# Methods

The source dataset for this work was a publicly available collection of synaptic traces and measurements mined from peer-reviewed publications and carefully annotated for detailed metadata as previously described (Moradi and Ascoli, 2020). In this study, we first reconstructed these signals into a set of systematic measurements. Next, we simulated the traces with a synapse model to unify the data format. Then, we created a predictive deep learning model of all the data to infer missing values, disambiguate the identity of presynaptic and postsynaptic neuron types, and normalize the data with respect to covariates. Lastly, we statistically analyzed the resultant completed and normalized dataset and corresponding synaptic simulations.

## Synaptic signal reconstruction

To digitize the mined traces, we used Engauge Digitizer, a multiplatform open-source software (digitizer.sourceforge.net). We implemented a custom Python algorithm, Trace Reconstructor, as part of our Synapse Modeling Utility, to extract a consistent set of data points from each synaptic event, including an initiation, a peak, and a decay point (Fig. S2a). Each data point consists of a time and a corresponding amplitude. We found data points either from digitizing traced or through interpolation of reported synaptometric measurements, such as the average amplitude, 10%-90% or 20%-80% rise times, half-height width (50% rise to 50% decay time), and half-decay time (100%-50% decay time). Six additional intermediate data points were interpolated using the Akima interpolator implemented by SciPy47.

For the accurate simulation of ST-P, we ensured all digitized signals had at least 10 successive synaptic events and a recovery event, interpolating them if needed from paired-pulse ratios (PPRs). To infer missing PPRs of facilitating and pseudolinear ST-Ps, we used bicubic interpolation. For depressing signals, we used a custom interpolator that assumed the PPRs exhibit exponential decay to a minimum. For depressing or pseudolinear signals that lacked a recovery event, we assigned a 2 s period for recovering from synaptic depression48-50. Specifically, we assumed this as the time for the recovery to reach 63% of the difference between the amplitude of the first and the last events in a successive series of events. For facilitating synapses, we did not add a recovery event.

Most synaptic signals start with a fast AMPA or a GABAA response which are gradually mixed with slower synaptic responses or non-synaptic membrane fluctuations. To diminish the impact of slower events, we corrected the signals either at the reconstruction stage or during parametric fitting (Fig. S1). When the ISIs of synaptic events were constant, we reconstructed the signal based on the amplitude and the decay time constant of the first synaptic event and the paired-pulse ratios of the successive events. When the ISIs were variable, we used simulated signals to correct the data as described below.

We implemented all the above-mentioned reconstruction algorithms in the Trace Reconstructor tool of the Synapse Modeling Utility.

## Biophysical synaptic model and parametric fitting

To facilitate comparison between current and voltage recordings, we reduced the signals to modality independent synaptic constants utilizing a specific version of the Tsodyks, Pawelzik, and Markram (TPM) model9, 16. The TPM model formulates a relationship between synaptic conductance (g), deactivation time constant (τd), recovery time constant (τr), facilitation time constant (τf), and the utilization ratio (U) of synaptic resources in one set of ordinary differential equations (Suppl. Methods). Calculating synaptic currents (Isyn) with an Ohmic model for ion channels (Fig. 2b) requires the reversal potential (Erev) and the postsynaptic membrane potential (Vm). Erev is experimentally measurable or can be accurately estimated from the ionic composition of bath and pipette solutions, temperature, and permeability of ion channels to different ions8. We assumed kinetically fast synaptic responses to be mediated by calcium-impermeable AMPA or GABAA channels, unless otherwise stated in the original publications. Because Isyn is recorded in voltage-clamp experiments, we calculated Vm by correcting holding potential (Vh) for liquid junction potential (Ej) as previously described (Moradi and Ascoli, 2020).

Using the TPM model, we can analytically simulate the amplitude, kinetics, and ST-P of Isyn (see Suppl. Methods). We numerically derived synaptic potentials (Vsyn) by feeding the simulated Isyn to a resistor-capacitor circuit (RC) model of neuronal membrane, from which we equated Vsyn as the evolution of Vm over time. We used the ODEPACK solver via SciPy for numerical integration. The RC model depends on three experimentally measurable parameters: the membrane time constant (τm), membrane capacitance (Cm), and the initial value of Vm. Since Vsyn is recorded in current-clamp experiments, we corrected resting or steady-state membrane potential for Ej to estimate the initial value of Vm. We used τm and Cm values when reported in the original study; otherwise, we utilized the values reported by Hippocampome.org for a matching postsynaptic neuron type in the closest available temperature, recording method, and solutions7. If parameters of the RC model could not be found in the original paper or Hippocampome.org, the values were optimized during parametric fitting. Only for 23% (603:2621) of the signals at least one of the τm and Cm values was found through optimization.

We found the optimal g, τd, τr, τf, and U values for each experimentally recorded synaptic signal by fitting TPM model simulations to the reconstructed data points. We created a high-performance and user-friendly Python simulator, the Synapse Modeling Utility, to aid in parametric fitting. Optimization was performed by an implementation of the SciPy toolbox genetic algorithm, *differential\_evolution* function, a bound-constrained global optimizer. As the objective function, we chose the mean soft L1 squared error, i.e., , where n is the number of reconstructed data points. We assigned the fitting error associated with the first synaptic event twice the weight relative to all other events, and the 6 interpolated data points half the weight of the initiation, peak, and decay points. We set the following bound constraints: 50 < τr < 3000 ms, 1 < τf < 300 ms, and 0.001 < U < 1. Optimization was stopped when the difference of fitting error between successive fits yielded a change of less than 0.001.

If more than one stimulation frequency was available for a given experiment, we pooled data prior to optimization to ensure the estimated parameters are more generalizable to different frequencies. We then re-expanded the data after optimization to match each of the original traces.

The Synapse Modeling Utility also implemented a correction for slow processes when the ISIs varied (see also Suppl. Materials Discussion). In the absence of a slower process, the signal should gradually decay to a baseline. Provided slower processes do not drastically affect the first simulated event such that τd estimation is accurate, the recorded and simulated signals should be most similar at the initiation points of the synaptic events. We defined the correction amounts at initiation points of two successive synaptic events as the amplitude differences (Fig. S1b). We then calculated the correction values for data points in between two initiation points by the triangulation method. Signals yielding τd values greater than 700 ms were excluded from subsequent analysis. For signals that only reported amplitude and not kinetics, we set the missing τd values to the median of the unitary GABAergic and glutamatergic responses as appropriate.

Fitting a single synaptic signal using our Synapse Modeling Utility required seconds to minutes, while fitting with a pipeline built using the Neuron Simulation Environment51 required hours. We optimized each trace at least 30 times and averaged the best 15 fits. The relative inter-trial variability was less than 0.001.

## Machine learning design and implementation

We employed machine learning to infer the five parameters of the TPM model (the *targets*) based on a set of *features*, namely the pre- and post-synaptic neuron types and covariates. Specifically, the set of features consisted of 319 one-hot encoded and numerical values (Table 1): 122 features encoded presynaptic neuron types, 122 postsynaptic neuron types, and the remaining 75 features encoded the experimental covariates. For instance, three columns encoded four stimulation methods (evoked, unitary, spontaneous, if all three columns were set to zero the stimulation method was miniature), one column encoded three species (1=rats, -1=mice, and 0=guinea pigs), and one column sex (1=male, -1=female, and 0=both or unknown). When the animal age was not reported, we estimated it based on weight, diet, species, and strain. One feature column encoded whether the target signal represented amplitude or potency, which differ in the averaging method: if failed events are excluded from the signal average, the peak quantifies synaptic potency rather than amplitude. For algorithm training, when the failure rate of the first synaptic event was reported, we added an additional pseudo-signal by converting amplitude to/from potency, resulting in a different g value as a target and a different potency value as a feature. We normalized features with the *MaxAbsScaler* function of Scikit-learn toolbox to preserve the sparsity of the feature matrix. Moreover, we normalized the targets with the *MinMaxScaler* method to allow usage of sigmoid activation functions in the deep learning output layer.

We implemented the machine learning pipeline in Jupyter and Python. We first trained a random forest model using the Scikit-learn package on two Xeon-E5 v3 CPUs to infer missing values of ST-P. Specifically, whenever a signal lacked estimates for τr, τf, and U (typically recordings of single synaptic events), we set those values to zero. However, to allow machine learning to distinguish such ‘not available’ entries from real zero values, we also set the ISI, a feature, to zero in those cases. The random forest model, in contrast to deep learning, could learn to predict missing values for τr, τf, and U, when ISI was zero. Similarly, the random forest model was able to infer the missing values for these parameters when ISI was non-zero. Specifically, we set missing ISI values to 50 ms, the mode of all ISIs. We then employed the inferred values together with the original values to train a deep learning model utilizing the Keras library with a TensorFlow 2.3 backend on seven NVIDIA Titan X GPUs (Fig. S3). We also fed back the originally missing values of τd, τr, τf, and U estimated by deep learning iteratively until we observed no further improvement in model performance (30 times).

We meticulously hand-tuned the hyperparameters of the deep learning model. Checking different deep learning topologies, we settled on a five-layer autoencoder perceptron, regularized with the latest available techniques to achieve state-of-the-art accuracy and generalization power. Specifically, we used the self-regularized mish activation function52, dropout layers53 combined with max-norm constraint54, batch normalization layers55, weight decay regularization56, noise regularization57, and early stopping technique58. As the objective function of the deep learning model, we employed symmetric mean absolute percentage error (), where n is the number of data points59, 60. It is advantageous to use SMAPE over the competing methods because it is scale-independent and unbiased. We trained the models with the lookahead optimization algorithm61 guided by the AdamW optimizer56 with weight decay = 0.001, learning rate = 0.015, and batch size = 2621. We implemented learning rate reduction on plateau to achieve the best fit (patience = 100 epochs, factor = 0.9). We used the “early stopping” technique for restoring the best weights at the end of training to avoid overfitting.

The exact predictions of the model depended on the (randomized) sorting of the training dataset. Thus, we trained 100 models and statistically analyzed the results for each potential connection (Fig. S8). The CV of model predictions was not significantly larger in any region and did not correlate with the number of data points available per potential connection. Among parameter predictions, the CV was largest for τf, and smallest for τd, U, and GABAergic τr. To maximize robustness, we reported the average value of 100 model inferences for each parameter and potential connection.

The sigmoid activation functions in the output layer ensure model inferences stay data-bound; nevertheless, we also made sure all g, τd, τr, τf, and U predictions are unique and biologically plausible, i.e., g > 0 nS, 0 < τd < 70 ms, τr > 50 ms, τf > 1 ms, and 0 < U < 1.

### Machine learning model validation

We computed *training* *accuracy* as the average SMAPE distance of the model output from the training data. In contrast, *prediction accuracy* is the average SMAPE distance of the model output from the unseen data. We monitored the prediction accuracy of the model after each training epoch with k-fold cross-validation62, 63 with k=4. We trained four models on four separate training runs, each using three-quarters of the data, used the remaining one-fourth of data to measure prediction error, and averaged the results over the four runs. We assessed the final model accuracy with the jackknife method (Figs. 3a-b and S4): we trained n (n=2,621) models with n-1 data points and assessed the prediction error of the model for one set aside data point.

Our dataset had more than one data point for most potential connections (Figs. 1, S2e, and S2f). In certain cases, these data points had identical features. *Target variability* is the average distance (in SMAPE) of each target value in a group from the group average. For one set of features, predictive models can only predict one set of targets. Therefore, the variability of targets imposes a limit on the maximum accuracy the model can achieve. Considering the average target values as the best estimates of the true values, we calculate the 95% confidence interval around the mean and defined a model prediction as *reliable* if it fell within the confidence interval. Prediction reliability (PR) is the percentage of model predictions that are within the confidence interval.

## Data normalization and statistical analyses

The training features were highly heterogeneous and typically mapped to multiple presynaptic and/or postsynaptic neuronal types (“fuzzy” or ambiguous mapping). Nevertheless, a trained model can predict targets (synaptic parameters) for an arbitrary set of features. We inferred values for the unambiguous (“proper”) mapping of all 3,120 potential connections in the entorhinal-hippocampal network. In other words, each inference feature was mapped to one presynaptic neuron type and one postsynaptic neuron type. We also set all other features except the presynaptic and postsynaptic neuronal types to identical values. For instance, we selected identical ionic concentrations for physiological solutions across all synapses and calculated Erev accordingly. We set no NMDA or GABAB contamination for the features. Using the trained deep learning models, we inferred unitary synaptic parameters for each potential connection always verifying that the predicted values remained within the upper and lower boundaries of the training set to avoid erroneous extrapolations. We chose unitary postsynaptic currents recorded from adult male rat slices kept in artificial cerebrospinal fluid at body temperature while using whole-cell patch pipettes devoid of high [Cl]i or [gluconate]i solutions as a standard condition for model inferences. When analyzing covariates, we changed one feature at a time to infer the corresponding synaptic parameter. We also generated the inferences for 32 different permutations of conditions, i.e., rat vs mice, male vs female, P14 (adolescent) vs P56 (adult), room (22 °C) vs body (32 °C) temperatures, and voltage-clamp vs current-clamp recording methods.

To compare synapses, we either analyzed the TPM model parameters directly or simulated each synapse separately and measured different synaptometrics. The paired-pulse ratio is the measure of ST-P, which requires the estimation of amplitude. For the first synaptic event, the baseline crosses the initiation points, but for later events, the overlap of initiation points with the baseline depends on the ISI and τd values. If the amplitude is measured from the baseline, we used the *ABi* term, where *i* is the event number. Otherwise, if the amplitude is measured from the initiation point, we used the *Ai* term. For example, *A1* is the amplitude of the first synaptic event, and *AB3* is the amplitude from the baseline of the third synaptic event (Figs. 2 and 6). Thus, *ABi:A1* represents the paired-pulse ratio if the ith event from baseline, which assesses the evolution of the synaptic activation (Suppl. Methods).

We compared groups with paired or unpaired Wilcoxon’s test as appropriate. We corrected all p-values for multiple comparisons using False Discovery Rate64 and selected 0.05 as the significance threshold. We corrected coefficients of variation (CVs) for sample sizes and used the bootstrapping method to find the confidence intervals65, 66. We used the Pearson method to compute correlations.

As a measure of central tendency, we defined trimmed-mean as a mean value in which 2.5% of outliers on both extremes are excluded. The interquartile range is the measure of spread. Since the synaptic parameters have different units, we used symmetric percentage distance () as a measure of change between two data points that is dimensionless and unbiased. We simply use the term percentage to refer to SPD in the Results and Discussion of this paper. Specifically, for covariates analysis (Fig. 5b), we computed the trimmed-mean of each potential connection and compared the differences with paired Wilcoxon’s test. For morphology and marker analysis (Figs. 7b-c), we used the unpaired Wilcoxon’s test. We converted the Wilcoxon’s estimate, a robust measure of the difference between groups, and 95% confidence intervals to SPD by multiplying these values by ; where (+) refers to the group that expressed the marker and (-) for the group that did not.