Multimedia Content Analysis- Course Project Report

Sanyam Kaul
CS23MTECH14011

1. Abstract

This report gives a brief account of the work done by the author as part of the Multimedia Content Analysis course project for the winter semester of the academic session 2023-24. The project aimed to study a recent research paper from any top conference, try to reproduce the results claimed in the paper, figure out the current gaps and propose a novelty (if any).

Main paper for the course project- [1]

Additional papers for finalizing novelty- [2] and [3]

2. Motivation

As part of the course project, the author studied the paper - RayDf: Neural Ray-surface Distance Fields with Multi-View Consistency.

The research paper has been published in NeurIPS 2023. The author of this report successfully demonstrated a deep understanding of the paper in both class presentations. Due to the institution's lack of GPU support, executing the model proposed on the whole dataset (mentioned in the paper) was impossible. However, the author successfully demonstrated the execution of the model in its local system (CPU-based) and achieved decent metrics.

The following sections provide a summary of the background of the problem domain in which the scope of the paper lies, recent work done in the domain, the model proposed in the studied paper, the re-produced metrics and the proposed scope of improvement/ novelty.

3. Problem Statement

The paper focuses on learning accurate and efficient 3-D shape representations. This has several use cases ranging from computer-aided design and augmented reality to virtual reality and robotics. The model from this problem domain is designed to work on various tasks like shape reconstruction, novel view synthesis, scene understanding, and obtaining 3D shapes and 2D viewers from the trained networks.

3.1. 3-D Representations

There are two approaches formulated to represent 3-D shapes in computers. The first one is the Coordinate based approach, and the second is the ray-based approach: -

3.1.1 Coordinate Based Approach

The main idea of this approach is to learn all the details about the edges, corners and faces of the 3-D shape/ object under study. Later, these measured values are used to regenerate the 3-D shape. Note that the surroundings of the object are not important while learning its measurements. Many techniques are built using this approach, which try to regenerate the 3-D shapes. Some examples are mesh grids, voxel grids, etc

3.1.2 Drawback of Coordinate Based Approach

The major drawback of using the coordinate-based approach is that as the complexity of the shapes of objects increases, the training time and amount of computations increase exponentially. It is becoming increasingly difficult for us to train models on real-life datasets. As a result, the efficiency of the models drops.

3.1.3 Ray Based Approach

The main idea of the Ray-based approach is that we don't bother measuring the object under observation. Rather, we assume a sphere around the object and measure the distance of the object's surface from the sphere's boundary. Once the distances are learned, the object's surface can be regenerated using the same sphere and the learned distances.

3.1.4 Advantage of Ray-Based Approach over coordinate-based approach

By using the ray-based approach in the model, we reduce the computational cost of our models as in the coordinate-based approach, we have to deal with high dimensional matrix multiplication, but in the case of the ray-based approach, we have to rely on Euclidian distances.

4. Challenges in the current work

The ray-based models outperform the coordinate-based models by a margin regarding training time and accuracy. However, the major issue with the current SOTA ray-based models is the problem of multi-view consistency. For example, we can see the regeneration of a sink by a SOTA ray-based model, LFN. We can see in Figure 1 that the regeneration is smooth and accurate from the front view, but there is considerable distortion on the other sides of the sink.

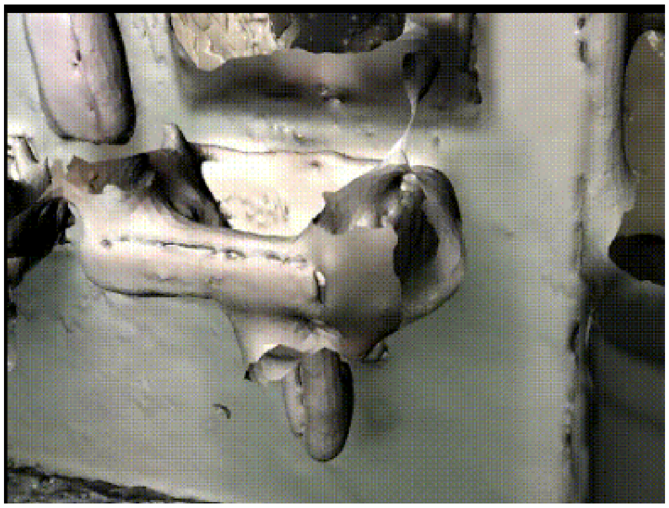


Figure 1. Regeneration by LFN

4.1. Cause of Multi-View Consistency

The main cause behind multi-view consistency is that the rays, which measure the distance of the object's surface from the sphere's surface, are considered independent. Note that 2 rays falling on the same point on the object provide us with two different views of looking at the same object. However, if these two rays are considered independent, the model learns two views of the same point on the object during object regeneration. This results in distorted regeneration.

In the proposed model, RayDF, the authors have devices that provide a novel way to add the notion of dependency among rays and use it to learn 3D objects. Figure **??** shows the regeneration of the sink using rayDF, which is the proposed model. We can see that the problem of multi-view consistency has been drastically reduced.

5. Proposed Methodology

Figure 3 shows the high-level overview of the proposed architecture, RayDF

The architecture has three modules. The following sections provide a brief description of the modules. The workings of each module were discussed in depth during the class presentations. Please refer to the main research paper for more details.

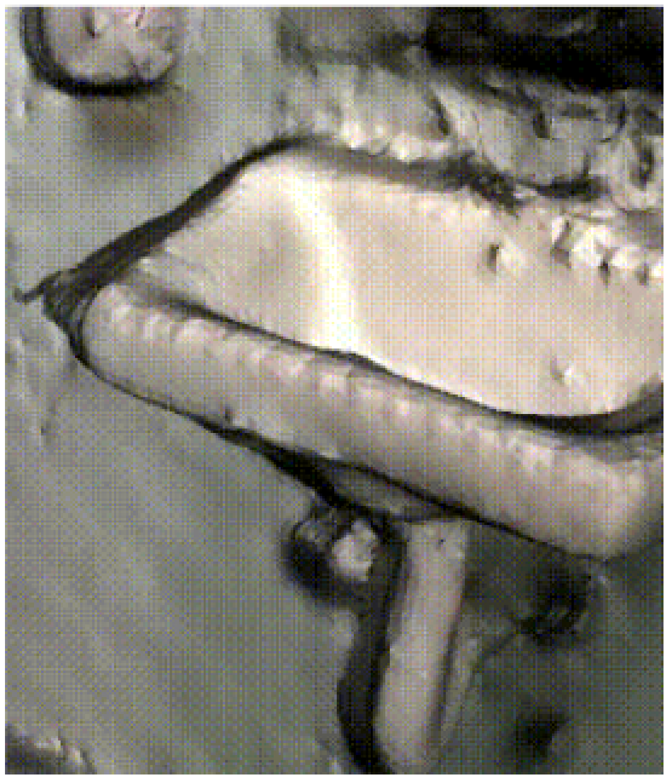


Figure 2. Regeneration by RayDF (Proposed Model)

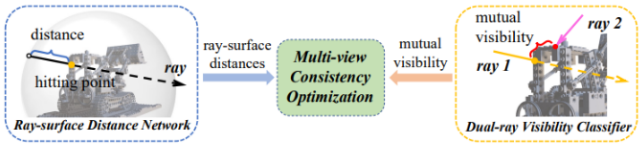

distance
hitting point
ray
Ray-surface Distance Network
ray-surface distances
Multi-view Consistency Optimization
mutual visibility
mutual visibility
ray 2
ray 1
Dual-ray Visibility Classifier

Figure 3. Regeneration by RayDF (Proposed Model)

5.1. Ray-Surface Distance Fields

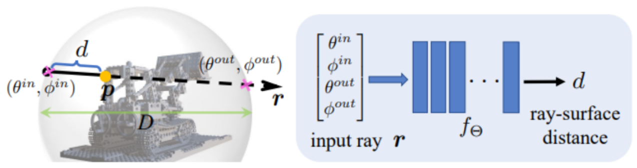
This module is used to learn the distance of the object's surface from the sphere's surface. It takes the spherical parameterization of the ray (radial distance, polar angle and azimuthal angle0 as input and regresses the distance between the ray starting point and the surface hitting point. Figure 4 shows the mathematical formulation of the module.

$$d = f_\Theta(\boldsymbol{r}), \quad \text{where } \boldsymbol{r} = (\theta^{in}, \phi^{in}, \theta^{out}, \phi^{out}), \quad \Theta \text{ are trainable parameters of MLPs}$$

Figure 4. Distance regression using Ray-Surface Distance Field module

Figure 5 shows the architecture of Ray- surface Distance

Field module.



d
(θ^{in}, ϕ^{in})
p
$(\theta^{out}, \phi^{out})$
r
D
θ^{in} ϕ^{in} θ^{out} ϕ^{out}
input ray r
f_Θ
d
ray-surface distance

Figure 5. Ray-Surface distance field module

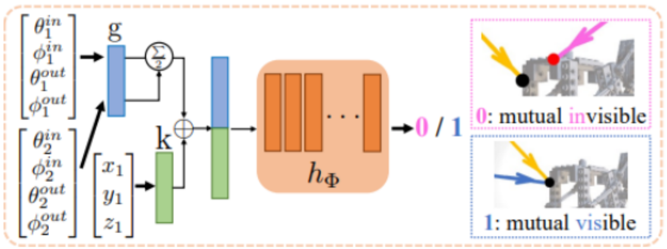
5.2. Dual Ray Visibility Classifier

This module is responsible for quantifying the dependencies among the rays hitting the surface of the object under observation. The main logic behind the computation done by this module is as follows: -

As the name specifies, the module is a simple classifier. It classifies whether a pair of rays hit the object at the same point on the surface or not. If the rays hit the same point on the surface, the classifier will classify the pair of rays as 1. Otherwise, it classifies them as 0, signifying that the rays hit different points on the object's surface. Here, it's interesting to note that the values of 0 and 1 can also be real values (ranging from 0 to 1) if a continuous activation function like sigmoid is used at the output layer of the classifier's fully connected deep neural network.

The classifier value signifies the level of visibility of the point on the surface, as seen by both the rays inputted to the classifier. The closer the rays' hitting points are on the object's surface, the higher the visibility value.

Figure 6 shows the architecture of the Dual Ray Visibility Classifier. The 2 rays are inputted to the classifier in their parameterized form. Both the rays are passed through a fully connected layer first and then global average pooling is performed on the output of the layer. This adds the parameterized formulation of both the rays. It's interesting to note that we could have just added the rays using the arithmetic addition operator. However, we are not doing arithmetic addition because we want to make the output of the addition independent of the location of the inputted rays.



θ_1^{in} ϕ_1^{in} θ_1^{out} ϕ_1^{out}
g
Σ
θ_2^{in} ϕ_2^{in} θ_2^{out} ϕ_2^{out}
x_1 y_1 z_1
k
h_Φ
0 / 1
0: mutual invisible
1: mutual visible

Figure 6. Ray-Surface distance field module

Figure 7 depicts three rays hitting the object's surface. We can see that r1 and r2 hit the surface at the same point (x1, y1, z1). So, the visibility value will be close to 1 for r1 and r2. However, r1 and r2 do not hit the surface simultaneously. Therefore, the visibility value for this pair will be close to 0.

Also, note that we must calculate the surface hitting point for all the rays to perform our computation. We take two rays and calculate their surface hitting points. If the surface hitting points of both the rays come out to be the same, it means that they are hitting the surface at the same point.

We get the point on the sphere where a ray enters and exits the sphere. Using this information, we calculate the unit vector of the ray, which tells us the direction of the ray. Then, we multiply the unit vector with the measured distance between the sphere and the object's surface and use Euclidian's formulation to get the surface hitting point. This calculation is formulated in Figure 8

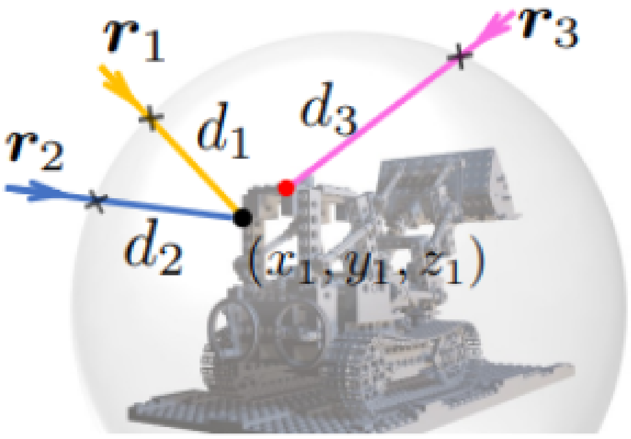


Figure 7. Ray-Surface distance field module

$$\boldsymbol{r}_1^{in} + d_1 \boldsymbol{r}_1^{\boldsymbol{d}} = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} = \boldsymbol{r}_2^{in} + d_2 \boldsymbol{r}_2^{\boldsymbol{d}}, \text{ where } \boldsymbol{r}^{\boldsymbol{d}} = \frac{\boldsymbol{r}^{out} - \boldsymbol{r}^{in}}{\|\boldsymbol{r}^{out} - \boldsymbol{r}^{in}\|} \text{ and } \boldsymbol{r}^* = \begin{pmatrix} \sin\theta^* \cos\phi^* \\ \sin\theta^* \sin\phi^* \\ \cos\phi^* \end{pmatrix}, * \in \{in, out\}$$

$$h_\Phi : \quad MLPs\left[\frac{g(\theta_1^{in}, \phi_1^{in}, \theta_1^{out}, \phi_1^{out}) + g(\theta_2^{in}, \phi_2^{in}, \theta_2^{out}, \phi_2^{out})}{2} \oplus k(x_1, y_1, z_1)\right] \rightarrow 0/1$$

Figure 8. Distance regression using Ray-Surface Distance Field module

5.3. Multi-View Consistency optimizer

This is the module which trains the ray surface module by taking into account the visibility values provided by the classifier. The following text in this section provides a summary of the training steps incorporated to train the proposed model: -

- Given - K posed depth images (H X W) of static 3D scenes

- Training module consists of 2 stages: Stage 1 - Training Dual-ray Visibility Classifier Stage 2 - Training Ray-Surface Distance Network
- Ultimate goal - optimize main Ray-surface Network and drive it to be Multi-View Consistent
- Converting depth images to ray-surface distance - We get K*H*W training ray-distance pairs for a specific 3D scene
- For 1 training ray(r,d), we sample M rays in a ball centring at surface point p - Figure 9
- Calculate distances between p and the M sampled rays - Figure 10
- Generate M ray pairs, feed them to Classifier and get Visibility Scores - Figure 11
- Feed primary ray and M sampled rays to Ray-surface distance network and estimate the surface distances - Figure 12
- Use the Multi-View Consistent Loss Function to optimize the ray-surface distance until convergence - Figure 13

$$\{\boldsymbol{r}^1 \cdots \boldsymbol{r}^m \cdots \boldsymbol{r}^M\}$$

Figure 9. M sampled ray along with 1 training ray

$$\{\tilde{d}^1 \cdots \tilde{d}^m \cdots \tilde{d}^M\}$$

Figure 10. Calculated distances for training and sampled rays

$$\{v^1 \cdots v^m \cdots v^M\}$$

Figure 11. Classifier scores for M rays

$$\{\hat{d}, \hat{d}^1 \cdots \hat{d}^m \cdots \hat{d}^M\}.$$

Figure 12. Estimated distances for the M rays

Looking at the loss function, we can see that it is normalizing the sum of all the errors in estimated and calculated distances. However, for the M sampled rays, we are multiplying the error in distance with the visibility score before adding it to the total loss. This is how the visibility score is

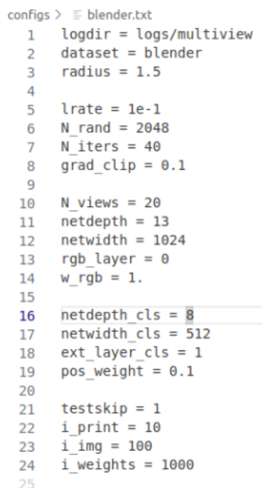
$$\ell_{mv} = \frac{1}{\sum_{m=1}^{M} v^m + 1}\left(|\hat{d} - d| + \sum_{m=1}^{M}\left(|\hat{d}^m - \tilde{d}^m| * v^m\right)\right)$$

Figure 13. Loss function

incorporated into the model's overall learning. Only those rays will contribute to the overall loss, which has a high visibility score with the training ray. Intuitively, we can see that the model will learn the view of a single point on the object's surface from different angles like this. The overall distortion in the regeneration will, therefore, be reduced.

6. Reproduced Results

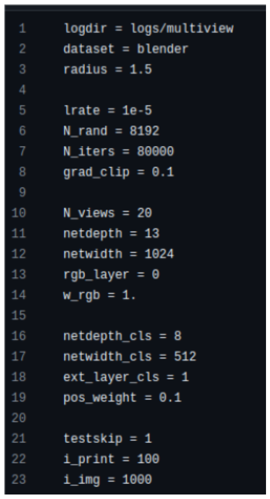
The paper's author evaluated the proposed model using Blender,DM-SR, and ScanNet datasets. I set up and ran the proposed model locally (on CPU). I made some modifications to the code base, which removed all the Cuda dependencies from the code to make it executable on CPU. Also, I modified the data loader file to load only 5 images from the Lego directory of the Blender dataset for local execution. Figure 14 shows the hyperparameters initialized while running the model locally, and Figure 15 shows the original hyperparameters. I reduced the scale of these parameters to make the code CPU executable. Similarly, the hyperparameters were also reduced for the classifier.



```
configs > blender.txt
logdir = logs/multiview
dataset = blender
radius = 1.5

lrate = 1e-1
N_rand = 2048
N_iters = 40
grad_clip = 0.1

N_views = 20
netdepth = 13
netwidth = 1024
rgb_layer = 0
w_rgb = 1.

netdepth_cls = 8
netwidth_cls = 512
ext_layer_cls = 1
pos_weight = 0.1

testskip = 1
i_print = 10
i_img = 100
i_weights = 1000
```

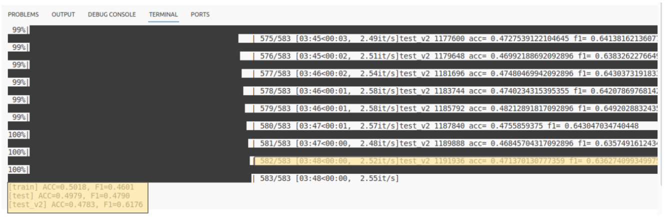
Figure 14. Hyperparameters used locally for Ray Surface Module

Figure 16 shows the ACC and F1 scores achieved during local execution. No modification was made to the logic of the model. The SOTA metrics can be achieved using the original hyperparameters, the whole dataset, and GPU-based training.



```
logdir = logs/multiview
dataset = blender
radius = 1.5

lrate = 1e-5
N_rand = 8192
N_iters = 80000
grad_clip = 0.1

N_views = 20
netdepth = 13
netwidth = 1024
rgb_layer = 0
w_rgb = 1.

netdepth_cls = 8
netwidth_cls = 512
ext_layer_cls = 1
pos_weight = 0.1

testskip = 1
i_print = 100
i_img = 1000
```

Figure 15. Original hyperparameters of Ray Surface Module



```
 99%|                                   | 575/583 [03:45<00:03,  2.49it/s]test_v2 1177600 acc= 0.4727539122104645 f1= 0.641381621360
 99%|                                   | 576/583 [03:45<00:02,  2.51it/s]test_v2 1179648 acc= 0.46992188692092896 f1= 0.63832622766
 99%|                                   | 577/583 [03:46<00:02,  2.54it/s]test_v2 1181696 acc= 0.47480469942092896 f1= 0.64303731918
 99%|                                   | 578/583 [03:46<00:01,  2.58it/s]test_v2 1183744 acc= 0.4740234315395355 f1= 0.642078697681
 99%|                                   | 579/583 [03:46<00:01,  2.58it/s]test_v2 1185792 acc= 0.48212891817092896 f1= 0.64920288324
 99%|                                   | 580/583 [03:47<00:01,  2.57it/s]test_v2 1187840 acc= 0.4755859375 f1= 0.643047034740448
100%|                                   | 581/583 [03:47<00:00,  2.48it/s]test_v2 1189888 acc= 0.46845704317092896 f1= 0.63574916124
100%|                                   | 582/583 [03:48<00:00,  2.52it/s]test_v2 1191936 acc= 0.471370130777359 f1= 0.636274099349
100%|                                   | 583/583 [03:48<00:00,  2.55it/s]
[train] ACC=0.5018, F1=0.4601
[test] ACC=0.4979, F1=0.4790
[test_v2] ACC=0.4783, F1=0.6176
```

Figure 16. Results of local execution

7. Limitations

The following limitations have been identified in the proposed approach: -

- Both module-1 and module-2 of the proposed model work on the parameterized form of rays stored locally on the machine where the modules are being trained. The data preparation thus plays a crucial role. The process of data preparation is likely decentralized among different nodes. This leads to an added overhead of data collection in one place before we can train the data. Transferring the data to a central node can lead to data leakage and tampering issues, which is not healthy for the overall accuracy of the model.

- Since the data preparation procedure can be decentralized, it is also likely that the final processed data at nodes will not be IID, which increases the training complexity

8. Proposed Novelty

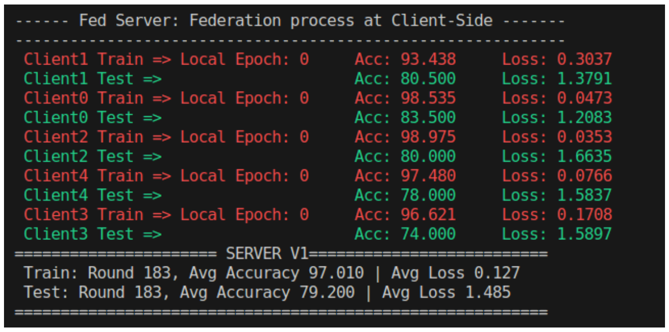
The current model can be enhanced in the following couple of ways: -

- The architecture of the current model can be enhanced into a Federated Learning set-up. The model can be trained locally at multiple endpoints and then averaged using Fed Averaging to get the final global model. This makes the whole training process more decentralized and increases data privacy. Moreover, this paper proposes a state-of-the-art hybrid algorithm called SplitFed, a hybrid of Split Learning and Federated. The author of this paper studied the algorithm separately, and the details are in the next section. The results indicate that the SplitFed algorithm performs better than the vanilla federated learning approach. So, the SplitFed algorithm can be utilised in the proposed model in our case while aiming to enhance the scope/ functionality of our model.

- Further, countering the presence of non-IID data at the endpoints, this paper proposes utilizing sample weights while training a model under a federated learning paradigm. More details are in the next section.

9. Novelty Implementation and Observation

The complete proposed novelty could not be implemented during the semester because of time constraints and lack of infrastructure support. However, The paper on SplitFed was successfully studied and implemented locally for result reproduction. Going into this paper's details would be a separate topic and beyond the scope of this report. But, talking about the results obtained after local execution, 17 and 18 show the snapshots of the accuracy score obtained.

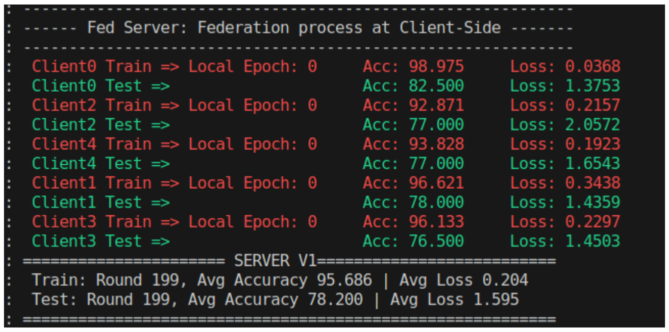


```
------ Fed Server: Federation process at Client-Side -------
-----------------------------------------------------------
 Client1 Train => Local Epoch: 0     Acc: 93.438     Loss: 0.3037
 Client1 Test =>                     Acc: 80.500     Loss: 1.3791
 Client0 Train => Local Epoch: 0     Acc: 98.535     Loss: 0.0473
 Client0 Test =>                     Acc: 83.500     Loss: 1.2083
 Client2 Train => Local Epoch: 0     Acc: 98.975     Loss: 0.0353
 Client2 Test =>                     Acc: 80.000     Loss: 1.6635
 Client4 Train => Local Epoch: 0     Acc: 97.480     Loss: 0.0766
 Client4 Test =>                     Acc: 78.000     Loss: 1.5837
 Client3 Train => Local Epoch: 0     Acc: 96.621     Loss: 0.1708
 Client3 Test =>                     Acc: 74.000     Loss: 1.5897
==================== SERVER V1======================
 Train: Round 183, Avg Accuracy 97.010 | Avg Loss 0.127
 Test: Round 183, Avg Accuracy 79.200 | Avg Loss 1.485
===================================================
```

Figure 17. Original hyperparameters of Ray Surface Module

The obtained accuracies in multiple runs are close to the SOTA metrics shown in the paper despite running the local execution for a lower number of epochs and on less dataset volume. These experiments show the reliability and consistency of the SplitFed algorithm in the decentralized learning paradigm.

The paper on tacking non-IID data using sample weights was also studied. However, this approach's implementation could not be completed within the course timelines.



```
: ------------------------------------------------------------
: ------ Fed Server: Federation process at Client-Side -------
: ------------------------------------------------------------
:  Client0 Train => Local Epoch: 0     Acc: 98.975     Loss: 0.0368
:  Client0 Test =>                     Acc: 82.500     Loss: 1.3753
:  Client2 Train => Local Epoch: 0     Acc: 92.871     Loss: 0.2157
:  Client2 Test =>                     Acc: 77.000     Loss: 2.0572
:  Client4 Train => Local Epoch: 0     Acc: 93.828     Loss: 0.1923
:  Client4 Test =>                     Acc: 77.000     Loss: 1.6543
:  Client1 Train => Local Epoch: 0     Acc: 96.621     Loss: 0.3438
:  Client1 Test =>                     Acc: 78.000     Loss: 1.4359
:  Client3 Train => Local Epoch: 0     Acc: 96.133     Loss: 0.2297
:  Client3 Test =>                     Acc: 76.500     Loss: 1.4503
: ====================== SERVER V1===========================
:  Train: Round 199, Avg Accuracy 95.686 | Avg Loss 0.204
:  Test: Round 199, Avg Accuracy 78.200 | Avg Loss 1.595
: ==========================================================
```

Figure 18. Results of local execution

10. Future Work

The future work itinerary focuses on implementing the proposed novel approaches to enhance the current proposed model, which includes: -

- Implementation of the Decentralized RayDF using the SplitFed algorithm
- Implementation of the algorithm of sample weights on the decentralized RayDF to enhance its efficiency and accuracy on non-IID data.
- Performing light and load tests on the modified model to look for enhancements in various metric scores

11. Conclusion

In conclusion, the author of the report takes pride in achieving the following milestones under this course project: -

- Successfully studied the main paper on RayDF and understood the proposed model's intuition, logic and implementational nitty-gritty.
- Successfully demonstrated a deep understanding of the topic in both class presentations.
- Successfully executed the proposed approach locally and achieved decent results given the resource constraints.
- Successfully studied two other papers for a foundational understanding for proposing novelties on the current model
- Successfully reproduced the results locally for SplitFed algorithm paper and proposed a novel approach to utilize the algorithm for decentralizing the current RayDF model
- Successfully proposed a novel approach to making the decentralized rayDF model work efficiently on non-IID, based on the technique of using sample weights.
- Successfully proposed the future work plan to achieve the pending implementation of the proposed novelty

References

[1] Zhuoman Liu, Bo Yang, Yan Luximon, Ajay Kumar, and Jinxi Li. Raydf: Neural ray-surface distance fields with multi-view consistency, 2023.

[2] Hung Nguyen, Peiyuan Wu, and Morris Chang. Federated learning for distribution skewed data using sample weights, 2024.

[3] Chandra Thapa, M. A. P. Chamikara, Seyit Camtepe, and Lichao Sun. Splitfed: When federated learning meets split learning, 2022.