A Style Transfer Approach To Appearance Agnostic Agents

Sreeganesh Valathara Rajendran
vrs@bu.edu

Department of Electrical and Computer Engineering
College of Engineering, Boston University

Abstract

Sim2Real is an approach in robotics where an agent is trained in simulation and deployed in reality. In this project, we investigate Neural Style Transfer as a domain randomization approach to Sim2Real. We present the effectiveness of Neural Style Transfer and how it fares against other SOTA approaches used for Sim2Real.

Task

In the context of reinforcement learning, a policy is a function that maps from state-space to action space. Through a dataset, we provide state-action pairs and train the agent to learn a function which maps these states to the corresponding actions. By doing so, we expect the function to generalize to unseen states. This approach works provided the states in the dataset form a significant, and ergodically distributed, subset of the state-space. In some contexts, the state-space is so large, that virtually no dataset can span it. An example is any context that takes in images, of non-trivial dimensions, as inputs. The number of possible images that a matrix of its shape can instantiate will easily dwarf the size of any dataset. Thus when we train a model with a dataset, the model can instantiate any function whos state-action pairs is a super-set of those in the dataset. In this large set of functions, we need to make sure that the instantiated function has state-action mappings similar to that of a rational human's. This is what we mean by the ability of a model to "generalize" to new inputs.

Sim2Real is an approach, commonly used in robotics, where agents are trained in simulation and deployed in real environments. There are a number of benefits to this approach:

- Obtain large amounts of data since we control the creation.
- Obtain favourable characteristics, such as balanced classes, since we have full control over its creation.
- Safety during training. Agents tend to make mistakes when they learn policies. In certain contexts such as autonomous driving, this poses a non-trivial amount of danger. Thus, training in simulation is highly preferred.

Despite the promises of Sim2Real, one of the challenges faced by this paradigm is, what is known as, "domain gap". Domain gap refers to the challenges that arise due to the difference in input characteristics of that obtained from simulation vs reality. Within domain-gap is, "appearance gap", and this is will be our objective in this particular project.

Appearance gap results in the simulation trained agent not being able to perform in a real environment. This occurs due to the difference in the "states" obtained from simulation compared to those obtained from a real environment. Even though the human mind finds the task of driving in a game vs that in real road rather trivial, that is not the case with models. As far as artificial agents are concerned, the input from a video-game is a different state-input compared to that obtained from the real-world, even if the frame obtained from the video game is the exact same scene or point-of-view. Thus, this means that when we train a model, we must train it in such a way that it "looks through" the textures and colours to see the underlying structure, which will inform the decision making part to produce the appropriate action. This is achieved by preventing the formation of non-causal correlations. One way to implement this is by randomizing those characteristics which is not important for the task. This approach is called domain randomization and the approach we're presenting here is a domain randomization approach for the task of Sim2Real.

Our aim is to enable this generalizability through a neural-style-transfer approach, "NST", for short. NST [1] is a technique from deep image processing where an image is created by combining the style of one image, called, "style image" and the content of another, named, "reference image". This technique preserves the "structure" information of the content image while allowing the colour and texture to change. This particular property of NST is the rea-

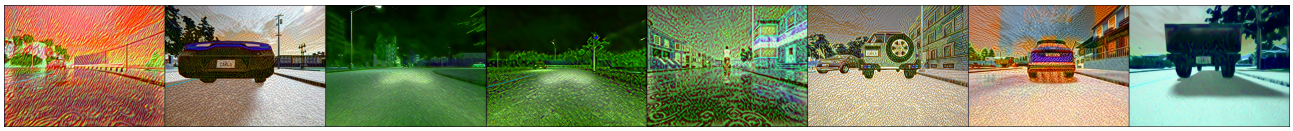

Figure 1: *Style Transferred Inputs*

son we investigate this approach because this allows us to prevent the formation of non-causal correlations the model might build.

An example within the context is when the model is trained from dataset obtained from CARLA[2]. In CARLA, all the roads are black, which means that the deep learning model might associate the colour, black, to be a defining property of a road. This was observed when the CARLA-trained baseline agent, deployed onto an RC car platform, refused to run on a red-brick paved road ran perfectly fine on a black road.

In this project, we investigate the validity of the hypothesis by training an autonomous agent using neural style transferred RGB images and testing its performance. In addition, we also set up some additional baselines that resemble similar Sim2Real SOTA approaches to see how they fare against our method.

Related Work

There are several approaches used to close the appearance gap. Domain randomization is one of the techniques employed in the field of computer vision and robotics. It involves randomizing aspects of the input which is inconsequential to the task at hand, thereby making the model more robust to variations and uncertainties in the real world. Here, we present some approaches that share our approach and contexts.

Tobin et al. employed domain adaptation for object localization tasks [3]. They demonstrated the feasibility of training a real-world object detector with an accuracy of 1.5 cm, robust to distractors and partial occlusions, using data solely from a simulator featuring non-realistic random textures [3]. The authors introduced random texture changes to objects utilizing the MuJoCo Physics Engine [4]. Employing a VGG-16 [5] architecture as the detector, they trained it to minimize the L2 loss between the network's estimated object positions and the true object positions [3]. The outcome was an object detector trained exclusively in simulation, yet achieving sufficiently high accuracy in the real world to perform grasping in clutter [3].

A flaw in this approach is that it takes a non-trivial amount of programming and effort to create datasets similar to test data and manually identifying and randomizing non-causal properties. Having an easier method would speed up the process.

Another approach employed segmentation as an intermediary between the real world and simulation. So et al. adopted this approach by converting randomized simulated RGB images into segmentation masks [6]. Their method comprises two primary components. Firstly, they trained a segmentation model that translates randomized RGB images from simulation into a shared canonical form, defined as a semantic segmentation map. Secondly, within this canonical representation, they trained a goal-reaching policy to navigate in off-terrain environments.

They begin by generating multiple simulation environments utilizing the Unity engine. The Sim2Seg model transforms the randomized image domains into their selected canonical form, a segmentation map comprising six classes: trees/bushes, ground, sky, rocks, road, and logs. The selection of these classes is aimed at supporting function. They gather 200k pairs of RGB images, segmentation maps, and depth data for each environment, employing a random policy. Domain randomization is applied to object textures, lighting color and direction, camera field of view, and camera position.

Utilizing Sim2Seg, they train a short-horizon navigation policy through reinforcement learning (RL). More specifically, they focus on a goal-conditioned policy that is dependent on visual and odometry data. The resulting model performed accurately and reliably in a real environment despite not having trained with that data.

However, one of the downside of this approach is that this work brings in an additional stage to the perception-to-action pipeline. An increase in the length of a sequential pipeline results in increased latency, which is detrimental in real-time contexts such as autonomous driving.

One specific approach akin to our work, albeit in a distinct context, is the study conducted by Loquercio et al. [7]. They integrated the capabilities of a state-of-the-art planning and control system with the perceptual awareness of a CNN, enhanced to disregard appearance through domain adaptation [7]. The drone's perception system, designed by them, employs a CNN to forecast a goal direction in local image coordinates and a desired navigation speed

based on a single image captured by a forward-facing camera [7]. The perception system undergoes training with imitation learning, utilizing automatically generated globally optimal trajectories as a source of optimization [7].

To enhance the generalization ability and robustness of their perception system, they used the concept of domain randomization [7]. The domain randomization was closely supervised, involving the randomization of factors to which the system must exhibit invariance, such as illumination, view-point, gate appearance, and background [7]. They demonstrate that the performance improves by up to 300% in simulation and up to 36% in real-world experiments [7]. However, just like the work presented by Tobin et al [3], this too requires non-trivial amount of programming for dataset creation.

Thus, we see how this approach of domain randomization has proven to work in different domains and contexts. In this project, we present a domain randomization approach that significantly reduces the amount of work required to help the model generalize to out-of-distribution (OOD) test data.

Background

Neural Style Transfer

Neural Style Transfer [1], a technique in computer vision and deep learning, merges the content of one image with the artistic "style" of another. The process involves two images: the content image, representing the desired content to be preserved in the final result, and the style image, providing the artistic style intended for the content image. The stylistic elements, including textures, colors, and patterns, are extracted from the style image to apply to the content image.

To capture the style of an input image, they leverage a feature space initially designed for texture information [1]. This feature space is constructed based on the filter responses across the spatial extent of the feature maps [1]. By incorporating the feature correlations from multiple layers, a stationary, multiscale representation of the input image is obtained, capturing its texture information while excluding the global arrangement. This separation is crucial because style, generally being global, necessitates distinct handling. The key insight from the paper is that the representation of content and style in a CNN is effectively separable [1].

They utilize the feature space provided by the 16 convolutional and 5 pooling layers of the 17-layer VGG network, commonly referred to as VGG19 [1][5]. Each layer in the network defines a non-linear filter bank, with the complexity increasing based on the layer's position in the network [1]. Thus, a given input image, $\vec{x}$, is encoded in each layer of the CNN by the filer response to that image [1]. A layer with N_l distinct filters has N_l feature maps, each of size Ml, where Ml represents the height times the width of the feature map [1]. Therefore, the responses in a layer l can be stored in a matrix $F^l \in R^{N_l M_l}$, where Fl is the activation of the ith filter at position j in ij layer l. To visualize the image information encoded at different layers of the hierarchy, gradient descent is performed on a white noise image to find another image that matches the feature responses of the original image [3]. Let $\vec{p}$ and $\vec{x}$ be the original image and the generated image, and P^l and F^l their respective feature representations in layer l [1]. The squared error loss between the two feature representations is given by

$$L_{\text{content}} = \frac{1}{2} \sum_{i,j} (F^l_{i,j} - P^l_{i,j})$$

Once obtained, the initial randolm image $\vec{x}$ is adjusted until it generates the same response in a certain layer of the CNN as the original image $\vec{p}$ [1].

Building on the CNN responses in each layer of the network, we construct a style representation that calculates correlations between different filter responses, with the expectation taken over the spatial extent of the input image. These feature correlations are expressed through the Gram matrix, $G^l \in R^{N_l N_l}$, where $G^l_{i,j}$ is the inner product between the vectorized feature map i and j in layer l:

$$G^l_{i,j} = \sum_k F^l_{i,j} F^l_{j,k}$$

To generate a texture matching the style of a given image, a gradient descent from a white noise image is employed to find another image that aligns with the style representation of the original image [1]. This involves minimizing the $L2$ distance between the entries of the Gram matrix from the original image and the generated image, with A^l and G^l representing their respective style representations in layer l. The contribution of that layer to the total loss is then:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G^l_{i,j} - A^l_{i,j})^2$$

The total loss is then given by

$$L_{\text{style}}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

where w_l represents the weighing factors determining the contribution of each layer to the total loss.

To create images that blend the content of a photograph with the style of a painting, we simultaneously minimize the distance of a white noise image from the content representation of the photograph in one layer of the network and the style representation of the painting in multiple layers of the CNN. The loss function subject to minimization is thus:

$$L_{\text{total}}(\overrightarrow{p}, \overrightarrow{a}, \overrightarrow{x}) = \alpha L_{\text{content}}(\overrightarrow{p}, \overrightarrow{x}) + \beta L_{\text{style}}(\overrightarrow{a}, \overrightarrow{x})$$

Here, α and β act as weighing factors for content and style reconstruction, respectively [1].

Conditional Imitation Learning

Conditional imitation learning is a machine learning framework used in the field of robotics and artificial intelligence. In this approach, an agent learns to imitate the actions of an expert demonstrator based on observed data. The "conditional" aspect refers to the incorporation of additional information such as steer controls in our context. The agent is trained to take a conditional information in addition to RGB information.

In our context, we instruct the agent to take turns by providing it with an instruction. The agent then turns according to the nature of the turn and its understanding of the surroundings.

Approach

Network Architecture

The convolutional block in our system comes from a family of RegNetY[8] models from Radosavovic et al (2020). The networks are appended with a custom five-layer convolution network, with layer widths of [512, 128, 64, 32, 1]. The convolutional network is designed to take in channels depending on the baseline and different methods. Apart from the number of channels in the input, other aspects of the networks are preserved.

The outputs of this model are passed onto a five-layer fully connected network with layer widths of [512, 128, 64, 32, 1]. Since this is a conditional imitation learning method, we also condition the network so that the appropriate output is produced. This is done by appending, to each layer of the fcc, an input that corresponds to the conditioning signal or input.

The fully connected network branches into two linear layers which output the scalar values corresponding to the driving controls: steer and throttle. The range of values for the steer is from -1 to 1, to represent different degrees of left and right, and throttle outputs go from 0 to 1 corresponding to values from no-throttle to full-throttle.

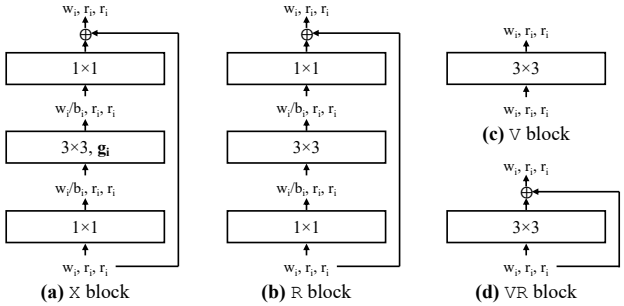

Figure 2: *Building Block of RegNetY family of networks*

Training

To train this particular model, we minimize the following loss function

$$L(\hat{a}, a) = ||\hat{S} - S||_1 + ||\hat{T} - T||_1$$

where, $\hat{S}$ represents the predicted steer value. S represents the ground-truth value. Similarly, $\hat{T}$ represents the predicted throttle value and T represents the ground-truth throttle value.

Dataset

Due to the nature of Sim2Real, we use datasets with different characteristics for training and testing.

The training dataset is obtained from the expert model introduced in Transfuser [12]. The expert model is a model with privileged information that produces the appropriate state-action pairs that will be used as the training dataset. The dataset is generated from two towns in the CARLA Simulator: Town01 and Town02. Overall the dataset contained 20,000 frames each tagged with control and command data. The whole dataset is used for training, running for 30 epochs.

The testing dataset is a set of state-action pairs obtained from a human expert [9]. The test dataset is further split into two sets: indoor and outdoor.

The indoor dataset was obtained by building a model city using a black coloured mat with taped lanes to simulate roadways, obtained from Aich [9]. This environment is structure-wise similar to the training dataset compared to the outdoor dataset.

The outdoor dataset, from Aich [9], was obtained by driving a remote-controlled vehicle equipped with a camera to capture real-time data and navigate through the paths. This dataset is significantly different from that of the training dataset. Despite this complexities, the outdoor setup offers valuable insights into the adaptability and robustness of driving

algorithms when put in contexts different from that of the training data.

Baselines

We compare the performance of our method with the following baselines, from Aich[9].

- RGB: Baseline with input data being RGB image.
- RGB with Depth: To the network, we feed the RGB as three channels plus an additional channel where we feed it the output of the Depth Anything model [10].
- Segmentation Mask: To the network, we feed just the output of Segment Anything Model [11]. The network takes one channel of input.
- Segmentation Mask with Depth: The network takes in two channels where the first channel is the segmentation mask, obtained from SAM [10], and the second model is the depth mask, obtained from Depth Anything [11].
- Contour + Depth: Using the countour information from SAM [11] as one channel and the depth information, obtained from Depth Anything [10], as second channel.
- Mask + Contour + Depth: Three channel input. The first channel being the mask obtained from SAM [11], the second channel being the contour, obtained from SAM [11] and the third channel being the depth, obtained from [10].

The primary contribution of our project is training a subset of the RegNetY[8] family network with these baselines as well as a set of style transferred inputs.

Method

For each of the baseline and our style transfer methods, the original RGB training dataset is converted to their appropriate inputs. Thus, we create a dataset for each of our baseline and our NST setups. These datasets are then used to train a subset of the RegNetY family of networks, specifically, RegNetY-002, RegNetY-004 and RegNetY-008.

For the style-transferred dataset, we style transfer the training dataset into style transferred dataset by using subjectively selected style images. Each NST dataset differs from the other based on the style-weight used for each dataset. The NST04 refers to images style transferred with the styleweight of 1e8 and linearly decreased until NST11, which corresponds to images style transferred with style-weight 1e2. The range of style-weights are selected by subjectively choosing weights that produced images where the content hasn't changed enough to change the action but while changing the texture and colour significantly.

Evaluation Metric

We use four metrics defined as follows

- Steer Mean Absolute Error

$$\mathbf{L}_{MAE} = ||a_i - \hat{a}_i||_1$$

- Speed Weighed Steer Mean Absolute Error

$$\mathbf{L}_{SW-MAE} = \frac{1}{N}\sum_{i=0}^{N-1} ||v(a_i - \hat{a}_i)||_1$$

- Action Mean Absolute Error

$$\mathbf{L}_{MAE} = ||[a_i, s_i] - [\hat{a}_i, \hat{s}_i]||_1$$

- Speed-weighed Action Mean Absolute Error

$$\mathbf{L}_{SW-MAE} = \frac{1}{N}\sum_{i=0}^{N-1} ||v([a_i, s_i] - [\hat{a}_i, \hat{s}_i])||_1$$

Results

The results of our experiments are presented in Tables, 1, 2, 3, 4, 5, and 6. Tables 1, 2, and 3, shows the results of the three models when tested on the indoor dataset. Tables, 4, 5, and 6, shows the results of the three models when tested on the outdoor dataset.

We see that the NST methods are outperforming the other baselines a significant proportion of the time. RegNetY-002 with NST produced a decrease of 20% in Steer-MAE, 12% in Steer-SW-MAE, 16% in Action-MAE. RegNetY-004 with NST produced an decrease of 14% in Steer-MAE, 11% in Steer-SW-MAE, 7% in Action-MAE and 18% in Action-SW-MAE.

This is promising because the NST methods do not add any stages to the existing pipelines whereas the others do. The methods based on SAM [11] or depth-anything [10] involve running RGB image through a large model, which significantly increases latency. This means that in order to reduce it, one would have to employ a substantial amount of additional compute resources to reduce latency. The NST method on the other hand only requires the compute during training, while also reducing the amount of human effort compared to other domain randomization methods.

Thus, NST is a best of both-worlds as it reduces human effort for domain randomization and increase performance, all without increasing latency.

Table 1: *Indoor Metrics for RegNetY-002.*

Model Name	Steer MAE ↓	SW Steer MAE ↓	Action MAE ↓	SW Action MAE ↓
RGB	0.294	0.391	0.319	0.85
RGB + Depth	0.276	0.383	0.331	**0.812**
SAM-Mask	0.295	0.397	0.304	0.955
SAM-Mask + Depth	0.308	0.405	0.332	0.861
Contour + Depth	0.324	0.414	0.347	0.876
SAM-Mask + Contour + Depth	0.284	0.377	0.312	0.853
RGB-NST04	0.27923	0.38324	0.28695	0.93556
RGB-NST05	0.27185	0.36734	0.28558	0.93269
RGB-NST06	0.27627	0.38895	0.29564	0.89651
RGB-NST07	0.26733	0.37641	0.28837	0.9218
RGB-NST08	**0.23931**	**0.34121**	**0.2647**	0.91443
RGB-NST09	0.59785	0.60309	0.60806	0.92599
RGB-NST10	0.30979	0.42027	0.31789	0.9683
RGB-NST11	0.27488	0.38933	0.29384	0.91739

Table 2: *Indoor Metrics for RegNetY-004.*

Model Name	Steer MAE ↓	SW Steer MAE ↓	Action MAE ↓	SW Action MAE ↓
RGB	0.278	0.376	0.296	0.935
RGB + Depth	0.304	0.407	0.321	0.945
SAM-Mask	0.253	0.355	0.278	0.888
SAM-Mask + Depth	0.321	0.418	0.331	0.892
Contour + Depth	0.296	0.397	0.33	0.838
SAM-Mask + Contour + Depth	0.3	0.392	0.317	0.909
RGB-NST04	0.26959	0.37282	0.28365	0.94059
RGB-NST05	0.23466	**0.33212**	0.28339	0.87438
RGB-NST06	0.26265	0.35814	0.29365	0.82078
RGB-NST07	0.35429	0.45833	0.39736	0.76461
RGB-NST08	0.25557	0.35246	**0.27438**	0.94198
RGB-NST09	1.2225	1.1184	0.8471	1.6286
RGB-NST10	0.29287	0.39248	0.33297	0.81092
RGB-NST11	**0.23205**	0.33841	0.28439	**0.75279**

Conclusion

In this project, we've shown the promise of Neural Style Transfer as a superior domain randomization alternative for Sim2Real, especially in the context of autonomous navigation. We've shown the improvement in performance this approach produces in addition to its computational advantages. We also make a case for NST in the ease of engineering aspect.

A future direction of research, based on the work presented here, would be on finding better ways of implementing our task, which is, essentially, prevention of non-causal correlations. NST as an approach was chosen based on its ability to randomize non-causal(with respect to action) aspects of the input state without changing the causal aspects of the frame. However, in addition to this, a significant amount of compute and latency is allocated to changing the appearance to meet the style of that of the reference style image. Thus, finding ways to obtain our objective while avoiding unnecessary steps would further speed up the training process in addition to increasing efficiency.

Ultimately, we're aiming to teach the neural network to learn state-action mappings that would resemble that of a rational human's. To that extent, a cursory understanding of the human mind can guide us in this endeavour. The concept of, "affordance", introduced by J.J. Gibson in *The Ecological Approach to Visual Perception*, 1977[13], highly guided the intuition for this project. The human mind's understanding that an object is defined, not by its appearance but by how we're to interact with it, is a crucial concept that must be transferred to artificial agents. The human mind classifies entities as to how one should interact with it. A road is not a road because it has a black

Table 3: *Indoor Metrics for RegNetY-008.*

Model Name	Steer MAE ↓	SW Steer MAE ↓	Action MAE ↓	SW Action MAE ↓
RGB	0.26	0.355	0.295	0.813
RGB + Depth	**0.25**	**0.342**	0.345	**0.786**
SAM-Mask	0.269	0.367	0.308	0.899
SAM-Mask + Depth	0.324	0.42	0.354	0.833
Contour + Depth	0.299	0.39	0.331	0.864
SAM-Mask + Contour + Depth	0.289	0.394	0.324	0.868
RGB-NST04	0.38345	0.45432	0.35487	0.95521
RGB-NST05	0.31622	0.41746	0.3311	0.90554
RGB-NST06	0.32741	0.41219	0.34132	0.88435
RGB-NST07	0.27905	0.35695	0.33679	0.79567
RGB-NST08	0.26435	0.35576	**0.2834**	0.82724
RGB-NST09	1.3357	1.2182	1.025	1.51
RGB-NST10	0.28295	0.36905	0.32159	0.91058
RGB-NST11	0.28285	0.37567	0.30697	0.86471

Table 4: *Outdoor Metrics for RegNetY-002.*

Model Name	Steer MAE ↓	SW Steer MAE ↓	Action MAE ↓	SW Action MAE ↓
RGB	0.08	0.178	0.357	1.586
RGB + Depth	0.079	0.179	**0.195**	**0.782**
SAM-Mask	0.1	0.22	0.439	1.986
SAM-Mask + Depth	0.086	0.191	0.335	1.475
Contour + Depth	0.122	0.264	0.394	1.753
SAM-Mask + Contour + Depth	0.097	0.214	0.316	1.37
RGB-NST04	**0.070168**	**0.16123**	0.42705	1.9455
RGB-NST05	0.18072	0.39289	0.47923	2.1633
RGB-NST06	0.072733	0.16189	0.40775	1.8347
RGB-NST07	0.074743	0.17182	0.40871	1.8513
RGB-NST08	0.11071	0.24416	0.36804	1.6446
RGB-NST09	0.43561	0.93371	0.25707	0.9663
RGB-NST10	0.16737	0.37257	0.43696	1.9682
RGB-NST11	0.077026	0.17704	0.29887	1.2985

colour and has white lines along the edges. A road is anything that allows our successful traversal on them. Though understood intuitively, the challenge lies in transferring this knowledge in training.

It's a long road to the ideal policy, but I hope this project will inform the reader and aid their understanding of the motivations and intuition for the development of this approach and, more importantly, why it worked.

Appendix A

Please refer to Table 7 for team-member contributions.

Appendix B

Link to github project page `https://github.com/vrsreeganesh/EC523_Project`

References

[1] Gatys, L. A., Ecker, A. S., and Bethge, M. (2015). *A Neural Algorithm of Artistic Style.* In Advances in Neural Information Processing Systems (NeurIPS).

[2] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, Vladlen Koltun. (2017) *CARLA: An Open Urban Driving Simulator*. CoRR 2017

[3] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., Abbeel, P. (2017). *Domain Random-*

Table 5: *Outdoor Metrics for RegNetY-004.*

Model Name	Steer MAE ↓	SW Steer MAE ↓	Action MAE ↓	SW Action MAE ↓
RGB	0.093	0.205	0.481	2.199
RGB + Depth	0.082	0.19	**0.242**	**1.015**
SAM-Mask	0.082	0.194	0.435	1.977
SAM-Mask + Depth	0.116	0.251	0.37	1.641
Contour + Depth	0.097	0.214	0.338	1.486
SAM-Mask + Contour + Depth	0.102	0.223	0.365	1.611
RGB-NST04	0.077964	0.17797	0.41577	1.8774
RGB-NST05	0.071103	0.16338	0.3812	1.7169
RGB-NST06	0.082481	0.18936	0.34499	1.5261
RGB-NST07	**0.068497**	**0.157**	0.31122	1.3787
RGB-NST08	0.07926	0.18112	0.40725	1.8437
RGB-NST09	0.98152	2.1218	0.75437	3.2444
RGB-NST10	0.093697	0.21122	0.38808	1.7472
RGB-NST11	0.076934	0.17668	0.3258	1.4395

Table 6: *Outdoor Metrics for RegNetY-008.*

Model Name	Steer MAE ↓	SW Steer MAE ↓	Action MAE ↓	SW Action MAE ↓
RGB	0.094	0.207	0.393	1.233
RGB + Depth	0.1	0.218	0.225	**0.919**
SAM-Mask	0.092	0.206	0.324	1.427
SAM-Mask + Depth	0.113	0.247	0.346	1.52
Contour + Depth	0.11	0.239	0.355	1.567
SAM-Mask + Contour + Depth	**0.086**	**0.198**	**0.218**	1.388
RGB-NST04	0.16652	0.36669	0.49693	2.2558
RGB-NST05	0.17071	0.36936	0.42486	1.9018
RGB-NST06	0.15499	0.33344	0.33888	1.4647
RGB-NST07	0.23014	0.5027	0.43839	1.9513
RGB-NST08	0.1252	0.27385	0.39739	1.7679
RGB-NST09	1.3153	2.8276	0.69692	2.8602
RGB-NST10	0.12408	0.27616	0.44262	2.0042
RGB-NST11	0.17621	0.37325	0.33304	1.4331

ization for Transferring Deep Neural Networks from Simulation to the Real World. arXiv:1703.06907. [cs.RO]

[4] Emanuel Todorov, Tom Erez, and Yuval Tassa. *Mujoco: A physics engine for model-based control*. IROS, 2012 IEEE/RSJ International Conference on, pages 5026– 5033. IEEE, 2012.

[5] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. arXiv, 2015.

[6] So, J., Xie, A., Jung, S., Edlund, J., Thakker, R., Agha-mohammadi, A., Abbeel, P., James, S. (2022). *Sim-to-Real via Sim-to-Seg: End-to-end Off-road Autonomous Driving Without Real Data* (No. 2210.14721). arXiv. https://arxiv.org/abs/2210.14721

[7] Loquercio, A., Kaufmann, E., Ranftl, R., Dosovitskiy, A., Koltun, V., Scaramuzza, D. (2020). *Deep Drone Racing: From Simulation to Reality With Domain Randomization*. IEEE Transactions on Robotics, 36(1), 1–14. http://dx.doi.org/10.1109/TRO.2019.2942989

[8] Ilija Radosavovic, Raj Prateek Kosaraju, Ross B. Girshick, Kaiming He and Piotr Dollar. *Designing Network Design Spaces*. CoRR, 2020.

[9] Animikh Aich, *Towards Closing The Generalization Gap In Autonomous Driving*. MS Thesis, Boston University, 2024.

[10] Yang, L., Kang, B., Huang, Z., Xu, X., Feng, J., and Zhao, H. (2024). *Depth anything: Unleashing the power of large-scale unlabeled data*. CVPR, 2024

[11] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, Ross Girshick. *Segment Anything*. Arxiv, 2023

Table 7: *Team Member Contributions*

Name	Task	File Names	Line Count
Sreeganesh	Dataset Creation	Run_CreateNST_∗.sh CreateNSTDataset∗.py	16 per script, 16 scripts 422 per script, 16 scripts
	Training	train∗ _regnety _∗. sh train _baseline ∗ _regnety _∗.py	55 per script, 70 scripts 552 per script, 70 scripts
	Evaluation	offline_eval ∗ _regnety _∗. sh evaluate _offline ∗ _regnety _∗. sh ∗.py	16 per script, 41 scripts 422 per script, 41 scripts

[12] Chitta, K., Prakash, A., Jaeger, B., Yu, Z., Renz, K., & Geiger, A. (2022). *Transfuser: Imitation with transformer-based sensor fusion for autonomous driving*. IEEE Transactions on Pattern Analysis and Machine Intelligence.

[13] Gibson, J. J. (1979). *The ecological approach to visual perception*. Houghton, Mifflin and Company.