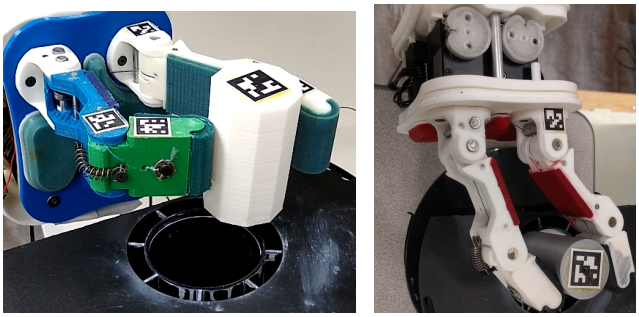
Learning to Transfer Dynamic Models of Underactuated Soft Robotic Hands

Liam Schramm, Avishai Sintov and Abdeslam Boularias

***Abstract*— Transfer learning is a popular approach to bypassing data limitations in one domain by leveraging data from another domain. This is especially useful in robotics, as it allows practitioners to reduce data collection with physical robots, which can be time-consuming and cause wear and tear. The most common way of doing this with neural networks is to take an existing neural network, and simply train it more with new data. However, we show that in some situations this can lead to significantly worse performance than simply using the transferred model without adaptation. We find that a major cause of these problems is that models trained on small amounts of data can have chaotic or divergent behavior in some regions. We derive an upper bound on the Lyapunov exponent of a trained transition model, and demonstrate two approaches that make use of this insight. Both show significant improvement over traditional fine-tuning. Experiments performed on real underactuated soft robotic hands clearly demonstrate the capability to transfer a dynamic model from one hand to another.**



(a) Source Hand (b) Target Hand

Fig. 1. Model T42 3D-printed adaptive hands used in the experiments. A transition model learned from the source hand is transferred to the second hand with a small amount of data, and used to accurately control it.

I. Introduction

Soft robots pose a series of challenges for open-loop control. Many aspects of these systems, such as the compliance and friction coefficients of each joint, are difficult to measure. This makes motion planning that relies on direct modeling difficult. Learning a dynamical model is a popular alternative to analytical and handcrafted modeling, but collecting data for these models is time-consuming and incurs wear and tear on the robot, often changing the system's dynamics before the model can be deployed. Efficient motion planning requires an accurate model that can be learned from relatively few data points. Neural networks are often used for this purpose. They offer the advantage of fast queries, which allows for long-horizon planning in large state and action spaces. However, neural nets also require large amounts of data. One way of solving this problem is to re-use data from a closely related system to improve data efficiency. This approach is known as *transfer learning*. The standard current practice for transferring neural networks across domains consists in first training a network on a *source* domain, where data is relatively abundant, and then fine-tuning the network on a final *target* system, where data is scarce. Fine-tuning is commonly used in deep learning.

In this paper, we show that in certain situations, fine-tuning can worsen the predictions of the source network. This is because in some dynamical systems, such as the 3D-printed soft robotic hands illustrated in Figure 1, the gradient of the state transition function can be important, due to sudden changes in friction and compliance coefficients of the joints across the state space. In chaotic systems for instance, the derivative determines the *Lyapunov exponent*, which describes how much trajectories diverge over time as a function of infinitesimally small differences in their start states. We take a system to be chaotic if its maximal Lyapunov exponent is greater than 0. We show that under certain constraints, gradients of learned transition functions converge to the true gradients of the underlying ground-truth functions. However, in sparse data regimes, derivatives may strongly diverge, leading to models that are significantly more chaotic than the underlying true transition function. Based on this observation, we present two simple methods designed to prevent a transferred neural network from becoming chaotic if its underlying function is not chaotic. The key idea here is to design methods that keep the *Lyapunov exponent* small as opposed to methods that focus on minimizing one-step prediction errors. Experiments on the real hands shown in Figure 1 demonstrate that both methods produce predictions that are more accurate than the popular fine-tuning approach and a few other methods.

The authors are with the Computer Science Department of Rutgers University. {lbs105,ab1544}@cs.rutgers.edu. This work is supported by NSF awards IIS-1734492, IIS-1723869, and IIS-1846043. The opinions and findings in this paper do not necessarily reflect the sponsor's views.

II. Problem Statement

We consider the problem of transferring a dynamic model of a source robot S to a target robot T that shares the same state space $\mathcal{X}$, a subset of $\mathbb{R}^n$, with metric d. We define a dynamical model as a transition function f that maps a state-action pair (x_t, μ_t) to a next state x_{t+1}, i.e. $f(x_t, \mu_t) = x_{t+1}$. We denote by f_S and f_T the transition function of the source and target robots, respectively. A policy is a function π that maps a state x_t into an action μ_t. We assume that we are given a sufficiently large set $\mathcal{D}_S$ of data

points (x_i, μ_i, x_{i+1}) generated with a random policy with the source robot, and a much smaller data set $\mathcal{D}_S$ obtained from the target robot, i.e. $|\mathcal{D}_T| \ll |\mathcal{D}_S|$. In this work, we study the difference between the responses of the two robots under the same sequence of actions $\mu_1, \mu_2, \mu_3, \ldots, \mu_n$, in an open-loop control. The sequence is chosen according to a desired final goal or for tracking a path. The focus of this work is not on how to choose the actions, but on how the two robots respond to the same sequence of actions. In order to ease notations, we drop the actions from the notations, and re-define the transition function accordingly as $f^n(x) = f(\ldots(f(f(f(x, \mu_1), \mu_2), \mu_3), \ldots, \mu_n)$. Let $\hat{f}_S$ denote an approximation of the unknown true function f_S, learned from the source data set $\mathcal{D}_S$. Let $\hat{f}_T$ denote an approximation of the unknown true function f_T, learned from the target data set $\mathcal{D}_T \cup \mathcal{D}_S$. In the present work, we use neural networks for learning both $\hat{f}_S$ and $\hat{f}_T$. Predicted trajectories will thus be given by $x_n = \hat{f}^n(x_0)$. Let the time before divergence be given by $t(x, \epsilon) = \max_{n \in \mathbb{N}}[d\big(\hat{f}_T^n(x), f_T^n(x)\big) < \epsilon]$ where ϵ is some constant. Our goal is to find some model $\hat{f}_T$ that maximizes the average value of $t(x, \epsilon)$ over all $x \in \mathcal{X}$ for some chosen value of ϵ. This metric is useful because it measures how far into a future a model can predict with reasonable accuracy, which is important for path planning, as it relates to the planning horizon. $t(x, \epsilon)$ is influenced by two main factors. The first is the learned model's prediction error at each step. The second is the Lyapunov exponent of the learned model, defined for discrete time systems with transition function f as $\lambda_f(x_0) = \lim_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n-1} \ln |f'(x_i)|$ with $x_{i+1} = f(x_i)$.[1]

The problem then consists in finding from $\mathcal{D}_T \cup \mathcal{D}_S$ a function $\hat{f}_T$ that approximates f_S but that also has a small Lyapunov exponent $\lambda_{\hat{f}_T}$ to avoid chaotic divergences that typically occur in recurrent neural networks.

III. Related Works

A. Underactuated Adaptive Hands

Deriving analytical models for underactuated hands is a challenging task due to the complex response of the passive joints and uncertainties in internal frictions of joints and external frictions between fingers and object. To predict the configuration of the gripper, one needs to know precisely the forces being applied on the object, which cannot be obtained when tactile sensing is not available to estimate friction at every point on the contact surface. Typical models assume a uniform friction on the contact surface [1]. Moreover, controlling individual joint positions in an underactuated system is a challenging problem. Models for underactuated manipulation typically use simplified frictional models while examining joint configurations, joint torques, and energy [2]–[4]. A popular approach applies screw theory to further simplify the derivation for a model [5]. Other modeling methods can be found in [6]–[8]. These proposed techniques have been shown to be sensitive to assumptions in external constraints. They are generally used for simulations only. To control real underactuated adaptive hands, models of dynamics need to be learned or tuned from data collected with the real hands.

B. Learning Dynamical Models

A dynamical model is a mapping from a given state and action to the next state. Such models play a key role in model-based RL [9]–[16]. A common approach for modeling of dynamical systems is the Gaussian Process (GP) [17]–[19]. Usages of data-driven models include learning the distribution of an object's position after a grasp [20] or during re-grasping [21], and a hybrid modeling approach combining analytical and data-based models to improve accuracy in feed-forward control [22]. Neural networks have became more popular recently thanks to their simplicity, capacity of learning, and scalability to large amounts of data, in contrast to nonparametric methods such as GPs.

C. Neural Dynamical Systems

A number of works [23]–[25] have studied the dynamical properties of recurrent neural networks, with both [23] and [24] describing situations in which chaotic behavior can arise. The work by [26] explores a variation of the recurrent neural network that does not exhibit chaotic behavior. However, we find that for dynamics models it is more accurate and more data-efficient to predict the change in state and feed the new state to the model than it is to predict the state directly and use memory to learn a representation of the state. Chaining together predictions as we do would in the present work causes the network of [26] to become chaotic, so this method is not applicable for open-loop control.

D. Transfer Learning in Robotics

Training dynamical models typically requires several hours of data collection [27]. To alleviate this problem, several works considered transferring learned models across different robots or tasks [28]–[38]. A typical approach to transfer learning consists in projecting data from two domains into a low-dimensional manifold [39], where correspondences between data points from the two domains can be learned without supervision [40]. This approach was adopted for transferring inverse dynamics models of rigid robotic manipulators [28], [34], [37], wherein the models are learned with the LWPR method while the low-dimensional manifold was given by PCA. Multi-robot transfer learning was also studied from a dynamical system perspective in [29] wherein sufficient conditions for a bounded-input, bounded-output (BIBO) stability are provided for transfer learning maps. A closely related transfer technique was used in [38] for trajectory tracking with quadrotors. While the present work also studies the transfer problem from a dynamical system perspective, it focuses on the Lyapunov stability.

An increasingly popular approach for reducing the data needs in robot learning is to transfer trained models from

[1] One-step error accounts for all "new" error, introduced by inaccuracy in the model. The Lyapunov exponent accounts for the growth of error from earlier steps.

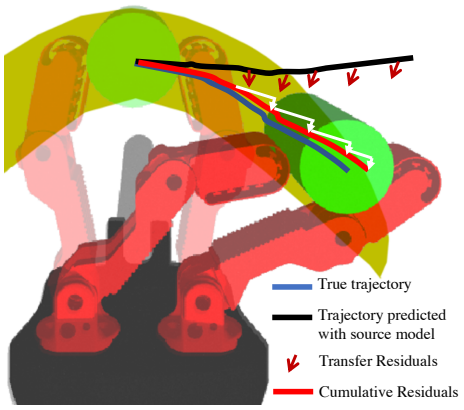


Fig. 2. Workspace of the considered hand. Errors of predicted future states accumulate over time. The proposed cumulative residuals approach solves this issue by bounding the Lyapunov exponent of the transferred model.

simulation to real world [30]–[33], [35]. *Sim-to-real* has been used mostly for transferring policies in model-free RL, with only a few works related to the transfer of dynamics. In [41], a dynamics model is adapted online by combining prior knowledge from previous tasks. Another technique [30] combines a forward dynamics model, trained in simulation, with an inverse dynamics model trained on a real robot.

IV. Proposed Analysis

To derive a new transfer learning algorithm that meets the specific challenges of our problem, we first analyse the Lyapunov exponent of a learned transition function $\hat{f}$ as a function of the training data $\mathcal{D}$ and the Lyapunov exponent of the underlying true dynamics f. We find the primary driver of divergence in the target regime is chaotic behavior, characterized by a large Lyapunov exponent. As we train a new model, we find that training not only reduces the average error of the model, but also its Lyapunov exponent. Even when a model does not show outright chaotic behavior, it tends to compound on its own errors in a way that damages its performance. This will be discussed further in the experiments section. Intuitively, we show that as we accumulate more data, the models we train become less chaotic. To do this, we derive an upper bound on the Lyapunov exponent of the learned transition function.

Definitions. Let r be the maximum distance from any point $x \in \mathcal{X}$ to the nearest point in $\mathcal{D}$. Let ϵ be the maximum error of $\hat{f}$'s predictions on data $\mathcal{D}$. In other terms, $\epsilon = \max_{x \in \mathcal{D}} \|\hat{f}(x) - f(x)\|_d$, according to metric d.

Assumptions. Suppose $max_{x \in \mathcal{X}} |f''(x)| \leq c$ and $max_{x \in \mathcal{X}} |\hat{f}''(x)| \leq c$ for some constant c.

Theorem 1.

$$\lambda_{\hat{f}}(x_0) \leq \lim_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n} \ln |\, f'(x_i) + 2\sqrt{\epsilon}(1+c) + 6rc \,|$$

for all x_0 in $\mathcal{X}$.

We provide the proof for the one-dimensional case. A similar result is possible for higher dimensional cases (up to a factor of $(1 + r/\epsilon)$). The proof for higher dimensions is longer and less intuitive to present here, it will be included in a longer version of the paper.

Proof:

Let x_1 be a data point in $\mathcal{D}$. Since no point is $\mathcal{X}$ is farther than r from the nearest point in $\mathcal{D}$, let us choose some point in $\mathcal{X}$ a distance Δx from x_1, and let x_2 be its nearest neighbor in $\mathcal{D}$, with $\Delta x > r$. Then $\Delta x - r \leq d(x_1, x_2) \leq \Delta x + r$. From the mean value theorem (MVT), we can see that for some x_3 with $x_1 < x_3 < x_2$, $\hat{f}'(x_3) = \frac{\hat{f}(x_2) - \hat{f}(x_1)}{x_2 - x_1}$. Thus, $\hat{f}'(x_3) \leq \frac{f(x_2) - f(x_1)}{x_2 - x_1} + \frac{2\epsilon}{x_2 - x_1} \leq \frac{f(x_2) - f(x_1)}{x_2 - x_1} + \frac{2\epsilon}{\Delta x - r}$. Applying the MVT again to f, we find that for some x_4 with $x_1 < x_4 < x_2$, we get $\hat{f}'(x_3) \leq f'(x_4) + \frac{2\epsilon}{\Delta x - r}$. We know that $|\, x_1 - x_3 \,| \leq \Delta x + r$ and $|\, x_1 - x_4 \,| \leq \Delta x + r$. From the constraint on f'', we can see that $\hat{f}'(x_1) \leq \hat{f}'(x_3) + (\Delta x + r)c \leq f'(x_4) + \frac{2\epsilon}{\Delta x - r} + (\Delta x + r)c \leq f'(x_1) + (\Delta x + r)c + \frac{2\epsilon}{\Delta x - r} + (\Delta x + r)c \leq f'(x_1) + \frac{2\epsilon}{\Delta x - r} + 2(\Delta x + r)c$ for any $x_1 \in \mathcal{D}$. Similarly, going from any $x \in \mathcal{X}$ to the nearest data point in $x_1 \in \mathcal{D}$, we get the bound $\hat{f}'(x) \leq \hat{f}'(x_1) + rc \leq f'(x_1) + \frac{2\epsilon}{\Delta x - r} + (2\Delta x + 3r)c \leq f'(x) + \frac{2\epsilon}{\Delta x - r} + (2\Delta x + 4r)c$

Let us choose $\Delta x = \sqrt{\epsilon} + r$. Then for any $x \in \mathcal{X}$, $\hat{f}'(x) \leq f'(x) + 2\sqrt{\epsilon} + (2\sqrt{\epsilon} + 6r)c$. The Lyapunov exponent of a function f is defined as $\lambda_f(x_0) = \lim_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n-1} \ln |f'(x_i)|$. Hence, the Lyapunov exponent of $\hat{f}$ has an upper bound given by $\lambda_{\hat{f}}(x_0) \leq \lim_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n} \ln |\, f'(x_i) + 2\sqrt{\epsilon}(1+c) + 6rc \,|$ □.

Corollary 1.

$\lambda_{\hat{f}}(x_0) \lessapprox \lambda_f + \ln(1 + e^{-\lambda_f} \,|\, 2\sqrt{\epsilon}(1+c) + 6rc \,|)$

Proof: Since we know the limit behavior of f', we approximate the bound from Theorem 1 as $\lambda_{\hat{f}}(x_0) \lessapprox \lim_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n} \ln(e^{\lambda_f} + |\, 2\sqrt{\epsilon}(1+c) + 6rc \,|)$.

From this, a bound clearly follows $\lambda_{\hat{f}}(x_0) \lessapprox \lambda_f + \lim_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n} \ln(1 + e^{-\lambda_f} \,|\, 2\sqrt{\epsilon}(1+c) + 6rc \,|)$.

Thus, $\lambda_{\hat{f}}(x_0) \lessapprox \lambda_f + \ln(1 + e^{-\lambda_f} \,|\, 2\sqrt{\epsilon}(1+c) + 6rc \,|)$. From this, we can derive the following bound: $\lambda_{\hat{f}}(x_0) \lessapprox \lambda_f + \lim_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n} \ln(1 + e^{-\lambda_f}(2\sqrt{\epsilon}(1+c) + 6rc)) = \lambda_{\hat{f}}(x_0) \lessapprox \lambda_f + \ln(1 + e^{-\lambda_f}(2\sqrt{\epsilon}(1+c) + 6rc))$ □.

Corollary 2. $\lim_{\epsilon, r \to 0} \lambda_{\hat{f}}(x_0) = \lambda_f(x_0)$ for all $x_0 \in \mathcal{X}$.

The N-dimension case is more complex, but we mention an abridged version of the proof. First, we obtain a bound on the Jacobians of f by using the Mean Value Theorem on each output dimension. This comes out to $J_{\hat{f}}(x)_i \leq J_f(x)_i + b$, where $b = (2\sqrt{\epsilon}(1+c) + 6rc)(1 + \frac{r}{\epsilon})$. Assuming all the Jacobians are invertible, we factor the product of the bounded Jacobians as $\prod_i^n J_i(I + bMJ_i^{-1})$, where M is a matrix s.t the norm of each row is ≤ 1. Using the fact that the eigenvalues of any matrix AB are equal to those of BA, rearrange the terms and use the definition of a Lyapunov exponent to get the bound $\lambda_{\hat{f}} \leq \lambda_f + \lim_{n \to \infty} \frac{1}{n} \ln \lambda_{max}(\prod_i^n (I + bMJ_i^{-1}))$, where λ denotes the eigenvalues of a matrix. Observe the following:

1) $\lambda(J^{-1}) = \lambda(J)^{-1}$,
2) $\lambda(I + A) = \lambda(A) + 1$,
3) $\lambda_{max}(M) \leq \|M\|$ for the induced norm,
4) and the induced norm of M is less than or equal to N, because the norm of each row is no greater than 1.

Using these observations and approximating $\lambda(J_i^{-1}))$ as $e^{-\lambda_{f_{min}}}$, we find that $\lambda_{\hat{f}} \lessapprox \lambda_f + \ln(1 + bNe^{-\lambda_{f_{min}}})$ $\square$.

The bound on the gradient $\hat{f}'$ tells us that the chaotic behavior of our model is limited by both the distance between data points in the training set and the error on the training set. For transfer learning using small datasets in the target domain, where the gaps between data points can be large and small errors are likely to be accompanied by overfitting, the upper bound on the gradient $\hat{f}'$ can be large. The loose bound on the gradient leads to a loose bound on its Lyapunov exponent, which can lead to divergence. The key insight here is that, to avoid divergence, the focus of the training in small data sets should be on reducing the upper bound on $\hat{f}'$ and not only on reducing the empirical loss on the data. Current transfer learning methods, such as fine-tuning of neural networks, are designed according to the empirical risk minimization (ERM) principle. They typically aggressively reduce the empirical loss without considering the fact that that would lead to higher derivatives of learned function $\hat{f}$, which would lead to higher Lyapunov exponents and divergent behaviors in open-loop control.

An important note here is the trade-off between chaotic behavior and overfitting. $\sqrt{\epsilon}$ goes to zero slower than r, and so dominates the bound when r is small. Although it is possible to aggressively fit the model to reduce ϵ to close to zero, this risks severe overfitting, which causes divergence even with non-chaotic networks. Thus, we want a transfer method that avoids chaotic behavior even when ϵ is fairly large, to avoid this tradeoff as much as possible.

V. Algorithm

In target domain, the upper bound on the Lyapunov exponent of a transferred model $\hat{f}_T$ is likely to be loose, due to large gaps between data points in $\mathcal{D}_T$ and high empirical errors. However, the original model $\hat{f}_S$ from the source domain is likely to have a tight upper bound on $\lambda_{\hat{f}_S}$ since it is trained with a dense data set $\mathcal{D}_S$. This leads us to an interesting question: When transferring a model, is it possible to keep the Lyapunov upper bound from the source domain, while still adapting the model to the target domain?

We begin investigation of this question with a few observations. Firstly, we note that fine-tuning our transferred model on the new data would do nothing to penalize the network for high derivatives in the large spaces between data points. This can mean that we lose the bound given by the source dataset and instead get the much weaker bound from the target dataset. Although it may be possible to construct a loss function that directly penalizes the steep gradients that lead to divergence, this would involve taking a second derivative and calculating the Hessian, which is time-intensive. Instead, we propose an efficient transfer method that is guaranteed to preserve in the target dynamics the Lyapunov bound given in the source domain. We present two algorithms for transfer designed with this constraint in mind.

Firstly, we present trajectory fine-tuning (Algorithm 1). Instead of fine-tuning to predict the next state in the sequence at each step, we apply the loss over an entire sequence. This penalizes models that would have a high step-wise accuracy, but also high Lyapunov exponents that lead to large errors when used to predict a trajectory. Since the network predicts the change in the state rather than the state itself, each prediction is of the form $x_{i+1} = \hat{f}(x_i) = x_i + h(x_i)$ where h is a neural network. This gives the network a ResNet structure that allows us to train over full trajectories while avoiding the issue of vanishing gradients, even though the network has no explicit memory. Since dropout causes some variation in the output space, this method of trajectory training also has the effect of exploring a broader input space than would be given by the data alone.

This method is not novel for systems with memory, it has been used previously for partially observable environments for example. The interesting result here is that even for Markovian systems, it performs better than step-wise fine-tuning methods.

Function `Trajectory_Prediction(`f,x_0`)`**:**
  f: transition model
  x_0: start state
  for i *in range(*n*)* **do**
    $x_{i+1} = f(x_i)$
  return $[x_0, x_1, \ldots, x_n]$

Algorithm 1: Trajectory Fine-tuning

1 $\hat{f}_S$: transition model trained in the source domain;
2 x_0: start state;
3 $x^* = [x_0, x_1^*, \ldots, x_n^*]$: ground-truth trajectory from the target domain;
4 $\hat{f}_T \leftarrow \hat{f}_S$;
5 **while** *Training* **do**
6 $[x_0, x_1, \ldots, x_n]$ = `Trajectory_Prediction` $(\hat{f}_T, x_0)$;
7 loss = $\sum_{i=0}^{n} \frac{1}{i}MSE(x_i, x_i^*)$;
8 update weights($\hat{f}_T$, loss);
9 Return $\hat{f}_T$: transition model for the target domain;

Since our model predicts changes in state space, the final state is of the form $x_n = x_0 + \sum_{i=0}^{n} h(x_i)$. This means that the gradient backpropagated from a given x_i to h scales with the length of the trajectory, leading to a stronger gradient signal from the later states. To account for this assymmetry, we weight the loss from each state with a factor of $\frac{1}{i}$.

The second approach, explained in Algorithm 2, instead avoids the divergence problem entirely by preventing the new network $\hat{f}_T$ from having any feedback to itself. Instead, the transferred model $\hat{f}_S$ predicts the entire trajectory, and the new network predicts the residual at each step. The final trajectory is given by the predicted states plus a cumulative sum of the predicted residuals, weighted by a discount factor. By explicitly controlling the dependence on previous states with the discount factor, we are able to manually set the Lyapunov exponent of the new network $\hat{f}_T$. Unlike trajectory training, the residual network g is trained using point-wise predictions in a standard supervised fashion.

Algorithm 2: Cumulative Residual Trajectory Prediction

$\hat{f}_S$: transition model trained in the source domain;
g: residual model;
α: weighting factor on $[0, 1]$;
γ: discount factor;
x_0: start state;
$y_0 = 0$;
for i *in range(n)* **do**
  $x_{i+1} = \hat{f}_S(x_i)$;
  $y_{i+1} = g(x_i) + \gamma y_i$;
return $[x_0 + y_0, x_1 + y_1, \ldots, x_n + y_n]$;

Next, we show that the model $\hat{f}_T$ obtained from the cumulative residual method has a Lyapunov exponent no greater than that of the source model $\hat{f}_S$.

Theorem 2. *Let $\lambda_{\hat{f}_S}$ be the maximal Lyapunov exponent of $\hat{f}_S$. Then the maximal Lyapunov exponent of $\hat{f}_T$ returned by Algorithm 2 is equal to* $\max(\lambda_{\hat{f}_S}, \gamma)$.

Proof: We treat x_i and y_i in Algorithm 2 as separate variables in the dynamical system. Let $(x_{i+1}, y_{i+1}) = \hat{f}_T(x_i, y_i)$, where $\hat{f}_T(x_i, y_i) = (\hat{f}_S(x_i), g(x_i) + \gamma y_i)$, $\hat{f}_S(x), g(x)$, and γ are defined as before. Since this is a multi-dimensional system, we first find the characteristic Lyapunov exponents using the following limit:

$$\lim_{n\to\infty} \frac{1}{n} \ln \lambda([J(x_n, y_n)J(x_{n-1}y_{n-1})...J(x_1, y_1)]) \quad (1)$$

where $J(x, y)$ is the Jacobian of $\hat{f}_T(x, y)$, and λ denotes the eigenvalues of a matrix. The Jacobian is thus given by

$$\begin{bmatrix} \frac{\partial \hat{f}_S(x)}{\partial x} & \frac{\partial \hat{f}_S(x)}{\partial y} \\ \frac{\partial g(x)+\gamma y}{\partial x} & \frac{\partial g(x)+\gamma y}{\partial y} \end{bmatrix} = \begin{bmatrix} \hat{f}'_S(x) & 0 \\ g'(x) & \gamma \end{bmatrix}. \quad (2)$$

Taking the limit and using the definition, we see that the Lyapunov exponents are $\lambda_{\hat{f}_S}$ and $\ln \gamma$. If we set γ to be less than or equal to $e^{\lambda_{\hat{f}_S}}$, then we are guaranteed that our system will not be chaotic unless the source model itself is chaotic.

For the higher dimensional case, we instead set γ as e raised to the minimum Lyapunov exponent of $\hat{f}_S$, or 1, whichever is smaller.

The cumulative residual method performs well when the Lyapunov exponent is less than or equal to 0, but cannot accurately model chaotic systems. Trajectory optimization is more general, but makes credit assignment harder, leading to higher noise and variance.

VI. Evaluation

We tested the ability of various transfer methods to predict open-loop trajectories of the Model-T42 adaptive hand [42]. We built two 3D-printed versions of the hand [42], illustrated in Figure 1. As discussed in [14], a sufficient representation of the state of an underactuated hand is an observable 4-dimensional state composed of the object's position and the actuator loads. The hand is controlled through the change of actuator angles, where an atomic action is, in practice, unit changes to the angles of the actuators at each time-step. That is, an action moves actuator i with an angle of $\lambda\gamma_i$, where λ is a predefined unit angle and γ_i is in the range [-1, 1]. In the below experiments, the hand models were trained while manipulating a cylinder with 35 mm diameter. The cylinder is placed in the center and grasped by the two fingers. A long sequence of random actions is then applied on the actuators. We collected from both hands two large data sets of 300 trajectories of 10^3 time-steps each. While the entire data set of the source hand was used to train a transition model, only small percentages of the target's dataset were used for transferring the transition model, as shown in Figure 3.

We used a simple feedforward network for our model, with two hidden layers of 128 neurons each. We used a SELU activation for the first layer and a tanh activation for the second. We found it was important to use an activation function that saturates on the last layer, to prevent the network from predicting arbitrary large movements. The SELU layer was followed by a 10% AlphaDropout rate, and the tanh layer was followed by a 10% dropout.

We compare the following methods: **New Model** – A new model trained from scratch, using only the data from the target hand. **Direct** – The model of the source hand applied directly to the target, without modification. **Naive Fine-tuning** – The source model, fine-tuned only on single-step predictions on the target hand. This is the standard practice and a popular way of transferring a neural network to a new domain. **Trajectory Fine-tuning** (Algorithm 1) – The source model, fine-tuned by optimizing over trajectories to minimize deviation from the true trajectory on the target hand. Unlike naive fine-tuning, this method penalizes error from feedback loops. **Cumulative Residual** (Algorithm 2) – The cumulative structure prevents feedback from dominating the loss. We used $\alpha = 0.3$ and $\gamma = 0.9997$. **Recurrent Residual** – Identical to cumulative residual, except it uses the adjusted state $(x_i + \alpha y_i)$ as an additional input to the model to predict the next step. We introduce this model in order to demonstrate the importance of feedback on the model's performance. We used $\alpha = 0.3$.

We report in Figure 3 how long predicted trajectories of the grasped object's positions were able to stay within a given radius of the true observed positions, using a separate test data set taken from the target hand. Duration here refers to the number of time-steps in the future before the model's predicted position differs from the true measured position by more than a given threshold. We can notice that the predictions of Algorithm 2 (cumulative residual) remain accurate for the longest time, compared to other methods, even when less than 2% of the target hand's data is used. Figure 4 shows the mean square error in the predicted positions of the object. The results confirm again our theoretical analysis.

The second set of experiments corresponds to the *peg-in-the-hole* task illustrated in Figure 1 (a) and the attached video. A hole is placed in the work-space of the target hand, and the cylinder is placed at the center. The learned

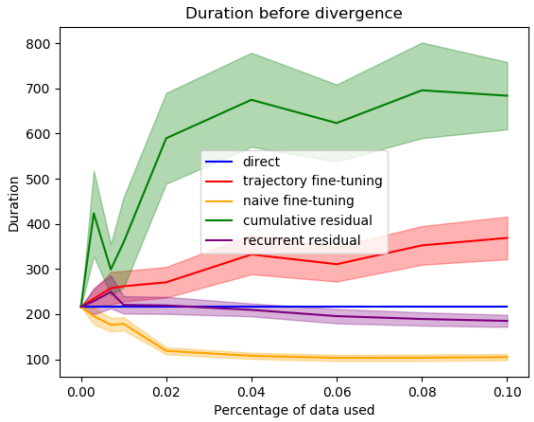


Fig. 3. Average number of time-steps in the future before the predicted path of the object deviates from the true observed path by more than $4mm$, as a function of the data used from the target hand.

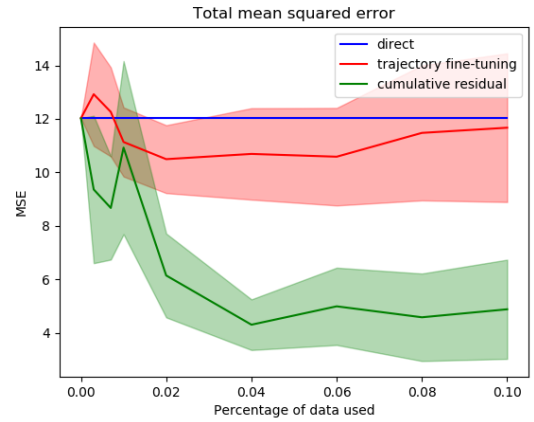


Fig. 4. Average mean square error (MSE) of the predicted position of the manipulated object, in *mm*, as a function of the percentage of data from the target hand used to transfer the transition model from the source hand. We report here the results of only the three most accurate methods, as the MSE of all the other methods was above 15 mm.

transition model is requested with predicting a sequence of actions that would displace the object into above the hole and drop it there without any intermediate feedback during the execution, that is in an open-loop control. The sequences of actions are generated with Monte Carlo sampling during planning. The sequence that is predicted to drop the object closer to the goal at the terminal state is selected for execution. In Figure 5, we compare the models' accuracy at predicting the final position of the manipulated object's center to within a tolerance of 1 cm, which is the radius of the hole minus that of the object. The distance from the start state to the final state is given in terms of steps, each lasting one tenth of a second. The results confirm that the cumulative residuals algorithm is accurate at predicting the final state of a trajectory. Naturally, the prediction degrades as the goal is moved further away from the start state.

Some of the results of these experiments are surprising. For example, the popular naive retraining approach performs much worse than we might normally expect. The key observations here are that methods that train single-step prediction directly (naive retrain and recurrent trajectory transfer) suffer from Lyapunov-caused divergence, performing much worse than the source model without any transfer training. Models that leverage their variation to explore more of the input space (retrain), or keep the source's Lyapunov exponent intact, (trajectory transfer) perform much better.

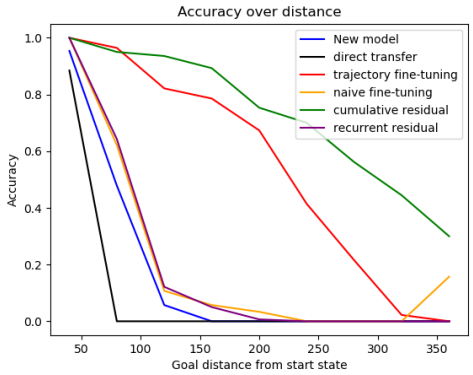


Fig. 5. *Peg-in-the-hole* experiment: Average success rate in predicting a sequence of actions that moves the object from the center to the hole, as a function of the distance from the initial position to the hole's position.

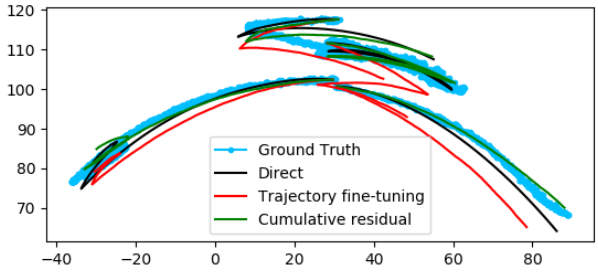


Fig. 6. Examples of predicted trajectories of an object manipulated by the hand in Figure 1 (b) in the x-y plane. The trajectories are predicted with different transfer methods.

VII. Conclusion

Open-loop control of under-actuated robotic hands is a challenging problem due to the difficulty of learning long-horizon predictive models. Transfer learning offers a promising direction to obtain accurate models without intensive data collection. In this work, we analyzed the Lyapunov exponents of predictive transition models, and provided a simple and efficient algorithm for transferring such models across robotic hands. Transfer learning with the proposed algorithm was successfully demonstrated on real robotic hands. Future research directions include integrating the transferred models with more efficient motion planners for in-hand manipulation, and extending the proposed algorithm to transferring policies in addition to transition models.

References

[1] K.-T. Yu, M. Bauzá, N. Fazeli, and A. Rodriguez, "More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing," *IEEE/RSJ Int. Conf. on Intel. Rob. and Sys.*, pp. 30–37, 2016.

[2] T. Laliberté and C. M. Gosselin, "Simulation and design of underactuated mechanical hands," *Mech. and Mach. The.*, vol. 33, no. 1-2, pp. 39–57, Jan 1998.

[3] L. U. Odhner and A. M. Dollar, "Dexterous manipulation with underactuated elastic hands," in *IEEE Int. Conf. on Rob. and Aut.* IEEE, May 2011, pp. 5254–5260.

[4] A. Rocchi, B. Ames, Z. Li, and K. Hauser, "Stable simulation of underactuated compliant hands," in *ICRA*, May 2016, pp. 4938–4944.

[5] J. Borras and A. M. Dollar, "A parallel robots framework to study precision grasping and dexterous manipulation," in *IEEE Int. Conf. on Rob. and Aut.* IEEE, May 2013, pp. 1595–1601.

[6] I. Hussain, F. Renda, Z. Iqbal, M. Malvezzi, G. Salvietti, L. Seneviratne, D. Gan, and D. Prattichizzo, "Modeling and Prototyping of an Underactuated Gripper Exploiting Joint Compliance and Modularity," *IEEE Rob. and Aut. Let.*, vol. 3, no. 4, pp. 2854–2861, Oct 2018.

[7] D. Prattichizzo, M. Malvezzi, M. Gabiccini, and A. Bicchi, "On the manipulability ellipsoids of underactuated robotic hands with compliance," *Rob. and Aut. Sys.*, vol. 60, no. 3, pp. 337 – 346, 2012.

[8] G. Grioli, M. Catalano, E. Silvestro, S. Tono, and A. Bicchi, "Adaptive synergies: An approach to the design of under-actuated robotic hands," in *IEEE/RSJ Int. Conf. on Intel. Rob. and Sys.* IEEE, Oct 2012, pp. 1251–1256.

[9] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *J. of Intel. & Rob. Sys.*, vol. 86, no. 2, pp. 153–173, May 2017.

[10] J. A. Bagnell and J. G. Schneider, "Autonomous helicopter control using reinforcement learning policy search methods," in *IEEE Int. Conf. on Rob. and Aut.*, vol. 2, May 2001, pp. 1615–1620.

[11] S. Zhu, D. Surovik, K. E. Bekris, and A. Boularias, "Efficient model identification for tensegrity locomotion," in *IEEE International Conference on Intelligent Robots and Systems , IROS, Madrid, Spain*, 2018.

[12] S. Zhu, A. Kimmel, K. E. Bekris, and A. Boularias, "Fast model identification via physics engines for improved policy search," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI), Stockholm, Sweden*, 2018.

[13] A. Boularias, J. A. Bagnell, and A. Stentz, "Learning to manipulate unknown objects in clutter by reinforcement," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, 2015, pp. 1336–1342. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9360

[14] A. Sintov, A. S. Morgan, A. Kimmel, A. M. Dollar, K. E. Bekris, and A. Boularias, "Learning a state transition model of an underactuated adaptive hand," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1287–1294, April 2019.

[15] A. Kimmel, A. Sintov, B. Wen, A. Boularias, and K. Bekris, "Belief-space planning using learned models with application to underactuated hands," in *International Symposium on Robotics Research (to appear)*, 2019.

[16] A. Sintov, A. Kimmel, B. Wen, A. Boularias, and K. Bekris, "Benchmarking data-based within-hand manipulation with underactuated adaptive hands," *IEEE Robotics and Automation Letters - Special Issue: Benchmarking Protocols for Robotic Manipulation*, Submitted, Aug. 2019.

[17] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, The MIT Press, 2005.

[18] J. Ko, D. J. Klein, D. Fox, and D. Haehnel, "Gaussian processes and reinforcement learning for identification and control of an autonomous blimp," in *IEEE Int. Conf. on Rob. and Aut.*, April 2007, pp. 742–747.

[19] M. P. Deisenroth, P. Englert, J. Peters, and D. Fox, "Multi-task policy search for robotics," in *IEEE Int. Conf. on Rob. and Aut.*, May 2014, pp. 3876–3881.

[20] R. Paolini, A. Rodriguez, S. S. Srinivasa, and M. T. Mason, "A data-driven statistical framework for post-grasp manipulation," *The International Journal of Robotics Research*, vol. 33, no. 4, pp. 600–615, 2014.

[21] R. Paolini and M. T. Mason, "Data-driven statistical modeling of a cube regrasp," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 2554–2560.

[22] F. Reinhart, Z. Shareef, and J. Steil, "Hybrid analytical and data-driven modeling for feed-forward robot control," *Sensors*, vol. 17, 02 2017.

[23] A. Zerroug, "Chaotic dynamical behavior of recurrent neural network," 2013.

[24] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13. JMLR.org, 2013, pp. III–1310–III–1318. [Online]. Available: http://dl.acm.org/citation.cfm?id=3042817.3043083

[25] D. Sussillo and O. Barak, "Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks," *Neural Comput.*, vol. 25, no. 3, pp. 626–649, Mar. 2013. [Online]. Available: http://dx.doi.org/10.1162/NECO_a_00409

[26] T. Laurent and J. von Brecht, "A recurrent neural network without chaos," *ArXiv*, vol. abs/1612.06212, 2016.

[27] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive Processing*, vol. 12, no. 4, pp. 319–340, Apr. 2011.

[28] N. Makondo, B. Rosman, and O. Hasegawa, "Accelerating model learning with inter-robot knowledge transfer," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 2417–2424.

[29] M. K. Helwa and A. P. Schoellig, "Multi-robot transfer learning: A dynamical system perspective," *CoRR*, vol. abs/1707.08689, 2017.

[30] P. F. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba, "Transfer from simulation to real world through learning deep inverse dynamics model," *CoRR*, vol. abs/1610.03518, 2016. [Online]. Available: http://arxiv.org/abs/1610.03518

[31] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, 2018, pp. 1–8. [Online]. Available: https://doi.org/10.1109/ICRA.2018.8460528

[32] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine, "Learning modular neural network policies for multi-task and multi-robot transfer," *CoRR*, vol. abs/1609.07088, 2016.

[33] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to transfer skills with reinforcement learning," *CoRR*, vol. abs/1703.02949, 2017.

[34] B. Bocsi, L. Csató, and J. Peters, "Alignment-based transfer learning for robot models," in *The 2013 International Joint Conference on Neural Networks, IJCNN 2013, Dallas, TX, USA, August 4-9, 2013*, 2013, pp. 1–7. [Online]. Available: https://doi.org/10.1109/IJCNN.2013.6706721

[35] A. Marco, F. Berkenkamp, P. Hennig, A. P. Schoellig, A. Krause, S. Schaal, and S. Trimpe, "Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with bayesian optimization," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1557–1563, 2017.

[36] M. Hamer, M. Waibel, and R. D'Andrea, "Knowledge transfer for high-performance quadrocopter maneuvers," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 1714–1719.

[37] N. Makondo, B. Rosman, and O. Hasegawa, "Knowledge transfer for learning robot models via local procrustes analysis," *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pp. 1075–1082, 2015.

[38] K. Pereida, M. K. Helwa, and A. P. Schoellig, "Data-efficient multirobot, multitask transfer learning for trajectory tracking," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, p. 1260–1267, Apr 2018. [Online]. Available: http://dx.doi.org/10.1109/LRA.2018.2795653

[39] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[40] C. Wang and S. Mahadevan, "Manifold alignment without correspondence," in *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, ser. IJCAI'09. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009, pp. 1273–1278. [Online]. Available: http://dl.acm.org/citation.cfm?id=1661445.1661649

[41] J. Fu, S. Levine, and P. Abbeel, "One-shot learning of manipulation skills with online dynamics adaptation and neural network priors," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 4019–4026.

[42] R. R. Ma and A. M. Dollar, "Yale openhand project: Optimizing open-source hand designs for ease of fabrication and adoption," *IEEE Rob. & Aut. Mag.*, vol. 24, pp. 32–40, 2017.