(D)GPs ⟷ DNNs

Andreas Damianou

Deep structures workshop, 20 Dec. 2019

With: Wesley Maddox, Shuai Tang, Andrew Wilson, Pablo Garcia Moreno

Different aspects of applied machine learning

Model Construction

- Capacity/Expressiveness
- Interpretability
- Encoding assumptions
- Uncertainty

Different aspects of applied machine learning

Model Construction

- Capacity/Expressiveness
- Interpretability
- Encoding assumptions
- Uncertainty

GP dominates

Learnability of the model

- Hierarchical concept learning (MacKay 2002)
- Algorithms: SGD, parallelization
- Numerical stability
- Data size needed/supported

DNN dominates

Different aspects of applied machine learning

Model Construction

- Capacity/Expressiveness
- Interpretability
- Encoding assumptions
- Uncertainty

Learnability of the model

- Hierarchical concept learning (MacKay 2002)
- Algorithms: SGD, parallelization

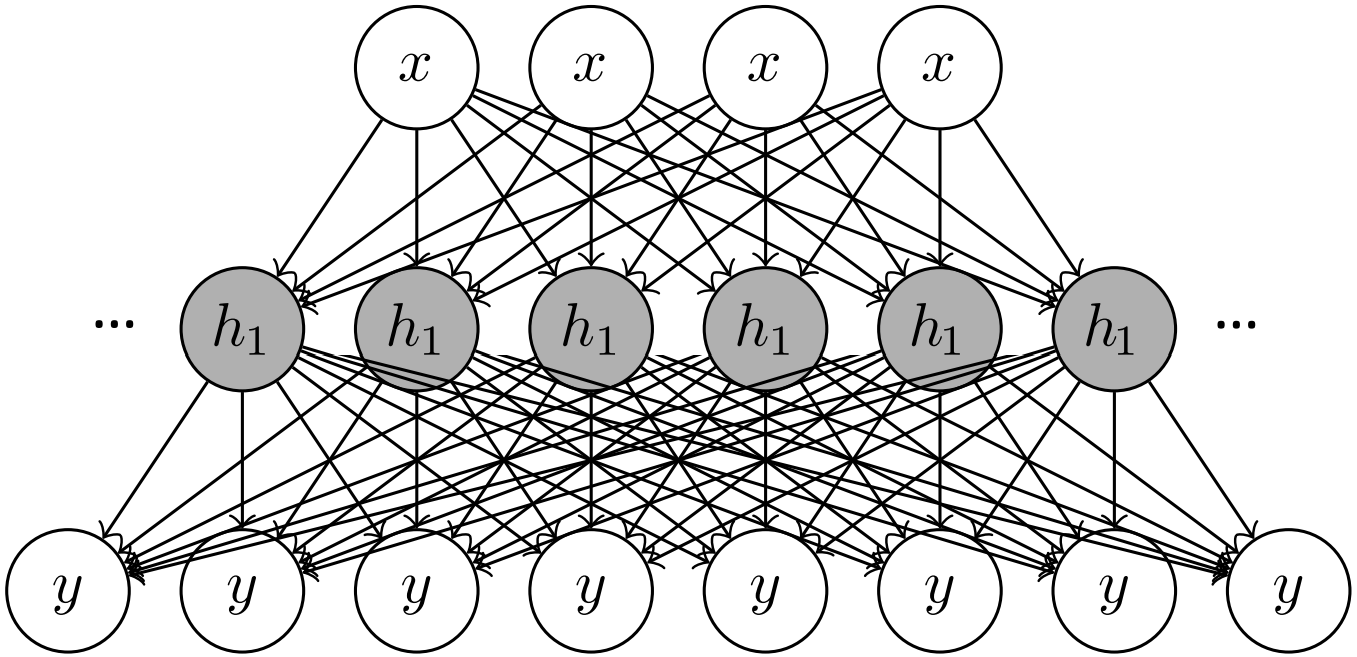
Outline

- (D)NN limits to GPs
- Deep GPs
- *Trained* DNN inductive biases in (degenerate) GPs
- … and rapid adaptation to new tasks

Part 1/3: Limit properties

Single layer, infinite width NN → GP

Also assuming Gaussian i.i.d noise for weights and biases



GP

Neal 1994; Williams 1997

Conditions for limit results

- Independence (i.i.d. noise)
- Bounded variance
- #nodes → Inf

Then use multivariate Central Limit Theorem.

Analytic covariance function (1 layer)

Single layer network:

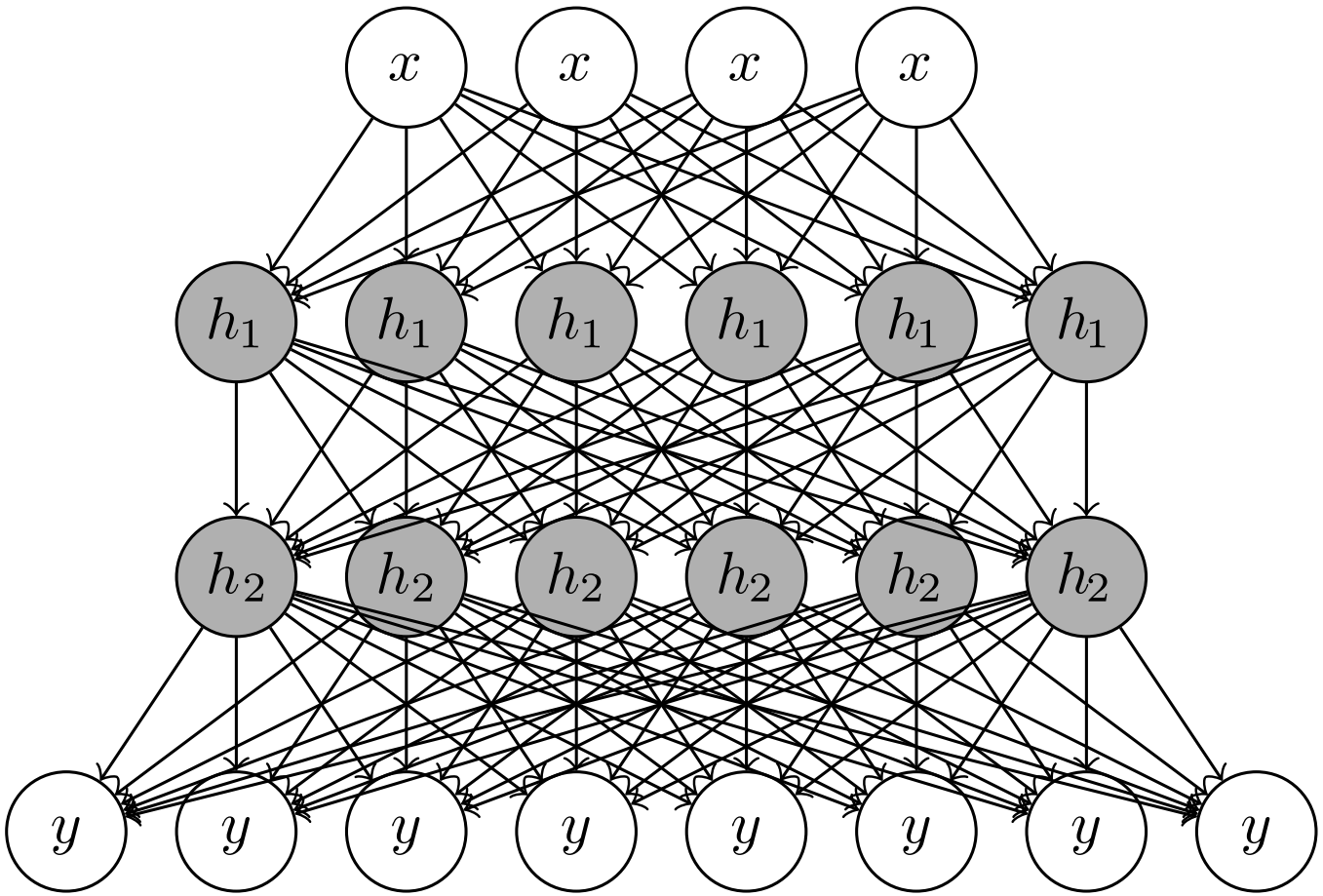
$$z_i^l(x) = b_i^l + \sum_{j=1}^{N_l} W_{ij}^l x_j^l(x), \quad x_j^l(x) = \phi(z_j^{l-1}(x)).$$

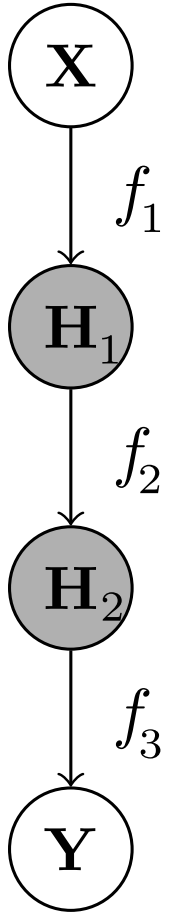
Corresponding GP cov. function:

$$K^1(x, x') \equiv \mathbb{E}\left[z_i^1(x) z_i^1(x')\right] = \sigma_b^2 + \sigma_w^2 \, \mathbb{E}\left[x_i^1(x) x_i^1(x')\right] \equiv \sigma_b^2 + \sigma_w^2 C(x, x').$$

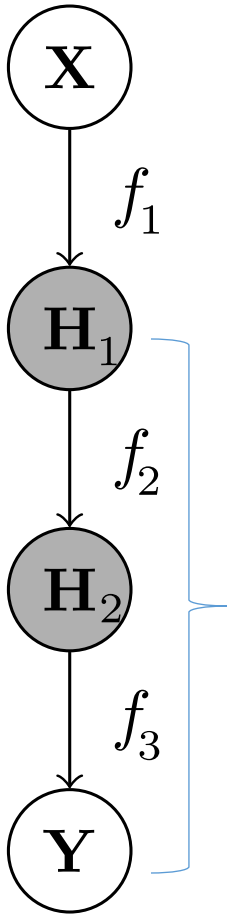
Lee et al. 2018, Matthews et al. 2018

Extension to deep models





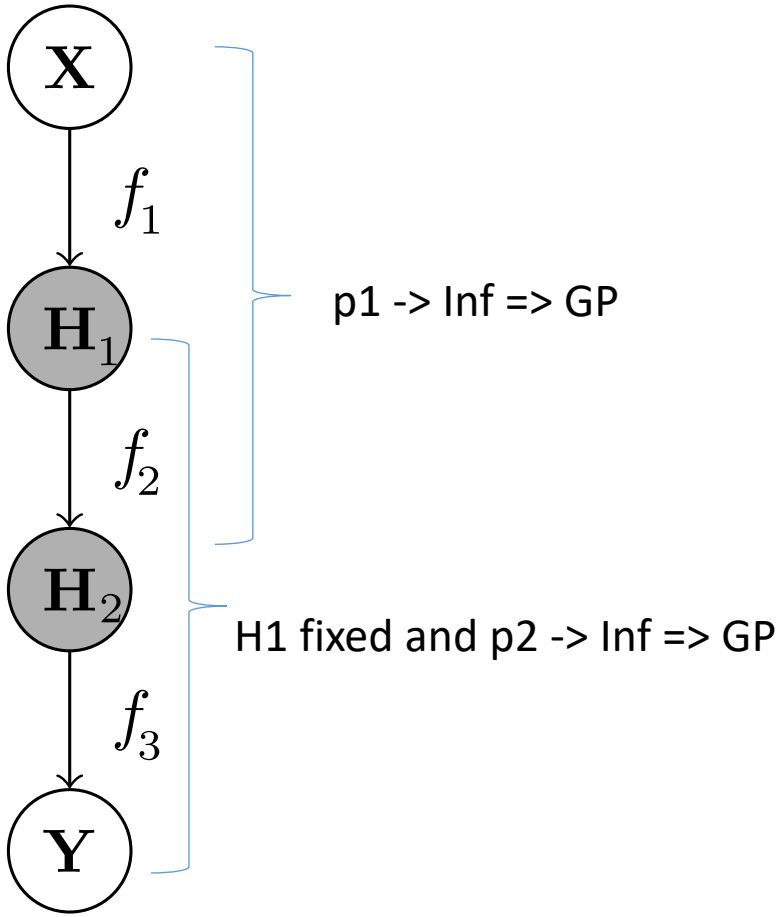
How to generalize the argument for >1 layers?



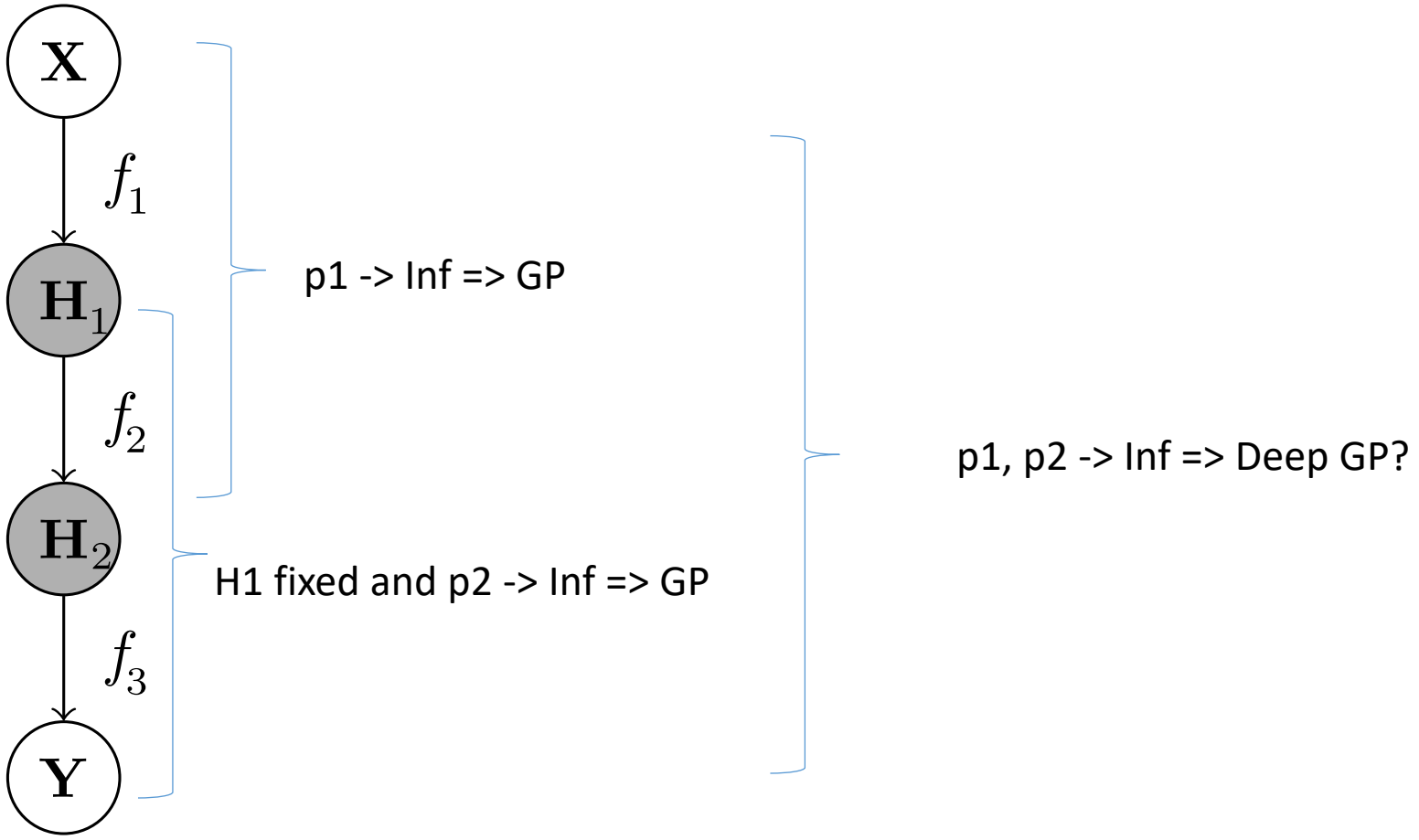
H1 fixed and p2 -> Inf => GP

(Neal 1994)

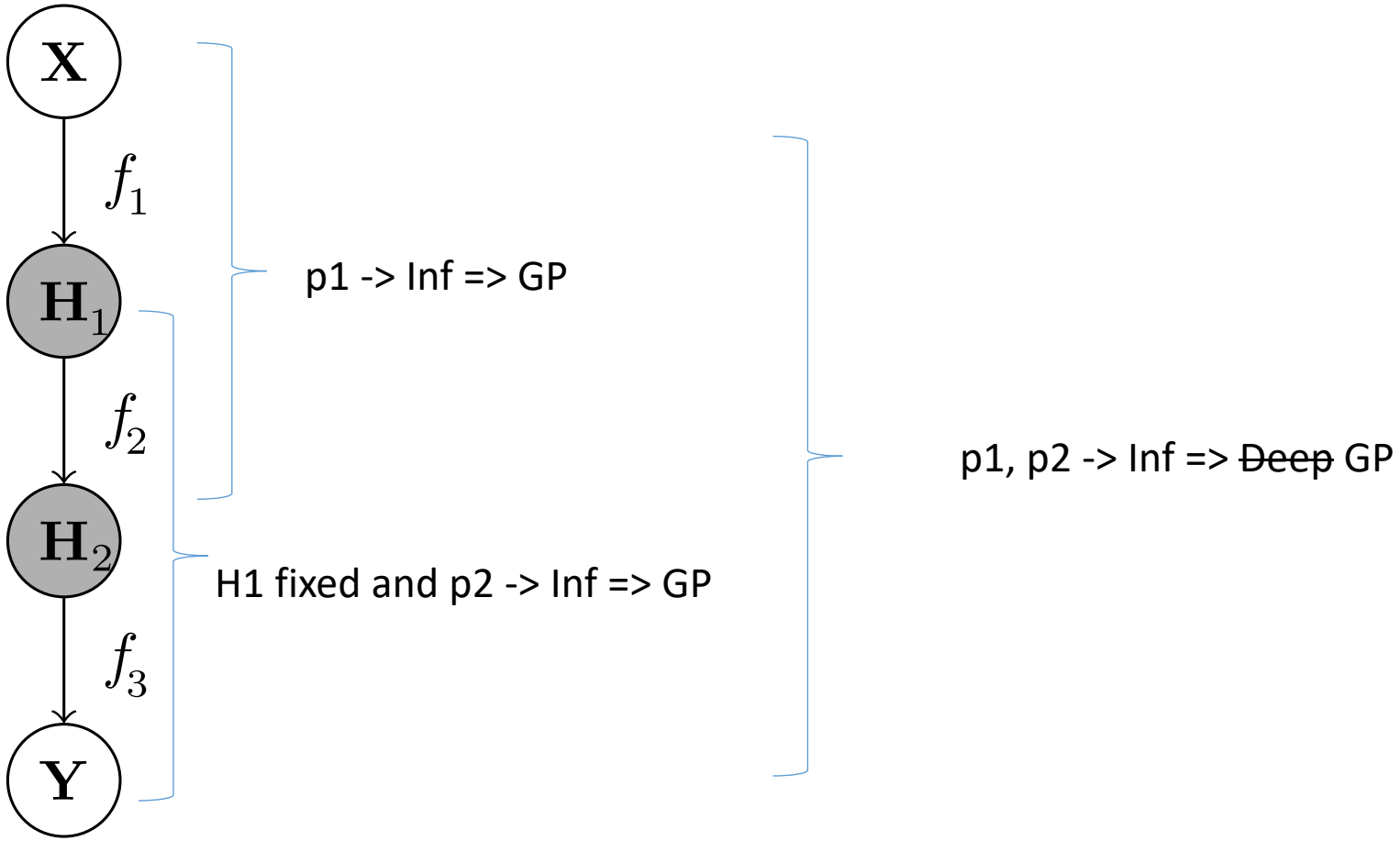
How to generalize the argument for >1 layers?



How to generalize the argument for >1 layers?



How to generalize the argument for >1 layers?



Lee et al., Matthews et al.

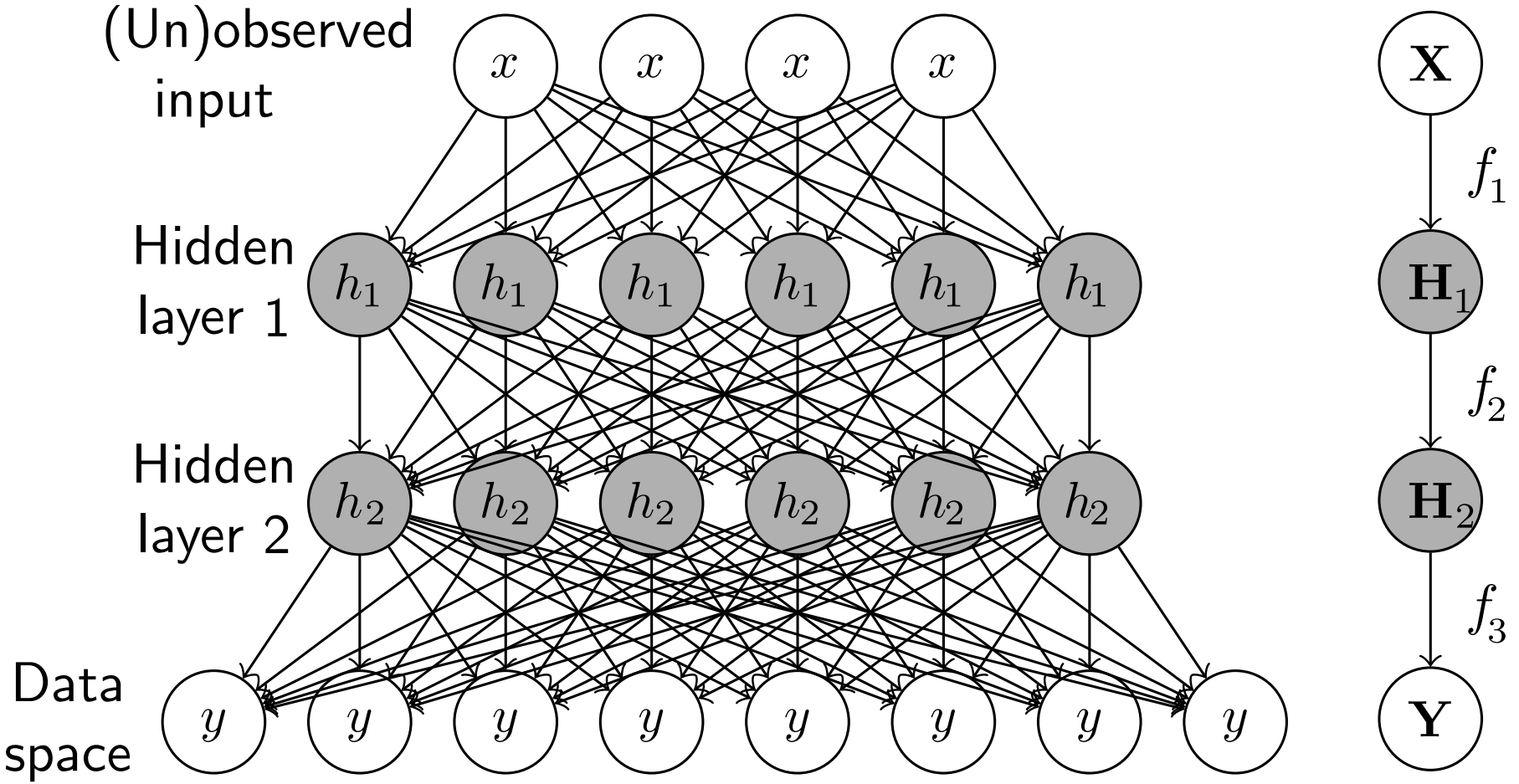
Recursive cov. function formulation

$$K^l(x,x') \equiv \mathbb{E}\left[z_i^l(x) z_i^l(x')\right]$$

$$= \sigma_b^2 + \sigma_w^2 \, \mathbb{E}_{z_i^{l-1} \sim \mathcal{GP}(0, K^{l-1})}\left[\phi(z_i^{l-1}(x))\phi(z_i^{l-1}(x'))\right]$$

$$= \sigma_b^2 + \sigma_w^2 \, F_\phi\Big(K^{l-1}(x,x'),\, K^{l-1}(x,x),\, K^{l-1}(x',x')\Big)$$

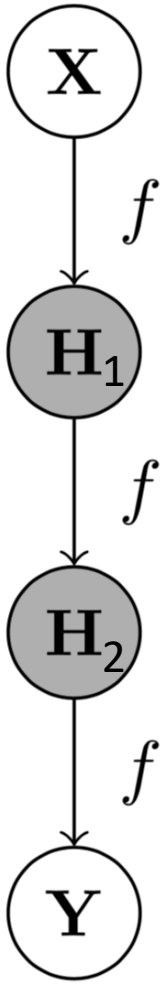
DGP: Function composition



$$\mathbf{Y} = f_3(f_2(\cdots f_1(\mathbf{X}))), \qquad \mathbf{H}_i = f_i(\mathbf{H}_{i-1})$$

Part 2/3: Deep GPs

Deep Gaussian process



- Objective: $p(y|x) = \int_{h_2} \left(p(y|h_2) \int_{h_1} p(h_2|h_1) p(h_1|x) \right)$
- $p(h_2|x) = \int_{h_1, f_2} p(h_2|f_2) p(f_2|h_1) \; p(h_1|x)$

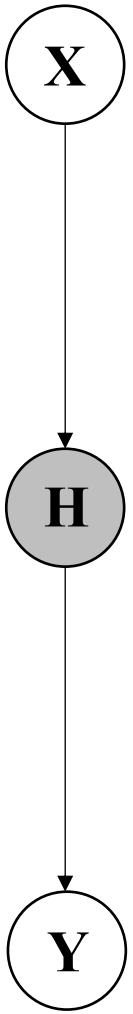
Damianou & Lawrence, 2013, Damianou, PhD Thesis 2015

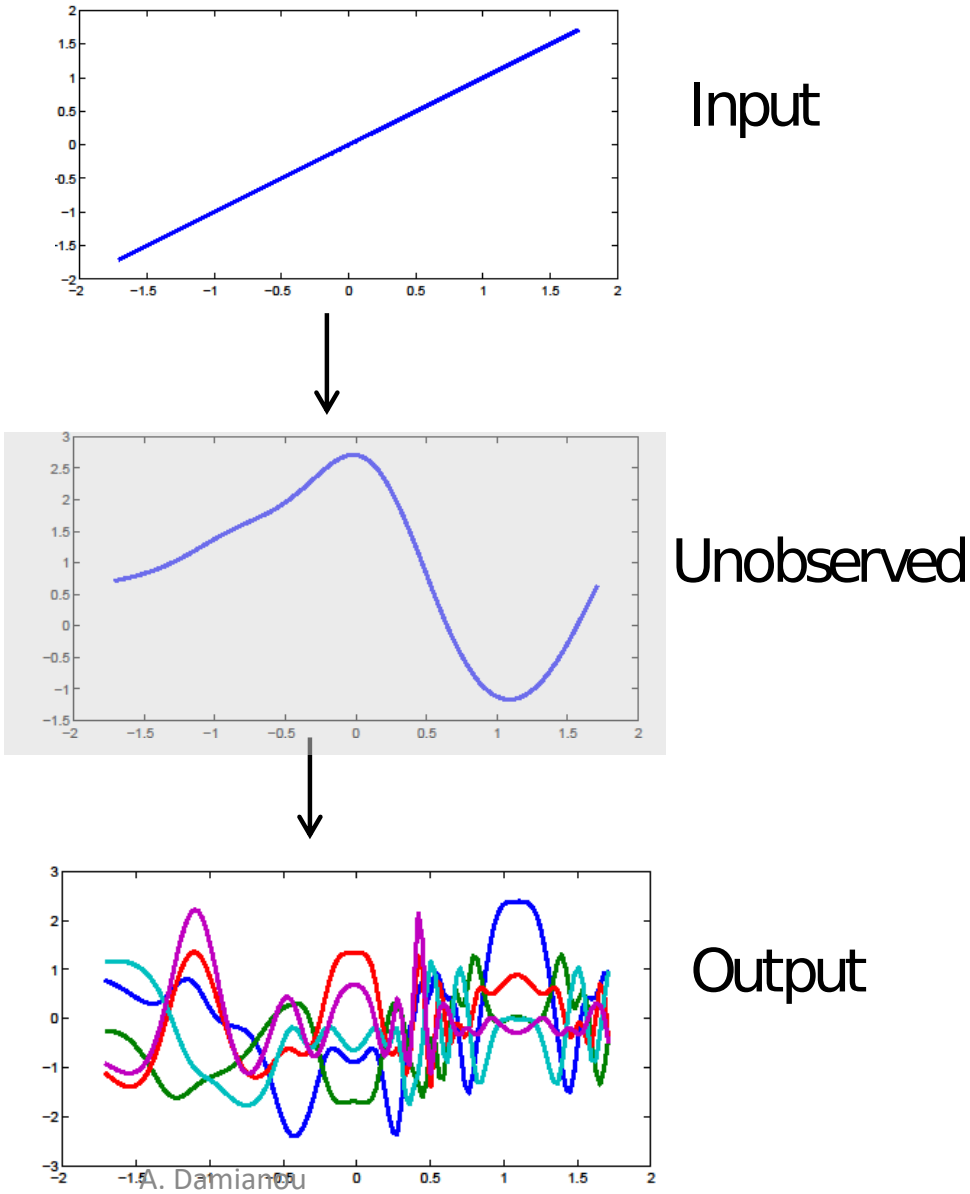
Variational bound and its properties

Bound on the log marginal likelihood *log p(y)*

$$\mathcal{F} = \overbrace{\sum_{l=2}^{L+1} \left\langle \sum_{n=1}^{N} \mathcal{L}(\mathbf{h}_l^{(n)}, \mathbf{u}_l) \right\rangle_{\mathcal{Q}}}^{\text{Data fit}} - \sum_{l=2}^{L+1} \mathsf{KL}\left(q(\mathbf{u}_l) \,\|\, p(\mathbf{u}_l)\right) \underbrace{-\mathsf{KL}\left(q(\mathbf{h}_1) \,\|\, p(\mathbf{h}_1)\right)}_{\text{Regularization}} + \sum_{l=2}^{L} \underbrace{\mathcal{H}\left(q(\mathbf{h}_l)\right)}_{\text{Regularization}}$$

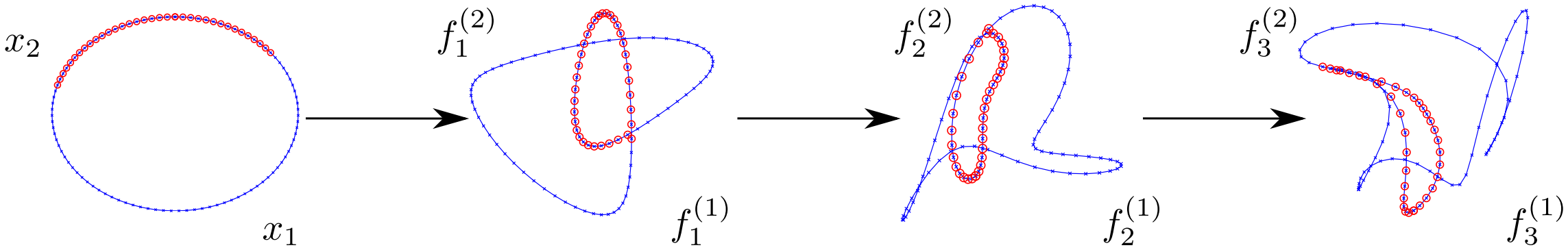
Sampling from a Deep GP





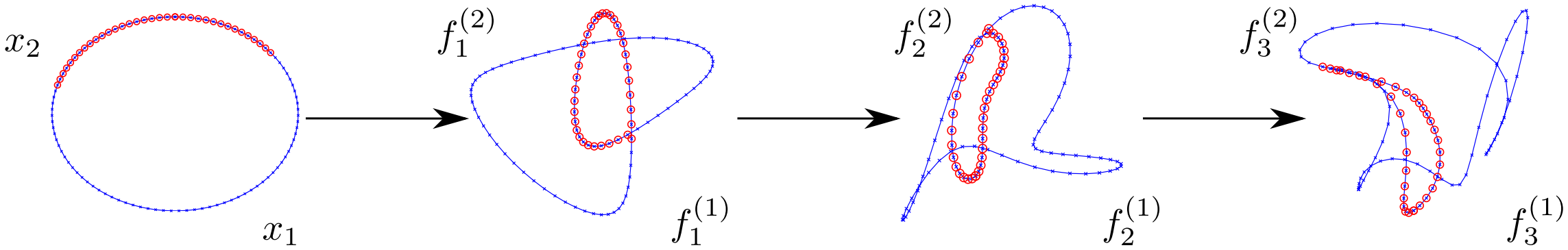
Feature learning

Features are learned as “knots” in the latent space, carried over from layer to layer.



Feature learning

Features are learned as “knots” in the latent space, carried over from layer to layer.



Narrow intermediate layers do give hierarchical feature learning.

DNN vs DGP

- Shallow NN → GP
- Deep NN → GP

DNN vs DGP

- Shallow NN → GP
- Deep NN → GP

Deep GP > Deep NN?
(Discussion)

Analysing DGPs

Limit properties

- Distribution of derivatives in deep models [Duvenaud et al. ‘14]
- How are effective depth and ergodicity connected? [Dunlop et al. ’18]

Few layers analysis through approximation:

- DGP moments and approximation of DGP with GP [Lu et al. ‘19]

Analysing DGPs

Limit properties

- Distribution of derivatives in deep models [Duvenaud et al. ‘14]
- How are effective depth and ergodicity connected? [Dunlop et al. ’18]

Few layers analysis through approximation:

- DGP moments and approximation of DGP with GP [Lu et al. ‘19]

Discussion: What does a DGP with > 2 layers mean?

Part 3/3: From DNNs to (degenerate) GPs

(i.e. Bayesian generalized linear model)

Why?

- Improve learnability (e.g. hierarchical features)

Why?

- Improve learnability (e.g. hierarchical features)
- Leverage DNN research / algorithms

Why?

- Improve learnability (e.g. hierarchical features)
- Leverage DNN research / algorithms
- In industry there are *loads* of DNNs trained (and you can't convince them to train GPs instead). You could reuse the knowledge stored in them.

GP ⟷ DNN beyond initialization (and practical use)

Initialization	**During training**	**Convergence**
Neal ‘94	Jacot et al. ‘18 NTK	Maddox et al. ‘19
Lee et al. ‘18	Lee et al. ‘19	
Matthews et al. ‘18	Hayou et al. ‘19	

GP ⟷ DNN beyond initialization (and practical use)

Initialization	During training	**Convergence**
Neal ‘94 Lee et al. ‘18 Matthews et al. ‘18	Jacot et al. ‘18 NTK Lee et al. ‘19 Hayou et al. ‘19	Maddox et al. ‘19

Model: A degenerate GP from DNNs obtained at convergence.

W. Maddox, S. Tang, P. Moreno, A. Wilson, A. Damianou: *Fast Adaptation with Linearized Neural Networks*. 2019

Motivation

- How can we can get the (desired) DNN properties in GPs?
  - Need to look at convergence to leverage DNN learning
  - Transfer learning from DNN to GP
  - Generalize to transfer learning for multiple tasks

Motivation

- How can we can get the (desired) DNN properties in GPs?
  - Need to look at convergence to leverage DNN learning
  - Transfer learning from DNN to GP
  - Generalize to transfer learning for multiple tasks

- Combines:
  - DNN learnability from #nodes < Inf (training)
  - GP uncertainty & analytic predictions (inference)

DNN training dynamics in GD

$$\frac{\mathrm{d}f(x)}{\mathrm{d}t} = -\eta \underbrace{\mathbf{J}_\theta(x)^\top \mathbf{J}_\theta(x)}_{\text{Neural Tangent Kernel (NTK)}} \nabla_f \log p(y|f,x)$$

- The NTK governs the dynamics of f throughout the GD training of θ.
- Taylor expand: $f(x,\theta) \approx f(x,\theta_0) + J_\theta(x,\theta_0)^T(\theta - \theta_0)$
- Linear in θ, non-linear in inputs (because of J)
- Linear model using feature map (kernel) $J_\theta(x,\theta_0)^\top J_\theta(x,\theta_0)$

Lee et al. *arXiv:1902.06720*

DNN training dynamics in GD

$$\frac{\mathrm{d}f(x)}{\mathrm{d}t} = -\eta \underbrace{\mathbf{J}_\theta(x)^\top \mathbf{J}_\theta(x)}_{\text{Neural Tangent Kernel (NTK)}} \nabla_f \log p(y|f,x)$$

- The NTK governs the dynamics of f throughout the GD training of θ.
- Taylor expand: $f(x,\theta) \approx f(x,\theta_0) + J_\theta(x,\theta_0)^T(\theta - \theta_0)$
- Linear in θ, non-linear in inputs (because of J)
- Linear model using feature map (kernel) $J_\theta(x,\theta_0)^\top J_\theta(x,\theta_0)$

Lee et al. *arXiv:1902.06720*

DNN training dynamics in GD

$$\frac{\mathrm{d}f(x)}{\mathrm{d}t} = -\eta \underbrace{\mathbf{J}_\theta(x)^\top \mathbf{J}_\theta(x)}_{\text{Neural Tangent Kernel (NTK)}} \nabla_f \log p(y|f,x)$$

- The NTK governs the dynamics of f throughout the GD training of θ.
- Taylor expand: $f(x,\theta) \approx f(x,\theta_0) + J_\theta(x,\theta_0)^T(\theta - \theta_0)$
- Linear in θ, non-linear in inputs (because of J)
- Linear model using feature map (kernel) $J_\theta(x,\theta_0)^\top J_\theta(x,\theta_0)$

Lee et al. *arXiv:1902.06720*

DNN -> GP through NTK

- For linearized networks around θ_init, the network output becomes a linear model with NTK.
- For non-linearized networks and small learning rate, same behavior arises with GD.

DNN -> GP through NTK

- For linearized networks around θ_init, the network output becomes a linear model with NTK.
- For non-linearized networks and small learning rate, same behavior arises with GD.
- We leverage this to linearize the network at convergence, θ_final, and use it within a GP with the NTK.

DNN -> GP through NTK

- For linearized networks around θ_init, the network output becomes a linear model with NTK.
- For non-linearized networks and small learning rate, same behavior arises with GD.
- We leverage this to linearize the network at convergence, θ_final, and use it within a GP with the NTK.
- This also allows us to do DNN transfer learning analytically with GPs: we transfer neural network parameters (= kernel parameters) across tasks.

Model

- Fit DNN's parameters θ on data $\mathbf{X}$ using the Jacobian: $J(\mathbf{X}; \theta)$ (i.e. SGD).
- Consider a GP with finite NTK kernel: $k(x, x') = J(x; \theta)^\top J(x'; \theta)$.
  No GP training required.
- Predictions: $k(x^*, \mathbf{X}) = J(x^*; \theta)^T J(\mathbf{X}; \theta)$ etc.
- Equivalent to Bayesian generalized linear model with Jacobians as features.

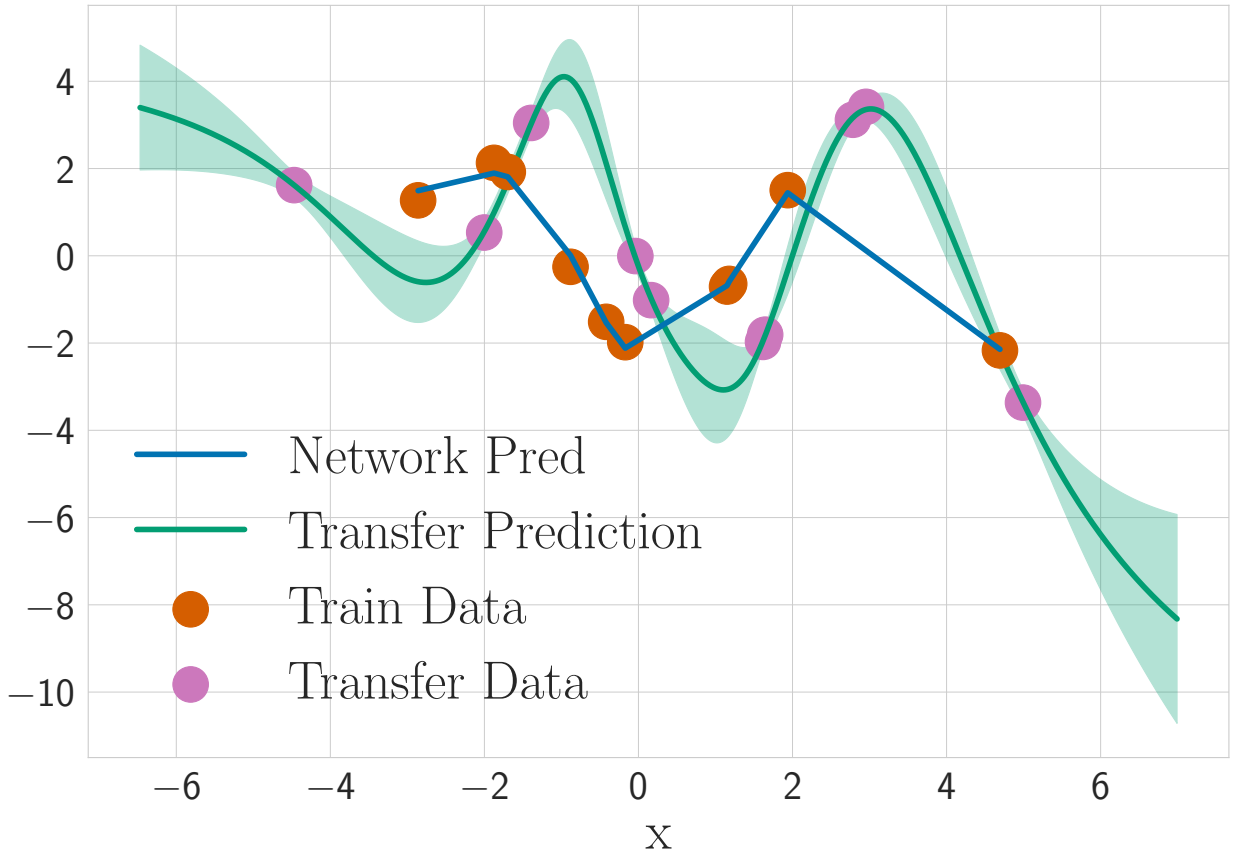
DNN adaptation without fine-tuning

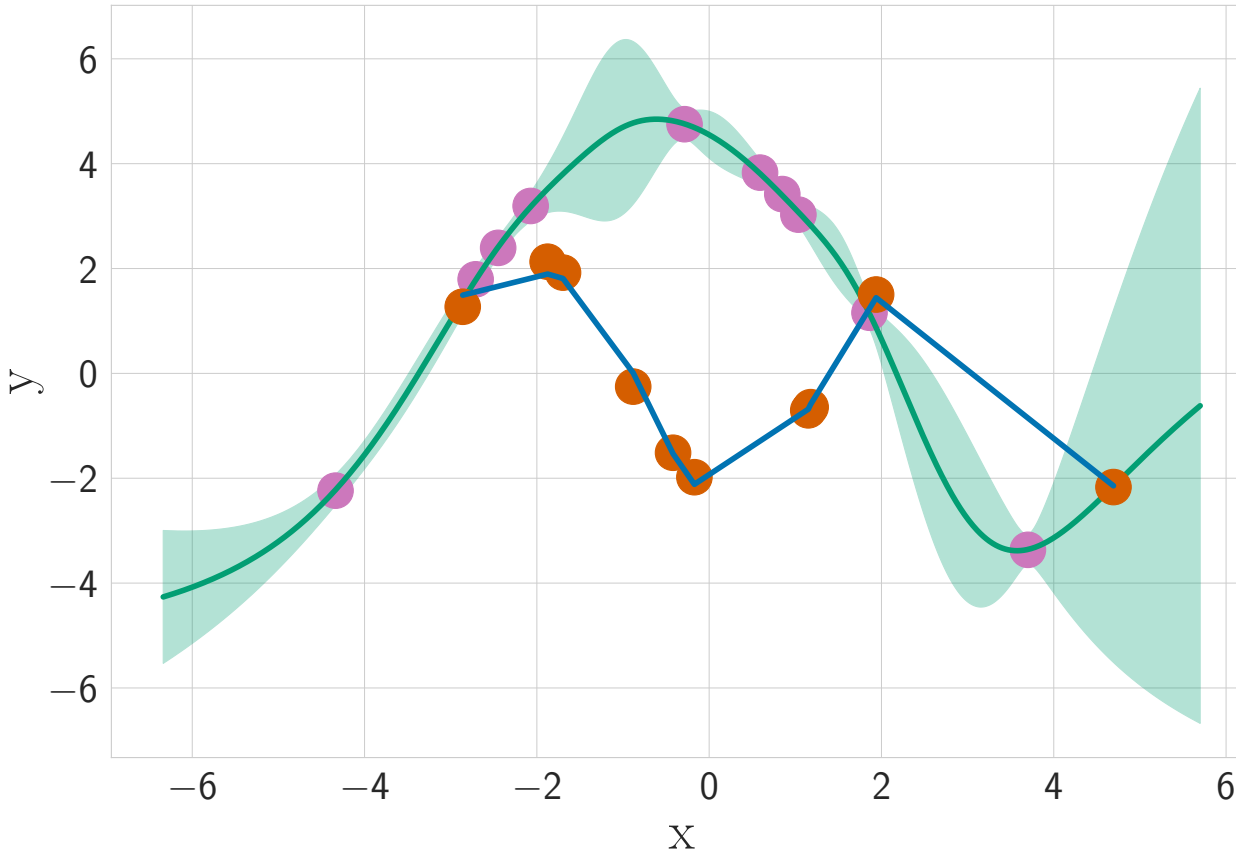
If #tasks $T > 1$:

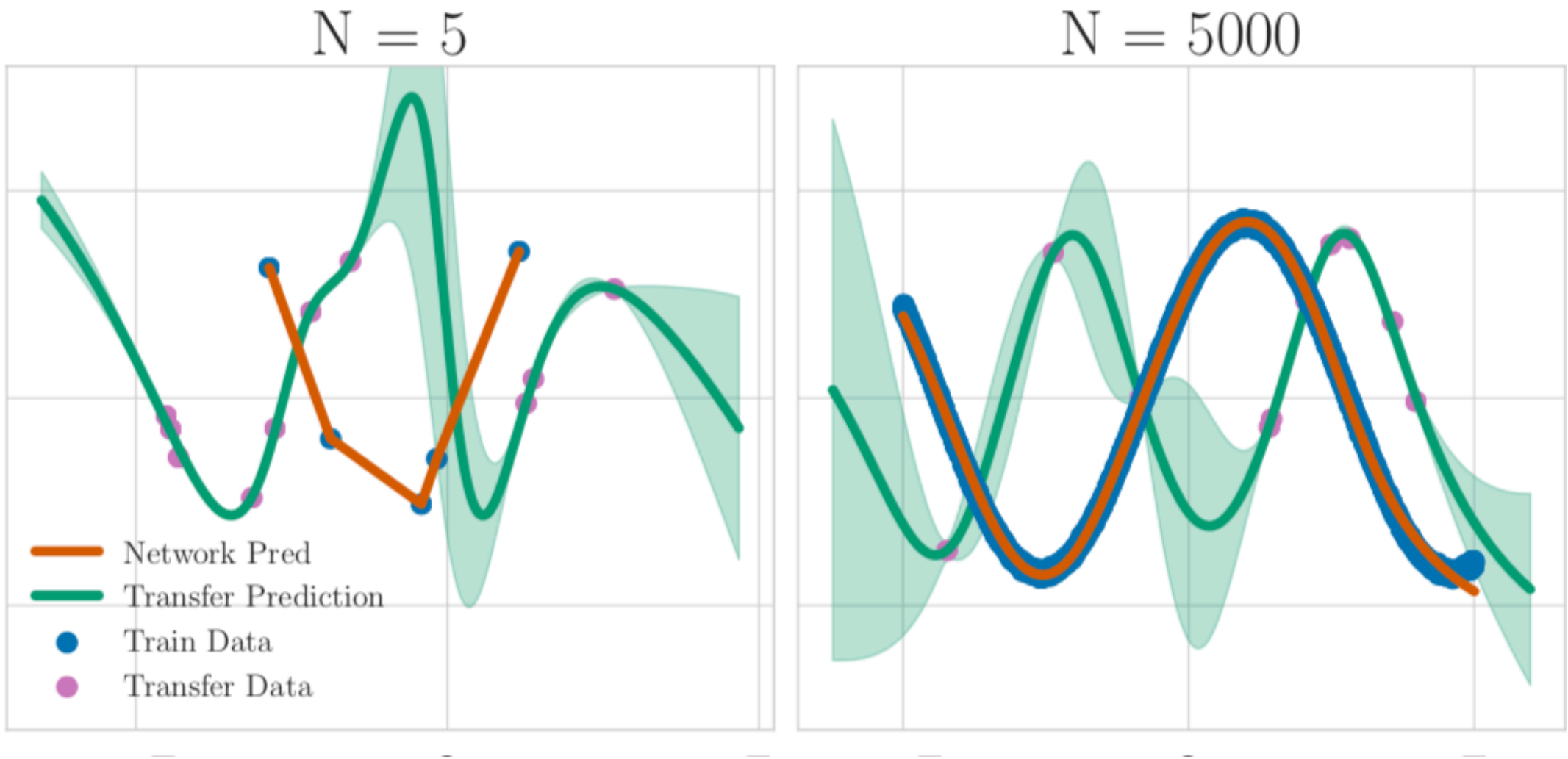
- Learn kernel hyperparameters θ_0 in source task
- Transfer kernel hyperparameters to all other thasks*: $\theta_{1,\dots,T} = \theta_0$
  $\Rightarrow$ closed form adaptation (in regression) with uncertainty!
- J depends on the whole network, not just the last layer. Also tells us about convergence conditions.

*see also Fisher Matrix similarity assumptions by *Liang et al. 17, Achille et al. 19*

Transfer learning (toy regression data)

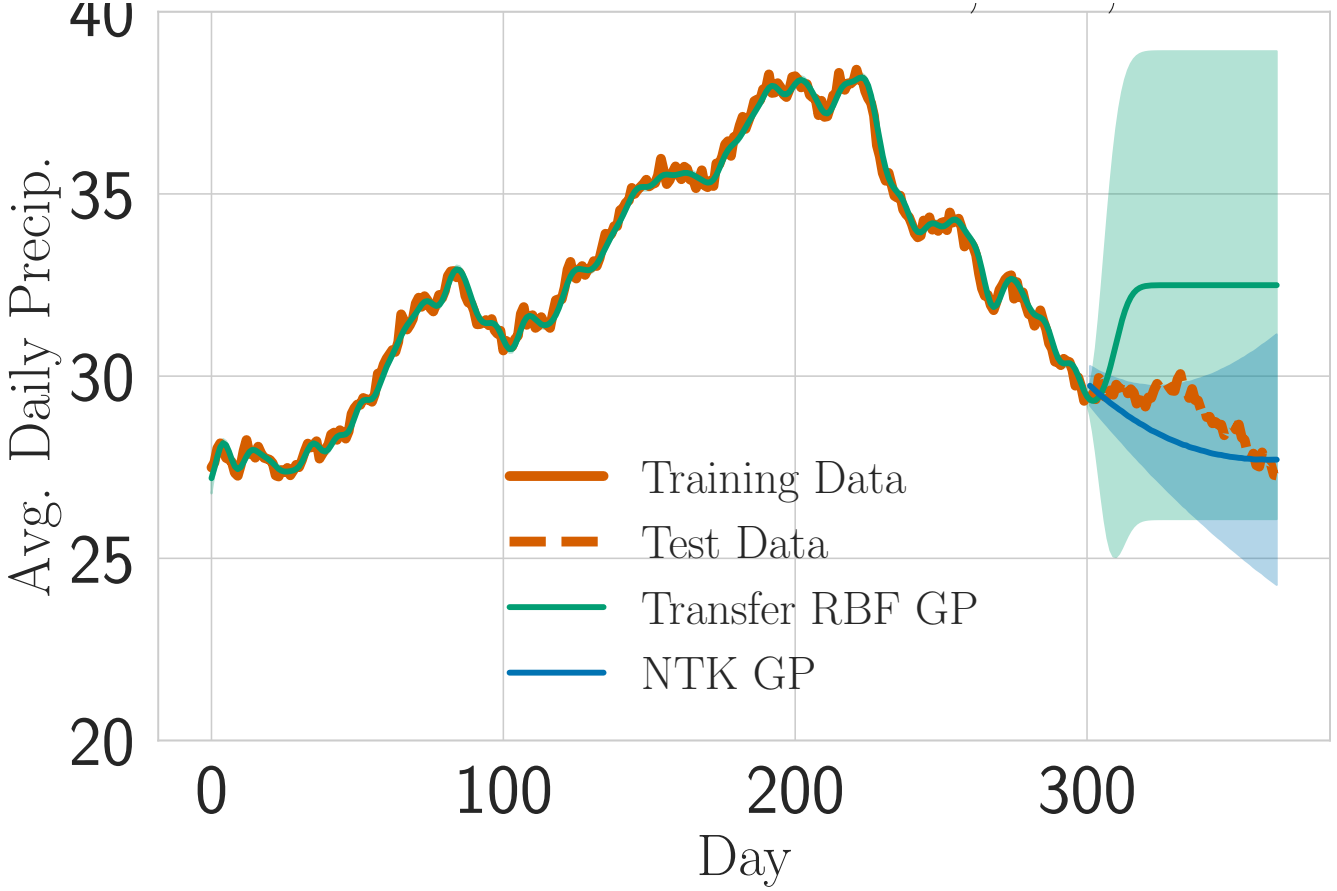




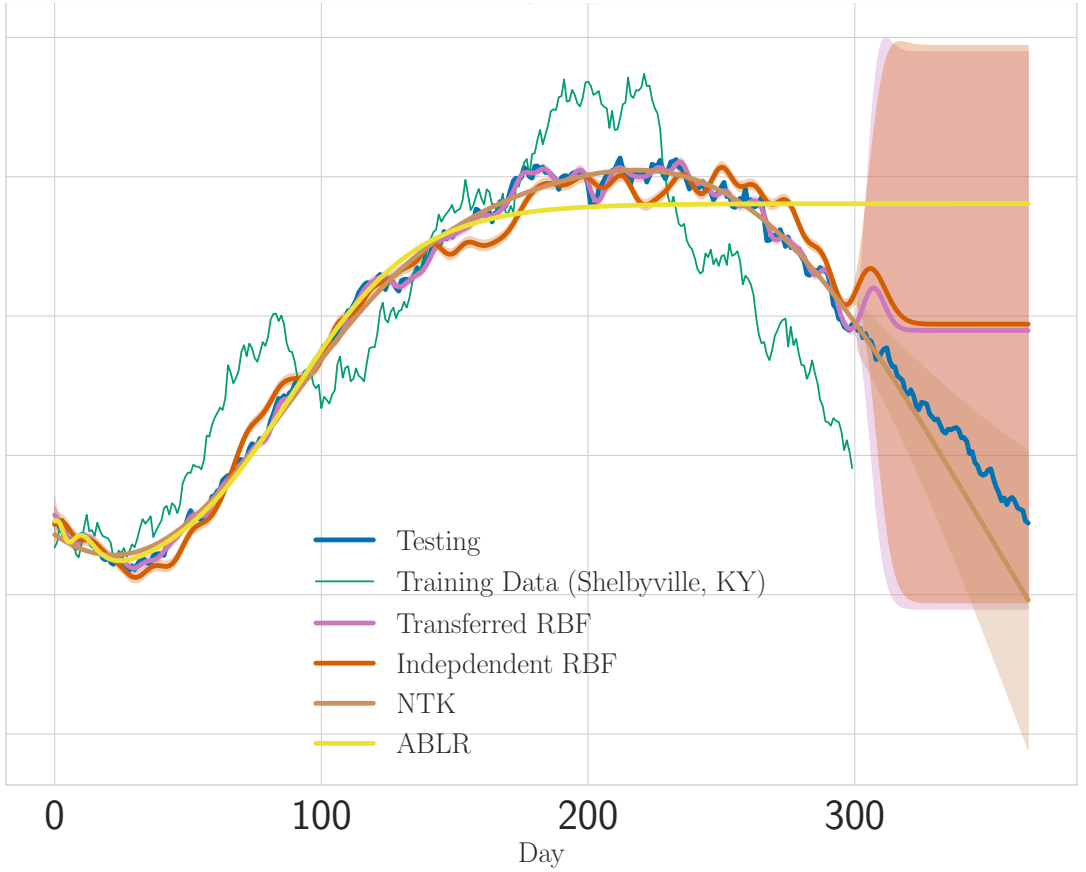

N = 5
N = 5000
Network Pred
Transfer Prediction
Train Data
Transfer Data

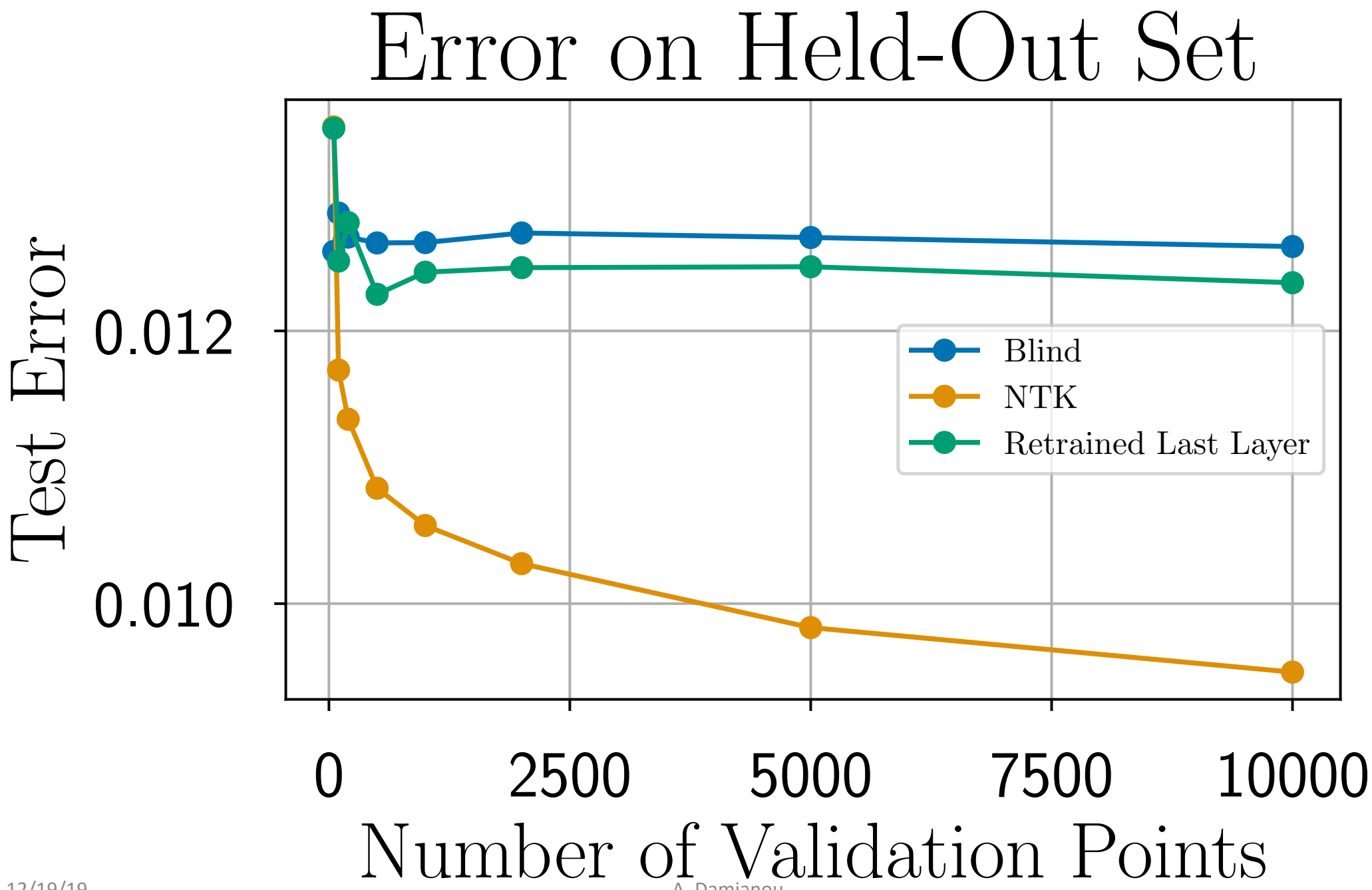
Transfer learning for precipitation data

Source task



Target task




Error on Held-Out Set
Test Error
0.012
0.010
0
2500
5000
7500
10000
Number of Validation Points
Blind
NTK
Retrained Last Layer

Other relevant works

- Perrone et al. 2018: Explicitly uses output features of DNN (we use NTK with Jacobian information from all layers)
- Wilson et al. 2015: DKL (whole NN embedded in kernel)
- Jaakkola et al. 1998 and others: gradients as features

Thanks!

Questions?

Appendix

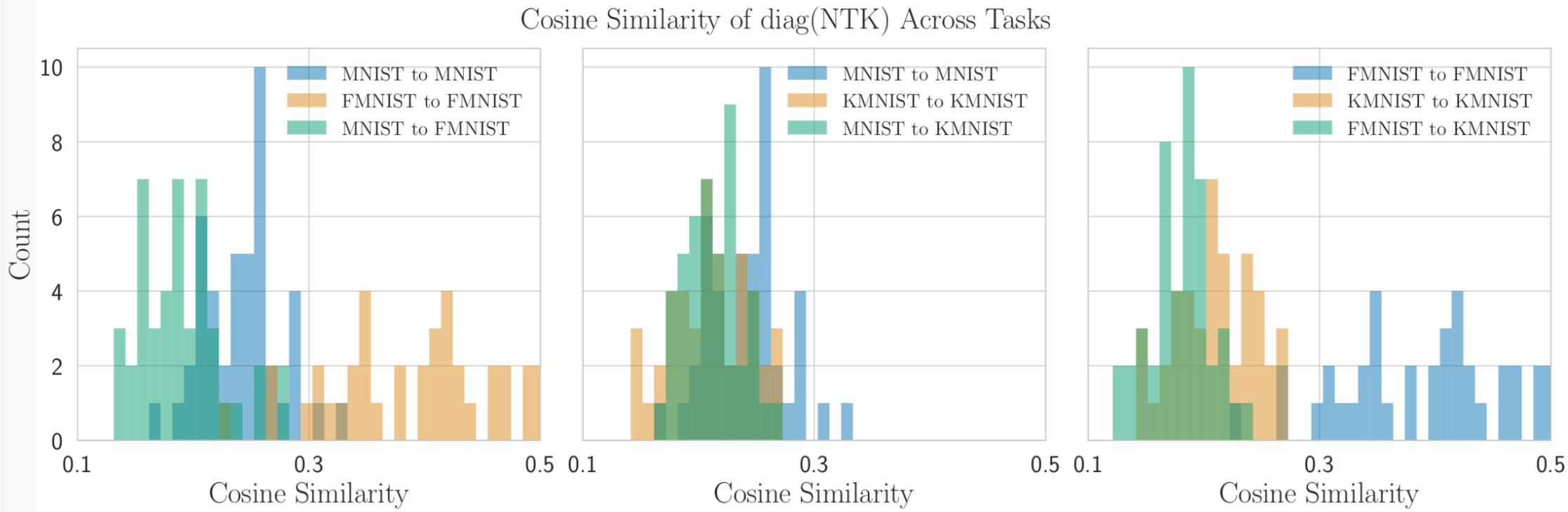


Figure 1: Cosine similarity of the diagonal of the Jacobian matrix for separately trained LeNet3s on three separate tasks. Note that MNIST is considerably more similar to KMNIST as they are both digits, while both are dissimilar to FMNIST which has different features.