Pytorch DISN

Hee Hwang, Edward Schneeweiss, Catherine Huang
University of Massachusetts Amherst
@cs.umass.edu

Abstract

The task of reconstructing 3D shapes from single-view images is a long-standing research problem and for good reason, as there are many applications like robotics, mapping, modeling, etc. The Paper DISN: Deep Implicit Surface Network for High-quality Single-view 3D Reconstruction, came to the field in 2019 and outperformed the state of the art (SOTA) at the time due to some key changes. First they utilized signed distance fields as it was shown that a neural network can very effectively model a signed distance field. This allowed them to densely sample points in key areas without blowing out the memory, which can happen when using explicit 3D representations like voxels, point clouds or meshes. Secondly they allow their model to attend to the specific region of the image where a sampled point is located, which greatly improved their models ability to capture complex high detailed structure, better than a global descriptor would allow them to. For our project we replicated their results, and experimented with some model changes.

1. Introduction

For our project we re-implemented the Paper DISN: Deep Implicit Surface Network for High-quality Single-view 3D Reconstruction, we replicated this paper because it was one of the early papers to use a signed distance field (SDF) to represent the 3D shape of an object, and the novel approach from this paper produced results that surpassed the SOTA at the time. The Challenge is to take in an RGB image and output a SDF that represents the shape of the object.

Up to when this paper was published many single-view 3D Reconstruction methods have been proposed where deep learning based methods have achieved the best results. To represent the 3D structure many of these methods utilize either voxels [11], point clouds [2], meshes, or Occtrees [5], because it is easy to represent them with a neural network. However these representations are limited in terms of resolution and their reconstruction loss. The resolution is an obvious problem in that to get better accuracy you must increase the resolution which in turn demands more memory, so the accuracy of the model is limited by memory constraints. Additionally the losses for these models are indirect since they will use Chamfer Distance (CD) and Earth Mover Distance (EMD), which only approximately measure shape similarity. To address the limitations of voxels, point clouds, and meshes, the authors of DISN use a SDF to represent the 3D structure implicitly. Where the neural network is a function that takes in a 3D point and outputs the SDF value at that point. This solves both problems, firstly since the 3D shape is expressed explicitly the resolution is infinite, and during training we can specifically sample points near the surface of the object avoiding wasted computation in empty space. Secondly the loss is direct, in that we generate the SDF for a the ground truth mesh then take the L1 loss between the ground truth SDF value and the models predicted SDF value.

While many single-view 3D reconstruction methods that learn a shape embedding from a 2D image are able to capture the global shape properties, they have a tendency to ignore details such as holes or thin structures. These fine-grained details only occupy a small portion of the 3D structure and thus sacrificing them does not incur a high loss. To address this problem the authors of DISN introduce a local feature extraction module, specifically they estimate the viewpoint pose and can then project the sampled 3D points onto the input image and extract the local features to predict the SDF. This enables the model to learn the relationship between projected pixels and 3D space, which greatly improved the fine details of the reconstruction.

Our contributions:

- Their model was implemented in TensorFlow and to rectify this we re-implemented it in PyTorch (the Superior deep learning library).
- We also explored some modifications such as: global features only and local features only. To show the impact of combining local and global features for reconstruction.

2. Related Works

The problem of reconstructing 3D structure from a 2D image is a popular challenge and has had many competing approaches such as 3DN [9], Pix2Mesh [8], AtlasNet [3], and OccNet [5]. All of these approaches to 3D reconstruction use explicit 3D representations and because of this suffer from problems such as limited resolution and fixed mesh topology. Implicit representations provide an alternative representation to overcome these limitations. It was shown in DeepSDF [6] that a neural network SDF can effectively represent a 3D shape, so the authors of DISN used a signed distance field to represent the 3d shape implicitly. Additionally models before DISN only used a global feature representation that is they encode the image to a single feature then decode that vector to build the 3D shape. The authors of DISN proposed the use of local features in combination with global features to allow their model to focus on the local structure.

3. Method and Architecture

Since we're re-implementing DISN [10] in PyTorch, we follow similar steps in setting up our neural network to perform feature extraction and SDF prediction.

3.1. DISN

The overall architecture of the network is shown in Figure 1. We take in sample points and an RGB image to perform feature extraction. After using the encoders: ResNet [4] for the image, and PointNet [7] for the points, we concatenate the features, then put it through the decoder predicting the SDF values. After obtaining the predicted SDF values from the model, we use marching cube to reconstruct the 3D mesh.

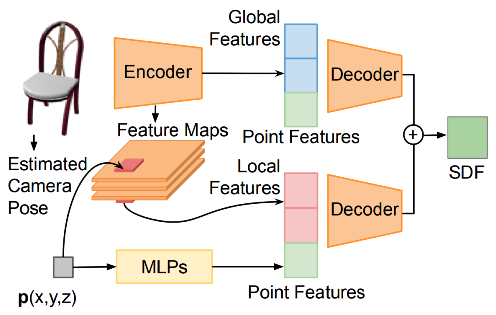


Figure 1. Given a point, an image and estimated camera pose, we project the point onto the image plane. DISN uses local features at the projected position, the global features, and the point features to predict the SDF of p. 'MLP' denotes multi-layer perceptrons.[10]

3.2. Data

The dataset to train our model is based on ShapeNet [1], but has been modified since ShapeNet only contains meshes while for training our model we need an SDFs, images, and poses. The SDFs are created from the ground truth mesh, and for the dataset we pre-sample 2,048 points near the surface of the object so that sampling the ground truth SDF is not a bottleneck during training. Then to get the images, which are the input to the model, we render 24 images of each mesh in ShapeNet from random camera poses and record the poses. This way for each training sample we have an image, the camera pose, the point samples and the ground truth SDF values. These are all the components needed to train.

3.3. Feature Extraction

Before estimating the SDF values for 3D reconstruction, the network needs to extract image features and point features. We utilize a pre-trained ResNet18 model to extract image feature from the image instead of the VGG-16 network used in the original DISN[10].

We picked ResNet instead of VGG because of its higher accuracy with less parameters and flops through layer design and the use of skip connections in their residual architecture. Due to time constrain, we focused only on one category, the chair category, from ShapeNet[1], instead of the 13 categories done in the original paper[10]. We loaded a pre-trained ResNet model and fine tuned it during our training procedure.

From each image batch, we extracted both global and local features, where the global features provide the general shape of the mesh and local features provide information on the detailed parts, like the arm and legs of a chair. The global features are from the last feature layer, the average pooling layer of ResNet, while the local features are taken from the output of every residual block, so different levels of detail can be obtained. To extract the local features for a sampled point, we project the point onto the image using the camera intrinsic and extrinsic parameters (i.e. focal length, pose). The projected point is used to select the local feature at each layer of the ResNet, then concatenated together to create a hyper feature. Then for every point we run it through a MLP to extract point features.

Finally the global and local features in combination with the point features go into their respective MLP decoders. The output of the two decoders is added together to produce the final SDF prediction.

In addition to the above setup that closely follows [10]'s architecture. To better understand the effectiveness and differences of including local features, we had set up pipelines to train three different models using our network. We trained a model with only global features, only local features, and of course a model with both local and global features. We trained each model 100 epochs using GeForce RTX 2080 Ti. For 100 epochs it takes roughly 3 days of training per model. At the time of this paper, we're only

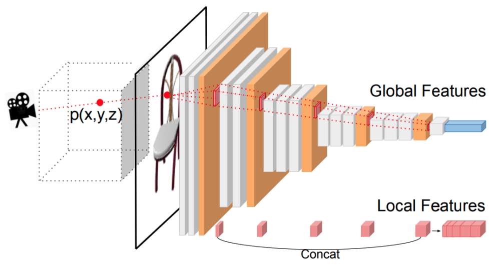


Figure 2. Local feature extraction. Given a 3D point p, we uses the estimated camera pose to project p onto the image plane. Then we get the projected locations of p on each local feature map layer. We concatenate the features at each layer to get the local features of p. [10]

able to train the local feature only model through 75 epochs after two days of training.

4. Experiment

graphicx egbib.bib

1. Implementation Details
   After making an SDF prediction using 257^3 grid points, we applied a marching cube algorithm based on these SDF values, normalize and center the generated mesh. At this point, there are small fragments that are not part of the chair. We cleaned the fragments and re-normalized the object, just like the original paper[10].

2. Evaluation Metrics
   According to the original DISN paper[10], we defined the following three metrics for measuring the performance: (1) Chamfer Distance, (2) Earth Mover's Distance, and (3) F-1 Score. Here, PC and PC_T are the sampled point clouds from the predicted and ground-truth mesh.

   (a) Chamfer Distance

$$CD(PC, PC_T) = \sum_{p_1 \in PC} \min_{p_2 \in PC_T} \|p_1 - p_2\|_2^2$$

$$+ \sum_{p_2 \in PC_T} \min_{p_1 \in PC} \|p_1 - p_2\|_2^2$$

   (b) Earth Mover's Distance

$$EMD(PC, PC_T) = \min_{\phi : PC -> PC_T} \sum_{p \in PC} \|p - \phi(p)\|_2$$

   where $\phi : PC \to PC_T$ is a bijection (one-to-one matching).

   (c) F1 Score

   i. Boundary: To calculate precision and recall, as stated below, we need a boundary for each object. We first get the candidates for the boundary by subtracting the minimum point from the maximum point per axis. After getting a candidate per axis, we take the largest value, and this becomes the boundary.

   ii. F1 Score:

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

   iii. Precision: Ratio of generated points, whether their closest points from the ground truth samples are within a threshold.

   iv. Recall: Ratio of ground truth points, whether their closest points from the generated samples are within a threshold.

3. Quantitative Analysis
   Here we picked seven randomly sampled chairs from the test dataset. For example, we have test objects 0100 to 1150. Using the three metrics that we have shown above, we computed the score of each object.

   (a) F1 Scores with Local+Global Feature

	1%	2%	5%	10%	20%
Ours	0.034	0.213	**0.886**	**0.997**	1.000
Theirs	**0.035**	**0.234**	0.836	0.993	1.000
Ours	0.013	0.075	0.456	0.964	**1.000**
Theirs	**0.014**	**0.109**	**0.754**	**0.977**	0.997
Ours	0.099	0.464	0.995	1.000	1.000
Theirs	**0.121**	**0.558**	**0.997**	1.000	1.000
Ours	**0.031**	**0.207**	**0.835**	**0.992**	1.000
Theirs	0.021	0.144	0.779	0.973	1.000
Ours	**0.023**	**0.148**	0.587	**0.986**	1.000
Theirs	0.020	0.142	**0.642**	0.983	1.000
Ours	0.044	0.282	0.794	0.964	0.998
Theirs	**0.055**	**0.319**	**0.955**	**1.000**	**1.000**
Ours	0.012	0.099	0.549	0.941	**1.000**
Theirs	**0.025**	**0.156**	**0.769**	**0.959**	0.994
Average	0.036	0.212	0.728	0.977	**0.999**
	0.041	**0.237**	**0.818**	**0.983**	0.998

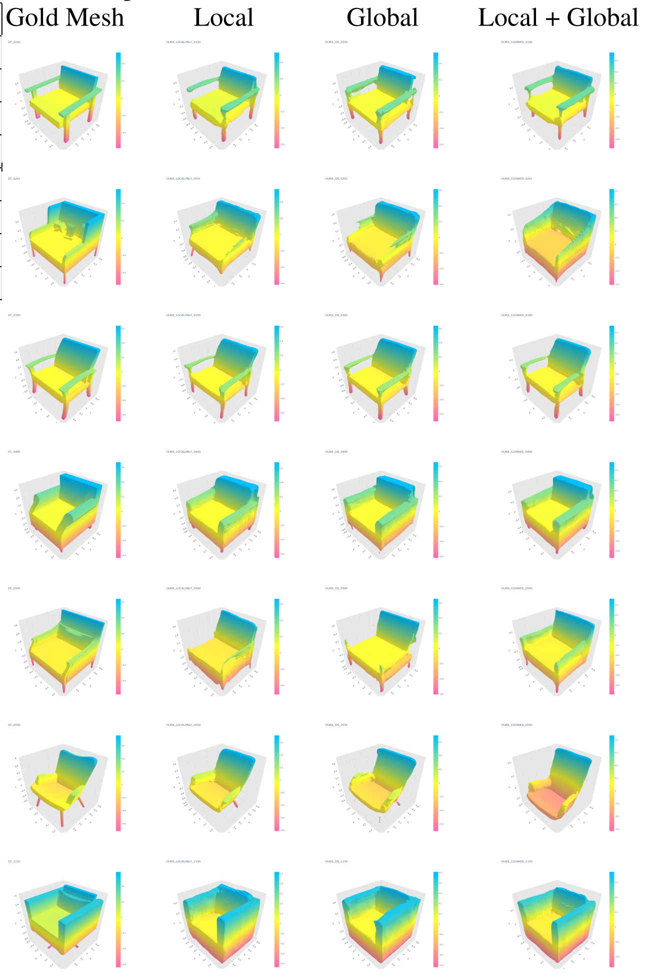
   (b) Chamfer Distance

Test Object	Local	Global	Local+Global	Theirs
0100	**9.56**	12.35	9.60	10.89
0201	16.07	16.45	25.04	**16.18**
0300	4.40	4.34	3.96	**3.50**
0400	12.84	16.17	**10.27**	14.75
0500	16.20	**15.41**	17.34	15.96
0550	17.96	13.84	18.86	**7.49**
1150	26.98	29.14	23.60	**16.93**
Average	14.85	15.38	15.52	**12.24**

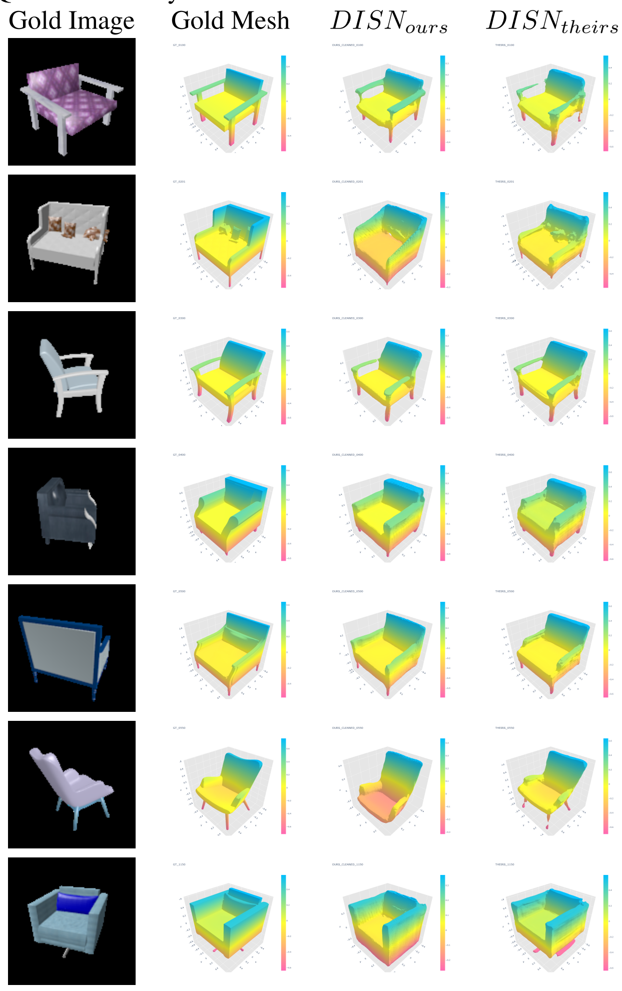
(c) Earth Mover's Distance

Test Object	Local	Global	Local+Global	Theirs
0100	166.74	167.62	**165.29**	172.92
0201	**186.80**	200.02	221.51	205.03
0300	126.12	109.47	108.83	**107.20**
0400	217.58	222.34	**194.50**	236.00
0500	194.82	216.64	199.67	**183.27**
0550	176.87	168.63	180.81	**147.89**
1150	240.26	243.54	237.28	**208.73**
Average	187.02	189.75	186.84	**180.14**

4. Feature Comparison



5. Qualitative Analysis



5. Discussion

Our results are slightly worse than the original paper's results, this is likely due to training time. Since we wanted to compare different models and only had one GPU available, we could only train our model for 100 epochs which is about 1/10 of the training time or their model. That being said we have comparable results.

When comparing our local feature only and global feature only models to the combined local and global feature model. We can see that the two restricted models perform worse overall. This shows that not only does adding local features improve the accuracy, but local or global features alone do not perform as well as the combination of the two.

6. Conclusion

We had successfully implemented a PyTorch version of DISN. Even though our overall results perform slightly worse than the original paper, we had more limitation in resources yet achieve comparable results. We also showed the importance of the combination of local and global features. There are several things we could try to improve our results. First of all, we can train our model with more training data and fore a longer time, using all 13 categories' and training for more epochs. Trying out different loss function could also improve our result, since currently we only use a simple L1 loss to train our network. Adding something like Eikonal regularization to enforce a consistent SDF, could potentially also improve accuracy.

References

[1] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 2

[2] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. *CoRR*, abs/1612.00603, 2016. 1

[3] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Atlasnet: A papier-mâché approach to learning 3d surface generation, 2018. 2

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 2

[5] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space, 2019. 1, 2

[6] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation, 2019. 2

[7] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017. 2

[8] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images, 2018. 2

[9] Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. 3dn: 3d deformation network, 2019. 2

[10] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 492–502. Curran Associates, Inc., 2019. 2, 3

[11] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision, 2017. 1