Towards Expressive Long-Horizon Planning in Off-Road Terrain via Diffusion Models

Conner Pulling
Robotics Institute
Carnegie Mellon University, United States
cpulling@andrew.cmu.edu

Matthew Sivaprakasam
Robotics Institute
Carnegie Mellon University United States
msivapra@andrew.cmu.edu

Nayana Suvarna
Robotics Institute
Carnegie Mellon University United States
nsuvarna@andrew.cmu.edu

Abstract:
Navigation in off-road environments is challenging due to their uneven and high-risk terrains. It's vital to reason over these complex terrains to avoid unsafe or unrecoverable behaviors. We present diffusion model based approach that conditions on local costmaps to generate long-horizon plans for off-road driving. We tested our approach using both pure gaussian noise as well as smart seeding and observe the differences in the outputted trajectories. We evaluate our method through measuring the average loss over the training period and observed that the random-seeding approach achieves convergence and can diffuse viable trajectories. However, the smart-seeding approach seems to not converge or perform well, likely due to a implementation error or issues with the reverse diffusion. Our code can be found at https://github.com/Matthewjsiv/path_diffusion

Keywords: Diffusion, Robots, Learning, Planning

1 Introduction

Navigation in off-road terrains introduce many challenges including untraversable regions such as ditches and mud as well as slopes and height irregularities. Failing to address these challenges may result in unsafe rides for passengers as well as potential damage to the vehicle. Thus, it's imperative to develop methods for generating traversable paths in these challenging terrains.

A common approach for autonomous off-road navigation is to use sampling-based planners to efficiently cover a wide set of possible trajectories and find the optimal path. For example our platform, a Yamaha all-terrain vehicle equipped with a suite of sensors, runs MPPI as a local planner using costmaps generated from inverse reinforcement learning to traverse environments without collision. This involves sampling random action sequences, feeding them through a kinematic bicycle model to get state sequences, and querying the state sequences on the costmap to obtain the best sequence. Due to its sequential nature, this process is computationally intensive a limited planning horizon must be used in order to run in real-time. This can sometimes lead to unideal behaviors such as getting stuck in cul-de-sacs and not handling turns comfortably.

Dense sampling is important in this paradigm in order to find the most-optimal path, but it is difficult to perform dense sampling over long-horizons using this method. Moreover, in precarious areas such as narrow trails or dense vegetation, a lot of computation is wasted rolling out actions into these areas since they are high cost. Given that our costmaps are learned from human demonstration and are of good quality, we propose use them offline to generate long-horizon trajectories as labels for a

7th Robot Learning (2023), Pittsburgh, USA.

diffusion model, train the model using the same general process as learning from demonstration, and then predict potential paths online.

This results in the following contributions:

1. We introduce the use of costmaps as a new modality on which diffusion models can be conditioned on.
2. We propose a way to generate large amounts of valid "demonstrations" by utilizing learned costmaps instead of human demonstrations directly.
3. We introduce a new way of seeding the initialization for diffusion to encourage multiple distributions while reducing the number of necessary diffusion steps.

Figure 1: Autonomous all-terrain vehicle used for data collection

2 Related Work

Utilizing diffusion models for trajectory planning has been explored by many groups. There is a main work by Janner et al. that introduces the idea of using diffusion models for trajectory planning [1]. While they use it for manipulation, we apply it in the context of off-road driving. In [2], diffusion models are used to predict the motions of other cars in an urban scenario, conditioned on an embedding that takes in scene information such as other car locations and lanes. We instead form a similar model but for off-road terrain and for planning motions of the ego vehicle, conditioned on costmaps rather than structured embeddings.

Sridhar et al. have made significant progress in using diffusion policies to propose multi-modal path distributions [3]. By incorporating a goal mask, they are able to produce both goal-conditioned and not goal-conditioned policies. We draw inspiration from their work, applying some of their ideas in the more complex context of off-road driving instead of simpler indoor scenarios.

3 Methodology

3.1 High-Level Idea

To achieve longer time horizons during planning without incurring the time cost of longer MPPI rollouts, a diffusion models are utilized. As seen in Figure 2, there are two main processes in the model: forward diffusion and reverse diffusion. In the forward diffusion process, a dataset of top-K optimized trajectories are created using cost maps obtained using an existing inverse reinforcement learning algorithm [4] and the TartanDrive dataset [5]. Using these cost maps, an existing trajectory library will generate a set of top-K trajectories that are N points long. To create the input, only the first S points are kept while the last $N - S$ points will create a straight line to the goal. We then add noise to the straight line to create a "noisy" input. Each denoising step is conditioned on a context vector of the cost map, produced using an encoder. The purpose of conditioning on the cost

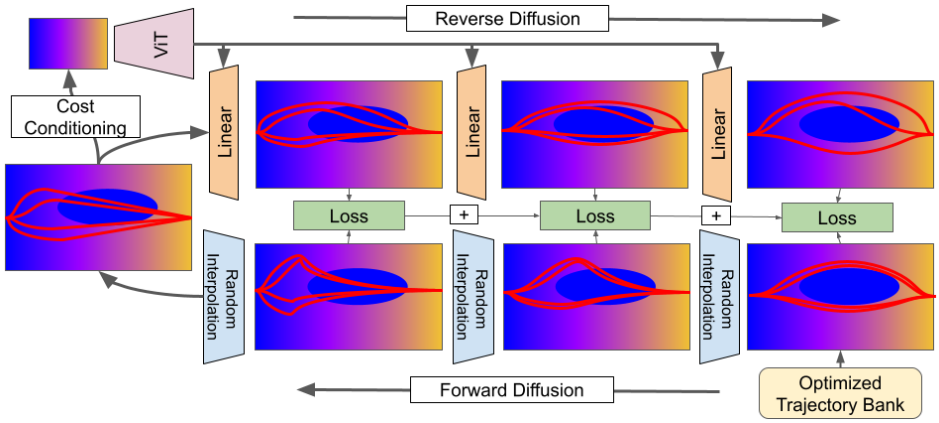

Figure 2: Overview of Training Process

map conditioning is to give the model a global representation of the cost map that can be used at each refinement step. Overall, this process teaches the model to refine long time-horizon trajectories while being goal-agnostic and capable of predicting multiple distributions of path proposals.

Intuitively, these training processes and architecture decisions detailed below make sense as they treat the trajectory refinement problem somewhat like an optimization problem. The intermediate results of an optimization problem could be considered "noisy" because they are inaccurate. The diffusion model process explicitly guides the network to learn what classical methods use heuristics to refine.

3.2 Trajectory Library Generation

For simplicity, rather than performing system-identification of the ATVs motion and using a bicycle model to generate kinematically feasible trajectories, we generate curvature-continuous cubic splines that roughly match the capabilities of the vehicle. We generate these trajectories of length approximately equal to the range of the costmap (30m in this experiment), with the following parameters:

1. Longitudinal separation between layers: 5 meters
2. Approximate separation between adjacent nodes along an arc: 1 meter
3. Angle with respect to previous layer: 20°
4. Number of concentric arcs: 6

The trajectories are then discretized at the same resolution as the costmaps (.5m), and their XY coordinates are used as labels for training.

3.3 Model Architecture

Our model architecture follows a similar approach to NoMaD [3], with the added simplicity of the fact that we use a single costmap instead of a sequence. The one-channel costmap is fed into an EfficientNet [6], resulting in a 256-dimensional feature vector. In training, a random diffusion timestep is also chosen, and an additional 256-dimensional vector is generated by feeding the timestep through a positional embedding and two small linear layers.

These two vectors are concatenated to form a 512-dimensional vector that the diffusion model is conditioned on. A 1-D U-Net takes in the conditional vector as well as a noised sample and is tasked with predicting the noise by attending to the conditioning vector.

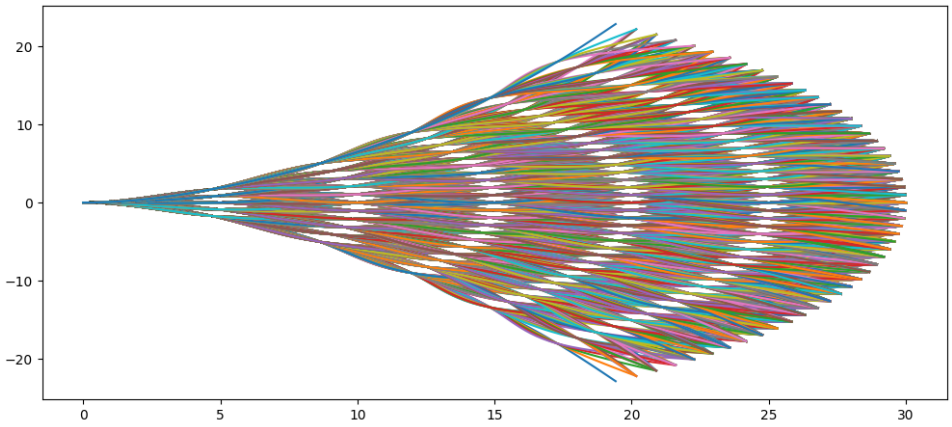

Figure 3: Trajectory library used to generate demonstrations

3.4 Training

3.4.1 Label Generation

For our labels, we wanted to generate the top k trajectories given a costmap. We began by splitting the trajectories into three separate bins: upper, middle, and low. For each of these bins, we then ordered the trajectories according to cost in increasing order. The cost for each trajectory was computed by summing up the costs from the costmap for each point along the trajectory. We then took the top $\frac{k}{3}$ trajectories for each bin to form our labels. We used binning for our labels to hopefully extract multiple distribution trajectories as seen in the upper image in Figure 4. When simply taking the top k trajectories from the entire trajectories, we found that the top trajectories would cluster around the same distribution as seen in the lower image of Figure 4.

3.4.2 Pipeline

At training, first the costmap is fed through the EfficientNet encoder to generate the conditioning vector. Then, based on a random timestep, a portion of noise is added to the ground-truth trajectories. The U-Net is then responsible for predicting this noise, and a simple MSE loss is used to train the model. We also added an additional cost, where resultant denoised predictions that go through high-cost regions are penalized. However, so far we haven't observed any improvement from this additional loss.

After each denoising step, the forward diffusion process will produce a number of intermediate denoising results that will be used in the loss function with the intermediate noising results from the forward diffusion process. This allows us to, at runtime, generate a sequence of timesteps (instead of random at training), to incrementally generate a sample with less noise until we converge on the desired result.

3.4.3 Vanilla Training

Our initial experiment involved using pure Gaussian noise as the initial noised sample. At training, the ground-truth paths are normalized to be between 0-1 to match the range of the noise. At the end of the diffusion process we scale the result back based on the original input. An example of the initial noised samples and the resulting diffused trajectories are shown below.

3.4.4 Smart-seed Training

Considering the ideal case where there are no obstacles, differences in traversability, or objectives to minimize other than distance - to get from point A to point B would be a straight line. However, in the real world, planners are employed to deal with these additional complexities. So the naive

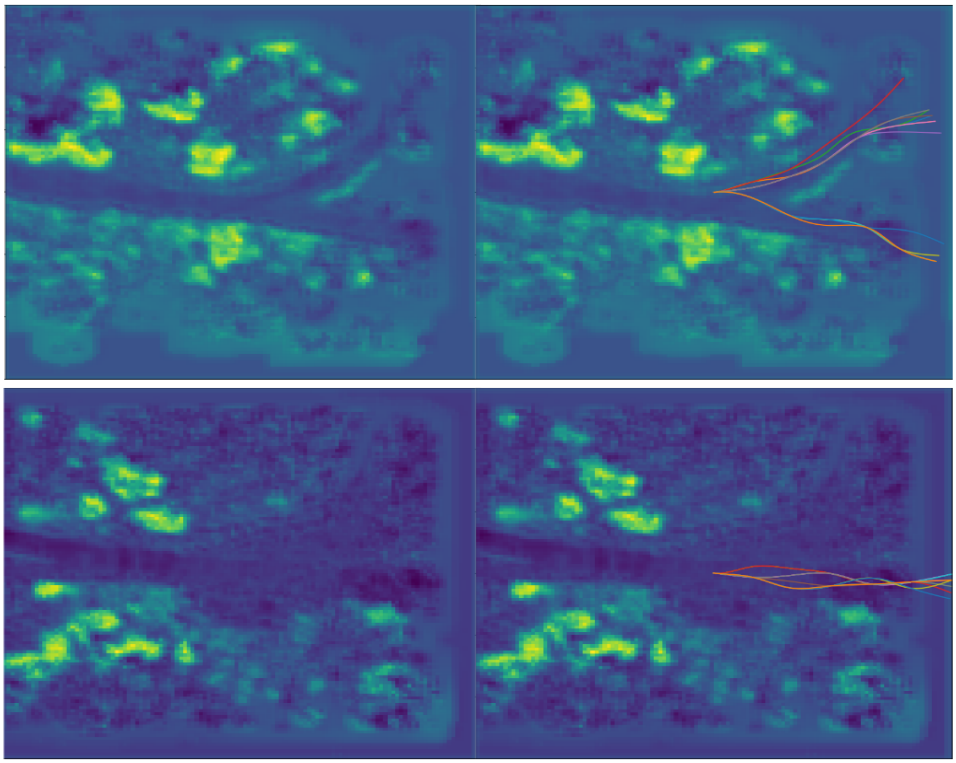

Figure 4: Examples of top-K trajectories on different costmaps. Note the multiple distributions the paths take in the top example. Brighter areas in the costmap have higher cost.

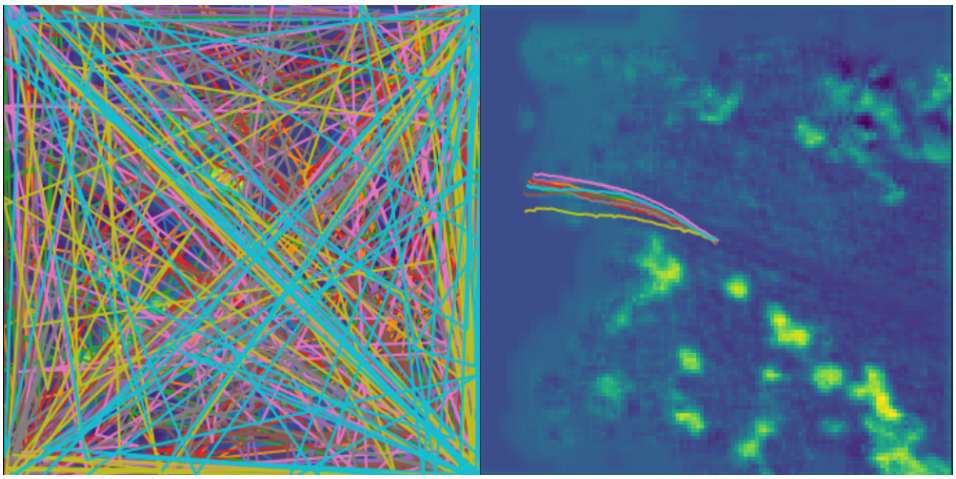

Figure 5: For our initial experiment, we start the diffusion process directly from pure guassian noise as the initial trajectory.

planning solution is a straight line between a starting point and a goal point. Therefore, we can generate a ”noisy” label by progressively smoothing the trajectory to be a straight line. Consider Figure 6,

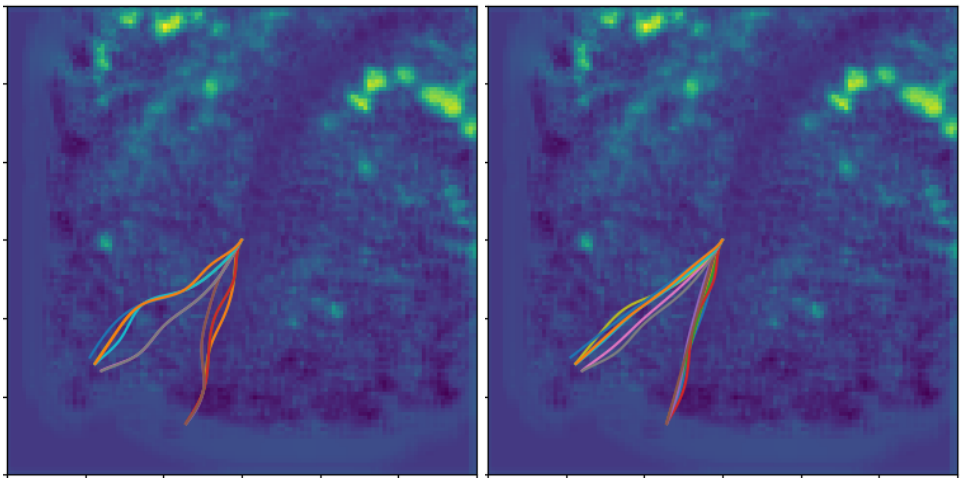

Figure 6: (Left) Original Labels. (Right) Noisy Labels after ”straightening” the original label.

A diffusion model that learns to denoise from this naive solution to the true solution will let the user of the model input a straight-line trajectory with an intentional goal and starting point rather than a random-noise trajectory.

4 Experimental Results

4.1 Training Results

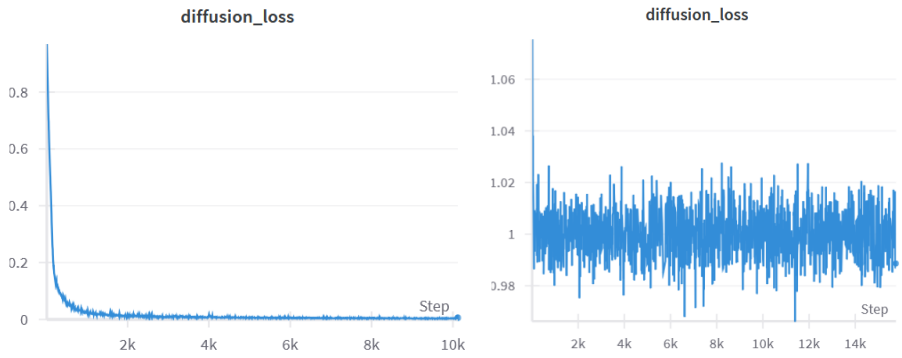

Figure 7: (Right) ”Random-Seeding” Diffusion Loss Results. (Left) ”Smart-Seeding” Diffusion Loss Results. Notice the difference in the y-axis scaling, which shows that the ”Smart-Seeding” loss did not decrease.

Overall, the random-seeding approach trained well and the loss value decreased substantially while the smart-seeding approach did not train well, likely due to a current implementation error. Additionally, both approaches plateaued after a large number of training steps, indicating that both models have finished training. However the random-seeding approach plateaus to a much more stable value, which is an indication of good performance and a correct implementation. On the other hand, the ”smart-seeding” approach has a much more noisy convergence, indicating that there might be something wrong with the loss function or diffusion process.

4.2 Random Seeding Qualitative Results and Analysis

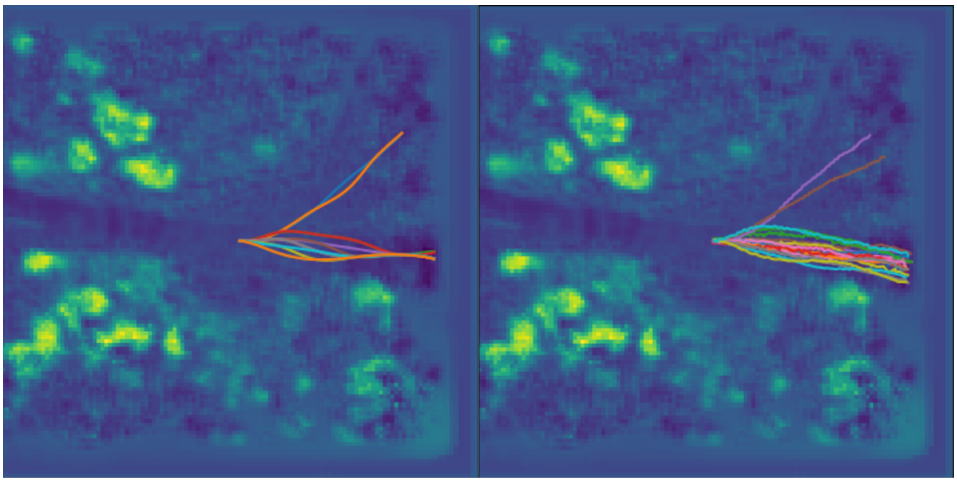

Figure 8: Our model is able to correctly identify scenarios in which multiple path distributions should be considered.

With random seeding, the model is correctly able to diffuse multimodal paths from random noise while conditioned on a cost map. However, as random noise is used, the goal position cannot be specified to the diffusion model.

4.3 Smart Seeding Qualitative Results and Analysis

Currently, the smart-seeded model does not train correctly. The smart-seeded model takes in straight-line trajectories with random end points and currently diffuses points that erronously are centered around the start point. It is likely this behavior is due to an incorrect formulation of the reverse diffusion process, which is different from normal diffusion because smart-seeding is structured noise rather than Gaussian. Consider Figure 9,

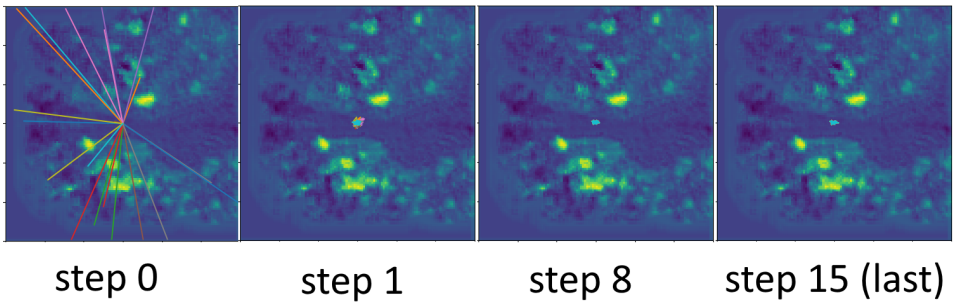

Figure 9: The current smartseed model erroneously diffuses to points centered around the start point.

which shows that the even after the first diffusion step, the trajectories are drastically pulled towards the (0,0) start point. In designing this novel noising approach, we found that while designing a forward diffusion process is straightforward, formulating a reverse diffusion process is not as such. Rather than ”guessing” what the previous sample could be using the known Gaussian distribution, the reverse diffusion process has to ”guess” an unknown original curve from a straightened curve the used an unknown interpolation factor.

5 Conclusion

5.1 Summary

In summary, as an alternative to expensive planners, this project aimed to design a diffusion model that can be conditioned on a cost map and produce viable trajectories. This diffusion model would act as an intermediate step to ”fill in the gaps” so the expensive planner would not need to be called as often. In this project, two diffusion processes were tried, one based on prior work and another that was novel to the problem. The first approach used random noise applied to expert trajectories in a standard diffusion process such that the model learned to ”denoise” random noise into valid trajectories. This approach worked well, producing viable trajectories compared with the generated labels. However, because the noisy trajectory is random, the end points could not be specified.

The second approach was to try our ”smart-seeding” approach, where each forward diffusion step interpolated between the expert trajectory and a straight line between the end point and the start point. The idea was that the model would learn to diffuse from an arbitrary straight line trajectory to a valid trajectory. However, there are currently bugs in the implementation of this approach. Future work will focus on exploring why this approach failed in detail.

5.2 Future Directions

There are a number of future directions we would like to try. One idea is to try predicting the entire top-K given a costmap, rather than one at a time, by replacing the 1D U-Net with a 2D U-Net. This would hopefully allow the model to learn the relationships between the paths its proposing in order to encourage more expressive paths. However, it is possible that this goes against the standard paradigm in diffusion where you can feed an infinite number of noised samples as a batch into the model. We would also like to increase the range of our predictions. While 30 meters is an acceptable range for planning in our case, increasing to 60 meters would have a larger impact. There are also a number of minor ablations we haven’t performed yet, such as tuning the cost threshold for ground-truth trajectory generation, and experimenting with the necessary number of diffusion steps. Finally, we would like to try predicting trajectories in an action-space instead of the waypoint-space that we use for this work. This would allow the vehicle to actuate directly on the predictions of the model rather than needing to figure out what action corresponds to the predicted waypoints.

References

[1] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis, 2022.

[2] C. M. Jiang, A. Cornman, C. Park, B. Sapp, Y. Zhou, and D. Anguelov. Motiondiffuser: Controllable multi-agent motion prediction using diffusion, 2023.

[3] A. Sridhar, D. Shah, C. Glossop, and S. Levine. Nomad: Goal masked diffusion policies for navigation and exploration, 2023.

[4] S. Triest, M. G. Castro, P. Maheshwari, M. Sivaprakasam, W. Wang, and S. Scherer. Learning risk-aware costmaps via inverse reinforcement learning for off-road navigation, 2023.

[5] S. Triest, M. Sivaprakasam, S. J. Wang, W. Wang, A. M. Johnson, and S. Scherer. Tartandrive: A large-scale dataset for learning off-road dynamics models, 2022.

[6] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.