Introduction

Concept: convert grayscale front + side 2D character sketch into 3D model using deep learning

→ Motivation: speed up asset generation for video games or animation
→ Prior Work: mostly single image reconstruction [2-4], whereas here, the hope is that 2 views will be more generalizable

3D Model Format

3D Models represented by point cloud set of \((x, y, z)\)

→ Typical 3D models in mesh format (edges and vertices)
→ But mesh requires a graph loss (difficult to define)
→ Point cloud loss is simpler

Define: Chamfer distance (CD) loss between two point sets \(S, \tilde{S} \in \mathbb{R}^3\)

\[
CD(S, \tilde{S}) = \sum_{x \in S} \min_{y \in \tilde{S}} ||x - y||^2 + \sum_{y \in \tilde{S}} \min_{x \in S} ||x - y||^2
\]

→ For \(S, \tilde{S}\) CD calculates distance to nearest neighbor point in other set

Network Architecture

Encoder & Decoder Architectures

Encoder: Modified Matrix Capsule Network (CapsNet)

\[ A: \# \text{conv. channels} \]
\[ B: \# \text{capsules} \]
\[ C, D: \# \text{capsules} \]
\[ K: \text{conv. kernel size} \]

CapsNet like a convolutional network except:

→ activations are matrices ("capsules") instead of scalars
→ non-linearity between layers from a trained dynamic routing
→ network figures out how to route activation outputs to next layer

Routing by "Expectation-Maximization Routing" (EM Routing) [1]:

- \(\text{EM Iteratively} \) fits gaussian to feature activations
- Routes low-level features to be part of a higher level feature

Decoder: Upconvolution Network

Decoder series of \(\text{upconv} \rightarrow \text{conv}\) layers

- upconv is a "transpose" of a \(\text{conv} \rightarrow \text{max pool}\) encoding layer
- maps lower dimensional features to higher dimensional output
- finish with linear fully connected layer to scale or spread output

Results

2D Front & 2D Side & 3D Input & 3D Generate & 3D Overlay

- Cost saturates, need more hyperparameter tuning and training examples
- Dev predicted output is overfitting to lizards in training set
- Predicted points tend to cluster near \((0,0,0)\), decoder needs to be tuned to give better spread over volume

General Conclusions

→ 2D \(\rightarrow\) 3D point cloud feasible, but not very good
→ Unordered output data bad for generation
→ Scales poorly with number of output points
→ Still need to do 3D mesh reconstruction (annoying)
→ Future: End-to-end 2D \(\rightarrow\) mesh approaches more desirable [2]

References


Acknowledgements: Xingyu Liu for TA project mentorship, Russell Kaplan for AWS GPU credits & advice, Andrew Ng for teaching class.