CS 554 Term Project: Random Walks for Adversarial Meshes

Charles Ison*
Oregon State University

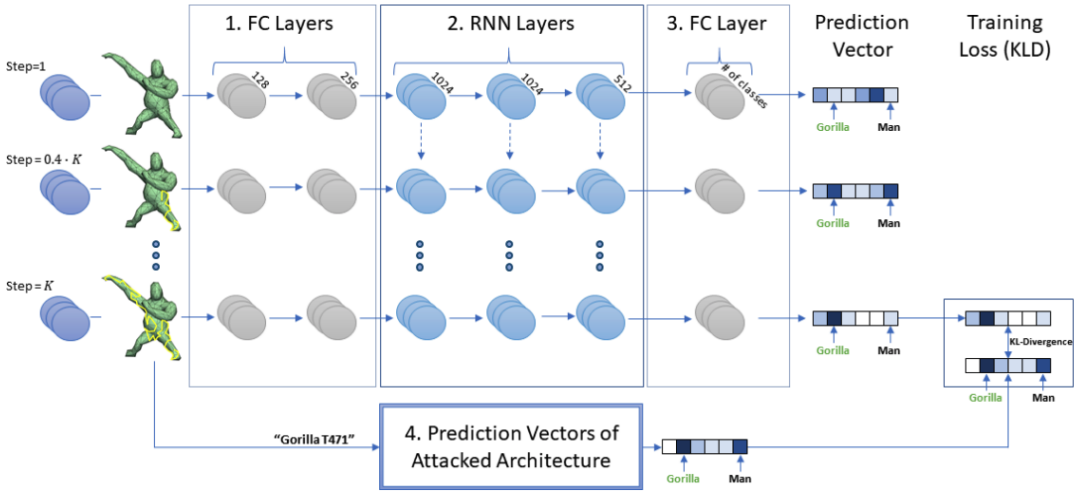

Figure 1: Architecture of attack that extracts random walk along mesh as input to imitation network for generating adversarial mesh. [Belder et al. 2022].

1 Introduction

In the 2022 paper "Random Walks for Adversarial Meshes," Belder et al. showed that adversarial attacks can be performed using minute mutations on input meshes almost imperceivable to the human eye [Belder et al. 2022]. Even though these changes to the mesh are extremely small and would not impact a human's ability to understand the mesh, they drastically reduce neural network classification accuracy. The potency and discreteness of these attacks, suggests vulnerabilities in geometric deep learning systems should be considered before deploying them in consequential fields. For my term project, I reimplemented the adversarial remeshing algorithm proposed by Belder et al. and leverage the newly generated adversarial meshes for attacks on popular mesh classification neural networks. I also reimplemented the MeshWalker classification network introduced in 2020 by Lahav and Tal [Lahav and Tal 2020]. Interestingly, I was able to achieve extremely high classification accuracy using my implementation of the MeshWalker classification network (which is crucial for building the attack meshes), but I was not able to generate adversarial meshes that achieved the results reported by Belder et al. Using my reimplementation of the adversarial remeshing algorithm, I was only able to degrade mesh classification accuracy 18.84% worse than a comparable random vertex perturbation remeshing algorithm.

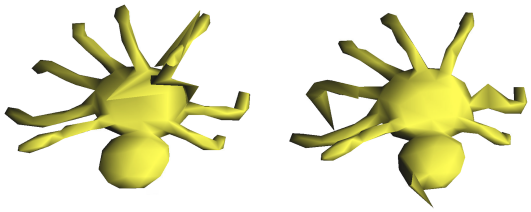

Figure 2: Example showing an adversarial spider mesh on the left that was misclassified as an alien by MeshCNN and a randomly perturbed mesh on the right that was still correctly classified as a spider by MeshCNN.

2 Previous Work

2.1 Geometric Deep Learning

Geometric deep learning refers to the application of neural models to non-Euclidean domains such as graphs and manifolds [Bronstein et al. 2016]. Many neural network architectures have been proposed for mesh classification and segmentation, frequently building off of recent advances in tangential deep learning fields such as computer vision or natural language processing (NLP). For example, one of the core components of MeshCNN [Hanocka et al. 2019] is the convolution and pooling operations, which were originally introduced in computer vision for convolutional neural networks (CNNs). Similarly, MeshWalker is built off a base recurrent neural network (RNN), which was originally introduced for sequence learning in NLP [Lahav and Tal 2020].

2.2 Adversarial Attacks

Adversarial attacks are when small perturbations, imperceivable to human eyes are applied to a machine learning model's input, causing the model to generate incorrect conclusions with high confidence [**?**]. In 2014, the Szegedy et al. first noticed that deep learning models operating on images were susceptible to misclassification if the images were updated in a manner that maximized the models error. There are many different flavors of these attacks, but

*e-mail: isonc@oregonstate.edu

the fundamental idea can be applied to any machine learning model that is trained by minimizing a loss function. As neural networks become more ubiquitous and tightly coupled with our day-to-day lives, the potential damage from an adversarial attacks will become higher. One example area with high levels of risk are self-driving vehicles.

3 Background

The attack mechanism proposed by Belder et al. offers two main contributions to the literature surrounding mesh classification neural networks. The first contribution is the proposed black-box attack mechanism which can be leveraged without any information about the internal model being exposed to a malicious actor. Furthermore, the required changes on adversarial meshes are so small, they are essentially unnoticeable to humans unless specifically highlighted. Despite these minute changes and blindness to any of the model's internal details, the remeshing algorithm was able to decrease classification accuracy from 98.6% to 16.0% for MeshWalker, 98.6% percent to 14.8% for MeshCNN, and 99.7% to 18.3% for PD-MeshNet. These drops in accuracy are greater than several recently proposed white-box attack mechanisms (where malicious attackers could have access to the internal details of the model under attack).

This leads to the second contribution from Belder et al., which is a discussion surrounding the fragileness of mesh classification algorithms to attack. Although mesh classification networks have been shown to be relatively robust to noise [Belder et al. 2022], they appear very vulnerable to targeted attacks. This is worrisome considering their continued adoption in the medical field, with workshops such as GeoMedIA existing to discuss their use for medical imaging. Ideally the results presented by Belder et al. could serve as a starting point for a discussion around how to make these systems more robust.

4 Methods

4.1 Datasets

There are two main datasets commonly used for benchmarking mesh classification algorithms. The first is SHREC11 which contains 600 meshes divided between 30 classes. The second is ModelNet40 which contains 12,311 meshes split between 40 classes. Both datasets would work as a vector for an attacks against a mesh classification model, but I opted to use SHREC11 as a starting point because there are publicly available weights published for MeshCNN that have already been pre-trained using SHREC11. This prevented me from having to train one of the mesh classification networks from the ground up, which could potentially have taken lots of compute power and time. One issue, I did run into with the SHREC11 dataset is not all of the meshes have triangular faces, which limited my ability to visualize the remeshing results using the OpenGL program we have been using for the class. One workaround I found was to use a third party program, such as MeshLab to view the results [Cignoni et al. 2008]. I also used MeshLab to convert the OBJ files used for storing the SHREC11 and ModelNet40 datasets to a PLY file format for visualizing the results in OpenGL.

4.2 Random Walk Algorithm

One of the fundamental questions for all mesh classifications networks, is what datatype will be represent the meshes as an input into the model. Lahav and Tal proposed the novel idea that meshes could be presented to the neural network as a random walk along the surface of the mesh [Lahav and Tal 2020]. This approach had two benefits, first it greatly reduced the amount of data that needed to be passed to the model and second it means the model can operate sequentially along the steps in the random walk. In order to capture a random walk along the mesh, the steps were to first pick a random vertex on the surface, then randomly select an adjacent vertex as the next step, and repeat this process while not crossing over the previous path until the number of desired steps as been achieved. Although the random walk idea is relatively trivial, there are two critical components that should not be overlooked. First, when generating a large number of random walks of sufficient length, it is possible the walk would become corned with no path forward that does not traverse over the previously walked path. In order to prevent this, a backtracking option is needed to retreat until a new open step is found. Secondly, in contrast to the name, the selection of the steps in the random walk need to be deterministic for each mesh so the classification model can learn about the shape of the mesh. Otherwise, during each training iteration, the model will see a different walk and the previously learned weights will not be able to generalize. This was a bug in one of the earlier iterations of my program that took a substantial amount of time to debug. The solution was to declare a unique seed for each mesh that could be used with the Python random number generator.

4.3 Imitation Network

The imitation network is critical for generating the adversarial mesh is an black-box manner. The goal is to train a separate model that the malicious actor controls to imitate the behavior of the mesh they would like to attack. Then the malicious actor can mutate the input mesh in a manner that maximizes the imitation network's loss. This can be done relatively easily using a modern deep learning framework such as Pytorch or TensorFlow by forcing their automatic differentiation engine to compute the gradients for the model's input. Then the most sensitive vertices can be moved against the gradient to increase the overall classification loss. Belder et al. opted to use MeshWalker as an imitation network because the random walk input's are relatively easy to extract as part of an attack and because the random walk is just a series of vertices which is easy to mutate according to the gradient ascent process described above [Belder et al. 2022]. MeshWalker is a hybrid model composed of two fully connected layers, an RNN that traverses along each step in the walk, and one final classification layer (see Figure 1. for more details) [Lahav and Tal 2020]. Due to compute limitations, I opted to reconstruct MeshWalker from the ground up with fewer dimension as part of this project. Although my implementation of MeshWalker was slightly smaller, the architecture was the same as the original and I achieved better than expected imitation performance, which will be discussed in the results.

4.4 Attacked Networks

The network I choose to attack first was MeshCNN due to it also being a SIGGRAPH paper in 2019 and the authors making the pre-trained weights for the SHREC11 dataset publicly available. Also, MeshCNN is architecturally very different than MeshWalker and relies on convolutions instead of sequential processing, so I was curious if the architectural differences would limit the effectiveness of the attack using the imitation network. Belder et al. reported equal success in their attacks against MeshCNN, MeshWalker, and PD-MeshNet, which was not what I would have originally expected. As a stretch goal, I was also hoping to attack PolyNet because some members of our group helped develop it. But due to spending large amounts of time trying to figure out discrepancies between my results and the results reported by Belder et al, I did not have time attempt a random walk adversarial attack against PolyNet.

4.5 Random Perturbations

In order to compare the potency of the adversarial meshes against a random baseline, I also implemented a remeshing algorithm where random vertices of the mesh were shifted small amounts in random directions. In order to ensure consistency between this approach and the adversarial meshes, I normalized the magnitude of the vertex coordinate shifts for both approaches to be within some prede-

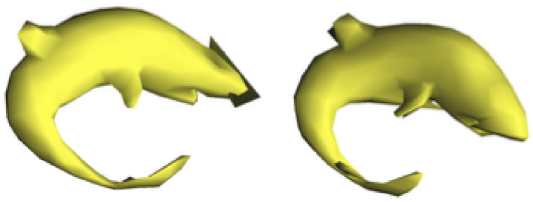

Figure 3: Example showing adversarial shark mesh on the left that was misclassified as a lamp by MeshCNN and a randomly perturbed mesh on the right that was still correctly classified as a shark by MeshCNN.

fined range (for the results presented in this paper -0.3 and 0.3 was used) and set the number of vertices changed to be consistent (for the results presented in this paper 5 vertices were changed).

5 Results

With no changes or attacks on the SHREC11 data, MeshCNN was able to achieve an classification accuracy of 99.17%. After the adversarial attack, MeshCNN classification accuracy dropped to 73.33% (a drop of 25.84%). And for comparison, after the random perturbations remeshing, the classification accuracy was 91.67% (a drop of 7.5%). Although the results align with our expectation that the adversarial attack will cause a larger decrease in accuracy than the random pertubations, based on the results reported by Belder et al. I would expect a substantially more dramatic result. They reported an 81.4% drop in accuracy after their adversarial attack on MeshCNN. Visually inspecting our results in Figure 2. and Figure 3., it is clear the changes required to get a misclassification are significantly more obvious than those by Belder et al. Figure 2 shows a spider mesh that was misclassified as an alien, but to get this misclassification the algorithm generated several evident spikes coming out of the mesh not consistent with the shape of a spider. The randomly perturbed mesh in Figure 2 is also clearly incorrect, but interestingly the classifier was able to get a correct result. Figure 3 shows similar change on a shark mesh, except this time the changes to the mesh are slightly less dramatic around the head area. Despite the implementation of adversarial meshes causing more dramatic mesh changes than expected, the changes are still less than what would be required to confuse a human.

6 Conclusions

As part of this term project and inspired by the paper from Belder et al. "Random Walks for Adversarial Meshes," I have implemented from scratch an algorithm to record random walks along the surface of meshes, a MeshWalker classification model trained to imitate the performance of other networks, an algorithm to mutate input meshes in a direction that maximizes the loss of the MeshWalker model, and finally run the new adversarial meshes with MeshCNN to document the performance. Although the decreases in performance did not align with the results reported by Belder et al., the results do show the adversarial meshes decrease accuracy more than corresponding magnitude random pertubations. For the next steps, further analysis should be done to find the discrepancy in results. I suspect there are two culprits for the misalignment of the results. First, Belder et al. used a Kullback-Leibler divergence loss function for training their imitation network, which I tried, but was not able to get to converge. Instead, I used a cross entropy loss function, which allowed my imitation network to mimic MeshCNN accurately 95% of the time, which I thought would be sufficient. Perhaps there is some logic to using Kullback-Leibler divergence loss that I am missing and I should revisit this as a next step. Furthermore, my algorithm for generating the adversarial meshes also, takes only takes one mutation step, whereas Belder et al. make small incremental changes to the mesh. This is also likely a cause for the discrepancies, but I did not discover the algorithm differences until it was too late to change. One more future research direction could also be leveraging the results here to build mesh classification models that are more robust to adversarial attacks.

References

BELDER, A., YEFET, G., BEN-ITZHAK, R., AND TAL, A. 2022. Random walks for adversarial meshes. In *ACM SIGGRAPH 2022 Conference Proceedings*, Association for Computing Machinery, New York, NY, USA, SIGGRAPH '22.

BRONSTEIN, M. M., BRUNA, J., LECUN, Y., SZLAM, A., AND VANDERGHEYNST, P. 2016. Geometric deep learning: going beyond euclidean data. *CoRR abs/1611.08097*.

CIGNONI, P., CALLIERI, M., CORSINI, M., DELLEPIANE, M., GANOVELLI, F., AND RANZUGLIA, G. 2008. MeshLab: an Open-Source Mesh Processing Tool. In *Eurographics Italian Chapter Conference*, The Eurographics Association, V. Scarano, R. D. Chiara, and U. Erra, Eds.

HANOCKA, R., HERTZ, A., FISH, N., GIRYES, R., FLEISHMAN, S., AND COHEN-OR, D. 2019. Meshcnn: A network with an edge. *ACM Trans. Graph. 38*, 4 (jul).

LAHAV, A., AND TAL, A. 2020. Meshwalker: Deep mesh understanding by random walks. *CoRR abs/2006.05353*.

MILANO, F., LOQUERCIO, A., ROSINOL, A., SCARAMUZZA, D., AND CARLONE, L. 2020. Primal-dual mesh convolutional neural networks. *CoRR abs/2010.12455*.

SZEGEDY, C., ZAREMBA, W., SUTSKEVER, I., BRUNA, J., ERHAN, D., GOODFELLOW, I. J., AND FERGUS, R. 2014. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds.

YAVARTANOO, M., HUNG, S.-H., NESHATAVAR, R., ZHANG, Y., AND LEE, K. M. 2021. Polynet: Polynomial neural network for 3d shape recognition with polyshape representation. In *2021 International Conference on 3D Vision (3DV)*, 1014–1023.

YUAN, X., HE, P., ZHU, Q., BHAT, R. R., AND LI, X. 2017. Adversarial examples: Attacks and defenses for deep learning. *CoRR abs/1712.07107*.