README

Norms VS Offset Experiment

This experiment determines whether the dataset built using offset points or the dataset using the surface normals is better for the model to learn 3D textures.
This experiment was conducted on 3 meshes:
crinkleball.stl
Spikeyball1.stl
blobball.stl
I first created these datasets for norms and offsets by using the norms_experiment_pipeline and the offset_experiment_pipeline. Each mesh is smoothed using laplacian for 3000 iterations. From there, I saved and loaded the dataset and ran each dataset through the GINR model with 20 epochs and batch sizes of 1000. I chose 20 epochs because it is a middle ground between the offset datasets gaining more texture over 20 epochs and the norms dataset becoming more elongated over 20 epochs.

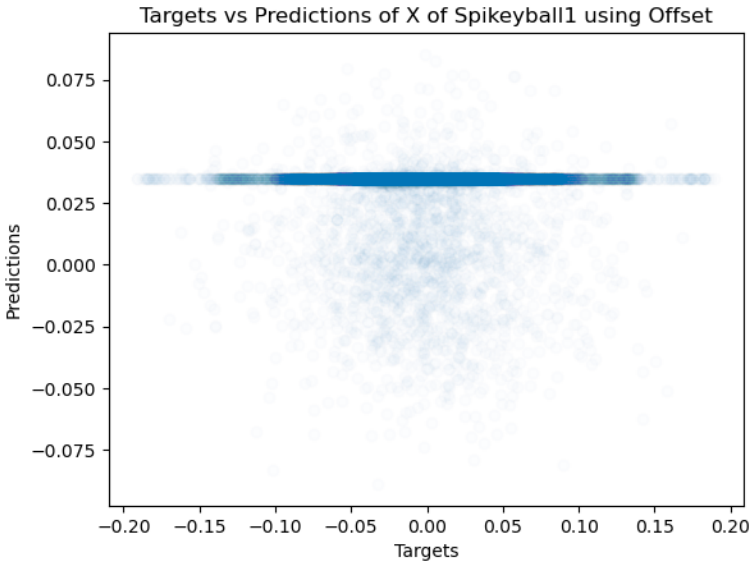
After calculating the predictions, we got the new points. In the norms_experiment_pipeline, the new_pts are shaped by reshaping the dataset points and assigning them to corresponding predictions. In the offset_experiment_pipeline, the new_pts are calculated by subtracting the predictions from the dataset points.

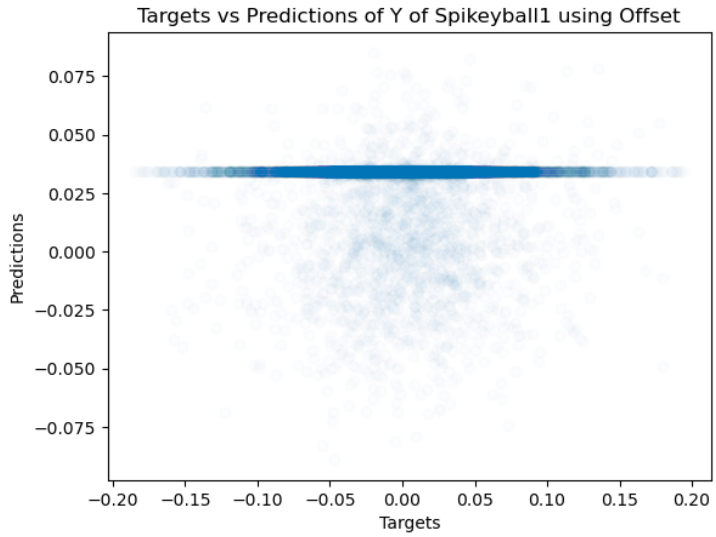
Then the final learned mesh is created using the new_pts and the original mesh faces to create a trimesh.

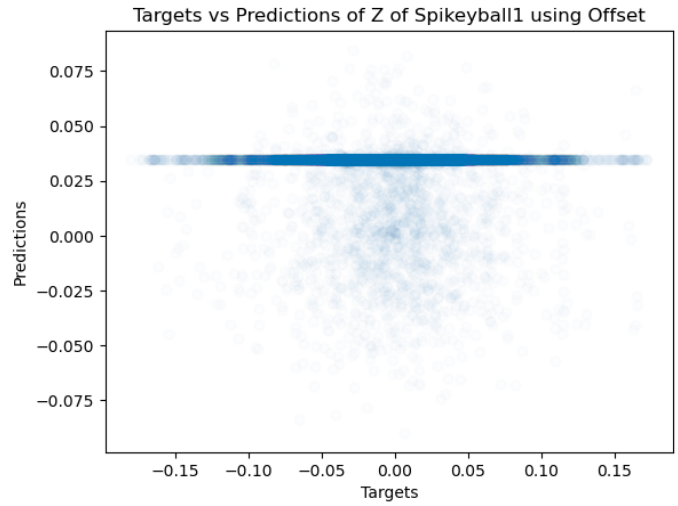
I created plots comparing the targets of the dataset versus the predicted values for x,y,z of the points. I did this for every dataset I created: blobball with norms,

After repeating this for every mesh listed in the norms_experiment_pipeline and the offset_experiment_pipeline, I was able to compare the plots to determine

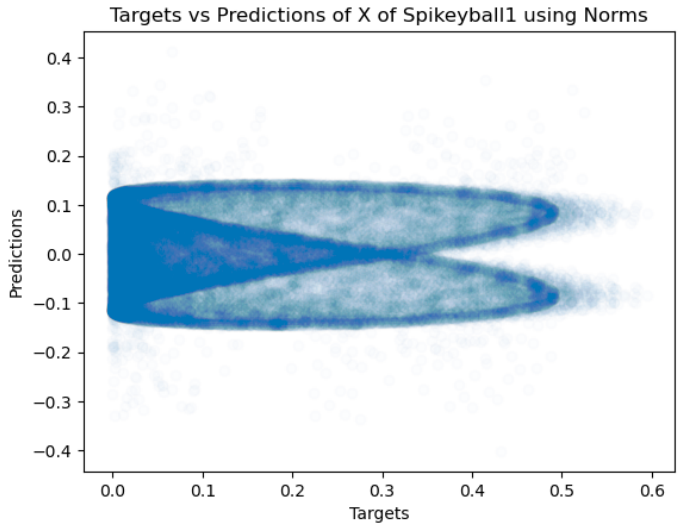
Plots:
Spikeyball1:
Offset:

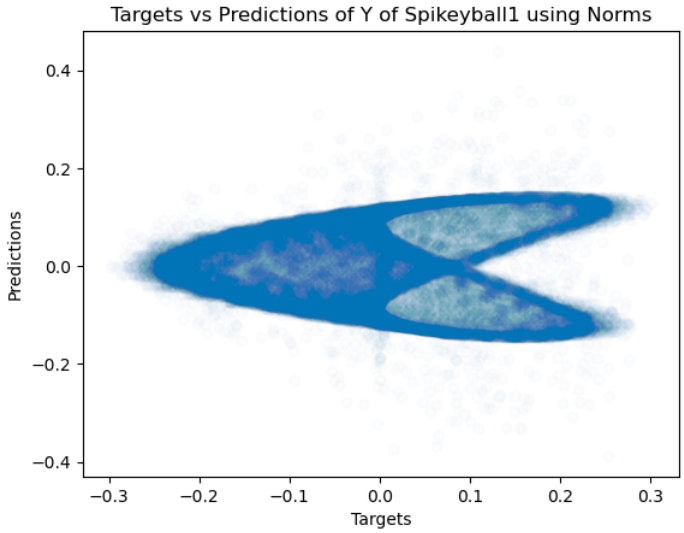

Targets vs Predictions of X of Spikeyball1 using Offset
Predictions
Targets
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20

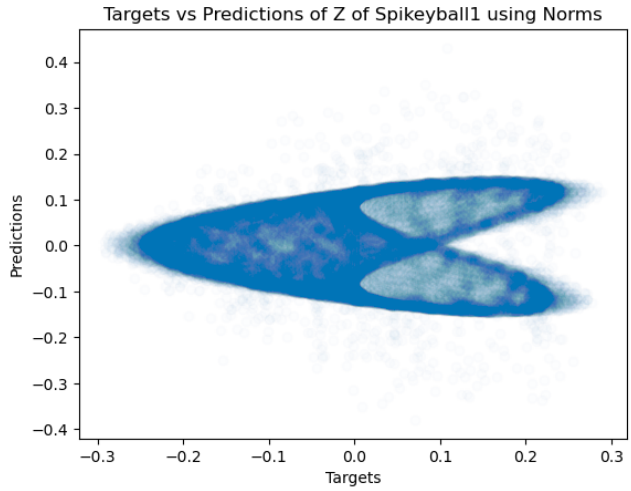

Targets vs Predictions of Y of Spikeyball1 using Offset
Predictions
Targets
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20


Targets vs Predictions of Z of Spikeyball1 using Offset
Predictions
Targets
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15

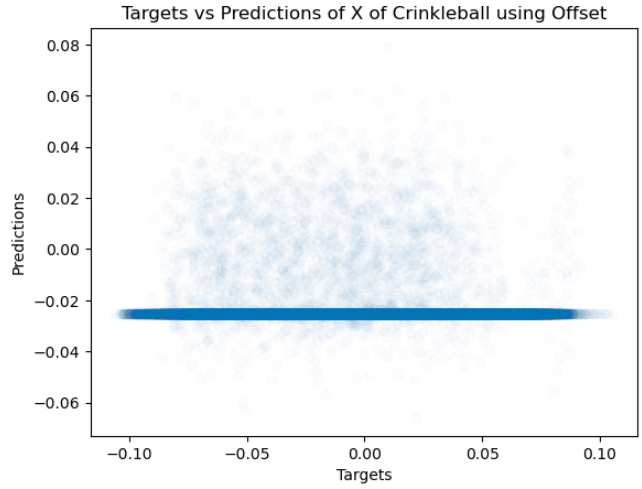
Norms:

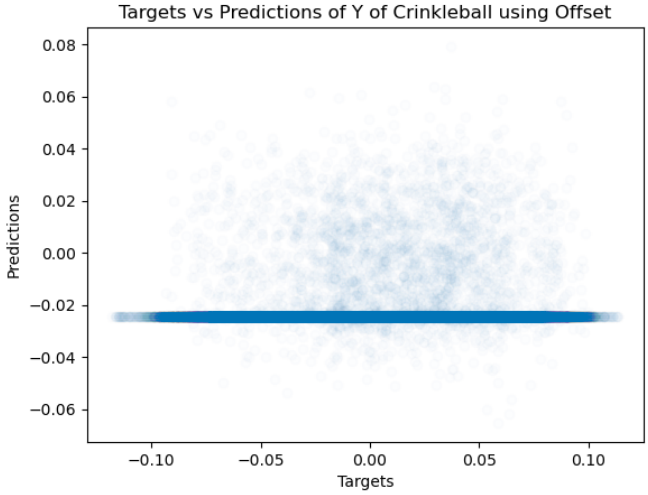

Targets vs Predictions of X of Spikeyball1 using Norms
Predictions
Targets
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
0.0
0.1
0.2
0.3
0.4
0.5
0.6

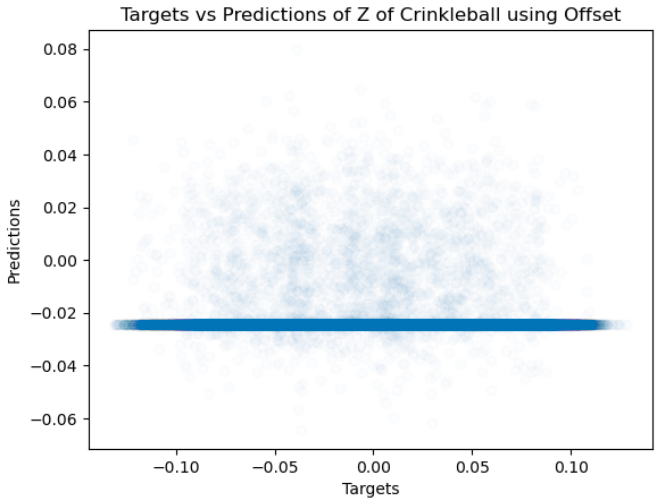

Targets vs Predictions of Y of Spikeyball1 using Norms
Predictions
Targets
0.4
0.2
0.0
−0.2
−0.4
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3


Targets vs Predictions of Z of Spikeyball1 using Norms
Predictions
Targets
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3

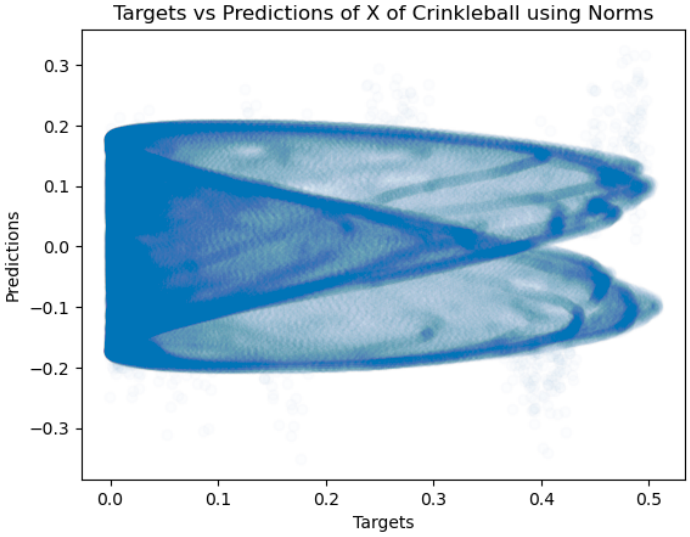
Crinkleball:
Offset:

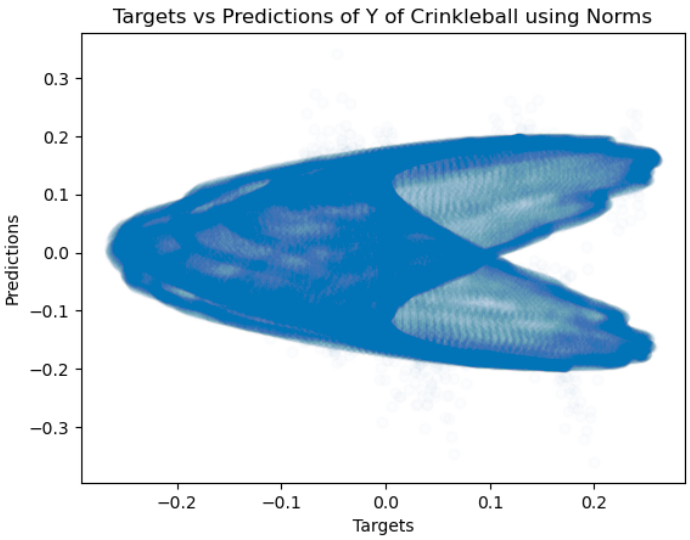


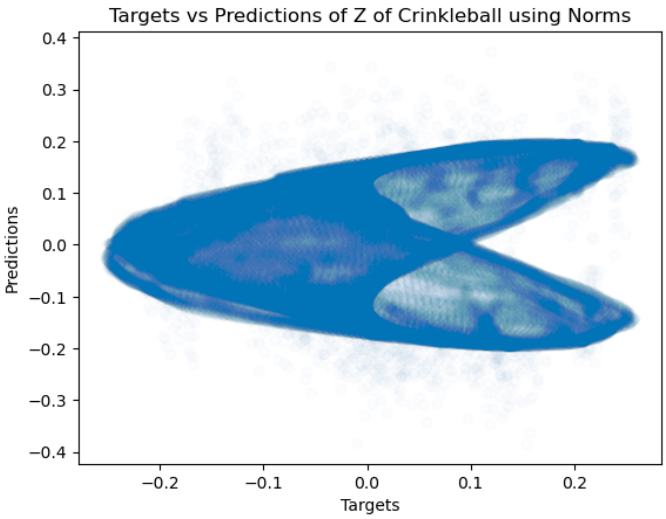



Targets vs Predictions of Z of Crinkleball using Offset
Predictions
Targets
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.10
−0.05
0.00
0.05
0.10

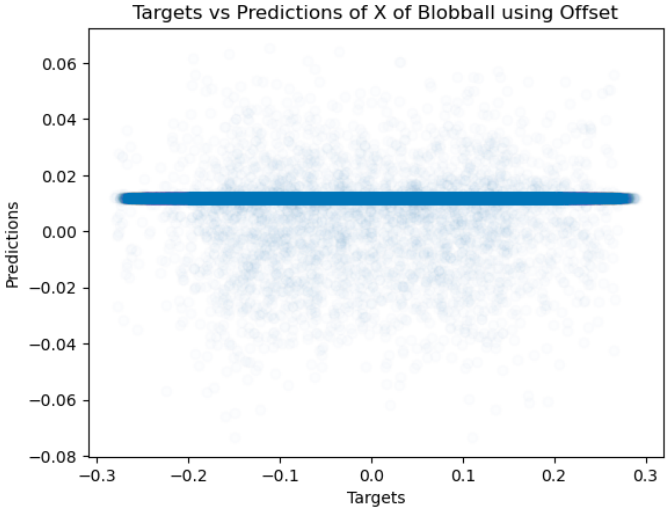
Norms:


Targets vs Predictions of X of Crinkleball using Norms
Predictions
Targets
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
0.0
0.1
0.2
0.3
0.4
0.5

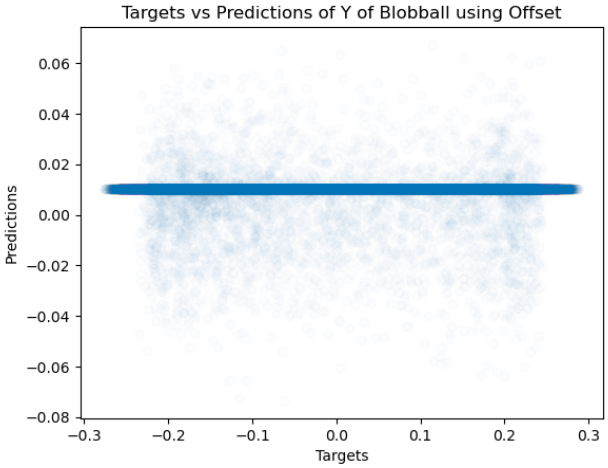

Targets vs Predictions of Y of Crinkleball using Norms
Predictions
Targets
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.2
−0.1
0.0
0.1
0.2

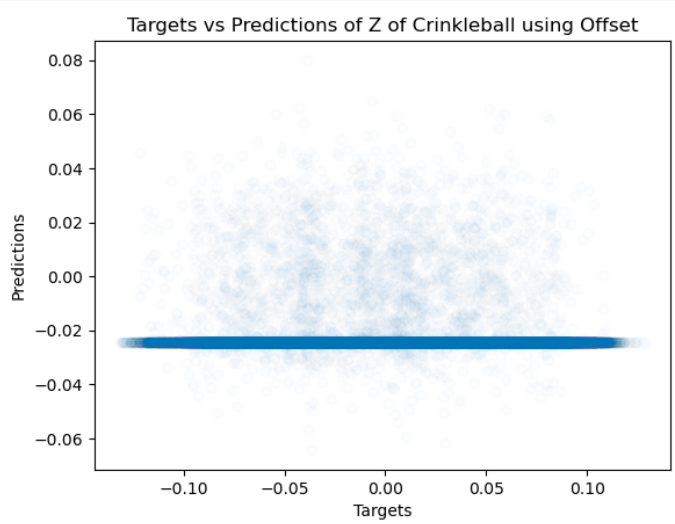


Blobball:
Offset

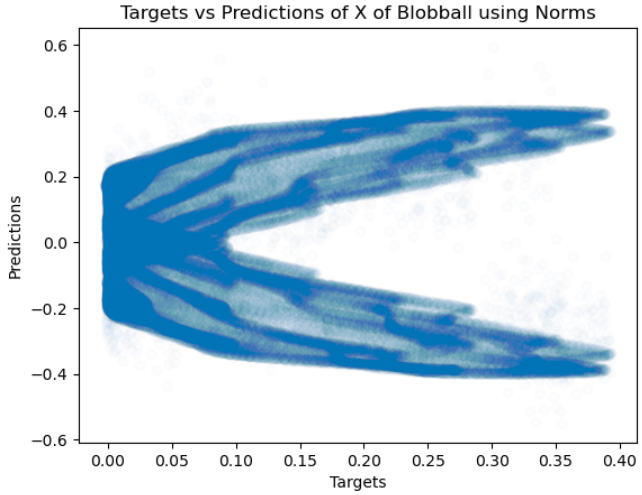


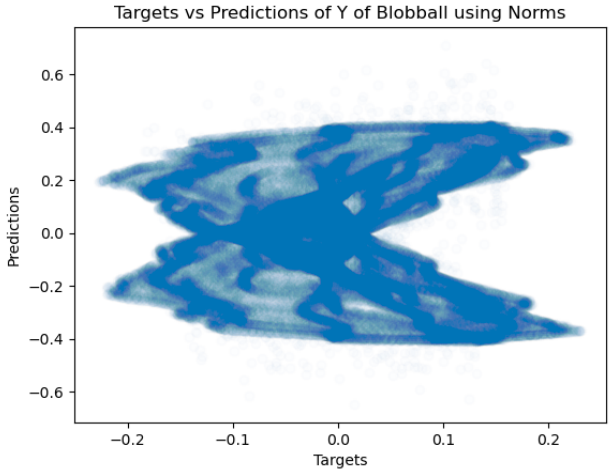
.

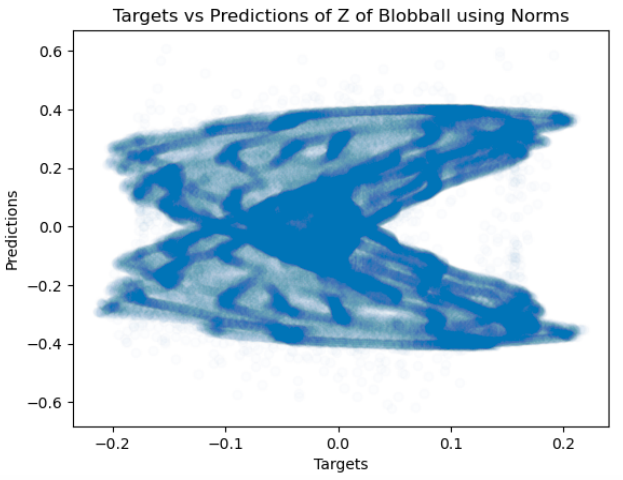



Targets vs Predictions of Z of Crinkleball using Offset
Predictions
Targets
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.10
−0.05
0.00
0.05
0.10

Norm:


Targets vs Predictions of X of Blobball using Norms
Predictions
Targets
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40


Targets vs Predictions of Y of Blobball using Norms
Predictions
Targets
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.2
−0.1
0.0
0.1
0.2



Timeset Surface Normal Dataset Experiments
This experiment utilizes the build_time_dataset to generate multiple smoothed timesets of the original mesh to feed into MLP's.

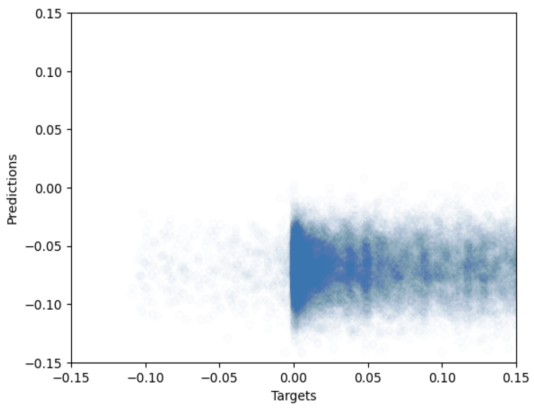
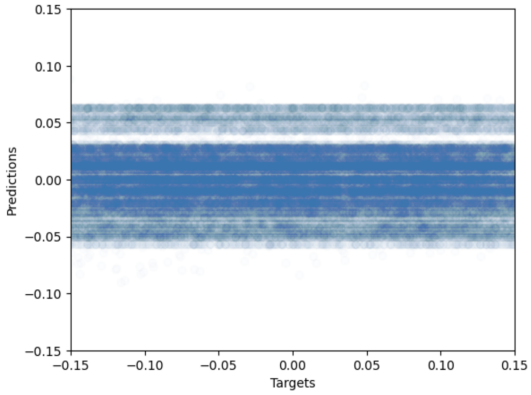
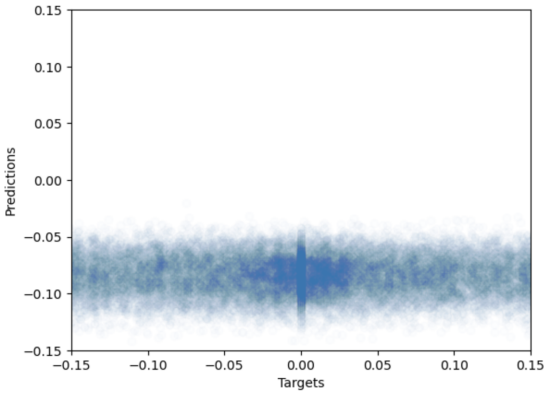
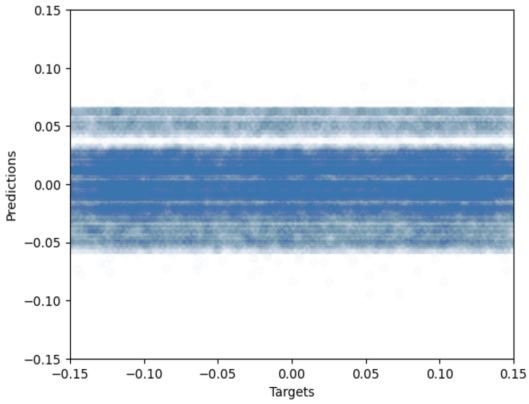
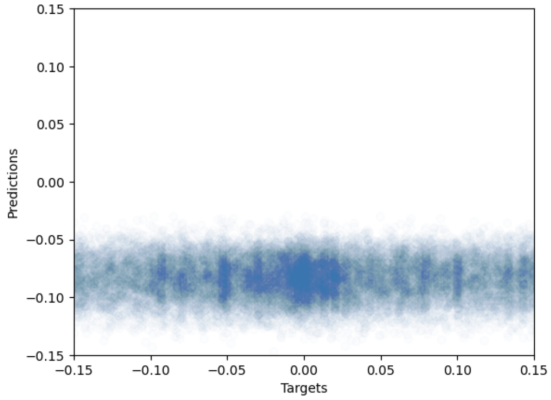
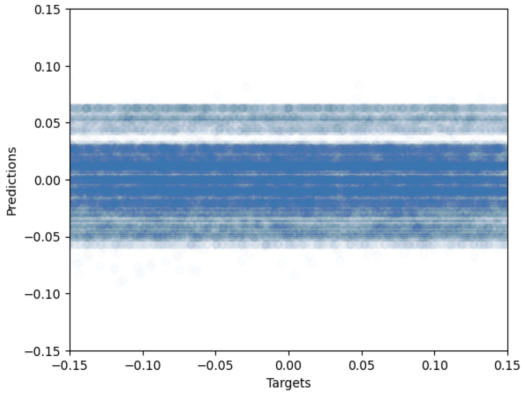
Two different meshes were used to conduct these experiments: blobball.stl and spikeball2.stl. The first step of the experiment is to create the timestep dataset. To generate the dataset, one has to plug in variables like the original mesh, the number of smoothing steps, the highest number of times the mesh is smoothed, the type of laplacian, the smoothing algorithm and the sample size.

For both meshes I've plugged in the same variables in which the 2000 iterations of the smoothing algorithm was split into 6 steps, with mesh laplacian and 0.8 sample size. The only thing that differed between both experiments were the smoothing algorithms, which in this case for blobball I've utilized laplacian smoothing and for spikeball2 I've used taubin smoothing. The reason why I used two different smoothing algorithms is because each mesh resulted in different outcomes, furthermore the corresponding smoothing algorithms worked best with the specific smoothing algorithms.

After creating the time step dataset, I loaded both datasets to the same GINR model which consisted of 512 hidden dimensions, 8 layers, utilized MSELoss, and Adam optimizer. I've ran both of the experiments for 20 epochs which gave the best result in terms of runtime and training outcome.

Once the predictions are obtained from the GINR models, in time_series_norms_exp_runner, the predictions are reshaped in order to form new points that can be generated to visualize the inference.

Though the inference visualization for both experiments look fairly similar, when plotting the predicted data to the target data we can see that there is *some* difference between the prediction accuracy of the experiments:

	Blobball Exp. Prediction vs Target	Spikeball Exp. Prediction vs Target
X axis		
Y axis		
Z axis		

Siren Activation vs No Siren Activation Experiment
This experiment focuses on the differences between the output predictions of offsets in regards to Siren Activation.

This experiment was conducted using 2 different meshes:

Spikeyball1.stl
Spikeyball2.stl

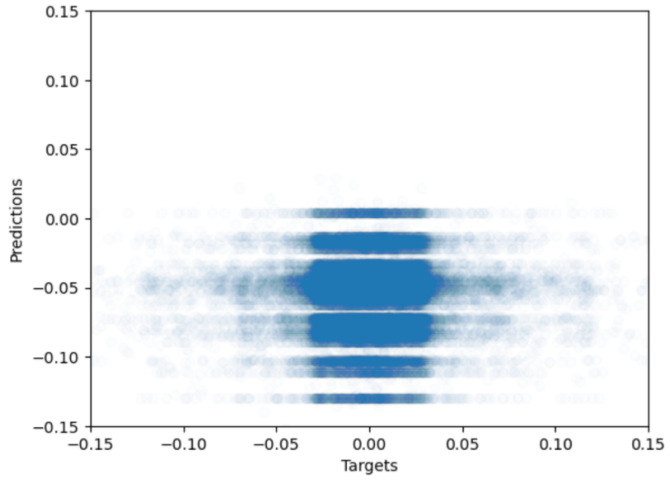
This experiment began by loading in one of the meshes and creating a MLP model using Siren Activation. Siren Activation could be turned on or off through the MLP parameters, sine and all_sine. For this experiment, both parameters were set to True. Afterwards, the model was run using 10 epochs, a MSELoss function, and an Adam optimizer. The last epoch of predictions was recorded in addition to the batch that was associated with the prediction. Finally, the predictions and the targets (from the recorded batches) were graphed on a matplotlib scatter plot.

This method was then repeated for the same mesh with Siren Activation turned off. This could be turned off by setting the parameters sine and all_sine to False on the model.

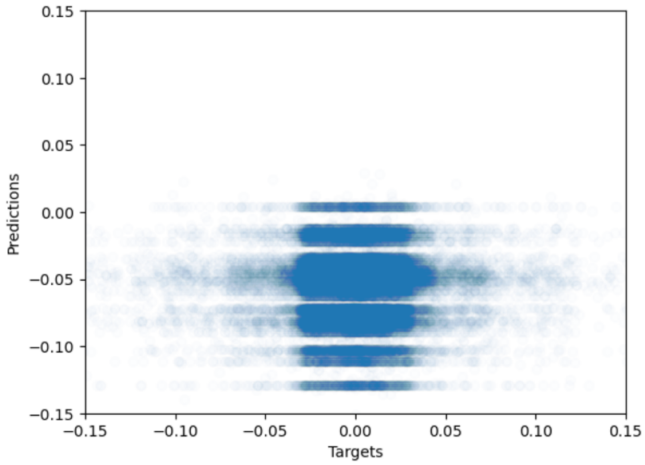
Both processes were then applied to the second mesh, Spikeball2.stl.

After completing both experiments, we were able to see the underperformance of the predictions in the models that had Siren Activation turned off. Although all graphs are difficult to interpret, we are able to see a wider spread within the graphs with Siren Activation turned on. Additionally, the model with Siren Activation turned on for the SpikeyBall2.stl mesh shows an optimal linear trend (since the predictions should match the target, an optimal graph would show a trend similar to y = x)

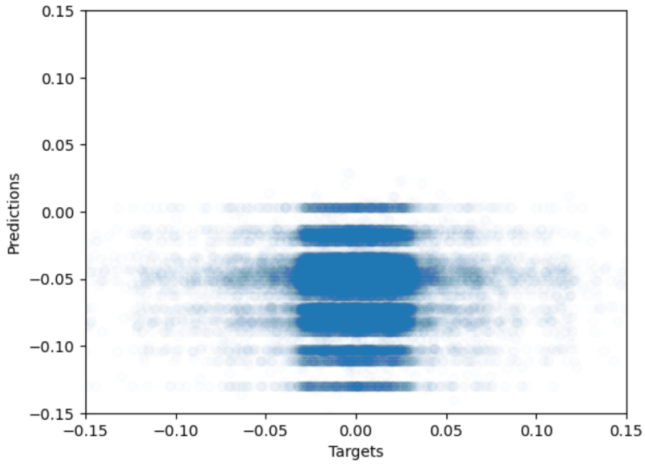
Siren Activation On (Spikeyball1.stl)
X value:



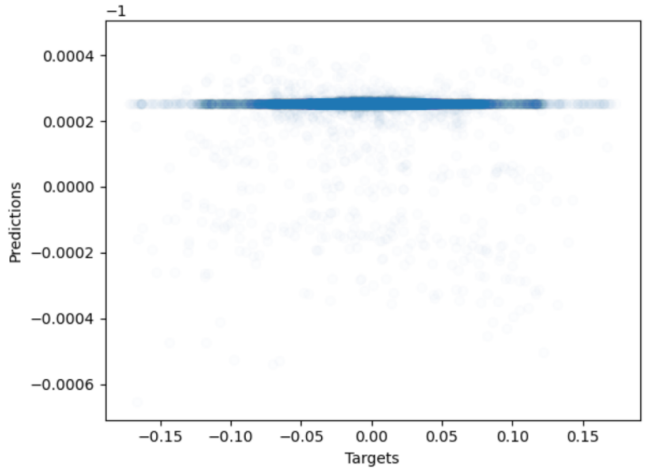
Y value:



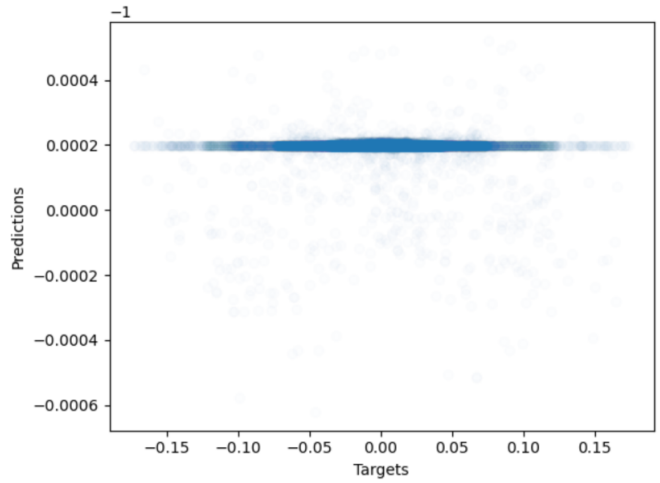
Z value:



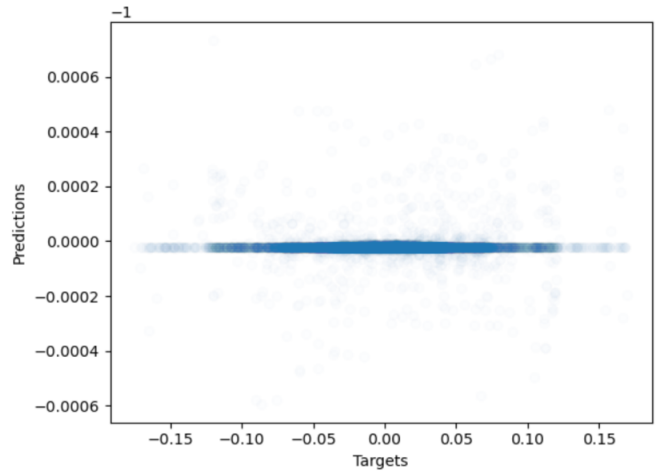
Siren Activation Off (Spikeyball1.stl)
X value:



Y value:

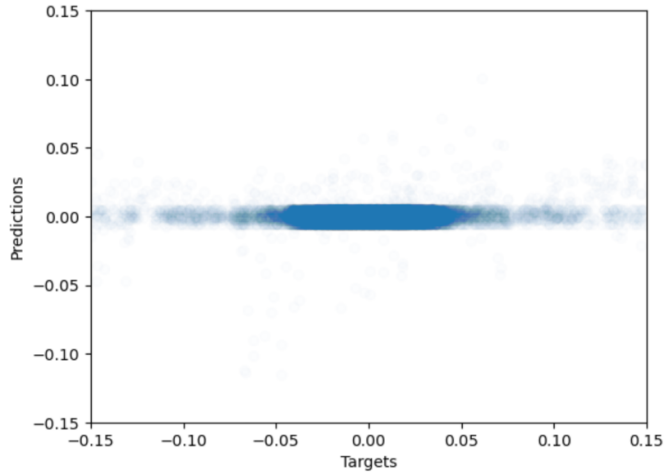


Z value:

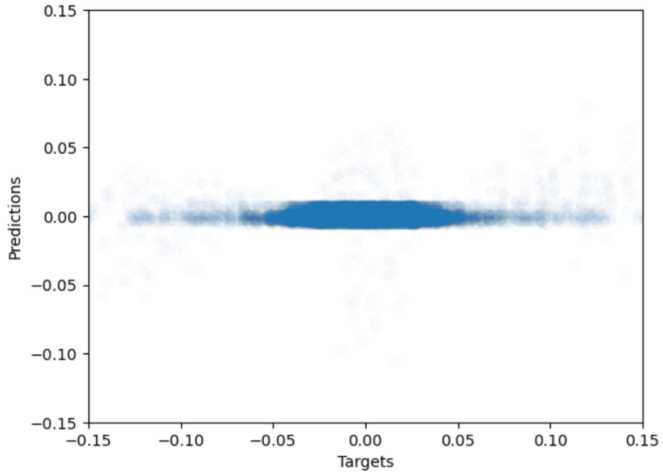


Siren Activation On (Spikeyball2.stl)

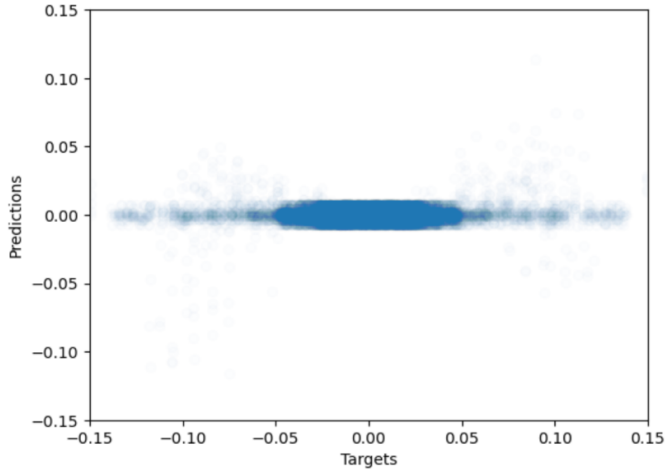
X value:



Y value:

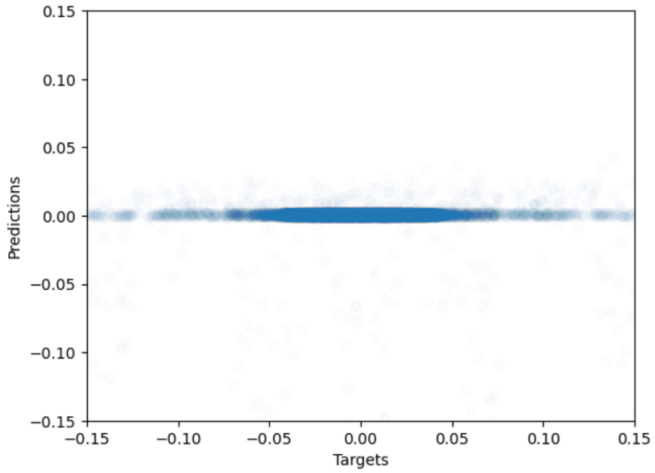


Z value:

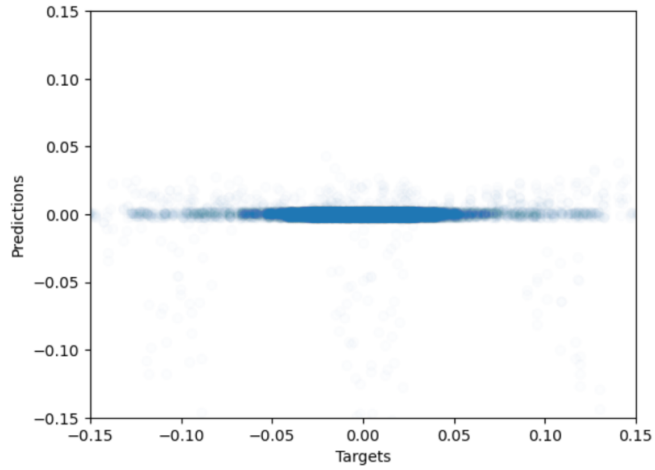


Siren Activation Off (Spikeyball2.stl)

X value:



Y value:



Z value: