Implicit Neural Representation with PHASE Implementation

Rishi Dey Chowdhury
Indian Statistical Institute Kolkata

Abstract
This paper summarizes all the implementation details and important considerations which I came up with while implementing the PHASE paper on learning Implicit Neural Representation for surface reconstruction. I discuss about the motivation for the loss function, the numerical estimation of the loss functions, training details and some preliminary results. Additionally, I have also implemented the Fourier Feature Mapping, which is supposed to give better high-frequency details. I discuss briefly the statistical properties of our estimation and show that the estimator of the loss functions used are unbiased. My code is based at https://github.com/RishiDarkDevil/IGR-Phase-INR

Contents

1 Introduction

Implicit Neural Representation (INR) refers to using Neural Networks to model a function whose 0-level set gives the surface of the object we are trying to reconstruct. Here, I will focus on $3D$ reconstruction, though the concepts and ideas discussed here are applicable to $2D$ as well. Below, $\mathcal{S}_\theta$ refers to the $3D$ surface of the object.

$$\mathcal{S}_\theta = \{\mathbf{x} \in \mathbf{R}^3 | f_\theta(\mathbf{x}) = 0\}$$

Above, f_θ is parametrized with a Neural Network. There are several ways to estimate f, but here I will discuss and implement the learning of INRs with phase transition loss[2] which puts in the inductive bias of minimizing a phase transition inspired potential energy for surfaces, which can be intuitively thought as exterior of an object being one fluid whereas the interior being another being at equilibrium.

In order to learn the surface, we will model a occupancy function, say u which takes the value 1 in the exterior of an object, -1 at the interior of an object. Though due to an approximate formulation of the problem which shows Γ-convergence, we can have u values in the interval $[-1, 1]$ with ~ 0 near the boundary or the surface.

As discuss in the paper, I take the compact support containing the point cloud, say Ω, to be the axis aligned bounding box which is scaled by 1.5.

2 Loss Function

Various loss functions in the PHASE-based INR learning

2.1 WCH Loss

The overall loss function for PHASE-based training of neural network for surface reconstruction is the combination of 3 loss functions. Here, we will take a look at the Waals-Cahn-Hiliard Loss given by,

$$\mathcal{L}_{WCH}(u) = \int_{\Omega} \epsilon||\nabla u||^2 + W(u) \tag{1}$$

where, W the double well potential is of the form,

$$W(u) = u^2 - 2|u| + 1 \tag{2}$$

$W(u)$ takes value 0 only at $\{-1, 1\}$. This can be thought of as a global optimization where we try to push all the points outside the object towards 1 and those in the interior of the object towards -1.

But to implement this we need to estimate a continuous integral. In order to do so, I use the Monte-Carlo estimation technique. Since, the integral is over Ω, first I sample points using an uniform distribution over i.e. $\mathbf{X} \sim \mathcal{U}(\Omega)$. The pdf of r.v. $\mathbf{X}$ is given by,

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{Vol(\Omega)}$$

We can see below a strategy to estimate RHS of 1.

$$\begin{aligned}\mathcal{L}_{WCH}(u) &= \int_{\Omega} \frac{\epsilon||\nabla u(\mathbf{x})||^2 + W(\mathbf{u}(\mathbf{x}))}{f_{\mathbf{X}}(\mathbf{x})} f_{\mathbf{X}}(\mathbf{x}) \\ &= \mathbf{E}_{\mathbf{X}} Vol(\Omega) \cdot (\epsilon||\nabla u(\mathbf{X})||^2 + W(u(\mathbf{X})))\end{aligned} \tag{3}$$

Now, that I have an expectation we can easily estimate it with the unbiased estimate sample mean which goes to the true value almost surely by Strong Law of Large Numbers. Thus, the estimator I use is given by,

$$\hat{\mathcal{L}}_{WCH}(u) = \frac{Vol(\Omega)}{N} \sum_{i=1}^{N} \epsilon||\nabla u(\mathbf{X_i})||^2 + W(u(\mathbf{X}_i)) \tag{4}$$

where $\{X_i\}_{i=1}^N$ is our sampled points from Ω.

This is used in as an estimator of the WCH Loss in the implementation.

2.2 Reconstruction Loss

The second piece that goes in the loss function is the Reconstruction Loss, which is given by,

$$\mathcal{L}_R(u) = \mathbf{E}_{\mathbf{X}} |\frac{1}{|B_{\mathbf{X}}|} \int_{B_{\mathbf{X}}} u| \tag{5}$$

This loss can be viewed as a local optimization where we force u (softly) to pass through an ϵ-ball around each point in the input point cloud, which is a desirable property of the surface I am reconstructing.

Now, observe the term over which expectation is taken is nothing but the mean value of u within each ball which we randomly select uniformly from the input point cloud. The outer expectation is the average of these means for randomly selected ϵ-balls.

Similarly, using SLLN, I use the sample mean as the estimator for the expectation and instead of continous mean calculated in each ball, I use the discrete mean, which by Riemann Approximation to integrals go to the actual value as the number of sampled points inside each ball goes to ∞. The final estimator is given by,

$$\hat{\mathcal{L}}_R(u) = \frac{1}{N \cdot M} \sum_{i=1}^{N} \sum_{j=1}^{M} u(\mathbf{X}_{ij}) \tag{6}$$

where, N is the number of randomly sampled ϵ-balls and M is the number of randomly sampled points inside each ball. The paper discusses about the sampling inside the ball to be $\mathbf{X}_i \sim \mathcal{N}(\mathbf{x}, \sigma^2)$, where $\mathbf{x}$ is one of the randomly sampled points from the input point

cloud. Although the ball's radius is infinite, for practical purpose it works fine as most of the mass is concetrated near the point.

2.3 Normal Loss

The paper also discusses about an additional loss which can be incorporated in the normal data is given along with the point cloud dataset. It is given by,

$$\mathcal{L}_N(u) = \mathbf{E}_\mathbf{X}||\mathbf{n}(\mathbf{x}) - \nabla\mathbf{w}(\mathbf{x})||^p \quad (7)$$

where, $n(x)$ is the normal vector for $\mathbf{x}$ and $\nabla\mathbf{w}(\mathbf{x}) = \sqrt{\epsilon}\mathbf{u}(\mathbf{x})$. In case, normal data is not provided the paper suggests using the following unit normal loss,

$$\mathcal{L}_N(u) = \mathbf{E}_\mathbf{X}|1 - ||\nabla\mathbf{w}(\mathbf{x})||^p \quad (8)$$

Again, we use the sample mean as the estimator for this loss given by,

$$\hat{\mathcal{L}}_N(u) = \frac{1}{N}\sum_{i=1}^{N}|1 - ||\nabla\mathbf{w}(\mathbf{X_i})||^p \quad (9)$$

2.4 Final Loss

Our overall loss function which needs to be minimized is given by

$$\hat{\mathbf{L}}(u) = \lambda\hat{\mathcal{L}}_R(u) + \hat{\mathcal{L}}_{WCH}(u) + \mu\hat{\mathcal{L}}_N(u) \quad (10)$$

3 Model

Architecture and Hyperparameters

3.1 Multi-Layer Perceptron

The paper describes using a simple Multi-Layer Perceptron for parametrization of the occupancy function u. I use two MLP architectures:

- **PHASE**: 8 Layers with 512 Neurons each with Softplus Activation and skip connection from input to 4^{th} Layer.
- **PHASE+FF**: 5 Layers with 256 Neurons each with Softplus Activation and skip connection from Fourier Feature Mapped Inputs to 3^{rd} Layer.

A learning rate of 0.05 is used to train both the models for 1000 epochs with a moderate batch size of 7000 points which are randomly selected per epoch. I sample 10000 points from Ω for estimation of WCH Loss and 50 points per ϵ-ball for reconstruction loss. The balls and points used in Reconstruction Loss and Normal Loss respectively are same as the 7000 points sampled.

3.2 Fourier Feature Mapping

The author mentions about using Fourier Feature Mapping for better high-frequency feature reconstruction. I used $k = 6$ Fourier features which are passed onto the PHASE+FF Network. These features are generated from the input $\mathbf{x} \in \mathbf{R}^3$ using the following function,

$$\gamma(\mathbf{x}) = [(2^i\pi\sin(\mathbf{x_j})), (2^i\pi\cos(\mathbf{x_j}))]_{i=1,j=1}^{i=k,j=3} \quad (11)$$

where, $\gamma : \mathbf{R}^3 \to \mathbf{R}^{6k}$ is a function which maps the input to the Fourier features.

3.3 Hyperparameters

I noticed that there is a trade-off between which loss the network prioritizes to minimize while using different choices of hyperparameter. Higher value of λ prioritizes minimizing Reconstruction Loss, but slightly ignores the WCH Loss. The hyperparameters that needs to be set are ϵ, μ, λ. The choice of hyperparameter values are:

- PHASE: $\mu = 0.2\lambda = 1, \epsilon = 0.01$
- PHASE+FF: $\mu = \lambda = 10, \epsilon = 0.01$

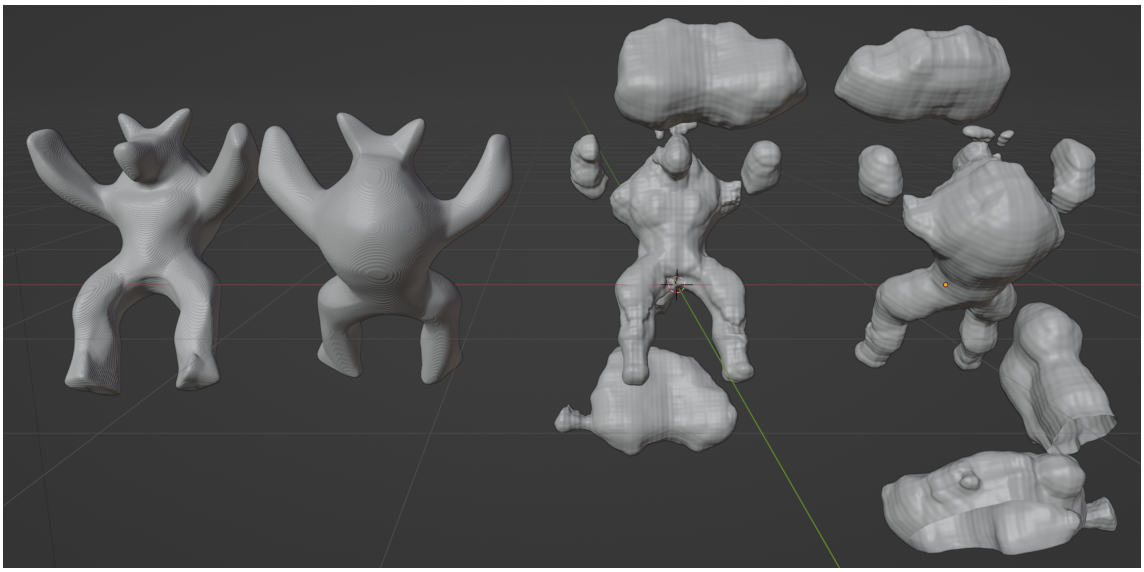

Figure 1: The left 2 reconstruction are by PHASE and the right 2 are by PHASE+FF

4 Results

Evaluation and Qualitative Results

4.1 Chamfer Distance

The metric used to evaluate the reconstructed $3D$ surface is the double-sided Chamfer distance, denoted by d_C. It is usually defined for 2 point clouds $\mathcal{X}$ and $\mathcal{Y}$. But, since we have reconstructed meshes or surfaces, we sample evenly and desnsely from both the target mesh and the reconstructed mesh and then compute the double-sided Chamfer Distance of these point clouds. The one-sided Chamfer Distances and double-sided Chamfer Distance is given by,

$$\begin{aligned} \overrightarrow{d}_C(\mathcal{X}, \mathcal{Y}) &= \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{y} \in \mathcal{Y}} ||\mathbf{x} - \mathbf{y}||_2 \\ \overrightarrow{d}_C(\mathcal{Y}, \mathcal{X}) &= \frac{1}{|\mathcal{Y}|} \sum_{\mathbf{y} \in \mathcal{Y}} \min_{\mathbf{x} \in \mathcal{X}} ||\mathbf{y} - \mathbf{x}||_2 \\ d_C(\mathcal{X}, \mathcal{Y}) &= \frac{1}{2}(\overrightarrow{d}_C(\mathcal{X}, \mathcal{Y}) + \overrightarrow{d}_C(\mathcal{Y}, \mathcal{X})) \end{aligned} \tag{12}$$

The number of points sampled evenly from both the target point cloud and the reconstructed point cloud is 1 million.

Model	PHASE	PHASE-FF
Armadillo	0.48	0.49

Above is the Chamfer Score for the final reconstructed surface given the target surface. The surface was reconstructed using 10000 points. Although the score isn't great but I ran for only 1000 epochs with a small batch size. Similar results regarding Chamfer Distance comparison is stated in the paper, except the distance is quite low because the author ran the training for longer duration.

4.2 Qualitative Results

Taking a look at figure 1, it seems that PHASE is consistent and smoother compared to PHASE+FF. But figure 2 shows that PHASE+FF is in fact faster, even though it misses some finer details and high-level frequency structure. It is clearly visible that the surface reconstructed by PHASE+FF is closer to the ground truth, except the fact that due to very less batch size and number of epochs of training, there are few artifacts

in the PHASE+FF surface reconstruction. Though it is clear that they will go away with further training. The authors mention that PHASE requires orders of magnitude more iterations to achieve better reconstruction than PHASE+FF but eventually gets there with slightly superior quality results compared to PHASE.

Figure 2: PHASE (left), Ground Truth (middle) and PHASE+FF (right) a closer look

5 Conclusions

Here, I have discussed the implementation details and results obtained from PHASE-based training of Implicit Neural Representation for $3D$ reconstruction. The results mentioned in this paper are highly constrained by the computational resources i.e. Colab Environment, with limited GPU RAM and usage time. Nonetheless, I am able to reproduce the paper with preliminary results and working code. The code for the PHASE implementation was added on top of the IGR: Implicit Geometric Regularization for Learning Shapes[1] codebase. Some issues faced are vanishing gradient in larger network, which I tried handling by scaling the losses and adding skip connections.

References

[1] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes, 2020.

[2] Yaron Lipman. Phase transitions, distance functions, and implicit neural representations, 2021.