Scalable Gaussian process inference of neural responses to natural images

Matías A Goldin, Samuele Virgili, Matthew Chalk

Scalable Gaussian process inference of neural responses to natural images

Matías A. Goldin, Samuele Virgili, Matthew Chalk*

Institut de la Vision, Sorbonne Université, INSERM, CNRS, Paris

*contact: matthew.chalk@inserm.fr

1 Abstract

Predicting the responses of sensory neurons is a long-standing neuroscience goal. However, while there has been much progress in modeling neural responses to simple and/or artificial stimuli, predicting responses to natural stimuli remains an ongoing challenge. On the one hand, deep neural networks perform very well on certain data-sets, but can fail when data is limited. On the other hand, gaussian processes (GPs) perform well on limited data, but are poor at predicting responses to high-dimensional stimuli, such as natural images. Here we show how structured priors, e.g. for local and smooth receptive fields, can be used to scale up GPs to model neural responses to high-dimensional stimuli. With this addition, GPs largely outperform a deep neural network trained to predict retinal responses to natural images, with largest differences observed when both models are trained on a small data-set. Further, since they allow us to quantify the uncertainty in their predictions, GPs are well-suited to closed-loop experiments, where stimuli are chosen actively so as to collect 'informative' neural data. We show how GPs can be used to actively select which stimuli to present, so as to: (i) efficiently learn a model of retinal responses to natural images, using little data, and (ii) rapidly distinguish between competing models (e.g. a linear vs a non-linear model). In the future, our approach could be applied to other sensory areas, beyond the retina.

2 Introduction

Accurately predicting the responses of neurons in visual areas is an important goal in visual neuroscience. However, while there has been much progress in modelling sensory neural responses to simple and/or artificial stimuli (e.g. moving gratings/checkerboard) [1, 2], predicting neural responses to natural stimuli remains an ongoing challenge [3]. Even in the retina, arguably the simplest visual circuit, this problem is hard to solve [4, 5, 6].

One approach, that has had a great deal of success in recent years, is to train deep convolutional neural networks (CNNs) to predict neural responses to presented stimuli [7, 8]. However, this approach can suffer from two important limitations. First, CNNs typically require large amounts of data to train, whereas in many experimental settings data may be extremely limited. In such cases, CNNs will likely over-fit noise in the data, or alternatively, require strong regularisation which can bias the results [8]. Second, designing a deep CNN to work well on a given neural data-set may require a number of design choices (e.g. about the network architecture, non-linearities, etc.), and it may not be clear how each of these choices affect the final results [9].

Another approach has been to fit neural responses using a type of non-parametric Bayesian model, called Gaussian processes (GPs) [10, 11, 13, 14]. These models can be trained on small data-sets, in comparison to CNNs, while avoiding overfitting. They also naturally capture uncertainty, which can be useful for assessing the robustness of their predictions, as well as for applications such as active learning, where stimuli are chosen in closed-loop based on previously recorded responses [11, 12, 13]. Unfortunately however, GPs typically perform very badly at predicting responses to high-dimensional inputs, such as natural images or movies [15, 16]. As a result, in visual neuroscience, they have previously only been used to model neural responses to low-dimensional artificial stimuli, such as oriented gratings.

Here we propose a method to scale up GPs to predict neural responses to high-dimensional stimuli, such as natural images. To do this, we extend the approach of Park & Pillow [17, 12], who showed how prior knowledge about visual neuron responses (e.g. that they have spatially local and smooth receptive fields), can be used to constrain and simplify inference of a simple, one-layer, linear-nonlinear (LN) model. We show that by incorporating this idea into a GP framework we can achieve good fits of retinal neuron responses to high-dimensional natural images, which out-perform state-of-the-art CNN models, as well as the LN models considered by Park & Pillow and others. Moreover, our GP model continues to perform well even when trained on very little data, and when a CNN model breaks down. Finally, we describe how our GP framework can be used to design a closed-loop experiment, where one chooses which stimuli to present on-line, based on previously recorded responses. We demonstrate, using retinal recordings data, how our closed-loop algorithm can be used to quickly and efficiently learn a model of neural responses, as well as to discriminate between alternative competing scientific hypotheses.

3 Results

3.1 GP framework

Let us imagine we have a data-set, $\mathcal{D}$, comprising N stimulus response pairs, (x,r). We denote the stimulus, x, by an n_x-d vector (n_x could be, e.g., the number of pixels in a presented image), while the response, r, corresponds to the number of spikes fired by a neuron in response to a presented stimulus. Our goal will be to use this data-set to learn a function, $f(x)$ (which we call the neuron's response curve) describing how many spikes the neuron fires on average to any given stimulus, x. Learning $f(x)$ can be difficult due to three main factors: (i) if we have limited, or noisy, data (small N); (ii) if the stimulus is high dimensional (large n_x); (iii) if the true response function, $f(x)$, is highly non-linear.

Given limited training data (small N), a model that is too complex (i.e. too many parameters) will likely over-fit the noise, and as a result, generalise poorly to new stimuli (**Fig 1a, red**). Likewise, a model that is too simple (i.e. too few parameters), will under-fit, also leading to poor predictions (**Fig 1a blue**). To overcome this problem of over/under-fitting, we can use Bayesian methods, such as Gaussian processes (GP). Instead of making a point estimate for the model parameters, GPs allow us to infer the full posterior probability distribution over response functions, $p(f|\mathcal{D})$. An important property is that they automatically apply Occam's razor, so as to adjust the inferred model complexity based on the available training data, and thus avoiding under/over-fitting [18]. Further, they are non-parametric, so prevent us having to make restrictive assumptions about the response function, $f(x)$.

Bayesian models require us to explicitly specify our prior assumptions in advance. For GPs, the prior is expressed via a kernel function, $K(x, x')$, which specifies the prior covariance between the response to two inputs, x and x'. A common choice are 'stationary kernels', which depend only on the distance between x and x', while being invariant to translations (i.e. $K(x, x') = K(x+z, x'+z)$). For example, the commonly used square exponential kernel assumes that the function is smooth over a given length-scale, σ. However, GPs with stationary kernels are typically poor at extrapolating to stimuli far away ($\gg \sigma$) from the training data (**Fig 1b, green**). Moreover, their ability to extrapolate becomes exponentially worse as we increase the number of input dimensions (since fewer training data points lie at short distance, σ, from a given input, x) [15]. This can be partially overcome by using a non-stationary kernel, which allows for long-range correlations between the responses to different high-dimensional inputs (at the expense of stronger assumptions about the response function)(**Fig 1b, red**). Here we choose to use an 'arc-cosine' kernel [21], which is equivalent to a two layer neural network, with infinitely many rectified-linear units in the middle layer. However, even these kernels can perform badly if the input dimensionality is very large. As a result, GPs have typically been used to treat problems with small training data-sets (where over-fitting is a problem) and low-dimensional inputs (where standard kernels work well) (**Fig 1c, yellow**). In contrast, artificial neural networks, which can deal with high dimensional inputs, typically require large data-sets to train (**Fig 1c, grey**).

To scale a GP to predict neural responses to high-dimensional stimuli, we need to incorporate additional prior information, to constrain and simplify inference. In the visual system, neurons typically respond to stimuli in a small region of space; i.e. they have spatially localised receptive fields (RFs). In addition, visual neuron RFs are typically spatially smooth. We thus assume a two-layer neural network, in which the linear weights in the first layer are *a priori* spatially localised and smooth (**Fig 1d**). In the limit where we have infinitely many units in the middle layer, this corresponds to a GP with the following kernel:

$$K(x, x') = \frac{1}{2\pi}\sqrt{x'^T C x' + \sigma_0^2}\sqrt{x^T C x + \sigma_0^2}\left(\sin\psi + (\pi - \psi)\cos\psi\right) \tag{1}$$

where σ_0 is a constant hyper-parameter, and

$$\psi = \cos^{-1}\left(\frac{x^T C x' + \sigma_0^2}{\sqrt{x^T C x + \sigma_0^2}\sqrt{x'^T C x' + \sigma_0^2}}\right), \tag{2}$$

while C is an $n \times n$ matrix, specifying the covariance between input weights. This covariance matrix encodes our prior assumption of smooth spatially local neural RFs, with ij^{th} element given by:

$$C_{ij} = e^{-\frac{\|\xi_i - \xi_0\|}{4\beta^2} - \frac{\|\xi_j - \xi_0\|}{4\beta^2} - \frac{\|\xi_i - \xi_j\|^2}{2\rho^2}} \tag{3}$$

where ξ_i & ξ_j are the 2-d spatial coordinates of the i^{th} and j^{th} image pixel, respectively, while ξ_0, β, ρ, are hyperparameters determining the location of the RF centre, its size, and smoothness, respectively. ~~As is standard in GP inference, the hyper-parameters determining the location and smoothness of neural RFs are learned from data, so as to maximise the lower bound on the log-likelihood. Note that this extends the work of Park & Pillow~~ [17] ~~to a two-layer neural network, thus allowing us to consider non-linear response curves. While our focus here is the visual system, similar methods could be used for other sensory areas. For example, for audition we could use prior assumptions about spectro-temporal RFs.~~

In addition, we need to adapt the standard GP framework, whose output can take any real value, to our data, with positive integer spike counts. To do this, we transform the GP output, $\lambda(x)$, by a positive monotonic nonlinearity, to give a mean spike count,

$$f(x) = e^{A\lambda(x) + \lambda_0}, \tag{4}$$

where A and λ_0 are hyperparameters of the model. Finally, we apply Poisson noise to generate the observed spike count, r (**Fig. 1d**). Inference is performed using a variational algorithm, in which we assume a multivariate Gaussian posterior over λ, whose parameters are learned to maximise a lower bound on the log-likelihood. Further, we use the inducing point framework introduced by Hensman, Fusi & Laurence [19] (which involves conditioning the variational posterior over a limited number of $n \ll N$ 'inducing' data points) to make inference feasible when N is large (see Methods section 5.3). The 7 hyperparameters (5 hyperparameters of the kernel, plus the gain A and bias λ_0) were learned from data so as to maximise a lower bound on the log-likelihood.

3.2 Predicting retinal responses to natural images

To test the performance of our framework we trained it on a data-set comprising the recorded spiking responses of 41 retinal ganglion cells to a set of presented natural images (Methods section 5.6 [5]). This data-set gives rise to all of the challenges listed in the previous section since: (i) the stimulus is high dimensional (i.e. large number of pixels $n_x = 108^2 = 11,664$); (ii) there are relatively few trials ($N \approx 3000 \ll n_x$); (iii) from previous work, we know that retinal ganglion cell responses to naturalistic stimuli are often highly non-linear [5].

We measured the predicted versus the observed mean spike count for two example neurons, on 30 hold-out stimuli repeated 30 times each not present in the training data-set (**Fig. 2a** & **c**). Since the model is probabilistic, it returns the uncertainty in its predictions (vertical lines), while the uncertainty in the data (horizontal lines) can be estimated using bootstrap. For both cells, the model achieves a reasonable prediction of the observed spike count, given the estimated experimental and modelling uncertainty (see later for quantitative assessment).

As well as fitting the mean spike count, we wanted to see if the GP model was able to predict other qualitative aspects of how the neural response curve depends on the presented image. In machine learning, a common approach to 'interpret' the output of an artificial neural network is to plot the gradient of the output with respect to its input [20]. This gives an indication of which input features the network is 'sensitive to' (i.e. which stimulus features increase/decrease the output). To do this, we plotted the inferred gradient of the firing rate, $\nabla f(x)|_{x_0}$, for a selection of 'reference images' x_0 (**Fig. 2b** & **d**, left column). We then compared this to an indepedent estimate of the gradient, obtained by probing experimentally the neuron's response to perturbations around the reference images, by presenting stimuli consisting of small white noise added on top of these images (**Fig. 2b** & **d**, second to left). ~~For historical reasons,~~ We call this estimated gradient, the 'local spike-triggered-average' (LSTA). It represents ~~the~~ the stimulus perturbation that produces the maximum change in the firing rate of the cell, given a reference image (Method section 5.6).

Most neurons exhibited an LSTA that varied qualitatively for different natural images, with occasional reversals in sign (**Fig. 2b**) and/or shape (**Fig. 2d**). In most cases, the GP model was able to reproduce these qualitative changes in the shape and sign of the recorded LSTA (**Fig. 2b** & **d**, middle column). In contrast, a single layer LN model was not able to reproduce these changes (**Fig. 2b** & **d**, right column), since by construction the gradient of the response function was the same for all stimuli (varying only in its total magnitude, due to the static non-linearity). We also compared our results to a 2-layer convolutional neural network (CNN), proposed previously to model this data-set [5] (**Fig. 2b** & **d**, second right; Methods section 5.7). While the CNN was able to produce most qualitative changes in the sign of the LSTAs, they appeared larger and less closely resembled the shape of the ground truth LSTAs, obtained using the perturbation stimuli. This could be because of the restricted class of functions captured by the CNN (i.e. 2-layer network with convolutional first layer), while in contrast, the GP can in theory fit any function given enough data.

To quantify the model performance, we estimated the lower bound on the log-likelihood achieved with the GP model with arc-cosine kernel, compared with a linear or quadratic kernel (with a prior for smooth & local receptive fields) (**Fig 2e**). Perhaps unsurprisingly, given the observed non-linear behaviour (**Fig. 2b** & **d**), the arc-cosine kernel greatly outperformed the linear kernel for every recorded cell ($p < 0.001$, signed-rank test). Interestingly, however, it also performed better than the quadratic kernel, which assumes that $\lambda(x)$ is quadratic, and which has been used previously to model retinal neural responses [22]. Removing the prior for smooth local receptive fields (i.e. by setting $C = I$) resulted in a significant drop in the log-likelihood ($p < 0.001$, signed rank test).

To compare our model to other 'non-Bayesian' models, such as the two layer CNN plotted in **Fig. 2b** & **d** and a two layer LNLN model (Methods section 5.8), we plotted the noise corrected correlation, or 'adjusted R^2', for each model on the 30 repeated hold-out images. To do this, we divide responses into even and odd trials and estimate their mean values, $\bar{r}_o$ and $\bar{r}_e$. We defined the reliability as the correlation between $\bar{r}_o$ and $\bar{r}_e$, $c_{\bar{r}_o,\bar{r}_e}$. Then, we estimated the adjusted R^2 value given a model prediction $\hat{r}$:

$$\text{Adjusted } R^2 \equiv \left(\frac{\frac{1}{2}(c_{\hat{r},\bar{r}_o} + c_{\hat{r},\bar{r}_e})}{\sqrt{c_{\bar{r}_o,\bar{r}_e}}} \right)^2 \tag{5}$$

This metric, introduced by Keshishian et al. [23], corresponds to the R^2 goodness of fit between observed and predicted firing rates, normalised based on the across-trial noise in the data. An adjusted R^2 of 1 means that the model predictions are as good as possible given the inherent noise in the data; an adjusted $R^2 < 0$ means that we would perform better by replacing the model predictions by a single number (the mean firing rate).

The GP performed well on this metric, with an adjusted R^2 >0.8 for 36/41 cells, and >0.6 for 40/41 cells. In contrast, while the CNN performed better than either the LNLN model ($p = 0.002$, signed rank) and the LN model ($p < 0.001$, signed rank test), it performed worse than the GP ($p < 0.001$, signed rank test), with an adjusted R^2 >0.8 for only 19/41 cells, and >0.6 for 34/41 cells (**Fig. 2f**). All models out-performed the GP when we removed the prior for smooth local RFs (i.e. by setting $C = I$).

3.3 Inference with little data

As stated earlier, one advantage of Bayesian models, such as GPs, is that they automatically apply Occam's razor, so as to avoid overfitting when trained on limited data. To test this, we trained our GP model on a small subset of $N = 500$ out of 3160 randomly selected trials. **Fig. 3a** shows the firing rate predicted by the GP model trained on 500 trials versus the model trained on $N = 3160$ trials, for the 30 repeated hold-out stimuli (here, we plot the cell shown in Fig. 1b-c). The predicted firing rate is similar in both cases, despite reducing the original data-set by a factor of 6. The same goes for the predicted LSTAs (**Fig. 3b**), which show very little difference when the model is trained on 500 versus 3160 trials. In contrast, the LSTAs predicted by the CNN appear drastically different.

To quantify the performance of both models when we reduce the size of the training data-set, we plotted the adjusted R^2 for the 30 hold-out stimuli for both GP and CNN models (**Fig 3c**). While the performance of the GP model was reduced, it was still comparable to the performance of the CNN trained on the full data-set of 3160 trials. In contrast, the adjusted R^2 achieved by the CNN model was greatly reduced when trained on fewer trials. Strikingly, the GP trained on 500 trials performed similarly to the CNN trained on 3160 trials, with no signicant difference in the adjusted R^2 ($p = 0.21$, signed rank test).

3.4 Approximating GP with a finite two-layer network

As stated earlier, our GP model is equivalent to a two layer neural network (i.e. an LNLN model) with infinitely many units in the middle layer (**Fig 1d**). We were thus interested to see how the GP model compared to an LNLN model with a finite number of units (**Fig 4a**). To do this, we constructed an LNLN model, with n hidden units whose linear filters were sampled from a multivariate Gaussian, $w \sim \mathcal{N}(w|0, C)$ (C is the convariance matrix enforcing smooth & localised RFs, with hyperparameters learned beforehand using the full GP model, with infinite units). For illustration, we focused on the same cell plotted in Fig 2a-b and Fig 3a-b. With $n = 2000$ units, the LNLN model behaved near identically to the full GP model, with $n = \infty$ units ($R^2 = 0.99$ for predicted firing rates to 3160 training images).

We next sought to find out how this LNLN model would perform when we systematically removed units. To do so, we assumed a Gaussian prior over the activation of each unit, with variance learned from data so as to maximise the log-likelihood. Adding a cost (i.e. a hyper-prior) that penalised large prior variance, resulted in a sparse model, where many of the hidden units were completely inactive. By varying this cost, we could progressively decrease the number of hidden units (see Methods).

Interestingly, we could drastically decrease the number of hidden units, to 22 units, while barely affecting the responses predicted by the model (**Fig 4b**). Reducing the number of units still further (e.g. to 2 units) allowed for a more interpretable model, at the expense of reduced performance (**Fig 4c**). For example, we could understand how changes in polarity of the cell's LSTA come about, by observing which of the two hidden units was activated by each natural image (**Fig 4d**).

3.5 Active learning of the response curve

When we have little data, there may be few stimuli that generate useful responses for learning the neural response curve, f. To overcome this, we propose a closed-loop approach where we use recorded responses to actively choose which stimuli to present. After each recorded stimulus-response pair (x, r), the GP model is updated, resulting in new predictions for the mean and variance of the GP output, $\lambda(x)$ ($\mu(x)$ & $\sigma(x)^2$, respectively) (**Fig 5a**). We use these predictions to evaluate a utility function, $U(x)$, which quantifies how 'useful' the neural response to a new stimulus, x, is likely to be. We choose the stimulus x for which $U(x)$ is maximal to present on the next trial.

We want recorded responses, r, to be maximally informative about the response function, f. Therefore, we define the utility function as the mutual information between the recorded response, r, and the response function, f, given a new presented stimulus, x, and data-set, $\mathcal{D}$:

$$U(x) = I(r; f|x, \mathcal{D}). \tag{6}$$

We can expand this as follows:

$$U(x) = H(r|x, \mathcal{D}) - \langle H(r|f, x)\rangle_{p(f|x,\mathcal{D})}, \tag{7}$$

where the first term denotes the conditional response entropy, given x and $\mathcal{D}$, and the second term is the expectation (over $p(f|x, \mathcal{D})$) of the conditional response entropy, given f and x. Thus, we seek x for which the model is marginally most uncertain about r (high $H(r|x, \mathcal{D})$), but for which individual settings of f are confident (low $\langle H(r|f, x)\rangle_{p(f|x,\mathcal{D})}$).

The above utility function was introduced in the context of binary classification by Houlsby et al. [24]. We show that it can be well approximated for the Poisson observation model considered here, as a closed form expression of the mean and variance of $\lambda(x)$ ($\mu(x)$ & $\sigma(x)^2$, respectively;

Methods section 5.5.1). For a Poisson model, U increases monotonically with μ and σ (**Fig. 5b**). We observe that U is largest when either: (1) the model predictions are uncertain (large σ) ~~, so that the first term of Eqn~~ 7 ~~is large~~; or (2) the predicted firing rate, and thus the signal-to-noise ratio, is large (large μ) ~~, so that the second term of Eqn~~ 7 ~~is small.~~ (Note that, in contrast, in a standard GP model with constant noise $U(x)$ doesn't depend on the mean, μ.)

To work on-line, each step of the active learning algorithm must be executed quickly. In our work, the bottleneck was updating the hyperparameters of the GP, since both the approximate inference step (to update μ and σ) and utility function, U, could be computed quickly. To speed things up, therefore, we developed a 'reduced' GP model, for which most of the hyper-parameters were fixed, barring the gain and bias of the log-firing rate, A and λ_0 (see Eqn 4), as these parameters could be updated quickly without altering the shape of the kernel $K(x, x')$. The parameters determining the location of each neuron's RF were estimated using its responses to a flashed white noise stimulus, which was presented routinely at the start of each experiment. All other hyperparameters were kept the same for all neurons.

We used the data-set described above, comprising retinal responses to natural images, to show how our approach could work in principle. To mimic a closed-loop experiment, we initially only trained the model using a small subset of retinal responses to 50 natural images, chosen at random. We then revealed data points one at a time, either at random, or using the closed-loop algorithm described above.

We plotted learned LSTAs for one cell (the same cell shown in Fig. 1b-c) after 150 stimulus presentations (**Fig. 5c**), with stimulus-response pairs selected using the closed loop algorithm (middle), or at random (right). The learned LSTA for this cell converged much more quickly to those inferred after 3160 trials (left) when we used the closed loop algorithm, versus choosing responses at random (right). To quantify the performance of the closed-loop algorithm, we measured the log-likelihood of the learned model, in predicting responses to a 'test' data-set (not used for training). Note that, while the closed-loop was performed using the 'reduced' GP model (with fixed hyperparameters) as described above, the plotted log-likelihood, used for evaluation, was computed using the full GP model, with all hyperparameters learned using the available stimulus response pairs at each step. Despite considerable variability across runs, the closed-loop algorithm resulted in a consistently higher log-likelihood, compared to choosing stimuli randomly (**Fig. 5d**). This was the case across the population of recorded cells, with significantly higher log-likelihood for the closed-loop vs random stimulus selection, evaluated on one run of 150 trials for each cell (**Fig. 5e**).

3.6 Actively distinguishing between different models

Often, the goal is not to learn the response function, f, but rather to test whether a specific scientific hypothesis, h, is true ($h = 1$) or false ($h = 0$). For example, we might enquire whether retinal responses are better explained by a single layer or a two-layer neural network (i.e. an LN or LNLN model). Alternatively, we might ask whether a recorded retinal ganglion cell belongs to a given known cell-type (with known stimulus-response properties) [31].

We propose a closed-loop approach to address this problem using limited data. We adapt the framework described above by learning two different GP models in parallel, one for each experimental hypothesis (**Fig. 6a**). We then use these models to compute a utility function, $U(x)$, which quantifies how much information the response to a given stimulus is expected to convey about the scientific hypotheses, h. Finally, we choose the stimulus that maximises $U(x)$ to present on the next trial.

For the utility function, we choose the Shannon mutual information between the response, r, and hypothesis, h, conditioned on the stimulus x, and data-set, $\mathcal{D}$:

$$U(x) = I(r, h|x, \mathcal{D}). \tag{8}$$

Expanding, we have:

$$U(x) = H(r|x, \mathcal{D}) - \langle H(r|x, h, \mathcal{D}) \rangle_{p(h|\mathcal{D})}, \tag{9}$$

where the first term denotes the conditional response entropy, given x and $\mathcal{D}$, and the second term is the expected response entropy (over $p(h|\mathcal{D})$), given hypothesis, h, and data x and data-set $\mathcal{D}$.

We showed that $U(x)$ can be well approximated for the Poisson observation model considered here, as a closed form expression of the mean and variance of $\lambda(x)$, given $h = 0$ and 1, ($\mu_1(x)$, $\mu_2(x)$, $\sigma_1(x)^2$ & $\sigma_2(x)^2$; Methods section 5.5.2). $U(x)$ is highest when both models give different predictions about the mean firing rate, (i.e. $\mu_1 \neq \mu_2$; **Fig. 6b**), but when the uncertainty associated with both models is small (i.e. σ_1 & σ_2 are small; **Fig. 6c**). Note that this contrasts with the utility function described in the previous section, where U increased monotonically with σ (Fig. 5b).

To test our approach, we considered the case where the two hypothesis, to be tested are: ($h = 0$) neural responses are quasi-linear (i.e. they can be described by a GP with linear kernel, followed by static non-linearity); ($h = 1$) neural responses are non-linear (i.e. they can be described by a GP with arc-cosine kernel, followed by static non-linearity). We created a synthetic data-set, generated using a GP model (fitted to a neuron in the real data-set) with either a linear ($h = 0$) or non-linear kernel ($h = 1$) (**Fig. 6d-e**, left). We then used our GP framework to infer the probability that $h = 1$, after observing a small number of trials from the synthetic data-set. As intended (**Fig. 6d-e**, right), we were significantly faster at correctly identifying whether the synthetic data was generated using a linear or a non-linear model when stimuli were chosen using the closed-loop algorithm, and not at random.

Finally, we evaluated the performance of the closed-loop algorithm on the real data-set. As there was no available ground truth, we defined the 'true' model class for each neuron to be the preferred model (i.e. linear or non-linear) selected by the reduced GP algorithm trained on the full data-set of 3160 trials. We then plotted how quickly the algorithm converged to 'correctly' identify the model class of each cell, when stimuli were selected either using the closed-loop algorithm or at random (**Fig. 6f**). As hoped, the algorithm was significantly quicker at correctly classifying the model class when stimuli were selected using the closed-loop algorithm, and not at random.

4 Discussion

Previous work using GP models to fit neural responses was restricted to low-dimensional, artificial stimuli, such as moving oriented gratings [11, 13, 14]. This is likely because most commonly used stationary kernels (e.g. the squared-exponential or Matern kernel) rely on simple assumptions about the response function (e.g. that it is smooth), which don't scale well to high-dimensional inputs such as natural images [15, 16]. To scale to higher input dimensions, additional assumptions are required, such as the fact that neural RFs are spatially local and smooth [17, 12, 25, 26], which act to reduce the effective dimensionality of the input. As we showed (**Fig 2-3**), this works well for modelling retinal neurons, and is also expected to work for other low-level visual areas such as V1 and V2, where receptive fields of neurons cover restricted areas of the visual scene. In addition, our approach could be adapted to other sensory areas, such as the primary auditory cortex, where neurons respond selectively to a small range of spectral frequencies.

To demonstrate the advantages of GP processes to model neural data, we started by considering neural responses to a static image. In the future it would be interesting to extend our model to neural responses to dynamic stimuli, such as natural movies [27]. This would of course increase the dimensionality of the input, and thus the difficulty of inference. Nonetheless, additional prior assumptions, such as local and smooth temporal response functions [26], could be used to keep inference tractable.

We fitted the GP model on each neuron separately, in contrast to the CNN model, whose first layer was trained using data from the entire recorded neural population [5]. Despite this, the GP model outperformed the CNN model (**Fig 2**), with strongest differences observed when both models were trained on very little data (**Fig. 3**). This is likely because Bayesian models, such as GPs, adjust their complexity to avoid over-fitting on limited data. In the future, the performance of the GP model could likely be increased further by fitting it jointly on multiple recorded neurons, as we did for the CNN. While a GP model doesn't contain multiple layers, one way to do this would be to assume that the responses of different neurons are obtained by linearly combining a small number of latent GPs [28, 29, 30] (while performing a spatial translation on the input, to take into account the different RF locations). For the retina, these latent GPs could have a biological relevance, as corresponding to different cell-types [4, 31]. Standard methods for classifying retinal cells types functionally are based on the response to simple stimuli (full field luminance amplitude or frequency variations, moving bars). However, the extension of our methods using latent GPs could classify the cells using more relevant stimuli (natural images), with the addition of providing a model for each cell [32].

Our GP model is equivalent to a two layer neural network (i.e. an LNLN model) with infinitely many units in the middle layer (**Fig 1d**). There are several advantages in taking this infinite limit. First, inference can be expressed as a convex optimisation problem. Thus, given the 7 hyperparameters (determining the position, size & smoothness of the neuron's RF), we always obtain the same posterior distribution over the response curves, $f(x)$. Second, as stated earlier, the resulting Bayesian model avoids over-fitting, and as a result, the GP out-performs the LNLN model when trained on limited data (**Fig 3**).

Nonetheless, in some cases it may be desirable to 'look inside' the black-box model, to understand how neural responses are generated by different activations of the hidden units in the middle layer. To do this, we showed how one could use our fitted GP as a starting point, to construct an equivalent two layer neural network model, which could be used to gain further insight into the underlying biological network (**Fig 4**).

We found that best results were obtained with a non-stationary arc-cosine kernel, proposed by Cho & Saul [21] (modified to include a prior for local & smooth RFs, without which the model performed poorly). One reason could be that this kernel corresponds to a 2-layer neural network, which roughly approximates the retinal architecture (where there is an intermediate bipoloar cell layer, before the recorded ganglion cells). In their original paper [21], Cho & Saul, showed how their kernel could be simply extended to incorporate more neural layers. While we found this did not significantly improve performance for the retinal data-set, it is possible that this could help for higher-level visual areas beyond the retina, such as V1 and V2. Indeed, to model these higher level areas, future work could look at extending the kernel used here, to include aspects of convolutional [33] and/or deep [34] neural networks.

In recent closed-loop experiments using deep neural networks, stimuli were chosen so as to elicit a maximal response in recorded neurons [35, 36, 37]. In contrast, our approach looks for stimuli that are most informative about the entire response curve, $f(x)$, and not just its maximum. This can be seen in the acquisition function, $U(x)$, used to choose new stimuli (**Fig. 5b**), which favoured stimuli for which there was high uncertainty (large σ), as well as a high predicted firing rate (large μ). Note that it would not be possible to do this for standard deep neural networks, as they do not allow us to compute the uncertainty in their predictions (σ). Moreover, our approach can be extended to choose stimuli, in closed-loop, that allow us to best address the specific scientific question at hand [38] (**Fig. 6**). To illustrate this, we considered a simple question: are retinal neurons best described with an LN or and LNLN model? Future work could consider a range of different questions. For example, we could choose stimuli such that we can best ascertain which groups of neurons in the retina belong to the same cell-type [31, 32]. Alternatively, in higher level visual areas we could select stimuli so as to best test the effects of top-down processes such as visual attention and expectations.

5 Methods

5.1 Gaussian process model

We construct a two layer linear-nonlinear (LNLN) model of neural responses. The spike count, r is assumed to be drawn from a Poisson distribution with mean $f(x) = e^{A\lambda(x)+\lambda_0}$, where x is an n_x-d stimulus vector, A and λ_0 are constant gain and bias terms, and $\lambda(x)$, is given by a weighted sum of J units:

$$\lambda(x) = \frac{1}{\sqrt{J}} \sum_{j=1}^{J} v_j \phi\left(w_j^T x + b_j\right) \tag{10}$$

where ϕ is a non-linear basis function, v_j are scalar weights, w_j is an n_x-d vector of linear weights, and b_j is a scalar bias term.

We assume *a priori* that parameters, w_j, v_j and b_j, are sampled from independent and identical distributions for different j. In this case, under fairly general conditions, in the limit $J \to \infty$, we can use the central limit theorem to see that $\lambda(x)$ is a Gaussian process, with mean, $\mu_0 \equiv \langle\lambda(x)\rangle$, and covariance, $K(x, x') \equiv \mathrm{cov}(\lambda(x), \lambda(x'))$.

With no loss of generality, we assume that weights in the second-layer are sampled from a standard normal distribution, $v_j \sim \mathcal{N}(0, 1)$. We assume a multivariate Gaussian prior over input weights in the first layer, with zero mean and covariance C, while the bias term, b_j is drawn from a Gaussian with zero mean and variance σ_0^2. It is straightforward to show that in the limit $J \to \infty$, the prior mean of λ is zero, while its covariance is given by:

$$K(x, x') = \frac{1}{(2\pi)^{\frac{n_x+1}{2}} \sqrt{|C|\sigma_0^2}} \int_{w,b} e^{-\frac{1}{2}w^T C^{-1} w - \frac{1}{2\sigma_0^2} b^2} \phi\left(w^T x + b\right) \phi\left(w^T x' + b\right) dw db \tag{11}$$

The above kernel depends on the form of the non-linearity, ϕ. Here we consider a rectified-nonlinearity, $\phi(x) = \max(x, 0)$, which gives the following kernel:

$$K(x, x') = \frac{1}{2\pi} \sqrt{x'^T C x' + \sigma_0^2} \sqrt{x^T C x + \sigma_0^2} \left(\sin\psi + (\pi - \psi)\cos\psi\right) \tag{12}$$

where,

$$\psi = \cos^{-1}\left(\frac{x^T C x' + \sigma_0^2}{\sqrt{x^T C x + \sigma_0^2}\sqrt{x'^T C x' + \sigma_0^2}}\right). \tag{13}$$

In the special case where $C = I$ and $\sigma_0 = 0$ this proposed kernel becomes equivalent to the arc-cosine kernel proposed by [21].

As a control, we also considered the Gaussian kernel,

$$K(x, x') = \alpha_0 e^{-\frac{1}{2}(x-x')^T C (x-x')} \tag{14}$$

where α_0 is a positive constant, as well as a linear kernel:

$$K(x, x') = x^T C x', \tag{15}$$

and a quadratic kernel:

$$K(x, x') = \left(x^T C x' + \sigma_0^2\right)^2. \tag{16}$$

5.2 Spatially local and smooth receptive fields

Extending the work of Park & Pillow [17], we assume a prior distribution over the weights in the first layer, $\mathcal{N}(w|0, C)$, to enforce spatially local, smooth, receptive fields. To do this, we associate each element, w_i, with a spatial location, ξ_i. We then construct a covariance matrix with elements:

$$C_{ij} = \alpha_i^{\mathrm{local}} \alpha_j^{\mathrm{local}} C_{ij}^{\mathrm{smooth},} \tag{17}$$

where α^{local} enforces w to be localised around a spatial location ξ_0, and to be zero everywhere else. It is given by,

$$\alpha_i^{\mathrm{local}} = e^{-\frac{\|\xi_i - \xi_0\|}{4\beta^2}}. \tag{18}$$

$C^{\mathrm{smooth},}$ enforces w to be spatially smooth, and is given by,

$$C_{i,j}^{\mathrm{smooth}} = e^{-\frac{\|\xi_i - \xi_j\|^2}{2\rho^2}}. \tag{19}$$

The covariance function, C contains 4 hyperparameters. β and ρ determine how spatially localized and smooth the receptive field is, respectively. A 2-d coordinate ξ_0 determines the spatial location of the receptive field.

5.3 Inference

We seek to learn the posterior distribution, $p(\lambda(x)|\mathcal{D})$, where $\lambda(x)$ is the log-firing rate, and $\mathcal{D} = (x_1, x_2, \ldots, x_N, r_1, r_2, \ldots r_N)$, is the observed data-set. Unfortunately, exact inference with a GP model quickly becomes infeasible as the number of data points, N, becomes large. To get around this, we define a variational distribution, $q\left(\tilde{\lambda}\right)$, over a set of $n \ll N$ 'inducing-points', $\tilde{\lambda} \equiv \lambda(\tilde{x}_1), \ldots, \lambda(\tilde{x}_n)$. Predictions are then made by computing $p(\lambda(x)|\mathcal{D}) \approx \left\langle p\left(\lambda(x)|\tilde{\lambda}\right)\right\rangle_{q(\tilde{\lambda})}$. Doing this reduces the cost of inference from $O\left(N^3\right)$ to $O\left(n^2 N\right)$. In the experiments described here we chose $n = 250$, since little improvement was observed when n was increased further. For the closed-loop experiments we set $n = N$.

We choose q to be a multivariate Gaussian with mean m and covariance V, $q\left(\tilde{\lambda}\right) = \mathcal{N}\left(\tilde{\lambda}\middle|m, V\right)$. We optimise m and V by maximising the following lower bound on the log-likelihood:

$$\log p(\mathcal{D}) \geq -D_{KL}\left(q\left(\tilde{\lambda}\right)\middle\|p\left(\tilde{\lambda}\right)\right) + \sum_{i=1}^{N} \left\langle \log p_{\mathrm{poiss.}}\left(r_i|e^{\lambda_i + \lambda_0}\right)\right\rangle_{p\left(\lambda_i|\tilde{\lambda}\right)q\left(\tilde{\lambda}\right)} \tag{20}$$

where $D_{KL}(p\|q)$ is the Kullback-Leibler divergence between p and q, $p_{\mathrm{poiss.}}$ is a Poisson distribution, and we have defined $\lambda_i \equiv \lambda(x_i)$.

The KL-divergence divergence term takes the following form:

$$D_{KL}\left(q\left(\tilde{\lambda}\right)\middle\| p\left(\tilde{\lambda}\right)\right) = \frac{1}{2}\log\frac{|K|}{|V|} + \frac{1}{2}\mathrm{Tr}\left(VK^{-1}\right) + \frac{1}{2}m^T K^{-1} m + \mathrm{const.} \tag{21}$$

where K is an $n \times n$ kernel-gram matrix with elements, $K_{ij} = K\left(\tilde{x}_i, \tilde{x}_j\right)$.

The second term is the average log-likelihood of the data,

$$\begin{aligned} \left\langle \log \mathrm{p_{poiss}}\left(r_i \middle| e^{\lambda_i+\lambda_0}\right)\right\rangle &= r_i\left(\langle\lambda_i\rangle + \lambda_0\right) - \left\langle e^{\lambda_i+\lambda_0}\right\rangle - \log r_i! \\ &= r_i\left(\langle\lambda_i\rangle + \lambda_0\right) - e^{\langle\lambda_i\rangle + \frac{1}{2}\mathrm{var}(\lambda_i)+\lambda_0} - \log r_i!, \end{aligned} \tag{22}$$

where,

$$\langle\lambda_i\rangle = k_i^T K^{-1} m \tag{23}$$

$$\mathrm{var}\left(\lambda_i\right) = k_{ii} + k_i^T K^{-1}\left(V - K\right) K^{-1} k_i. \tag{24}$$

Here, k_{ii} is a scalar given by $K\left(x_i, x_i\right)$, while k_i is an n-d vector, with j^{th} element $K\left(x_j, \tilde{x}_i\right)$.

The lower bound on the log-likelihood, in Eqn 20, is concave with respect to m and V. We can efficiently maximise this lower bound by performing the following Newton updates:

$$V \leftarrow K\left(K+G\right)^{-1} K \tag{25}$$

$$m \leftarrow K\left(K+G\right)^{-1}\left(g + GK^{-1}m\right) \tag{26}$$

where:

$$g = \sum_{i=1}^{N} k_i\left(r_i - \left\langle e^{\lambda_i+\lambda_0}\right\rangle\right) \tag{27}$$

$$G = \sum_{i=1}^{N} k_i k_i^T \left\langle e^{\lambda_i+\lambda_0}\right\rangle \tag{28}$$

The hyperparameters determining the kernel matrix, K, and bias term, λ_0, are then updated using gradient-descent to maximise the lower-bound on the log-likelihood, for fixed m and V. Optimisation proceeds via an EM algorithm, by alternately updating $q\left(\tilde{\lambda}\right)$ for fixed θ and λ_0 (the E-step), then updating θ and λ_0 for fixed $q\left(\tilde{\lambda}\right)$ (the M-step).

After learning, the mean and variance of the predicted firing rate are given by:

$$\langle f\left(x\right)\rangle = e^{\langle\lambda(x)\rangle + \frac{1}{2}\mathrm{var}(\lambda(x))+\lambda_0} \tag{29}$$

$$\mathrm{var}\left(f\left(x\right)\right) = \langle f\left(x\right)\rangle^2\left(e^{\mathrm{var}(\lambda(x))} - 1\right), \tag{30}$$

where we recall that $\langle\lambda_i\rangle$ & $\mathrm{var}\left(\lambda_i\right)$ are given by Eqns 23 & 24, respectively.

5.4 Equivalent neural network

In order to get insight into the subunits and activation functions of the GP model obtained, we sought an approximation with a finite number of J units in the middle layer. To do this, we expressed the log firing rate, $\lambda\left(x\right)$, according to Eqn 10, with weights and bias sampled from $w_j \sim \mathcal{N}\left(w|0, C\right)$ and $b_j \sim \mathcal{N}\left(b_j|0, \sigma_0^2\right)$, where C and σ_0^2, where obtained from the fitted infinite GP model. We then performed variational inference (see Methods subsection above 5.3) to infer the posterior distribution over weights in the second layer, v, so as to maximise a lower bound on the log-likelihood. With 2000 units (see Fig. 4), the predicted firing rates closely approximated the full GP model (with $J = \infty$). Note, however, that the full GP model was necessary to obtain hyperparameters determining C and b, without which the finite approximation would perform poorly.

We next sought a 'sparse' model, with few hidden units in the middle layer. To do this we assumed a prior distribution over weights in the second layer,

$$p(v|\alpha) = \prod_j \mathcal{N}\left(v_j|0, \alpha_j^2\right) \tag{31}$$

where α_j is a hyper-parameter determining the prior variance of v_j. To encourage sparsity, we assumed a hyper-prior penalising large values of α_j:

$$p(\alpha_j) = \gamma e^{-\gamma|\alpha_j|} \tag{32}$$

where γ is a free parameter. We then learned α_j via MAP inference, by maximising a lower bound on the log-likelihood (see Methods subsection above 5.3) plus the log-prior, $-\gamma|\alpha_j|$. With large γ, the model learned to set $\alpha_j = 0$ for many units, thereby effectively removing these units from the model. Varying γ allowed us to vary the sparsity of the model.

5.5 Optimal experimental design

5.5.1 Active learning of the response function

We seek to choose a new stimulus so as to maximise the mutual information between the response, r, and the firing rate, f, given data, $\mathcal{D}$ and a new stimulus, x:

$$I(r; f|x, \mathcal{D}) = H(r|x, \mathcal{D}) - \langle H(r|f)\rangle_{p(f|x,\mathcal{D})} \tag{33}$$

The first, response entropy, term is given by:

$$H(r|x, \mathcal{D}) = -\sum_{r=0}^{\infty} p(r|x, D) \log p(r|x, D) \tag{34}$$

where,

$$\begin{aligned} p(r|x, \mathcal{D}) &= \int_{-\infty}^{\infty} \mathrm{Poiss}(r|\lambda)\, p(\lambda|x, \mathcal{D})\, d\lambda, && (35)\\ &= \frac{1}{r!}\int_{-\infty}^{\infty} e^{r\lambda - e^{\lambda}} \mathcal{N}\left(\lambda|\mu(x), \sigma(x)^2\right) d\lambda, && (36) \end{aligned}$$

and $\mu(x)$ and $\sigma(x)^2$ are defined here as the posterior mean and variance of $\lambda(x)$ given previous observations $\mathcal{D}$. From here-in we will drop the dependence on x for notational simplicity. While the above integral is intractable, we can closely approximate it using Laplace approximation:

$$\log p(r|x, \mathcal{D}) \approx \bar{\lambda}_r r - e^{\bar{\lambda}_r} - \frac{1}{2\sigma^2}\left(\bar{\lambda}_r - \mu\right)^2 - \frac{1}{2}\log\left(\sigma^2 e^{\bar{\lambda}_r} + 1\right) - \log r! \tag{37}$$

where

$$\bar{\lambda}_r = \arg\max_{\lambda}\left[\lambda r - e^{\lambda} - \frac{1}{2\sigma^2}(\lambda - \mu)^2\right] = r\sigma^2 + \mu - W_0\left(\sigma^2 e^{r\sigma^2 + \mu}\right). \tag{38}$$

where W_0 is the zeroth branch of the Lambert W-function. We then substitute this approximation into Eqn 34 to find the entropy. Note that while in theory, the sum in Eqn 34 goes to infinity, in practice we can do fairly well with a truncated version of this sum, so long as the mean firing rates are small (which is the case for our data-sets).

Next, we need to find the conditional noise entropy, given by:

$$\langle H(r|f)\rangle_{p(f|x,\mathcal{D})} = \langle H(r|\lambda)\rangle_{p(\lambda|x,\mathcal{D})} = \int_{-\infty}^{\infty} p(\lambda|x,\mathcal{D})\, H(r|\lambda)\, d\lambda \tag{39}$$

$$= \int_{-\infty}^{\infty} p(\lambda|x,\mathcal{D}) \left(e^{\lambda}(1-\lambda) + \sum_{r=0}^{\infty} p(r|\lambda) \log r! \right) \tag{40}$$

$$= e^{\mu+\frac{1}{2}\sigma^2}(\mu + \sigma^2 - 1) + \sum_{r=0}^{\infty} p(r|x,\mathcal{D}) \log r! \tag{41}$$

The last term can be approximated using Eqn 37. Again, the summation can be truncated with little loss in accuracy if firing rates are small.

5.5.2 Active learning of the model class

Here, we seek to maximise the mutual information between the response, r, and the model class, h, given data, $\mathcal{D}$ and new stimulus, x:

$$I(r;h|x,\mathcal{D}) = H(r|x,\mathcal{D}) - \sum_{h=0,1} \langle H(r|x,\mathcal{D},h)\rangle_{p(h|D)} \tag{42}$$

The first term is the entropy of the response if we don't know the model class. To obtain this, we need to compute the posterior distribution of responses, if the model class is unknown, given by:

$$p(r|x,\mathcal{D}) = \sum_{h=0,1} p(r|x,h,\mathcal{D})\, p(h|\mathcal{D}). \tag{43}$$

$p(r|x,h,\mathcal{D})$ is approximated using the Laplace approximation described in the previous section (Eqn 37). $p(h|\mathcal{D})$ is given by:

$$p(h|\mathcal{D}) = \frac{p(\mathcal{D}|h)}{\sum_{h'=0,1} p(\mathcal{D}|h')} \tag{44}$$

where $p(\mathcal{D}|h)$ is approximated using our variational lower bound (Eqn 20), which we compute at each stage of inference.

5.6 Experimental details

We used a data-set consisting of the responses of 41 retinal ganglion cells of a mouse to various visual stimuli, reported in Goldin et al., 2022 [5]. In this experiment, perturbed and unperturbed natural images, together with pure white noise stimuli, were presented to the retina, while recording extra-cellularly retinal ganglion cells with a multi-electrode-array. There were 3190 unperturbed natural images, 30 of which were repeated 30 times to form the test set, and 8000 perturbed images (i.e. natural image plus white noise perturbation). Perturbed and unperturbed natural images were grey-scale (8-bits) and 864x864 pixels, which we down-sampled to 108×108 for modelling.

Characterization of the receptive field (RF) size and position of cells was done with white noise random checkerboard stimuli at 30 Hz. Receptive fields were estimated using a classical 'Spike

Triggered Average' (STA), and were fitted with a two dimensional Gaussian function to estimate its parameters.

A 'local spike-triggered-average' (LSTA) for each cell (**Fig. 2b** & **d**) was calculated for each reference image by counting the spikes evoked between 30 and 350 ms after each perturbed natural image presentation, and averaging the perturbation patterns weighted by their corresponding spike counts.

Spike count responses r evoked by a cell to unperturbed natural images, and used for model fitting were also obtained by counting the number of spikes between 30 and 350 ms after each natural image presentation.

5.7 CNN model

The CNN model is described in Goldin et al. 2022 [5]. It consists of a two layer neural network. The first layer was convolutional, with four two-dimensional kernels, learned from data, and common to all the recorded cells. The second layer consisted of one filter for each cell, followed by a nonlinearity. To reduce dimensionality, we factorized the weights of each filter into a two-dimensional spatial mask and a feature weight vector. In the first layer, we used Laplacian regularization on the convolutional kernels, and on the second layer, we used L_1 regularisation on the spatial weights and on the feature weights. Regularisation hyperparameters were selected using cross-validation. This was done separately when the model was trained on a reduced data-set (**Fig 3**).

5.8 LNLN model

As for the CNN, the LNLN model consists of a two layer neural network, where each layer linearly combines its inputs before applying a (soft-plus) non-linearity (with bias and gain learned from data). In contrast to the CNN, there are no shared weights between neurons: each neuron was trained separately. The first layer consists of two, 7×7 convolutional kernels, $k_{i,j}^{\text{on}}$ and $k_{i,j}^{\text{off}}$ (where i and j index space) that compute two convolutions of the image, followed by a soft-plus nonlinearity. The $k_{i,j}^{\text{on}}$ and $k_{i,j}^{\text{off}}$ were chosen to be 2D Gaussian spatial profiles of opposite polarity but sharing the same spatial scale (following previous results [5]), and were kept constant during training. The second layer consists of linear filter weights $w_{i,j}^{\text{on}}$ and $w_{i,j}^{\text{off}}$, followed by a soft-plus non-linearity, which is applied to the output of the first layer. We assume a Poisson observation model for training. We used a L_1 and a Laplacian regularization to train the linear filters of the second layer, whose hyperparameters were selected by cross-validation.

References

[1] Pillow, J. W., Shlens, J., Paninski, L., Sher, A., Litke, A. M., Chichilnisky, E. J., & Simoncelli, E. P. (2008). Spatio-temporal correlations and visual signalling in a complete neuronal population. Nature, 454(7207), 995-999.

[2] Bartsch, F., Cumming, B. G., & Butts, D. A. (2022). Model-based characterization of the selectivity of neurons in primary visual cortex. Journal of Neurophysiology, 128(2), 350-363.

[3] Willeke, K. F., Fahey, P. G., Bashiri, M., Pede, L., Burg, M. F., Blessing, C., ... & Sinz, F. H. (2022). The Sensorium competition on predicting large-scale mouse primary visual cortex activity. arXiv preprint arXiv:2206.08666.

[4] Klindt, D., Ecker, A. S., Euler, T., & Bethge, M. (2017). Neural system identification for large populations separating "what" and "where". Advances in Neural Information Processing Systems, 30.

[5] Goldin, M.A., Lefebvre, B., Virgili, S., Pham Van Cang, M.K., Ecker, A., Mora, T., Ferrari, U., & Marre, O. (2022). Context-dependent selectivity to natural images in the retina. Nature Communications, 13(1), pp.1-12.

[6] Maheswaranathan, N., Kastner, D. B., Baccus, S. A., & Ganguli, S. (2018). Inferring hidden structure in multilayered neural circuits. PLoS computational biology, 14(8), e1006291.

[7] Kriegeskorte, N. (2015). Deep neural networks: a new framework for modelling biological vision and brain information processing. Annu. Rev. Vis. Sci 1 (2015): 417-46.

[8] Yamins, D. L., & DiCarlo, J. J. (2016). Using goal-driven deep learning models to understand sensory cortex. Nature neuroscience, 19(3), 356-365. Chicago

[9] Tanaka, H., Nayebi, A., Maheswaranathan, N., McIntosh, L., Baccus, S., & Ganguli, S. (2019). From deep learning to mechanistic understanding in neuroscience: the structure of retinal prediction. Advances in neural information processing systems, 32.

[10] Williams, C. K., & Rasmussen, C. E. (2006). Gaussian processes for machine learning (Vol. 2, No. 3, p. 4). Cambridge, MA: MIT press.

[11] Park, M., Horwitz, G., & Pillow, J. (2011). Active learning of neural response functions with Gaussian processes. Advances in neural information processing systems, 24.

[12] Park, M. & Pillow, J. W. (2012). Bayesian active learning with localized priors for fast receptive field characterization, Advances in Neural Information Processing Systems 25, 2357-2365

[13] Park, M., Weller, J. P., Horwitz, G. D., & Pillow, J. W. (2014). Bayesian active learning of neural firing rate maps with transformed gaussian process priors. Neural computation, 26(8), 1519-1541.

[14] Greenidge, C. D., Scholl, B., Yates, J. L., & Pillow, J. W. (2022). Efficient decoding of large-scale neural population responses with Gaussian-process multiclass regression. bioRxiv, 2021-08.

[15] Binois, M., & Wycoff, N. (2022). A survey on high-dimensional Gaussian process modeling with application to Bayesian optimization. ACM Transactions on Evolutionary Learning and Optimization, 2(2), 1-26.

[16] Delbridge, I., Bindel, D., & Wilson, A. G. (2020). Randomly projected additive Gaussian processes for regression. In International Conference on Machine Learning (pp. 2453-2463). PMLR.

[17] Park, M., & Pillow, J. W. (2011). Receptive field inference with localized priors. PLoS computational biology, 7(10), e1002219.

[18] MacKay, D. J. (2003). Information theory, inference and learning algorithms. Cambridge university press, Chapter 28, page 343

[19] Hensman, J., Fusi, N., & Lawrence, N. D. (2013). Gaussian processes for big data. arXiv preprint arXiv:1309.6835.

[20] Ancona, M., Ceolini, E., Öztireli, C., & Gross, M. (2019). Gradient-based attribution methods. Explainable AI: Interpreting, Explaining and Visualizing Deep Learning, 169-191.

[21] Cho, Y., & Saul, L. (2009). Kernel methods for deep learning. Advances in neural information processing systems, 22.

[22] Rajan, K., Marre, O., & Tkačik, G. (2013). Learning quadratic receptive fields from neural responses to natural stimuli. Neural computation, 25(7), 1661-1692.

[23] Keshishian M, Akbari H, Khalighinejad B, Herrero JL. Estimating and interpreting nonlinear receptive field of sensory neural responses with deep neural network models. 2020:1-24.

[24] Houlsby, N., Huszár, F., Ghahramani, Z., & Lengyel, M. (2011). Bayesian active learning for classification and preference learning. arXiv preprint arXiv:1112.5745.

[25] Huang, Z., Ran, Y., Oesterle, J., Euler, T., & Berens, P. (2021). Estimating smooth and sparse neural receptive fields with a flexible spline basis. arXiv preprint arXiv:2108.07537.

[26] Duncker, L., Ruda, K. M., Field, G. D., & Pillow, J. W. (2022). Scalable variational inference for low-rank spatio-temporal receptive fields. bioRxiv.

[27] Hoefling, L., Szatko, K. P., Behrens, C., Qiu, Y., Klindt, D. A., Jessen, Z., ... & Euler, T. (2022). A chromatic feature detector in the retina signals visual context changes. bioRxiv.

[28] Yu, B. M., Cunningham, J. P., Santhanam, G., Ryu, S., Shenoy, K. V., & Sahani, M. (2008). Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. Advances in neural information processing systems, 21.

[29] Rutten, V., Bernacchia, A., Sahani, M., & Hennequin, G. (2020). Non-reversible Gaussian processes for identifying latent dynamical structure in neural data. Advances in neural information processing systems, 33, 9622-9632.

[30] Wu, A., Roy, N.A., Keeley, S., & Pillow, J.W. (2017). Gaussian process based nonlinear latent structure discovery in multivariate spike train data. Advances in neural information processing systems, 30.

[31] Baden, T., Berens, P., Franke, K., Román Rosón, M., Bethge, M., & Euler, T. (2016). The functional diversity of retinal ganglion cells in the mouse. Nature, 529(7586), 345-350.

[32] Trapani, F., Spampinato, G., Yger, P., & Marre, O. (2022). Differences in non-linearities determine retinal cell types. bioRxiv.

[33] Van der Wilk, M., Rasmussen, C. E., & Hensman, J. (2017). Convolutional gaussian processes. Advances in Neural Information Processing Systems, 30.

[34] Blomqvist, K., Kaski, S., & Heinonen, M. (2020). Deep convolutional Gaussian processes. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 582-597). Springer, Cham.

[35] Cadena, Santiago A., George H. Denfield, Edgar Y. Walker, Leon A. Gatys, Andreas S. Tolias, Matthias Bethge, & Alexander S. Ecker. "Deep convolutional models improve predictions of macaque V1 responses to natural images." PLoS computational biology 15, no. 4 (2019): e1006897.

[36] Ponce, C. R., Xiao, W., Schade, P. F., Hartmann, T. S., Kreiman, G., & Livingstone, M. S. (2019). Evolving images for visual neurons using a deep generative network reveals coding principles and neuronal preferences. Cell, 177(4), 999-1009.

[37] Walker, E. Y., Sinz, F. H., Cobos, E., Muhammad, T., Froudarakis, E., Fahey, P. G., ... & Tolias, A. S. (2019). Inception loops discover what excites neurons most using deep predictive models. Nature neuroscience, 22(12), 2060-2065.

[38] DiMattina, C., & Zhang, K. (2011). Active data collection for efficient estimation and comparison of nonlinear neural models. Neural computation, 23(9), 2242-2288.