

Appendix

A1 Whisk Dataset Generation

Our whisking dataset uses the same 9981 ShapeNet objects and 117 category labels as in [Zhuang et al. \[2017\]](#), but using an improved whisker model and various sweep augmentations, which are listed in Table [1](#).

	Sim. Freq.	Speed	Height	Rotation	Distance	Size	Total Sweeps
(1)	1000 Hz	30 mm/s	-5, 0 mm	0, 30, 90, 120°	5, 8 mm	40 mm	316,192
(2)	110 Hz	[30~60]	-3, 0, 3	[0~359]	5	[20~60]	2,076,048

Table 1: **Sweep Augmentations** used for the two whisking datasets, which we refer to as (1) Low-Variation High-Fidelity and (2) High-Variation Low-Fidelity. Simulation Frequency refers to the frequency corresponding to the physics timestep used in Bullet physics engine [\[Coumans and Bai, 2016–2021\]](#), the backend for WHISKiT simulator [\[Zweifel et al., 2021\]](#). For dataset (2), the speed, rotation, and size is each sampled from the range 26 times. The total number of sweeps equals the number of sweep augmentation combinations \times 9981 objects.

WHISKiT Simulator Modifications. We make a few enhancements to the original WHISKiT simulator [\[Zweifel et al., 2021\]](#), available in our code:

- Number of whisker links (where whiskers are modeled as a chain of springs [\[Zhuang et al., 2017\]](#)) are dynamically adjusted by the length of the whisker instead of a fixed number.
- Allow for loading objects with or without convex hull. In the whisk datasets, objects are loaded with convex hull (using V-HACD [\[Mamou and Ghorbel, 2009\]](#)) to keep collisions more stable. In generating the 6 simulated stimuli for model input neural evaluation, convex hull was not used in order to preserve the concavity of the “concave” object. The object was geometrically simple enough to not affect collision stability.
- Add camera settings for viewing in orthographic or perspective projection.
- Load multiple mice/rats at a time.

A2 Model Training

All of our experiments are conducted on Nvidia A6000 GPUs. For supervised learning, we use a batch size of 256 for all the models and train for 100 epochs. For SSL, during the pre-training stage, we use a batch size of 256 for SimCLR and autoencoding (AE), and a batch size of 1024 for SimSiam, following [\[Nayebi* et al., 2023\]](#), and train for 100 epochs. During the linear probing stage, we freeze the checkpoint saved with the lowest validation loss and add a trainable linear layer. We further train such a model with labels for 100 epochs, with a batch size of 256, an initial learning rate of 0.1, with the StepLR scheduler, and the SGD optimizer with momentum [\[Bottou, 2010\]](#).

We detail the optimizers [\[Bottou, 2010\]](#), [\[Kingma and Ba, 2015\]](#), [\[Loshchilov and Hutter, 2019\]](#), [\[You et al.\]](#) and schedulers, and their configurations we used in supervised learning and SSL in Table [2](#).

For supervised learning, we present the model training configurations in Table [3](#), where we detail the choices of optimizer, scheduler, learning rate, the encoder and attender of our EAD architecture, and we omit the decoder due to space limits, as it is always a linear layer/MLP. As Zhuang+GPT is the best performing supervised model in terms of classification accuracy, we explore different variants of Zhuang’s encoder with attender being GPT.

For SSL, we follow [\[Nayebi* et al., 2023\]](#) and use specific configurations of optimizer and scheduler for different losses, which are shown in Table [4](#). For SimSiam, we try the CosineAnnealing scheduler both with and without 10 epochs of warmup to search for better model performance. For AE, we use a 3-layer deconvolution network to decode the sparse latent representation from our EAD framework into the original tactile input, regardless of the choice of model architectures.

We present the model architectures explored for SSL in Table [5](#), where we present the encoder and attender of our EAD architecture, as during the pre-training stage, only the encoder and attender are used, and during the linear probing stage, the decoder is always a one-layer linear classification head.

Optimizer	Configuration
SGD	momentum = 0.9, weight-decay = 10^{-4}
Adam	weight-decay = 10^{-4}
AdamW	weight-decay = 10^{-4}
LARS	momentum = 0.9, weight-decay = 10^{-4}
Scheduler	Configuration
StepLR	step_size = 30
ConstantLR	fixed learning rate
CosineLR	warmup_epoch = 10
CosineAnnealing	min_lr = 0.0, warmup_epoch = 10 warmup_ratio = 10^{-4}

Table 2: **Optimizers and Schedulers** with default configurations of supervised learning and SSL when training different model architectures.

Encoder	Attender	Optimizer	Scheduler	Learning Rate
ResNet	None	SGD	StepLR	$10^{\{-1,-2,-3\}}$
Zhuang	None	SGD SGD	StepLR ConstantLR	$10^{\{-1,-2,-3,-4\}}$ $10^{-2}, 5 \times 10^{-3}$
Zhuang-{UGRNN, IntersectionRNN, GRU, LSTM}	None	{SGD, Adam, AdamW} (LayerNorm) - AdamW	StepLR StepLR	$10^{\{-1,-2,-3,-4\}}$ $10^{\{-1,-2,-3,-4\}}$
{ResNet, Zhuang, Zhuang-{UGRNN IntersectionRNN-LN, GRU, LSTM}, S4}	GPT	AdamW {SGD, AdamW}	CosineLR StepLR	10^{-4} $10^{\{-1,-2,-3\}}$
{ResNet, Zhuang, S4}	Mamba	AdamW {SGD, AdamW}	CosineLR StepLR	10^{-4} $10^{\{-1,-2,-3\}}$

Table 3: **Model Training Configurations for Supervised Learning.** We use { } to indicate different choices of a specific component (i.e., encoder, optimizer, learning rate) in the search space. We consider adding layer norm as a variant when searching the best configuration for Zhuang’s variants (i.e., the second row), which is denoted as “-LN”.

A3 Neural Evaluation

We use the NeuroAI Turing Test [Feather* et al., 2025] to evaluate the neural similarity of mice whisking to models performing tactile categorization.

RSA Correlation. Due to the low number of stimuli, we use RSA [Kriegeskorte et al., 2008] as our correlation metric. RSA is computed over stimuli and neurons, where the average is over source animals/subsampled source neurons, bootstrapped trials, and train/test splits. This yields a vector of these average values, which we can take median and s.e.m. over, across animals.

For the neurons of animal A to animal B in the set of animals \mathcal{A} we estimate the RSA correlation:

$$\langle \text{RSA}(t^A, t^B) \rangle_{A \in \mathcal{A}: (A, B) \in \mathcal{A} \times \mathcal{A}} \sim \left\langle \frac{\text{RSA}(s_1^A, s_2^B)}{\sqrt{\text{RSA}(s_1^A, s_2^A) \times \text{RSA}(s_1^B, s_2^B)}} \right\rangle_{A \in \mathcal{A}: (A, B) \in \mathcal{A} \times \mathcal{A}}. \quad (1)$$

We additionally bootstrap across trials, and have all 50%/50% train/test splits ($\binom{6}{3} = 20$ splits). The average on the right hand side of the equation includes averages across these as well. Each neuron in our analysis is associated with this average value when it was a target animal (B), averaged over source animals or subsampled source neurons, bootstrapped trials, and train/test splits. This yields a vector of these average values, which we can take median and standard error of the mean (s.e.m.) over, as we do with standard explained variance metrics.

Spearman-Brown Correction. The Spearman-Brown correction can be applied to each of the terms in the denominator individually, as they are each correlations of observations from half the trials of

Loss	Optimizer	Scheduler	Learning Rate
SimCLR	LARS	CosineAnnealing	$10^{-1,-2,-3,-4}$
SimSiam	SGD	CosineAnnealing (with & w/o warmup)	$10^{-1,-2,-3,-4}$
AE	SGD	StepLR	$10^{-1,-2,-3,-4}$

Table 4: **Optimizer, Scheduler, and Learning Rate Configurations for SSL** when training different model architectures.

Encoder	Attender
Zhuang	None
Zhuang-{UGRNN, IntersectionRNN-LN, GRU, LSTM}	None
{ResNet, Zhuang-IntersectionRNN-LN, Zhuang, S4}	GPT
{ResNet, Zhuang, S4}	Mamba

Table 5: **Model Architecture Configurations for SSL**. We use {} to indicate different choices of encoders in the search space. We consider adding layer norm (LN) as a variant when searching the best configuration for Zhuang’s variants (i.e., the second row)

586 the *same* underlying process to itself (unlike the numerator).

$$\begin{aligned}\widetilde{\text{RSA}}(X, Y) &:= \widetilde{\text{Corr}}(\text{RDM}(x), \text{RDM}(y)) \\ &= \frac{2 \text{RSA}(X, Y)}{1 + \text{RSA}(X, Y)}.\end{aligned}$$

587 **Inter-Animal Consistency.** To estimate the inter-animal consistency, we evaluate the pooled animal
588 consistency for each animal. One animal is held out at a time, then compared to the pseudo-population
589 aggregated across units from the remaining animals. We found the mean pooled animal score was
590 0.175 with a s.e.m. of 0.161 and maximum score of 1.34.

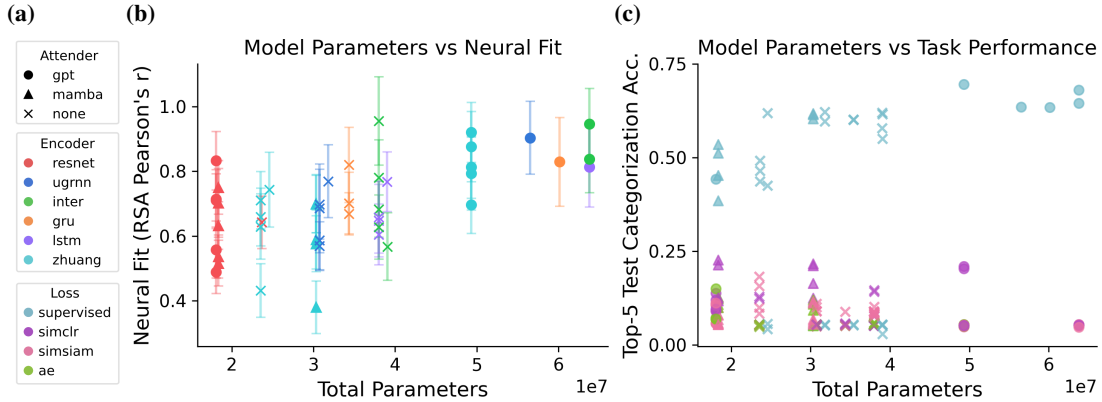


Figure A1: **Model Parameters** compared with categorization performance and neural fit. (a) Legends for (b) (Encoder colors) and (c) (Loss colors). (b) Models with GPT as an attender have a higher correlation score slightly higher than those without, but high neural fit is still achievable without more parameters as demonstrated by the Inter+SimCLR model (high green “x”). (c) Models with GPT as the attender has more parameters and, when trained with supervised learning, tends to have higher top-5 categorization accuracy.

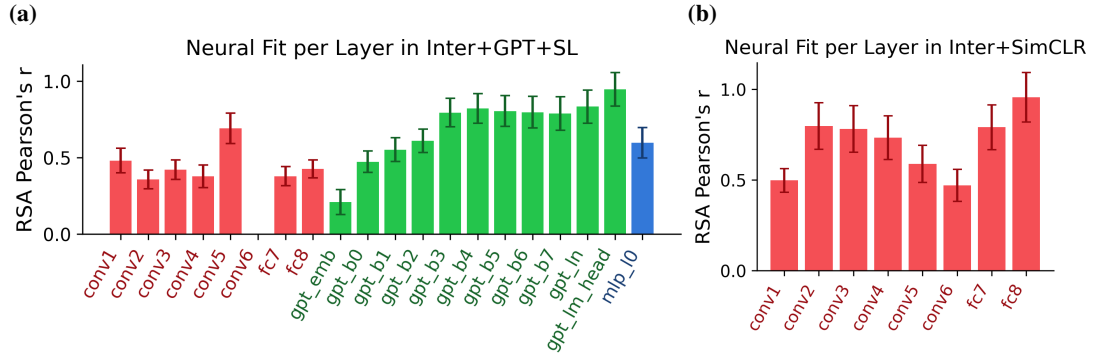


Figure A2: **Neural Fit Score per Model Layers** colored by **encoder**, **attender**, and **decoder**. No bar means the score is NaN for that layer. (a) Inter+GPT+SupervisedLearning is the model that scored the highest neural fit out of the supervised models. We observe that later GPT layers perform increasingly better. (b) The last fully-connected layer of Inter+SimCLR achieved the highest r value.

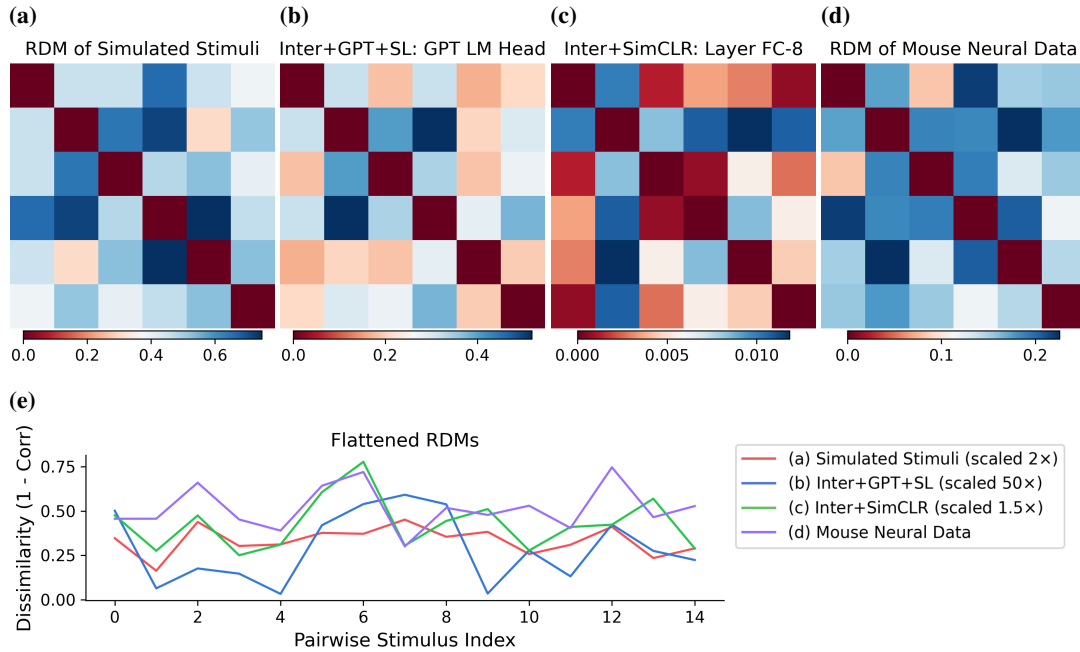


Figure A3: **Representational Dissimilarity Matrices (RDM)** for tactile data and neural evaluation. (a) RDM of the 6 simulated stimuli which is used as the model input in neural evaluation. (b) The GPT LM Head layer of Inter+GPT+SupervisedLearning is the supervised model with the highest neural fit score. (c) The last Fully-Connected layer in Inter+SimCLR achieves the highest neural fit score out of SSL models. (d) RDM performed on the 6 stimuli in the mice neural data. (e) A visualization of the flattened RDMs, scaled approximately to fit (RSA correlation does not take scale into account).