

## A Theia Model Architecture

**Backbone.** We use the DeiT-Tiny, DeiT-Small, and DeiT-Base models [39] as our backbone architectures. We keep the [CLS] token in the model and in the forward pass, but there is no supervisory signal provided for it. As a result, the [CLS] token serves as a “register token” [24], which provides some benefits for learning high quality representations. We train Theia from scratch (no pre-trained DeiT [39] weights are applied).

**Feature Translators.** The feature translators are composed primarily of CNNs, with a linear layer appended at the end to match the teacher’s representation dimension. Pure linear transforms might not be able to map Theia-representations to all three teacher representations well, resulting in a failure of learning (See Table 6a). Thus, we use three CNN layers to account for the fact that each teacher model’s representations are very different from one another. Details are listed in Table 4, where we show the architectural details of the translators used for our (student, teacher)-feature pairs.

Table 4: Feature Translator configurations

Student $d_s \times 14 \times 14$ (Theia backbone=ViT-T, -S, -B) to Teacher $d_t \times 16 \times 16$ (CLIP, DINOv2, ViT)
ConvTranspose2d( $d_s, d_s$ , kernel_size=3, stride=1, output_padding=0)
LayerNorm
Conv2d( $d_s, d_s$ , kernel_size=3, padding=1)
ReLU+LayerNorm
Conv2d( $d_s, d_s$ , kernel_size=3, padding=1)
ReLU+LayerNorm
Flatten and Linear( $d_s, d_t$ )
Student $d_s \times 14 \times 14$ to Teacher $d_t \times 64 \times 64$ (SAM and Depth-Anything)
ConvTranspose2d( $d_s, d_s$ , kernel_size=3, stride=2, padding=1)
LayerNorm
ConvTranspose2d( $d_s, d_s$ , kernel_size=3, stride=2, output_padding=1)
ReLU+LayerNorm
Conv2d( $d_s, d_s$ , kernel_size=3, padding=1)
ReLU+LayerNorm
Flatten and Linear( $d_s, d_t$ )

## B Training

We train Theia on 8 NVIDIA H100 GPUs. The main bottlenecks in training are the data transfer speed between devices and the GPU memory bandwidth to load large spatial feature tensors, for example, of size  $1280 \times 16 \times 16$  for ViT-H and  $256 \times 64 \times 64$  for SAM. We pre-compute the features from all teacher models instead of doing inference on the fly. This approach requires extra storage space to save all the features extracted from the VFMs, but significantly saves on training time and avoids loading models with high GPU memory usage during training, such as Depth-Anything or SAM (a batch size of 16 cannot fit into 80GB of GPU memory). All training configurations are listed in Table 5.

**Teacher VFM Features.** We use the output representations at the last layer of ViT [5], CLIP [2], and DINOv2 [7]. For SAM [10], we use its encoder output. For Depth-Anything [1], since it is initialized from DINOv2, we use the latent representation before the final convolution layer. When decoding SAM and Depth-Anything results from Theia-predicted representations, we send the predicted representations through the remaining layers of original models and obtain the output.

Table 5: Theia Training Configuration

Hyperparameters	
# GPUs	8
Batch size	16 / GPU (128 effective)
Learning rate (LR)	5e-4
LR Schedule	Constant
Weight decay	0.01
Optimizer	AdamW
Betas	[0.9, 0.999]
Epochs	50
Warm-up epochs	5
Warm-up LR schedule	Linear (1e-2*LR)
Gradient clipping	None
Image augmentation	None
Total GPU hours	152

Table 6: More ablation studies on model design. All experiments are based on Theia-T. Scores are the average performance from the MuJoCo subset of CortexBench.

(a) Feature Translator Architecture

CNNs	Linear
<b>78.9</b> $\pm$ 0.8	41.7 $\pm$ 1.6

(b) Training from Scratch vs Pre-trained Backbone

Training from Scratch	Pre-trained Backbone
78.9 $\pm$ 0.8	<b>80.8</b> $\pm$ 1.5

## C Additional Ablation Studies

We conduct two additional ablation studies to verify design choices in the Theia model. The first is a comparison between the current CNN-based feature translator and a linear feature translator. In Table 6a, we find that using a Linear feature translator leads to a significant performance drop. The second ablation studies whether Theia should be trained from scratch or be initialized using the pre-trained DeiT [39] weights. In Table 6b, we find that using pre-trained weights improves the downstream performance. This could be interpreted as the positive effect of incorporating knowledge from an additional useful model into the distillation process. We would expect to see similar performance improvements as more informative models are included during training.

## D Full Experimental Settings

Table 7: Comparison of model architectures, training datasets, total number of images, objectives, and training duration (epochs or GPU hours) across the models used in this paper. We use the numbers reported in their original papers and - stands for we could not find such information.

Model	Architecture	Dataset(s)	Total # Images	Objective	Training Duration
Theia	ViT	ImageNet-1k [23]	1.2M	Distillation	50 epochs / 152 GPU hours on H100s
RADIO / E-RADIO [50]	ViT/Self-designed	DataComp-1B [70]	1.4B	Distillation	-
VC-1 [20]	ViT/MAE [14]	ImageNet-1k [23]+V [31, 51, 53, 52]+N	5.6M	MAE [14]	182 epochs / over 10,000 GPU hours
MVP [19]	ViT/MAE [14]	ImageNet-1k [23]+Video [53, 52, 51]	1.9M	MAE [14]	1600 epochs
R3M [17]	ResNet [4]	Ego4D [31]	-	Time Contrastive [33]+Vision-Language Alignment	1.5M steps
VIP [18]	ResNet [4]	Ego4D [31]	4.3M	VIP [18]	-
DINOv2 [7]	ViT	LVD-142M [7]	142M	Self-distillation	22,016 GPU hours for DINOv2-g
CLIP [2] (Vision Encoder)	ViT	400M	400M	image-text contrastive learning [2]	73,728 GPU hours for CLIP ViT-L/14 on V100s
ViT [5]	ViT	ImageNet-21k [23] / JFT-300M	14M / 300M	Classification	90 epochs / 7 epochs
DeiT [39]	ViT	ImageNet-1k	1.2M	Classification+Distillation	300 epochs / 288 GPU hours on V100s

### D.1 Baseline Models

Theia and baseline models are trained on different sizes of datasets using different objectives. We organize these details in Table 7 to provide a comprehensive comparison between them.

## D.2 CortexBench

For all of our CortexBench experiments, we use the original setup [20], except for a few modifications to produce more reliable results. The modifications include:

- We increase the number of evaluation roll-outs from 10 (original) to 25 (ours) in DMC tasks. The mean scores reported are from a total of 75 runs (25 per seed x 3).
- We remove the noise added to the policy network output in the CortexBench code base. The noise causes minor performance degradation (about 1.0 on overall mean score for MuJoCo tasks) compared to the version without noise.
- We modify the policy networks to take spatial feature inputs for MuJoCo and Trifinger tasks (details follow and are presented in Table 8).

Note that prior models including R3M, VIP, MVP, and VC-1 are all re-run using the same settings in MuJoCo tasks for the purposes of making a fair comparison when evaluating against Theia.

**Policy Networks.** For MuJoCo and Trifinger tasks we utilize a three-layer MLP for vector-based representations, including ResNet models and Transformer models that use the [CLS] token. For models that generate spatial feature maps, such as Transformers using spatial tokens, we introduce a three-layer CNN before the MLP. For Habitat tasks, we exclusively benchmark models that produce spatial feature maps and adopt the same policy network as used by Majumdar et al. [20]. Details can be found in Table 8.

Table 8: Policy Networks for MuJoCo Tasks

Spatial Representation dimension $d \times H \times W$
Conv2d( $d$ , 256, kernel_size=4, stride=2, padding=1)
ReLU
Conv2d(256, 256, kernel_size=3, stride=2)
ReLU
Conv2d(256, 256, kernel_size=3, stride=1)
Linear(256, 256)
Linear(256, 256)
Linear(256, action dimension)
Vector Representation dimension $d$
Linear( $d$ , 256)
Linear(256, 256)
Linear(256, action dimension)

## D.3 Real World Robot Learning

### D.3.1 WidowX Arm Experiments

**WidowX Arm Setup.** The robot used for these experiments is a 6-degree-of-freedom (DOF) WidowX 250s arm. The data collection and evaluation framework is based on [56].

We train a behavior-cloning policy for each of the four evaluated setups and for each evaluated baseline (see Table 2 and Figure 3); the training hyperparameters are shown in Table 9. The policy’s observations are RGB images and robot joint states. Images are encoded by a pre-trained visual encoder and a randomly initialized, unfrozen feature neck (“compression layer” [68]). We use the same feature neck as we did for the previously discussed MuJoCo tasks in Section 4.2. The encoded vector is concatenated with the robot’s joint states, which is fed into a 3-layer MLP with a hidden dimension of size 256. The policy outputs end-effector commands, consisting of the end-effector’s delta positions (Cartesian coordinates), delta rotations (Euler angles), and the gripper’s opening/closing command; such commands are tracked by the robot at a frequency of 5 Hz. In addition to the hyperparameters listed in Table 9, we vary the policy action prediction

Table 9: WidowX Policy Training Configuration

Hyperparameters	
Batch size	16
Learning rate	1e-4
Weight decay	0.01
Optimizer	AdamW
Betas	[0.9, 0.999]
Epochs	400
Loss	SmoothL1

Table 10: Spot Policy Training Configuration

Hyperparameters	
Batch size	32
Learning rate	3e-4
Weight decay	cosine
Optimizer	Adam
Betas	[0.9, 0.999]
Training Iterations	3000
Loss	MSE
Policy Horizon	4

horizon depending of the difficulty of the task, i.e., at each step the policy predicts the next 10, 10 and 5 actions for Door Opening, Toy-Microwave Cooking, and Pick-and-Place, respectively.

In the following, we give more details about the WidowX tasks showcased in our work.

**Door Opening.** In this task the robot has to open a fridge door in a toy-kitchen setup; we identify two stages to evaluate the task’s success: *Open* and *Fully Open* (see Figure 3). We place the robot in front of the fridge and collect 63 demonstrations to train the behavioral cloning policy. We vary the height (z-axis) of the robot base between 40-46cm, and the position (x-axis, parallel to the toy kitchen) of the robot base. At inference time, we vary the height of the robot base among {40, 42, 43, 44, 46} cm, and select between 5 randomly-picked positions along the x-axis (for all policies). For each robot position, we evaluate the policy twice, for a total of 50 runs.

**Pick-and-Place.** In this task, the robot has to pick up a pink cup from a toy-sink and drop it into a drying rack located on the left of the sink. We collected 48 demonstrations to train the policy. During evaluation, we vary the cup’s starting position amongst a total of 10 positions, of which 8 positions are equally distributed about the perimeter, and 2 are in the center of the sink. We also roughly vary the direction of the cup handle towards the left or the right. In total, we evaluate this task for 20 runs. There are two key stages for which we measure the success rate: picking up the cup and successfully releasing it into the drying rack.

**Toy-Microwave Cooking.** In this task, the robot has to pick up an object from within the pot on the stovetop, putting the object into a toy-microwave, and closing the microwave. In each test, we initialize the environment with the microwave door open. In this task, we collected 100 demonstrations across 10 different toy-food objects (10 demonstrations per object) with randomized object positions. During evaluation, we test 40 runs on 10 seen objects (4 runs per object), and 10 runs on 5 unseen objects (2 each), for a total of 50 runs. Furthermore, we vary the position of the pot that holds the object. The longer horizon and the variety of objects in this task make it particularly challenging, so using frozen visual encoders was not effective (0% success rate). However, with fine-tuning, the policies performed reasonably well.

### 592 D.3.2 Spot Experiments

593 **Spot Setup.** For the Spot experiments, we train a diffusion policy [69] conditioned on the encoded  
 594 image. The diffusion policy outputs the desired absolute positions and rotations of the end-effector  
 595 and the gripper state. Hyperparameters for the policies are shown in Table 10.

596 **Drawer Opening.** In this task, the robot has to open the top drawer of a cabinet. The policy  
 597 receives color images from a single forward facing camera mounted on the body of the Spot. A  
 598 fiducial marker is used to enable the robot to walk to a random position and orientation for each  
 599 trial. The starting locations vary by  $\pm 5$  cm in x and y and  $\pm 0.2$  radians in orientation. 50 successful  
 600 demonstrations were collected using a scripted policy and we evaluate each policy for 20 trials. A  
 601 trial is considered successful if the drawer is opened at least 10 cm. After each trial, a scripted policy  
 602 uses the fiducial marker to reset the environment and the robot moves to a new random location.

## 603 E More Visualizations of Translating to Teacher Model Features

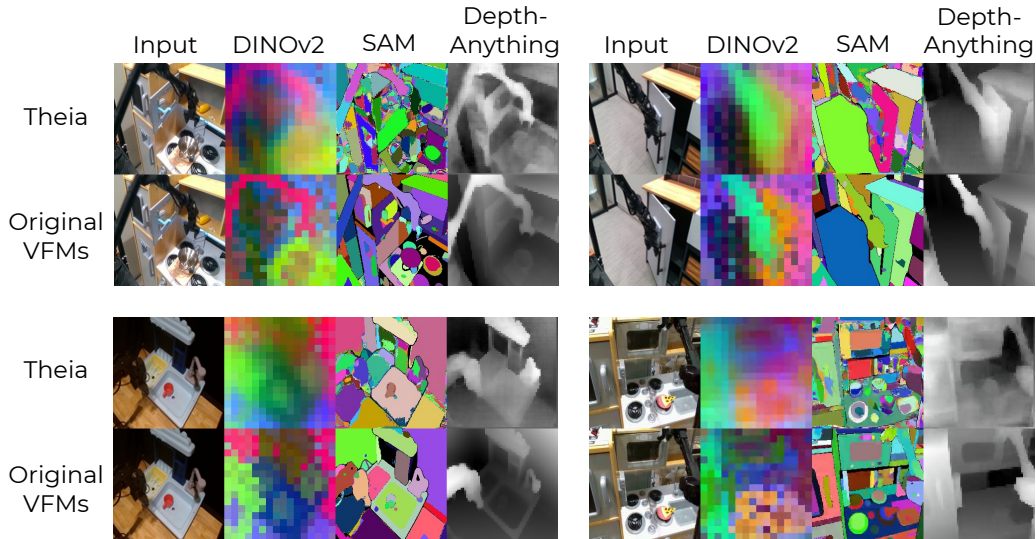


Figure 8: More examples of decoding Theia-representation to VFM outputs using feature translator and original VFM decoders. We select robot images from our experiment recordings. Theia and VFMs are not trained on these images.

604 We attach examples of decoding Theia-representations of 4 frames from a robot video into VFM  
 605 outputs in Figure 8. Note that Theia and VFMs are not trained on the robot images on which we run  
 606 this evaluation.

## 607 F Per-Task CortexBench Results

608 In Table 11 we report per-task scores of the models evaluated in Figure 4, over the MuJoCo subset  
 609 of tasks. In Table 12, we report per-task scores of all 14 tasks we evaluated in Cortexbench, corre-  
 610 sponding to Table 1. Note that we perform the evaluation following the original Cortexbench [20]  
 611 protocol, where there are a total of 75 runs per MuJoCo task (we increased it from 30 to 75 for DMC  
 612 tasks), 100 runs in Reach Cube, and 4200 runs in ImageNav.

## 613 G Analysis of Visual Representations

614 **Entropy of the Representation Norm Distribution.** Given  $N$  representations produced by en-  
 615 coding  $N$  images per model, where each representation contains  $P$  spatial tokens, we discretize the  
 616 distribution of token norms over all  $N \times P$  tokens by using a histogram. We normalize the count

Table 11: Per-Task Results on the MuJoCo Subset.

Model	Assembly	Bin-picking	Button-press	Cheetah-run	Finger-spin	Reacher-easy	Walker-stand	Walker-walk	Drawer-open	Hammer	Pen	Relocate
Theia-T	90.00±8.49	74.00±8.49	80.00±0.00	63.78±1.65	69.76±0.97	79.18±3.87	90.59±1.60	81.06±1.92	100.00±0.00	98.00±2.83	74.00±2.83	46.00±2.83
Theia-S	94.67±9.24	70.67±11.55	72.00±14.42	67.37±4.25	70.46±0.80	83.25±5.19	92.24±0.75	82.62±1.91	100.00±0.00	97.33±4.62	81.33±2.31	50.67±6.11
Theia-B	93.33±8.33	76.00±4.00	82.67±6.11	67.67±1.92	70.84±1.37	83.23±5.05	92.55±3.67	81.33±3.11	100.00±0.00	98.67±2.31	78.67±2.31	46.67±2.31
DINOv2-L	93.33±8.33	80.00±8.00	61.33±2.31	45.66±5.69	70.95±0.25	74.24±16.02	92.84±4.55	83.70±1.34	100.00±0.00	100.00±0.00	77.33±4.62	36.00±0.00
DINOv2-B	92.00±8.00	76.00±14.42	72.00±4.00	48.02±3.71	70.77±0.59	75.84±2.63	92.64±1.81	83.82±2.26	100.00±0.00	98.67±2.31	68.00±4.00	33.33±2.31
DINOv2-S	93.33±8.33	68.00±8.00	81.33±12.22	45.43±5.33	70.70±0.81	59.86±7.73	88.21±1.90	77.62±6.32	100.00±0.00	98.67±2.31	77.33±4.62	36.00±4.00
CLIP-L	69.33±4.62	76.00±4.00	64.00±8.00	33.63±1.21	69.97±2.02	89.42±3.92	95.12±0.89	75.87±5.33	100.00±0.00	96.00±4.00	73.33±8.33	37.33±6.11
ViT-H	94.67±9.24	76.00±12.00	77.33±9.24	47.89±10.10	69.84±1.16	84.33±4.46	90.97±5.36	79.99±6.24	100.00±0.00	94.67±2.31	56.00±0.00	41.33±4.62
ViT-L	96.00±5.66	54.00±25.46	81.60±3.58	50.32±5.24	69.54±0.80	84.49±3.26	89.43±2.30	77.43±1.80	100.00±0.00	96.00±3.27	68.00±3.27	41.60±8.29
ViT-B	96.00±6.93	74.67±10.07	76.00±10.58	46.28±5.32	72.05±1.23	73.71±0.80	77.13±6.50	69.75±3.36	100.00±0.00	96.00±4.00	68.00±0.00	32.00±4.00
ViT-S	93.33±8.33	77.33±2.31	61.33±6.11	42.99±1.13	70.71±0.68	70.46±2.59	88.34±2.63	67.83±4.00	100.00±0.00	90.67±9.24	68.00±4.00	37.33±4.62
ViT-T	93.33±8.33	82.67±4.62	65.33±10.07	39.02±1.47	71.45±1.07	71.13±3.24	84.05±0.84	70.74±3.03	100.00±0.00	84.00±6.93	60.00±10.58	25.33±4.62
VC-1-L-sp	85.33±8.33	66.67±12.22	56.00±8.00	66.88±6.66	71.19±0.67	70.67±8.36	93.43±6.08	83.28±2.40	100.00±0.00	93.33±2.31	68.00±0.00	24.00±8.00
CDV	73.33±24.44	74.40±11.87	30.67±39.26	50.53±12.08	71.94±1.07	71.60±17.03	94.85±1.27	78.36±5.72	100.00±0.00	93.60±6.07	75.33±5.89	39.33±10.25
RADIO	96.00±6.93	84.00±8.00	82.67±12.86	35.92±1.59	71.98±1.41	78.46±6.50	89.06±2.42	81.33±4.91	100.00±0.00	98.67±2.31	68.00±0.00	40.00±6.93
E-RADIO	94.67±9.24	82.67±2.31	80.00±4.00	57.37±2.27	69.68±1.05	75.59±1.29	91.73±2.03	80.32±2.06	100.00±0.00	100.00±0.00	66.67±12.86	45.33±2.31
MVP-L-sp	93.33±8.33	73.33±4.62	82.67±10.07	68.07±1.71	71.03±2.10	69.87±6.75	88.44±4.21	80.14±1.50	100.00±0.00	97.33±2.31	77.33±12.22	26.67±11.55
MVP-L	94.67±9.24	82.67±9.24	89.33±8.33	34.62±5.83	68.63±2.02	67.95±3.18	74.50±1.65	48.04±1.37	100.00±0.00	88.00±6.93	62.67±6.11	20.00±4.00
R3M	96.00±6.93	92.00±4.00	68.00±4.00	55.88±1.12	70.65±0.34	82.37±3.70	88.88±2.70	69.52±4.94	100.00±0.00	98.67±2.31	73.33±2.31	58.67±4.62
VIP	93.33±8.33	70.67±8.33	76.00±4.00	45.10±4.02	69.02±0.67	68.08±3.45	78.50±2.49	63.52±1.40	98.67±2.31	96.00±4.00	73.33±6.11	29.33±10.07

Table 12: Per-task results on CortexBench (excluding Move Cube, ObjectNav, and MobilePick due to reproducibility issues).

Model	Assembly	Bin-picking	Button-press	Cheetah-run	Finger-spin	Reacher-easy	Walker-stand	Walker-walk	Drawer-open	Hammer	Pen	Relocate	Reach-cube	ImageNav
Theia-B	93.33±8.33	76.00±4.00	82.67±6.11	67.67±1.92	70.84±1.37	83.23±5.05	92.55±3.67	81.33±3.11	100.00±0.00	98.67±2.31	78.67±2.31	46.67±2.31	86.19±0.11	59.3±0.7
VC-1-L-sp	85.33±8.33	66.67±12.22	56.00±8.00	66.88±6.66	71.19±0.67	70.67±8.36	93.43±6.08	83.28±2.40	100.00±0.00	93.33±2.31	68.00±0.00	24.00±8.00	84.79±0.63	70.3±0.7
E-RADIO	94.67±9.24	82.67±2.31	80.00±4.00	57.37±2.27	69.68±1.05	75.59±1.29	91.73±2.03	80.32±2.06	100.00±0.00	100.00±0.00	66.67±12.86	45.33±2.31	87.81±0.12	53.0±0.7
MVP-L-sp	93.33±8.33	73.33±4.62	82.67±10.07	68.07±1.71	71.03±2.10	69.87±6.75	88.44±4.21	80.14±1.50	100.00±0.00	97.33±2.31	77.33±12.22	26.67±11.55	87.54±0.2	68.1±0.7
R3M	96.00±6.93	92.00±4.00	68.00±4.00	55.88±1.12	70.65±0.34	82.37±3.70	88.88±2.70	69.52±4.94	100.00±0.00	98.67±2.31	73.33±2.31	58.67±4.62	86.5	30.6±0.7
VIP	93.33±8.33	70.67±8.33	76.00±4.00	45.10±4.02	69.02±0.67	68.08±3.45	78.50±2.49	63.52±1.40	98.67±2.31	96.00±4.00	73.33±6.11	29.33±10.07	86.2	48.8±0.8

of each bin in histogram by the total number of tokens to obtain the probabilities of each bin. We then calculate the Shannon entropy, given by  $-\sum_i p_i \log(p_i)$ . We find that the distilled models have higher entropy than the regular models, so we divide them into two distinct groups. Results are plotted as model performance vs entropy on the MuJoCo tasks. We attach the full version of plot presented on the left of Figure 7 here in Figure 9, including two plots corresponding to each category of models and one plot for all models.

At the top of Figure 10, we find that both CLIP and VC-1 have high-norm outlier tokens. To better visualize the values of normal tokens, we use the median of norm values to clip the values. Specifically, we clip the norm values to range  $[0, 2 * \text{median}]$  and visualize the clipped norm values on the bottom of Figure 10. We find that the high-norm tokens are still not task-relevant.

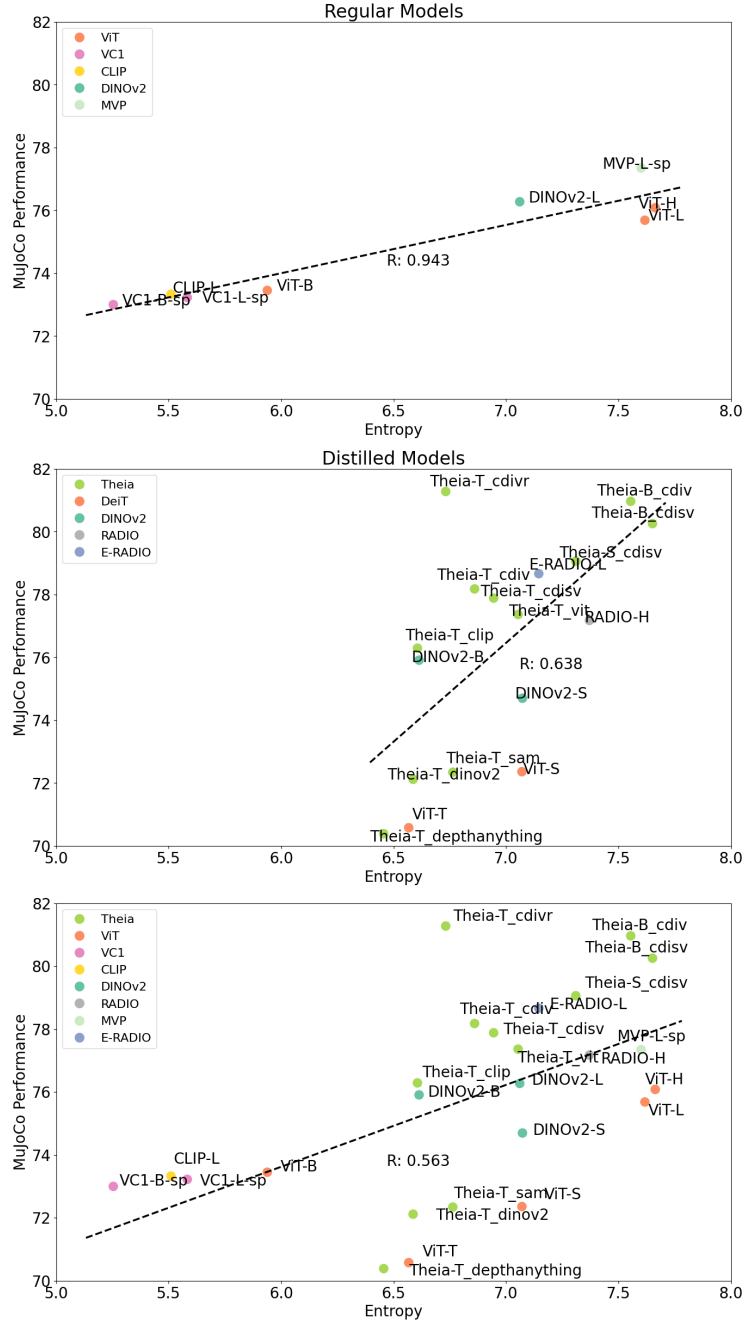
In Figure 11, we attach the feature norm map of all other models. Among those, we find that MVP [19], which performs well on CortexBench, also produces features without outlier tokens. Feature norms from Depth-Anything [1] and SAM [10], in contrast, have low diversity.

**PCA Explained-Variance Ratio of Representations.** Similar to the entropy analysis, given  $N \times P$  spatial token representations, we apply PCA to them and extract the explained-variance ratio (EVR) of each latent dimension. We calculate and plot the cumulative sum of EVRs, as well as calculate the Area Under the Curve (AUC) of cumulative sum of the EVR. When comparing Theia-B with ViT-B, DINOv2-B and VC-1-B (Figure 12), we find that Theia-B has the lowest AUC and the best MuJoCo performance, while VC-1 has the highest AUC and the worse MuJoCo performance amongst these 4 models. The higher AUC is caused by one or few principle components that have very high EVRs, indicating that these components are capturing the majority of the variance of the feature representations. This means that less information is encoded within such representations. In contrast, the Theia-representation has a low AUC which we believe is due to the rich information that has been encoded within the latent space.

However, when extending the scope to encompass all the models we evaluated (Figure 13 left), we find that the AUC of the EVR does not have a strong correlation with robot learning performance.

**Cosine Similarity of Representations.** We also use cosine similarity to analyze the representations from different models by first calculating the mean of all representations and then computing cosine similarity between each representation and this mean representation. Results are shown on the right of Figure 13, which shows very weak correlation between cosine similarity and performance on CortexBench.





## H Linear Probing on ImageNet

In addition to robot learning, we evaluate the Theia-representation on vision tasks to show how well such abilities are maintained after the distillation process. For example, to evaluate image classification performance we apply linear probing on the Theia-representation to classify images from ImageNet-1k [23]. We use mean pooling of the Theia-representation (i.e. spatial tokens) and the same training schedule as MAE [14]. Results are shown in Table 13, where we find that Theia outperforms MAE [14] at the same model size, but is not comparable to SOTA results from models like DINOv2 [7].

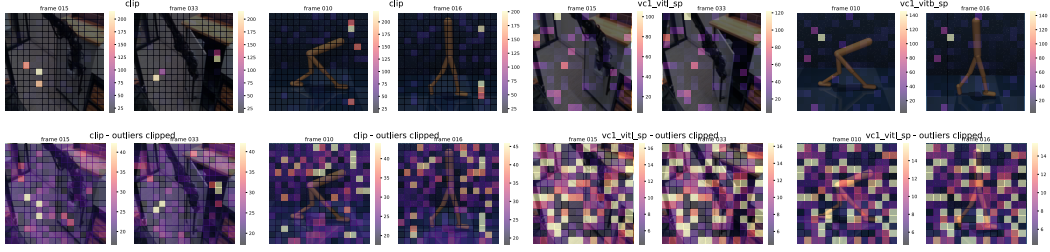


Figure 10: Feature norm map visualizations of CLIP and VC1, original (top) and with high-norm outlier values clipped (bottom)



Figure 11: Feature norm map visualizations of ViT [5], DINOv2 [7], MVP [19], Depth-Anything [1], SAM [10], RADIO [50], and E-RADIO [50].

Table 13: ImageNet-1k [23] evaluation accuracy using linear probing.

Model	Accuracy
Theia-B	72.1%
Theia-B (initialized from DeiT-B [39] weights)	75.2%
MAE (ViT-B) [14]	67.5%
DINOv2 [7] (ViT-L)	84.5%



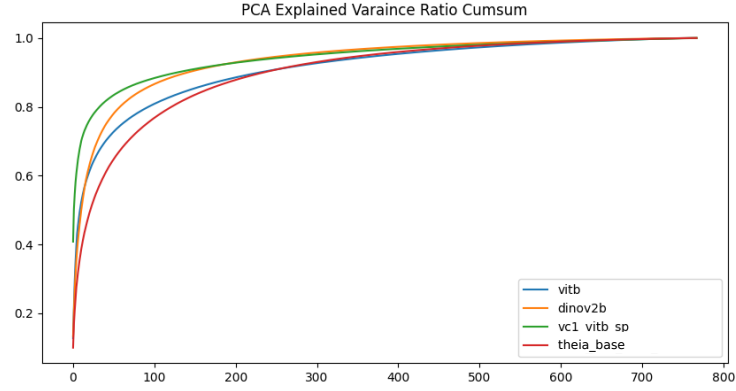


Figure 12: Cumulative sum of PCA Explained Variance Ratio of features from ViT-B, DINOv2-B, VC-1-B, and Theia-B.

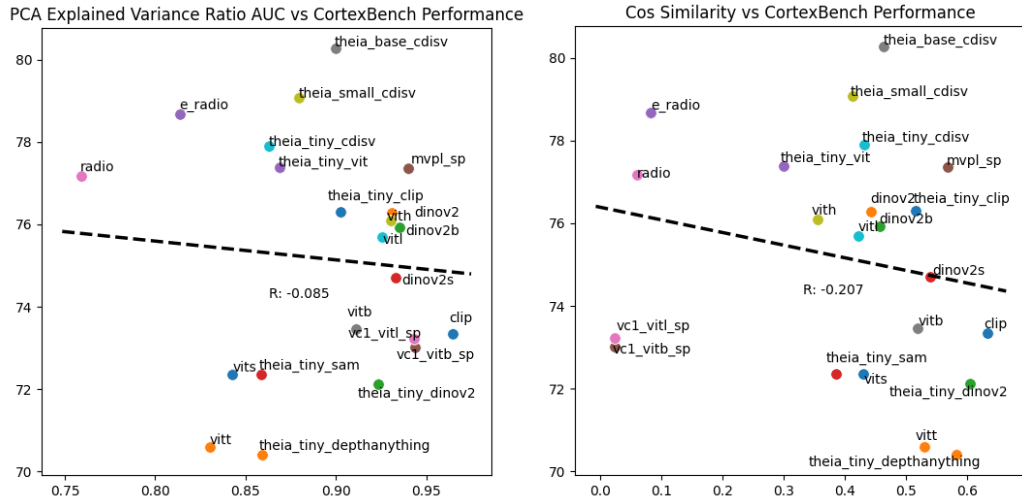


Figure 13: PCA explained variance ratio-AUC (left) and cosine similarity (right) vs MuJoCo performance of many models evaluated.