^l, \sum_{j \in \epsilon(i)} m_{ij}, \sum_{(j,k) \in \tau(i)} m_{ijk}\right). \tag{5}$$

Here, the edge message m_{ij} is the same as EGNN in Eq. (4). The equivariant feature x_i^{l+1} is calculated similarly to EGNN update (5), with the difference that the normal to neighbouring faces is used as an equivariant vector in the update. This vector is scaled by the invariant factor based on m_{ijk}:

$$x_i^{l+1} = x_i^l + \sum_{j \in \epsilon(i)} (x_i^l - x_j^l) \phi_x(m_{ij}) + \sum_{j,k \in \tau(i)} \left((x_j^l - x_i^l) \times (x_k^l - x_i^l)\right) \phi_t(m_{ijk}). \tag{6}$$

Multiple Vector Channels We also design a multiple vector channels version for EMNN, in which the vector features $x_i \in \mathbb{R}^3$ are replaced by feature matrices $X_i \in \mathbb{R}^{3 \times c}$, where c is the number of vector channels:

$$m_{ijk} = \phi_s\left(h_i^l, h_j^l + h_k^l, \|(X_j^l - X_i^l) \otimes (X_k^l - X_i^l)\|_r\right). \tag{7}$$

Similarly, (6) becomes

$$X_i^{l+1} = X_i^l + \sum_{j \in \epsilon(i)} (X_i^l - X_j^l) \phi_x(m_{ij}) + \sum_{j,k \in \tau(i)} \left((X_j^l - X_i^l) \otimes (X_k^l - X_i^l)\right) \phi_t(m_{ijk}), \tag{8}$$

where, ϕ_x and ϕ_t produce invariant $c \times c$ *channel mixing matrices* as their output.

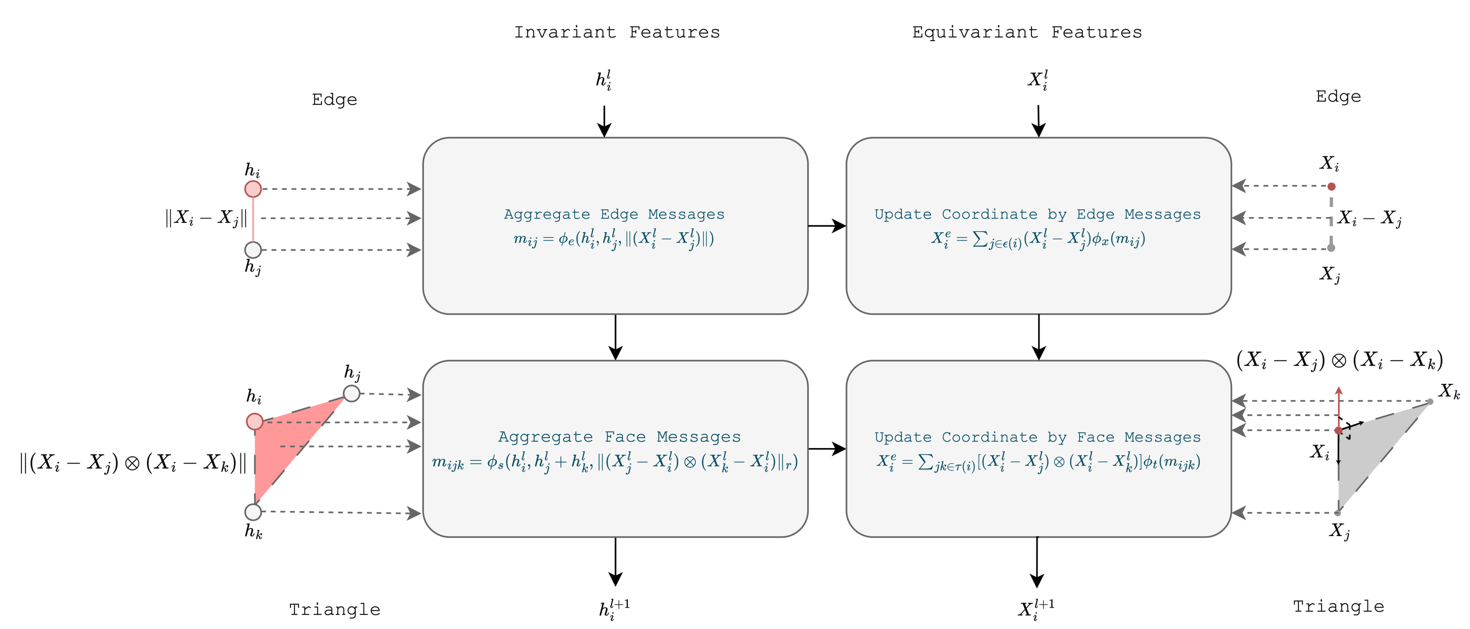

Figure 2: EMNN Layer.

Hierarchical Structure To facilitate long-range communications between nodes, after extracting information using MC-EMNN layers, we employ a hierarchical structure that pools and unpools feature at different resolutions. The pooling block is defined as:

$$h_i^{l+1} = \max_{j \in \mathcal{N}(i^{l+1})} (\phi_P(h_j^l)) \quad \text{where} \quad i^{l+1} \in \text{FPS}(i^l) \tag{9}$$

Here the neighbourhood of each vertex $\mathcal{N}(i)$ is gathered using a ball of radius r around that vertex. The formula for the Unpooling is :

$$h_i^{l-1} = \phi_U\left(\left[\frac{\sum_{j \in \text{KNN}(i^{l-1}, i^l)} \frac{1}{\|x_{ij}\|_2} h_j}{\sum_{j \in \text{KNN}(i^{l-1}, i^l)} \frac{1}{\|x_{ij}\|_2}}, h_i^{l-1}\right]\right). \tag{10}$$

Software

https://github.com/HySonLab/EquiMesh

Experiments

Model	Initial Features	Train	Test	Gauge	Rot-Tr-Ref-Scale	Perm
GEM-CNN	XYZ	99.42(0.15)	97.92(0.30)	96.90(0.25)	2.14(1.49)	97.92(0.30)
	GET	99.42(0.15)	98.03(0.17)	97.15(0.39)	1.47(1.60)	98.03(0.17)
	RELTAN [0.7]	99.69(0.05)	98.62(0.06)	98.04(0.12)	98.62(0.06)	98.62(0.06)
	RELTAN [0.5, 0.7]	99.70(0.09)	98.64(0.22)	97.99(0.18)	98.64(0.22)	98.64(0.22)
EMAN	XYZ	99.62(0.09)	98.46(0.15)	97.26(0.34)	0.02(0.00)	98.46(0.15)
	GET	99.60(0.08)	98.43(0.17)	97.32(0.46)	0.02(0.00)	98.43(0.17)
	RELTAN [0.7]	99.27(1.01)	98.13(1.19)	97.44(1.26)	98.13(1.19)	98.13(1.19)
	RELTAN[0.5, 0.7]	99.68(0.00)	98.66(0.07)	98.41(0.25)	98.66(0.07)	98.66(0.07)
EMNN (ours) + MC + Hier	XYZ + Normal	**100.00**(0.00)	**100.00**(0.00)	**100.00**(0.00)	**100.00**(0.00)	**100.00**(0.00)

Table 1: Results on FAUST dataset (Segmentation).

Model	Initial Features	Train	Test	Gauge	Rot-Tr-Ref-Scale
GEM-CNN	XYZ	97.78(2.41)	82.35(5.88)	82.35(5.88)	12.94(2.63)
	GET	90.79(2.84)	82.35(9.30)	82.35(9.30)	17.65(7.20)
	RELTAN[0.7]	93.97(4.26)	91.76(6.71)	91.76(6.71)	91.76(6.71)
	RELTAN[0.5, 0.7]	90.16(8.43)	89.41(14.65)	89.41(14.65)	89.41(14.65)
EMAN	XYZ	47.30(4.55)	42.35(20.55)	44.71(18.88)	12.94(2.63)
	GET	44.13(7.39)	42.35(11.31)	41.18(9.30)	10.59(2.63)
	RELTAN[0.7]	92.70(4.14)	94.12(4.16)	94.12(4.16)	94.12(4.16)
	RELTAN[0.5, 0.7]	97.46(4.14)	98.82(2.63)	98.82(2.63)	98.82(2.63)
EMNN + MC + Hier	XYZ + Normal	**100.00**(0.00)	**100.00**(0.00)	**100.00**(0.00)	**100.00**(0.00)

Table 2: Results on TOSCA dataset (Classification).

Method	Runtime	Memory	Split-16	Split-10	
GWCNN	—	—	96.6%	90.3%	Invariant
MeshCNN	50s	1.2GB	98.6%	91.0%	
PD-MeshNet	—	—	99.7%	99.1%	
MeshWalker	—	—	98.6%	97.1%	
HodgeNet	—	—	99.2%	94.7%	
SubdivNet	25s	0.9GB	99.9%	99.5%	Non-invariant
DiffusionNet	16s	1.0GB	—	99.5%	
Laplacian2Mesh	30s	2.8GB	**100%**	**100%**	
Mesh-MLP	—	—	**100%**	99.7%	
EMNN + MC + Hier	26s	1.2GB	***100%***	***100%***%	

Table 3: Results on SHREC dataset (Classification).

Method	Input	Runtime	Memory	Accuracy	
PointNet	point cloud	12s	1.2GB	74.7%	Point cloud
PointNet++	point cloud	10s	0.9GB	82.3%	
MeshCNN	mesh	137s	1.4GB	85.4%	Invariant
PD-MeshNet	mesh	—	—	85.6%	
HodgeNet	mesh	—	—	85.0%	
SubdivNet	mesh	100s	1.3GB	**91.7**%	Non-invariant
DiffusionNet	mesh	16s	2.0GB	90.3%	
Laplacian2Mesh	mesh	70s	4.8GB	88.6%	
Mesh-MLP	mesh	—	—	88.8%	
EMNN+MC+Hier (ours)	mesh	26s	1.0GB	*88.7*%	

Table 4: Results on Human Body dataset (Segmetation).