Analyzing the Robustness of Depth Estimation Networks

Daniel Silva, Kesar Murthy, Blake Wyatt
University of Central Florida
{danielzgsilva, kesar, blakewyatt}@knights.ucf.edu

***Abstract*—Depth estimation is a widely-used application of machine learning. From autonomous systems to navigation to mapping, it's important for depth estimation to be accurate and robust to adversarial attacks. To our knowledge, recent state-of-the-art approaches to monocular depth estimation such as Fully Convolutional Residual Networks (FCRNs), AdaBins, and Dense Prediction Transformer (DPT) have not yet been analyzed for adversarial robustness. We therefore take this opportunity to provide that analysis for both targeted and untargeted attacks using the Projected Gradient Descent (PGD) and Momentum Iterative Fast Gradient Sign Method (MI-FGSM). While there has been research on the adversarialness of monocular depth estimation algorithms, there has not been much research on robustness due to adversarial training. Thus, after comparing the robustness of each model without adversarial training, we adversarially train each model and compare their results. Finally, we analyze the improvement adversarial training had on these models and hypothesize why we find the results that we do.**
The code for this work is publicly available at *https://github.com/danielzgsilva/MonoDepthAttacks*

I. Introduction

Depth estimation networks can be found in self-driving cars, Google Maps, Augmented Reality, Virtual Reality, smartphone camera applications, and more. They are widely-used and have critical use cases. For example, if a self-driving car's camera mispredicts the depth of a person, expecting them to be farther away than they actually are, the car could kill that person. To avoid such problems, we need to do what we can to make depth estimation networks robust.

In this paper, we focus on attacking certain state-of-the-art monocular depth estimation networks which, to our knowledge, have not yet been evaluated against adversarial attacks. Specifically, we will be covering Fully Convolutional Residual Networks (FCRNs) [4], AdaBins [5], and Dense Prediction Transformer (DPT) [6]. We attack all three networks with the well-known Projected Gradient Descent (PGD) [9] [7], Momentum Iterative Fast Gradient Sign Method (MI-FGSM) [12] [7], and perhaps introduce a novel attack technique built specifically for depth estimation attacks. The attacks are done in both both targeted and un-targeted scenarios, compared against their respective baselines, and compared against each other to determine which of these three state-of-the-art networks is most adversarially robust.

Despite adversarial training being a common technique to improve a network's robustness in classification networks, there hasn't been much research applying adversarial training to monocular depth estimation networks. To this end, We adversarially train all three of our state-of-the-art neural networks FCRN, AdaBins, and DPT to determine how effective adversarial training is on modern monocular depth estimation networks. Finally, the results obtained when attacking these adversarially trained networks are compared against our results when attacking without adversarially training.

II. Related Work

A. Monocular Depth Estimation Models

In this section, we briefly describe the architectures and algorithmic approach of the depth estimation networks chosen for our analysis. We specifically leverage Fully Convolutional Residual Networks (FCRN) [4] as a baseline, which follows the standard encoder-decoder structure typically used for depth estimation. Networks such as these have already been studied in the context of adversarial attack [7], and thus serve as strong baseline for this work.

Due to their down-sampling operations, it is well known that these approaches struggle with feature resolution and granularity being lost in the encoder and thus difficult to recover in the decoder. While this is less important for tasks such as image classification, feature resolution is critical for dense prediction tasks such as depth estimation. To solve this, recent approaches introduce a Vision Transformer (ViT) [1] module either in the feature extractor itself [6] or as a post-decoding step [5]. Unlike fully convolutional networks [4], whose feature processing at high resolution is limited to small and local receptive fields, ViT architectures are able to process representations at high resolutions in a global manner. The strong performance of these networks [6], [5] backs this intuition, showing that the properties of ViT models are advantageous for dense prediction tasks lead to fine grained and globally coherent predictions.

We hypothesize that this global processing ability of ViT models will improve their robustness to adversarial attack, especially when compared to our baseline FCRN model. Experiments to test this hypothesis are described in later sections.

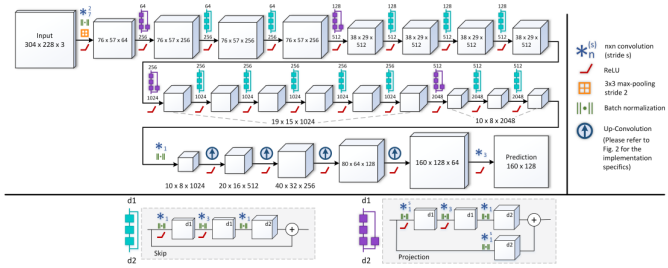

Fig. 1. Architecture of the Fully Convolutional Residual Network proposed in [4] with a ResNet50 encoder and up-projection blocks.

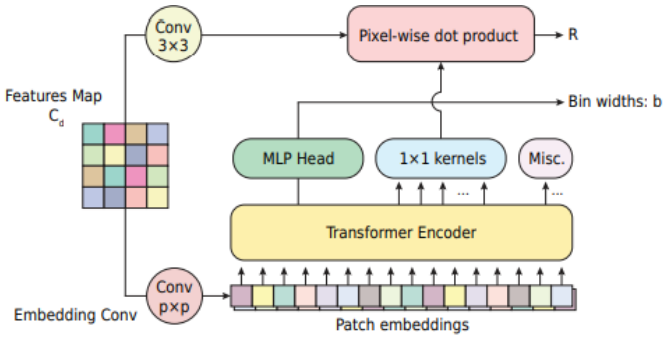

Fig. 2. The mini-ViT block proposed in [5]. A feature map is fed to a transformer that is applied on patch embeddings to estimate bin widths and a Range-Attention-Map R.

Fully Convolutional Residual Network (FCRN) A fully convolutional network typically applies a pretrained classification network to progressively downscale the input image and build a feature representation. Rather than a fully connected layer, the representation is given to a decoder to upscale the representation into a prediction. We follow Laina et al. [4] in leveraging a ResNet-50 backbone with skip connections that bypass two or more convolutional layers. The final fully connected layer is replaced with their novel up-projection block, consisting of repeated reverse pooling and convolutional layers with projection connections. We assume an input size of 304 204 and build the decoder to outsize a map of the same spatial size. These models are typically trained with an L_1 or L_2 loss, as well as a reverse Huber loss that balances the previous two losses based on each pixel's absolute difference between prediction and ground-truth. The architecture of this model is displayed in Figure 1.

AdaBins: Depth Estimation with Adaptive Bins In a normal encoder-decoder architecture the feature resolution and granularity is lost in the deeper stages of the model and are harder to recover in the decoder. Proposed by Bhat et al. [5], one approach to solving this is to leverage a Vision Transformer network to explicitly learn adaptive depth bins and enforce global consistency. Specifically, an encoder-decoder network similar to [4] is adapted to predict a feature map of size $H\ x\ W\ x\ C$, which is fed to the mini-transformer architecture pictured in Figure 2. This module predicts a vector of **b** bin widths, defining how to divide the depth interval of an image, as well as a Range-Attention map with per-pixel information.

To produce the bin width vector, the feature representation is fed through an embedding convolution and flattened into patch embeddings before enter the transformer. The output of the transformer is cast onto N dimensional space and normalized to create vector **b** of **N** bin depths. This enables the network to focus on image-specific depth intervals and maintain global coherence once applied with the global attention map **R**.

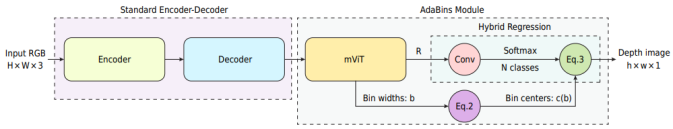

Fig. 3. The AdaBins [5] encoder-decoder and adaptive bin estimation architecture.

To do so, these Range-Attention maps **R** are passed through a 1x1 convolution and Softmax function to create a probability distribution for **N** bins for each pixel. The final per-pixel depth is then given by a linear combination between bin probabilities and bin centers from vector **b**. This encoder-decoder, transformer, and adaptive bin architecture is visualized in Figure 2.

Dense Prediction Transformer (DPT)

The Dense Prediction Transformer [6] leverages vision transformers in place of convolutional blocks as the basic computational block for the encoders for the dense prediction task. Like AdaBins, it uses ViT [1] as the backbone having global receptive field at every stage progressively increasing the receptive fields as features pass through downsampling layers. Ranftl et. al. further show how this leads to fine-grained and globally coherent predictions. The paper presents 3 unique architectures: DPT-Base, DPT-Large and DPT-Hybrid. We use DPT-Hybrid for all our experiments.

The non-overlapping patches from are individually embedded using linear projections in DPT-Large. DPT-Hybrid however, uses feature extracted by ResNet-50 as the tokens. This is concatenated with positional embeddings and a special readout token to capture and distribute global information. The special token is not grounded by the input image and serves as the final, global image representation. This is passed through 12 transformer layers. These feature representations are progressively fused using a 3 stage reassemble operation as described in the Figure 4. The features are reassembled at four different stages with four different resolutions. These consecutive feature maps are combined using RefineNet-based feature fusion blocks. The final representation is half the resolution of the inut image, which is resized to evaluate against the targets.

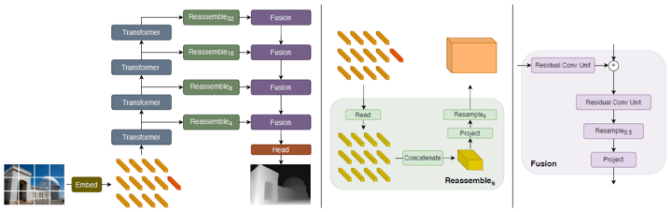

Fig. 4. The image is transformed into tokens using a ResNet-50 feature extractor. The image embeddings are augmented with positional embeddings and the readout token (red). The tokens are passed through multiple transformer stages represented in blue. The reassemble blocks (green) convert these tokens into image-like representations at different resolutions. The fine-grained predictions are generated using the upsampling blocks in the fusion module (Purple).

The DPT-Hybrid model which is trained on Ade20k dataset is fine-tuned on KITTI and NYUv2 datasets. Since the network is trained with an affine-invariant loss, the predictions can be arbitrarily scaled and the depth values can have a larger magnitude. For the KITTI model, the gradient-matching loss is disabled since it only provides sparce ground-truth. The network performs extremely well in monocular depth estimation tasks on KITTI and NYUv2 datasets.

III. Adversarial Attacks

To attack our depth estimation networks, we implement two common methods for attacking neural networks; Fast Gradient Sign Method (FGSM) and Momentum Iterative Fast Gradient Sign Method (MI-FGSM). We also implement both targeted or untargeted losses that are applied and optimized to learn perturbations.

A. *Projected Gradient Descent (PGD)*

PGD is a very successful Iterative white-box attack. It frames the generation of adversarial perturbations as a constrained optimization problem. PGD attempts to find perturbations that maximize the model loss for a given instance, keeping the perturbations lower than some value ε. The algorithm starts with a random perturbation within the L_p ball, and repeatedly takes a desired step size in the direction of the greatest loss w.r.t the gradients being calculated with each iteration. Finally, it projects the perturbation back onto the L_p ball to satisify the magnitude constraint in order to make the perturbations imperceptible. The algorithm can be formalized as

TABLE I
BASELINE MONOCULAR DEPTH ESTIMATION RESULTS OF FCRN, ADABINS, AND DPT-HYBRID ON CLEAN IMAGES OF BOTH KITTI AND NYU-DEPTH 2.

			Lower is better		Accuracy: higher is better		
Dataset	Model	Backbone	RMSE	Abs Rel	δ_1	δ_2	δ_3
KITTI	FCRN	ResNet-50	6.032	0.159	0.766	0.932	0.979
KITTI	AdaBins	ResNet-50	**2.360**	**0.058**	**0.964**	**0.995**	**0.999**
KITTI	DPT-hybrid	ResNet-50	2.573	0.062	0.959	**0.995**	**0.999**
NYUv2	FCRN	ResNet-50	0.573	0.127	0.811	0.953	0.988
NYUv2	AdaBins	ResNet-50	0.364	**0.103**	0.903	0.984	0.997
NYUv2	DPT-hybrid	ResNet-50	**0.357**	0.110	**0.904**	**0.988**	**0.998**

Initialize δ;
Repeat for n iterations:
$$\delta = P(\delta + \alpha \nabla_\delta L(x+\delta, y, \theta))$$

where δ represents the generated perturbation, P denotes the projection onto the L_p ball, x is the original input, y is its ground truth label, θ are the static model weights, L is the model loss, ∇_δ is the loss gradient with respect to the input, and α is a desired step size.

B. Momentum Iterative Fast Gradient Sign Method (MI-FGSM)

Derived from FGSM, MI-FGSM applies FGSM in an iterative fashion with a momentum term to carry momentum from one iteration to the next.

$$\mathbf{g}_{t+1} = \mu \cdot \mathbf{g}_t + \frac{\nabla_{\mathbf{x}} L(f(\mathbf{x};\theta), \mathbf{y}^{\text{true}})}{||\nabla_{\mathbf{x}} L(f(\mathbf{x};\theta), \mathbf{y}^{\text{true}})||_1}, \quad (1)$$

Similarly, MI-FGSM uses $\mathbf{x}$ and parameter θ as input to network f. The output from f is a prediction given those two inputs and L computes the loss between the predicted output value and the ground truth value $\mathbf{y}^{\text{true}}$. The gradient is computed from the loss function and divided over its absolute value. g_t is the momentum gradient term from the previous iteration and μ is the decay factor for that momentum. The previous gradient term is reduced by the decay factor and applied to the current gradient term, retaining momentum from the previous iteration.

It should be noted that in classical adversarial attack settings, the loss function is a standard Cross-Entropy function for classification. In the case of depth

$$\mathbf{x}^{\text{adv}}_{t+1} = \text{Clip}^{\varepsilon}_{\mathbf{x}}\{\mathbf{x}^{\text{adv}}_t + \alpha \cdot \text{sign}(\mathbf{g}_{t+1})\}. \quad (2)$$

The *sign* is taken to determine the direction in which we need to perturb the input. α controls the step size of the perturbation and $\mathbf{x}^{\text{adv}}_t$ is the input which we are applying the perturbation to. $\text{Clip}^{\varepsilon}_{\mathbf{x}}$ iterates over its contents ε times and finally, produces the final adversarial example $\mathbf{x}^{\text{adv}}_{t+1}$.

IV. TYPES OF DEPTH ATTACKS

PGD and MI-FGSM utilize a loss function L to perturb their input in the direction that decreases robustness. Depending on our intent, we can use a loss function that performs a targeted attack or we can use a loss function that performs an untargeted attack. The direction of the gradient will then be oriented to best achieve whichever goal we set.

A. Untargeted Attacks

Untargeted depth attacks seek only to maximize the global prediction error of a given image. That is, to make the network output the depth values farthest away from what the depth of the input actually is, regardless of any specific objects in the input.

$$\max_{\mathbf{x}^{\text{adv}} - \mathbf{x}} L(f(\mathbf{x}^{\text{adv}};\theta), \mathbf{y}^{\text{true}}) \;\; s.t. \; \|\mathbf{x} - \mathbf{x}^{\text{adv}}\|_\infty < \varepsilon. \quad (3)$$

In this case, we are maximizing the loss by modifying the perturbations we apply to the input while staying within ε. $\mathbf{x}$ and parameter θ are used as input to network f. The output from f is a prediction given those two inputs and L computes the loss between the predicted output value and the ground truth value $\mathbf{y}^{\text{true}}$.
Since this isn't a classification task which includes

TABLE II
MONOCULAR DEPTH ESTIMATION RESULTS AFTER ATTACKING WITH PGD.

						Lower is better			Accuracy: higher is better		
Dataset	Model	Backbone	Loss	Epsilon	Iterations	RMSE	Abs Rel	Log10	δ_1	δ_2	δ_3
KITTI	FCRN	ResNet-50	L1	3	7	33.899	1.638	0.467	0.067	0.170	0.305
KITTI	FCRN	ResNet-50	L1	5	9	43.907	2.187	0.519	0.048	0.129	0.254
KITTI	FCRN	ResNet-50	L1	10	14	62.492	3.178	0.585	0.033	0.094	0.200
KITTI	FCRN	ResNet-50	L1	16	20	81.124	4.158	0.641	0.024	0.071	0.160
KITTI	FCRN	ResNet-50	L2	3	7	32.581	1.480	0.439	0.122	0.237	0.365
KITTI	FCRN	ResNet-50	L2	5	9	41.109	1.919	0.481	0.101	0.201	0.326
KITTI	FCRN	ResNet-50	L2	10	14	56.351	2.671	0.528	0.079	0.169	0.289
KITTI	FCRN	ResNet-50	L2	16	20	70.770	3.376	0.568	0.061	0.141	0.253
KITTI	FCRN	ResNet-50	berHu	3	7	31.158	1.384	0.426	0.141	0.261	0.389
KITTI	FCRN	ResNet-50	berHu	5	9	38.891	1.770	0.461	0.125	0.232	0.357
KITTI	FCRN	ResNet-50	berHu	10	14	52.345	2.409	0.499	0.108	0.208	0.331
KITTI	FCRN	ResNet-50	berHu	16	20	64.801	3.002	0.531	0.089	0.182	0.300
KITTI	AdaBins	ResNet-50	L1	3	7	16.366	0.591	0.347	0.166	0.322	0.490
KITTI	AdaBins	ResNet-50	L1	5	9	19.350	0.787	0.417	0.127	0.258	0.407
KITTI	AdaBins	ResNet-50	L1	10	14	26.239	1.225	0.512	0.098	0.218	0.325
KITTI	AdaBins	ResNet-50	L1	16	20	31.429	1.605	0.563	0.060	0.142	0.243
KITTI	AdaBins	ResNet-50	L2	3	7	17.818	0.635	0.417	0.162	0.286	0.353
KITTI	AdaBins	ResNet-50	L2	5	9	20.134	0.785	0.485	0.126	0.251	0.329
KITTI	AdaBins	ResNet-50	L2	10	14	25.062	1.074	0.527	0.112	0.226	0.316
KITTI	AdaBins	ResNet-50	L2	16	20	28.910	1.320	0.527	0.098	0.202	0.304
KITTI	AdaBins	ResNet-50	berHu	3	7	18.000	0.634	0.429	0.135	0.257	0.346
KITTI	AdaBins	ResNet-50	berHu	5	9	20.072	0.760	0.487	0.122	0.246	0.335
KITTI	AdaBins	ResNet-50	berHu	10	14	24.539	1.015	0.518	0.114	0.232	0.330
KITTI	AdaBins	ResNet-50	berHu	16	20	28.161	1.240	0.511	0.107	0.219	0.326
KITTI	DPT-Hybrid	ResNet-50	L1	3	7	22.299	1.148	0.432	0.349	0.244	0.296
KITTI	DPT-Hybrid	ResNet-50	L1	5	9	26.149	1.456	0.506	0.261	0.210	0.286
KITTI	DPT-Hybrid	ResNet-50	L1	10	14	31.835	1.976	0.596	0.177	0.171	0.278
KITTI	DPT-Hybrid	ResNet-50	L1	16	20	36.148	2.425	0.651	0.128	0.155	0.271
KITTI	DPT-Hybrid	ResNet-50	L2	3	7	20.983	0.880	0.363	0.139	0.311	0.509
KITTI	DPT-Hybrid	ResNet-50	L2	5	9	24.218	1.099	0.429	0.121	0.276	0.467
KITTI	DPT-Hybrid	ResNet-50	L2	10	14	28.988	1.442	0.499	0.104	0.197	0.401
KITTI	DPT-Hybrid	ResNet-50	L2	16	20	32.723	1.720	0.537	0.098	0.130	0.355
KITTI	DPT-Hybrid	ResNet-50	berHu	3	7	18.663	0.683	0.322	0.111	0.274	0.324
KITTI	DPT-Hybrid	ResNet-50	berHu	5	9	21.251	0.812	0.376	0.93	0.232	0.275
KITTI	DPT-Hybrid	ResNet-50	berHu	10	14	25.310	1.080	0.446	0.088	0.176	0.199
KITTI	DPT-Hybrid	ResNet-50	berHu	16	20	28.652	1.271	0.476	0.047	0.120	0.144
NYUv2	DPT-Hybrid	ResNet-50	L1	3	7	3.156	1.256	0.422	0.032	0.117	0.263
NYUv2	DPT-Hybrid	ResNet-50	L1	5	9	3.462	1.440	0.468	0.021	0.077	0.193
NYUv2	DPT-Hybrid	ResNet-50	L1	10	14	3.798	1.658	0.518	0.012	0.046	0.128
NYUv2	DPT-Hybrid	ResNet-50	L1	16	20	3.998	1.793	0.548	0.008	0.034	0.098
NYUv2	DPT-Hybrid	ResNet-50	L2	3	7	3.119	1.177	0.417	0.058	0.164	0.303
NYUv2	DPT-Hybrid	ResNet-50	L2	5	9	3.402	1.348	0.464	0.039	0.114	0.233
NYUv2	DPT-Hybrid	ResNet-50	L2	10	14	3.732	1.570	0.517	0.024	0.068	0.159
NYUv2	DPT-Hybrid	ResNet-50	L2	16	20	3.935	1.713	0.548	0.018	0.050	0.121
NYUv2	DPT-Hybrid	ResNet-50	berHu	3	7	3.145	1.195	0.416	0.057	0.165	0.303
NYUv2	DPT-Hybrid	ResNet-50	berHu	5	9	3.432	1.370	0.464	0.037	0.112	0.231
NYUv2	DPT-Hybrid	ResNet-50	berHu	10	14	3.758	1.592	0.517	0.023	0.068	0.157
NYUv2	DPT-Hybrid	ResNet-50	berHu	16	20	3.957	1.733	0.548	0.016	0.050	0.199

TABLE III

MONOCULAR DEPTH ESTIMATION RESULTS AFTER ATTACKING WITH UNTARGETED MI-FGSM.

						Lower is better			Accuracy: higher is better		
Dataset	Model	Backbone	Loss	Epsilon	Iterations	RMSE	Abs Rel	Log10	δ_1	δ_2	δ_3
KITTI	FCRN	ResNet-50	L1	3	7	32.477	1.541	0.451	0.075	0.184	0.321
KITTI	FCRN	ResNet-50	L1	5	9	35.392	1.735	0.490	0.054	0.142	0.270
KITTI	FCRN	ResNet-50	L1	10	14	38.061	1.952	0.534	0.035	0.100	0.214
KITTI	FCRN	ResNet-50	L1	16	20	39.704	2.068	0.553	0.028	0.083	0.190
KITTI	FCRN	ResNet-50	L2	3	7	31.057	1.386	0.423	0.126	0.247	0.378
KITTI	FCRN	ResNet-50	L2	5	9	33.559	1.546	0.463	0.090	0.194	0.324
KITTI	FCRN	ResNet-50	L2	10	14	35.724	1.733	0.509	0.054	0.136	0.261
KITTI	FCRN	ResNet-50	L2	16	20	37.661	1.869	0.533	0.040	0.111	0.229
KITTI	FCRN	ResNet-50	berHu	3	7	29.221	1.262	0.406	0.143	0.274	0.409
KITTI	FCRN	ResNet-50	berHu	5	9	31.566	1.414	0.446	0.106	0.221	0.354
KITTI	FCRN	ResNet-50	berHu	10	14	33.762	1.577	0.490	0.072	0.166	0.296
KITTI	FCRN	ResNet-50	berHu	16	20	35.795	1.730	0.518	0.050	0.133	0.257
KITTI	AdaBins	ResNet-50	L1	3	7	16.439	0.587	0.346	0.188	0.413	0.577
KITTI	AdaBins	ResNet-50	L1	5	9	19.459	0.784	0.418	0.135	0.341	0.422
KITTI	AdaBins	ResNet-50	L1	10	14	26.803	1.243	0.534	0.098	0.202	0.297
KITTI	AdaBins	ResNet-50	L1	16	20	31.403	1.607	0.611	0.063	0.123	0.189
KITTI	AdaBins	ResNet-50	L2	3	7	17.845	0.632	0.413	0.145	0.291	0.393
KITTI	AdaBins	ResNet-50	L2	5	9	20.187	0.780	0.474	0.127	0.254	0.342
KITTI	AdaBins	ResNet-50	L2	10	14	25.565	1.085	0.528	0.110	0.222	0.309
KITTI	AdaBins	ResNet-50	L2	16	20	29.073	1.311	0.538	0.104	0.193	0.283
KITTI	AdaBins	ResNet-50	berHu	3	7	17.990	0.630	0.422	0.143	0.277	0.369
KITTI	AdaBins	ResNet-50	berHu	5	9	20.111	0.756	0.474	0.119	0.245	0.343
KITTI	AdaBins	ResNet-50	berHu	10	14	24.975	1.029	0.512	0.112	0.229	0.329
KITTI	AdaBins	ResNet-50	berHu	16	20	28.434	1.250	0.518	0.108	0.211	0.315
KITTI	DPT-Hybrid	ResNet-50	L1	3	7	19.203	0.723	0.357	0.132	0.288	0.310
KITTI	DPT-Hybrid	ResNet-50	L1	5	9	21.182	0.866	0.430	0.099	0.213	0.274
KITTI	DPT-Hybrid	ResNet-50	L1	10	14	29.971	1.960	0.623	0.079	0.157	0.223
KITTI	DPT-Hybrid	ResNet-50	L1	16	20	33.494	2.376	0.687	0.038	0.104	0.187
KITTI	DPT-Hybrid	ResNet-50	L2	3	7	21.424	0.912	0.392	0.165	0.315	0.437
KITTI	DPT-Hybrid	ResNet-50	L2	5	9	23.685	1.089	0.460	0.111	0.235	0.343
KITTI	DPT-Hybrid	ResNet-50	L2	10	14	26.866	1.378	0.544	0.061	0.152	0.245
KITTI	DPT-Hybrid	ResNet-50	L2	16	20	29.778	1.652	0.605	0.052	0.120	0.201
KITTI	DPT-Hybrid	ResNet-50	berHu	3	7	19.225	0.724	0.357	0.225	0.390	0.512
KITTI	DPT-Hybrid	ResNet-50	berHu	5	9	21.114	0.861	0.430	0.163	0.307	0.414
KITTI	DPT-Hybrid	ResNet-50	berHu	10	14	23.965	1.100	0.530	0.089	0.203	0.301
KITTI	DPT-Hybrid	ResNet-50	berHu	16	20	26.320	1.291	0.582	0.076	0.167	0.256
NYUv2	DPT-Hybrid	ResNet-50	L1	3	7	3.214	1.283	0.434	0.029	0.107	0.245
NYUv2	DPT-Hybrid	ResNet-50	L1	5	9	3.451	1.423	0.476	0.019	0.073	0.187
NYUv2	DPT-Hybrid	ResNet-50	L1	10	14	3.702	1.591	0.541	0.009	0.037	0.112
NYUv2	DPT-Hybrid	ResNet-50	L1	16	20	3.751	1.645	0.598	0.006	0.019	0.061
NYUv2	DPT-Hybrid	ResNet-50	L2	3	7	3.143	1.183	0.430	0.056	0.155	0.287
NYUv2	DPT-Hybrid	ResNet-50	L2	5	9	3.346	1.306	0.475	0.040	0.112	0.226
NYUv2	DPT-Hybrid	ResNet-50	L2	10	14	3.564	1.459	0.543	0.025	0.066	0.146
NYUv2	DPT-Hybrid	ResNet-50	L2	16	20	3.565	1.487	0.602	0.017	0.041	0.088
NYUv2	DPT-Hybrid	ResNet-50	berHu	3	7	3.166	1.201	0.428	0.055	0.156	0.287
NYUv2	DPT-Hybrid	ResNet-50	berHu	5	9	3.388	1.334	0.472	0.037	0.113	0.227
NYUv2	DPT-Hybrid	ResNet-50	berHu	10	14	3.610	1.492	0.540	0.022	0.063	0.144
NYUv2	DPT-Hybrid	ResNet-50	berHu	16	20	3.634	1.534	0.599	0.015	0.038	0.087

an accuracy metric, as we increase the perturbations, the RMSE loss keeps increasing. Hence, we cap the ε value to 16 as pointed out in [7]. For iterative attacks, the number of iterations is chosen to be $max(\varepsilon+4, 1.25*\varepsilon)$.

B. Targeted Attacks

Targeted attacks aim to not only cause the input to mispredict when given to the network, but to cause it to mispredict as a certain depth value for a certain object mask. Thus, the goal is not to maximize the prediction error, it is to minimize the prediction error between the current prediction and the targeted depth prediction for an object in the input.

$$\min_{\mathbf{x}^{\text{adv}}-\mathbf{x}} L(f(\mathbf{x}^{\text{adv}};\theta),(1-M)\cdot f_d(x)+(1+\alpha)M\cdot f_d(x)) \quad s.t.\ \|\mathbf{x}-\mathbf{x}^{\text{adv}}\|_\infty < \varepsilon, \tag{4}$$

In this case, we are minimizing the loss between the adversarial input $\mathbf{x}^{\text{adv}}$'s prediction and the ground truth value $\mathbf{y}^{\text{true}}$ at a targeted depth prediction after it has been masked to only focus on a particular object in the input. M is a binary mask $M \in \{1,0\}^{HxW}$ which is derived from semantic segmentation, α is the arbitrary distance the target is to be moved and $f_d(x)$ is the targeted depth prediction.

For this task specifically, we use the segmentation model from DPT which has been trained on Ade20k. The targeted class is chosen and the pixel values of this particular class are made 1 and the rest zero in our segmentation mask M. The α value is the distance by which the object is to be moved. We select a value between {-0.5, 0.5}

V. Evaluation Metrics

In order to evaluate and compare the performance of various depth estimation networks, we leverage the commonly accepted evaluation methods: **RMSE, RMSE log, Abs Rel, and Accuracy Threshold**. These indicators are formulated as:

- **RMSE** $= \sqrt{\frac{1}{|N|}\sum_{i\in N} \| d_i - d_i^* \|^2}$,
- **RMSE log** $= \sqrt{\frac{1}{|N|}\sum_{i\in N} \| \log(d_i) - \log(d_i^*) \|^2}$,
- **Abs Rel** $= \frac{1}{|N|}\sum_{i\in N} \frac{|d_i - d_i^*|}{d_i^*}$,
- **Accuracies:** % of d_i s.t. $\max(\frac{d_i}{d_i^*}, \frac{d_i^*}{d_i}) = \delta < thr$,

where d_i is the predicted depth value of pixel i, and d_i^* stands for the ground truth of depth. Besides, N denotes the total number of pixels with real-depth values, and thr denotes the threshold.

VI. Results

In these experiments, we consider the FCRN network which does not use a transformer architecture as our baseline and compare its robustness to that of AdaBins and DPT. To give context for the results below, we first reproduce state-of-the-art results for each model on clean images from the KITTI and NYUv2 datasets. These baseline results can be found in Table I. As expected, both the vision transformer based networks strongly outperform the baseline fully-convolutional network. Moreover, Adabins and DPT perform similarly on both datasets. A visualization of these predictions on clean images is provided in Figure 5.

In the sections below, we perform a variety of untargeted and targeted attack experiments in both white-box and black-box settings to compare the robustness of these models, as well as the transferability of generated adversaries. We then perform adversarial training of our strongest performing models and further evaluate model robustness.

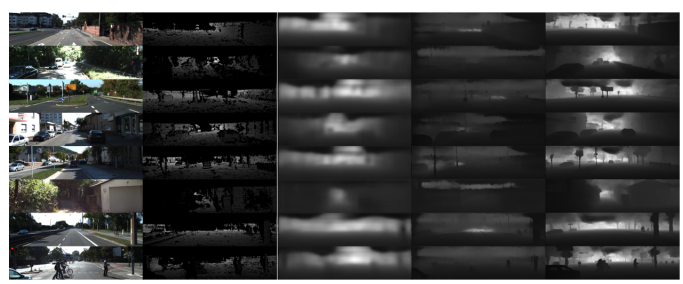

Fig. 5. The figure visualizes the depth maps for all models with KITTI dataset. The columns from Left to Right show: Clean images, Ground-Truth, FCRN results, AdaBins results, and DPT results respectively.

A. Untargeted White-Box Experiments

To begin our analysis, we conduct a large ablation study of PGD and MI-FGSM attacks on all three models while varying the loss, perturbation strength, and iterations used in the attack. Specifically, Table II displays PGD attack results against all three

models while using the L1, L2, or reverse Huber loss and using perturbation magnitudes (epsilon) of 3, 5, 10, or 16. Table III also mimics this ablation study but with the MI-FGSM attack. For ease of comparison, we plot these ablation results FCRN, AdaBins, and DPT in Figures 6, 7, and 8, respectively. We also visualize perturbations from MI-FGSM and their corresponding predictions from each model in Figures 9, 10, and 11.

An interesting trend in these plots, is that when varying epsilon for PGD against the FCRN model, the resulting RMSE value increases nearly linearly. On the other hand, the MI-FGSM results are somewhat stagnant as epsilon increases. Furthermore, this trend does not continue for the AdaBins and DPT models, where MI-FGSM and PGD perform quite similarly across epsilon values. From these ablation studies we can also gather that using the L1 loss during attack results in the strongest adversaries on average.

Using this knowledge, we then plot results for each model being attacked by MI-FGSM with an L1 loss on a single chart, shown in Figure 12. Note that each model's baseline on clean images is given at $\delta = 0$. From this plot we gather that the two vision transformer based networks are in fact more robust to adversarial attack than the fully convolutional model. We attribute this to the transformer's ability to capture more global information through their globally receptive fields, whereas the FCRN model tends to lose feature granularity in its encoding layers and tends to focus on more local relationships. Between the two transformer models, we find that AdaBins tends to slightly outperform DPT across all epsilon values. This suggests that AdaBins' approach of leveraging a transformer as a sort of feature post-processing step, rather than DPT's architecture of a transformer-based encoder, is slightly more robust.

B. Targeted White-Box Experiments

For the targeted attack, we use the DPT segmentation model that is trained on Ade20k dataset. The binary mask M is created by choosing a target class which in our case is a car. We move the target closer and further away by varying α. The respective depth values are change significantly as α changes.
Figure 13 shows depth maps of few images that contain significant number of cars. The car is brought 10% closer in the third column where the depth values of the car are higher than the ground-truth. Similarly we notice the car slowly disappear as we increase α to 50% and 100% in the fourth and the fifth column respectively.
We notice that some of the semantic labels are harder to attack than others, that is some of the categories such as traffic signs and humans are harder to alter than others such as cars, vegetation etc. for the same α values.

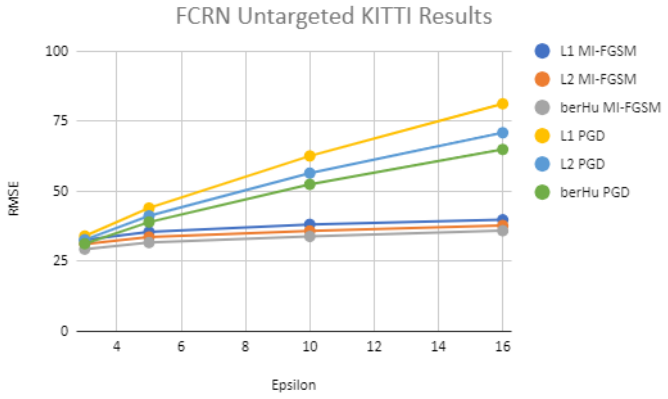

Fig. 6. Plotting untargeted ablation study of FCRN architecture. X axis represents perturbation magnitudes while the legend denotes the attack and loss used.

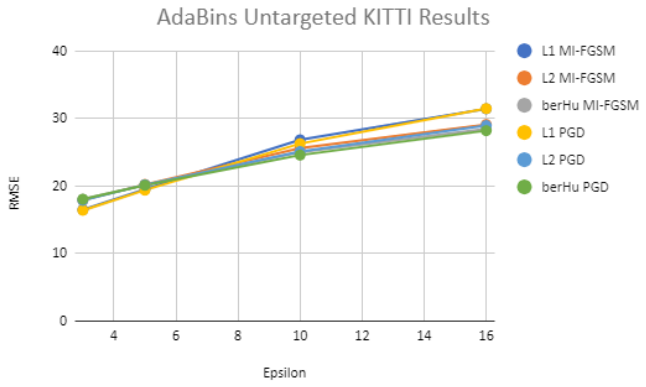

Fig. 7. Plotting untargeted ablation study of AdaBins architecture. X axis represents perturbation magnitudes while the legend denotes the attack and loss used.

C. Adversarial Training Experiments

We also evaluate the effectiveness of adversarial training in the context of depth estimation. To do so, we fine-tune our strongest performing FCRN model for an additional 5 epochs via FGSM-based

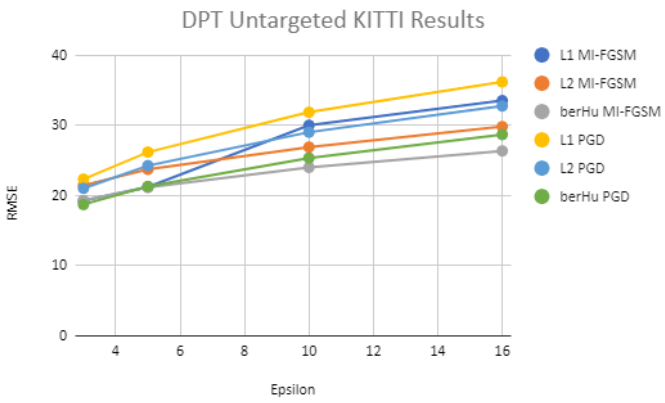

Fig. 8. Plotting untargeted ablation study of DPT architecture. X axis represents perturbation magnitudes while the legend denotes the attack and loss used.

Fig. 9. This figure visualizes adversaries created with MI-FGSM, $delta = 5$, and L1 loss along with corresponding predicted depth maps from FCRN with the KITTI dataset. The columns from Left to Right show: Clean images, Perturbed adversarial images, Ground-truth, Clean predictions from FCRN, and Adversarial predictions respectively.

adversarial training. Note that this is an FGSM [11] attack with a single gradient step using an L1 loss. We perform adversarial training with various perturbation magnitudes, or epsilon, including 3, 5, and 10, resulting in three unique robust models.

These adversarially trained models, along with their baseline non-robust counterpart, are then attacked with MI-FGSM attacks to evaluate the effects of this finetuning. Specifically, we attack with MI-FGSM and L1, L2, and reverse huber losses while varying epsilon and display the results in Tables 14, 15, and 16, respectively. In these plots, the legend denotes the epsilon value used during adversarial training, while the x-axis denotes the epsilon values used during evaluation. We also attack these models with translation invariant perturbations following the Gaussian smoothing technique introduced in [10], which is also denoted in the plot legends.

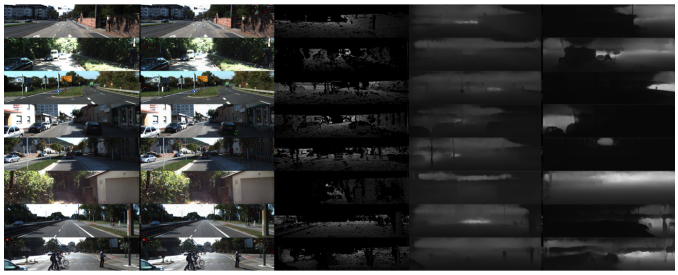

Fig. 10. This figure visualizes adversaries created with MI-FGSM, $delta = 5$, and L1 loss along with corresponding predicted depth maps from AdaBins with the KITTI dataset. The columns from Left to Right show: Clean images, Perturbed adversarial images, Ground-truth, Clean predictions from AdaBins, and Adversarial predictions respectively.

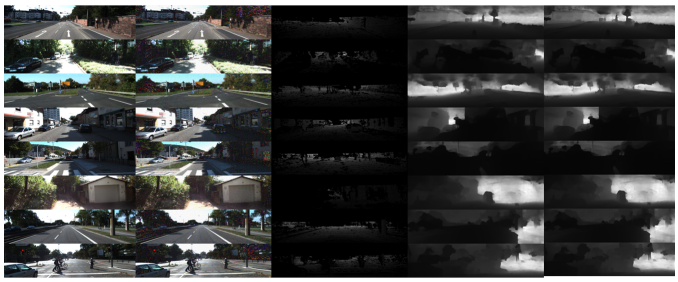

Fig. 11. This figure visualizes adversaries created with MI-FGSM, $delta = 5$, and L1 loss along with corresponding predicted depth maps from DPT with the KITTI dataset. The columns from Left to Right show: Clean images, Perturbed adversarial images, Ground-truth, Clean predictions from DPT, and Adversarial predictions respectively.

Overall, these plots show that adversarial training is quite effective in defending against depth attacks. This is seen in the wide gap between each plot's model that was not adversarially trained and those that were. It is also interesting to note that increasing epsilon, or the perturbation magnitude, during adversarial training does not necessarily increase model robustness. In some cases, even, there is a strong diminishing returns effect when $\delta >= 5$. Furthermore, we see that one does not need to adversarially train with a high epsilon to achieve robustness against attacks with large magnitudes. For example, a model trained with $\delta = 3$ achieves roughly the same robustness when attacked with $\delta = 3$, $\delta = 5$, or $\delta = 10$. Lastly, we determine that translation invariant perturbations are not necessarily stronger against adversarially trained models. This is in contrast to the original work [10], which shows that this gradient smoothing increases performance against robust models.

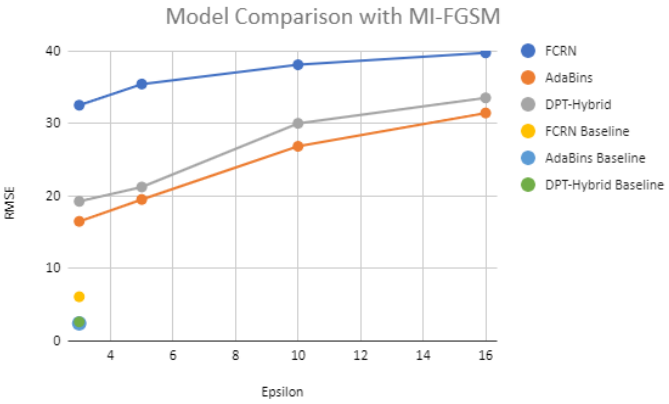

Fig. 12. Direct comparison of model robustness against an MI-FGSM attack across varying perturbation magnitudes δ.

Fig. 13. Targeted attack for DPT on KITTI dataset with MIFGSM attack. The columns from Left to Right: Clean Image, clean prediction, 10% closer, 50% farther, 100% farther

D. Un-targeted Black-Box Experiments

In this section, we perform cross-model black-box experiments to further evaluate the robustness of these models, as well as the transferabilty of their generated adversaries. In Table IV, the left-hand column denotes the model used to generate adversaries. Note that in all experiments here we apply an untargeted MI-FGSM attack with L1 loss and $\delta = 5$ on the first 100 images of the KITTI depth validation set, then save this subset of images for black-box application. Then, the righter most columns in Table IV represent which model is being attacked by adversaries generated by the model on the left in the same row. Naturally, cells along the diagonal in this table represent white-box attacks.

Overall, the results in this table back the intuition that the vision transformer based models are more robust. For example, adversaries generated with the FCRN model and used to attack AdaBins and DPT are not very effective. More specifically, the RMSE drops nearly $4\times$ when comparing the white-box attack against FCRN with the black-box attacks against AdaBins and DPT. We also continue to see that adversarial training is quite effective for defending against depth attacks. Furthermore, adversaries

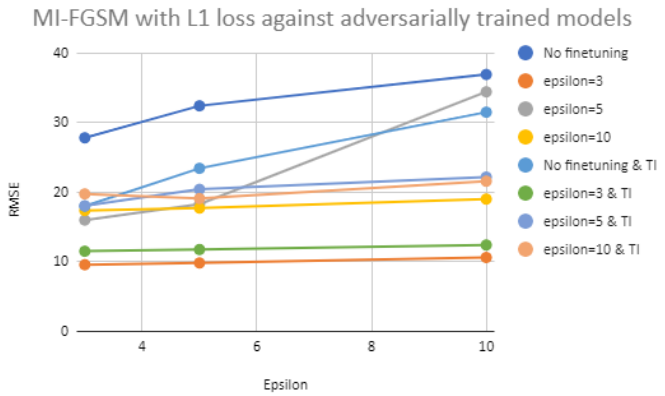

Fig. 14. Attacking adversarially trained models with an MI-FGSM attack and L1 loss. X axis represents epsilons used during evaluation. Epsilons in the legend represent those used during adversarial training. TI in the legend signifies attacking this model with a translation invariant perturbation [10]

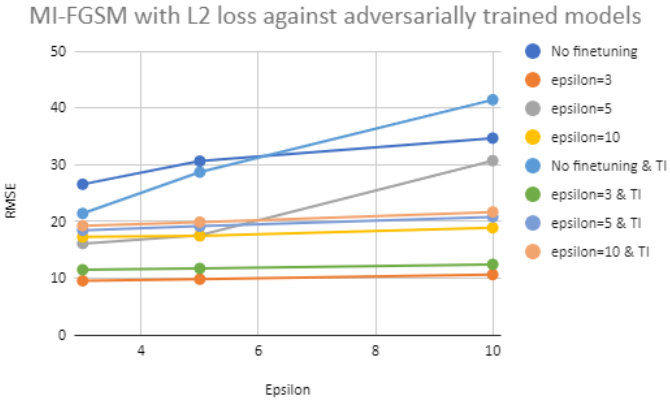

Fig. 15. Attacking adversarially trained models with an MI-FGSM attack and L2 loss. X axis represents epsilons used during evaluation. Epsilons in the legend represent those used during adversarial training. TI in the legend signifies attacking this model with a translation invariant perturbation [10]

generated with the transformer networks are highly effective even against an adversarially trained FCRN model. Lastly, we see that adversarially training an FCRN model does not necessarily increase its resulting adversaries' transferability to these other networks.

VII. Conclusion

In this work, we implement and evaluate a baseline Fully Convolutional Residual Network [4], two state-of-the-art Vision Transformer based approaches (AdaBins and DPT) [5], [6], and adapted un-targeted gradient-based [9], [11], [12] adversarial

TABLE IV

CROSS-MODEL BLACK-BOX EXPERIMENTS PERFORMED USING KITTI IMAGES AND AN MI-FGSM ATTACK WITH L1 LOSS AND $\delta = 5$. LEFT-HAND SIDE COLUMN DENOTES MODELS USED TO GENERATE ADVERSARIES. RIGHT SIDE COLUMNS DENOTE MODELS BEING EVALUATED IN THE BLACK-BOX SETTING. CELLS ALONG THE DIAGONAL ARE NATURALLY WHITE-BOX ATTACKS. ALL RESULTS SHOWN ARE RMSE VALUES.

	Evaluated Models			
Generation model	FCRN	Adabins	DPT	FCRN_adv
FCRN	35.39	8.32	9.07	17.38
AdaBins	20.39	19.45	18.92	17.59
DPT-hybrid	20.26	18.76	19.38	17.47
FCRN_adv	20.023	9.02	9.45	16.91

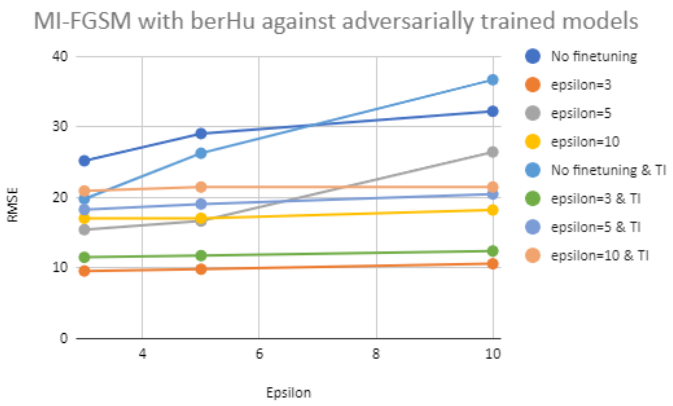

Fig. 16. Attacking adversarially trained models with an MI-FGSM attack and Reverse Huber loss. X axis represents epsilons used during evaluation. Epsilons in the legend represent those used during adversarial training. TI in the legend signifies attacking this model with a translation invariant perturbation [10]

attacks to compare the robustness of depth estimation networks.The targeted versions of these attacks, with the use of segmentation masks, are also implemented by moving an arbitrary target class nearer and farther away. The adversarial robustness of these models is then tested in white-box and cross-model black-box settings. Lastly, we also perform adversarial training of our FCRN model to evaluate the effectiveness of this technique in the context of depth estimation defense.

Our extensive experiments show that the transformer based networks are significantly more robust than their fully-convolutional counter-part. This can be attributed to the models' ability to capture Global Receptive fields at each stage in the Vision Transformer [1] which leads to globally coherent, fine-grained predictions. Furthermore, we find that Adabins which leverages a transformer for post-processing of the feature maps is more robust to adversarial noise than DPT, which leverages a transformer directly in its encoder.

The targeted attack demonstrates that these imperceptible additive perturbations to selective regions of the image can change the perceived geometry of the scene. Moreover, we notice that each category exhibits a different level robustness to adversarial attacks as pointed out by [14]. Targeted attacks in the depth estimation space seems to be a relatively less explored with a lot of potential for future research, such as altering the predictions to fit a preset scene, symmetrically inverting the depth predictions, horizontal and vertical translation of instances etc.

Our cross-model black-box analysis sheds further light on the robustness of the vision transformer networks, showing that adversaries generated with these networks are more transferable than those generated with fully-convolutional models. We hope this work can provide a baseline and guidance for adversarial attacks research on transformer-based monocular depth estimation techniques.

REFERENCES

[1] An Image is worth 16x16 words: Transformers for Image recognition at scale, 2020. Dosovitskiy, Alexey and Beyer, Lucas and Kolesnikov, Alexander and Weissenborn, Dirk and Zhai, Xiaohua and Unterthiner, Thomas and Dehghani, Mostafa and Minderer, Matthias and Heigold, Georg and Gelly, Sylvain and Uszkoreit, Jakob and Houlsby, Neil. arXiv preprint arXiv:2010.11929

[2] Geiger, Andreas Lenz, Philip Urtasun, Raquel. (2012). Are we ready for autonomous driving? The KITTI Vision Benchmark Suite. Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 3354-3361. 10.1109/CVPR.2012.6248074.

[3] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In ECCV, 2012

[4] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, Nassir Navab. (2016). Deeper Depth Prediction with Fully Convolutional Residual Networks.
[5] Shariq Farooq Bhat, Ibraheem Alhashim, Peter Wonka. (2020). AdaBins: Depth Estimation using Adaptive Bins.
[6] René Ranftl, Alexey Bochkovskiy, Vladlen Koltun. (2021). Vision Transformers for Dense Prediction.
[7] Ziqi Zhang, Xinge Zhu, Yingwei Li, Xiangqun Chen, Yao Guo. (2020). Adversarial Attacks on Monocular Depth Estimation.
[8] Alwyn Mathew, Aditya Prakash Patra, Jimson Mathew. (2020). Monocular Depth Estimators: Vulnerabilities and Attacks.
[9] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, Adrian Vladu. (2019). Towards Deep Learning Models Resistant to Adversarial Attacks.
[10] Yinpeng Dong, Tianyu Pang, Hang Su, Jun Zhu. (2019). Evading Defenses to Transferable Adversarial Examples by Translation-Invariant Attacks.
[11] Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy. (2015). Explaining and Harnessing Adversarial Examples.
[12] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, Jianguo Li. (2018). Boosting Adversarial Attacks with Momentum.
[13] Eric Wong, Leslie Rice, J. Zico Kolter. (2020). Fast is better than free: Revisiting adversarial training.
[14] Alex Wong, Safa Cicek, Stefano Soatto. (2020). Targeted Adversarial Perturbations for Monocular Depth Prediction