3D Point Cloud Classification with Deep Learning

Mentees: Troy Eggertson, Sam Doss-Hammel, Erika McPhillips, Mentor: Christian Bueno

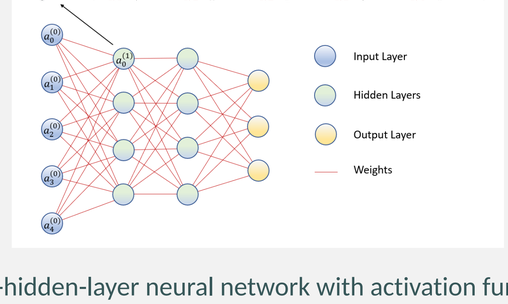
UCSB's Mathematics Directed Reading Program

Introduction

Point clouds are sets of points that arise in many situations such as LiDAR data and samplings of 3D meshes. Traditional neural networks struggle with point clouds because their output depends on the order of the input, and sets with n points have $n!$ permutations to worry about. PointNet [1] solves this problem directly by being permutation invariant by design.

Feed-Forward Neural Networks

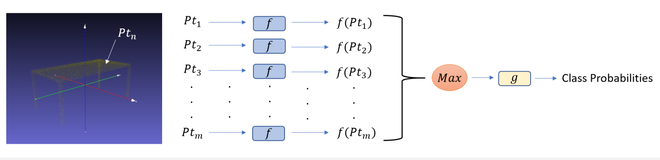
$$a_0^{(1)} = \sigma(w_{0,0}a_0^{(0)} + w_{0,1}a_1^{(0)} + w_{0,2}a_2^{(0)} + w_{0,3}a_3^{(0)} + w_{0,4}a_4^{(0)} + b)$$



Above is a 2-hidden-layer neural network with activation function σ. In general, a feed-forward neural network applies an alternating sequence of linear transformations L_i and nonlinear transformations σ_i to an input x:

$$f(x) = \sigma_n(L_n(\ldots \sigma_2(L_2(\sigma_1(L_1(x))))\ldots))$$

PointNet



$$F_{PN}(A) = g(\max_{a \in A} f(a))$$

Above, f and g are traditional neural networks and F_{PN} is a PointNet. By taking the component-wise maximum (a.k.a. max-pooling) of transformed elements of the set, PointNet achieves permutation invariance.

Theoretical Properties of PointNet

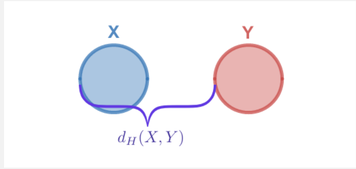
- **PointNet is continuous with respect to the Hausdorff distance.** The Hausdorff distance d_H is a metric that measures how different non-empty compact subsets are. For a metric space (Ω, d), it's given by:

$$d_H(X, Y) = \inf\{\epsilon \geq 0 : X \subseteq Y^\epsilon \text{ and } Y \subseteq X^\epsilon\}$$

where X^ϵ is ϵ-fattening of X, i.e., the set of all points within ϵ of X.

Theoretical Properties (Continued)

This can be visualized in $(\mathbb{R}^2, d_{Euclidean})$ as:



- **Universal Approximation.** For any $\epsilon > 0$ and uniformly d_H-continuous function F, there's a PointNet F_{PN} so that $|F(A) - F_{PN}(A)| < \epsilon$ for all point clouds A in the unit ball.

Training in General

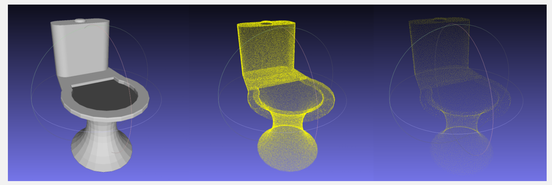
Given some data $D = \{(x_i, y_i)\}_{i=1}^n$ & parametric model f_w with weights w.

- **Loss Function / Objective Function:** For the machine to know how far predictions are from actual observations we use a loss function $\ell(y, \hat{y})$, where $\hat{y}$ is the predicted value and y is the ground-truth.
- **Gradient Descent (GD):** For a given loss function, GD is an optimization algorithm with the goal of $\min_w \frac{1}{n}\sum_{i=1}^n \ell(y_i, f_w(x_i))$. GD updates the weights by $w^{i+1} = w^i - \alpha\nabla_w \frac{1}{n}\sum_{i=1}^n \ell(y_i, f_w(x_i))$ where $\alpha > 0$ is the so-called learning rate.
- **Stochastic Gradient Descent (SGD):** Similar to GD except the whole-data gradient is estimated using gradients over smaller batches of data. SGD has a reduced computation time per iteration and smaller memory cost than GD. Many epochs (i.e. number of times the algorithm sees the whole dataset) may be needed.
- **Backpropagation:** An algorithm which efficiently computes the gradient $\nabla_w(\cdot)$ with respect to network weights via careful use of the chain rule, iterating backwards from the final layer to 'unpack' one layer at a time, thus avoiding calculating anything twice. [2]

Training a Classifier

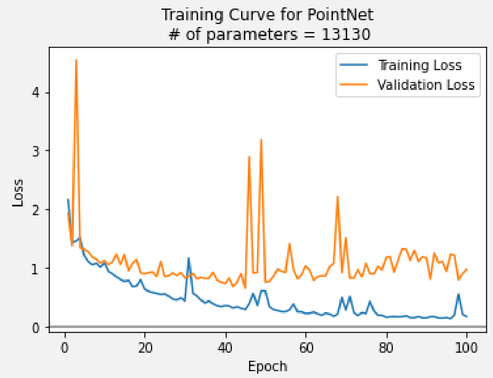
- **Softmax:** A function for transforming $x = (x_1, \ldots, x_m) \in \mathbb{R}^m$ into a probability distribution by $s(x)_i = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}$. Classification models output the class with the largest associated probability, e.g. 77% toilet.
- **Information Entropy:** Given by $H(p) = -\sum_i p_i \log p_i$ for a probability distribution p. It's deeply related to data compression.
- **Cross-Entropy Loss:** The distance, in bits of information, separating two probability distributions. Cross entropy loss can be mathematically defined as $H(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_i \mathbf{y_i} \log_{\mathbf{e}}(\hat{\mathbf{y}}_\mathbf{i})$, where $\hat{\mathbf{y}}$ and $\mathbf{y}$ are as defined above. Often used in classification problems, including our experiment.

Sampling Meshes



For this project we use the ModelNet10 data set, 3D meshes broken into 10 classes (bed, chair, desk, etc.). We use Barycentric coordinates to sample 1,024 random points from surface of the mesh. These form the point clouds our model takes as input. Some meshes are sampled multiple times to ensure a balanced data set. After sampling the 3D meshes, we have 1,000 point clouds per class.

Our Results



Class:	Bathtub	Bed	Chair	Desk	Dresser	Monitor	Night Stand	Sofa	Table	Toilet
Accuracy(%):	52	94	94	78	81	98	85	93	70	93

We implement PointNet with PyTorch. We tried SGD but found the Adam optimizer better for training and used it instead. We train using: Epochs $= 100$, Batch Size $= 32$, Learning Rate $= 10^{-3}$. We chose the sub-networks of PointNet to be of type $f : \mathbb{R}^3 \to \mathbb{R}^{32} \to \mathbb{R}^{32}$ and $g : \mathbb{R}^{32} \to \mathbb{R}^{32} \to \mathbb{R}^{10}$ with Softmax at the end. Our model achieves an accuracy of 80.2% on ModelNet10 (random guessing would give 10%).

References

[1] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[2] Justing Domke. Automatic differentiation and neural networks, 2011.

[3] Eman Ahmed, Alexandre Saint, Abd El Rahman Shabayek, Kseniya Cherenkova, Rig Das, Gleb Gusev, Djamila Aouada, and Bjorn Ottersten. A survey on deep learning advances on different 3d data representations. *arXiv preprint arXiv:1808.01462*, 2018.