CS230

3dBARC (3D Bipedal Anatomical Region Classifier)

Ethan C. Robinson
Undeclared
Stanford University
etchro@stanford.edu

Abstract

In the 3D Creative pipeline, there are many tasks that benefit from having an accurate way to anatomically classify the vertices of a mesh. Specifically for Bipeds in the animation and game pipelines, an accurate classification of anatomical regions would be a helpful jumping off point for a number of steps of the creative process. Although there are a number of auto-rigging solutions, including online ones like Mixamo and offline ones like Blender's Rigify, they are either end-end (meaning it is difficult to easily tweak tolerances and structures/manipulate the result), or done via joint approximation by vertex density, instead of solutions that rely on deeper anatomical understandings. Therefore, 3dBARC aims to accurately classify vertices into anatomical regions in order to serve as a common and open baseline for animation tools, skeletal generation, and even voxel weighting.

1 Introduction

The 3D Creative pipeline has a few bottlenecks. Finished models must be re-topologized. Re-topologized models must be UV unwrapped. Final models must be rigged, etc. Although each step in the process is one that can be mastered in an artistic way, they can greatly slow the process, and for small teams attempting to tackle large projects, this can prove challenging. 3dBARC aims not simply to classify the vertices of Bipedal meshes into anatomical regions, but to model an open solution tailored to a specific bottleneck that would benefit the industry. 3dBARC aims to be a lean model, with "anatomical understanding" inherent, and therefore takes only the vertices of a mesh as input to the neural network. As output, 3dBARC returns an array of integers corresponding to a predicted anatomical region (17 predefined regions; see Figure 1c).

2 PointNet's influence

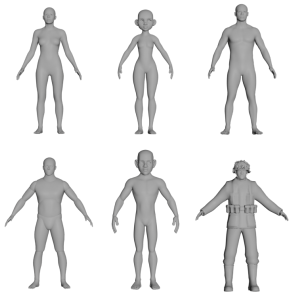
3dBARC was heavily inspired by PointNet [3], which proved a neural network can directly process a point cloud and perform segmentation. Accordingly, the architecture is quite similar to PointNet, but differs in that 3dBARC does not include the global pooling and T-Net. The global pooling is something I aim to add before the final paper, and I will experiment with T-Net to see if it benefits the model.

3 Dataset and Features

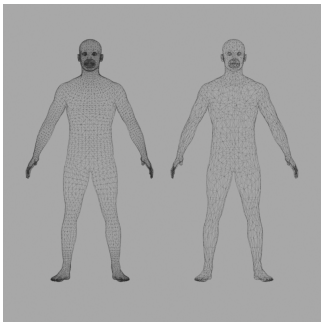
Presently, the dataset consists of 300 point clouds (5 from Blender [1] and 1 from Sketchfab [2]), all of which originate from CC0 models. In order to generate variations of each base CC0 model, an algorithm randomly decimates each model to alter its topology. Repeating this process N times

CS230: Deep Learning, Autumn 2025, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

yields N + 1 point clouds derived from the same base shape. For this implementation, I chose N = 49, which results in the dataset of 300 point clouds.



(a) CC0 Base Meshes



(b) Original (left) & Augmented Copy (right)

ID	Region	Description	Color
1	pelvis	Pelvic region	
2	spine	Lower spine	
3	spine_upper	Upper spine (spine1-3)	
4	neck	Neck	
5	head	Head	
6	arm_l	Left upper arm	
7	forearm_l	Left forearm	
8	hand_l	Left hand	
9	arm_r	Right upper arm	
10	forearm_r	Right forearm	
11	hand_r	Right hand	
12	thigh_l	Left thigh	
13	calf_l	Left calf	
14	foot_l	Left foot	
15	thigh_r	Right thigh	
16	calf_r	Right calf	
17	foot_r	Right foot	

(c) Pre-defined anatomical regions and their colors

Figure 1: Dataset

Each point cloud in the dataset is rigged with group names that correspond to the 17 regions the model can classify, and upon converting a point cloud to training data, each vertex is assigned to the region corresponding to the most influential vertex group. The dataset has a mean of 9,817.75 and a median of 10,237.50 vertices. Right prior to training, the dataset undergoes even more augmentation, including duplication, slight noise, and small rotations. Augmentation also increases the training data by a factor of 3. As such, with an 80/20 split, this first model was trained on 720 point clouds, and 60 were reserved for validation. It was tested on 5 models chosen to be diverse from Mixamo.

4 Methods

As part of version 1, all training data is padded to match the size of the largest point cloud. As such, a custom loss function (and accuracy functions, though those are less important) had to be defined. 3dBARC uses cross-entropy loss, therefore the loss function implements that while ignoring padding ($\hat{y} = -1$) :

$$L = \frac{\sum_t \mathbf{1}[y_t \neq -1] \left(-\log \hat{p}_{t,y_t} \right)}{\sum_t \mathbf{1}[y_t \neq -1]}$$

Future versions will likely do this padding per-batch to make the model more efficient.

5 Architecture

The architecture is a fairly simple per-point multi-layer perceptron that processes vertices through dense layers (with batch normalization) ending in a softmax for the final classification. It will lend itself to increased capacity once the dataset is more diverse (and larger).

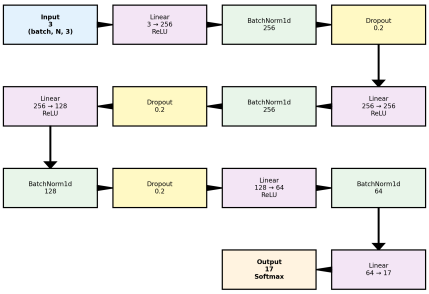


Figure 2: Architecture

6 Results

For the first version, the results aren't terrible. The model trained well, with both training and validation steadily improving. It plateaued around 80% accuracy, which is a number I aim to improve.

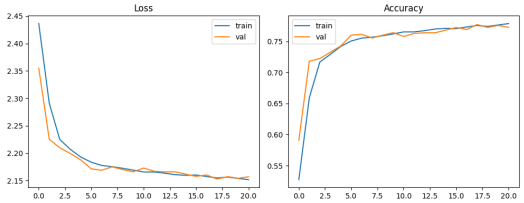
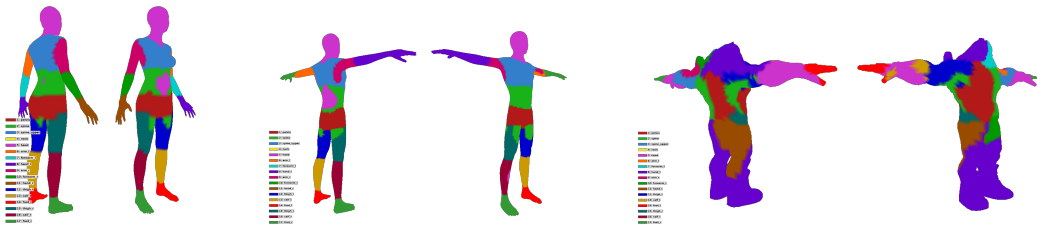


Figure 3: Training

Upon inference, 3dBARC yields mixed results. On one of the base meshes (a), it does relatively well, 80% (number of vertices classified correctly) seems like it might be an accurate figure. For a mesh that is somewhat similar in shape to a majority of the training set (b), the model performs decently. For a mesh completely novel in shape and form (c), the model performs quite poorly. This is likely due to the small difference in distribution among the training data.



(a) Trained model (b) Good result (test set) (c) Worse result (test set)

Figure 4: Inferences

The next steps are to increase my base model count. Increasing from 6 to 25+ base models, with much more diverse shapes and figures will allow the model to learn from a wider distribution. After preparing all of these additional models, I will add global context to the model. Giving vertices global context with a global pool will also likely reduce "splotches" seen clearly in Figure 4. Next, I will test T-Net to attempt and achieve translation and rotation invariance, and see if it helps in this case. With any extra time, assuming the results continue along the current trajectory, the focus will likely be on diversifying the training data.

References

[1] Blender Foundation. Blender demo files (cc0 assets). https://www.blender.org/download/demo-files/, 2024. Accessed: 2025-01-23.

[2] britdawgmasterfunk. WW2 Panzergrenadier 2k (CC0). https://skfb.ly/oAGuZ, 2023. Licensed under Creative Commons Attribution 4.0 International (CC BY 4.0).

[3] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017.