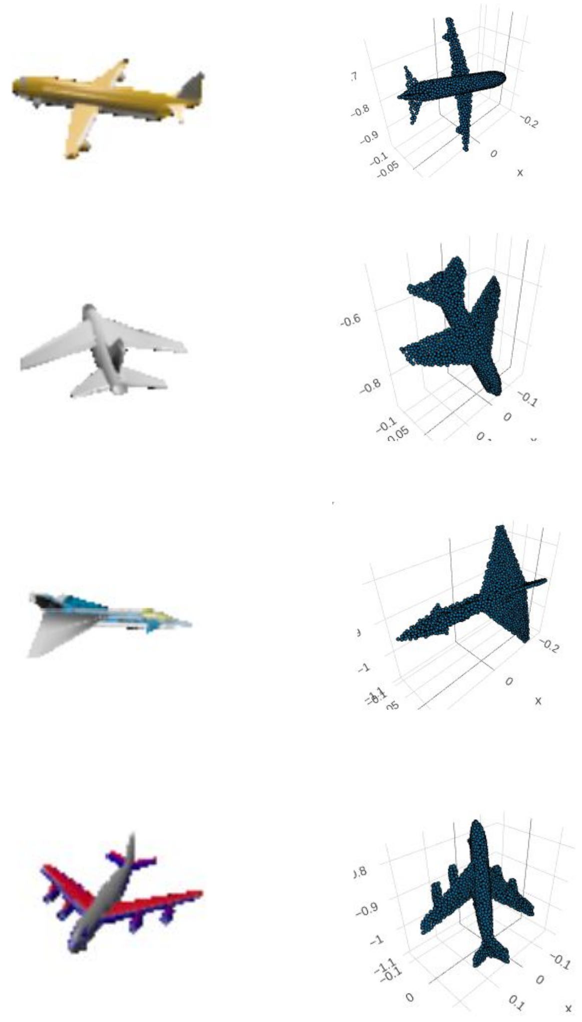
From Pixel to Mesh

Project of RecVis18 - M2 MVA

Tong ZHAO

Oscar CLIVIO



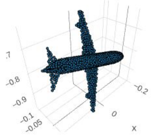
Content

Found around the house!

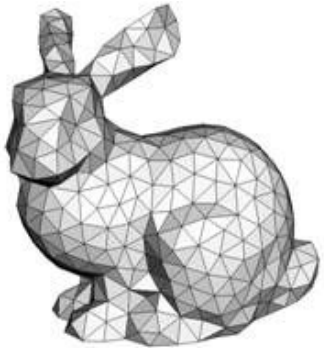
- Project Goal
- Pixel2Mesh
- Experiment
- Conclusion

Project Goals

Project Goals



- **Topic** : from a single RGB view of a 3D object, reconstruct its 3D representation.
- **Preferred 3D representation** : meshes —— easier to deform, model shape details.



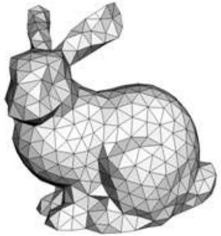
The model we used: **Pixel2Mesh** introduced in *Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images*, Wang et al, EECV, 2018

Our goals :

- Entirely reimplement the model in PyTorch √
- Reproduce its results on 1 class ×
- Test different regularization methods and play with different parameters √
- Improve net structure √

Pixel2Mesh

Mesh representation and main modules



- Mesh = **Graph** : vertices, edges, and features for each vertex.

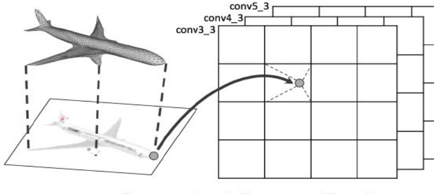
Main ideas :

- Progressively deform an original mesh (eg ellipsoid)
- Introduce more and more points in the mesh

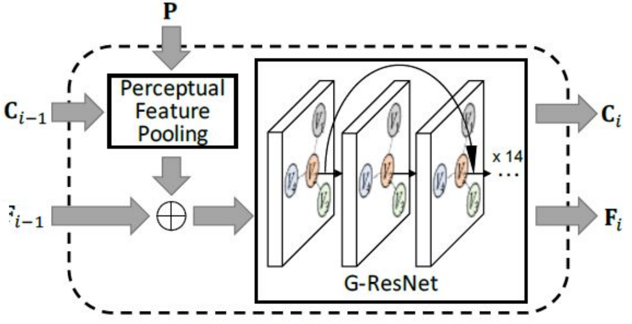
$$f_p^{l+1} = w_0 f_p^l + \sum_{q \in \mathcal{N}(p)} w_1 f_q^l$$

Modules:

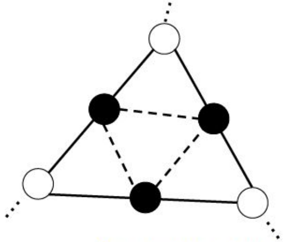
- **Graph Convolutional Networks** (GCN).
- **Perceptual feature pooling Layer** :
- **Mesh Deformation Block** : 14 GCNs with shortcut connections (G-ResNet)
- **Graph Unpooling Layer** : create vertices and edges.



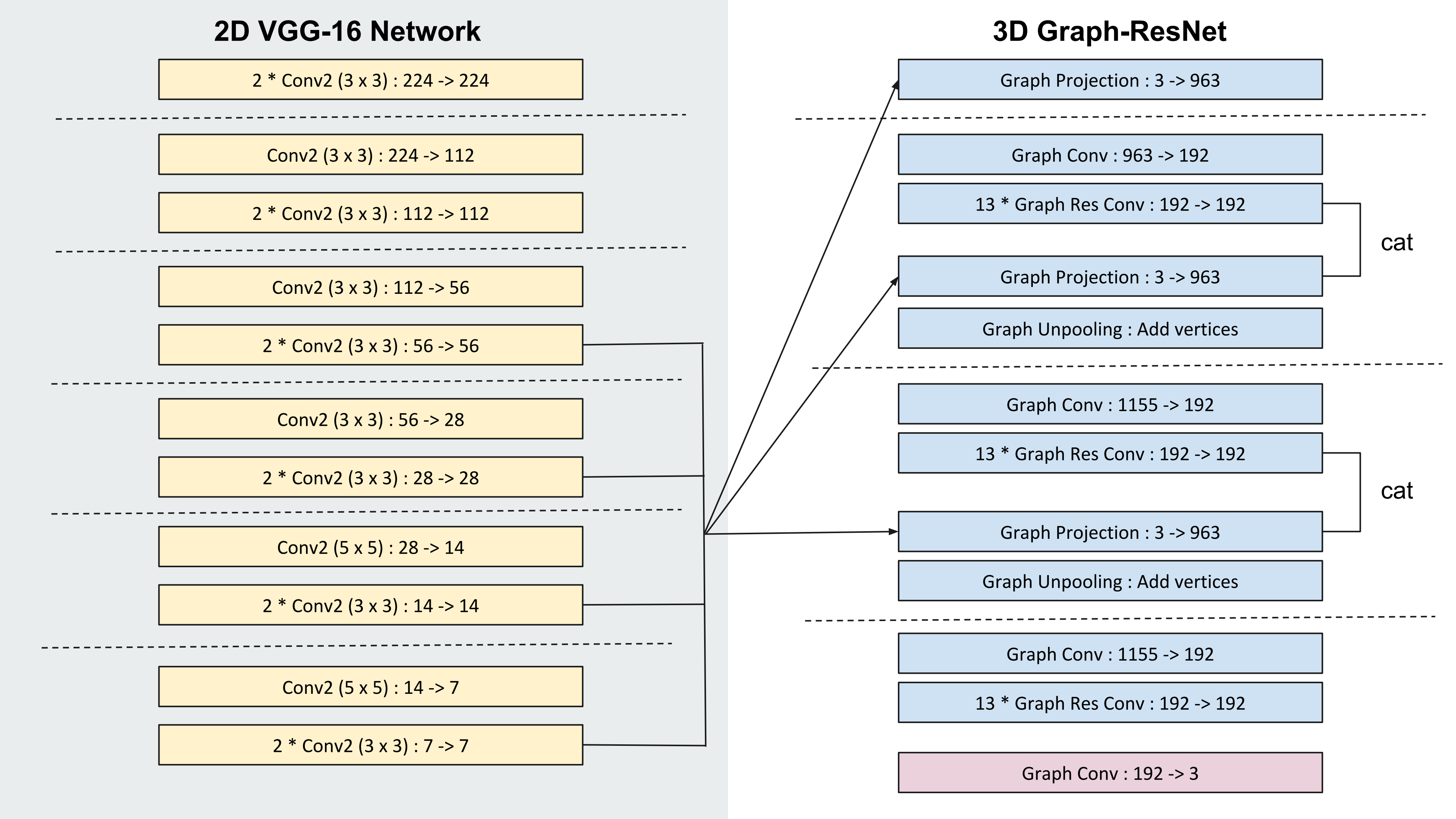
Perceptual Feature Pooling



Mesh Deformation Block

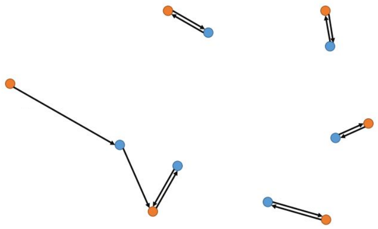


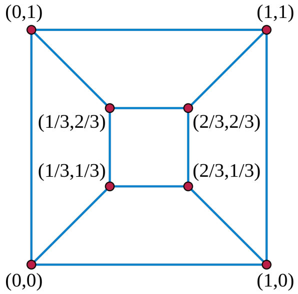
Graph Unpooling

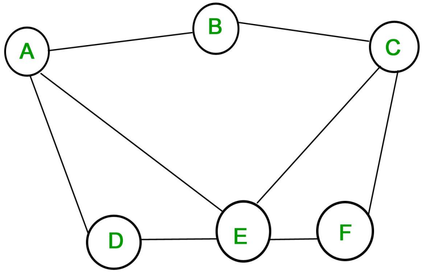

2D VGG-16 Network
2 * Conv2 (3 x 3) : 224 -> 224
Conv2 (3 x 3) : 224 -> 112
2 * Conv2 (3 x 3) : 112 -> 112
Conv2 (3 x 3) : 112 -> 56
2 * Conv2 (3 x 3) : 56 -> 56
Conv2 (3 x 3) : 56 -> 28
2 * Conv2 (3 x 3) : 28 -> 28
Conv2 (5 x 5) : 28 -> 14
2 * Conv2 (3 x 3) : 14 -> 14
Conv2 (5 x 5) : 14 -> 7
2 * Conv2 (3 x 3) : 7 -> 7
3D Graph-ResNet
Graph Projection : 3 -> 963
Graph Conv : 963 -> 192
13 * Graph Res Conv : 192 -> 192
cat
Graph Projection : 3 -> 963
Graph Unpooling : Add vertices
Graph Conv : 1155 -> 192
13 * Graph Res Conv : 192 -> 192
cat
Graph Projection : 3 -> 963
Graph Unpooling : Add vertices
Graph Conv : 1155 -> 192
13 * Graph Res Conv : 192 -> 192
Graph Conv : 192 -> 3

Loss Design

- **Chamfer Loss**: Measure the **discrepancy** between two point clouds. It does not guarantee any connectivity of the graph.
- **Tutte Laplacian Loss**: Prevent the vertices from moving too freely after a deformation block.
- **Edge Length Loss**: Penalize long edges in the mesh by minimizing its L2 norm.





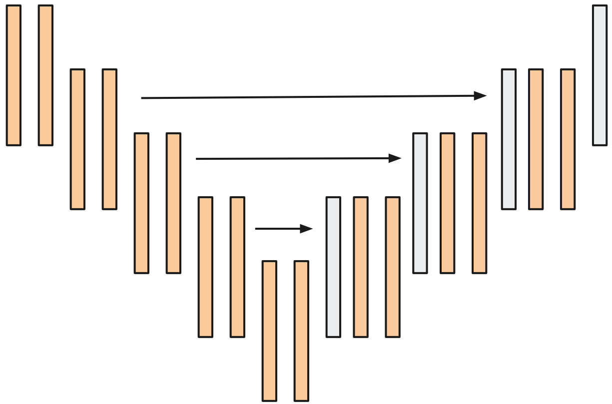


No loss has direct relation with 2D VGG net!

AutoEncoder-based 2D Net

Motivation: A warm start for 2D image feature, provide strong features to the 3D GNN.

Structure: All-convolutional U-net



Loss :

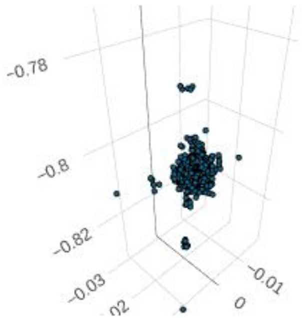
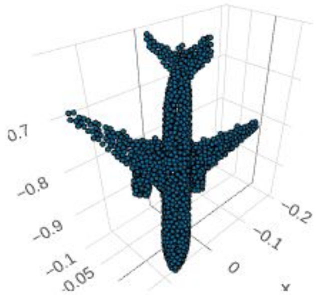
+ Binary cross entropy - approach the distribution
+ L1 loss - avoid blurry reconstructed image

Strategy:

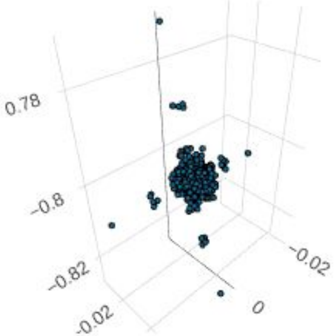
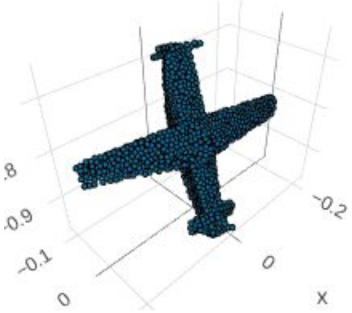
+ first epoch, we minimize the image loss and the mesh loss
+ After then, we minimize only the mesh loss.

Experiments

An Example

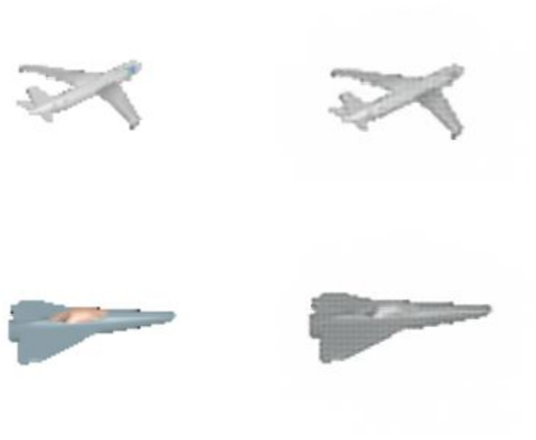
Input 2D Image	Output 3D Point Cloud	Output 2D Image	3D GroundTruth
			

An Example

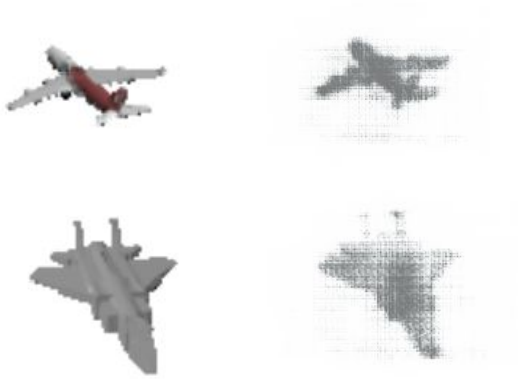
Input 2D Image	Output 3D Point Cloud	Output 2D Image	3D GroundTruth
			

Detail 1 - Influence of 3D GCN on 2D CNN

The auto-encoder can reconstruct a high-quality image in a few (~2000) iterations. But the 3D GCN leads the network to focus on the general shape instead of details.



With Image Loss

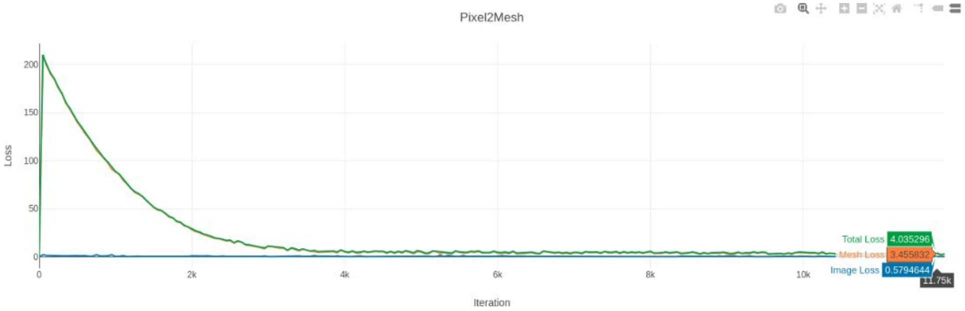


Without Image Loss

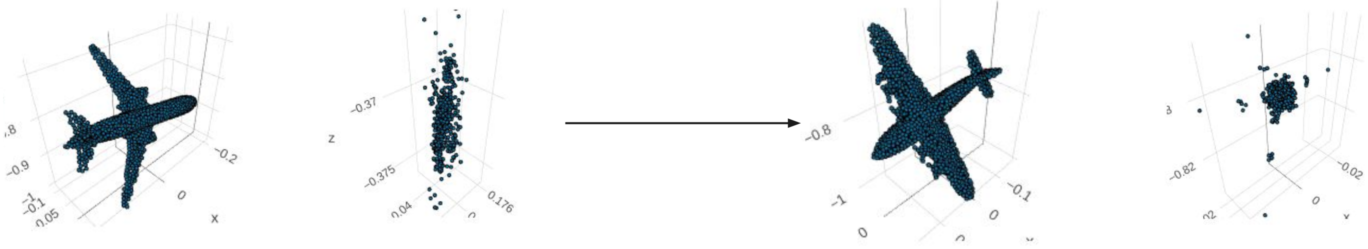
Detail 2 - Where is the weird result from?

√ **The network structure**: the output shape in each layer is consistent.

√ **Image loss**: the reconstructed image is satisfactory



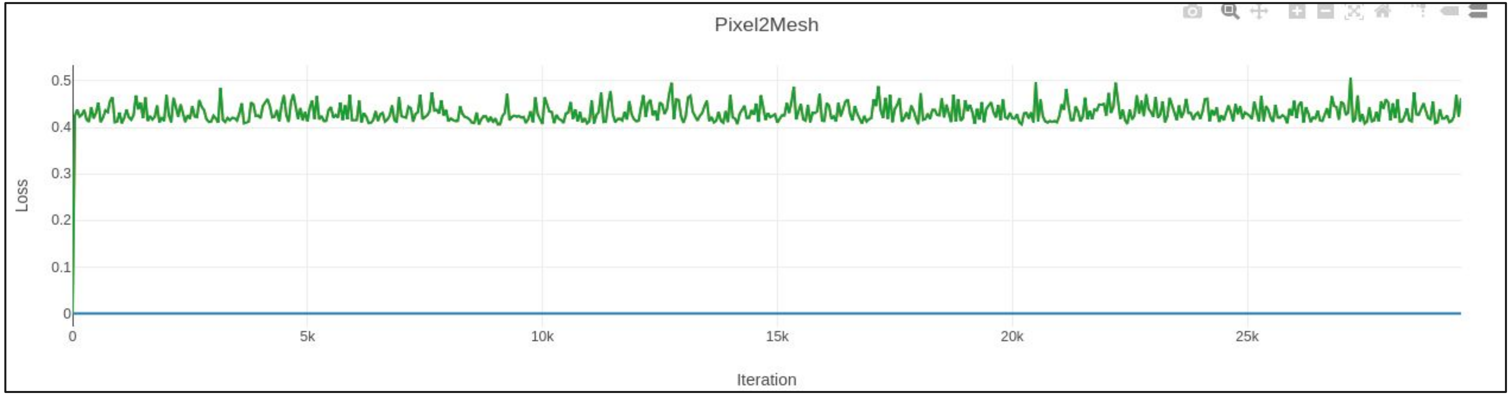
√ **Chamfer loss**: the network learned through the training process the camera intrinsic matrix



Detail 2 - Where is the weird result from?

× **Laplacian loss** & **Edge length loss**: function 'index_select' is used in both functions. Its gradient goes wrong when there are repeated indices.

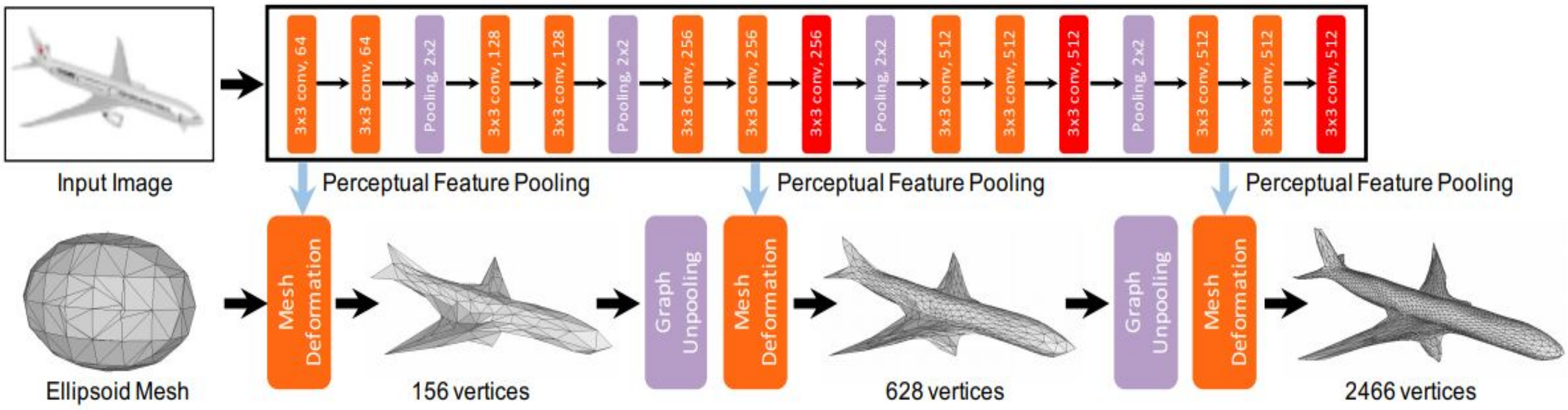
https://github.com/pytorch/pytorch/issues/3644 & https://github.com/pytorch/pytorch/issues/5159



× **Batch size is 1**: the graph projection layer only works when batch size is 1, which makes the network easily get stuck in a local minimum.

Detail 2 - Where is the weird result from?

- **Overuse of 2D image features:** all 3 graph projection layers use the same image features.



Conclusion

Conclusion

- Pixel2Mesh is a complicated network based on 2D CNN and 3D GCN.
- Additional image reconstruction task accelerates the convergence of the model
- Further tests need to be done to reproduce its performance on TensorFlow

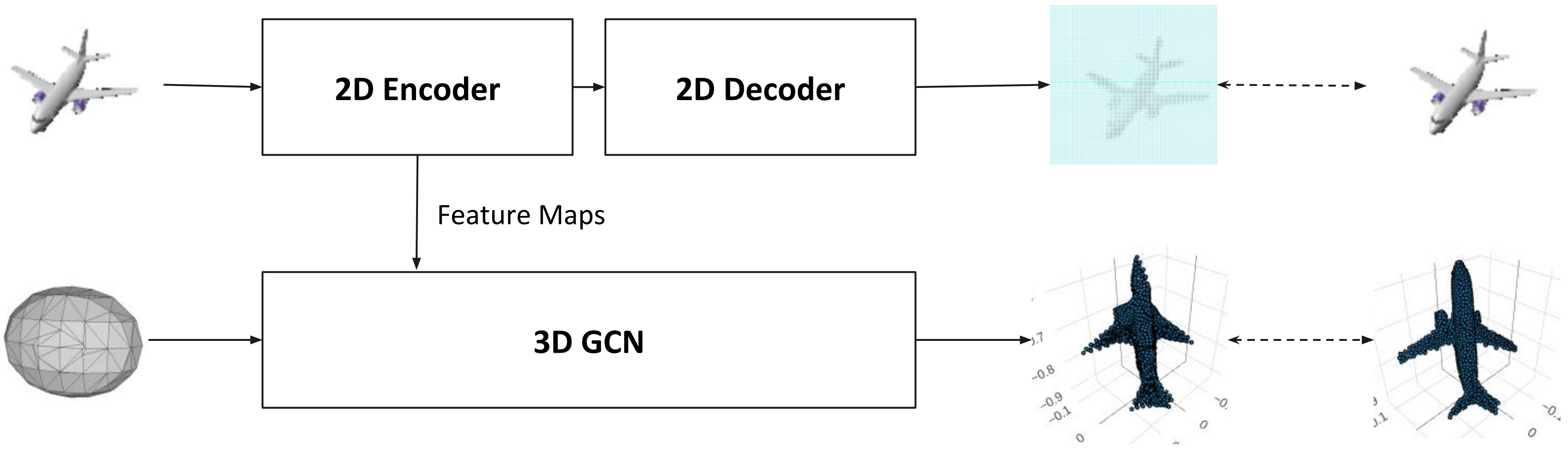
Code available at: https://github.com/Tong-ZHAO/Pixel2Mesh-Pytorch

References

[1] Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W. and Jiang, Y.G., 2018. Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images. *arXiv preprint arXiv:1804.01654*.

[2] Groueix, T., Fisher, M., Kim, V.G., Russell, B.C. and Aubry, M., 2018. AtlasNet: A Papier-M\^ ach\'e Approach to Learning 3D Surface Generation. *arXiv preprint arXiv:1802.05384*.

[3] Kipf, T.N. and Welling, M., 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.


2D Encoder
2D Decoder
Feature Maps
3D GCN