A REVIEW OF GAUSSIAN PROCESSES

We briefly review here the main ideas of Gaussian Processes for machine learning, see Stein (2012); Rasmussen et al. (2006) for more details. We start to explain how to use stochastic processes for Bayesian inference. We see the stochastic process as prior over functions $p(f)$ and as we are given samples $\boldsymbol{x} = (x_1, \ldots, x_N), \boldsymbol{y} = (y_1, \ldots, y_N), y_i \equiv f(x_i)$, we update our beliefs about the function by constructing the posterior via Bayes rule: $p(f|\boldsymbol{y}, \boldsymbol{x}) = p(\boldsymbol{y}|f, \boldsymbol{x})p(f)/p(\boldsymbol{y}|\boldsymbol{x})$. Here we need to specify the likelihood of the data with our model $p(\boldsymbol{y}|f, \boldsymbol{x}) = \prod_{i=1}^{N} p(y_i|f, x_i)$ and the denominator, called the evidence or marginal likelihood, follows: $p(\boldsymbol{y}|\boldsymbol{x}) = \mathbb{E}_{f\sim p(f)}[p(\boldsymbol{y}|f, \boldsymbol{x})]$. The power of this approach is that the value of the signal y at an unseen point x has an uncertainty which depends on our knowledge of its neighbourhood: $p(y|x, \boldsymbol{y}, \boldsymbol{x}) = \mathbb{E}_{f\sim p(f|\boldsymbol{y},\boldsymbol{x})}[p(y|f, x)]$ and allows us to reason probabilistically about the underlying signal.

A particular convenient class of random function is Gaussian processes (GPs) for which inference can be done exactly. A stochastic process can be presented in terms of the finite distributions of the random variables $\{f(x_i)\}_{i=1}^{M}$ at points $\{x_i\}_{i=1}^{M}$. For a GP these distributions are Gaussian and can be defined uniquely by specifying means and covariances, and so a GP is specified entirely by its mean function $\mu(x)$ and covariance kernel $k(x, x')$. We shall write $f \sim \mathcal{GP}(\mu, k)$. Let us assume a Gaussian likelihood model as well, i.e. $p(y_i|f, x_i) = \mathcal{N}(f(x_i), \sigma_i^2)$, where σ_n represents aleatoric uncertainty on the measurement. (For simplicity we take here the function to be scalar valued but the reasoning can be easily generalized.) Then properties of the Gaussian distribution (see (Rasmussen et al., 2006, Chap. 2) for a detailed derivation of the formulas) lead to the following posterior distributions after seeing data $\boldsymbol{y}, \boldsymbol{x}$: $p(f|\boldsymbol{y}, \boldsymbol{x}) = \mathcal{GP}(\mu_p, k_p)$, with

$$\mu_p(x) = \boldsymbol{k}(x)^T[K + S]^{-1}\boldsymbol{y}\,, \quad k_p(x, x') = k(x, x') - \boldsymbol{k}(x)^T[K + S]^{-1}\boldsymbol{k}(x')\,. \tag{12}$$

where $K_{ij} = k(x_i, x_j), k(x)_i = k(x, x_i)$ and $S = \mathrm{diag}(\sigma_i^2)$.

B FROM DISCRETE TO CONTINUOUS CONVOLUTIONAL LAYERS

We here show that the general formula

$$\mathcal{A} = \sum_k W_k \mathrm{e}^{\mathcal{D}_k} \tag{13}$$

with $\mathcal{D}_k$ a function of spatial derivatives, reduces in the case of discrete input space $\mathcal{X}$ to the usual convolution we encounter in deep learning.

For simplicity we shall assume a 1d grid as input space $\mathcal{X} = \{1, \ldots, N\}$. Let us start by recalling the form of the classical discrete convolution when $C_\ell = C_{\ell+1} = 1$. We define a convolutional layer as a linear map that commutes with the translation operator. To make the symmetry exact, we need assume periodic boundaries. Then in the standard basis of $\mathbb{R}^N$, $\{e_i\}_{i=1}^{N}$ of vectors localized at site i, the translation operator τ acts as $\tau e_i = e_{i+1 \mod N}$. An $N \times N$ matrix B is translation invariant iff $\tau B = B\tau$. Since τ is diagonal in Fourier space, the most general solution is $B = F\mathrm{diag}(\hat{\boldsymbol{b}})F^{-1}$, where $F_{jk} = \mathrm{e}^{\frac{2\pi i}{N}jk}$ is the discrete Fourier transform. Such matrices are called circulant and can be written alternatively as $B = \sum_{i=0}^{N-1} b_{N-i}\tau^i$, $\boldsymbol{b} = F\hat{\boldsymbol{b}}$. Explicitly:

$$B = \begin{pmatrix} b_0 & b_1 & \ldots & b_{N-2} & b_{N-1} \\ b_{N-1} & b_0 & b_1 & & b_{N-2} \\ \vdots & b_{N-1} & b_0 & \ddots & \vdots \\ b_2 & & \ddots & \ddots & b_1 \\ b_1 & b_2 & \ldots & b_{N-1} & b_0 \end{pmatrix}. \tag{14}$$

This shows that the most general convolutional layer is a circulant matrix. E.g. if $b_i = 0$ unless $i = 0, 1, N-1$, B coincides with the matrix representing a periodic convolution of filter size 3. The matrix B is invertible as long as $\hat{b}_k \neq 0$ for all k. In a convolutional network the parameters b_i are random variables and the measure of the set where B is not invertible is zero. Thus the role of $\mathrm{e}^{\mathcal{D}}$ is replaced in the discrete case by the B.

The discrete analog of $\mathcal{A}$ is then:

$$A = \sum_i W_i \otimes B_i\,. \tag{15}$$

Introducing the unit matrices $E_{\alpha,\beta}$ which have 1 at the row α and column β and 0 otherwise, we can rewrite it as:

$$A = \sum_{j,\alpha,\beta} E_{\alpha,\beta} \otimes \tau^j W_j^{\alpha,\beta}\,, \quad W_j^{\alpha,\beta} = \sum_i W_i^{\alpha,\beta} b_{i,N-j}\,. \tag{16}$$

Since $E_{\alpha,\beta} \otimes \tau^j$ is a linear basis of the space of convolutional layers, we see that equation 13 indeed reduces to the usual one when discretizing the input domain and is a principled generalization to the continuous domain.

C GREENS FUNCTION

Given the operator $\mathcal{D} = \beta^\top \nabla + \frac{1}{2}\nabla^\top \Sigma \nabla$ we can compute the action of $\mathrm{e}^{t\mathcal{D}}$ in terms of convolutions. Using the d dimensional Fourier transforms $\mathcal{F}[h](k) = (2\pi)^{-\frac{d}{2}} \int h(x) e^{-ik^\top x} dx$ and $\mathcal{F}^{-1} = \mathcal{F}^\dagger$, we can rewrite the derivative operator $\mathcal{D}$ in terms of elementwise multiplication in the Fourier domain, which diagonalizes $\mathcal{D}$. Since $\nabla = \mathcal{F}^{-1}(ik)\mathcal{F}$,

$$\mathcal{D} = \mathcal{F}^{-1}(i\beta^\top k - \tfrac{1}{2}k^\top \Sigma k)\mathcal{F}. \tag{17}$$

Using the series definition $\mathrm{e}^{t\mathcal{D}} = \sum_{n=0}^{\infty} (t\mathcal{D})^n / n!$, we have:

$$e^{t\mathcal{D}} = \mathcal{F}^{-1} e^{t(i\beta^\top k - \frac{1}{2}k^\top \Sigma k)} \mathcal{F}. \tag{18}$$

Applying this operator to a test function $h(x)$ yields

$$\mathrm{e}^{t\mathcal{D}} h = \mathcal{F}^{-1}[e^{t(i\beta^\top k - \frac{1}{2}k^\top \Sigma k)} \mathcal{F}[h](k)] = \mathcal{F}^{-1}[\mathcal{F}[G_t] \cdot \mathcal{F}[h]] = G_t * h, \tag{19}$$

where the final step follows from the Fourier convolution theorem, and we define the function $G_t = \mathcal{F}^{-1}[\mathrm{e}^{t(i\beta^\top k - \frac{1}{2}k^\top \Sigma k)}]$. Directly applying the Fourier integral yields a Gaussian integral

$$G_t(x) = (2\pi)^{-\frac{d}{2}} \int \mathrm{e}^{ik^\top (x+t\beta) - \frac{1}{2}k^\top t\Sigma k} dk = \mathrm{e}^{-\frac{1}{2}(x+t\beta)^\top (t\Sigma)^{-1}(x+t\beta)} \det(2\pi t\Sigma)^{-1/2}. \tag{20}$$

This function $G_t(x) = \mathcal{N}(x; -t\beta, t\Sigma)$ is nothing but a multivariate heat kernel, the Greens function (also known as the fundamental solution or time propagator) for the diffusion equation $\partial_t G_t(x - x') = \mathcal{D} G_t(x - x')$, and indeed $\lim_{t \to 0} G_t(x - x') = \delta(x - x')$.

D INTEGRAL POOLING

The integral pooling operator $\mathcal{P}[f] = \int_{\mathbb{R}^d} f(x) dx$ can be applied to the Gaussian process just like any other linear operator. Given $f^{(L)} \sim \mathcal{GP}(\mu, k)$, we have that

$$\mathcal{P} f^{(L)} \sim \mathcal{GP}(\mathcal{P}\mu, \mathcal{P}k\mathcal{P}') = \mathcal{N}(\mathcal{P}\mu, \mathcal{P}k\mathcal{P}'). \tag{21}$$

Again, computing the mean $\mu_P = \mathcal{P}\mu$ and covariance matrix $\Sigma_P = \mathcal{P}k\mathcal{P}'$ we need just to be able to apply $\mathcal{P}$ to the RBF kernel.

$$\mathcal{P}k_{\mathrm{RBF}}(x') = \int_{\mathbb{R}^d} k_{\mathrm{RBF}}(x, x') dx = a \tag{22}$$

$$\mathcal{P}k_{\mathrm{RBF}}\mathcal{P}' = \int_{\mathbb{R}^d \times \mathbb{R}^d} k_{\mathrm{RBF}}(x, x') dx dx' = \infty \tag{23}$$

For many applications such as image classification using the mean logit value, we require only the predictive mean, so an unbounded covariance matrix Σ_P is acceptable. We use this form for all of our experiments.

However for some applications an output uncertainty can be useful, so we also provide a variant that integrates over a finite region $[0,1]^d$, $\mathcal{P}f = \int_{[0,1]^d} f(x)dx$.

$$\mathcal{P}k_{\mathrm{RBF}}(x') = \int_{[0,1]^d} k_{\mathrm{RBF}}(x,x')dx = a\prod_{i=1}^{d}\big[\Phi(\tfrac{x_i'}{\ell}) - \Phi(\tfrac{x_i'-1}{\ell})\big] \tag{24}$$

$$\mathcal{P}k_{\mathrm{RBF}}\mathcal{P}' = \int_{[0,1]^d\times[0,1]^d} k_{\mathrm{RBF}}(x,x')dxdx' = a\big[\ell\sqrt{\tfrac{2}{\pi}}(e^{-1/2\ell^2}-1) + 2\Phi(\tfrac{1}{\ell}) - 1\big]^d \tag{25}$$

where Φ is again the univariate standard normal CDF.

E EQUIVARIANCE

E.1 TRANSLATION EQUIVARIANCE

A key factor in the generalization of convolutional neural networks is their translation equivariance. Patterns in different parts of an input signal can be seen in the same way because convolution is translation equivariant. Our learnable linear operators $\mathcal{A}$ are equivariant to continuous transformations. Two linear operators $e^{\mathcal{C}}$ and $\mathrm{e}^{\mathcal{B}}$ commute $[\mathrm{e}^{\mathcal{C}},\mathrm{e}^{\mathcal{B}}]=0$ if and only if their generators commute: $[\mathcal{C},\mathcal{B}]=0$. Since the generator of diffusions $\mathcal{D}_i$ is a sum of derivative operators, and the generators of translations are just ∇ as mentioned in section 4.2, the two commute: $[\mathcal{D}_k,\nabla]=0$. Therefore $[\mathcal{A},\tau_a] = [\sum_k W_k\mathrm{e}^{\mathcal{D}_k},\tau_a] = \sum_k W_k[\mathrm{e}^{\mathcal{D}_k},\mathrm{e}^{a^\top\nabla}] = 0$ and $\mathcal{A}$ is translation equivariant.

E.2 STEERABLE EQUIVARIANCE FOR LINEAR OPERATORS

For some tasks like medical segmentation, aerial imaging, and chemical property prediction there are additional symmetries in the data it makes sense to exploit other than mere translation equivariance. Below we show how to enforce equivariance of the Linear operator $\mathcal{A}$ to other symmetry groups G such as the group of continuous rotations $\mathrm{SO}(d)$ in $\mathbb{R}^d$. Applying equivariance constraints separately on each of the components of $\mathrm{e}^{\mathcal{D}_i}$ on top of translation equivariance yields very restricted set of operators. For example, enforcing equivariance to continuous rotations $G=\mathrm{SO}(d)$, the operator must be an isotropic heat kernel: $\mathcal{D}_k = c_k\nabla^\top\nabla$. The reason for this apparent restriction is a result of considering the different channels independently, as scalar fields.

The alternative is to use features fields which transform under more general representations of the symmetry group, introduced in steerable-CNNs (Cohen and Welling, 2016) and used in (Worrall et al., 2017; Thomas et al., 2018; Weiler et al., 2018; Weiler and Cesa, 2019) and others. In this way, the symmetry transformation acts not only on the spatial domain $\mathcal{X}$, but also transforms the channels. The way that the group acts on $\mathbb{R}^c$ (i.e. the channels) is formalized by a representation matrix $\rho(g)\in\mathbb{R}^{c\times c}$ for each element $g\in G$ in the transformation group that satisfies $\forall g,h\in G$: $\rho(gh)=\rho(g)\rho(h)$. Choosing the type of each intermediate feature map is equivalent to choosing their representations, and we describe a simple way of doing this with tensor representations in the later section.

Operator Equivariance Constraint: Returning to linear operators, we derive the equivariance constraint and show how to use constructs from the previous sections to implement steerable rotation equivariance. Equivariance of a linear operator $\mathcal{A} : (\mathbb{R}^d\to\mathbb{R}^{c_{in}})\to(\mathbb{R}^d\to\mathbb{R}^{c_{out}})$ requires that, for any input function, transforming the input function first (both argument and channels) and applying $\mathcal{A}$ is equivalent to first applying $\mathcal{A}$ and then transforming the output: $\mathcal{A}\rho_{in}(g)L_gf = \rho_{out}(g)L_g\mathcal{A}f$ where $L_gf(x)=f(g^{-1}x)$. Rearranging the terms, one sees that the equivariance constraint on the linear operator $\mathcal{A}$ is:

$$\rho_{out}(g)L_g\mathcal{A}L_{g^{-1}}\rho_{in}(g^{-1}) = \mathcal{A}, \tag{26}$$

where the operators L_g and L_g^{-1} are understood not to act on the representation matrices ρ (although implicitly a function of g). As shown in Appendix E.3, eq. 26 is a direct generalization of the equivariance constraint for convolutions $\forall x : \rho_{out}(g)K(g^{-1}x)\rho_{in}(g^{-1}) = K(x)$ described in the literature (Weiler and Cesa, 2019; Cohen et al., 2019).

As shown in Appendix E.5, the equivariance constraint for continuous rotations applied to the diffusion operators $\mathcal{A}=\sum_k W_k\mathrm{e}^{\mathcal{D}_k}$ has only the trivial solutions of isotropic diffusion without any drift. For this reason we instead consider a more general form of diffusion operator where the PDE itself couples the different channels. For the coupled PDE:

$$\frac{\partial f}{\partial t} = \sum_k W_k \mathcal{D}_k f \tag{27}$$

the time evolution contains the matrices W_k *in* the exponential $\mathcal{A} = e^{\sum_k W_k \mathcal{D}_k}$. Like with the example of translation above, this operator is equivariant if and only if the infinitesmal generator $\sum_k W_k \mathcal{D}_k$ is equivariant.

Because equation 26 applies generally to linear operators and not just convolutions, we can compute the equivariance constraint for these derivative operators. We can simplify the summation $\sum_k W_k \mathcal{D}_k = \sum_k W_k(\beta_k^T \nabla + (1/2)\nabla^T \Sigma_k \nabla)$ by writing it in terms of the collection of matrices $B_i = \sum_k W_k \beta_{ki}$ and $S_{ij} = (1/2)\sum_k W_k \Sigma_{kij}$ to express $\mathcal{A}_{\text{deriv}} = \sum_i B_i \partial_i + \sum_{i,j} S_{ij}\partial_i\partial_j$ where the indices $i, j = 1, 2..., d$ enumerate the spatial dimensions of each vector β_k and each matrix Σ_k. As we derive in appendix E.4, the necessary and sufficient conditions for the equivariance of $\sum_k W_k \mathcal{D}_k$ and therefore $\mathcal{A}$ is that $\forall g \in G : [\rho_{out} \otimes \rho_{in}^* \otimes \rho_{(1,0)}](g)\text{vec}(B) = \text{vec}(B)$ and $\forall g \in G : [\rho_{out} \otimes \rho_{in}^* \otimes \rho_{(2,0)}](g)\text{vec}(S) = \text{vec}(S)$ where $\text{vec}(\cdot)$ denotes flattening the elements into a single vector and $\rho_{(r,s)}$ is the tensor representation with r covariant and s contravariant indices.

E.3 GENERALIZATION OF EQUIVARIANCE CONSTRAINT FOR CONVOLUTIONS

This equivariance constraint is a direct generalization of the equivariance constraint for convolution kernels as described in Weiler and Cesa (2019); Cohen et al. (2019). In fact, when $\mathcal{A}$ is a *convolution* operator, $\mathcal{A}f = K * f$, the action of L_g by conjugation $\mathcal{A}$ is equivalent to transforming the argument of the kernel K:

$$\begin{aligned} L_g(K*)L_{g^{-1}}f(x) &= \int K(g^{-1}x - x')f(gx')d\mu(x') \\ &= \int K(g^{-1}(x - x''))f(x'')d\mu(x'') = (L_g[K]) * f. \end{aligned}$$

Letting both sides of eq 26 act on the product of a constant unit vector e_i and a delta function, $f = e_i\delta$ the expression $\forall e_i : \rho_{out}(g)L_g[K]\rho_{in}(g^{-1}) * e_i\delta = K * e_i\delta$ can be rewritten as $\forall x : \rho_{out}(g)K(g^{-1}x)\rho_{in}(g^{-1}) = K(x)$ which is precisely the constraint for steerable equivariance for convolution described in the literature. [5]

E.4 EQUIVARIANT DIFFUSIONS WITH MATRIX EXPONENTIAL

Below we solve for the **necessary** and **sufficient** conditions for the equivariance of the operator $\mathcal{A}_{\text{deriv}}$.

We will use tensor representations for their convenience, but the approach is general to allow other kinds of representations. A rank (p, q) tensor t is an element of the vector space $T_{(p,q)} := V^{\otimes p} \otimes (V^*)^{\otimes q}$ where V is some underlying vector space, V^* is its dual and $(\cdot)^{\otimes p}$ is the tensor product iterated p times. In common language $T_{(0,0)}$ are scalars, $T_{(1,0)}$ are vectors, and $T_{(1,1)}$ are matrices. Given the action of a group G on the vector space V, the representation on $T_{(p,q)}$ is $\rho_{(p,q)}(g) = g^{\otimes p} \otimes (g^{-\top})^{\otimes q}$ where $-\top$ is inverse transpose and $\otimes$ on the matrices is the tensor product (Kronecker product) of matrices. Composite representations can be formed by stacking different tensor ranks together, such as a representation of 50 scalars, 25 vectors, 10 matrices and 5 higher order tensors: $T_{(0,0)}^{50} \oplus T_{(1,0)}^{25} \oplus T_{(1,1)}^{10} \oplus T_{(1,2)}^{5}$, where $\oplus$ in this context is the same as the Cartesian product. For a composite representation $U = \bigoplus_i T_{(p_i,q_i)}$ the group representation is similarly $\rho_U(g) = \bigoplus_i \rho_{(p_i,q_i)}(g)$ where $\oplus$ concatenates matrices as blocks on the diagonal.

Noting that the operator L_g that acts only on the argument and the matrix $\rho_{in}(g)$ acts only on the components, the two commute and we can rewrite the constraint for $\mathcal{A}_{\text{deriv}}$ as

$$\sum_i \rho_{out}(g)B_i\rho_{in}(g^{-1})L_g\partial_i L_{g^{-1}} + \sum_{ij} \rho_{out}(g)S_{ij}\rho_{in}(g^{-1})L_g\partial_i\partial_j L_{g^{-1}} = \mathcal{A}_{\text{deriv}} \tag{28}$$

---

[5] This assumes as is typically done that measure μ over which the convolution is performed is left invariant. For the more general case, see the discussion in Bekkers (2019).

We can simplify the expression $L_g \partial_i L_{g^{-1}}$ by seeing how it acts on a function. For any differentiable function $\partial_i L_{g^{-1}} f(x) = \frac{\partial}{\partial x_i}[f(gx)] = \sum_j g_{ji}[\partial_j f](gx) = L_{g^{-1}} \sum_j g_{ji} \partial_j f(x)$ where g_{ij} are the components of the matrix g. Since this holds for any f, we find that $L_g \nabla L_{g^{-1}} = g^T \nabla$ and therefore $L_g \nabla \nabla^T L_{g^{-1}} = L_g \nabla L_{g^{-1}} L_g \nabla^T L_{g^{-1}} = g^T \nabla \nabla^T g$.

Since equation 28 holds as an operator equation, it must be true separately for each component ∂_i and $\partial_i \partial_j$. This means that the constraint separates into a constraint for B and a constraint for S:

1. $\forall g, i : \sum_j g_{ij} \rho_{out}(g) B_j \rho_{in}(g^{-1}) = B_i$
2. $\forall g, i, j : \sum_{kl} g_{i\ell} g_{ik} \rho_{out}(g) S_{\ell k} \rho_{in}(g^{-1}) = S_{ij}$.

These relationships can be expressed more succinctly by flattening the elements of B and S into vectors: $[\rho_{out}(g) \otimes \rho_{in}(g^{-T}) \otimes \rho_{(1,0)}(g)]\mathrm{vec}(B) = \mathrm{vec}(B)$ and $[\rho_{out}(g) \otimes \rho_{in}(g^{-T}) \otimes \rho_{(2,0)}(g)]\mathrm{vec}(S) = \mathrm{vec}(S)$.

E.5 ROTATION EQUIVARIANCE CONSTRAINT FOR SCALAR DIFFUSIONS HAS ONLY TRIVIAL SOLUTIONS

The diffusion operator $\mathcal{A} = \sum_k W_k e^{\mathcal{D}_k}$ leads to only trivial $\beta_k = 0$ and $\Sigma_k \propto I$ if it satisfies the continuous rotation equivariance constraint.

Proof:

The application of $e^{\mathcal{D}_k}$ is just a convolution with the Greens function

$$\sum_k W_k e^{\mathcal{D}_k} f = \sum_k W_k [e^{-\frac{1}{2}(x+\beta_k)^\top \Sigma_k^{-1}(x+\beta_k)} \det(2\pi\Sigma_k)^{-1/2}] * f = \sum_k W_k G_k * f \tag{29}$$

where the Greens function is the multivariate Gaussian density: $G_k(x) = \mathcal{N}(x; -\beta_k, \Sigma_k)$.

As shown in appendix E.3, for convolutions the operator constraint is equivalent to the kernel equivariance constraint $\rho_{out}(g)K(g^{-1}x)\rho_{in}(g^{-1}) = K(x)$ from (Weiler and Cesa, 2019). With $K(x) = \sum_k W_k G_k(x)$ this reads:

$$\forall x \in \mathbb{R}^d, g \in G : \quad \sum_k \rho_{out}(g) W_k \mathcal{N}(g^{-1}x; -\beta_k, \Sigma_k) \rho_{in}(g^{-1}) = \sum_k W_k \mathcal{N}(x; -\beta_k, \Sigma_k),$$

For rotations $g \in \mathrm{SO}(2)$ where we can parametrize $g_\theta = e^{\theta J}$ in terms of the antisymmetric matrix $J = [[0,1],[-1,0]] \in \mathbb{R}^{2\times 2}$ and the translation operator can be written $L_g = e^{-\theta x^T J^T \nabla}$, we can take derivatives with respect to θ to get (now with double sums implicit):

$$\forall x \in \mathbb{R}^d : \sum_k \left[d\rho_{out} W_k \mathcal{N}(x; -\beta_k, \Sigma_k) - W_k \mathcal{N}(x; -\beta_k, \Sigma_k) d\rho_{in} - W_k (x^T J^T \nabla) \mathcal{N}(x; -\beta_k, \Sigma_k) \right] = 0.$$

Here the Lie Algebra representation of J is $d\rho := \frac{\partial}{\partial\theta}\rho(g_\theta)|_{\theta=0}$. Factoring out the normal density:

$$\forall x \in \mathbb{R}^d : \quad \sum_k \left[d\rho_{out} W_k - W_k d\rho_{in} - W_k (x^T J^T \Sigma_k^{-1}(x + \beta_k)) \right] \mathcal{N}(x; -\beta_k, \Sigma_k) = 0.$$

Without loss of generality we may assume that each of the Gaussians β_k, Σ_k pairs are distinct since if they were not then we could replace the collection with a single element. Since the (finite) sum of distinct Gaussian densities is never a Gaussian density, and monomials of order > 0 multiplied by a Gaussian density cannot be formed with sums of Gaussian densities or sums multiplied by monomials of a different order and Gaussian densities are never 0, this constraint separates out into several independent constraints.

1. $\forall i : d\rho_{out} W_k = W_k d\rho_{in}$
2. $\forall i, x : W_k (x^T J^T \Sigma_k^{-1} \beta_k) = 0$
3. $\forall i, x : W_k (x^T J^T \Sigma_k^{-1} x) = 0$

We may assume w.l.o.g. that W_k is not 0 for all components of the matrix (otherwise we could have deleted this element of k and continue). Therefore there is some component which is nonzero, and the expressions in parentheses in equations 2 and 3 must be 0. Given that this holds for all x, eq 3 implies: $J^T \Sigma_k^{-1} = 0$ or equivalently $\Sigma_k^{-1} J = 0$ because Σ_k is symmetric, and since $J = -J^T$ this can be expressed concisely as $[\Sigma_k^{-1}, J] = 0$ for which the only symmetric solution is proportional to the identity $\Sigma_k = c_k I$. Since both Σ_k and J are invertible, equation 2 yields $\beta_k = 0$. Therefore there are no nontrivial solutions for β, Σ in $\mathcal{A} = \sum_k W_k e^{\mathcal{D}_k}$ for continuous rotation equivariance.

F DATASET AND TRAINING DETAILS

In this section we elaborate on some of the details regarding hyperparameters, network architecture, and the datasets.

As described in the main text, the PNCNN is composed of a chain of convolutional blocks containing a convolution layer, a probabilistic ReLUs, and linear channel mixing layer (analogue of the colloquial 1×1 convolution). In each of these convolutional blocks, the input is a collection of points and feature mean value at those points along with the feature elementwise standard deviation at those points: $\{(x_i, \mu(x_i), \sigma(x_i)\}_{i=1}^N$. These observations seed the GP layer, and the block is evaluated at the same collection of points for the output (although it can be evaluated elsewhere since it is a continuous process, and we make use this fact to visualize the features in figures 1 and 2).

Hyperparameters: For the PNCNN on the Superpixel MNIST dataset, we use 4 PNCNN convolution blocks with $c = 128$ channels and with $K = 9$ basis elements for the different drift and diffusion parameters in $\sum_{k=1}^K W_k e^{\mathcal{D}_k}$. We train for 20 epochs using the Adam optimizer (Kingma and Ba, 2014) with $\mathrm{lr} = 310^{-3}$ with batch size 50.

For the PNCNN on the PhysioNet2012 dataset, we use the variant of the PNCNN convolution layer that uses the stochastic diagonal estimator described in appendix G with $P = 20$ probes. In the convolution blocks we use $c = 96$ channels, $K = 5$ basis elements and we train for 10 epochs using the same optimizer settings above. For both datasets we tuned hyperparameters on a validation set of size 10% before folding the validation set back into the training set for the final runs. Both models take about 2 hours to train.

SuperPixel-MNIST We source the SuperPixel MNIST dataset (Monti et al., 2017) from Fey and Lenssen (2019) consisting of $60k$ training examples and $10k$ test represented as collections of positions and grayscale values $\{(x_i, f(x_i))\}_{i=1}^{75}$ at the $N = 75$ super pixel centroids.

PhysioNet2012 We follow the data preprocessing from Horn et al. (2019) and the $10k$-$2k$ train test split. The individual data points consist of 42 irregularly spaced vital sign time series signals as well as 5 static variables: Gender, ICU Type, Age, Height, Weight. We use one hot embeddings for the first two categoric variables, and we treat each of these static signals as fully observed constant time series signals. As the binary classification task exhibits a strong label imbalance, 14% positive signals, we apply an inverse frequency weighting of $1/.14$ to the binary cross entropy loss.

G STOCHASTIC DIAGONAL ESTIMATION FOR PHYSIONET2012

In order to compute the mean and variance of the rectified Gaussian process, the activations of the probabilistic ReLU, we need compute the diagonal of $\mathcal{A} k_p \mathcal{A}'(x_n, x_n)$ for the relevant points $\{x_n\}_{n=1}^N$.

In the usual case where each of the channels $\alpha = 1, 2, ..., c$ are observed at the same locations this can be done efficiently. First one computes the application of $e^{\mathcal{D}_i}$ on the left and $e^{\mathcal{D}'_j}$ on the right onto the posterior k_p:

$$N_{ij} = (e^{\mathcal{D}_i} k_p e^{\mathcal{D}'_j})(x_n, x_n) = (e^{\mathcal{D}_i} k e^{\mathcal{D}'_j})(x_n, x_n) - (e^{\mathcal{D}_i} \mathbf{k}^\top)(x_n)[K + S]^{-1}(\mathbf{k} e^{\mathcal{D}'_j})(x_n)$$

where k is the RBF kernel and we have reused the notation from appendix A. Notably, this quantity is the same for each of the channels, and the elementwise variance is just:

$$v_\alpha(x_n) = (\mathcal{A} k_p \mathcal{A}')_{\alpha\alpha}(x_n, x_n) = \sum_{i,j,\beta} W_i^{\alpha\beta} N_{ij} W_j^{\alpha\beta} \tag{30}$$

where the α, β index the channels of each of the matrices W_i. Because N is the same for all channels, we can compute this quantity efficiently with a reasonable memory cost and compute.

For the PhysioNet2012 dataset where the observation points differ between the channels we must consider a different observation set $\{x_n^\beta\}_{n=1}^N$ for each channel β. This means that evaluated kernel depends on the channel and we have the objects: $\mathbf{k}^\beta$, K^β and S^β. As a result, we have an additional index for N_{ij}^β and the desired computation is

$$v_\alpha(x_n^\alpha) = (\mathcal{A}k_p\mathcal{A}')_{\alpha\alpha}(x_n^\alpha, x_n^\alpha) = \sum_{i,j,\beta} W_i^{\alpha\beta} N_{ij}^\beta W_j^{\alpha\beta}. \tag{31}$$

While each of the terms in the computation can be computed without much difficulty, performing the summation explicitly requires an unreasonably large memory cost and also compute.

However, by the same approach we can consider the full covariance matrix $\mathbf{B}_{(\alpha n)(\beta m)} = (\mathcal{A}k_p\mathcal{A}')_{\alpha\beta}(x_n^\alpha, x_m^\beta)$, and while it would not be feasible to compute this matrix directly we *can* define matrix vector multiplies onto vectors of size $\mathbb{R}^{cN}$ implicitly using the sequence of operations that define it. Crucially, this sequence of operations has much more modest memory consumption (and compute cost) over the direct expression in equation 31. These implicit matrix vector multiplies can then be used to compute a stochastic diagonal estimator (Bekas et al., 2007) given by:

$$\hat{v}_\alpha(x_n^\alpha) = \tfrac{1}{P}\sum_{p=1}^{P} z_p \odot \mathbf{B} z_p \tag{32}$$

with Gaussian probe vectors $z_p \sim \mathcal{N}(0, I)$, and where $\odot$ is elementwise multiplication (see Bekas et al. (2007) for more details on this stochastic diagonal estimator). We use this estimator with $P = 20$ probes for computing the variances for PhysioNet. We note that with $P = 20$ the variance estimates are still quite noisy, however without the estimator cannot readily apply the PNCNN to PhysioNet. We leave a better approach for handling this kind of data to future work.

H PATHOLOGIES IN PROJECTION TO RBF GAUSSIAN PROCESS

In section 4.7 describe an approach by which a Gaussian process with a complex mean and covariance function is *projected* down to the posterior of a (simpler) RBF kernel GP from a set of observations. We know given the representation capacity of the RBF kernel that with the right set of observations, a complex function can be well approximated in principle. However, the relationship for uncertainty is less straightforward.

The properties of the input Gaussian process must be conveyed to the output Gaussian process by only the (uncorrelated) noisy observations $\{(x_i, \mu(x_i), \sigma(x_i))\}_{i=1}^N$. As the uncertainty in original GP increases, so do the measurement uncertainties in the transmission, and therefore the output GP also has a higher uncertainty. However, the uncertainty in the input GP is in the form of a full covariance kernel $k(x, x')$ and it seems that individual observations will not easily be able to communicate the *covariance* of the values of the GP function at different spatial locations despite the heterogeneous noise model.

Fundamentally, the problem is that the observation values are treated as independent, an incorrect assumption which has other knock-on effects when the number of observations is large. With some fixed measurement error no matter how high but a large enough set of independent observations, the mean value can be pinned down precisely. If in contrast the observations are not independent, then there may be a situation where the mean value cannot be known more precisely than some limiting uncertainty. This effect leads the output GP to have less uncertainty and be more confident in the values that it should be given the input GP.

If the observations are sparse, then the effective sample size of the estimator for the mean of the GP at any given location is small, and then the amount by which uncertainty is underestimated is small. However, if there are many many observations then this kind of observation transmission of information with the independence assumption will attenuate the uncertainty. We would also expect that over the course of many layers, this attenuation can accumulate. We believe that this is what causes the poorer uncertainty calibration in layers 3 and 4 of the PNCNN shown in figure 2. We hope that this problem can be resolved perhaps by removing the independence assumption or providing an alternative projection method in future work.

I CENTRAL LIMIT THEOREM FOR STOCHASTIC PROCESSES

We derive a variant a variant of the Lyapunov central limit theorem (CLT) holding for stochastic processes. The main ideas is that the result for processes follows from applying the multivariate Lyapunov CLT to the joint distribution of each finite collection of values as per the definition of a Gaussian process.

In more details the argument goes as follows. We are given C stochastic processes $g_c(x)$ and we assume that they are weakly dependent, i.e. $\mathbb{E}[g_c(x)g_{c'}(x')] \to \mathbb{E}[g_c(x)]\mathbb{E}[g_{c'}(x')]$ as $|c-c'| \gg 1$, for any x, x'. We would like to show that $\bar{g}(x) \sim \mathcal{GP}(\mu, k)$, where $\mu(x) = \sum_{c=1}^{C} g_c(x)$ and $k(x,x') = \sum_{c,c'=1}^{C} \mathbb{E}[g_c(x)g_{c'}(x')]$. For these formulas to make sense, we need some bounds on the moments of g_c. If the individual components g_c scale as $1/\sqrt{C}$, then the covariance if finite.

Now choose any finite collection of indices $x_1, x_2, \ldots, x_N$. Then consider the random vector $\bar{g}(x_i) = \sum_{c=1}^{C} g_c(x_i)$, $i = 1, \ldots, N$. We can now apply the CLT to deduce that $\{\bar{g}(x_i)\}_{i=1}^{N}$ is Gaussian distributed. Since a stochastic process is determined by its finite distributions, we can conclude that the random function $\bar{g}(x) \to \mathcal{GP}(\mu, k)$, as was to be shown.

J MOMENTS OF RECTIFIED GAUSSIAN RANDOM VARIABLES

Let $\boldsymbol{f} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ be a d dimensional random Gaussian vector. We compute here

$$\mathbb{E}(\mathrm{ReLU}(f_1)\cdots\mathrm{ReLU}(f_d)) = \frac{1}{N_\Sigma}\int_{\boldsymbol{f}>0} \mathrm{d}^d\boldsymbol{f}\, f_1\cdots f_d \exp -\tfrac{1}{2}(\boldsymbol{f}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{f}-\boldsymbol{\mu}) \tag{33}$$

$$N_{\boldsymbol{\Sigma}} = (2\pi)^{d/2}\det(\boldsymbol{\Sigma})^{1/2}\,. \tag{34}$$

We use the generating function technique. Define

$$Z(\boldsymbol{b}) = \frac{1}{N_\Sigma}\int_{\boldsymbol{f}>0} \mathrm{d}^d\boldsymbol{f} \exp[-\tfrac{1}{2}(\boldsymbol{f}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{f}-\boldsymbol{\mu}) + \boldsymbol{b}^T\boldsymbol{f}] \tag{35}$$

$$= \frac{1}{N_\Sigma}e^{\boldsymbol{b}^T\boldsymbol{\mu}}\int_{\boldsymbol{f}>-\boldsymbol{\mu}} \mathrm{d}^d\boldsymbol{f} \exp[-\tfrac{1}{2}\boldsymbol{f}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{f} + \boldsymbol{b}^T\boldsymbol{f}]\,. \tag{36}$$

Then

$$\mathbb{E}(\mathrm{ReLU}(f_1)\cdots\mathrm{ReLU}(f_d)) = \frac{\partial}{\partial b_1}\cdots\frac{\partial}{\partial b_d}Z(\boldsymbol{b})\bigg|_{\boldsymbol{b}=0}\,. \tag{37}$$

To compute $Z(\boldsymbol{b})$ we proceed as in Gaussian case. We change variables to

$$\boldsymbol{f} = \boldsymbol{\Sigma}\boldsymbol{b} + \boldsymbol{g}\,, \tag{38}$$

and define $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\Sigma}\boldsymbol{b}$ to get:

$$Z(\boldsymbol{b}) = e^{\boldsymbol{b}^T\boldsymbol{\mu}+\frac{1}{2}\boldsymbol{b}^T\boldsymbol{\Sigma}\boldsymbol{b}}\frac{1}{N_{\boldsymbol{\Sigma}}}\int_{\boldsymbol{g}<+\boldsymbol{z}} \mathrm{d}^d\boldsymbol{g} \exp[-\tfrac{1}{2}\boldsymbol{g}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{g}] \tag{39}$$

$$= e^{S(\boldsymbol{b})}\,\Phi^{(d)}(\boldsymbol{z}; 0, \boldsymbol{\Sigma})\,, \quad S(\boldsymbol{b}) = \boldsymbol{b}^T\boldsymbol{\mu} + \tfrac{1}{2}\boldsymbol{b}^T\boldsymbol{\Sigma}\boldsymbol{b}\,. \tag{40}$$

Φ being the multivariate standard Normal CDF:

$$\Phi^{(d)}(\boldsymbol{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \int_{\boldsymbol{g}<+\boldsymbol{z}} \mathrm{d}^d\boldsymbol{g}\psi^{(d)}(\boldsymbol{g}; \boldsymbol{\mu}, \boldsymbol{\Sigma})\,, \tag{41}$$

$$\psi^{(d)}(\boldsymbol{g}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{N_{\boldsymbol{\Sigma}}}\exp[-\tfrac{1}{2}(\boldsymbol{g}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{g}-\boldsymbol{\mu})]\,. \tag{42}$$

Now we compute the first two derivatives. Note that in $d = 1$, denoting $\sigma^2 = \Sigma$:

$$\frac{\partial}{\partial z}\Phi^{(1)}(z, 0, \sigma^2) = \psi^{(1)}(z, 0, \sigma^2)\,. \tag{43}$$

In $d = 2$, we can use the conditional probability decomposition to get the required derivatives:

$$\psi^{(2)}(\boldsymbol{g}; 0, \boldsymbol{\Sigma}) = \psi^{(1)}(g_1; \alpha_1 g_2, \beta_1) \cdot \psi^{(1)}(g_2; 0, \Sigma_{22}) \tag{44}$$

$$\alpha_1 = \Sigma_{12}\Sigma_{22}^{-1}\,, \beta_1 = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} \tag{45}$$

$$\partial_{z_2}\Phi^{(2)}(\boldsymbol{z}, 0, \boldsymbol{\Sigma}) = \psi^{(1)}(z_2; 0, \Sigma_{22}) \int_{-\infty}^{z_1} \mathrm{d}g_1 \psi^{(1)}(g_1; \alpha_1 z_2, \beta_1)\,, \tag{46}$$

$$= \psi^{(1)}(z_2; 0, \Sigma_{22})\Phi^{(1)}(z_1, \alpha_1 z_2, \beta_1) \tag{47}$$

$$\partial_{z_2}^2\Phi^{(2)}(\boldsymbol{z}, 0, \boldsymbol{\Sigma}) = -\frac{z_2}{\Sigma_{22}}\psi^{(1)}(z_2; 0, \Sigma_{22})\Phi^{(1)}(z_1, \alpha_1 z_2, \beta_1) \tag{48}$$

$$+ \psi^{(1)}(z_2; 0, \Sigma_{22})\partial_{z_2}\Phi^{(1)}(z_1, \alpha_1 z_2, \beta_1) \tag{49}$$

$$\partial_{z_1}\partial_{z_2}\Phi^{(2)}(\boldsymbol{z}, 0, \boldsymbol{\Sigma}) = \psi^{(2)}(\boldsymbol{z}; 0, \boldsymbol{\Sigma})\,. \tag{50}$$

So denoting $\partial_i = \frac{\partial}{\partial b_i}$, we get:

$$\partial_i Z(\boldsymbol{b}) = (\mu_i + \sum_j \Sigma_{ij} b_j) Z(\boldsymbol{b}) + \underbrace{e^{S(\boldsymbol{b})} \sum_\ell \partial_{z_\ell}\Phi^{(d)}(\boldsymbol{z}, 0, \boldsymbol{\Sigma})\Sigma_{\ell,i}}_{m_i(\boldsymbol{b})} \tag{51}$$

$$\partial_k\partial_i Z(\boldsymbol{b}) = \Sigma_{ik} Z(\boldsymbol{b}) + (\mu_i + \sum_j \Sigma_{ij} b_j)\partial_k Z(\boldsymbol{b}) \tag{52}$$

$$+ (\mu_k + \sum_j \Sigma_{kj} b_j) m_i(\boldsymbol{b}) + e^{S(\boldsymbol{b})} \sum_{\ell,q} \partial_{z_\ell}\partial_{z_q}\Phi^{(d)}(\boldsymbol{z}, 0, \boldsymbol{\Sigma})\Sigma_{\ell,i}\Sigma_{q,k}\,. \tag{53}$$

In particular, we have for $d = 1$:

$$\mathbb{E}(\mathrm{ReLU}(f)) = \mu\Phi^{(1)}(\mu; 0, \sigma^2) + \psi^{(1)}(\mu, 0, \sigma^2)\sigma^2\,, \tag{54}$$

which coincides with equation 8 and for $d = 2$:

$$\mathbb{E}(\mathrm{ReLU}(f_1)\mathrm{ReLU}(f_2)) = \Sigma_{12}\Phi^{(2)}(\boldsymbol{\mu}; 0, \boldsymbol{\Sigma}) + \mu_1\mu_2\Phi^{(2)}(\boldsymbol{\mu}; 0, \boldsymbol{\Sigma}) + \mu_1 m_2(0) + \mu_2 m_1(0) \tag{55}$$

$$+ \sum_{\ell,q=1,2} \Sigma_{\ell,1}\Sigma_{q,2}\partial_{z_\ell}\partial_{z_q}\Phi^{(2)}(\boldsymbol{z}, 0, \boldsymbol{\Sigma})|_{\boldsymbol{b}=0}\,. \tag{56}$$

which can be rewritten in the form of equation 9.