A Method Details

To be clearer, in vcConv or vcTransConv layer, the weight coefficients $A_{i,j}$ are different for each edge between y_i and $\mathcal{N}(i)$, but B is shared for each local patch in one layer. Similarly, in vcPool or vcUnpool, the density coefficients $\rho_{i,j}$ are also different for each edge. Since the topology is fixed, their indices remain the same and their values are shared across the entire dataset. $A_{i,j}$, B and $\rho_{i,j}$ are all training parameters.

A.1 Network Implementation

In our implementation, we precompute the graph sampling process and record $\{\mathcal{N}(i)\}$ in a table of vertex connections. Each line i in the table contains the indices of the vertices in $\{\mathcal{N}(i)\}$ in the input graph. In practice, we store the table using a $N \times Max(E_i)$ integer tensor. The training parameters for convolution coefficients are stored in a tensor of size $N \times Max(E_i) \times M$ and the basis is a tensor of $M \times O \times I$. We use a $N \times Max(E_i)$ mask to mask out the vacant entries.

For training and testing, the whole network forward and backward processes are fully parallelized in GPU written in Pytorch. During testing, the kernels are pre-computed using equation 2 to accelerate the inference time.

A.2 Graph Sampling Algorithm

To select a subset of sampled vertices from the original graph with stride s, for each connected component, we start from a random vertex, mark it as selected and mark its $(s-1)$-ring neighbors as removed, and then traverse the vertices on the s-ring until finding a vertex that is not marked and has no (s-1)-ring neighbors marked as selected. Next, we mark this vertex as selected and start a new round of searching. We repeat those steps recurrently using a queue structure until all vertices are marked. One can also manually assign certain vertices to be included in the sampling set by marking them before starting the searching.

After collecting the sampled vertices, we create the input and output graphs for the up-sampling process and reverse them for the down-sampling process, and then create edges between the input and output graphs regarding the topology of the original graph under certain kernel radius size as illustrated in Figure 10.

A.3 Mistakes in Main Paper

We apologize for some mistakes in the paper and list them below. In the caption of Figure 9, “triangulated” should be removed. In Table 4.1, “SpiralCNN” should be Neural3DMM. SpiralCNN is the convolution layer used in Neural3DMM. In Line 178, “Section:vdPool” should be “Section 3.3”.

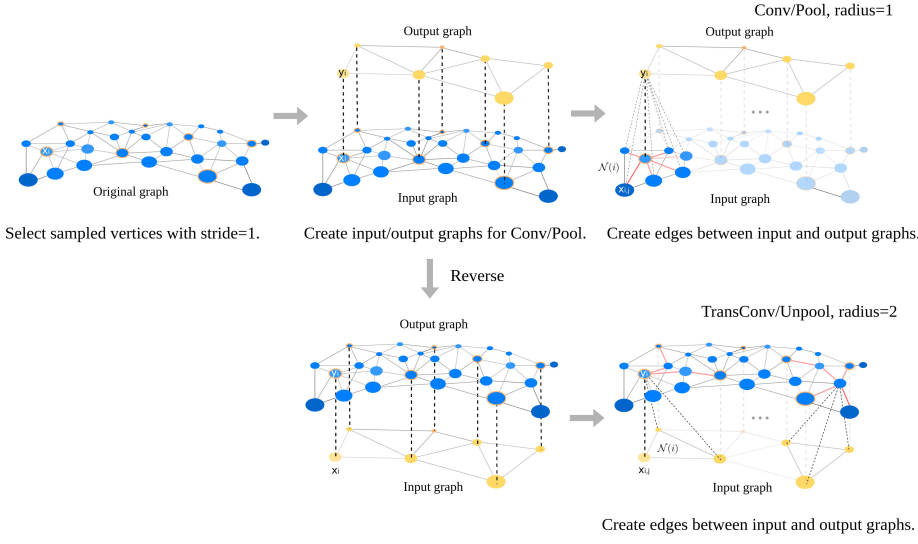

Fig. 10. Steps for graph up and down-sampling.

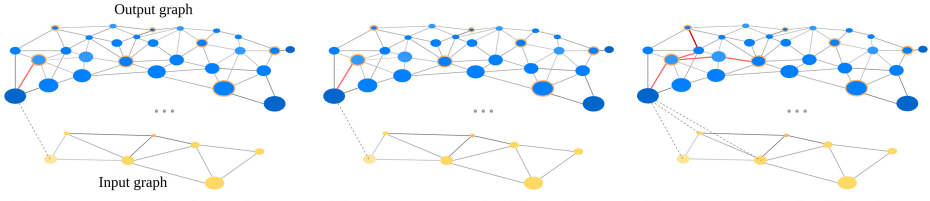

Fig. 11. More up-sampling examples.