Taking Limits of Deep Representations

Anonymous Author(s)
Affiliation
Address
email

Abstract

A good latent representation captures all the relevant degrees of freedom of the manifold on which the data live. We show, for typical deep architectures, that as the number of layers increase, the representational capacity of the model tends to capture fewer degrees of freedom. In the limit, deep representations only retain a single degree of freedom locally. In addition, gradient-based learning becomes intractable. We propose several solutions to address these pathologies.

1 Introduction

Deep networks have become an important tool for machine learning [cite]. However, training these models are difficult, Many arguments have been made for the need for deep architectures [cite Bengio]. However, it is hard to know what effect the deepness of an architecture has. Also, the weights don't necessarily move that much from their initialization.

2 Main Results

The derivatives of a function drawn from a GP prior with an isotropic SE kernel are i.i.d. Normal Because differentiation is a linear operator, the derivatives of a function drawn from a GP prior are also jointly Gaussian distributed, with covariance between derivatives w.r.t. different dimensions of $\mathbf{x}$ given by:

$$\operatorname{cov}\left(\frac{\partial f(\mathbf{x})}{\partial x_{d_1}}, \frac{\partial f(\mathbf{x})}{\partial x_{d_2}}\right) = \frac{\partial^2 k(\mathbf{x}, \mathbf{x}'))}{\partial x_{d_1} \partial x'_{d_2}}\bigg|_{\mathbf{x}=\mathbf{x}'} \tag{1}$$

[cite carl's paper?] If our kernel is a product over individual dimensions $k(\mathbf{x}, \mathbf{x}') = \prod_d^D k_d(x_d, x'_d)$, as in the case of the isotropic squared-exp kernel, then the diagonal covariances are given by $\frac{\sigma_o^2}{\ell^2}$, and the off-diagonal entries are zero. This means that elements are independent and identically distributed.

The elements of the Jacobian of a GP with an isotropic SE kernel are i.i.d. Gaussians The Jacobian of the ℓ^{th}function is:

$$J^\ell_{\mathbf{x}\to\mathbf{y}}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1^\ell(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f_1^\ell(\mathbf{x})}{\partial x_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_D^\ell(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f_D^\ell(\mathbf{x})}{\partial x_D} \end{bmatrix} \tag{2}$$

Because we've assumed that the GP on each output dimension $f_d(\mathbf{x}) \sim \mathcal{GP}$ is independent, it follows that for a given $\mathbf{x}$, each row of $J_{\mathbf{x}\to\mathbf{y}}(\mathbf{x})$ is independent. Above, we showed that the elements of each

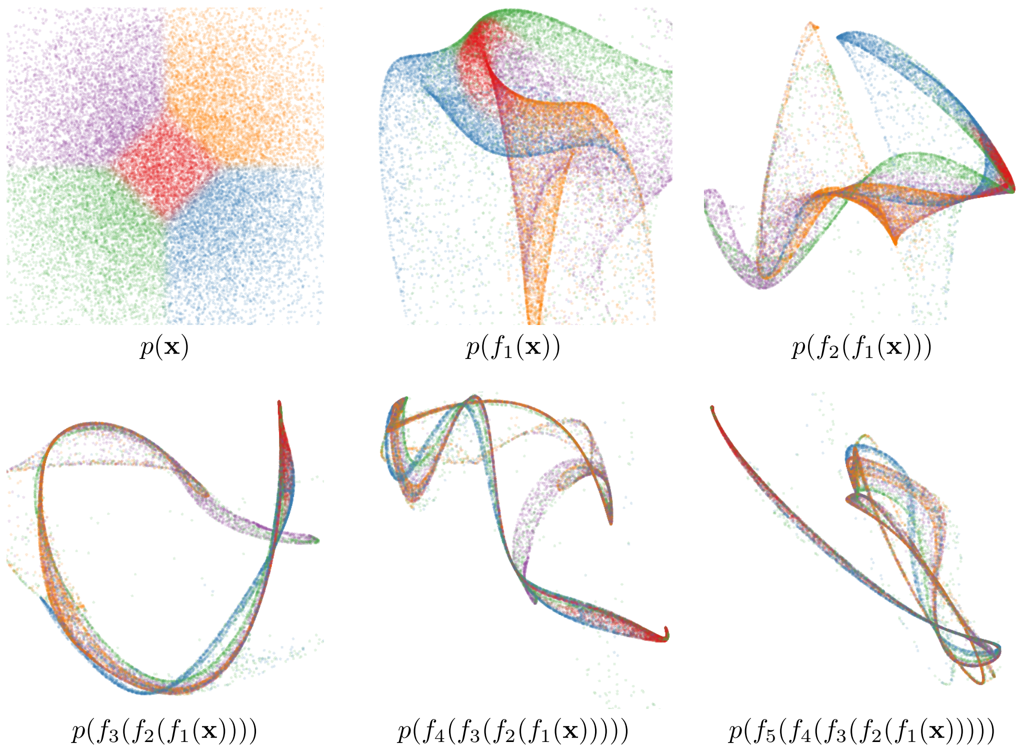


Figure 1: Draws from a deep GP. A distribution is warped by successive functions drawn from a GP prior.

row are independent. This means that each entry in the Jacobian of a GP-distributed transformation is i.i.d. Normal.

We also have that if $\mathbf{x} = f(\mathbf{y})$, then $J^{-1}_{\mathbf{y}\to\mathbf{x}}(\mathbf{y}) = J_{\mathbf{x}\to\mathbf{y}}(\mathbf{x})$.

The Jacobian of a deep GP is a product of random normal matrices By the multivariate chain rule, the derivative (Jacobian) of any compositions of functions is simply the product of the Jacobians of each function. and the Jacobian of the composed (deep) function is:

$$J^{1:L}(x) = \prod_{\ell=1}^{L} J^{L}(x) \qquad (3)$$

2.1 Filamentation

Attempted definition of a filament A continuous, twice-differentiable region of a pdf is called a *filament* to the degree that, weighted by density, only one eigenvalue of the Hessian of the pdf are is small relative to the average eigenvalue.

3 Conclusions

Deep neural networks and deep Gaussian processes are analyzable using random matrix theory. After proving that the Jacobian is an i.i.d. Gaussian matrix, many other forms of

If you want to use very deep nets, you won't be able to do so if you initialize/regularize all your weights independently We might want to think about different ways of

If you initialize independently, the density becomes fractal Points close in x-space can be very far in y-space, and vice versa.

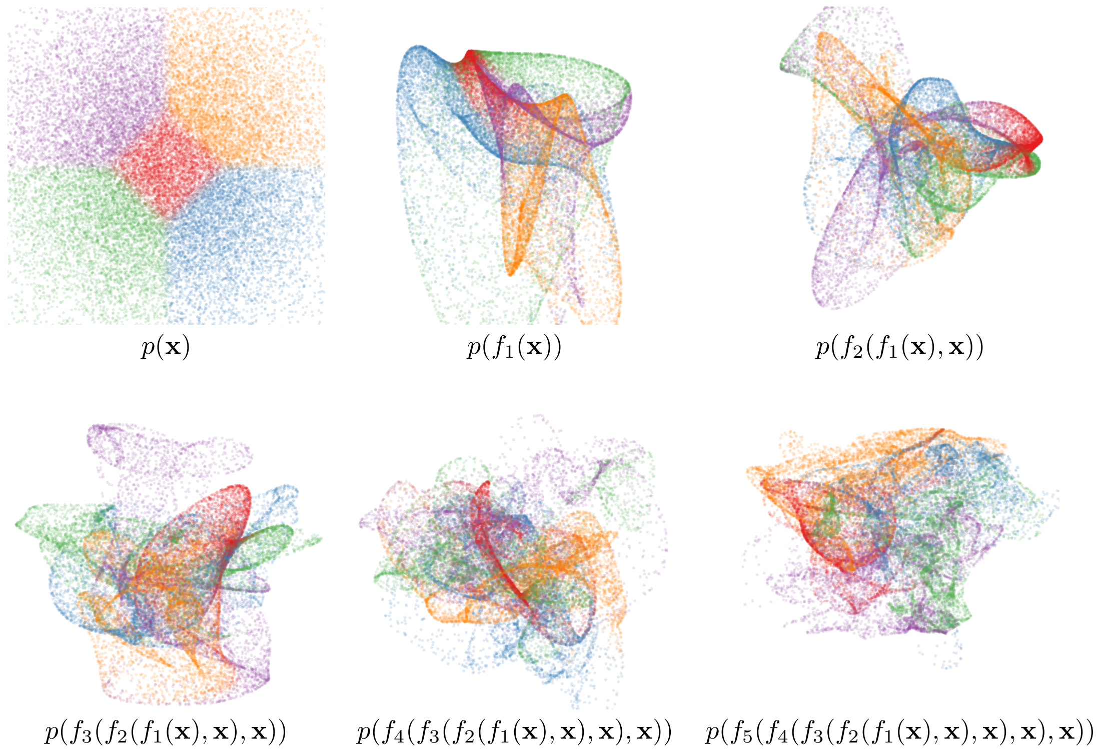


Figure 2: Draws from a deep GP. A distribution is warped by successive functions drawn from a GP prior.

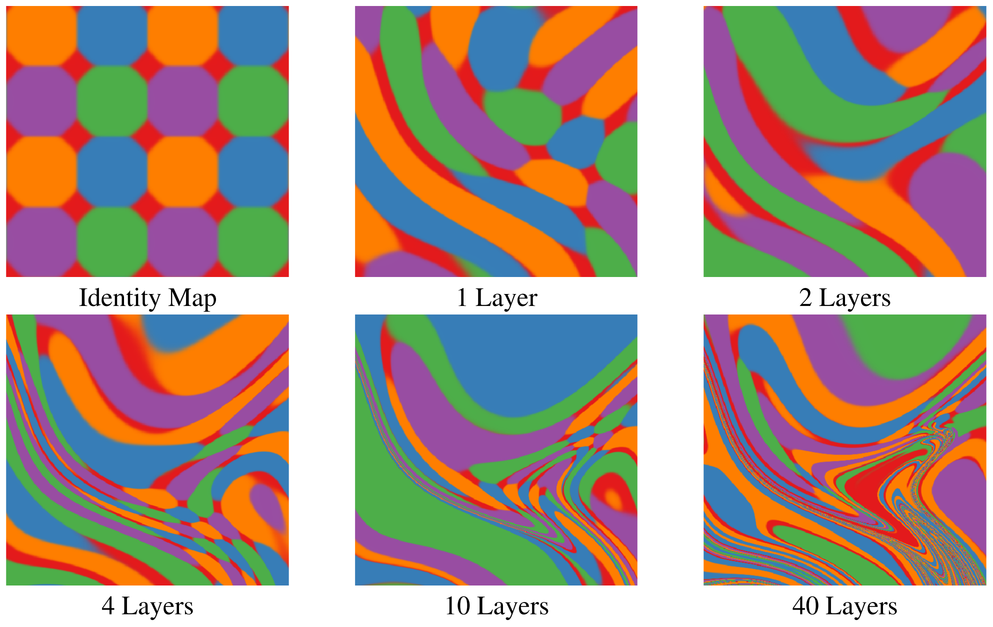


Figure 3: Feature Mapping of a deep GP.

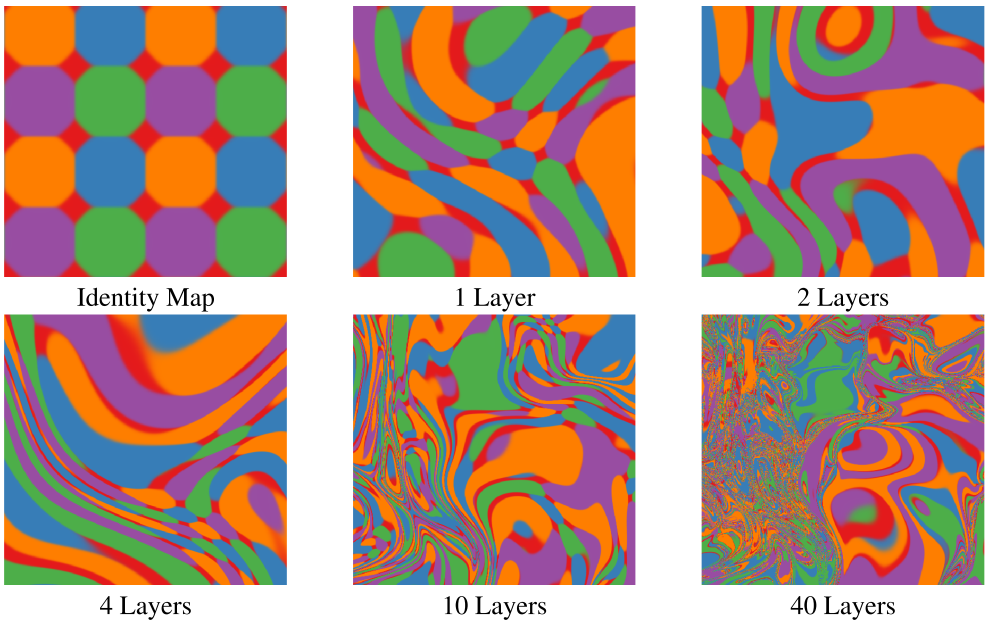


Figure 4: Feature Mapping of a deep GP with each layer connected to the output.

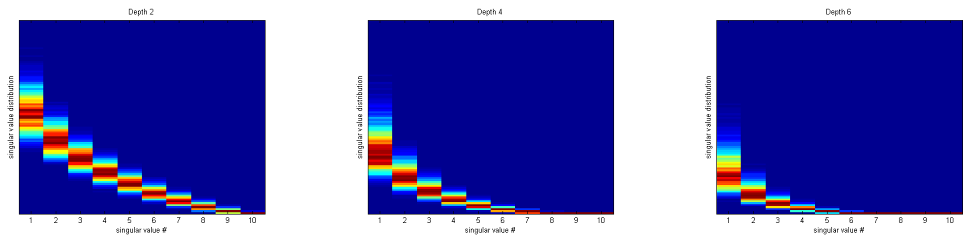
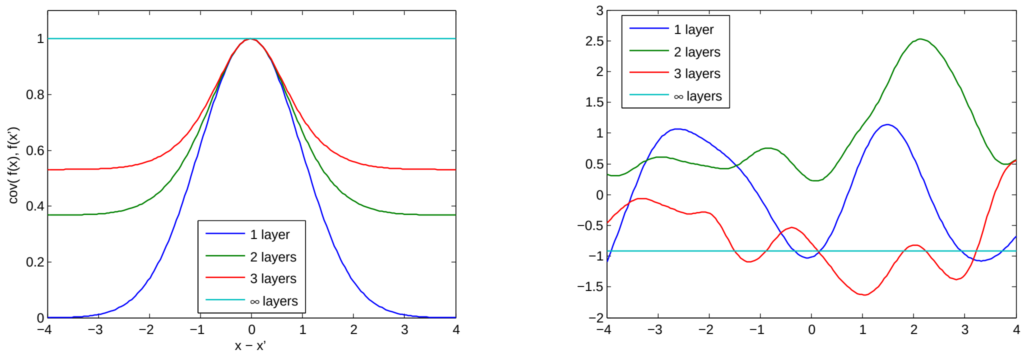


Figure 5: Eigenspectrum of the Jacobian. As the net gets deeper, the largest eigenvalue comes to dominate.



Kernel derived from iterated feature transforms Draws from the corresponding kernel

Figure 6: A degenerate kernel produced by repeatedly applying a feature transform.

A spikey eigenspecturm will lead to saturation Maybe we should initialize differently in order to avoid such saturation, like Martens' sparse initialization: http://www.cs.toronto.edu/~jmartens/docs/Deep_HessianFree.pdf

4 Infinitely Deep Kernels

[cite Youngmin Cho's thesis, and Radford Neal's]

One can derive a Gaussian process as a neural network: $f(x) = \alpha^T \Phi(x) = \sum_{i=i}^K \alpha_i \phi_i(x)$

We can consider applying the feature transform $\Phi(\cdot)$ to the features themselves: $\Phi(\Phi(\mathbf{x}))$,

$k_1(x,y) = e^{-||x-y||2}$, then the two-layer kernel is simply $k_2(x,y) = e^{k_1(x,y)-1}$. This formula is true for every layer: $k_{n+1}(x,y) = e^{k_n(x,y)-1}$.

$$k_2(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{1}{2}||\Phi(\mathbf{x}) - \Phi(\mathbf{x}')||_2^2\right) \tag{4}$$

$$k_2(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{1}{2}\sum_i \left[\phi_i(\mathbf{x}) - \phi_i(\mathbf{x}')\right]^2\right) \tag{5}$$

$$k_2(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{1}{2}\sum_i \left[\phi_i(\mathbf{x})^2 - 2\phi_i(\mathbf{x})\phi_i(\mathbf{x}') + \phi_i(\mathbf{x}')^2\right]\right) \tag{6}$$

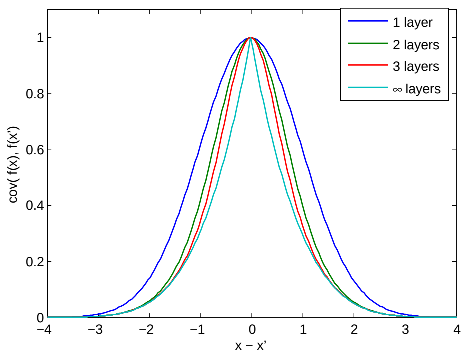
$$k_2(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{1}{2}\left[\sum_i \phi_i(\mathbf{x})^2 - 2\sum_i \phi_i(\mathbf{x})\phi_i(\mathbf{x}') + \sum_i \phi_i(\mathbf{x}')^2\right]\right) \tag{7}$$

$$k_2(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{1}{2}\left[k_1(\mathbf{x},\mathbf{x}) - 2k_1(\mathbf{x},\mathbf{x}') + k_1(\mathbf{x}',\mathbf{x}')\right]\right) \tag{8}$$

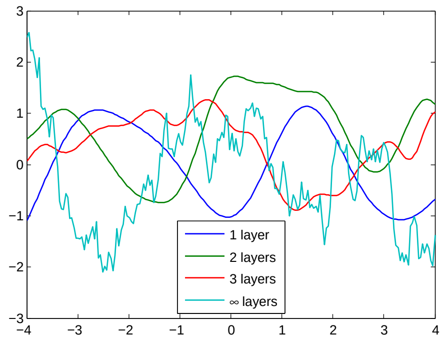
$$k_2(\mathbf{x},\mathbf{x}') = \exp\left(k_1(\mathbf{x},\mathbf{x}') - 1\right) \tag{9}$$

Note that nothing in this derivation depends on details of k_1, except that $k_1(\mathbf{x},\mathbf{x}) = 1$. Because this is true for k_2 as well, this recursion holds in general, and we have that $k_{n+1}(x,y) = e^{k_n(x,y)-1}$. The only stable solution for this recursion is $k(x,y) = 1$.

This solution is degenerate. One interpretation of why repeated feature transforms lead to this degenerate prior is that each layer can only lose information about the previous set of features. In the limit, the transformed features contain no information about the original input $\mathbf{x}$. Since the function doesn't depend on its input, it must be the same everywhere.



Kernel derived from iterated feature transforms with all layers connected to the input



Draws from the corresponding kernel

Figure 7: A non-degenerate version of the infinitely deep feature transform kernel. By connecting the inputs $\mathbf{x}$ to each layer, the function can still depend on its input even after arbitrarily many layers of computation.

4.1 Fixing the deep kernel

Follow a suggestion from Radford Nearl, we connect the inputs to each layer of features. We do this simply by augmenting the feature vector $\Phi(\mathbf{x})$ with the extra features $\mathbf{x}$:

$$k_2(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{1}{2}\left|\left|\begin{bmatrix}\Phi(\mathbf{x})\\ \mathbf{x}\end{bmatrix} - \begin{bmatrix}\Phi(\mathbf{x}')\\ \mathbf{x}'\end{bmatrix}\right|\right|_2^2\right) \tag{10}$$

$$k_2(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{1}{2}\sum_i \left[\phi_i(\mathbf{x}) - \phi_i(\mathbf{x}')\right]^2 - \frac{1}{2}||\mathbf{x}-\mathbf{x}'||_2^2\right) \tag{11}$$

$$k_2(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{1}{2}\sum_i \left[\phi_i(\mathbf{x})^2 - 2\phi_i(\mathbf{x})\phi_i(\mathbf{x}') + \phi_i(\mathbf{x}')^2\right] - \frac{1}{2}||\mathbf{x}-\mathbf{x}'||_2^2\right) \tag{12}$$

$$k_2(\mathbf{x},\mathbf{x}') = \exp\left(k_1(\mathbf{x},\mathbf{x}') - 1 - \frac{1}{2}||\mathbf{x}-\mathbf{x}'||_2^2\right) \tag{13}$$

This kernel satisfies the recurrence $k - \log(k) = 1 + \frac{1}{2}||\mathbf{x}-\mathbf{x}'||_2^2$.

Properties of this kernel The solution to this recurrence has no closed form, but it is continuous and differentiable everywhere except at $\mathbf{x} = \mathbf{x}'$. Samples from a GP with this prior are not differentiable.

Conjectures: This kernel has smaller covariance than the squared-exp everywhere except at $\mathbf{x} = \mathbf{x}'$.

5 Related Work

[Ryan Adams, Wallach and Zoubin on nonparametric deep nets]

[Deep GP Kernels by Youngmin Cho] `http://cseweb.ucsd.edu/~yoc002/paper/thesis_youngmincho.pdf`

[GP Dynamical systems]

[Warping a 1d uniform distribution]

Layer-wise analysis of deep networks with Gaussian kernels: `http://books.nips.cc/papers/files/nips23/NIPS2010_0206.pdf`

Deep Gaussian Processes We introduce a generative non-parametric model to address this problem. Our approach is based on the GP-LVM [1, 2, 3], a flexible nonparametric density model.

Acknowledgments

We would like to thank Andrew McHutchon, Neil Lawrence, Josh Tenenbaum, Andreas Diamanadou, James Lloyd, and Mark van der Wilk for helpful discussions.

References

[1] N.D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. *Advances in Neural Information Processing Systems*, 16:329–336, 2004.

[2] M. Salzmann, R. Urtasun, and P. Fua. Local deformation models for monocular 3D shape recovery. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR, pages 1–8, 2008.

[3] N.D. Lawrence and R. Urtasun. Non-linear matrix factorization with Gaussian processes. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 601–608. ACM, 2009.