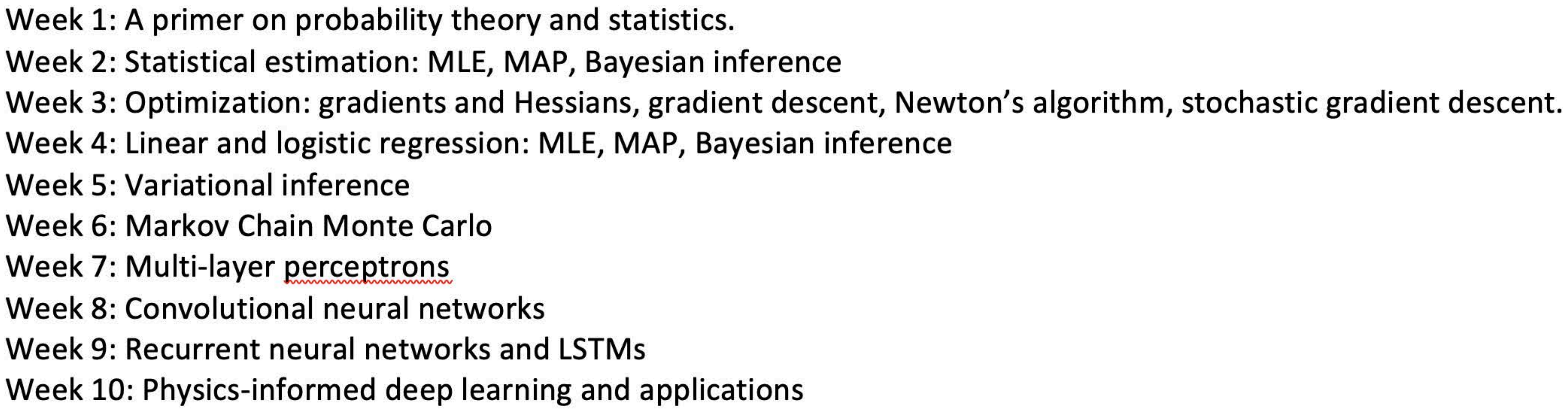
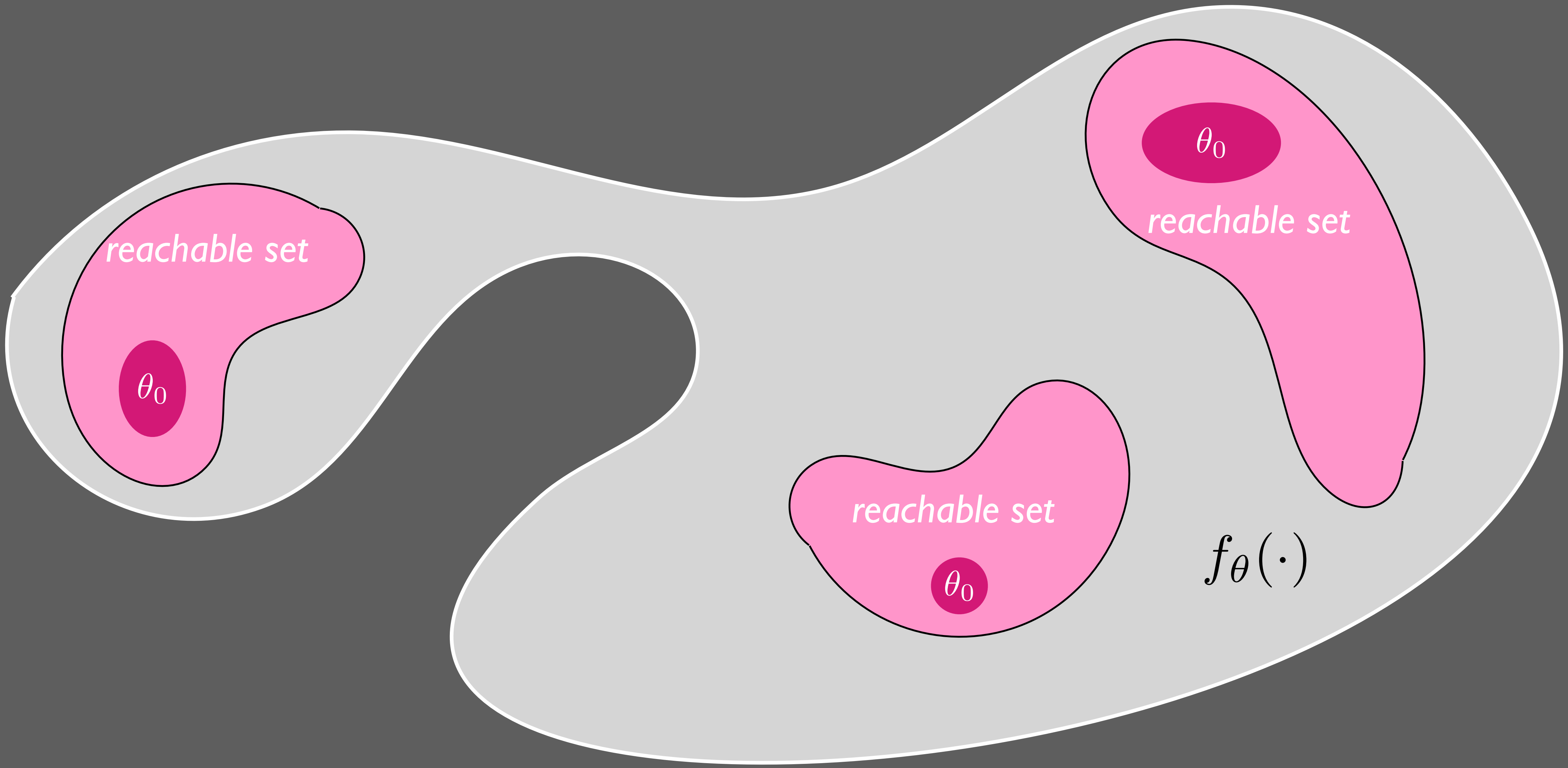
ENM 5310: Data-driven Modeling and Probabilistic Scientific Computing

Lecture #18: Inductive bias & sources of uncertainty

Recap


Week 1: A primer on probability theory and statistics.
Week 2: Statistical estimation: MLE, MAP, Bayesian inference
Week 3: Optimization: gradients and Hessians, gradient descent, Newton’s algorithm, stochastic gradient descent.
Week 4: Linear and logistic regression: MLE, MAP, Bayesian inference
Week 5: Variational inference
Week 6: Markov Chain Monte Carlo
Week 7: Multi-layer perceptrons
Week 8: Convolutional neural networks
Week 9: Recurrent neural networks and LSTMs
Week 10: Physics-informed deep learning and applications



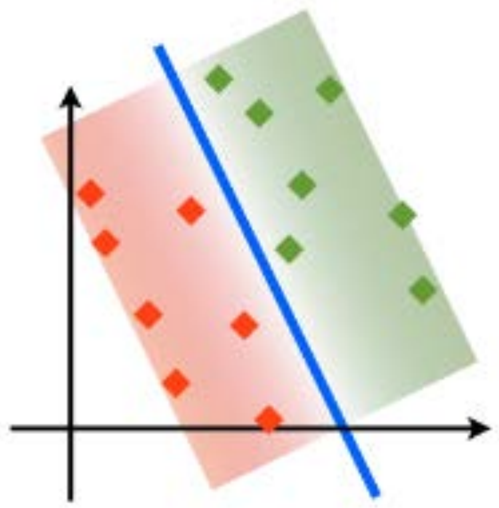
Key factors:

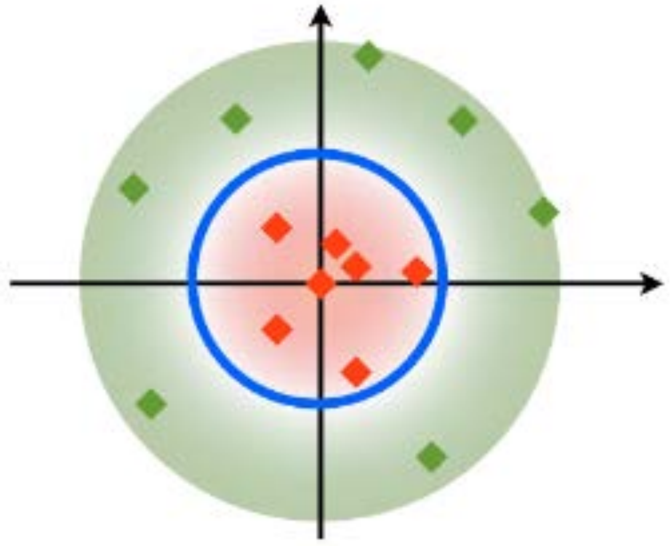
- Architecture/inductive bias
- Initialization
- Learning algorithm
- Training data

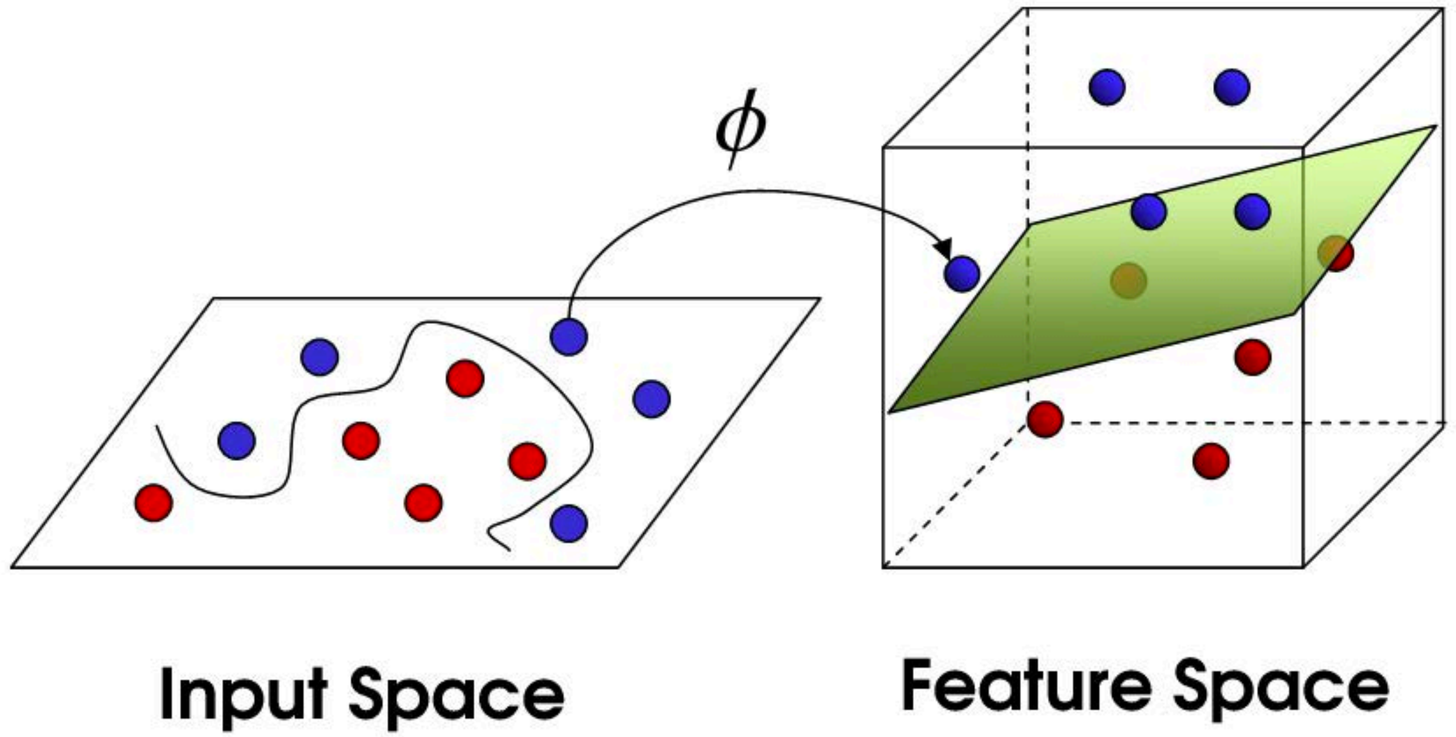
all possible functions

"Linearization" by embedding to higher dimensions

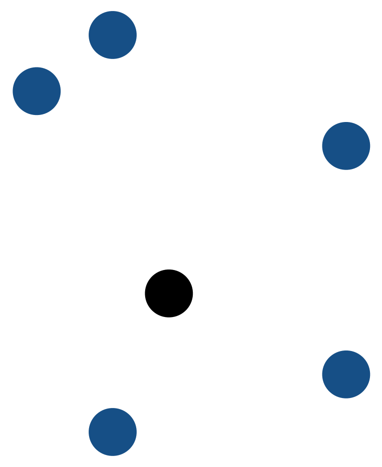
$$f(\boldsymbol{x}) = \langle \theta, \phi(\boldsymbol{x}) \rangle_{\mathcal{H}}, \quad \phi : \mathbb{R}^d \to \mathbb{R}^m$$







Kernel methods



$$f(\boldsymbol{x}) = \langle \theta, \phi(\boldsymbol{x}) \rangle_{\mathcal{H}}, \quad \phi : \mathbb{R}^d \to \mathbb{R}^m$$

$$\Big\downarrow \; m \to \infty \qquad k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle_{\mathcal{H}}$$

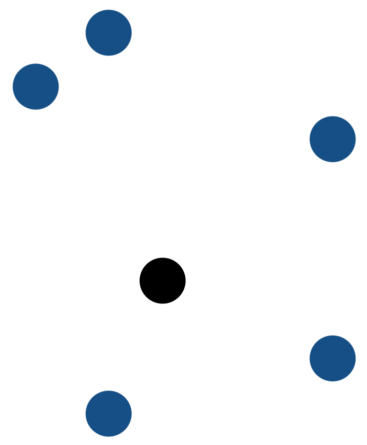
$$f(\boldsymbol{x}) = \sum_{i=1}^{n} (\boldsymbol{K}^{-1}\boldsymbol{y})_i k(\boldsymbol{x}_i, \boldsymbol{x}), \quad \boldsymbol{K}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

Key take-away points:

-Kernels and representer theorems: learning with infinite-dimensional linear models can be done in time that depend on the number of observations by using a kernel function.
-Kernels on $\mathbb{R}^d$: such models include polynomials and classical Sobolev spaces (functions with square-integrable partial derivatives).
-Algorithms: convex optimization algorithms can be applied with theoretical guarantees and many dedicated developments to avoid the quadratic complexity of computing the kernel matrix.
-Analysis of well-specified models: When the target function is in the associated function space, learning can be done with rates that are independent of dimension.
-Analysis of mis-specified models: if the target is not in the the RKHS, the curse of dimensionality cannot be avoided in the worst case situations of few existing derivatives of the target function, but the methods are adaptive to any amount of intermediate smoothness.
-Sharp analysis of ridge regression: for the square loss, a more involded analysis leads to optimal rates in a variety of situations in $\mathbb{R}^d$.

https://www.di.ens.fr/~fbach/ltfp_book.pdf

Kernel methods



$$f(\boldsymbol{x}) = \langle \theta, \phi(\boldsymbol{x}) \rangle_{\mathcal{H}}, \quad \phi : \mathbb{R}^d \to \mathbb{R}^m$$

$$m \to \infty \quad \Big\downarrow \quad k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle_{\mathcal{H}}$$

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} (\boldsymbol{K}^{-1}\boldsymbol{y})_i k(\boldsymbol{x}_i, \boldsymbol{x}), \quad \boldsymbol{K}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

Why is this relevant? The study of infinite-dimensional linear methods is important for several reasons:

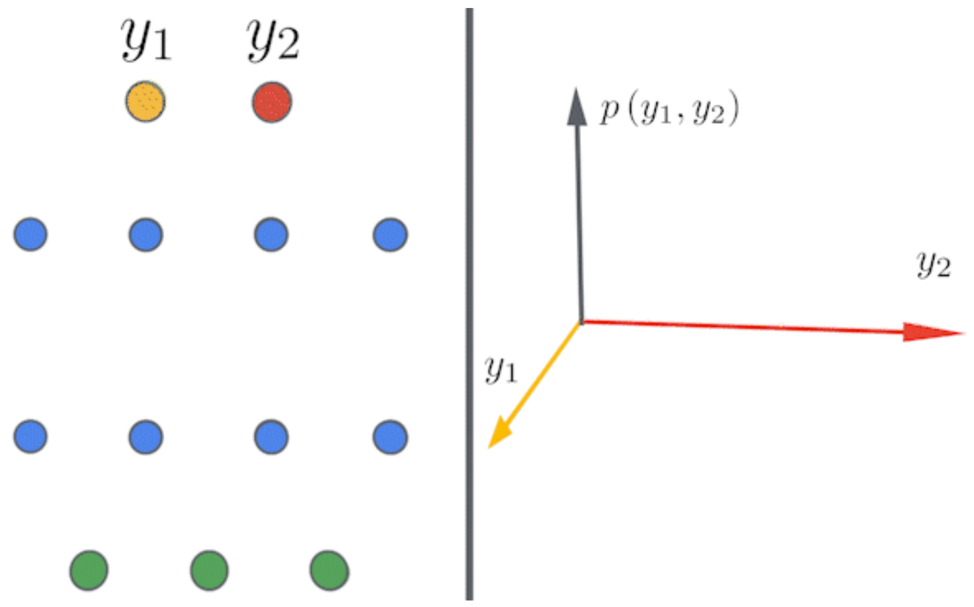
- Understanding linear models in finite but very large input dimensions requires tools from infinite-dimensional analysis.
- Kernel methods lead to simple and stable algorithms, with theoretical guarantees, and adaptivity to smoothness of the target functon (as opposed to local averaging techniques). They can be applied in high dimensions, with good practical performance (note that for supervised learning problems with many observations in domains such as computer vision and natural language processing, they do not achieve the state of the art anymore, which is achieved by neural networks presented in Chapter 9).
- They can be easily applied when input observations are not vectors.
- They are useful to understand other models such as neural networks (see Chapter 9).

https://www.di.ens.fr/~fbach/ltfp_book.pdf

Infinitely wide neural networks are kernel machines

Key concepts:

- Explore the connection between deep neural networks and kernel regression methods to elucidate the training dynamics of deep learning models.
- At initialization, fully-connected networks are equivalent to Gaussian processes in the infinite-width limit.
- The evolution of an infinite-width network during training can also be described by another kernel, the so-called Neural Tangent Kernel (NTK).
- As its width tends to infinity, a deep neural network's behavior under gradient descent can become simplified and predictable, characterized by its limiting NTK.

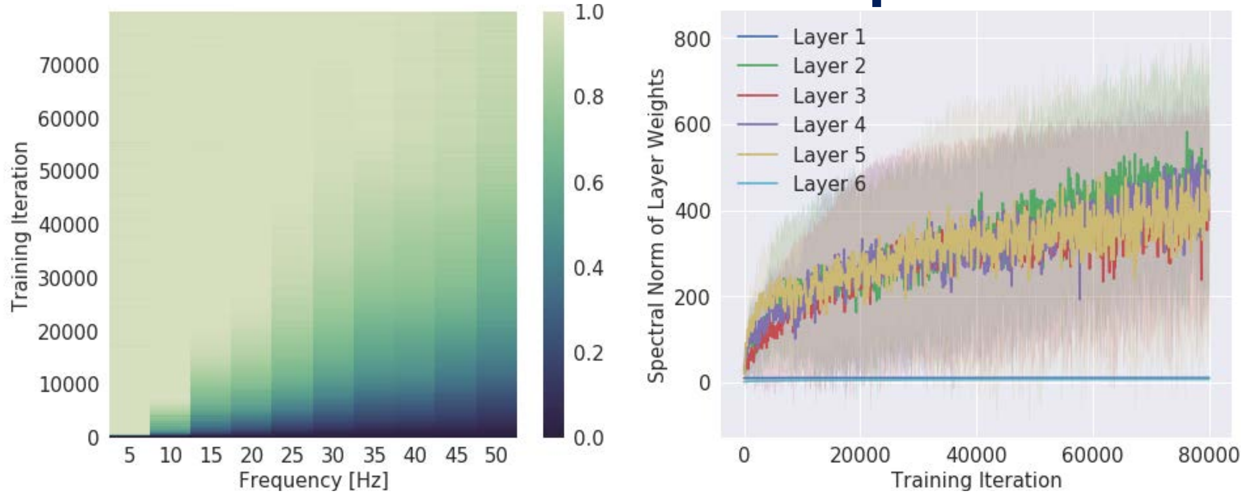


$$k_\theta(x, x') = \left\langle \frac{df_\theta(x)}{d\theta}, \frac{f_\theta(x')}{d\theta} \right\rangle$$

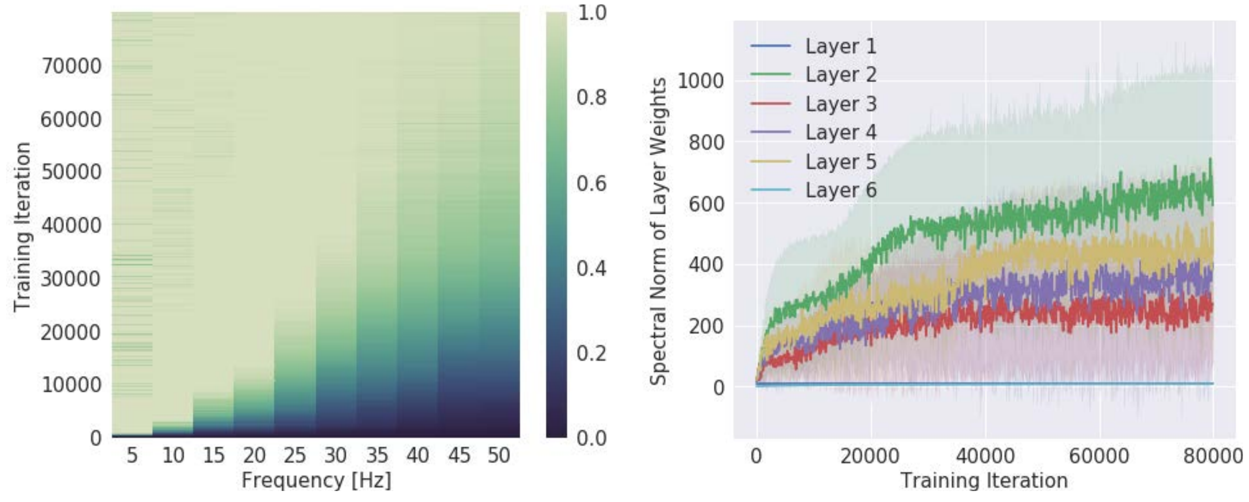
Intuition: $k(x,x')$ measures how sensitive the function value at x is to prediction errors at x′.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., & Sohl-Dickstein, J. (2017). Deep neural networks as Gaussian processes.
Jacot, A., Gabriel, F., & Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. In Advances in neural information processing systems (pp. 8571-8580).
Shin, Y., Darbon, J., & Karniadakis, G. E. (2020). On the convergence and generalization of physics informed neural networks. arXiv preprint arXiv:2004.01806.

Spectral bias in MLPs

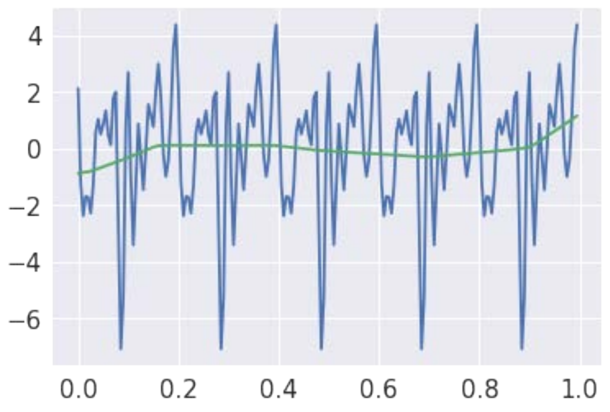


(a) Equal Amplitudes

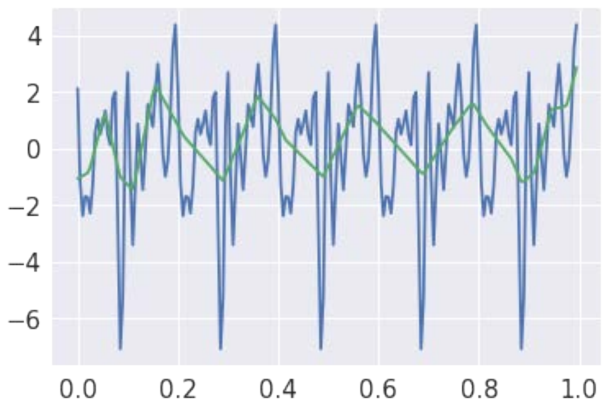


(b) Increasing Amplitudes

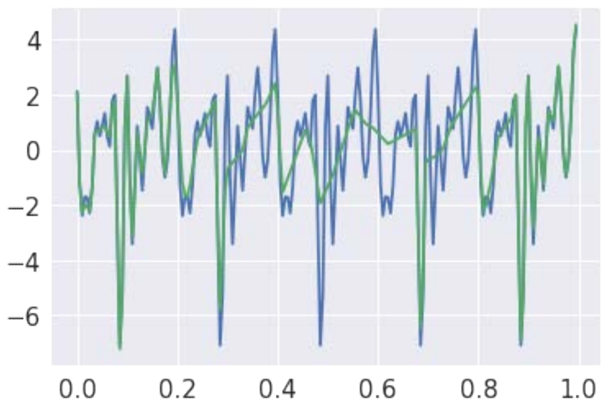
Figure 1. Left (a, b): Evolution of the spectrum (x-axis for frequency) during training (y-axis). The colors show the measured amplitude of the network spectrum at the corresponding frequency, normalized by the target amplitude at the same frequency (i.e. $|\tilde{f}_{k_i}|/A_i$) and the colorbar is clipped between 0 and 1. Right (a, b): Evolution of the spectral norm (y-axis) of each layer during training (x-axis). Figure-set (a) shows the setting where all frequency components in the target function have the same amplitude, and (b) where higher frequencies have larger amplitudes. **Gist**: We find that even when higher frequencies have larger amplitudes, the model prioritizes learning lower frequencies first. We also find that the spectral norm of weights increases as the model fits higher frequency, which is what we expect from Theorem 1.



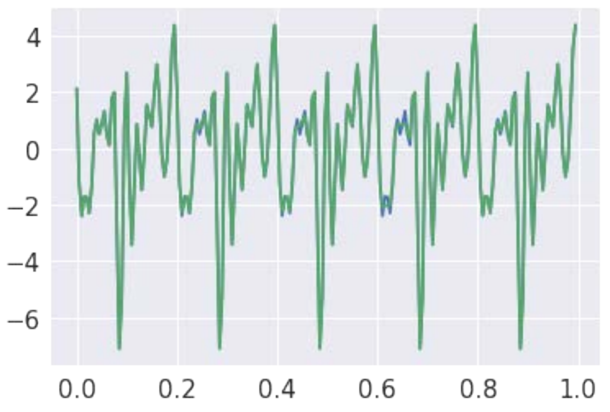
(a) Iteration 100



(b) Iteration 1000



(c) Iteration 10000



(d) Iteration 80000

Figure 2. The learnt function (green) overlayed on the target function (blue) as the training progresses. The target function is a superposition of sinusoids of frequencies $\kappa = (5, 10, ..., 45, 50)$, equal amplitudes and randomly sampled phases.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y. and Courville, A., 2019, May. On the spectral bias of neural networks. In International Conference on Machine Learning (pp. 5301-5310). PMLR.

Spectral bias in deep learning

- Since the kernel K is positive semi-definite, we can take its spectral decomposition: $\boldsymbol{K} = \boldsymbol{Q}^T \boldsymbol{\Lambda} \boldsymbol{Q}$
- This allows us to decompose the training error into the eigen-space of the NTK as:

$$\boldsymbol{Q}^T (f(\boldsymbol{X}_{\text{train}}, \boldsymbol{\theta}(t)) - \boldsymbol{Y}_{\text{train}}) = -e^{\boldsymbol{\Lambda} t} \boldsymbol{Q}^T \boldsymbol{Y}_{\text{train}}$$

$$\Downarrow$$

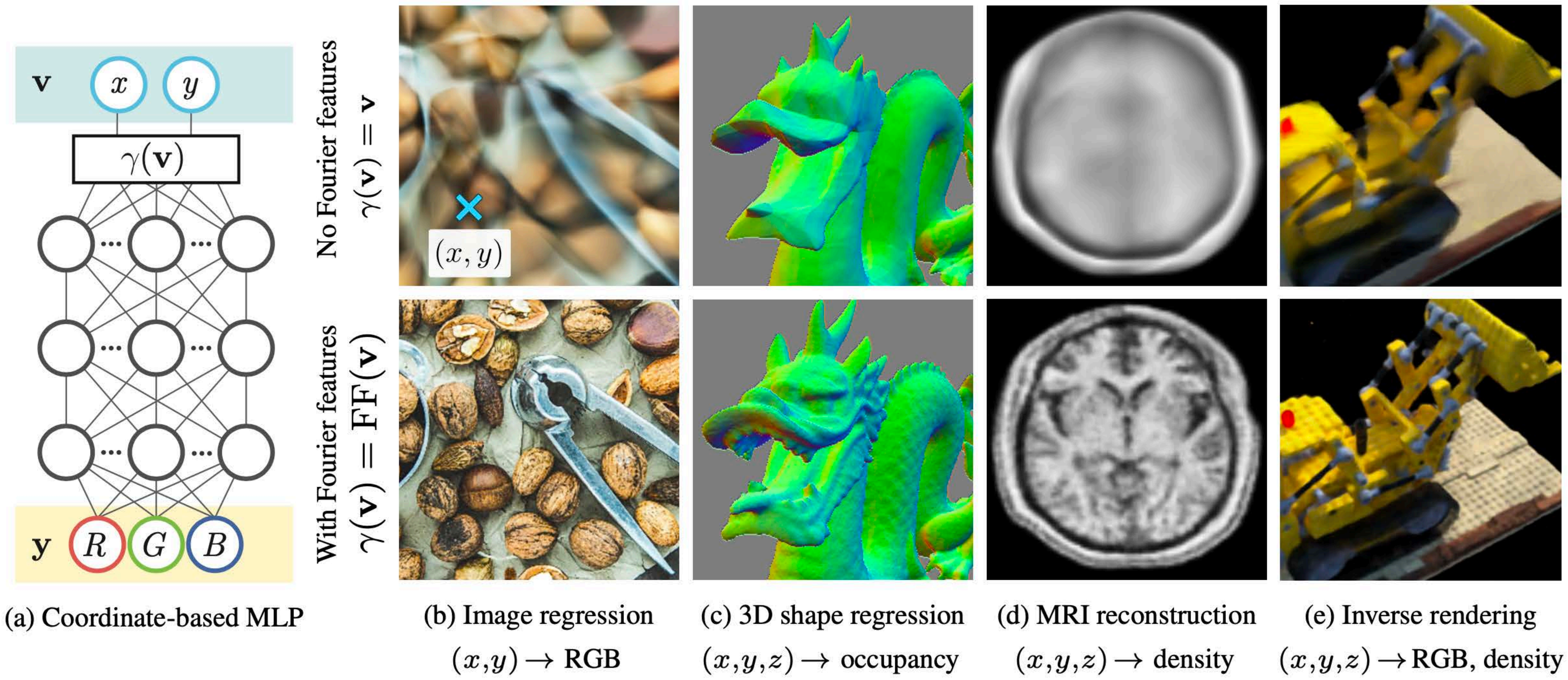
$$\begin{aligned} f(\boldsymbol{X}_{\text{train}}, \boldsymbol{\theta}(t)) - \boldsymbol{Y}_{\text{train}} &= \sum_{i=1}^{N} \boldsymbol{q}_i^T \left(f(\boldsymbol{X}_{\text{train}}, \boldsymbol{\theta}(t)) - \boldsymbol{Y}_{\text{train}} \right) \boldsymbol{q}_i \\ &= \sum_{i=1}^{N} \left(e^{-\lambda_i t} \boldsymbol{q}_i^T \boldsymbol{Y}_{\text{train}} \right) \boldsymbol{q}_i \end{aligned}$$

Conclusion: Neural networks will first learn the target function along the eigen-directions of their neural tangent kernel with larger eigenvalues, and then learn the remaining components corresponding to smaller eigenvalues.

- *The eigenvalues of the NTK characterize how fast the absolute training error decreases.*
- *Components of the target function that correspond to NTK eigenvectors with larger eigenvalues will be learned faster.*
- *For fully-connected networks, the eigenvectors corresponding to higher eigenvalues of the NTK matrix generally exhibit lower frequencies.*
- *In practice, we observe that the eigenvalues of the NTK decay rapidly. This results in extremely slow convergence to the high-frequency components of the target function.*

Wang, S., Wang, H., & Perdikaris, P. (2020). On the eigenvector bias of Fourier feature networks: From regression to solving multi-scale PDEs with physics-informed neural networks. arXiv preprint arXiv:2012.10047.

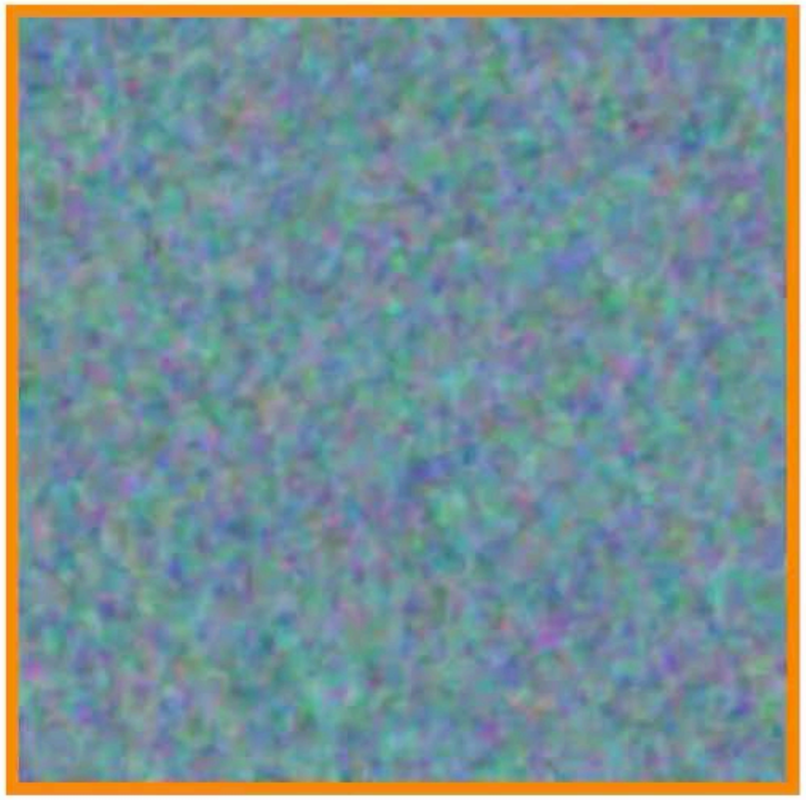
Fourier feature networks

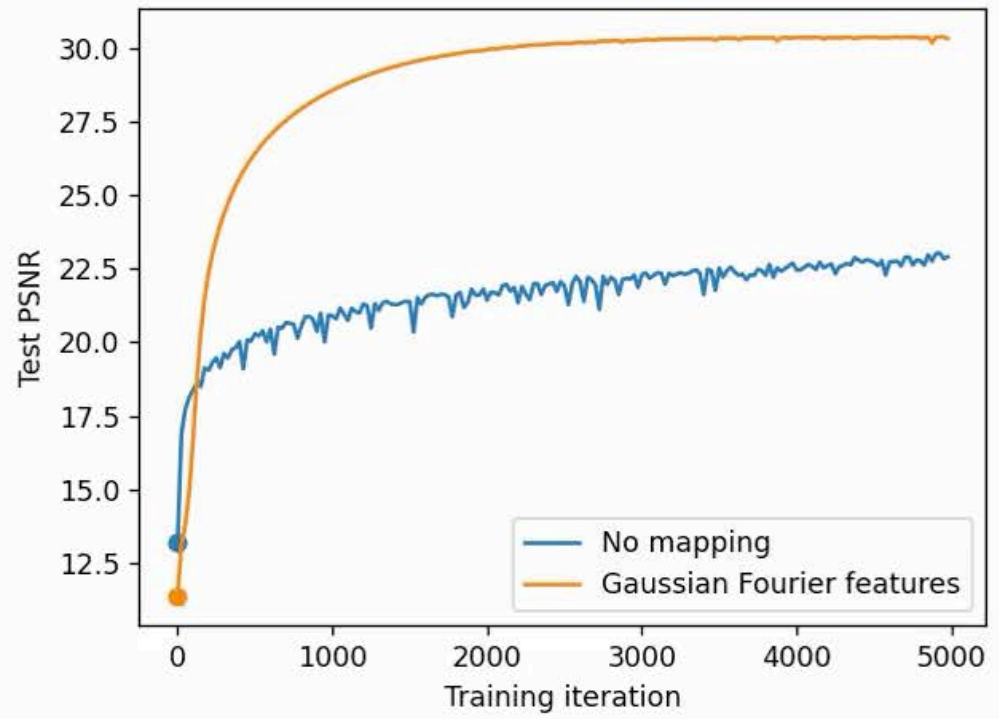


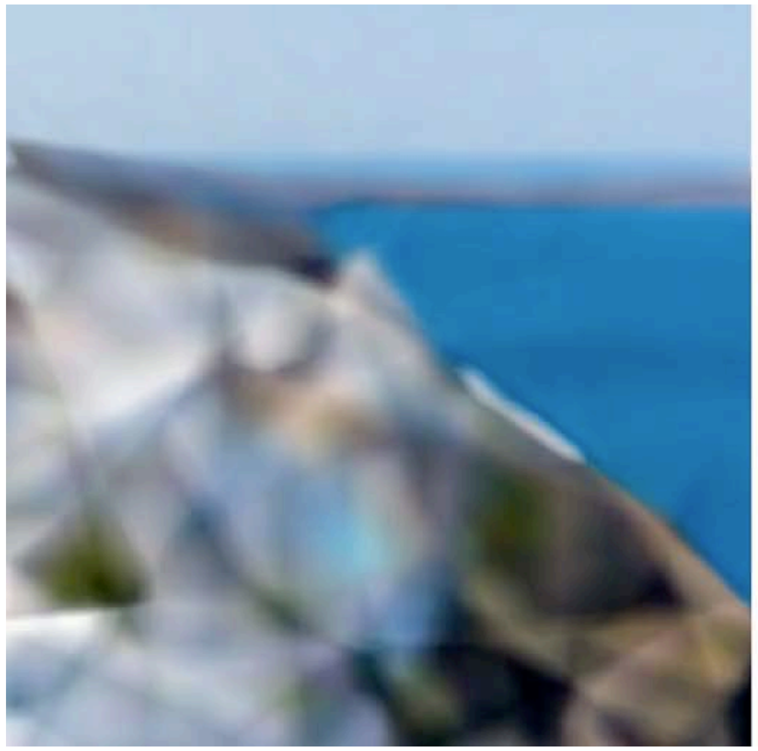
$$\gamma(\mathbf{v}) = [\cos(2\pi\mathbf{B}\mathbf{v}), \sin(2\pi\mathbf{B}\mathbf{v})]^{\mathrm{T}}$$

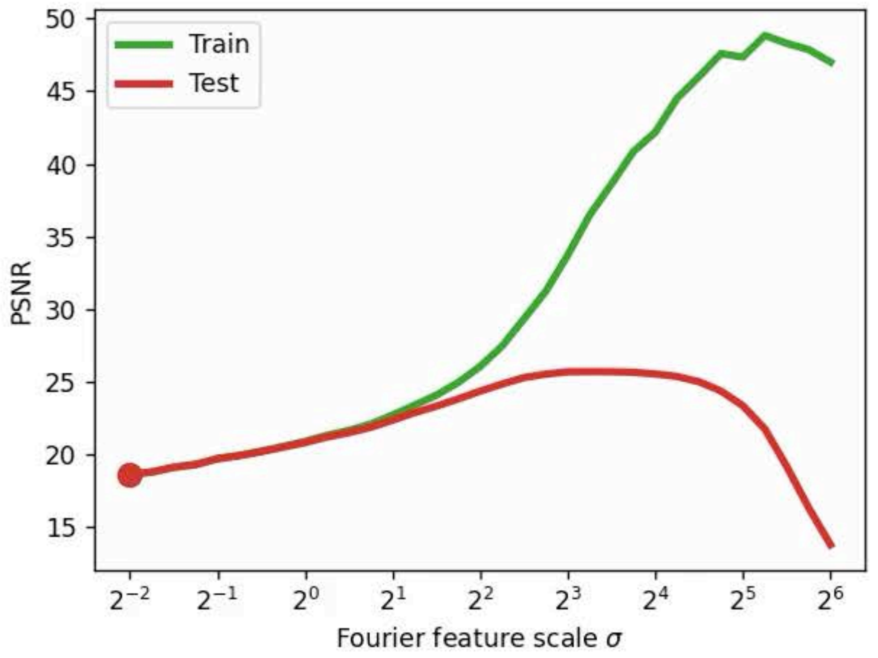
Tancik, M., Srinivasan, P.P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J.T. and Ng, R., 2020. Fourier features let networks learn high frequency functions in low dimensional domains. arXiv preprint arXiv:2006.10739.

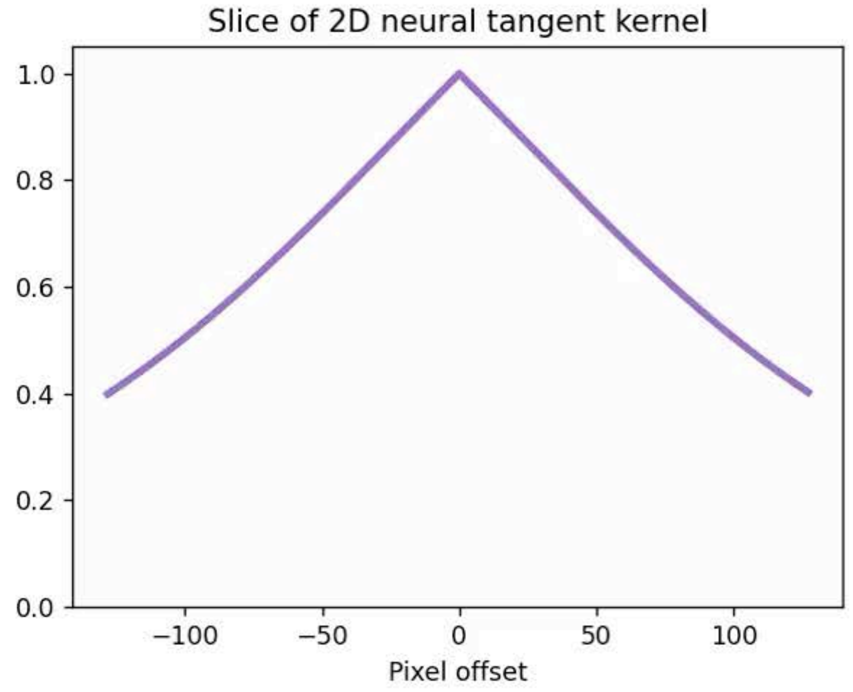
Fourier feature networks





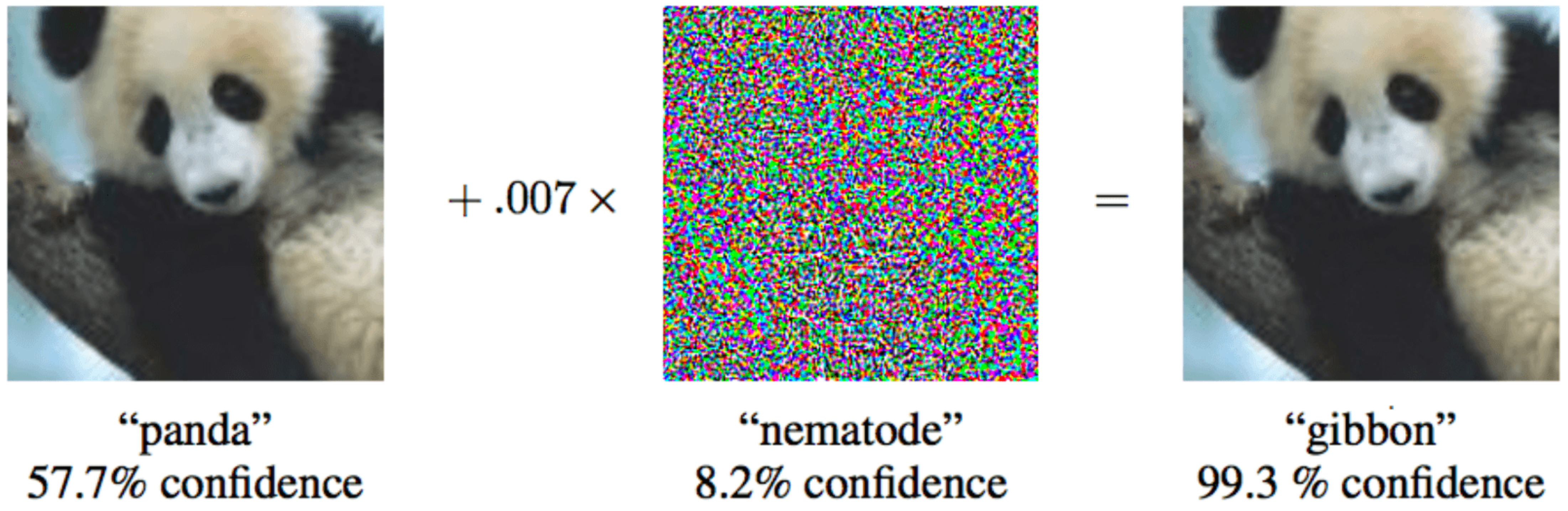




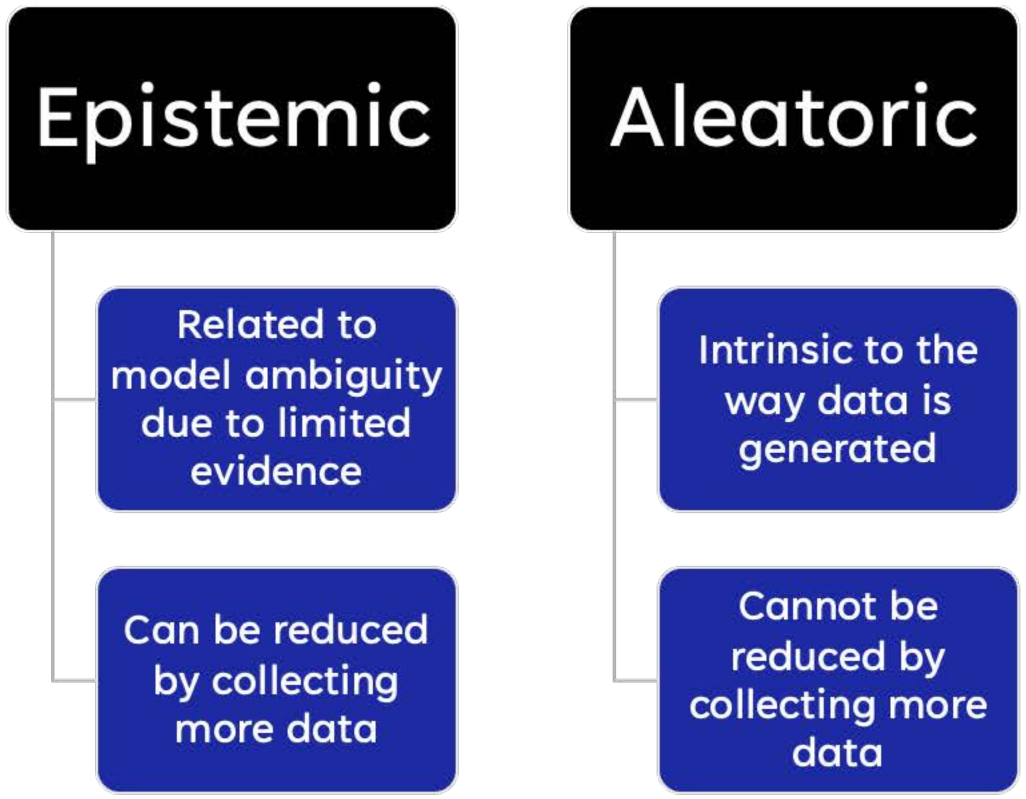


Tancik, M., Srinivasan, P.P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J.T. and Ng, R., 2020. Fourier features let networks learn high frequency functions in low dimensional domains. arXiv preprint arXiv:2006.10739.

A need for robustness and uncertainty quantification



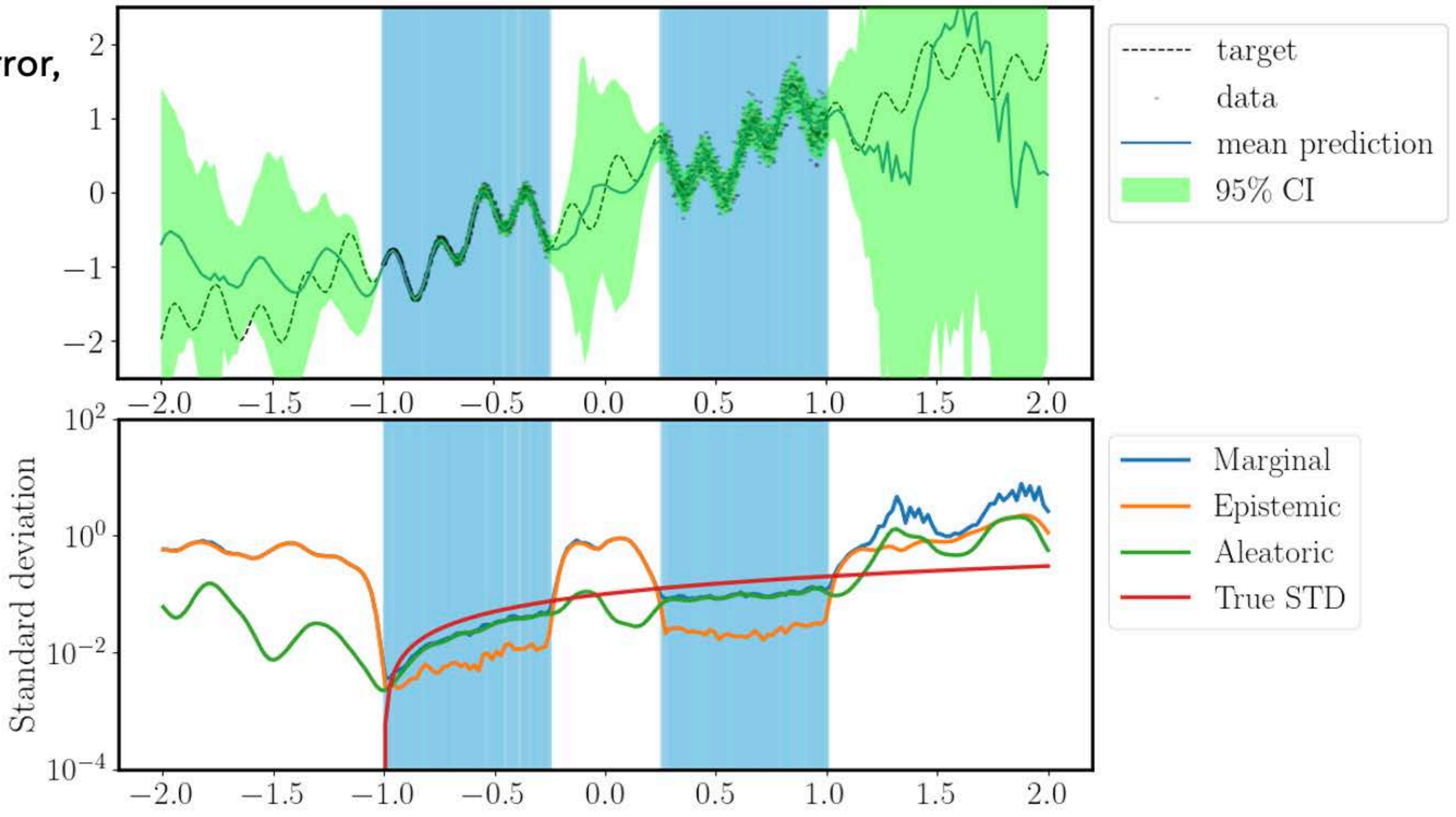
Epistemic vs Aleatoric Uncertainty


Epistemic
Related to model ambiguity due to limited evidence
Can be reduced by collecting more data
Aleatoric
Intrinsic to the way data is generated
Cannot be reduced by collecting more data

Epistemic vs Aleatoric Uncertainty

A More Realistic Scenario

High Epistemic Error,
Heteroscedastic
Aleatoric Error



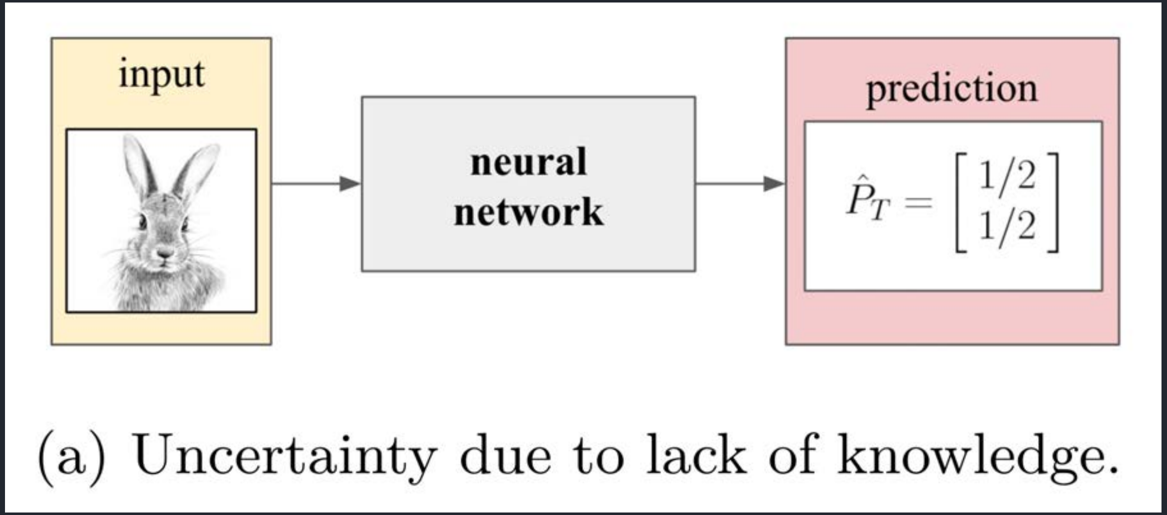
Distinguishing between Aleatory and Epistemic Uncertainty

Neural networks don't know what they don't know.

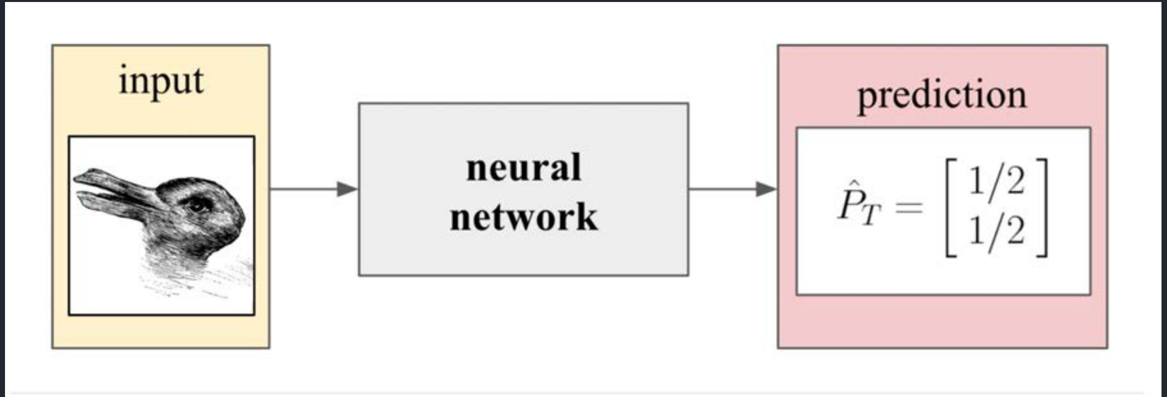
(a) Insufficient training data = epistemic uncertainty

(b) Ambiguous data = aleatory uncertainty

Conventional neural networks cannot distinguish between (a) and (b)



(a) Uncertainty due to lack of knowledge.



(b) Uncertainty due to ambiguous image.

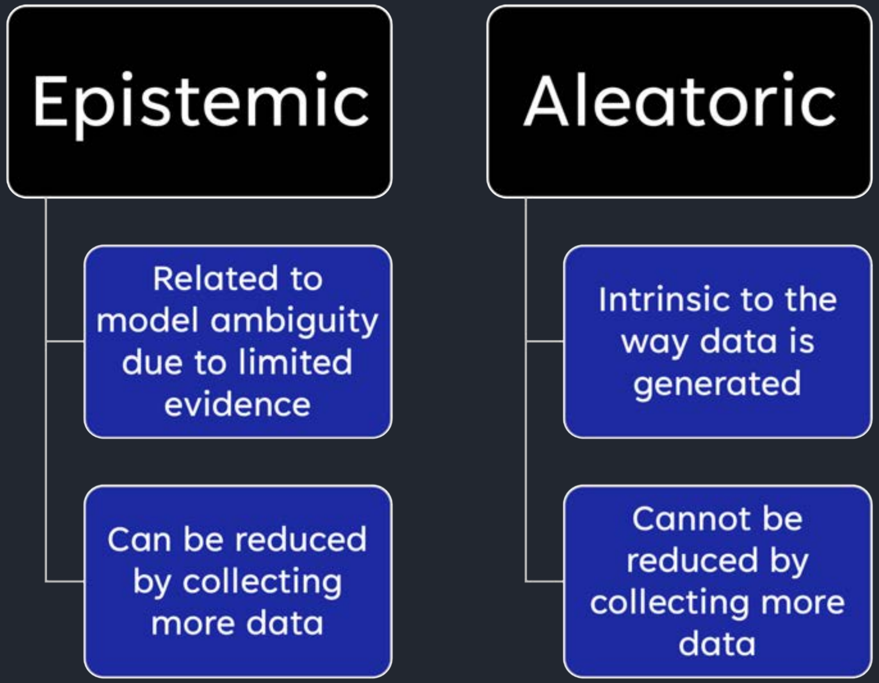
A Taxonomy of UQ Tasks

Aleatory

$$\mathcal{D} = \{x_1, x_2, \dots, x_n\} \mapsto p(x)$$

$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\} \mapsto p(y|x)$$

$$\mathcal{M} : \mathcal{X} \to \mathcal{Y}, \quad x \sim p(x) \mapsto p(\mathcal{M}(x)|x)$$


Epistemic
Related to model ambiguity due to limited evidence
Can be reduced by collecting more data
Aleatoric
Intrinsic to the way data is generated
Cannot be reduced by collecting more data

Epistemic

$$\mathcal{M}_\theta : \mathcal{X} \to \mathcal{Y}, \quad \mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\} \mapsto p(\theta|\mathcal{D})$$

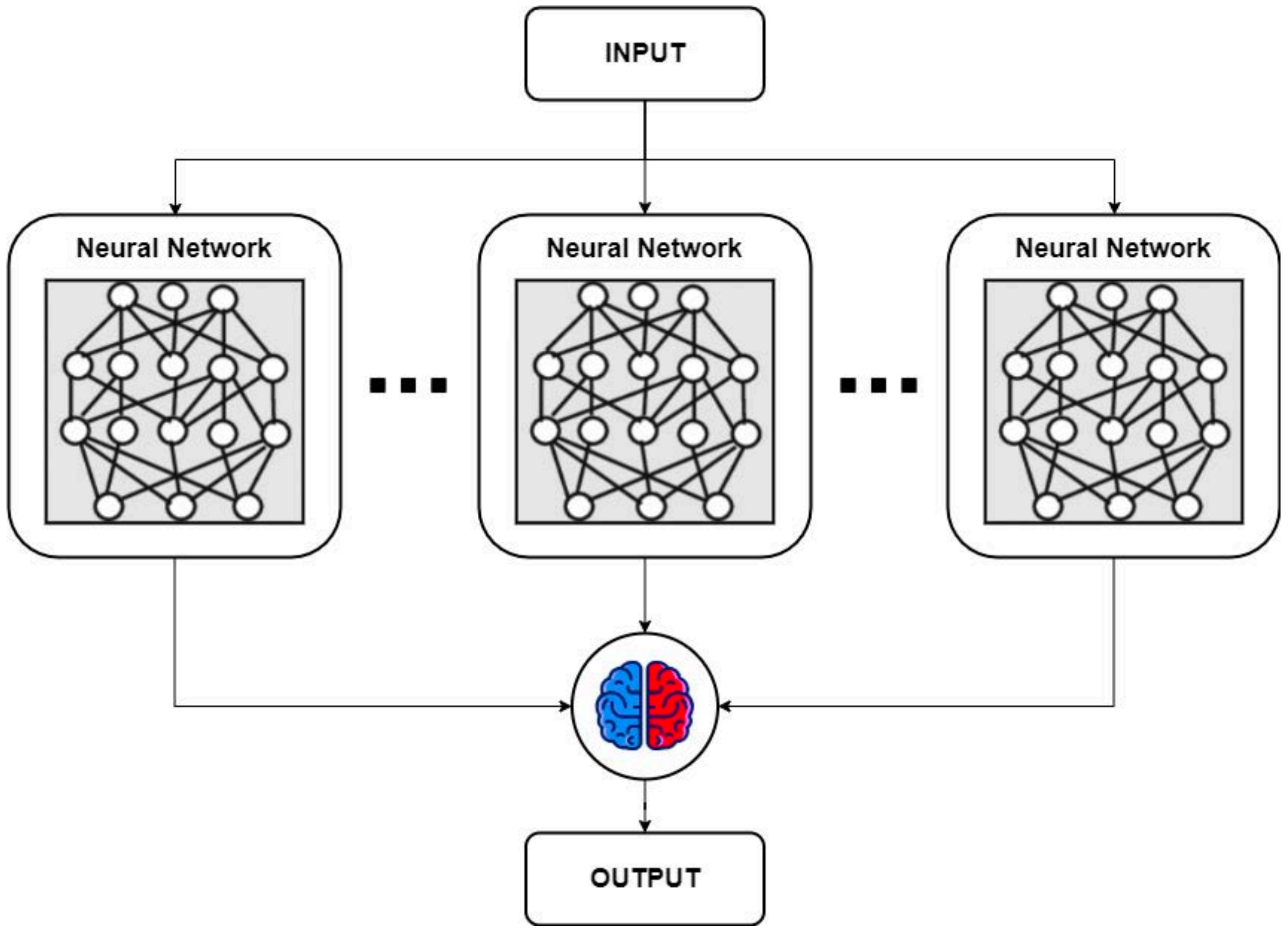
$$\mathcal{M}_\theta : \mathcal{X} \to \mathcal{Y}, \quad \mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\} \mapsto p(\mathcal{M}_\theta(x^*)|\mathcal{D}, x^*)$$

$$= \int p(\mathcal{M}_\theta|\mathcal{D}, x^*, \theta)\; p(\theta|\mathcal{D})\; d\theta$$

A need for robustness and uncertainty quantification

Becomes particularly important when:

- We are working with small data-sets (over-fitting regime).
- We need to make high-consequence decisions.
- We require performance/accuracy guarantees.
- We work under a limited budget.



$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int p(\mathcal{D}|\theta)p(\theta)d\theta}$$

The frequentist approach:
Ensemble averaging

The Bayesian approach:
Probabilistic programming