A Implementation Details

For training the NeRF models for objects in simulation and in the real world, we employ the default configurations in the original NeRF paper [12] using a PyTorch implementation [35]. For the cGAN model, we use a U-Net architecture that employs Conv-Norm-ReLU blocks as both encoder and decoder layers [36]. Both the encoder and the decoder use 7 layers of convolutional blocks. RGB, depth, and reference background images are encoded using separate decoders that have the same architecture, where depth is treated as grayscale.

B Network Hyperparameters

The configuration and hyperparameters used in the NeRF models are shown in Table 4.

Environment	Simulation	Real-world	Real-world deployed
Batch size	1024	4096	1024
Optimiser	Adam	Adam	Adam
No. of layers in coarse network	8	8	8
Channels per layer in coarse network	256	256	256
No. of layers in fine network	8	8	8
Channels per layer in fine network	256	256	256
Learning rate	5e-4	5e-4	5e-4
Exponential learning rate decay (no. of steps)	500,000	250,000	250,000
Number of coarse samples per ray	64	64	64
Number of additional fine samples per ray	128	128	64

Table 4: NeRF training configuration

The architecture of the cGAN model is shown in Table 5.

Layer no.	Type
1	Conv
2	ReLU
3	Conv
4	Norm
5	ReLu
6	Conv
7	Norm
8	ReLU
9	Conv

Layer no.	Type
10	Norm
11	ReLU
12	Conv
13	Norm
14	ReLU
15	Conv
16	Norm
17	ReLU
18	Conv

(a) Encoder architecture

Layer no.	Type
1	ReLU
2	TransConv
3	Norm
4	ReLU
5	TransConv
6	Norm
7	Dropout
8	ReLU
9	TransConv

Layer no.	Type
10	Norm
11	Dropout
12	ReLU
13	TransConv
14	Norm
15	ReLU
16	TransConv
17	Norm
18	ReLU

Layer no.	Type
19	TransConv
20	Norm
21	ReLU
22	TransConv
23	Tanh

(b) Decoder architecture

Table 5: Architecture of the cGAN encoder and decoder

The hyperparameters used in training the cGAN models are shown in Table 6.

Hyperparameter	Value
Batch size	1
Learning rate	2e-4
Optimiser	Adam

Table 6: NeRF training configuration

C Object Dataset

The objects used in the simulation experiments are shown in Figure 5, and the objects used in the real-world experiments are shown in Figure 6.

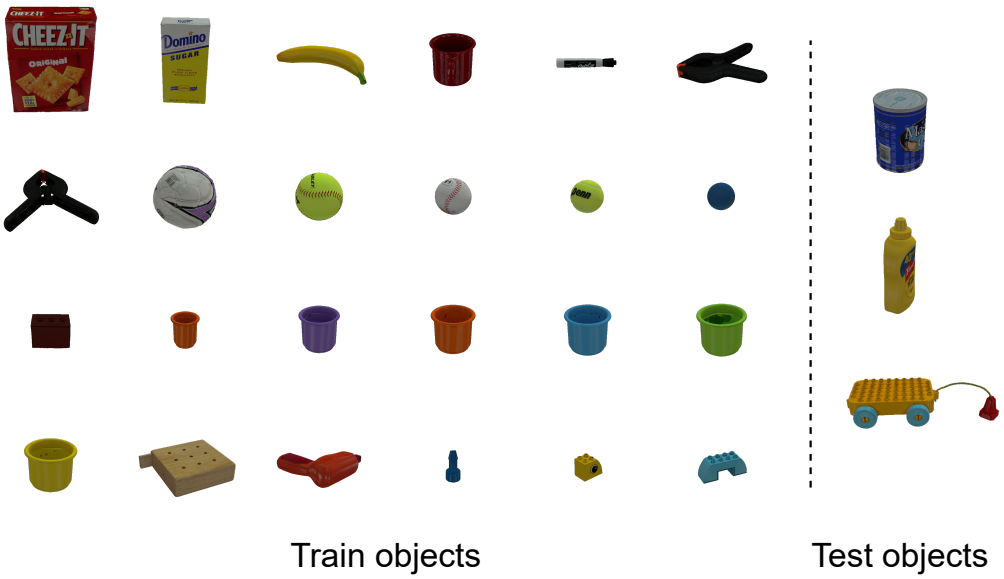


Figure 5: Objects used in simulation experiments.

Figure 6: Objects used in real-world experiments.

D Additional Results

Additional qualitative results for the novel view test sets are shown in Figure 7 and Figure 8.

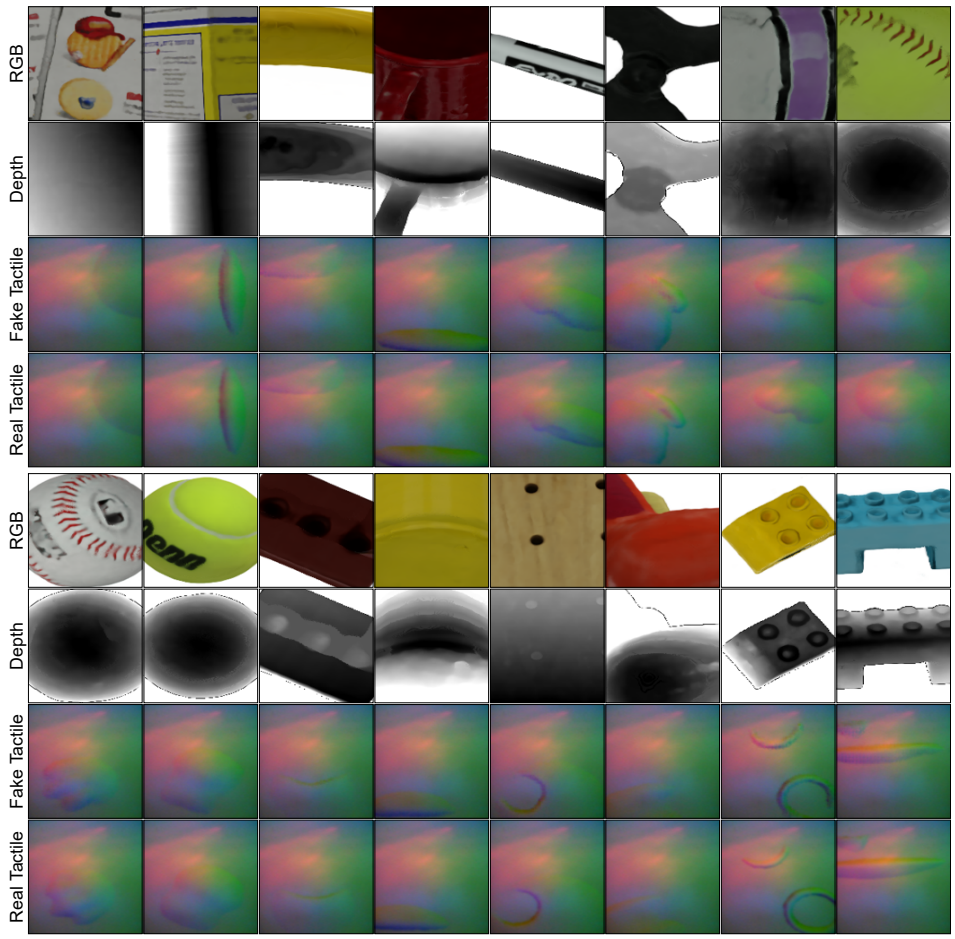


Figure 7: Qualitative results on the novel view test set in simulation. Real tactile refers to ground-truth (simulated) tactile images. Fake tactile refers to tactile images generated by the cGAN model.

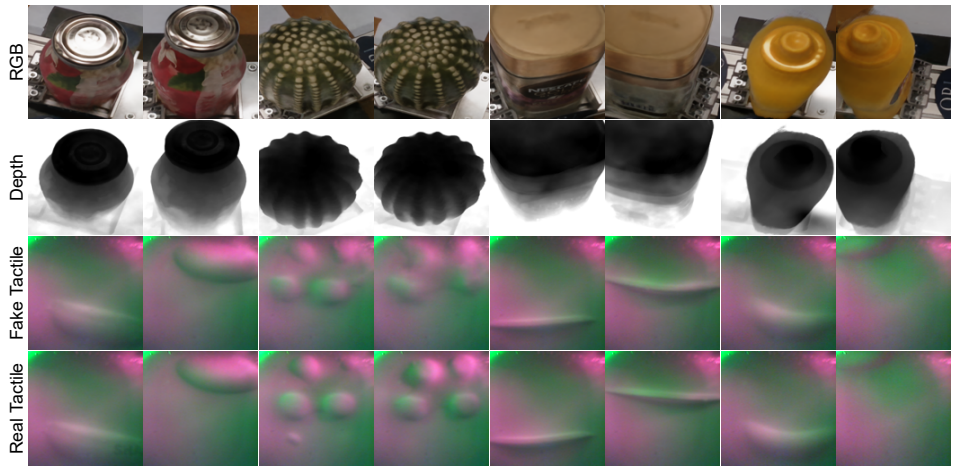
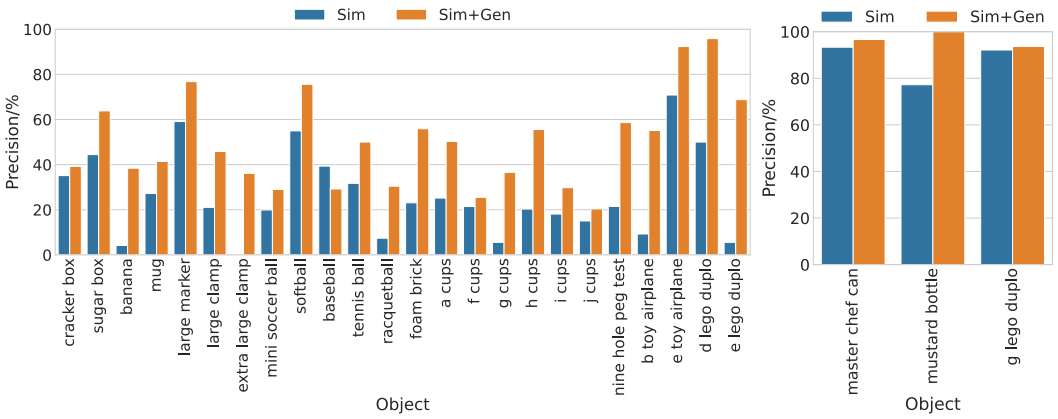


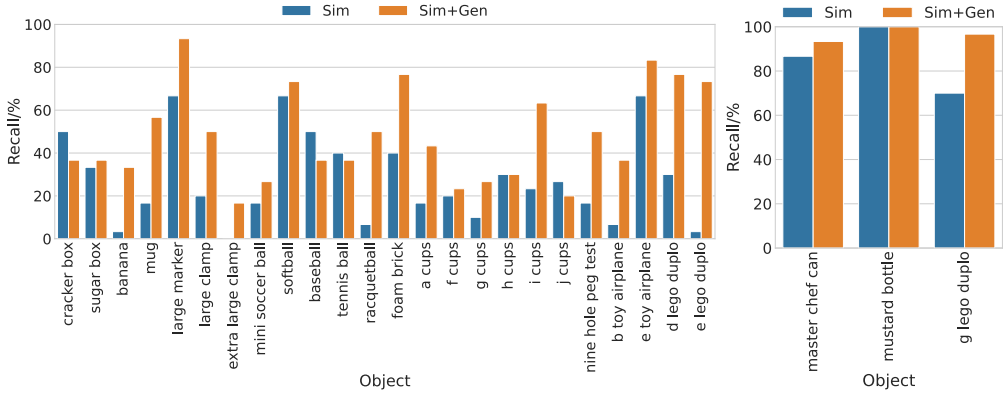
Figure 8: Qualitative results on the novel view test set in the real world. Real tactile refers to ground-truth tactile images collected in the real-world experiments. Fake tactile refers to tactile images generated by the cGAN model.

E Additional Classification Results

Additional precision-recall analyses of the results of classification are shown in Figures 9, 10, and 11.

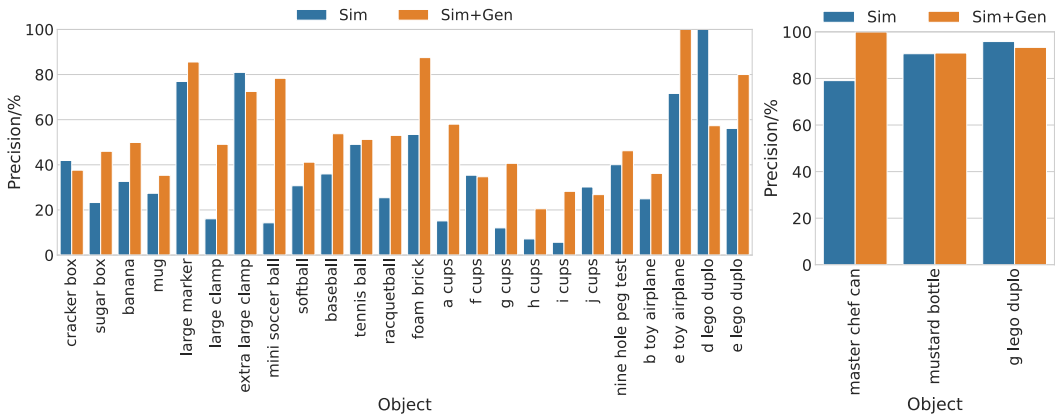


(a) Precision of classification. Left: train objects. Right: test objects.

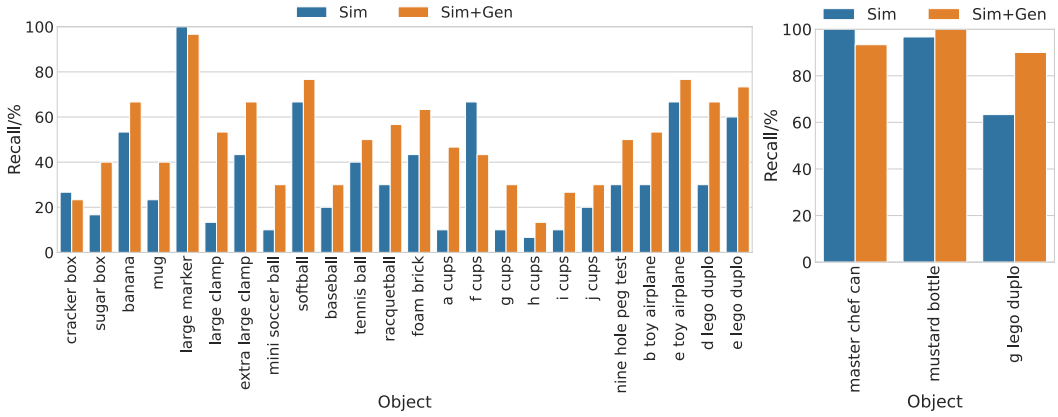


(b) Recall of classification. Left: train objects. Right: test objects.

Figure 9: Change in classification results for each object after augmenting the tactile dataset with generated tactile images for the **simulated Digit** sensor. For each object, the blue bar indicates the value of the metric before augmentation, and the orange bar indicates the value of the metric after augmentation. As can be seen, the precision and recall rate for most of the objects increased after augmenting the dataset, leading to an increase in overall classification success. This points to the usefulness of the generated tactile images for this downstream task. The reason for the decrease in the precision and recall rates for some objects (e.g. baseball) is most likely due to the fact that there were multiple objects of similar shapes. This could have made the generation of accurate tactile images more difficult, thus negatively impacting classification success. It should be noted that the classification experiments for train objects and test objects are conducted separately.

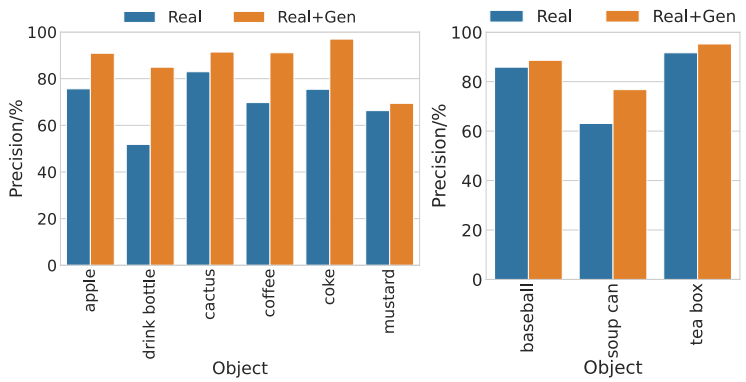


(a) Precision of classification. Left: train objects. Right: test objects.

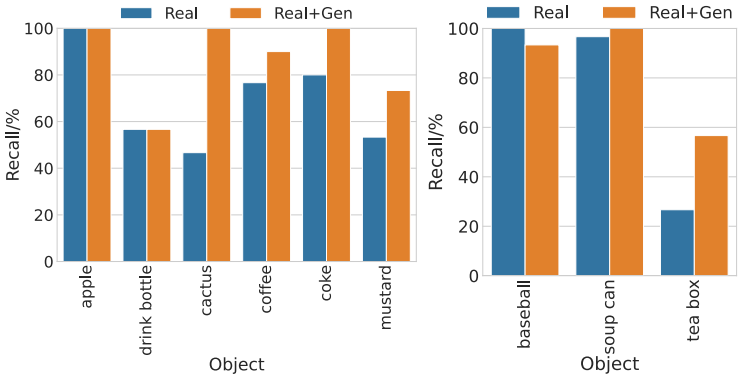


(b) Recall of classification. Left: train objects. Right: test objects.

Figure 10: Change in classification results for each object after augmenting the tactile dataset with generated tactile images for the **simulated OmniTact** sensor. Similar to the results for the Digit dataset, for each object, the blue bar indicates the value of the metric before augmentation, and the orange bar indicates the value of the metric after augmentation. From the figure, it can be seen that the classification metrics for most objects have increased, pointing to the usefulness of the generated tactile image dataset. Further, the cGAN model is only trained on a small fine-tuning dataset for the OmniTact sensor. Thus, this points to the potential of the proposed approach in leveraging a different tactile dataset for pre-training, and transferring the learned model to a new tactile sensor. It should also be noted that the classification experiments for train objects and test objects are conducted separately.



(a) Precision of classification. Left: train objects. Right: test objects.



(b) Recall of classification. Left: train objects. Right: test objects.

Figure 11: Change in classification results for each object after augmenting the tactile dataset with generated tactile images for the **real Digit** sensor. The blue bar indicates the value of the metric before augmentation, and the orange bar indicates the value of the metric after augmentation. As seen from the figure, for most objects, the precision and recall metrics have increased after augmenting the dataset with generated tactile images. This points to the potential of the proposed approach in generating realistic tactile images useful for downstream tasks. It should also be noted that the classification experiments for train objects and test objects are conducted separately.

Dataset	Accuracy/%↑
Sim	27 ± 0
Sim + T-N	47 ± 0
Sim + Lee	20 ± 1
Sim + Cycle	24 ± 1

(a) Simulated Digit

Dataset	Accuracy/%↑
Sim	34 ± 2
Sim + T-N	50 ± 1
Sim + FS	30 ± 2

(b) Simulated OmniTact

Dataset	Accuracy/%↑
Real	69 ± 5
Real + T-N	86 ± 1
Real + Lee	71 ± 4
Real+ Cycle	72 ± 2

(c) Real-world

Table 7: Classification results on train objects seen by the cGAN model in training.

F Classification Dataset Details

The details of the datasets for the example tactile classification task are shown in Table 8 for the train objects that were used in cGAN training, and in Table 9 for the test objects. The experiments are conducted separately for the train objects and test objects. The datasets are all balanced across the objects, and for each object, the datasets include 10 simulated/real images and 50 generated images for training, and 10 test images.

Experiment	Simulation (Digit)	Simulation (OmniTact)	Real-world
No. of classes	24	24	6
Training set size (sim/real)	240	240	60
Training set size (sim/real + gen)	1440	1440	360
Testing set size	240	240	60

Table 8: Details of classification datasets for train objects

Experiment	Simulation (Digit)	Simulation (OmniTact)	Real-world
No. of classes	3	3	3
Training set size (sim/real)	30	30	30
Training set size (sim/real + gen)	180	180	180
Testing set size	30	30	30

Table 9: Details of classification datasets for test objects