Leveraging Forward Model Prediction Error for Learning Control

Sarah Bechtle[1] Bilal Hammoud[1,2] Akshara Rai[3] Franziska Meier[3] and Ludovic Righetti[1,2]

***Abstract*— Learning for model based control can be sample-efficient and generalize well, however successfully learning models and controllers that represent the problem at hand can be challenging for complex tasks. Using inaccurate models for learning can lead to sub-optimal solutions, that are unlikely to perform well in practice. In this work, we present a learning approach which iterates between model learning and data collection and leverages forward model prediction error for learning control. We show how using the controller's prediction as input to a forward model can create a differentiable connection between the controller and the model, allowing us to formulate a loss in the state space. This lets us include forward model prediction error during controller learning and we show that this creates a loss objective that significantly improves learning on different motor control tasks. We provide empirical and theoretical results that show the benefits of our method and present evaluations in simulation for learning control on a 7 DoF manipulator and an underactuated 12 DoF quadruped. We show that our approach successfully learns controllers for challenging motor control tasks involving contact switching.**

I. INTRODUCTION

Data driven model based approaches to learn control have been introduced as an alternative to overcome the limitations of imperfect analytical models of the robotic tasks [6]. In this work, we consider iterative model based learning, where we iterate between learning a forward dynamics model of the robot, and using it to learn a controller. The controller essentially inverts the forward model, computing an action given a current and desired next state. Two challenges arise here: first the quality of the learned forward model is decisive for the success of learning control and second, it might be challenging to learn a controller based on the learned model, in order to perform a task successfully. We present an approach that couples model and controller learning by leveraging forward model prediction error during controller learning. The controller predicts the motor command required to achieve a desired state. The forward model predicts the next state, from the current measured state and motor command predicted by the controller, thus representing the causal relationship of the movement [37]. This connects the models, as the predicted action is used as input to the forward model. From a robotics perspective, forward and inverse models are representations of the physical properties of the robot. In the neuroscience and cognitive science literature [14], [24], the presence of internal models, as representation of the body in the human brain [13] are believed to play an important role. In [39] the authors explain the necessity for a connection between forward and inverse models in the cerebellum by pointing out that acquiring an inverse model purely from motor learning is difficult, since the optimal motor command is not available during learning (otherwise the learning would not be necessary). From a robotics perspective, this argument holds as well, since a desired trajectory is usually defined in the state space and not in the action space.

Following this observation, and unlike the more common approach in the robotics literature, where the forward or the inverse model are trained separately using supervised learning from data (as for example in [2], [7], [25], [26]), we show how connecting the models, and formulating a loss in the state space, improves the performance when learning control. In contrast to other work [16] that considers learning these models together, we show how including the prediction error of the forward model during controller learning creates an unbiased loss signal, that leads to a significant improvement in performance.

In a nutshell, the contributions of this paper are: 1) We explore the effects of connecting controller and forward model during learning control in an iterative fashion on a manipulator and a quadruped. 2) We show, with theoretical and empirical results, how including forward model prediction error during controller learning significantly improves learning a motor control task on a robot. We present manipulation and locomotion experiments, specifically we also show learning of a walking controller that can inherently handle contact switching.

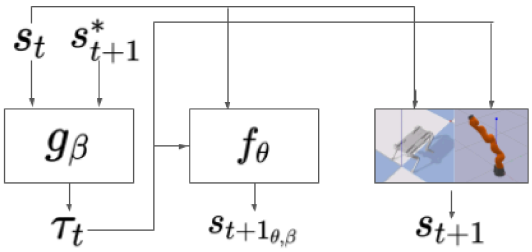


Fig. 1: Overview of connecting inverse and forward models: the motor command, that is the output of the controller, is fed to the forward model used to predict the next state. The motor command is also run on the robot in order to observer the real next state. The learned controller is then updated by taking the gradient of either (2) or (4)

[1]Max Planck Institute for Intelligent Systems, Tübingen, Germany sbechtle@tuebingen.mpg.de

[2]Tandon School of Engineering, New York University, Brooklyn, NY

[3]Facebook AI Research, Menlo Park, CA

This work was in part supported by the European Union's Horizon 2020 research, innovation program (grant agreement 780684 and European Research Councils grant 637935), the National Science Foundation (grants 1825993, 1932187 and 2026479). Sarah Bechtle was in part supported by the International Max Planck Research School for Intelligent Systems.

II. Related Work

A. Coupling forward and inverse models

Wolpert et al. present in [38] an architecture for multiple paired inverse and forward models, the pairs are coupled and trained jointly. The predictions of the forward models determine which inverse model to use. [18] extends this for a manipulator. [30] present results on coupled learning of kinematic models for tool use. In [23] the authors present a deep neural network that structures the learning of a manipulator's dynamics model following Lagrangian mechanics. The trained model can be used for forward as well as inverse dynamics computation, but does not directly connect the models. Most similar to our work is [16], where the authors show the benefits of using a 'distal teacher' for training the inverse model on a 2 link 2D arm. Their approach is based on a stochastic gradient, computed by comparing the observed states with the desired state. In contrast to these approaches, we present an iterative method to train the models jointly. Our experiments are conducted on two different robots, in 3D, and present a loss function that considers the forward model prediction error during controller learning. We show in Section III how our approach mathematically differs from [16], and in Section IV that it achieves significantly better results on higher dimensional systems. In particular, our approach can easily include contact interactions.

B. Using model prediction error for learning

The idea of using model prediction error during learning has been explored within the reinforcement learning literature mostly form the perspective of intrinsically motivated agents. For example, [3], [32], [33] propose rewarding agents to minimize prediction errors of sensory events to explore the state space. This work is limited to low-dimensional and discrete state-and-action spaces. More recently [5], [19], [27], [36] present results on higher dimensional systems, however this work focuses on model free reinforcement learning where the learned models are purely used to provide an additional learning signal to train a policy. In contrast to this work, our approach uses forward model prediction error during learning in a setting where the learned model is actually used to learn a motor control task.

C. Improving model learning in model based approaches

Fewer works have included additional learning signals during model based learning. [31] proposes a measure of disagreement in an ensemble of forward models as an exploration signal. [4] shows that including the predictive uncertainty of the forward model during controller optimization could improve forward model learning. In [22], an empirical measure of learning progress in included on a low dimensional discrete MDP. Similarly, self correcting forward models were proposed in [34], [35] but the considered problem remains low dimensional. While it is widely acknowledged that model quality is of crucial importance in model based approaches, to the best of our knowledge this problem is seldom tackled for high dimensional systems.

D. Learning models including force measurements

Learning models that include non-trivial contact interactions is especially challenging as contacts create discontinuous force measurements and control actions. In [40] the authors use force measurements as an additional input to their model for a manipulator. However, the measurements are not used for controller learning but only to discriminate between different tasks. In [20] multimodal input signals, including forces, are used to train an embedding for a downstream model free reinforcement learning task that takes as input the learned embedding but does not use the learned model during policy learning. Even with accurate physical models, the conception of inverse dynamics controllers is challenging with changing contacts [12] as special care is necessary at each contact transitions, i.e. typically involving manual design of switching events or advanced constraint switching strategies [15]. We show in Section IV how our approach enables to learn a walking controller for a quadruped by including measured contact forces not only as inputs, but also as predictions during controller learning. Importantly, the learned controller seamlessly handles contact switches without any additional assumptions as it learns to predict contact switches using the forward model.

III. Problem Formulation and Approach

The goal of model based learning control is to learn a forward model f of the dynamics of the robot and a controller, or inverse model, g. In general, g can be learned from data but can also be optimized using trajectory optimization algorithms see [4], [9], [21] for a variety of approaches of iteratively learning a model and a controller.

In this work, we propose an algorithm inspired by the concept of connected forward and inverse models, while still being able to iteratively collect data and update the models. We learn a forward model f_θ that performs one step prediction of the form $s_{t+1} = f_\theta(s_t, \tau_t)$, where θ are the parameters of the forward model, s_t and τ_t the state and action at time t. We also learn a controller g_β that predicts $\tau_t = g_\beta(s_t, s^*_{t+1})$, given the current state s_t and the desired state s^*_{t+1}. β are the parameters of the controller and s^* can be the immediate desired next state, or a final goal state. We learn both models from data collected on the robot,while alternating between model learning and data collection. Algorithm 1 shows the training procedure. We create a direct connection between f_θ and g_β by using the action predicted by g as an input to f. Since $s_{t+1\theta,\beta} = f_\theta(s_t, g_\beta(s_t, s^*_{t+1}))$, the next state is a function not only of the parameters of f but also of g. This means that, using $s_{t+1\theta,\beta}$ we can formulate a loss that enables us to compute a gradient to update the parameters β of g.

In Figure 1 the coupling of the forward and the inverse model is illustrated. Using $s_{t+1\theta,\beta}$ has the advantage of representing the actual effect that the action, that was predicted by g has. In contrast to learning g in a supervised fashion from collected data, this approach is conceptually more sound as the correct or desired supervision signal for

the action is usually not available. However the goal of the task, s^*_{t+1}, is available in the state space.

In model based approaches, forward models and controllers are inherently intertwined: during the training phase, the forward model predicts the possible next state, and the controller is learned based on this prediction. The controller is the acting component of the loop, facilitating data collection on the robot that is used to update the models. It becomes clear here, that if the forward model prediction is inaccurate, controller training will fail and the collected data might not be meaningful for the current task. This brings us back to one of the major challenges of model based learning, which is to learn models that are accurate enough to use to act on a robot.

In the next section, we introduce a new loss function as well as other, more standard, losses used as comparison. We propose a loss function for controller learning that ultimately reduces model bias, by including forward model prediction error for learning control. As a result, this improves model prediction and task performance.

A. Learning control via coupled models with joint loss

Our approach (Algorithm 1) alternates between model learning and data collection. g and f are randomly initialized at the beginning of the learning loop. Each iteration collects data using the controller g for the duration of a predefined horizon T. After the roll-out, the collected data is used to update both the forward model f and the controller g.

Algorithm 1 Learning control with Coupled Models

1: $\mathcal{D} \leftarrow$ motor babbling data(s_t, u_t, s_{t+1})
2: $f_\theta \leftarrow$ initialize forward model
3: $g_\beta \leftarrow$ initialize inverse model
4: train model f_θ on $\mathcal{D}$
5: train model g_β on $\mathcal{D}$
6: **while** $i <$ iter **do**
7: $\quad D_{\text{new}} \leftarrow$ rollout g_θ on system(s_t, u_t, s_{t+1})
8: $\quad \mathcal{D} = \mathcal{D} \cup D_{\text{new}}$
9: $\quad$ train model f_θ on $\mathcal{D}$ with Loss from (1)
10: $\quad$ train model g_β on $\mathcal{D}$ with Loss from (2) or (4)
11: **end while**

To update the forward model, we use a regular supervised learning objective representing the model prediction error

$$\mathcal{L}_{sup}(\theta) = (f_\theta(s_t, \tau_t) - s_{t+1})^2 \tag{1}$$

where s_{t+1} is the next state observed on the robot and $f_\theta(s_t, \tau_t)$ is the next state predicted by f.

To learn g_β, we propose a loss function *joint loss* that trades-off actual robot behavior and control performance prediction using the forward model. We compare it with two other, simpler, approaches: one, *task loss* that only improves control performance prediction using the forward model and a supervised approach that does not use the forward model.

1) Comparison - updating g with task loss : The *task loss* computes a learning objective by comparing the prediction of the forward model (that was coupled with the output of g_β): $s_{t+1\theta,\beta} = f_\theta(s_t, g_\beta(s_t, s^*_{t+1}))$ with the desired next state s^*_{t+1}

$$\mathcal{L}_{\textit{task loss}}(\beta) = (f_\theta(s_t, g_\beta(s_t, s^*_{t+1})) - s^*_{t+1})^2 \tag{2}$$

This loss evaluates how well the action of g will be able to achieve the desired state s^*_{t+1} by using f to predict the next state. Intuitively, this will lead to the desired behaviour only if the prediction of the forward model is accurate enough, making the learned controller susceptible to model-bias and inaccuracies.

2) Comparison - updating g with supervised loss: Alternatively, a general supervised learning loss can be used, of the form

$$\mathcal{L}_{\textit{inverse sup}}(\beta) = (g_\beta(s_t, s_{t+1}) - \tau^{run}_t)^2 \tag{3}$$

where s_{t+1} is the observed next state when executing τ^{run}_t on the robot, and τ^{run}_t is the output of $g_\beta(s_t, s^*_{t+1})$. This loss is the most common in the literature, especially for inverse dynamics learning [7], [27]. $\mathcal{L}_{\textit{inverse sup}}(\beta)$ uses the observed data to update the controller. In contrast to the *task loss* and also our *joint loss* , this loss is not goal oriented, but purely tries to learn the state-control relationship by fitting observed data.

3) Updating g with joint loss : Our proposed *joint loss* accounts for the quality of the dynamics model, by adding a term that compares the predicted next state with the actual next state.

$$\begin{aligned}\mathcal{L}_{\textit{joint loss}}(\beta) = (f_\theta(s_t, g_\beta(s_t, s^*_{t+1})) - s^*_{t+1})^2 \\ +(f_\theta(s_t, g_\beta(s_t, s^*_{t+1})) - s_{t+1})^2\end{aligned} \tag{4}$$

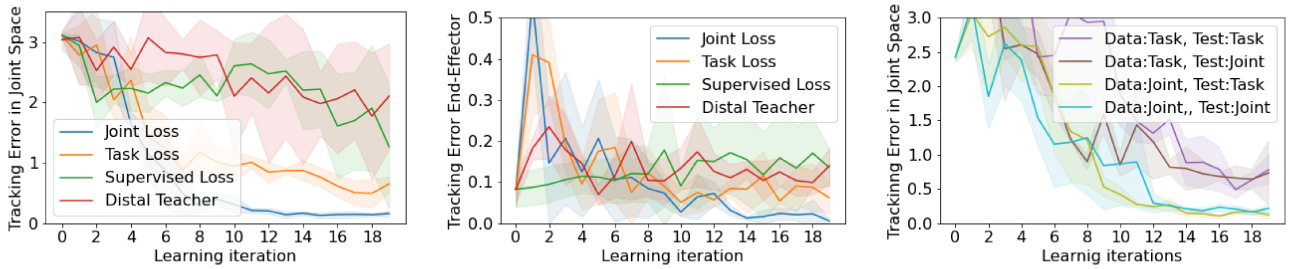
where s_{t+1} is the next state observed on the robot. The *joint loss* thus evaluates not only how well τ_β was able to achieve the desired next state (as predicted by the forward model), but also how good the predictive performance of the forward model actually is. This essentially creates a trade-off between controller and forward model performance, shifting the data distribution seen during roll-out towards a solution that is desirable in reality. In all cases, the parameters of g_β are then optimized with gradient descent by taking the gradient $\nabla_\beta \mathcal{L}(\beta)$.

In the next section, we analyse in details the *task loss* and the *joint loss* . We show why adding the forward model prediction error benefits the controller, and as a consequence, also forward model learning. We then experimentally compare in Section IV these losses with the supervised loss, and show the benefits of our *joint loss* .

B. Theoretical analysis of loss functions

To show the benefit of including the forward model prediction error during inverse model learning let's consider a simplified 1D example: $s_p = f_\theta(s, g_\beta(s, s_d))$. Where s_p is the prediction of the forward model f_θ, s is the current state, s_a is the actual next state observed on the robot and s_d is the desired next state. g_β computes the action for given s and s_d. In order to update parameters β of g the gradients that have to be computed are

$$\nabla_\beta \mathcal{L}_{\textit{task loss}} = 2\frac{\delta f_\theta}{\delta g_\beta}\frac{\delta g_\beta}{\delta \beta}(s_p - s_d) \tag{5}$$



(a) Inverse dynamics learning (b) operational-space controller learning (c) Comparing losses on collected datasets

Fig. 2: Experiments on 7 DoF Kuka arm, the MSE tracking errors (mean and standard deviation over all learning experiments) are reported over learning iterations.

and

$$\nabla_\beta \mathcal{L}_{\textit{joint loss}} = 2\frac{\delta f_\theta}{\delta g_\beta}\frac{\delta g_\beta}{\delta \beta}(2s_p - s_d - s_a) \tag{6}$$

When looking at (5) it becomes clear, that $\nabla_\beta \mathcal{L}_{\textit{task loss}} = 0$ when $s_d = s_p$ which means, when the predicted next state is equal to the desired next state. This is a desirable equilibrium, if the forward model prediction is accurate enough, meaning that the predictions of f are not biased. However, if this is not the case, g reaches its equilibrium given a biased model and converges to the wrong solution. We are going to show in section IV how this model bias can affect negatively the learning performance, even if the forward model keeps being improved.

In the case of (6), the general solution for equilibrium is $s_p = \frac{s_d+s_a}{2}$, which is the average between the desired next state and the measured next state. The special solution $s_p = s_d = s_a$ would be desired. However, since we optimize in an iterative way, if $s_p = \frac{s_d+s_a}{2}$ and we continue optimizing, we can plug the general solution back into $\mathcal{L}_{\textit{joint loss}}$ and we get

$$\begin{aligned}\mathcal{L}_{\textit{joint loss}} &= (\frac{s_a + s_d - 2s_d}{2})^2_{\beta,\theta} \\ +(\frac{s_a + s_d - 2s_a}{2})^2_{\beta,\theta} &= \frac{1}{2}(s_a - s_d)^2_{\beta,\theta}\end{aligned} \tag{7}$$

This means, the loss will reach its global minimum when $s_d = s_a$ which becomes an unbiased loss function. It is worthwhile noting that this loss still carries gradient information for β to further improve the inverse model, and is directly affected, through the forward model, by changes of the inverse model. The general solution, $s_p = \frac{s_d+s_a}{2}$, is a local minimum, that the optimization could get stuck in. However we observe that, because our approach alternates between learning the models and collecting new data, the *joint loss* and its trade-off between controller performance and forward model prediction error, facilitates data collection that allows to reach the global minimum $s_d = s_a$. We show empirical evidence for this hypothesis in section IV. In addition to being an unbiased loss, this loss also now reflects a kind of feedback controller loss, trying to push the inverse model to match the observed data with the desired data. Once $s_a = s_d$ then also the special solution $s_p = s_d = s_a$ holds and in particular also $s_p = s_a$.

C. Comparison distal teacher loss proposed by [16]

In [16] the authors propose a stochastic gradient of the form

$$\nabla_\beta \mathcal{L}_{\,[16]} = \frac{\delta f_\theta}{\delta g_\beta}\frac{\delta g_\beta}{\delta \beta}(s_d - s_a) \tag{8}$$

Here the current gradient of the forward model w.r.t. β is used, but the loss does not carry gradient information, as it is purely specified in terms of the observed and desired data. This is equivalent of formulating a loss of the form

$$\mathcal{L}_{\,[16]} = (s_p - s_d)^2 - (s_p - s_a)^2 \tag{9}$$

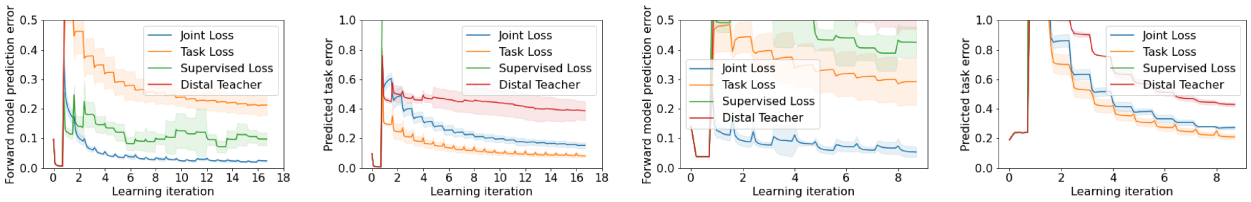
which effectively subtracts the forward model prediction error from the *task loss* error and eventually does not care about the quality of the forward model, as long as the loss between actual and desired next state is decreasing. In simpler scenarios this can have the effect that goal oriented behaviour is achieved even if the forward model is not perfect [16]. On the other hand, this loss does not account for wrong gradients taken through the forward model, that way biasing the solution because of an inaccurate forward model. In practice this seems to be a significant drawback for higher dimensional systems as we show in Sec. IV.

IV. Experiments

In this section, we present experiments to show empirically the benefits of learning control with coupled models and our *joint loss* . We show how our method of including the forward model prediction error during controller learning outperforms all the other methods. We show evidence that including the forward model prediction error leads to a robot behaviour that favours data collection to improve forward model learning and ultimately less biased models. We perform experiments in simulation with a 7DoF iiwa Kuka arm [1] and the 12 DoF quadruped robot Solo [11] (Fig.1). All robots are simulated with PyBullet [10].

A. Experiments with Kuka iiwa

In these experiments, we learn the forward model with an ensemble of probabilistic neural networks, similar to [8]. For the forward model we use three hidden layers with 400 neurons each and ReLU activation functions and an ensemble size of 3. The controller is a neural network with three



(a) Forward model prediction error during learning(walking) (b) Predicted task error during learning(walking) (c) Forward model prediction error during learning(Jump) (d) Predicted task error during learning(Jump)

Fig. 3: Comparing forward model prediction error and predicted task error during inverse model optimization for walking and jumping

hidden layers, with 300, 200 and 100 neurons each and ReLU activation function.

In all these experiments, we compare performance for reaching tasks for five different target positions. Each time, the controller needs to track a desired reaching trajectory (either in joint space or in end-effector space) that is computed in advance, i.e. we learn a tracking controller.

1) Inverse dynamics learning: In this experiments we train the inverse dynamics model of the Kuka arm. The inverse dynamics model $\tau_t = g_\beta(x_t, \ddot{q}^*_{t+1})$ takes as an input the state $s_t = [q_t, \dot{q}_t]$, where q_t are the joint angles and $\dot{q}_t$ the joint velocities at time t, and the desired joint acceleration at the next time step $\ddot{q}^*_{t+1}$ and outputs the torque τ_t. The forward model takes as an input the current state s_t and action τ_t coming from the inverse model. In Fig.2a we can see that training the inverse model with the *joint loss* leads to faster and more stable convergence when compared to the other methods. Notably, it can consistently learn a very good tracking controller in less than 10 iterations.

2) Operational space controller learning with forward model as a disambiguator: In this set of experiments we show how the coupled learning with *joint loss* can also be used to learn a task-space controller [17]. g_β takes as an input the current state $s_t = [q_t, \dot{q}_t]$ and a desired acceleration in end-effector space $\ddot{x}^*_{ee_{t+1}}$. The forward model learns a combination of forward dynamics and kinematics, it takes as an input s_t and τ_t and outputs $\ddot{q}_t$ and $\ddot{x}_{ee_{t+1}}$. The problem of learning an operational space controller is more challenging than learning an inverse dynamics model, since joint redundancy implies that an infinite number of controllers can lead to $\ddot{x}^*_{ee_{t+1}}$. The non-uniqueness of a perfect tracking controller renders learning difficult when done in a supervised way from collected data, since the same input to g can have different output values. In this scenario, using the coupled models approach, we can disambiguate the problem since the forward model's mapping is unique. In Fig.2b, we can see experimentally how our approach outperforms others enabling to consistently learn an operational space controller in a few iterations. It becomes evident here that learning an operational space controller from data in a supervised learning fashion does not perform satisfactory because of the redundancy in mapping torques to end-effector accelerations. Previous approaches to learn operational space controllers have been proposed [28], however they only consider learning in the vicinity of a local model, to force the selection of only one of the many redundancy resolution strategy.

3) Is the joint loss improving data collection?: In model based learning the controller g significantly influences the data seen during learning, since it is the acting component which enables the robot to move and collect more data. Therefore, it is natural to ask whether the performance observed with the *joint loss* is only due to the quality of the data collected during learning. To test this hypothesis, we collect two dataset while running our learning loop with *task loss* and *joint loss* for inverse dynamics learning. After collecting the two datasets, we re-train the inverse and forward models from scratch with both losses on the datasets collected. The results of this experiment are shown in Fig.2c, where we can see that the *task loss* and *joint loss* perform similarly when they are deployed on the exact same data. In particular, both losses perform better when trained with the data collected using the controller optimized using the *joint loss* . This suggests that the data collected while learning with the *joint loss* contains more useful information to accomplish a given motor control task.

B. Experiments on Solo

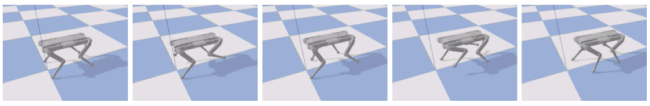
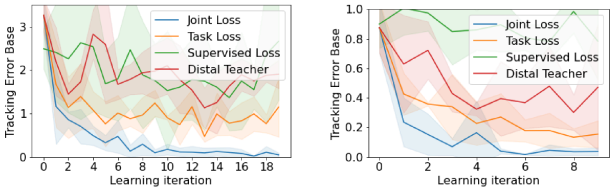


Fig. 4: Image sequence of solo successfully walking

In this section we present experiments performed on the Solo quadruped robot [11]. The forward model is learned with a neural network with three hidden layers of 1000, 500, 500 neurons each and Relu activation function. The input to the forward model is the current torque τ_t and the current state $s_t = [x_{bb_t}, q_t, f_{[x,y,z]_t}, \dot{x}_{bb_t}, \dot{q}_t]$ where x_{bb_t} is the current base pose (position and orientation), $\dot{x}_{bb_t}$ the current base velocity and $f_{[x,y,z]_t}$ are measured contact forces at the four end effectors. The forward model predicts $\Delta s = [\ddot{x}_{bb_{t+1}}\Delta t, f_{[x,y,z]_{t+1}}, \ddot{q}_t\Delta t]$, which is the change in acceleration for the base and joints together with the expected contact forces at $t+1$ when applying τ_t in s_t. The inverse model's architecture is of three hidden layers with 300 neurons each, the input is s_t and s^*_{t+1} which is the desired accelerations of the joints and base together with the desired contact forces at the next time step. We



(a) Solo walking (b) Solo jumping

Fig. 5: Inverse model learning on the quadruped, the tracking error in base position and orientation is reported over iterations. Only the *joint loss* is successful at the task.

compute the desired trajectories (walking and jumping) using the kino-dynamic planner presented in [29]. The goal of these experiments is to show that our approach can also use force measurements and handle hard contact switching for unstable underactuated systems. This is a significantly more challenging task compared to inverse model learning of a fixed base manipulator.

1) Learning to walk: In this set of experiments we show how we can learn to control walking. We average our results over 5 different walking horizons of different length and report the mean and the standard deviation of the experiments. When walking, the robot has to make and break contact with the floor multiple times during the trajectory. Making and breaking contact, and transitioning between these two states, is a challenging task that requires careful control at the moment of contact to avoid slipping and preventing the robot from falling. Contact forces are explicitly included in the state as well during model learning and controller optimization. Thus, the *joint loss* also includes the error on the contact forces, between predicted, desired and observed contact forces.

In Fig. 5a we show the tracking error of the base over learning iterations. We can see clearly here that our *joint loss* outperforms all the other approaches. Qualitatively the controller trained with *joint loss* is the only one that was able to generate stable walking (cf. Fig.4). This also becomes evident when looking at Table I, where we report the tracking error of the ground reaction forces, together with the tracking performance visualized in Fig.6, where we show the predicted (by the forward model), desired and observed ground reaction forces at the front right foot. The controller trained with the *joint loss* (left), tracks the desired forces more accurately than the one trained with *task loss* (right). Also the forward model's prediction of the contact forces, is more accurate in the *joint loss* case. Importantly, the controller is again learned in less than 10 iterations, which makes it amenable to real robot applications.

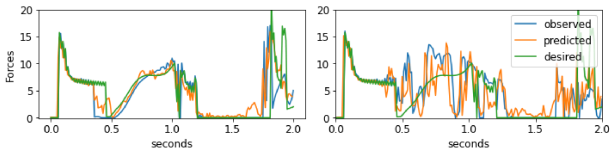


Fig. 6: Tracking of ground reaction forces for *joint loss* (left) and *task loss* (right)

	MSE Tracking Error: mean(std) in N			
	Joint Loss	**Task Loss**	**Distal Teacher**	**Supervised**
forces (walking)	**0.09 (0.1)**	0.16 (0.18)	0.34 (0.35)	0.16 (0.26)
forces (jumping)	**0.02 (0.01)**	0.3 (0.28)	0.38 (0.4)	0.28 (4.3)

TABLE I: Tracking error in ground reaction forces

2) Learning to jump: This experiment shows how Solo can learn to control a jump, which is a task with high impact dynamics and requires precise control especially during take-off (to create the right amount of momentum) and landing (to dissipate the impact). We show the results over 5 different jumping heights in Fig.5b where we can again see that the *joint loss* learns how to successfully accomplish the task. We show again, in Table I the tracking error for the ground reaction forces.

C. Model bias and its effect on performance

When looking at Fig. 3 we can see how the optimization with *task loss* is prone to find sub-optimal solutions due to a biased forward model. In Fig.3a and 3c the prediction error of the forward model during learning is shown for both experiments. We see that the prediction error of the forward model trained while running the experiment using the *joint loss* is lower than the prediction error of the forward model when using the *task loss* . On the other side in Fig.3b and 3d the predicted task error ($(s_{t+1_{\theta,\beta}} - s^*_{t+1})^2$) is shown. For the experiment trained with the *task loss* , the predicted task error is the lowest, however the prediction error of the forward model is high. This means that the model trained with *task loss* is predicting s^*_{t+1}, but the prediction is not correct. This leads to a biased solution, which explains the higher tracking error. Training with the *joint loss* does not result in this scenario, since the prediction of the forward model is accurate, in turn, leading to a better performing controller.

V. CONCLUSIONS

In this work, we show how to leverage forward model prediction error for learning control in an iterative way. Our approach connects controller and forward model learning by using the predicted control signal as an input to the forward model. We show how using forward model prediction error during controller learning results in a learned controller that enables the robot to successfully accomplish non-trivial motor control tasks. We present theoretical and empirical evidence that the improved performance is due to an unbiased loss function, that reduces bias in the learning problem. We also show empirical evidence that when using the controller trained with our *joint loss* on the robot, the collected data is more meaningful for the current task and model learning. This could explain the reduced model bias of the forward model during learning control. In simulation, our approach systematically outperforms other approaches also for contact rich tasks on underactuated, unstable systems and enables learning controllers in a few iterations. In future work, we will extend this work for more complex policy learning and test our approach on real robots.

REFERENCES

[1] KUKA AG. Kuka ag, 2020.

[2] Christopher G Atkeson and David J Reinkensmeyer. Using associative content-addressable memories to control robots. In *Proceedings of the 27th IEEE Conference on Decision and Control*, pages 792–797. IEEE, 1988.

[3] Andrew G. Barto. Intrinsically motivated learning of hierarchical collections of skills. *International Conference on Developmental Learning and Epigenetic Robotic*, pages 112–119, 2004.

[4] Sarah Bechtle, Yixin Lin, Akshara Rai, Ludovic Righetti, and Franziska Meier. Curious ilqr: Resolving uncertainty in model-based rl. In *Conference on Robot Learning*, pages 162–171, 2020.

[5] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[6] Douglas A Bristow, Marina Tharayil, and Andrew G Alleyne. A survey of iterative learning control. *IEEE control systems magazine*, 26(3):96–114, 2006.

[7] Raffaello Camoriano, Silvio Traversaro, Lorenzo Rosasco, Giorgio Metta, and Francesco Nori. Incremental semiparametric inverse dynamics learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 544–550. IEEE, 2016.

[8] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, 2018.

[9] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 2011.

[10] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation in robotics, games and machine learning. `http://pybullet.org/`, 2016–2019.

[11] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, J. Fiene, A. Badri-Spröwitz, and L. Righetti. An open torque-controlled modular robot architecture for legged locomotion research. *IEEE Robotics and Automation Letters*, 5(2):3650–3657, 2020.

[12] Alexander Herzog, Nicholas Rotella, Sean Mason, Felix Grimminger, Stefan Schaal, and Ludovic Righetti. Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid. *Autonomous Robots*, 40(3):473–491, 2016.

[13] Takahiro Ishikawa, Saeka Tomatsu, Jun Izawa, and Shinji Kakei. The cerebro-cerebellum: Could it be loci of forward models? *Neuroscience research*, 104:72–79, 2016.

[14] Masao Ito. Neurophysiological aspects of the cerebellar motor control system. *Int. J. Neurol.*, 7:126–179, 1970.

[15] G. Jarquín, A. Escande, G. Arechavaleta, T. Moulard, E. Yoshida, and V. Parra-Vega. Real-time smooth task transitions for hierarchical inverse kinematics. In *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 528–533, 2013.

[16] Michael I Jordan and David E Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive science*, 16(3):307–354, 1992.

[17] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

[18] Dorothea Koert, Guilherme Maeda, Gerhard Neumann, and Jan Pcters. Learning coupled forward-inverse models with combined prediction errors. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2433–2439. IEEE, 2018.

[19] Adrien Laversanne-Finot, Alexandre Pierre, and Pierre-Yves Oudeyer. Curiosity driven exploration of learned disentangled goal spaces. *arXiv preprint arXiv:1807.01521*, 2018.

[20] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg. Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8943–8950, 2019.

[21] Sergey Levine and Vladlen Koltun. Guided policy search. In *International Conference on Machine Learning*, pages 1–9, 2013.

[22] Manuel Lopes, Tobias Lang, Marc Toussaint, and Pierre-Yves Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. In *Advances in neural information processing systems*, pages 206–214, 2012.

[23] Michael Lutter, Christian Ritter, and Jan Peters. Deep lagrangian networks: Using physics as model prior for deep learning. *arXiv preprint arXiv:1907.04490*, 2019.

[24] R Chris Miall and Daniel M Wolpert. Forward models for physiological motor control. *Neural networks*, 9(8):1265–1279, 1996.

[25] W Miller. Sensor-based control of robotic manipulators using a general learning algorithm. *IEEE Journal on Robotics and Automation*, 3(2):157–165, 1987.

[26] Duy Nguyen-Tuong, Matthias Seeger, and Jan Peters. Computed torque control with nonparametric regression models. In *2008 American Control Conference*, pages 212–217. IEEE, 2008.

[27] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-Driven Exploration by Self-Supervised Prediction. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2017-July:488–489, 2017.

[28] Jan Peters and Stefan Schaal. Learning operational space control. In *Robotics: Science and Systems*, 2006.

[29] Brahayam Ponton, Alexander Herzog, Andrea Del Prete, Stefan Schaal, and Ludovic Righetti. On time optimization of centroidal momentum dynamics. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5776–5782, Brisbane, Australia, 2018. IEEE.

[30] Guido Schillaci, Verena V Hafner, and Bruno Lara. Coupled inverse-forward models for action execution leading to tool-use in a humanoid robot. In *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 231–232. IEEE, 2012.

[31] Pranav Shyam, Wojciech Jaskowski, and Faustino Gomez. Model-based active exploration. *arXiv preprint arXiv:1810.12162*, 2018.

[32] S. Singh, A.G. Barto, and N. Chentanez. Intrinsically motivated reinforcement learning. *18th Annual Conference on Neural Information Processing Systems (NIPS)*, 2004.

[33] Satinder Singh, Richard L. Lewis, Andrew G. Barto, and Jonathan Sorg. Intrinsically Motivated Reinforcement Learning: An Evolutionary Perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010.

[34] Erik Talvitie. Model regularization for stable sample rollouts. In *UAI*, pages 780–789, 2014.

[35] Erik Talvitie. Self-correcting models for model-based reinforcement learning. *arXiv preprint arXiv:1612.06018*, 2016.

[36] Daniel Tanneberg, Jan Peters, and Elmar Rueckert. Intrinsic motivation and mental replay enable efficient online adaptation in stochastic recurrent networks. *Neural Networks*, 109:67–80, 2019.

[37] Daniel M Wolpert, Zoubin Ghahramani, and Michael I Jordan. An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882, 1995.

[38] Daniel M Wolpert and Mitsuo Kawato. Multiple paired forward and inverse models for motor control. *Neural networks*, 11(7-8):1317–1329, 1998.

[39] Daniel M Wolpert, R Chris Miall, and Mitsuo Kawato. Internal models in the cerebellum. *Trends in cognitive sciences*, 2(9):338–347, 1998.

[40] Kevin Zhang, Mohit Sharma, Manuela Veloso, and Oliver Kroemer. Leveraging multimodal haptic sensory data for robust cutting. In *Proceedings of IEEE-RAS International Conference on Humanoid Robots*, October 2019.