Learning Dynamical Systems across Environments

Yuan Yin[1], **Ibrahim Ayed**[1,2], **Emmanuel de Bézenac**[1], **Patrick Gallinari**[1,3]
[1] Sorbonne Université, CNRS, LIP6, Paris, France
[2] Theresis Lab, Thales
[3] Criteo AI Lab, Paris, France
{yuan.yin, ibrahim.ayed, emmanuel.de-bezenac, patrick.gallinari}@lip6.fr

Abstract

Learning the behavior of natural phenomena automatically from the data has gained much traction these last years. However, in most real world scenarios, the environment in which the data samples are acquired is varying and may not be the same for each data sample. This is due to different circumstances e.g. acquisition in different spatial locations, or simply experimental settings which slightly differ. This severely hinders the training process, and makes the standard learning framework inapplicable. In this work, we propose a novel framework for modeling physical systems in this context, where we are able to leverage the data across different environments in order to learn the underlying dynamical systems, ensuring generalization without compromising the model's expressiveness and predictive performance. We instantiate our framework on two different families of dynamical systems, proving that our approach yields superior results over the classical learning approach as well as against competitive baselines. Finally, we also show that we are also able to accelerate and improve the learning for environments that have never been seen before.

Introduction

Often, natural phenomena may be difficult to understand and comprehend due to the complex and nonlinear interactions between composing elements, making it cumbersome to derive a mathematical model describing it. In this context, a data-driven approach arises as a powerful alternative to classical modeling methods, as an unknown model can be learned automatically from the data. Recently, much effort has been focused in this direction (Giannakis and Majda 2012; Mangan et al. 2017), with a particular emphasis on using neural networks (Raissi, Perdikaris, and Karniadakis 2019; Chen et al. 2018; Ayed et al. 2019) for treating cases where the underlying processes are largely unknown. Despite promising results, these methods usually postulate an idealized setting where the data is abundant and the environment in which it is acquired is always the same. However, in practice, this is never the case as obtaining real world data samples may be expensive. Perhaps more importantly, the environment in which they are acquired may vary. These changes can be caused by different factors: For example, in climatic modeling, there are external forces such as the Coriolis force that varies in different spatial locations (Madec et al. 2019), or in cardiac computational model parameters need to be personalized for each patient (Neic et al. 2017).

The classical learning paradigm in this context is to treat all the data as independent and identically distributed, thus disregarding the discrepancies between the environments. As this assumption is not valid, it leads to a biased solution and results in an average model that performs poorly. Conversely, one may also choose to avoid making this assumption by splitting the data from different environments and learning one dynamical system per environment, separately. However, this ignores the similarities between environments and would severely affect generalization performance, specifically in settings where per-environment data is limited.

In this work, our goal is to take into account the difference between environments and make use of the similarities across them. Thus, we propose the LEarning Across Dynamical Systems (*LEADS*) framework, a novel learning methodology where the dynamics are decomposed into two components, one *shared* across all environments, and another that takes into account the dynamics that cannot be expressed by the shared component and only those. This allows us to leverage the data from similar environments automatically, without compromising the expressiveness of the model. We demonstrate the effectiveness of our framework on two standard examples of dynamics given by differential equations: the Lotka-Volterra predator-prey model, expressed as an ODE, and the Gray-Scott reaction-diffusion equations, expressed as PDEs. Finally, we also show that our method accelerates and improves learning for similar unseen environments.

Approach

Problem Setting

We consider the problem of learning unknown physical processes with data acquired from different environments. For each environment $e \in E$, we assume that the data is generated from an unknown governing differential equation:

$$\frac{\mathrm{d}X_t}{\mathrm{d}t} = f_e(X_t) \tag{1}$$

defined over a finite time interval $[0, T]$ where the state X is either vector-valued, i.e. we have $X_t \in \mathbb{R}^d$ (Lotka-Volterra equations in the section Experiments) or is a d-dimensional vector field over a bounded spatial domain $\Omega \in \mathbb{R}^k$, i.e. for $t \in [0, T]$ and $x \in \Omega$, $X_t(x) \in \mathbb{R}^d$. As stated above, modifications in the environment have an impact on dynamics of the system and thus the evolution terms f_e are expected to be different. Nevertheless, we do assume they yield some form of similarity between environments: as we will see in the following, this is not a necessary condition for our framework to be applicable, but this is what will allow us to leverage the data from the other environments.

As in Arjovsky et al. (2020), we choose to not discard the information from where the data was collected. We construct our training set with training sample $(e, \{X^{e,i}\}_{i=1,\dots,N_e}) \in \mathcal{D}$. Each sample is thus composed of the environment identifier e as well as a set of trajectories where each $X^{e,i}$, denoting here the i-th trajectory in the environment e, is a function verifying Equation 1.

Related Work

To make the prediction performance invariant across environments, IRM (Arjovsky et al. 2020) aims at finding a classifier that retains the correlations independent of different environments by excluding other spurious environment-related ones. However, in the context of dynamical systems, modeling bias in each environment is as important as modeling the invariant information, as both of them are indispensable for prediction. This makes IRM incompatible with our setting. Spieckermann et al. (2015); Bird and Williams (2019) use RNNs conditioned on an environment code to perform biased learning in different environments. Nonetheless, the similarity between environments are not explicitly exploited as common invariant dynamical information.

In terms of robustness at test time, our formulation with common term is related to Multi-Task Learning (MTL) and Distributionally Robust optimization (DRO). Baxter (2000) suggests that jointly learning related tasks in MLT can potentially result in better generalization than models learned individually from each task. DRO approaches such as Bietti et al. (2019); Staib and Jegelka (2019) suggest that, in general loss minimization, imposing certain norm penalty on neural networks (or other models) can encourage better generalization.

The Proposed Framework: *LEADS*

As the dynamical systems in equation (1) are unknown, we will learn them from the data by parametrizing the evolution terms f_e with neural networks as in Ayed et al. (2019); Chen et al. (2018). The problem now lies in how these terms will be instantiated. We consider decomposing the dynamics in two components one $g \in \mathcal{F}$ shared across environments, and another environment dependent component $h_e \in \mathcal{F}$, such that if $\mathcal{F}$ is large enough, their should exist a couple $(g, h_e) \in \mathcal{F}^2$ such that by their sum, we recover the dynamics for environment e, i.e.

$$\forall e \in E, f_e = g + h_e \tag{2}$$

The general idea here is that as g is the same for each environment it can be learned using all data points, across all environments. However, this decomposition yields a potentially infinite number of solutions, and in particular the trivial solution obtained by setting g to be the null function: in this case, data across environments cannot be leveraged.

In order to avoid the aforementioned trivial solution, we would like the shared function g to explain the dynamics as much as possible, and in turn make the environment dependent function h_e be as small as possible. The following constrained optimization problem embeds this general idea:

$$\min_{g, h_e \in \mathcal{F}} \sum_e \|h_e\|^2 \quad \text{subject to}$$

$$\forall (e, X_t^{e,i}) \in \mathcal{D}, \frac{\mathrm{d}X_t^{e,i}}{\mathrm{d}t} = (g + h_e)(X_t^{e,i}) \tag{3}$$

Let us consider the limit case where the dynamics are the same across environments, i.e. $\forall e \in E, f_e = f$: this objective will then yield as solution the couple $(g = f, h = 0)$, meaning that the common information, which is all there is, will entirely be captured by g as expected. This will benefit its generalization performance as all the data will be used, even those from different environments.

We will now instantiate our method, providing a practical implementation to solve the previous objective. In practice, we do not have access to the data trajectories at every instant t but only to a finite number of snapshots $\{X^i_{k\Delta t}\}_{0 \leq k \leq T/\Delta t}$ at a temporal resolution Δt. We consider the Lagrangian formulation of the proposed objective as our training loss. Instead of comparing the evolution terms as in Equation 3, we directly compare the trajectories induced by these instead[1][2]:

$$\mathcal{L}(g, h, \lambda) = \sum_{e \in E} \left(\frac{1}{\lambda} \|h_e\|^2 + \sum_{i=1}^{N_e} \sum_{k=1}^{K} \left\| X^{e,i}_{k\Delta t} - \tilde{X}^{e,i}_{k\Delta t} \right\|^2 \right) \tag{4}$$

where $\tilde{X}^{e,i}_{k\Delta t} = X^{e,i}_0 + \int_0^{k\Delta t} (g + h_e)(\tilde{X}^i_s)\, \mathrm{d}s$, which are the trajectory states starting from $X^{e,i}_0$ solved by a DE solver with $g + h_e$ up to $t = k\Delta t$. Note that λ is treated as divisor under $\|h_e\|$ rather than a multiplier of the constraints. This is equivalent to optimize the original Lagrangian but more friendly with the gradient-descent-based methods when λ is very large. With an adequate algorithm in practice optimizing g, h and λ, we should arrive at the optimum g and h_e when $\lambda \to +\infty$. However, solving such optimization problem is difficult as a varying λ changes constantly the loss surface, which makes the learning difficult in the context of dynamical system. We therefore treat the λ as a hyperparameter for each experiment, which should not affect the non-nullity of g even though in this case the constraints in Equation 3 will not be perfectly satisfied at optimum.

It is important to note that *LEADS* is actually independent of the choice of the function space $\mathcal{F}$. We choose here neural networks for its expressiveness, in order to validate our framework. One can apply *LEADS* to any feasible function space expressed by other data-driven methods.

[1] Note that both are equivalent when Δt tends to 0.

[2] Directly comparing the (approximate) evolution terms is possible using finite differences, but led to worse results.

Method	L-V ($\#E = 4$)		L-V ($\#E = 10$)		G-S ($\#E = 3$)	
	MSE train	MSE test	MSE train	MSE test	MSE train	MSE test
Env. Indep.	4.79e-1	7.00±1.71 e-1	4.57e-1	5.08±0.56 e-1	1.55e-2	1.43±0.15 e-2
Env. Dep. Sum	6.87e-6	1.26±1.21 e-2	7.32e-6	1.22±1.68 e-2	8.48e-5	6.43±3.42 e-3
LEADS no min.	4.89e-6	3.33±3.14 e-3	3.28e-6	3.07±2.58 e-3	7.65e-5	5.53±3.43 e-3
LEADS	8.15e-6	**2.36±2.16 e-3**	7.63e-6	**1.77±1.58 e-3**	8.60e-5	**3.38±3.31 e-3**

Table 1: Comparison between *LEADS* and baselines for Lotka-Volterra (in 4 and 10 envs.) and Gray-Scott equations (in 3 envs.).

(a) (b)

Figure 1: Comparison between test trajectories (blue) and ground truth (red), shown in phase space. Blue trajectories are predicted by (a) *Env. Dep. Sum* and (b) *LEADS* for Lotka-Volterra in 4 environments (env. 1 to 4 from left to right).

Experiments

We conduct our experiments for two complex nonlinear dynamical systems. The first one is an ODE-driven biological dynamical system, and the second one is a PDE-driven reaction-diffusion model in which we find many complex behaviors.

Lotka-Volterra Equation We consider this classical model (Lotka 1926), frequently used for describing the dynamics of interaction between a pair of predator and prey in an ecosystem. The dynamics follow the equations:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \alpha x - \beta xy, \frac{\mathrm{d}y}{\mathrm{d}t} = \delta xy - \gamma y$$

where x, y are the quantity of the predator and the prey, $\alpha, \beta, \gamma, \delta$ define how two species interact. In fact, by a proper rescaling one can absorb β and δ into x and y. We therefore leave β, δ constant by setting $\beta = \delta = 1$ across all environments and let α, γ depend on the environments. The nonlinear interaction between two species are therefore non-environment component and the linear terms are linked to environments.

We thus define the parameter $\theta_e = (\alpha_e, \gamma_e)$ for each environment e. Note that choosing θ_e determines consequently the second fixed point of the system (γ_e, α_e), around which the trajectories orbit. The system state is $X_t = (x_t, y_t)$. The initial conditions are fixed across the environments, i.e. $\forall e, X_0^{e,i} = X_0^i$. Starting from the same initial condition $X_0^i = (x_0^i, y_0^i)$, we simulate only 1 trajectory per environment for training and 32 for test. Note that the test set is much larger than the training one. The step size is $\Delta t = 0.5$ and the dataset horizon is $T = K\Delta t = 10$. The experiments are conducted in 4 and 10 environments.

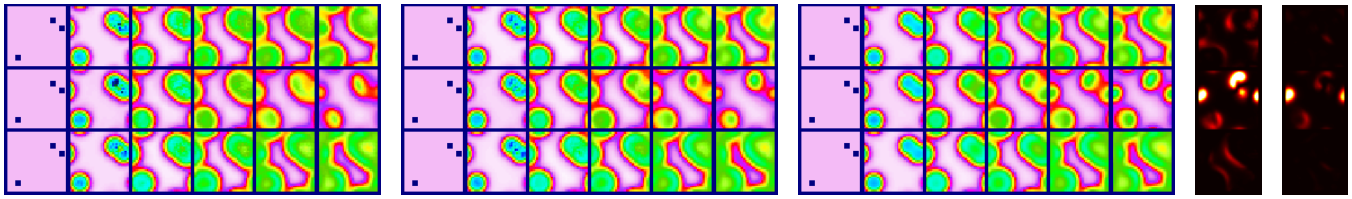
Gray-Scott Equation This reaction-diffusion model is famous for its Turing patterns and complex behaviors w.r.t its simplistic equation (Pearson 1993). The governing PDE is:

$$\frac{\partial u}{\partial t} = D_u \Delta u - uv^2 + F(1-u)$$
$$\frac{\partial v}{\partial t} = D_v \Delta v + uv^2 - (F+k)v$$

where $X_t^e = (u_t^e, v_t^e)$ is state in a given spatial domain Ω, with periodic boundary conditions. D_u, D_v denotes respectively the diffusion coefficient for u and v, which are constant (Pearson 1993). F and k together define the type of corresponding patterns and behaviors. This means that the diffusion and reaction terms are respectively non-environment and environment component.

We therefore choose parameters $\theta_e = (F_e, k_e)$ for each environment e to simulate data. Same as the Lotka-Volterra Equation, the initial conditions are shared across environments and we simulate one trajectory per environment for training and 32 trajectories for test. The step size is $\Delta t = 20$ and the horizon is $T = K\Delta t = 200$. The experiments are conducted in 3 environments.

Training Details Within the experiments for each equation, functions g, h are NNs with the same architecture. We use 4-layer MLPs for Lotka-Volterra and 4-layer ConvNets for Gray-Scott. We apply Swish as the default activation function (Ramachandran, Zoph, and Le 2017). These networks are integrated in time using the differentiable solver implemented by Chen et al. (2018). The basic backpropagation through the internals of the solver is used instead. We apply an exponential Scheduled Sampling (Lamb et al. 2016) with exponent at 0.99 to stabilize the training. We use across all experiments Adam optimizer (Kingma and Ba 2015) with the same learning rate of 1×10^{-3} and $(\beta_1, \beta_2) = (0.9, 0.999)$. For the operator norm acting on h_e, we opt for $\max_{i,k} \left\| h_e(X_{k\Delta t}^{e,i}) \right\|^2 / \left\| X_{k\Delta t}^{e,i} \right\|^2$, where the $X^{e,i}$ correspond to training sample trajectories. In order for the estimation of the norm on the test data to not deviate



(a) (b) (c) (d) (e)

Figure 2: Comparison of trajectories from (a) *Env. Dep. Sum* and (b) *LEADS* with (c) the ground truth for Grey-Scott equation. Each row represents an environment. We show the state of channel u at $t = 0, \ldots, 5\Delta T$. They are accompanied by the maps of prediction error at the rightmost timestep by (d) *Env. Dep. Sum* and (e) *LEADS*. The larger the error, the brighter the pixel at the corresponding coordinates.

too much from its norm of the training data, we also penalize the sum of spectral norms of the weight at each layer $\sum_{l=1}^{L} \|W_l^{h_e}\|^2$, an upper bound on the associated Lipshitz constant, as suggested in Bietti et al. (2019).

Baselines We introduce following baselines to compare with the proposed formulation:

- *Env. Indep.*: the sum of two environment-independent neural networks $g + h$, learned with the standard ERM learning principle, as in Ayed et al. (2019)[3],
- *Env. Dep. Sum*: the sum of two environment-dependent NNs $g_e + h_e$.
- *LEADS no min.*: our proposal without norm penalty, equivalent to *LEADS* with $\lambda = +\infty$.

We show the results in Table 1. For Lotka-Volterra systems, we confirm at first that the entire dataset cannot be fit with a single pair of NNs (*Env. Indep.*). Comparing with other baselines, our method *LEADS* reduces nearly 4/5 of the test MSE by *Env. Dep. Sum* and 1/3 of the test MSE by *LEADS no min.* when there are $\#E = 4$ environments. Figure 1 shows the samples of predicted trajectories in test, *LEADS* almost overlaps the ground true trajectory, while *Env. Dep. Sum* underperforms in most environments. When the number of environments is increased to $\#E = 10$, the error cut is over 85% w.r.t *Env. Dep. Sum* and over 40% w.r.t *LEADS no min.*

We observe the same improving tendency for Gray-Scott systems. The error by *LEADS* is around 1/2 of *Env. Dep. Sum* test MSE and 60% of *LEADS no min.* test MSE. In Figure 2(a)-(c), the states obtained with our method is qualitatively closer to the ground truth. With the help of the error maps in Figure 2(d) and (e), we see that at the rightmost end-time frames, the errors are systematically reduced across all environments. This shows that *LEADS* accumulates less errors through the integration, which suggests that *LEADS* alleviates the overfitting on the support.

[3]We have opted for the sum as it allows for a proper comparison with our method.

Adaptation	MSE test at iteration			
	50	250	500	10000
No adapt.	— 0.36 —			
Env. Dep. Sum from scratch	0.23	5.02e-2	0.25	3.05e-3
Env. Dep. Single from scratch	1.65	18.3	8.87e-2	4.13e-3
LEADS boosted Env. Dep. Single	0.73	**2.06e-3**	**1.84e-3**	**1.11e-3**

Table 2: Comparison of different adaptation strategies in 2 new environments of Lotka-Volterra at different iterations.

Learning in Unknown Environments

We demonstrate how the learned invariant dynamics can boost the fitting in new similar environments. We suppose now that we have an invariant function $\hat{g}$ learned with *LEADS* from L-V ($\#E = 4$). We then generate another Lotka-Volterra dataset in new environments E_{new}, still 1 trajectory per environment in training set and 32 in test.

Let us consider the following adaptation strategies:

- *No adapt.*: a sanity check to ensure that the new dynamics cannot be predicted by $\hat{g}$ without further adaptation.
- *Env. Dep. Sum* from scratch: the sum of two environment dependent NNs, trained from scratch with
- *Env. Dep. Single* from scratch: an environment dependent NN, trained from scratch, no boosting by $\hat{g}$.
- *LEADS boosted Env. Dep. Single*: train environment dependent NN h_e boosted by learned $\hat{g}$.

Table 2 contains the adaptation results at training iterations from 50 to 10000. With *No adapt.*, we firstly show that $\hat{g}$ alone is not able to predict in any of these new environments, even if they are closely related to the original ones. At the iteration 50, we observe that three last adaptations perform poorly as expected since they are at early stage of training. As soon as iteration 250, *LEADS boosted Env. Dep. Single* surpasses already the best performance of the training from scratch methods (*Env. Dep. Sum* and *Env. Dep. Single* from scratch) at iteration 10000. This clearly shows that the

learned shared dynamics improves and accelerates the learning in new environments.

Conclusion

We introduce a data-driven framework *LEADS* to learn dynamics from the data that is collected from a set of similar yet different dynamical systems. Demonstrated with two complex families of systems, our framework can significantly improve the test performance in every environment, especially when the number of available trajectories is limited. We finally show that the extracted dynamics by *LEADS* can boost the learning in similar new environments, which leads us towards a more flexible framework for prediction and generalization in new environments.

Acknowledgements

This work was partially funded by Locust ANR-15-CE23-0027 and Chaires de recherche et d'enseignement en intelligence artificielle (Chaires IA), DL4Clim project (PG).

References

Arjovsky, M.; Bottou, L.; Gulrajani, I.; and Lopez-Paz, D. 2020. Invariant Risk Minimization. *arXiv:1907.02893 [cs, stat]* ArXiv: 1907.02893.

Ayed, I.; de Bézenac, E.; Pajot, A.; Brajard, J.; and Gallinari, P. 2019. Learning Dynamical Systems from Partial Observations. *CoRR* abs/1902.11136.

Baxter, J. 2000. A Model of Inductive Bias Learning. *J. Artif. Int. Res.* 12(1): 149–198. ISSN 1076-9757.

Bietti, A.; Mialon, G.; Chen, D.; and Mairal, J. 2019. A Kernel Perspective for Regularizing Deep Neural Networks. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 664–674. Long Beach, California, USA: PMLR.

Bird, A.; and Williams, C. K. I. 2019. Customizing Sequence Generation with Multi-Task Dynamical Systems. *CoRR* abs/1910.05026.

Chen, R. T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural Ordinary Differential Equations. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31, 6571–6583. Curran Associates, Inc.

Giannakis, D.; and Majda, A. J. 2012. Nonlinear Laplacian spectral analysis for time series with intermittency and low-frequency variability. *Proceedings of the National Academy of Sciences* 109(7): 2222–2227. ISSN 0027-8424. doi:10.1073/pnas.1118984109. URL https://www.pnas.org/content/109/7/2222.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Lamb, A.; Goyal, A.; Zhang, Y.; Zhang, S.; Courville, A.; and Bengio, Y. 2016. Professor Forcing: A New Algorithm for Training Recurrent Networks. *arXiv:1610.09038 [cs, stat]* ArXiv: 1610.09038.

Lotka, A. J. 1926. ELEMENTS OF PHYSICAL BIOLOGY. *Science Progress in the Twentieth Century (1919-1933)* 21(82): 341–343. ISSN 20594941. URL http://www.jstor.org/stable/43430362.

Madec, G.; Bourdallé-Badie, R.; Chanut, J.; Clementi, E.; Coward, A.; Ethé, C.; Iovino, D.; Lea, D.; Lévy, C.; Lovato, T.; Martin, N.; Masson, S.; Mocavero, S.; Rousset, C.; Storkey, D.; Vancoppenolle, M.; Müeller, S.; Nurser, G.; Bell, M.; and Samson, G. 2019. NEMO ocean engine. Add SI3 and TOP reference manuals.

Mangan, N. M.; Kutz, J. N.; Brunton, S. L.; and Proctor, J. L. 2017. Model selection for dynamical systems via sparse regression and information criteria. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 473(2204): 20170009. doi:10.1098/rspa.2017.0009.

Neic, A.; Campos, F. O.; Prassl, A. J.; Niederer, S. A.; Bishop, M. J.; Vigmond, E. J.; and Plank, G. 2017. Efficient computation of electrograms and ECGs in human whole heart simulations using a reaction-eikonal model. *Journal of Computational Physics* 346: 191 – 211. ISSN 0021-9991.

Pearson, J. E. 1993. Complex Patterns in a Simple System. *Science* 261(5118): 189–192. ISSN 0036-8075. doi:10.1126/science.261.5118.189.

Raissi, M.; Perdikaris, P.; and Karniadakis, G. E. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics* 378: 686–707.

Ramachandran, P.; Zoph, B.; and Le, Q. V. 2017. Searching for Activation Functions. *CoRR* abs/1710.05941.

Spieckermann, S.; Düll, S.; Udluft, S.; Hentschel, A.; and Runkler, T. 2015. Exploiting similarity in system identification tasks with recurrent neural networks. *Neurocomputing* 169: 343 – 349. ISSN 0925-2312. Learning for Visual Semantic Understanding in Big Data ESANN 2014 Industrial Data Processing and Analysis.

Staib, M.; and Jegelka, S. 2019. Distributionally robust optimization and generalization in kernel methods. In *Advances in Neural Information Processing Systems*, 9134–9144.