

Figure 8: PercepSkin block diagram for self-supervised learning of skin representations. Our approach follows the student-teacher framework and loss functions used in self-distillation. However, we adapt the transformer input tokenization to accommodate time-series Xela data.

494 5.1 PercepSkin self-supervision details

495 5.1.1 Training details

496 We train PercepSkin on 8 Nvidia A-100 (80G) GPUs. To monitor learning, we use reconstruction
 497 online probe and classification via linear probing. We use AdamW optimizer and use a linear rampup
 498 followed by a cosine schedule as the learning scheduler. Further, we find that tuning momentum
 499 value as well as the weight decay factor was important in observing training convergence. Additional
 500 information of hyperparameters is detailed in Table 2.

Architecture	ViT-Tiny (adapted)
Embedding dim	192
EMA decay	[0.994, 1.0]
LR	1e-4
Batch size	64

Table 2: Training hyperparameters for PercepSkin. All models run for 500 epochs with optimizer AdamW, a weight decay cosine schedule from 0.04 to 0.4, and a learning rate warmup of 30 epochs.).

501 5.1.2 Architecture details

502 Our encoder model is a modified version of Vision Transformers [43]. Specifically, we adapt the
 503 tokenization of the time-series Xela with sensor pose data. After flattening the 3D-axis magnetic
 504 reading per magnetometer (368) and concatenating their corresponding pose in chunks of 0.1 second,
 505 the inputs $x \in \mathbb{R}^{10 \times 368 \times 6}$ are tokenized through a linear projection to the dimension d of the
 506 representation $f_{linear}(x) \in \mathbb{R}^{368 \times d}$. We use a tiny model with $d = 192$. We add a learnable
 507 embedding to identify different types of xela pads (palm, phalanges and fingertips). Then, we
 508 construct different cropped view of the data, two global views and eight local views. We mask sensor

data from contiguous blocks by removing those sensors from the input. For the local view we retain between 10% and 40% of the tactile signal, whereas for the global views we retain 40% to 100%. An illustration of the masking and diagram block of the pipeline for self-supervised learning of Xela representations is shown in Figure 8.

The student and teacher share the same encoder and projector head architecture, both initialized with the same weights. The projector head corresponds to a 3-layer MLP with an output dimension of $k = 65536$. We use the projection head for the proxy prediction task to distill knowledge to match output distributions over k dimensions between student and teacher networks. The student network is updated via back-propagation, while the teacher network is updated at a lower frequency via exponential moving average (EMA) on the student weights. We pass the global and local views to the student encoder, while the teacher only has access to the global views. The register tokens from global/local views are passed through the projection head. For the teacher only, the output is also centered and sharpened via softmax normalization.

5.2 Additional task details

We provide additional information about the decoder architectures for each task, as well as additional results to highlight the performance on downstream tasks when using frozen or fine-tuned PercepSkin representations. Also, please refer to Table 3 for details on labeled data curation for evaluation tasks.

Task	Dataset	Size	Collector	Label
Force estimation	Normal load (indenter: sphere, flat)	50k datapoints	Robot	3-axis force
Pose estimation	Object sliding	108 trajectories	Human	Object pose $\mathbf{SE}(2)$
Joystick state estimation	Joystick motion	817 trajectories	Human	Normalized roll, pitch, yaw
Plug insertion	Demonstrations	100 trajectories	Human	Absolute EE pose

Table 3: Datasets for evaluating PercepSkin representations on downstream tasks.

5.2.1 Force estimation

PercepSkin features are pooled via attentive pooling to obtain a full-hand representation $z_{hand} \in \mathbb{R}^d$. The force decoder consist of shallow 2-layer MLP with 3 outputs regressing to normalized force for each axis.

In Figure 9 we illustrate the data protocol followed for force estimation, which we note is different from the protocol that is usually followed for force characterization of tactile sensors. We note that we indent the tactile sensor pads at both, positions on top of the sensor as well as positions in between magnetometer locations, while choosing these positions randomly. This results in cases where the probe may slide and present slightly uneven force outputs. Specifically, in figure 9(b) we note that PercepSkin predicts the correct normal forces, while accumulation (mean) of normal forces from the magnetometers over the sensor pad results in inconsistent force outputs compared to ground truth.

In Figure 10, we present the correlation metrics between ground truth and predicted forces on test data for decoders trained with a 33% data budget. The results show that end-to-end training leads to overfitting, resulting in poor generalization to unseen strokes and essentially random normal force predictions. In contrast, using PercepSkin (frozen) representations yields better fitting, which can be further improved by adapting these representations to in-domain data.

In Figure 11, we present a comparison between ground-truth testing strokes (normal loading sequences) and their reconstructed counterparts, obtained by passing Xela data through the frozen force decoder to recreate the sequences. The forces estimated via PercepSkin (frozen) are able to capture increasing/ decreasing changes in the normal loading, as opposed to the end-to-end model. Shear from skin representations is not as accurate as normal force prediction, but the trend of the tangential forces matches the ground truth.

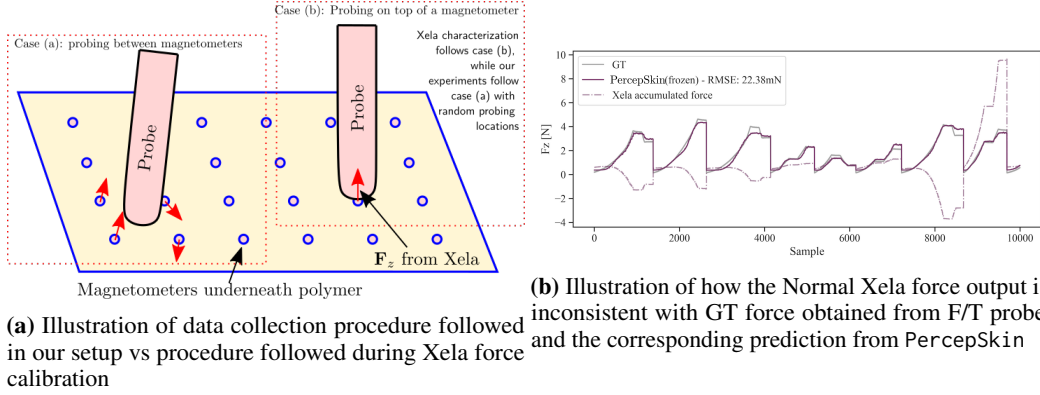


Figure 9: Illustration of data collection protocol followed for Force estimation with Xela sensors

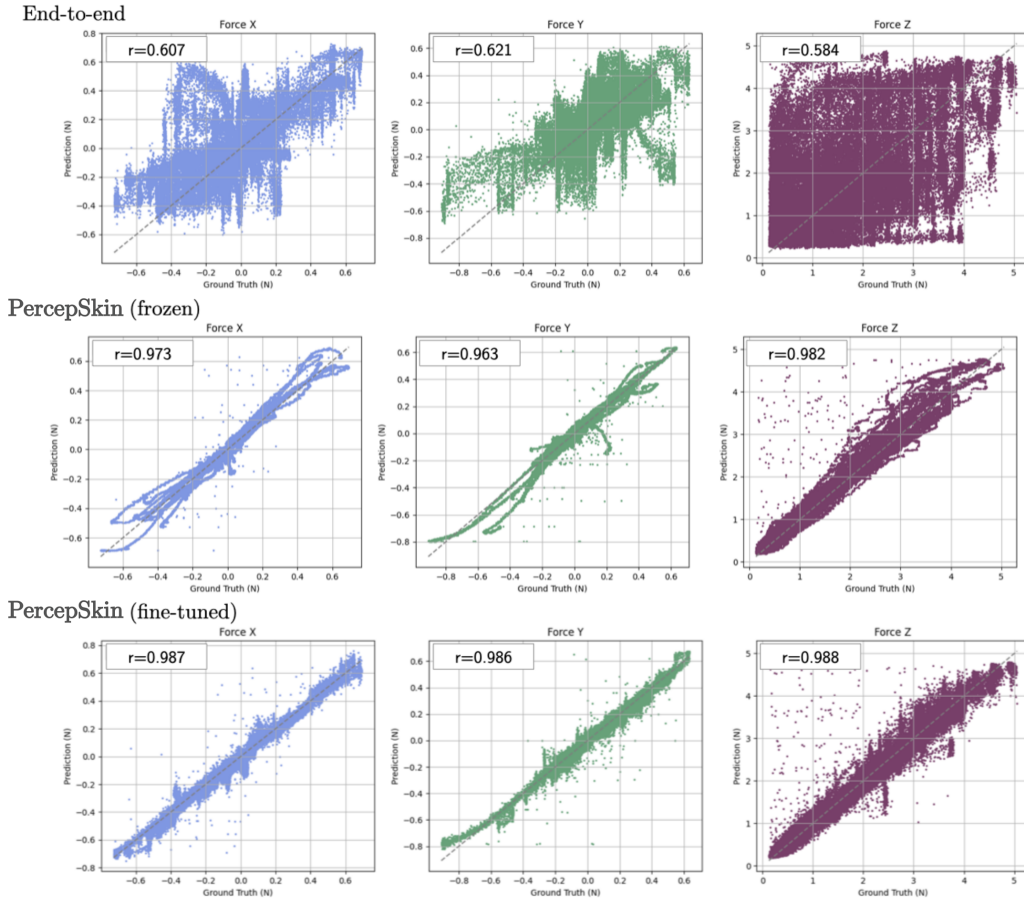


Figure 10: Correlation between ground truth and predicted forces on unseen normal loading with an indenter on Xela sensors.

5.2.2 Joystick state estimation

For this task, we highlight that when we train decoders using pretrained representations as the input, the convergence rate of the validation RMSE is significantly higher (see Figure 12) than training the decoder using raw observations through uninitialized models. Specifically observe that PercepSkin (fine-tuned) is able to reach performance on par with end-to-end pretrained model within 12.9k optimization steps.

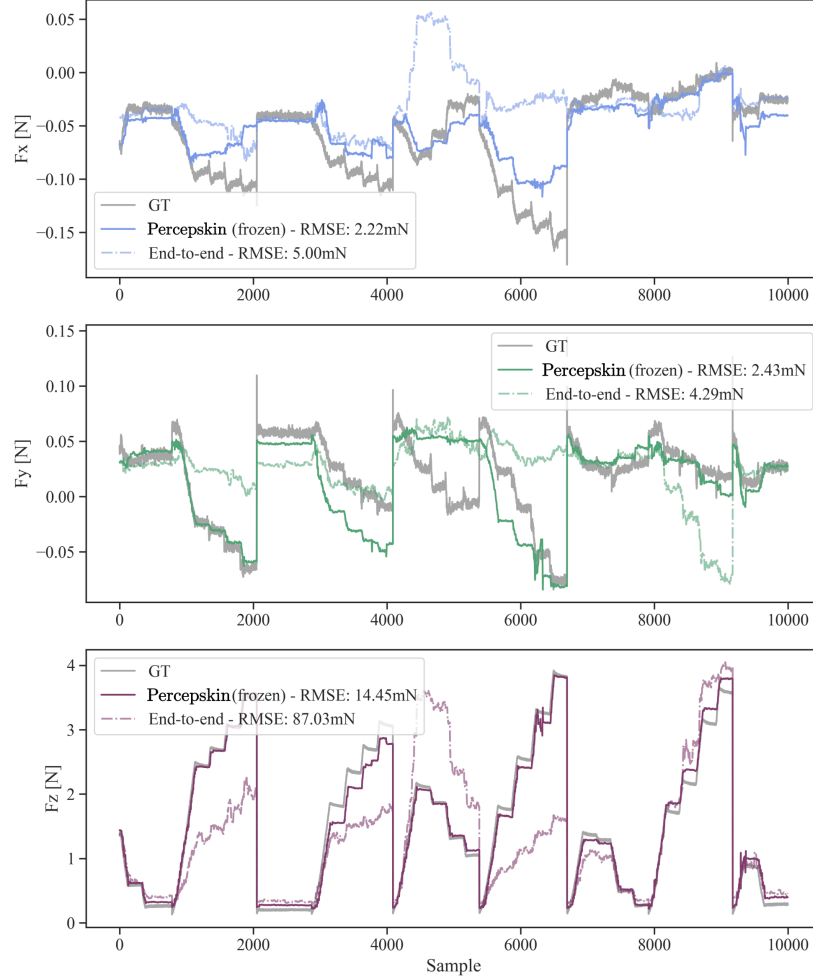


Figure 11: Ground truth tangential and normal force from test strokes with flat indenter (gray) and force sequence reconstruction from PercepSkin (frozen) and end-to-end model.

5.2.3 Pose estimation

In this task, we aim to predict the object pose over 1-second trajectories. Xela observations at 100Hz are converted into tactile representation tokens at the output frequency using PercepSkin in a cascaded manner. Following attentive pooling, a single-layer transformer block is applied to reason about the 1-second context window of full-hand tactile features.

Figure 13 compares ground-truth test pose sequences with their reconstructed counterparts, obtained from task models trained on 100% and 33% of the available data. The results show that fine-tuning PercepSkin on the full dataset yields higher accuracy in estimating object pose changes over time compared to traditional end-to-end approaches. Moreover, even with a drastic reduction in labeled samples (to 33%), the model still achieves relatively good performance, particularly in tracking translation changes. Furthermore, for this task, we also visualize that this task requires *full-hand* sensing. For instance, in Figure 14, we observe that when we use PercepSkin by removing palm sensing on the Xela hand results in $\geq 10\%$ drop in pose tracking performance.

5.2.4 Policy learning (plug insertion)

For this task, we use a transformer decoder to predict action sequences given camera and tactile observations. Figure 15 illustrates the architecture of the transformer decoder used in this work.

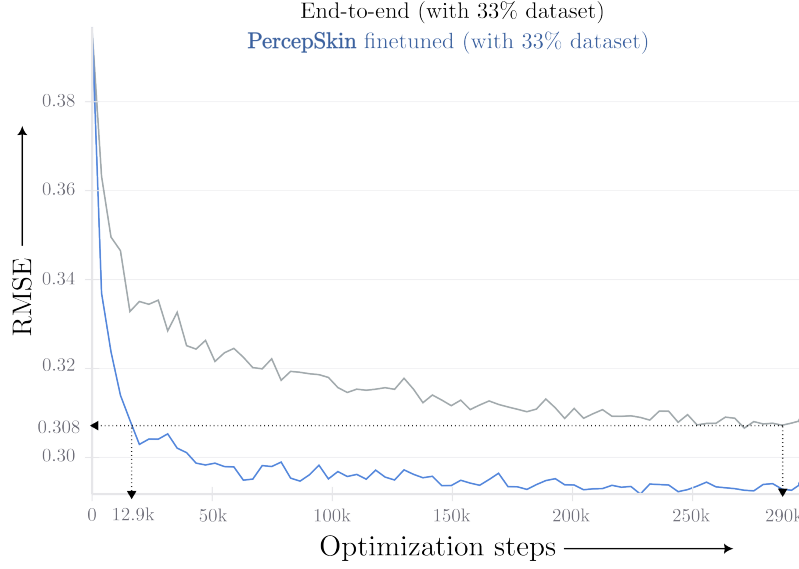


Figure 12: Validation RMSE convergence rates between PercepSkin fine-tuned and PercepSkin end-to-end: We find that PercepSkin fine-tuned allows the model to generalize and learn the patterns required to infer joystick states significantly faster during training.

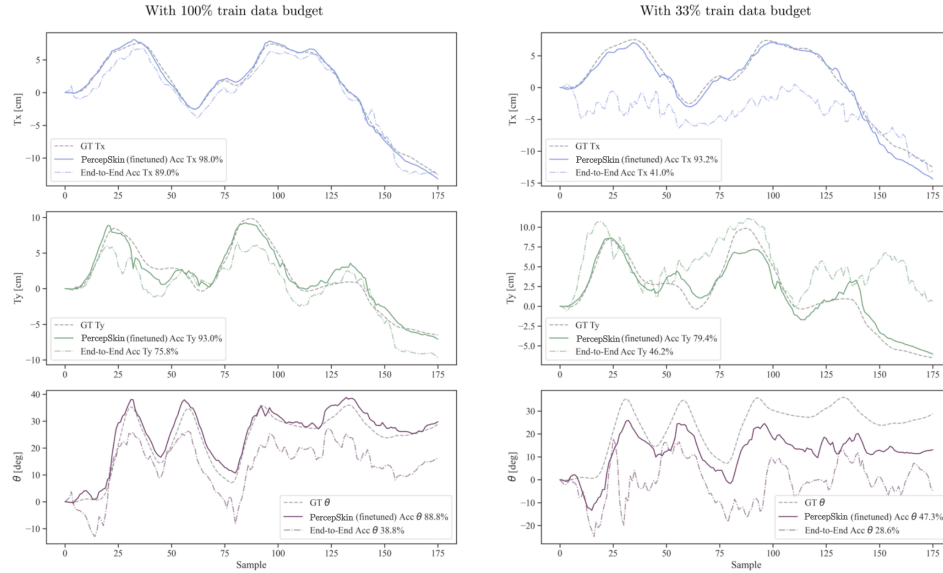


Figure 13: Ground truth pose sequence for object in test set and reconstructed trajectory via end-to-end and PercepSkin (finetuned) representations. (left) Task decoders trained with 100% of train data budget, corresponding to 108 sequences. (right) Task decoders trained with 33% of train sequences.

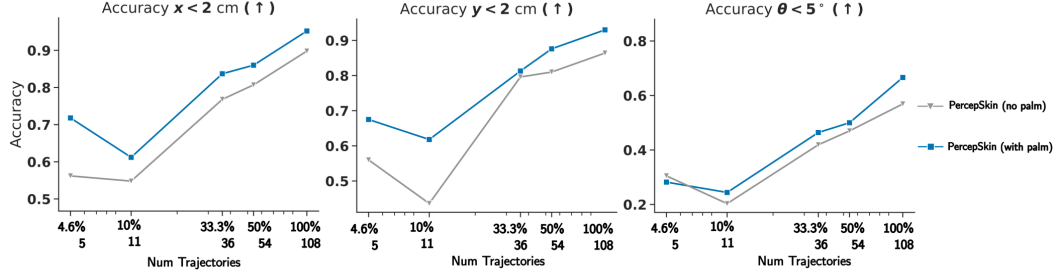


Figure 14: Comparison of pose estimation accuracy of PercepSkin with and without palm sensing.

570 Images are encoded using a Resnet18 CNN, which are trained from scratch to produce image features,
 571 while the tactile observations are processed through PercepSkin. Further, a learnable token (CLS
 572 / action) token is also concatenated with the observation tokens. After processing through the
 573 transformer, we extract the action token, which is then passed into a small 2-layer MLP to predict a
 574 sequence of actions. For this task, follow an *receding-horizon* control approach, where we choose a
 575 prediction action sequence length of 16, of which 8 actions are executed, given only the observations
 576 from the current timestep.

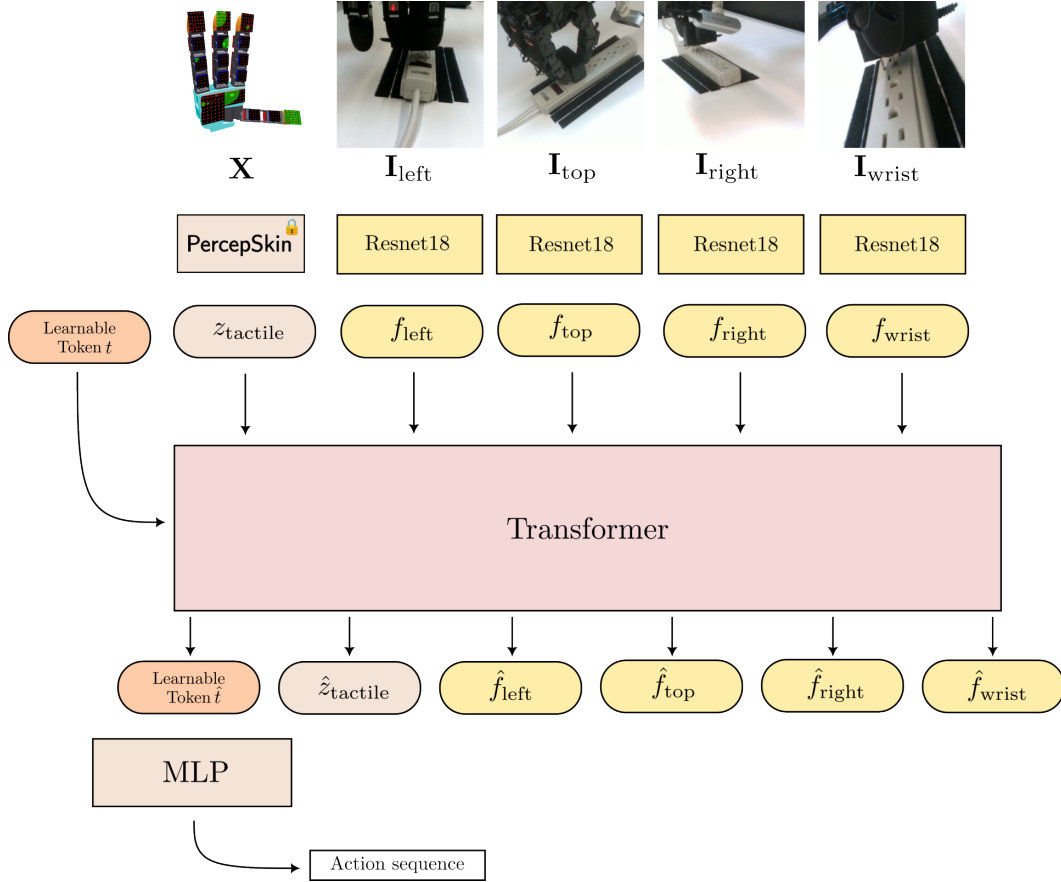


Figure 15: Illustration of the policy architecture: We use a transformer to fuse information from visual and tactile modalities, through the use of a learnable action token, which is then used to subsequently predict action sequences.