On Transfer Learning via Linearized Neural Networks

Linearizing trained neural networks produces a finite width neural tangent kernel that can be used for fast adaptation.

Background:

- We expect the features across a neural network to be similar across tasks.
- Jacobian matrices are similar across tasks because Fisher information matrices are similar (Achille et al, 2019).

Description:

- Infinitely wide neural networks produce a kernel function that is often useful.
- What about the finite width regime?
  - Use the Jacobian matrix so that the kernel becomes

$$K(x, x') := J_\theta(x)^T J_\theta(x')$$

- We share the parameters θ across tasks.

- Work with the Jacobian matrix implicitly via matrix vector products

- Only need $J_\theta(x)v$ and $J_\theta(x)^T v$ (Pearlmutter, 1994)
- Then use conjugate gradients and Lanczos as in **GPyTorch** (Gardner et al, 2018)

- To be able to mini batch computation, use fast Fisher-vector products

- Regression:

$$\mathbb{F}(\theta) = \frac{1}{N} J_\theta(x) J_\theta(x)^T$$

- (Approximate) Fisher vector products:

$$\nabla_{\theta'} \mathrm{KL}(p(y|\theta)||p(y|\theta')|_{\theta'=\theta+\epsilon v} = \epsilon \mathbb{F}(\theta)v + \mathcal{O}(\epsilon^2 ||v||)$$

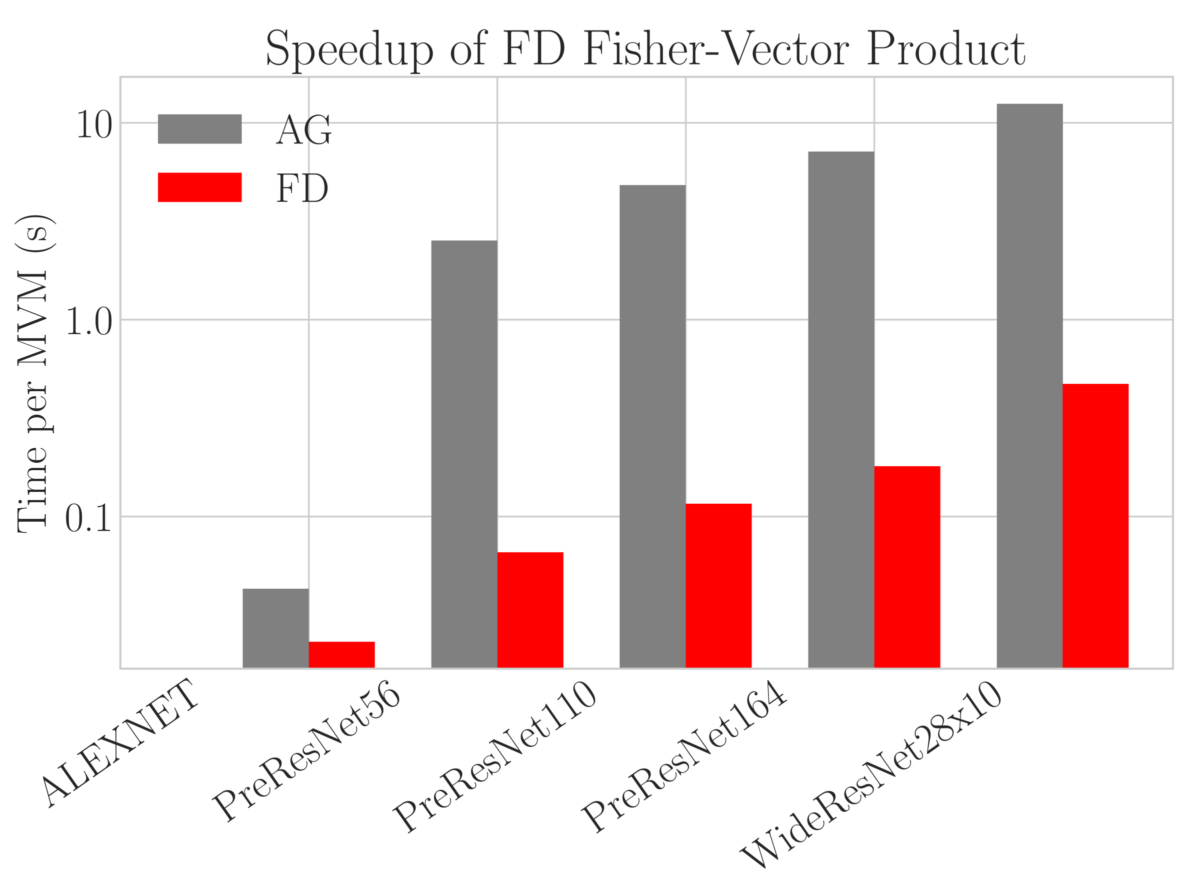
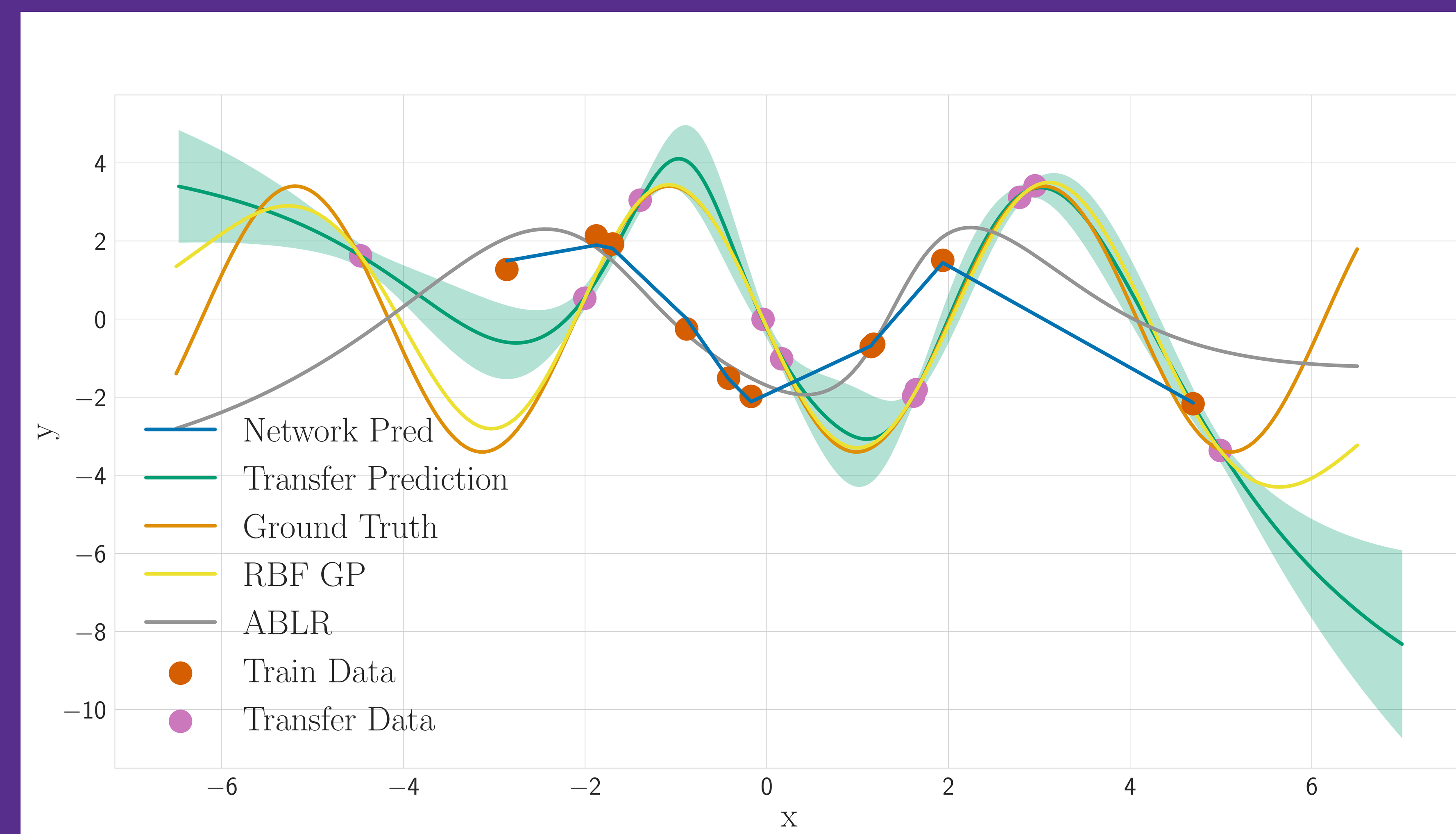
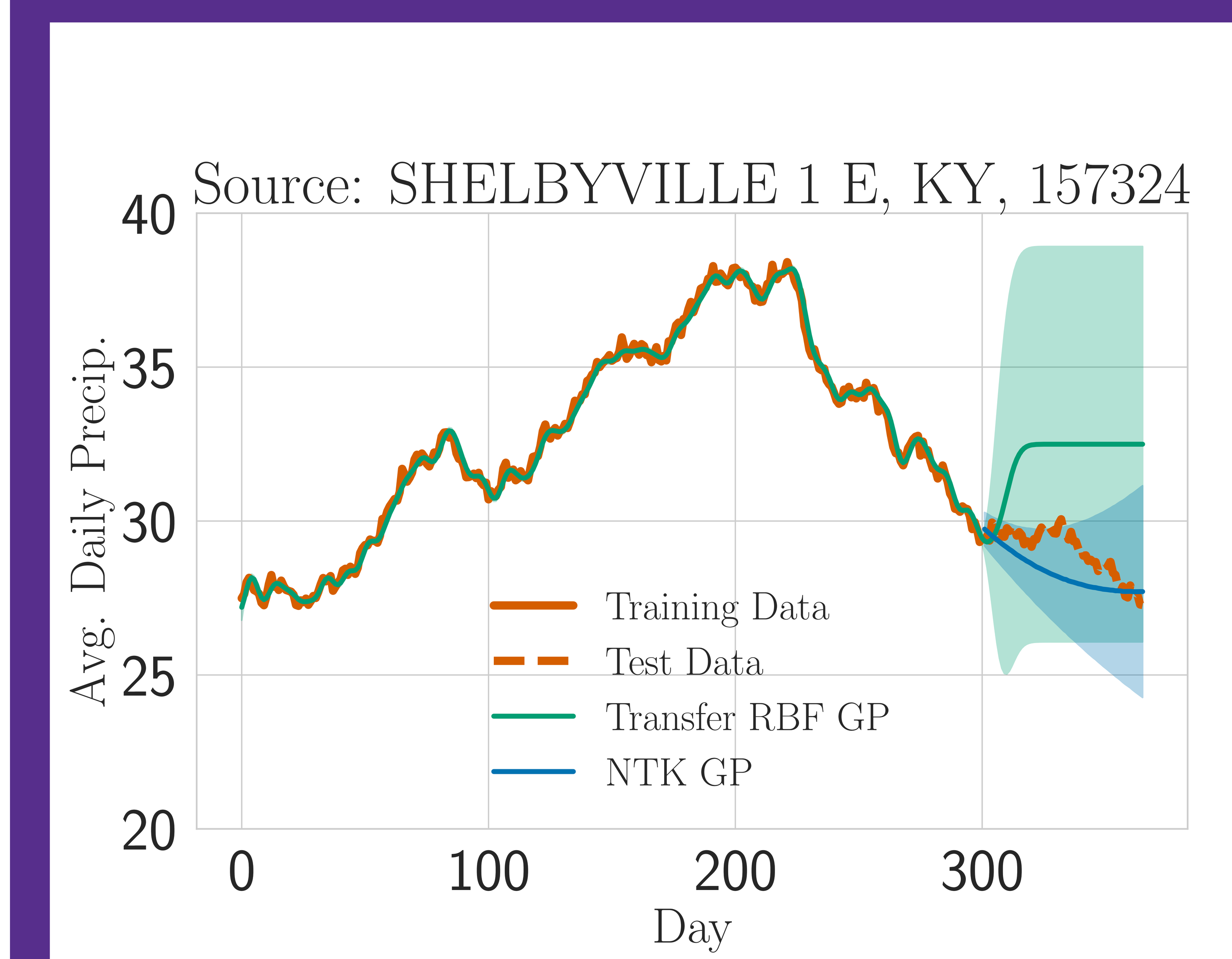


Figure: Speed comparison of Fisher vector products on CIFAR10. 30x speedup on most models.

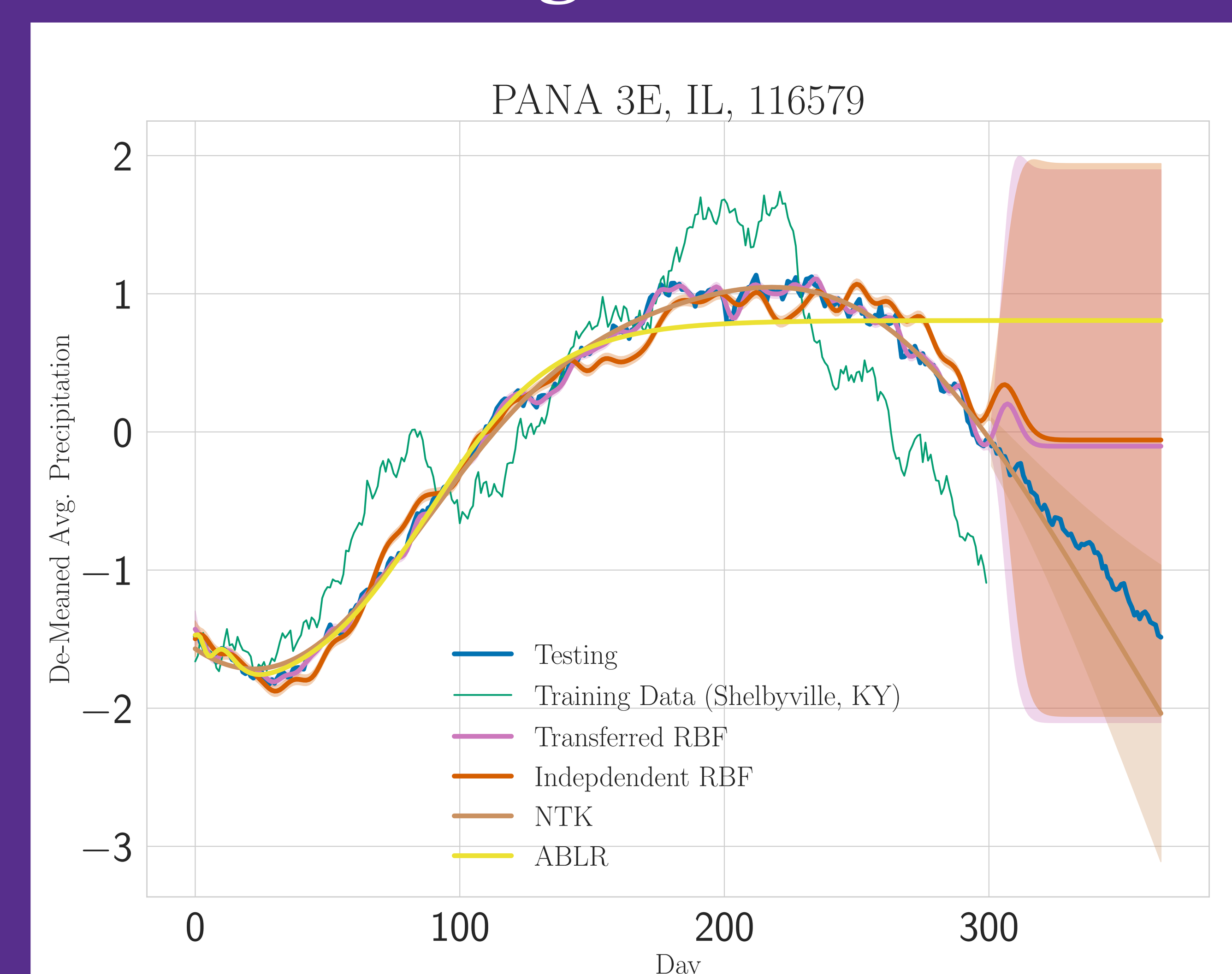
Transferring Sinusoids



Source Task



Target Task



Take a picture to see the code and paper.

Probabilistic Model over Tasks

$$\theta_t' \sim p(\theta_t)$$

$$f_t = J_\theta(X_t)^T \theta_t + \mu_t$$

$$y_t | f_t \sim p(y_t | f_t)$$

- Evaluation: train on one task and then evaluate on every other task.

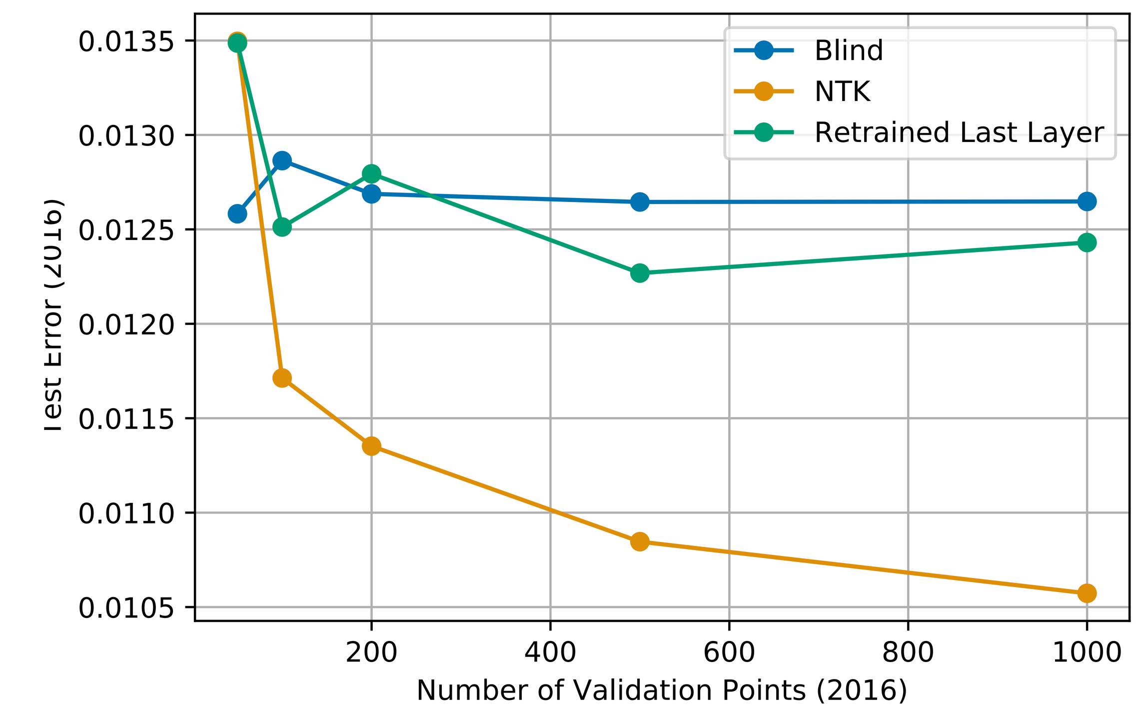


Figure: *Large Scale malaria dataset from Malaria Global Atlas. Trained on 2000 data points from 2012, but tested on 2016 data - given various amounts of validation data to the network.* ***Finite NTK improves with more validation data from 2016 seen for a fixed testing set.***

Future work will be to back propagate through the Jacobian vector products and convert into a closed form meta learning objective.

References:

- *Pearlmutter, 1994*. Fast exact multiplication by the Hessian, Neural Computation.
- *Gardner et al, 2018*. Gpytorch: Black Box Matrix Matrix Gaussian Process Inference with GPU Acceleration, NeurIPS.
- *Achille et al, 2019*. Task2vec: Task Embedding for meta-learning, arXiv:1902.03545.

Wesley Maddox[+*], Shuai Tang[+a], Pablo Garcia Moreno[^], Andrew Gordon Wilson[*], Andreas Damianou[^]

[*] NEW YORK UNIVERSITY

[^] amazon.com

[a] UC San Diego Cognitive Science

[+] Work performed during an internship at Amazon Cambridge.