## A. Details of the CCP Model

### A.1. Encodings

In order to parametrize the prior, likelihood and posterior of the CCP model, it is convenient to define first some symmetric encodings for different subsets of the data set $\mathbf{x}$ at iteration $k$. Remember that the notation $\mathbf{x}_k$ indicates that the dataset is split into three groups, $\mathbf{x}_k = (\mathbf{x_a}, x_{d_k}, \mathbf{x_s})$, where

| | |
|---|---|
| $\mathbf{x_a} = (x_{a_1} \dots x_{a_{m_k}})$ | $m_k$ available points for cluster k |
| $x_{d_k}$ | First data point in cluster k |
| $\mathbf{x_s} = (\mathbf{x_{s_1}} \dots \mathbf{x_{s_{k-1}}})$ | Points already assigned to clusters. |

The symmetric encodings we need are:

| Definition | Encoded Points |
|---|---|
| $D_k = \sum_{i=1}^{N_k} \delta_{s_{k,i},d_k} u(x_{s_{k,i}})$ | $x_{d_k}$, the first point in cluster k |
| $A_k^{in} = \sum_{i=1}^{N_k} (1-\delta_{s_{k,i},d_k}) u(x_{s_{k,i}})$ | Points from $\mathbf{x_a}$ that join cluster $k$. |
| $A_k^{out} = \sum_{i=1, b_i=0}^{m_k} u(x_{a_i})$ | Points from $\mathbf{x_a}$ that do not join cluster $k$ |
| $A_k = A_k^{in} + A_k^{out}$ | $\mathbf{x_a}$, all the $m_k$ points available to join $x_{d_k}$ |
| $S_k = D_k + A_k^{in}$ | All points $\mathbf{s}_k$ in cluster k |
| $H_j = \sum_{x:\, x \in \mathbf{s}_j} h(x) \quad j = 1 \dots k-1$ | All points in cluster $j < k$ |
| $G_k = \sum_{j=1}^{k-1} g(H_j)$ | All the clusters $\mathbf{s}_{1:k-1}$. |

(22)

### A.2. Prior and Likelihood

Remember from Section 4 that, having generated $k-1$ clusters $\mathbf{s}_{1:k-1}$, the elements of $\mathbf{s}_k$ are generated in a process with latent variables $d_k, \mathbf{z}_k$ and joint distribution

$$p_\theta(\mathbf{s}_k, \mathbf{z}_k, d_k | \mathbf{s}_{1:k-1}, \mathbf{x}) = p_\theta(\mathbf{b}_k | \mathbf{z}_k, \mathbf{x}_k) p_\theta(\mathbf{z}_k | \mathbf{x}_k) p(d_k | \mathbf{s}_{1:k-1}) \tag{23}$$

where

$$p_\theta(\mathbf{b}_k | \mathbf{z}_k, \mathbf{x}_k) = \prod_{i=1}^{m_k} p_{\theta,i}(b_i | \mathbf{z}_k, \mathbf{x}_k)\,. \tag{24}$$

The priors and likelihood are

$$p(d_k | \mathbf{s}_{1:k-1}) = \begin{cases} 1/|I_k| & \text{for } d_k \in I_k\,, \\ 0 & \text{for } d_k \notin I_k\,, \end{cases} \tag{25}$$

$$p_\theta(\mathbf{z}_k | \mathbf{x}_k) = \mathcal{N}(\mathbf{z}_k | \mu(\mathbf{x}_k), \sigma(\mathbf{x}_k)) \tag{26}$$

$$p_{\theta,i}(b_i | \mathbf{z}_k, \mathbf{x}_k) = \text{sigmoid}[\rho_i(\mathbf{z}_k, \mathbf{x}_k)] \tag{27}$$

and can be defined in terms of

$$\begin{aligned} \mu(\mathbf{x}_k) &= \mu(D_k, A_k, G_k) && (28)\\ \sigma(\mathbf{x}_k) &= \sigma(D_k, A_k, G_k), && (29)\\ \rho_i(\mathbf{z}_k, \mathbf{x}_k) &= \rho(\mathbf{z}_k, x_{a_i}, D_k, A_k, G_k) \qquad i = 1\dots m_k && (30) \end{aligned}$$

where $\mu, \sigma, \rho$ are represented with MLPs. Note that in all the cases the functions depend on encodings in (22) that are consistent with the permutation symmetries dictated by the conditioning information.

## A.3. ELBO

The ELBO that we want to maximize is given by

$$\begin{aligned} &\mathbb{E}_{p(\mathbf{x},\mathbf{s}_{1:K})} \log p_\theta(\mathbf{s}_{1:K}|\mathbf{x}) && (31)\\ &= \mathbb{E}_{p(\mathbf{x},\mathbf{s}_{1:K})} \sum_{k=1}^{K} \log \left[ \sum_{d_k=1}^{N_k} \int d\mathbf{z}_k p_\theta(\mathbf{s}_k, \mathbf{z}_k, d_k|\mathbf{s}_{1:k-1}, \mathbf{x}) \right] && (32)\\ &\geq \mathbb{E}_{p(\mathbf{x},\mathbf{s}_{1:K})} \sum_{k=1}^{K} \mathbb{E}_{q_\phi(\mathbf{z}_k, d_k|\mathbf{s}_{1:k},\mathbf{x})} \log \left[ \frac{p_\theta(\mathbf{s}_k, \mathbf{z}_k, d_k|\mathbf{s}_{1:k-1}, \mathbf{x})}{q_\phi(\mathbf{z}_k, d_k|\mathbf{s}_{1:k}, \mathbf{x})} \right] && (33)\\ &= \mathbb{E}_{p(\mathbf{x},\mathbf{s}_{1:K})} \sum_{k=1}^{K} \mathbb{E}_{q_\phi(\mathbf{z}_k, d_k|\mathbf{s}_{1:k},\mathbf{x})} \log \left[ \frac{p_\theta(\mathbf{b}_k|\mathbf{z}_k, \mathbf{x}_k) p_\theta(\mathbf{z}_k|\mathbf{x}_k) p(d_k|\mathbf{s}_{1:k-1})}{q_\phi(\mathbf{z}_k|\mathbf{b}_k, d_k, \mathbf{x}_k) q_\phi(d_k|\mathbf{s}_{1:k}, \mathbf{x})} \right] && (34) \end{aligned}$$

where we introduced the posterior $q_\phi(\mathbf{z}_k, d_k|\mathbf{s}_{1:k}, \mathbf{x}) = q_\phi(\mathbf{z}_k|\mathbf{b}_k, d_k, \mathbf{x}_k) q_\phi(d_k|\mathbf{s}_{1:k}, \mathbf{x})$. For the first factor we assume a form

$$q_\phi(\mathbf{z}_k|\mathbf{b}_k, d_k, \mathbf{x}_k) = \mathcal{N}(\mathbf{z}_k|\mu_q(D_k, A_k^{in}, A_k^{out}, G_k), \sigma_q(D_k, A_k^{in}, A_k^{out}, G_k)) \qquad (35)$$

where $\mu_q, \sigma_q$ are MLPs. The most challenging aspect of maximizing the ELBO concerns the factor $q_\phi(d_k|\mathbf{s}_{1:k}, \mathbf{x})$, a multinomial over the $N_k$ components of $s_k$ for which we consider next two different approaches.[4]

## A.4. Gumbel-Softmax Relaxation

We start by modeling

$$q_\phi(d_k = s_{k,i}|\mathbf{s}_{1:k}, \mathbf{x}) = \text{Softmax}[\varphi(x_{s_{k,i}}, S_k, A_k^{out}, G_k)] \qquad i = 1\dots N_k \qquad (37)$$

Following (Jang et al., 2016; Maddison et al., 2016), we define

$$y_i = \frac{e^{(\varphi_i + g_i)/\tau}}{\sum_{j=1}^{N_k} e^{(\varphi_j + g_j)/\tau}} \qquad (38)$$

where $\tau$ is a temperature parameter, $g_i$'s are samples from the Gumbel distribution and

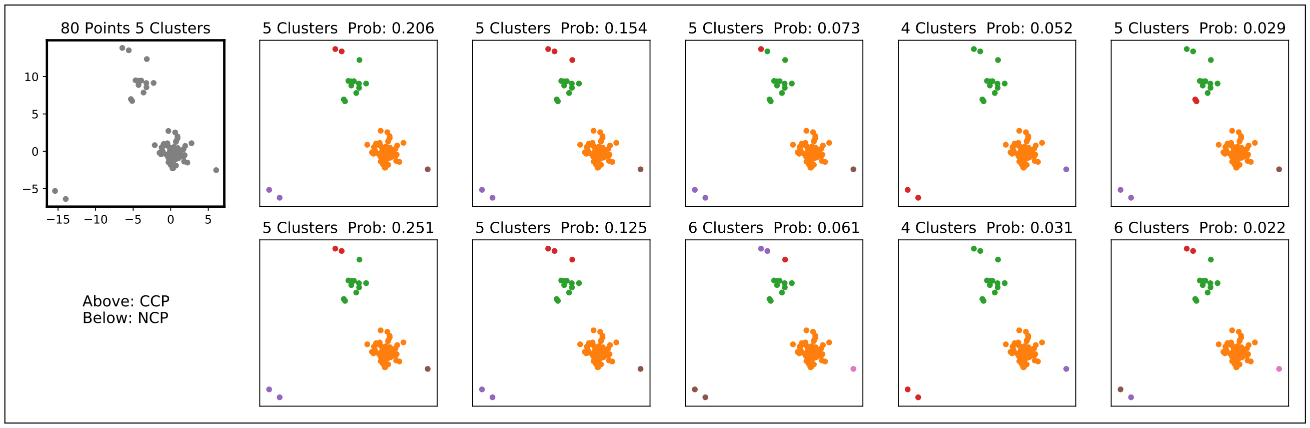
$$\varphi_i = \varphi(x_{s_{k,i}}, S_k, A_k^{out}, G_k)\,. \qquad (39)$$

The $y_i$'s samples are a relaxed version of one-hot samples of $d_k$ from (37) that live in the $N_k$-simplex. To apply the relaxation of $d_k$, we just replace $\delta_{s_{k,i},d_k}$ with $y_i$ in the definitions of $D_k$ and $A_k^{in}$. Following the recommendation of (Maddison et al., 2016), we express the ELBO in terms of

$$t_i = \log(y) \qquad (40)$$

[4]A third alternative would be to model $q_\phi(d_k|\cdot)$ as (37) and compute the expectation exactly

$$\mathbb{E}_{p(\mathbf{x},\mathbf{s}_{1:K})} \sum_{k=1}^{K} \sum_{i=1}^{N_k} q_\phi(d_k = s_{k,i}|\mathbf{s}_{1:k}, \mathbf{x}) \mathbb{E}_{q_\phi(\mathbf{z}_k|\mathbf{b}_k, d_k, \mathbf{x}_k)} \log \left[ \frac{p_\theta(\mathbf{b}_k|\mathbf{z}_k, \mathbf{x}_k) p_\theta(\mathbf{z}_k|\mathbf{x}_k) p(d_k|\mathbf{s}_{1:k-1})}{q_\phi(\mathbf{z}_k|\mathbf{b}_k, d_k, \mathbf{x}_k) q_\phi(d_k|\mathbf{s}_{1:k}, \mathbf{x})} \right] \qquad (36)$$



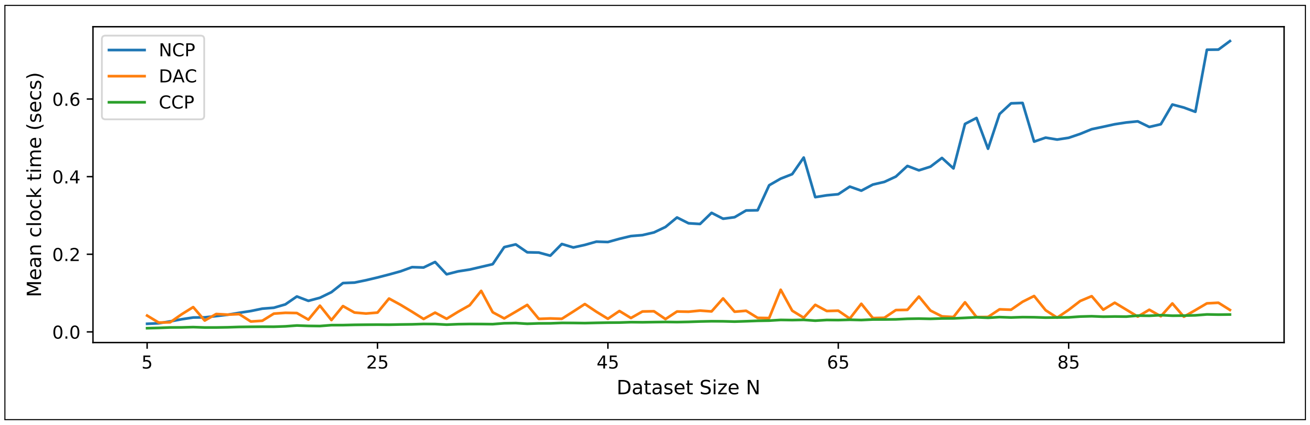


*Figure S1.* **Comparing Samples and Times.** *Above:* Samples from NCP and CCP on the same data set. Both models were trained using the generative model for mixtures of 2D Gaussians from Section 5. Note that the higher probability clusters agree in their labels and approximately in the assigned probabilities. *Below:* Clock time as a function of the dataset size for NCP, CCP and DAC (Lee et al., 2019),[6] all trained and tested with the same 2D Gaussian model as above. Each point in the curve is the average over 25 datasets. For NCP and CCP we sampled 200 full posterior samples, while DAC gives a single deterministic output.

Calling $\kappa_{g,\tau}(t)$ their probability density (see (Maddison et al., 2016) for the explicit form), the relaxed ELBO becomes

$$\mathbb{E}_{p(\mathbf{x},\mathbf{s}_{1:K})} \sum_{k=1}^{K} \mathbb{E}_{q_\phi(\mathbf{z}_k,\mathbf{y}|\mathbf{s}_{1:k},\mathbf{x})} \log \left[ \frac{\prod_{i=1}^{N_k} [p_{\theta,i}(b_i = 1|\mathbf{z}_k,\mathbf{y})]^{1-y_i} \prod_{i=1,b_i=0}^{m_k} p_{\theta,i}(b_i = 0|\mathbf{z}_k,\mathbf{y}) p_\theta(\mathbf{z}_k|\mathbf{y})}{q_\phi(\mathbf{z}_k|\mathbf{y})\kappa_{g,\tau}(t(y))} \right] + const.$$

In this relaxed version the reparametrization trick can be used in the usual way to compute derivatives of $q_\phi$.

## A.5. Uniform Discrete Posterior

A simpler approach to model $q_\phi(d_k|\mathbf{s}_{1:k},\mathbf{x})$ is by approximating it as uniform, given by

$$q(d_k|\mathbf{s}_{1:k}) \quad = \quad \begin{cases} 1/N_k & \text{for } d_k \in \mathbf{s}_k, \\ 0 & \text{for } d_k \notin \mathbf{s}_k. \end{cases} \tag{41}$$

This approximation is very good in cases of well separated clusters. Since $q(d_k|\mathbf{s}_{1:k})$ has no parameters now, this avoids the problem of backpropagation through discrete variables. In the examples we considered, this simpler approach yielded better results, as measured, e.g., by a better agreement in Geweke's test (see Figure 5). So this was the approach we adopted in the results we present in this work.

[6] We used the DAC code available at https://github.com/ICLR2020anonymous/dac.

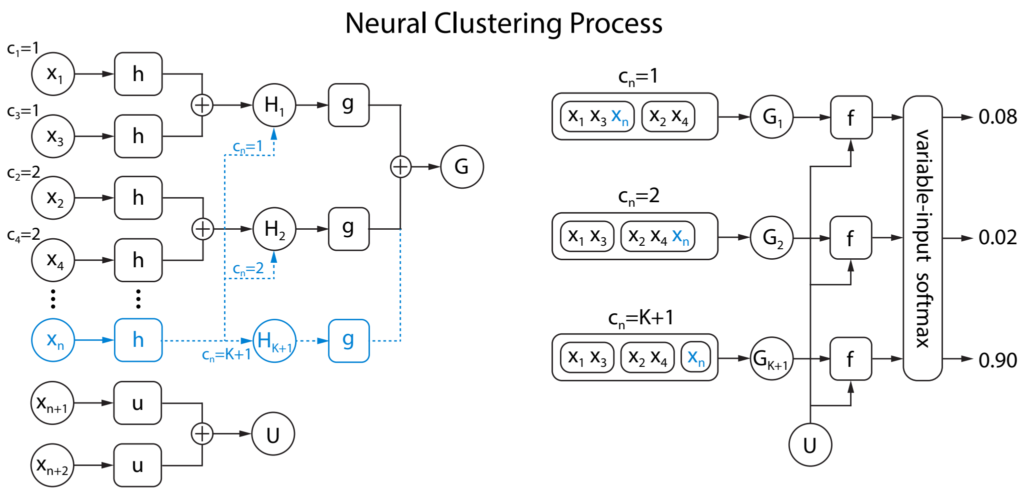


*Figure S2.* **Architecture of the Neural Clustering Process.** The full model is composed by the deep networks $h, g, u, f$. *Left:* After assigning the cluster labels $c_{1:n-1}$, each possible discrete value $k$ for $c_n$ gives a different symmetry-invariant encoding of $x_{1:n}$ into the vector $G_k$, using the functions $h$ and $g$. The remaining, yet-unassigned points $x_{n+1:N}$ are encoded by $u$ and summed into the vector $U$. *Right:* Each pair $G_k, U$ is mapped by $f$ into a real number (logit), which in turn is mapped into the multinomial distribution $q_\theta(c_n|c_{1:n-1}, \mathbf{x})$ via a variable-input softmax.

## B. Neural Clustering Process for Exponential Families

The details of the NCP architecture are fully explained in Section 3, and Figure S2 shows the architecture diagramatically.

In the section we consider the spacial case of exponential family likelihoods, given by

$$\begin{aligned} p(x|\mu) &= e^{\mu\cdot t(x)-\psi(\mu)} m(x) && (42)\\ &= e^{\lambda\cdot h(x)} m(x) && (43)\end{aligned}$$

where $t(x)$ is a vector of sufficient statistics, and we defined

$$\begin{aligned} h(x) &= (1, t(x)) && (44)\\ \lambda &= (-\psi(\mu), \mu) && (45)\end{aligned}$$

Let us denote by $K$ and $K' \geq K$ the total number of distinct values in $c_{1:n}$ and $c_{1:N}$, respectively. Consider the joint distribution

$$p(c_{1:N}, \mathbf{x}, \mu) = p(c_{1:N})p(\mu)\prod_{k=1}^{K'} e^{\lambda_k\cdot\sum_{i:c_i=k} h(x_i)} \prod_{i=1}^{N} m(x_i) \tag{46}$$

from which we obtain the marginal distributions

$$\begin{aligned} p(c_{1:n}, \mathbf{x}) &= \sum_{c_{n+1}\dots c_N} p(c_{1:N}, \mathbf{x}) && (47)\\ &= \sum_{c_{n+1}\dots c_N} \int d\mu p(c_{1:N})p(\mu) \prod_{k=1}^{K'} e^{\lambda_k\cdot(H_k+\sum_{i>n:c_i=k} h(x_i))} \prod_{i=1}^{N} m(x_i) && (48)\\ &= F(H_1, \dots, H_K, h(x_{n+1}), \dots, h(x_N)) \prod_{i=1}^{N} m(x_i) && (49)\end{aligned}$$

where we defined

$$H_k = \sum_{i\leq n, c_i=k} h(x_i) \qquad k = 1\dots K \tag{50}$$

and $H_k = 0$ for $k > K$.

Note now that if $p(c_{1:N})$ is constant, all the dependence of $F$ on $c_{1:n}, x_{1:n}$ is encoded in the $H_k$'s, and $F$ is symmetric under separate permutations of the $H_k$'s and the $h(x_i)$'s for $i > n$. Based on these symmetries we can approximate $F$ as

$$F \simeq e^{f(G,U)} \tag{51}$$

modulo adding to $f$ any function symmetric on all $x_i$'s, where

$$G = \sum_{k=1}^{K} g(H_k) \tag{52}$$

$$U = \sum_{i=n+1}^{N} u(x_i) \tag{53}$$

In the conditional probability we are interested in,

$$p(c_n|c_{1:n-1}, \mathbf{x}) = \frac{p(c_{1:n}, \mathbf{x})}{\sum_{c_n} p(c_{1:n}, \mathbf{x})}, \tag{54}$$

the product of the $m(x_i)$'s in (49) cancels. Similarly, adding to $f$ a function symmetric on all $x_i$'s leaves invariant our proposed approximation

$$q_\theta(c_n = k|c_{1:n-1}, \mathbf{x}) = \frac{e^{f(G_k,U)}}{\sum_{k'=1}^{K+1} e^{f(G_{k'},U)}} \qquad k = 1 \dots K+1\,. \tag{55}$$

## C. Monitoring global permutation invariance

As mentioned in Section 7, we must verify the symmetry of the posterior likelihood under global permutations of all the data points. We show such a check in Figure S3.

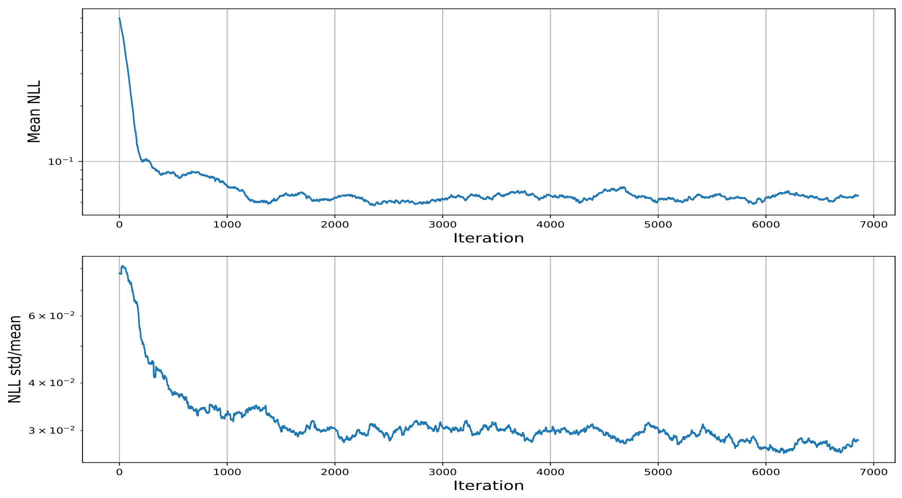


*Figure S3.* **Global permutation invariance.** Training curves for the NCP model of 2D Gaussians in Section 2. Each minibatch was evaluated for 8 random permutations of the order of the points in the dataset. *Above:* Mean of the NLL over the permutations. *Below:* NLL standard deviation/NLL mean. Note that the ratio is of order $10^{-2}$.

## D. Details of spike sorting using NCP

**Data preprocessing.** Training and test data come from the retinal recordings in (Chichilnisky & Kalmar, 2002) using a 512-channel 2D hexagonal MEA with 20 kHz sampling rate. After spike detection (Lee et al., 2017), each multi-channel spike waveform was assigned to the channel where the waveform has the maximum peak-to-peak (PTP) amplitude (i.e. the center channel, ch0). This partitioned the recording data by channel such that each center-channel-based partition only contains multi-channel spike waveforms centered at that channel. Each spike waveform is represented as a $7 \times 32$ array containing the 32 time steps surrounding the peak from the center channel and the same time window from the 6 immediate neighbor channels (Figure 6). These $7 \times 32$ arrays are the spikes on which clustering was performed.

**Neural architecture for NCP spike sorting.** The overall architecture is the same as the one described in Section 3 and Figure S2. To extract useful features from the spatial-temporal patterns of spike waveforms, we use a 1D ConvNet as the $h$ and $u$ encoder functions. The convolution is applied along the time axis, with each electrode channel treated as a feature dimension. The ConvNet uses a ResNet architecture (He et al., 2016) with 4 residual blocks, each having 32, 64, 128, 256 feature maps (kernel size = 3, stride = [1, 2, 2, 2]). The last block is followed by an averaged pooling layer and a final linear layer. The outputs of the ResNet encoder are the $h_i$ and $u_i$ vectors of NCP, i.e. $h_i = \text{ResNetEncoder}(x_i)$. We used $d_h = d_u = 256$. The other two functions, $g$ and $f$, are identical to those in the 2D Gaussian example.

**Training NCP using synthetic data.** To train NCP for spike clustering, we created synthetic labeled training data using a MFM generative model (Miller & Harrison, 2018) of noisy spike waveforms that mimic the distribution of real spikes:

$$N \sim \text{Uniform}[N_{min}, N_{max}] \quad (56)$$

$$K \sim 1 + \text{Poisson}(\lambda) \quad (57)$$

$$\pi_1 \dots \pi_K \sim \text{Dirichlet}(\alpha_1, \dots, \alpha_K) \quad (58)$$

$$c_1 \dots c_N \sim \text{Cat}(\pi_1, \dots, \pi_K) \quad (59)$$

$$\mu_k \sim p(\mu) \quad k = 1 \dots K \quad (60)$$

$$x_i \sim p(x_i|\mu_{c_i}, \Sigma_s \otimes \Sigma_t) \quad i = 1 \dots N \quad (61)$$

Here, $N$ is the number of spikes between $[200, 500]$. The number of clusters $K$ is sampled from a shifted Poisson distribution with $\lambda = 2$ so that each channel has on average 3 clusters. $\pi_{1:K}$ represents the proportion of each cluster and is sampled from a Dirichlet distribution with $\alpha_{1:K} = 1$. The training spike templates $\mu_k \in \mathbb{R}^{7\times 32}$ are sampled from a reservoir of 957 ground-truth templates not present in any test data, with the temporal axis slightly jittered by random resampling. Finally, each waveform $x_i$ is obtained by adding to $\mu_{c_i}$ Gaussian noise with covariance given by the Kronecker product of spatial and temporal correlation matrices estimated from the training data. This method creates spatially and temporally correlated noise patterns similar to real data (Figure S4). We trained NCP for 20000 iterations on a GPU with a batch size of 32 to optimize the NLL loss by the Adam optimizer (Kingma & Ba, 2015). A learning rate of 0.0001 was used (reduced by half at 10k and 17k iterations).

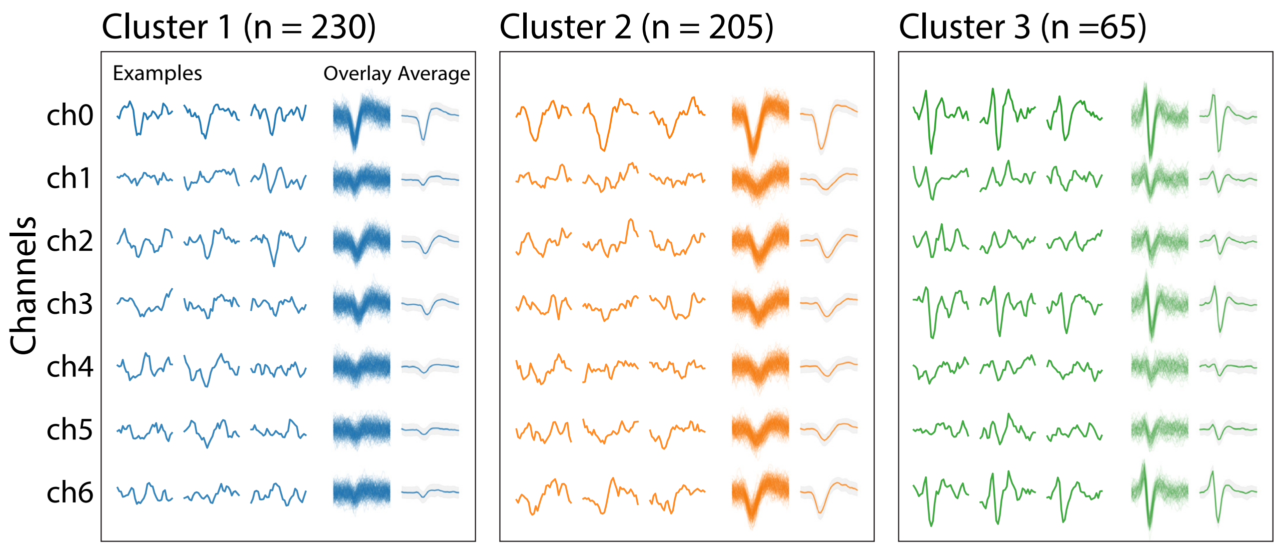


*Figure S4.* **Synthetic data examples.** Example of 500 synthetic spikes from 3 clusters.

**Probabilistic spike clustering using NCP.** At inference time, we fed the 7 x 32 arrays of spike waveforms to NCP, and performed GPU-parallelized posterior sampling of cluster labels (Figure S2 and Figure 6). Using beam search (Graves, 2012; Sutskever et al., 2014) with a beam size of 150, we were able to efficiently sample 150 high-likelihood clustering configurations for 2000 spikes in less than 10 seconds on a single GPU. After clustering, we obtained a spike template for each cluster as the average shape of the spike waveforms. The clustering configuration with the highest probability was used for most experiments.

**The spike sorting pipelines for real and hybrid data.** The real data is a 49-channel, 20-minute retina recording with white noise stimulus. To create the hybrid test data, 20 ground-truth spike templates were manually selected from a 49-channel test recording and inserted into another test dataset according to the original spike times.

For NCP and vGMFM, we performed clustering on 2000 randomly sampled spikes from each channel (clusters containing less than 20 spikes were discarded), and assigned all remaining spikes to a cluster based on the L2 distance to the cluster centers. Then, a final set of unique spike templates were computed, and each detected spike was assigned to one of the templates. The clustering step of vGMFM uses the first 5 PCA components of the spike waveforms as input features. For Kilosort, we run the entire pipeline using the Kilosort2 package (Pachitariu, 2019). After extracting spike templates and RFs from each pipeline, we matched pairs of templates from different methods by L-infinity distance and pairs of RFs by cosine distance.

**Electrode drift in real MEA data.** The NCP spike sorting pipeline described above does not take into consideration electrode drift over time, which is present in some real recording data. As a step towards addressing the problem of spike sorting in the presence of electrode drift (Calabrese & Paninski, 2011; Shan et al., 2017), we describe in Sup. Material F a generalization of NCP to handle data in which the per-cluster parameters (e.g. the cluster means) are nonstationary in time.

## E. Experimental results for NCP spike sorting

**Synthetic Data.** We run NCP and vGMFM on 20 sets of synthetic test data each with 500, 1000, and 2000 spikes. As the ground-truth cluster labels are known, we compared the clustering quality using Adjusted Mutual Information (AMI) (Vinh et al., 2010). The AMI of NCP is on average 11% higher than vGMFM (SM Figure S5), showing better performance of NCP on synthetic data.

**Real Data.** We run NCP, vGMFM and Kilosort on a retina recording with white noise stimulus as described in SM Section D, and extracted the averaged spike template of each cluster (i.e. putative neuron). Example clustering results in SM Figure S6 (top) show that NCP produces clean clusters with visually more distinct spike waveforms compared to vGMFM. As real data do not come with ground-truth cluster labels, we compared the spike templates extracted from NCP and Kilosort using retinal receptive field (RF), which is computed for each cluster as the mean of the stimulus present at each spike. A clearly demarcated RF provides encouraging evidence that the spike template corresponds to a real neuron. Side-by-side comparisons of matched RF pairs are shown in SM Figure S6 (bottom-left) and SM Figure S8. Overall, NCP found 103 templates with clear RFs, among which 48 were not found by Kilosort, while Kilosort found 72 and 17 of them were not found by NCP (SM Figure S6 bottom-right). Thus NCP performs at least as well as Kilosort, and finds many additional templates with clear RFs.

**Hybrid Data.** We compared NCP against vGMFM and Kilosort on a hybrid recording with partial ground truth as in (Pachitariu et al., 2016). Spikes from 20 ground-truth templates were inserted into a real recording to test the spike sorting performance on realistic recordings with complex background noise and colliding spikes. As shown in SM Figure S7, NCP recovered 13 of the 20 injected ground-truth templates, outperforming both Kilosort and vGMFM, which recovered 8 and 6, respectively.

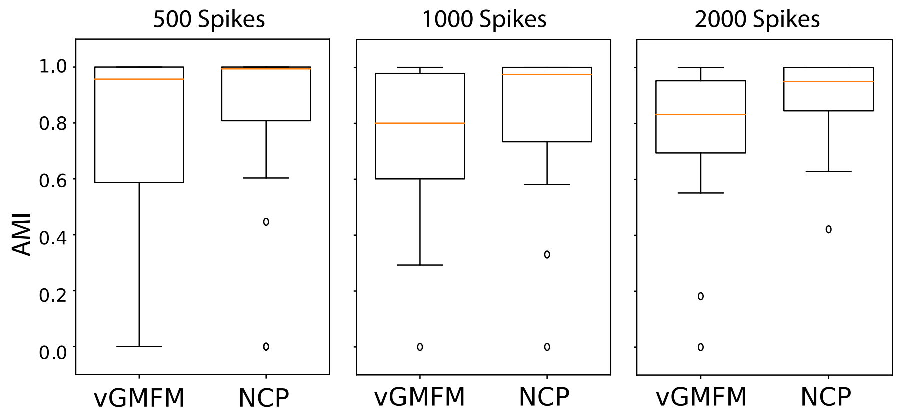


*Figure S5.* **Clustering synthetic data.** The AMI scores for clustering 20 sets of 500, 1000, and 2000 unseen synthetic spikes.

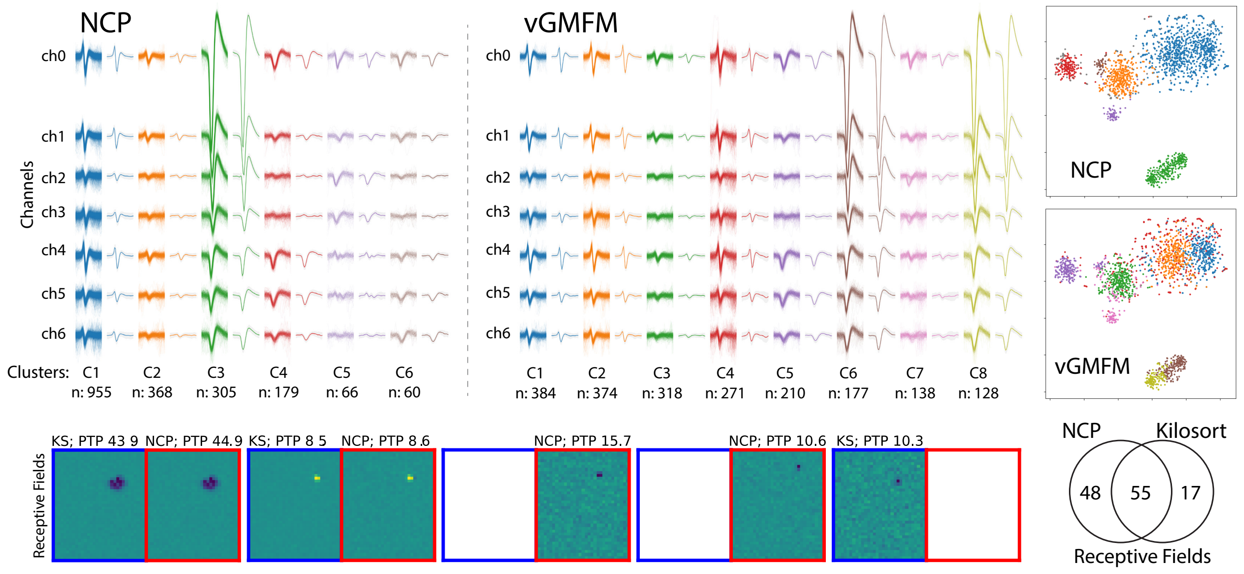


*Figure S6.* **Spike sorting on real data.** 2000 spikes from real data were clustered by NCP (*top-left*) and vGMFM (*top-mid*). Each column shows the spikes assigned to one cluster (overlaying traces and their average). Each row is one electrode channel. *Top-right:* t-SNE visualization of the spike clusters. *Bottom-left:* Example pairs of matched RFs recovered by NCP (red boxes) and Kilosort (blue boxes). Blank indicates no matched counterpart. *Bottom-right:* Venn diagram of recovered RFs.

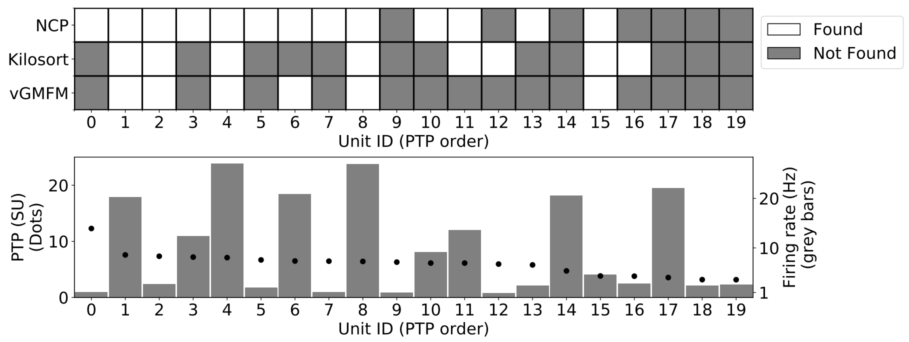


*Figure S7.* **Spike sorting on hybrid data.** *Top:* NCP, Kilosort, vGMFM recovered 13, 8, and 6 of the 20 injected ground-truth templates. *Bottom:* Peak-to-peak (PTP) size and firing rate of each injected template. (Smaller templates with lower firing rates are more challenging.)

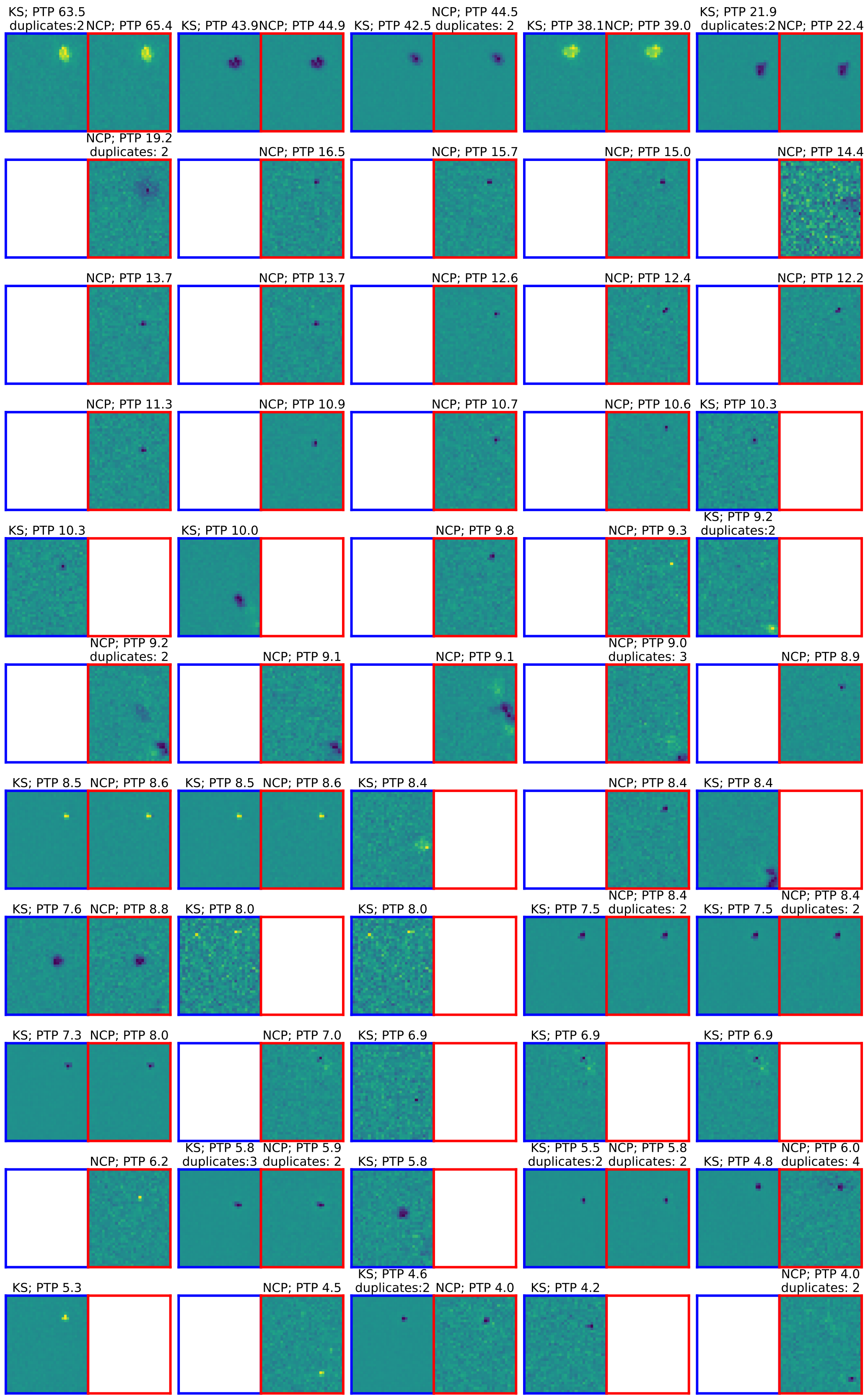


*Figure S8.* **Spike sorting on real data.** Receptive fields of 55 randomly selected pairs of units recovered from Kilosort and NCP spike sorting. (Red boxes indicate units found by NCP; blue boxes by Kilosort.) Both approaches find the spikes with the biggest peak-to-peak (PTP) size. For smaller-PTP units often one sorting method finds a cell that the other sorter misses. NCP and KS find a comparable number of units with receptive fields here, with NCP finding a few more than KS; see text for details.

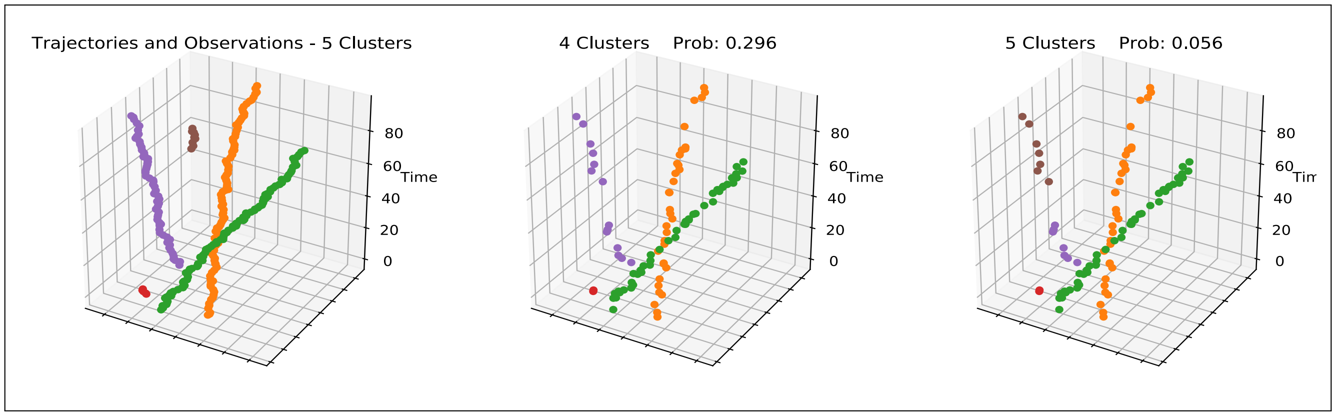


*Figure S9.* **Neural Particle Tracking.** *Left:* Time trajectories of 5 2D particles. Note that particles can appear or disappear at arbitrary times. *Middle and right:* Two posterior samples. Note that since only one particle is observed at each time, a particle not observed for some time leads to a possible ambiguity on the number of particles. (Best seen in color.)

## F. Particle tracking

Inspired by the problem of electrode drift (Calabrese & Paninski, 2011; Pachitariu, 2019; Shan et al., 2017), let us consider now a generative model given by

$$
\begin{aligned}
c_t &\sim p(c_t|c_1,\dots,c_{t-1}) \qquad t=1,\dots,T && (62)\\
\mu_{k,t} &\sim p(\mu_{k,t}|\mu_{k,t-1}) \qquad k=1\dots K \qquad t=1,\dots,T && (63)\\
x_t &\sim p(x_t|\mu_{c_t,t}) \qquad t=1,\dots,T && (64)
\end{aligned}
$$

In this model, a cluster corresponds to the points along the time trajectory of a particle, and (63) represents the time evolution of the cluster parameters. The cluster labels $c_t$ indicate which particle is observed at time $t$, and note that particles can in principle appear or disappear at any time.

To take the time evolution into account, we let particles influence one another with a weight that depends on their time distance. For this, let us introduce a time-decay constant $b > 0$, and generalize the NCP equations to

$$
\begin{aligned}
H_{k,t} &= \sum_{t'=1:c_{t'}=k}^{t} e^{-b|t-t'|} h(x_{t'}) \qquad k=1\dots K\,, && (65)\\
G_t &= \sum_{k=1}^{K} g(H_{k,t})\,, && (66)\\
U_t &= \sum_{t'=t+1}^{T} e^{-b|t-t'|} u(x_{t'})\,. && (67)
\end{aligned}
$$

The conditional assignment probability for $c_t$ is now

$$
q_\theta(c_t=k|c_{1:t-1},\mathbf{x}) = \frac{e^{f(G_{k,t},U_t)}}{\sum_{k'=1}^{K+1} e^{f(G_{k',t},U_t)}} \tag{68}
$$

for $k=1\dots K+1$. The time-decay constant $b$ is learnt along with all the other parameters. We can also consider replacing $e^{-b|t-t'|}$ with a general distance function $e^{-d(|t-t'|)}$. Figure S9 illustrates this model in a simple 2D example. We call this approach Neural Particle Tracking.

## G. Neural architectures in the examples

To train the networks in the examples, we used stochastic gradient descent with Adam (Kingma & Ba, 2015), with learning rate $10^{-4}$. The number of samples in each mini-batch were: 1 for $p(N)$, 1 for $p(c_{1:N})$, 64 for $p(\mathbf{x}|c_{1:N})$. The architecture of the functions in each case were:

**NCP: 2D Gaussians**

- $h$: MLP [2-256-256-256-128] with ReLUs
- $u$: MLP [2-256-256-256-128] with ReLUs
- $g$: MLP [128-256-256-256-256] with ReLUs
- $f$: MLP [384-256-256-256-1] with ReLUs

**NCP: MNIST**

- $h$: 2 layers of [convolutional + maxpool + ReLU] + MLP [320-256-128] with ReLUs
- $u$: same as $h$
- $g$: MLP [256-128-128-128-128-256] with ReLUs
- $f$: MLP [384-256-256-256-1] with ReLUs