Pre-Training Attention Module with Additional Supervision

Govind Joshi
CSCE 685, Dr. Shinjiro Sueda

April 24, 2025

1 Attention Mask Construction

1.1 Background

RigNet demonstrates that pre-training the per-vertex attention module via a cross-entropy loss against binary attention masks can boost joint-prediction accuracy. In this notebook we explore how to *construct* those masks.

1.2 Mask Construction Ideas

The key intuition is that vertices lying in the plane orthogonal to a bone at each joint should receive high attention. Two simple heuristics I contemplated:

- **Radius on plane:** For each joint j, pick one incident bone, compute its orthogonal plane, and mark all vertices within a fixed radius r of j on that plane.
- **Dynamic slack:** Find the mesh vertex $p_{\min}$ closest to j that lies exactly on the plane (distance d), then mark all vertices within $d + \varepsilon$ of j.

Both introduce a hyperparameter (r or ε) that must be tuned.

1.3 RigNet Implementation via Ray-Casting

The published code instead uses a ray-casting scheme:

1. Decimate mesh to $\sim$3000 vertices.

2. For each joint–bone pair, cast $K = 14$ rays from both joint endpoints in the plane orthogonal to the bone.

3. Perform triangle–ray intersection to collect hit points.

   - If fewer than 6 vertices are found, fall back to the 6 nearest neighbors of the joint.
   - If no intersections occur, call `nearby_faces()` for a guaranteed fallback.

4. Compute the 20th percentile of all hit distances, multiply by 2, and retain only those hit-points below this threshold.

5. Map retained hit-points back to their nearest mesh vertices to form a binary attention mask.

1.4 Current Progress

- Implemented `form_rays()`: computes two orthonormal directions per bone and samples $2K$ ray origins & directions in parallel.
- Integrated ray–triangle intersection via `trimesh.intersects_location`.
- Prepared to group, filter and threshold hit-points for mask generation.

2 JointNet Sanity Check

Additionally, i have now implemented the *entire* JointNet architecture as described in the paper:

- Three stacked GMEdgeConv layers (output channels 64, 256, 512),
- Separate displacement and attention heads (each its own GMEdgeNet with sigmoid for attention),
- Differentiable mean-shift clustering (with fixed bandwidth h for now),

However, on our tiny dataset (20 train / 5 validation examples), and without any attention supervision or trainable bandwidth h, we observe *minimal improvement* over the simpler baseline. This suggests:

- The dataset is too small to effectively train such a deep graph network end-to-end.

- Without pre-trained attention masks, the attention head learns poorly.
- Keeping h fixed prevents optimal clustering granularity.

3 Training Process Overview

1. **Attention pre-training:** cross-entropy on binary masks.
2. **Displacement module pre-training:** Chamfer loss between displaced and ground-truth joints.
3. **Fine-tuning:** jointly optimize attention, displacement, and clustering bandwidth.

4 Next Steps

- Finish attention-mask construction code using the ray-casting rules above.
- Pre-train the attention network against these masks.
- Train the vertex displacement module under Chamfer supervision.
- Make the bandwidth h trainable during mean-shift clustering.
- Jointly fine-tune all parameters and evaluate BoneNet and RootNet connectivity.