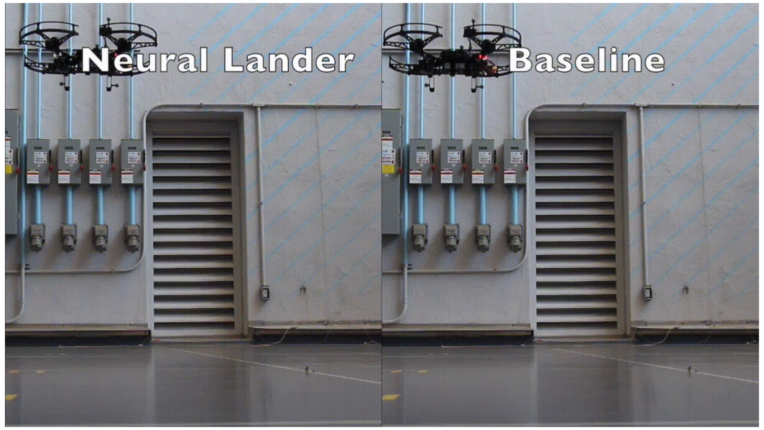
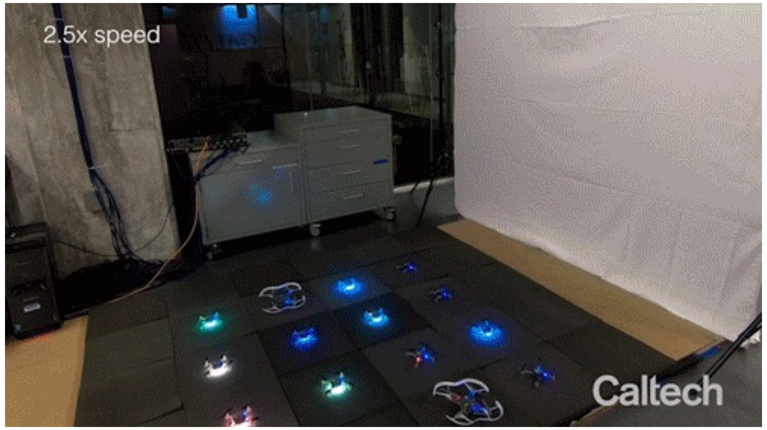
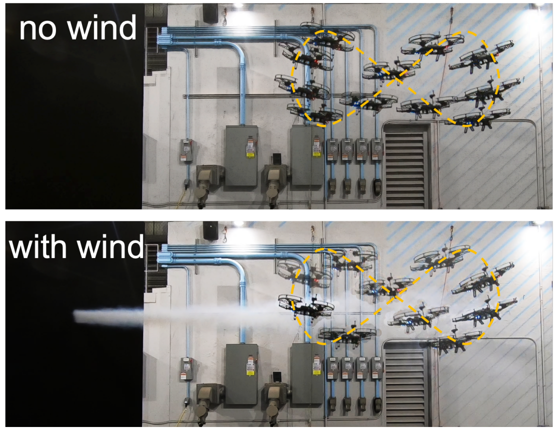
Deep Residual Learning for Control



Shi & Shi & O'Connell et al., ICRA' 19



Shi et al., ICRA' 20, T-RO



O'Connell & Shi et al., ongoing work

Le et al., ICML' 16

"1+1>2"

Zeng et al., RSS' 19, T-RO

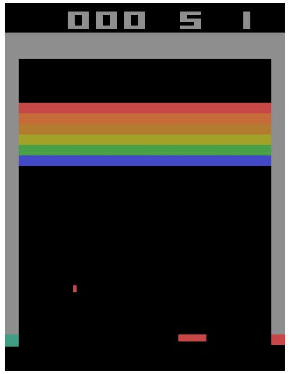
Guanya Shi
www.gshi.me

Caltech CS 159 Guest Lecture

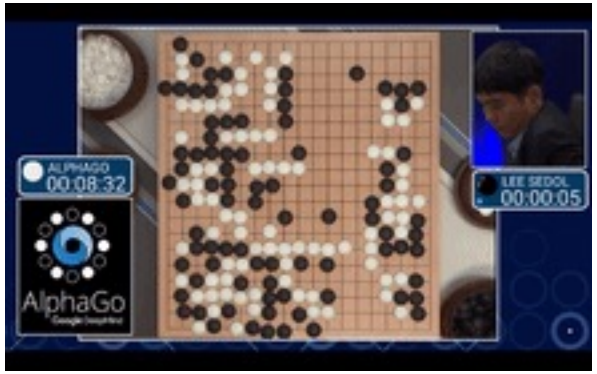
Caltech

Motivations: Real-World Decision Making is Hard

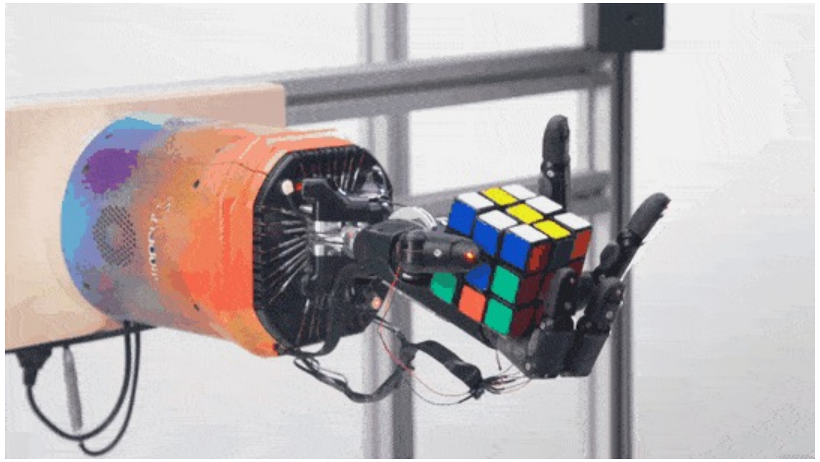
- Q: Can we broaden the successful horizon of deep learning?



Atari

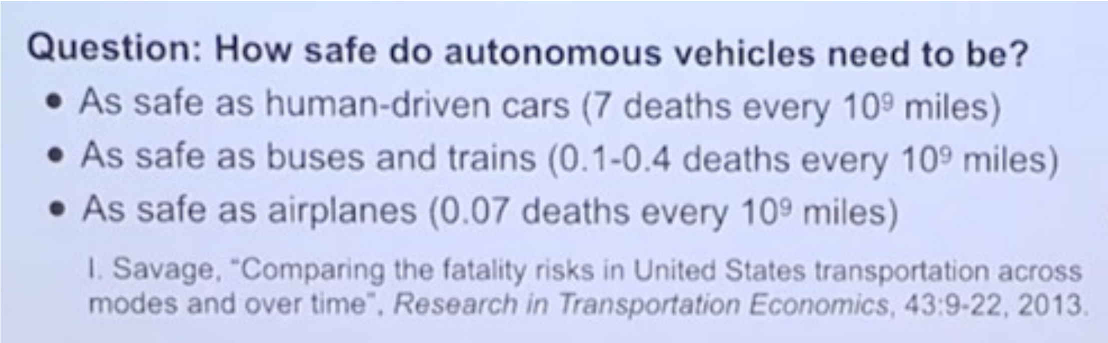


AlphaGo



Manipulator

"I want to use deep learning to optimize the design, manufacturing and operation of our aircrafts. But I need some guarantees." -- Aerospace Director



Question: How safe do autonomous vehicles need to be?

- As safe as human-driven cars (7 deaths every 10^9 miles)
- As safe as buses and trains (0.1-0.4 deaths every 10^9 miles)
- As safe as airplanes (0.07 deaths every 10^9 miles)

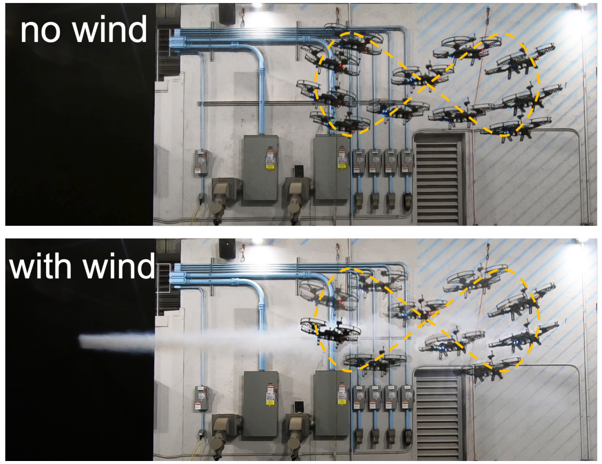
I. Savage, "Comparing the fatality risks in United States transportation across modes and over time", *Research in Transportation Economics*, 43:9-22, 2013.

"Can we really use ML in safety critical system?"

Richard Murray, *IPAM ICLO workshop 2020*

Where are the Challenges from?

- Uncertainties
  - Modelling mismatch, sim2real gap
  - Delay, motor failures, saturations, …

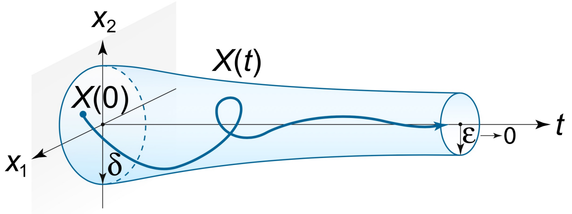


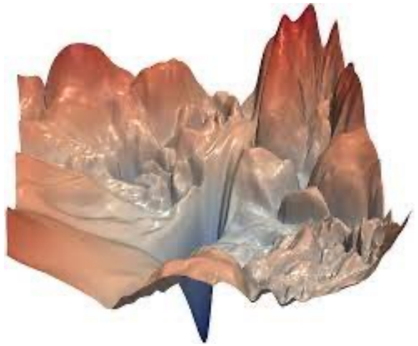
- Limited computational power
  - Data collection is also not cheap

R.I.P., propellers

Crazyflie, weight 34g

- We need some "formal" interpretability
  - E.g., stability, safety
  - Neural networks are "hard" to control





NN landscape, *Li et al., NeurIPS 2018*

Idea: Control Theory Meets Machine Learning

- A typical pipeline in control

System Identification → Dynamical System → Control Synthesis → 

+ Formal guarantees
– System Id is often expensive, inaccurate and hard to generalize
– Control synthesis might be too complex or conservative

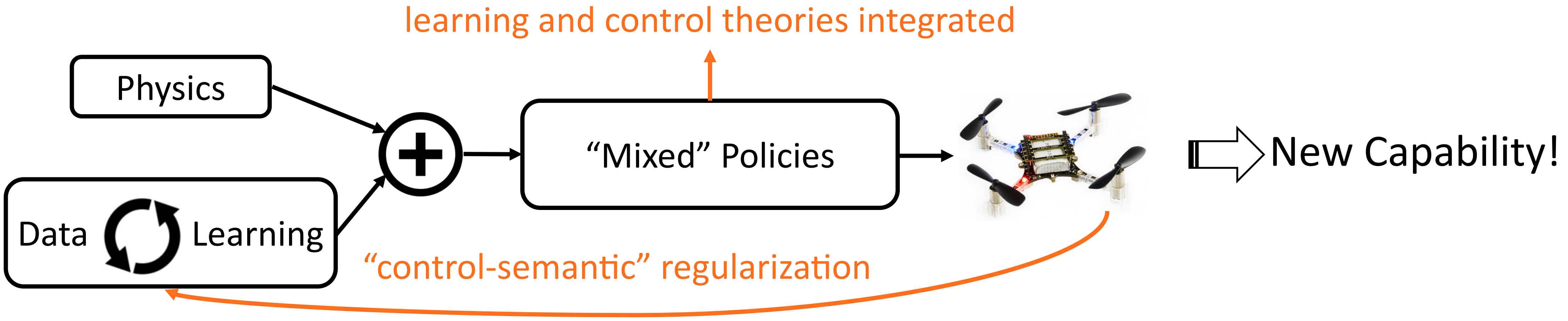
wind tunnel test

- A typical pipeline in learning

Data ⟳ Learning → Policies → 

+ Flexible, generalizable, transferable, …
– No ”formal” interpretability
– DL is data hungry

This Lecture



- Topic I: dynamics-level residual learning
- Topic II: action-level residual learning
- Topic III: program-level residual learning

* Not formal classifications. The boundaries are NOT clear!

Many People Work on this Topic!

Control Meets Learning seminar series

L4DC conference

Advanced Topics in Machine Learning   Control   Learning
CS 159 · Caltech · Spring 2021

AA 203: Optimal and Learning-Based Control
Spring 2021

ESE 680, Fall 2019 – Learning and Control

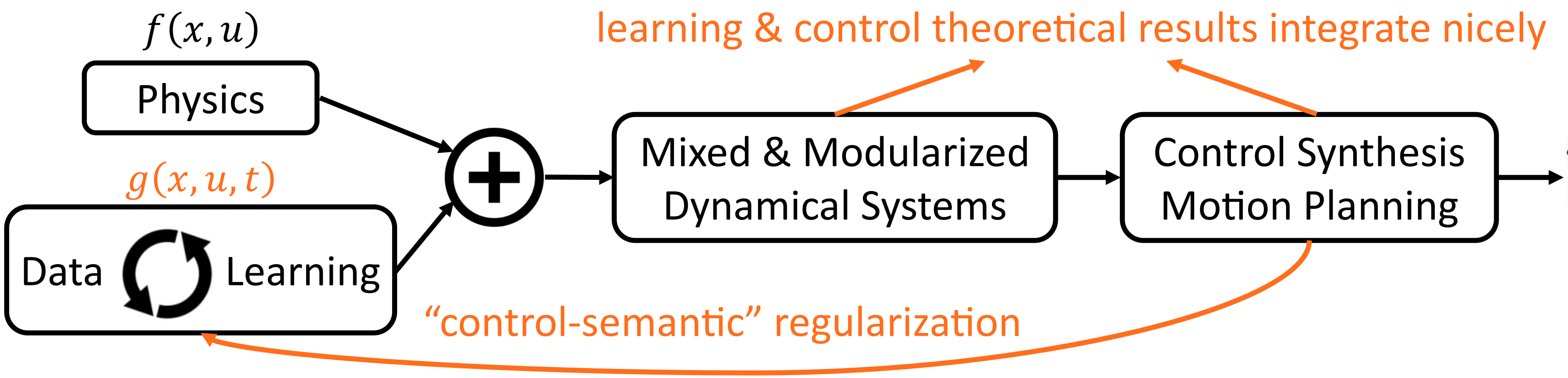
courses: Caltech CS159, Stanford AA 203, UPenn ESE 680, …

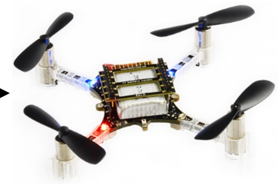
Workshop on Learning for Control
57th IEEE Conference on Decision and Control
Miami Beach, Florida, December 16, 2018

workshops at CDC, NeurIPS, ICML, ICRA, RSS…

Topic I: Dynamics-level Residual Learning

$$\dot{x} = f(x,u) + \boxed{g(x,u,t)} \text{ unknown}$$

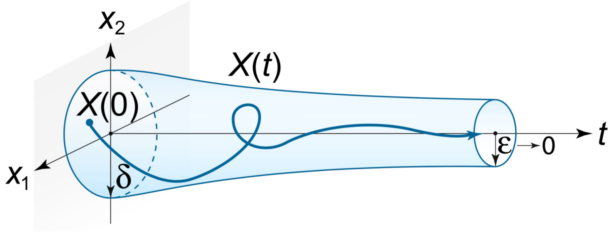




- Key features we pursue:

Generalizability

Guarantees



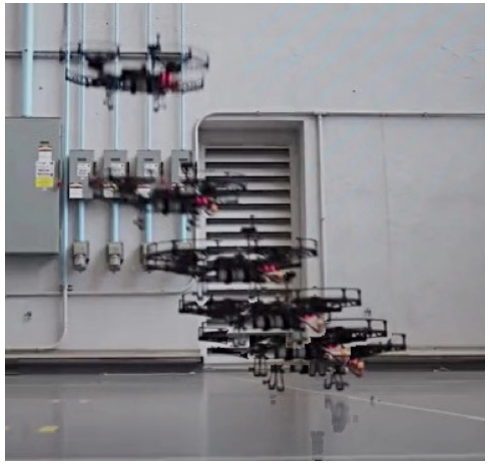
Compact & Efficient Representation

⇨ New Capability! "1+1>2"

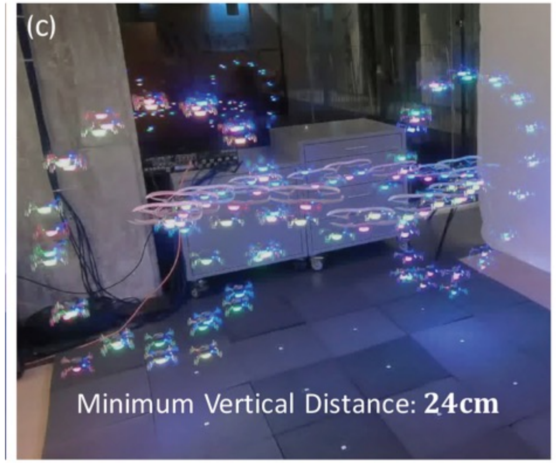
Neural-Lander Family

$$\dot{x} = f(x,u) + \underbrace{\boxed{g(x,u,t)}}_{\text{unknown}}$$

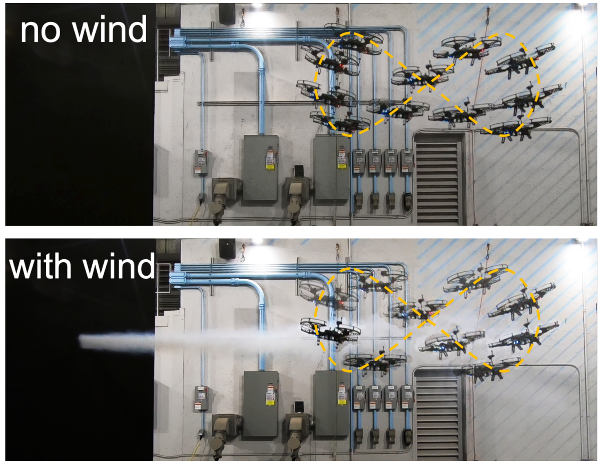
- **(Neural-Lander)** learn g using normalized DNNs
- **(Neural-Swarm)** learn g in heterogeneous swarms
- **(Neural-Fly)** meta-learn g and online fast adapt
- (**Safe Exploration**) safe learn g without experts



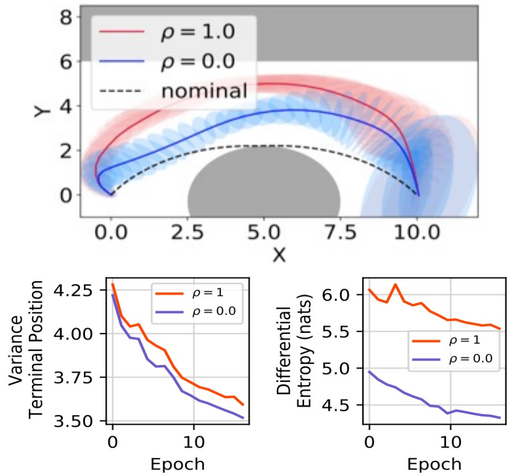
Neural-Lander
***Shi** & Shi & O'Connell et al., ICRA 2019*



Neural-Swarm and N-Swarm2
***Shi** et al., ICRA 2020 and T-RO*



Neural-Fly, *ongoing work*
*O'Connell & **Shi** et al., arXiv preprint*



Safe Exploration
Liu et al., L4DC 2020
Nakka et al., RA-L 2020

Neural-Lander

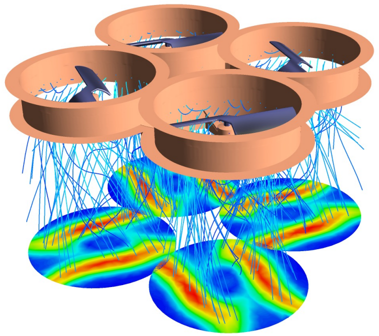
$$\dot{x} = f(x,u) + \underbrace{\boxed{g(x,u,t)}}_{\text{unknown}}$$

(Neural-Lander) learn g using normalized DNNs

(Neural-Swarm) learn g in heterogeneous swarms

(Neural-Fly) meta-learn g and online fast adapt

(Safe Exploration) safe learn g without experts



ground effect

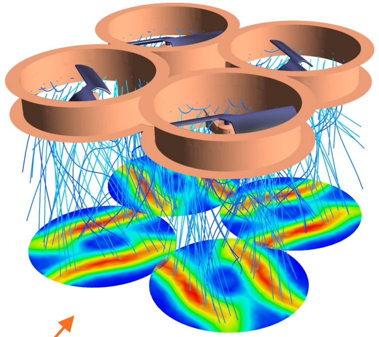
__Shi__ & Shi & O'Connell et al., ICRA 2019

Quadrotor Dynamics

$$\boxed{\dot{x} = f(x,u) + g(x,u)}$$

- $x = [\mathbf{p}, \mathbf{v}, R, \boldsymbol{\omega}], u = [n_1^2, n_2^2, n_3^2, n_4^2]$
- Dynamics $\begin{cases} \dot{\mathbf{p}} = \mathbf{v}, & m\dot{\mathbf{v}} = m\mathbf{g} + R\mathbf{f}_u + \mathbf{f}_a \\ \dot{R} = RS(\boldsymbol{\omega}), & J\dot{\boldsymbol{\omega}} = J\boldsymbol{\omega} \times \boldsymbol{\omega} + \boldsymbol{\tau}_u + \boldsymbol{\tau}_a \end{cases}$

$$\begin{cases} \mathbf{f}_u = [0, 0, T]^\top \\ \boldsymbol{\tau}_u = [\tau_x, \tau_y, \tau_z]^\top \\ \begin{bmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} c_T & c_T & c_T & c_T \\ 0 & c_T l_{\text{arm}} & 0 & -c_T l_{\text{arm}} \\ -c_T l_{\text{arm}} & 0 & c_T l_{\text{arm}} & 0 \\ -c_Q & c_Q & -c_Q & c_Q \end{bmatrix} \begin{bmatrix} n_1^2 \\ n_2^2 \\ n_3^2 \\ n_4^2 \end{bmatrix} \end{cases}$$



much harder to model!

- $f(x,u)$ is the nominal dynamics
- $g(x,u)$ is the **residual** dynamics (e.g., modelling mismatch, ground effect)

Controller Design (Sketch)

A feedback-linearization-style nonlinear controller

$$u = \pi(x, x^d, \hat{\mathbf{f}}_a(x, u))$$

$\hat{\mathbf{f}}_a$ is the learned dynamics (a neural network)

- Challenge I: sample efficiency
  - Learning $f(x,u) + g(x,u)$ needs **1 hour** flight data
  - Learning the residual $g(x,u)$ only needs **~5 mins**
- Challenge II: control allocation
  - Non-affine control synthesis problem ($\hat{\mathbf{f}}_a$ explicitly depends on u)
- Challenge III: stability & robustness in control / generalization in learning

spectrally normalized DNNs

Spectral Normalization and Stability Guarantees

- Key idea: use u_{t-1} in the RHS:

$$\boxed{u_t = \pi(x_t, x_t^d, \hat{\mathbf{f}}_a(x_t, u_{t-1}))}$$

- If the Lipschitz constant of the DNN $\hat{\mathbf{f}}_a$ is upper bounded by γ, we have

tracking error　　control gain　　Lip constant　　approximation error of DNN ($\|\hat{\mathbf{f}}_a - \mathbf{f}_a\|$)

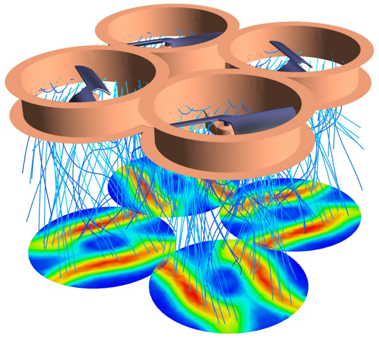
$$\|\mathbf{s}(t)\| \leq \|\mathbf{s}(t_0)\| \exp\left(-\frac{\lambda - L_a\rho}{m}(t - t_0)\right) + \frac{\epsilon_m}{\lambda - L_a\rho}$$

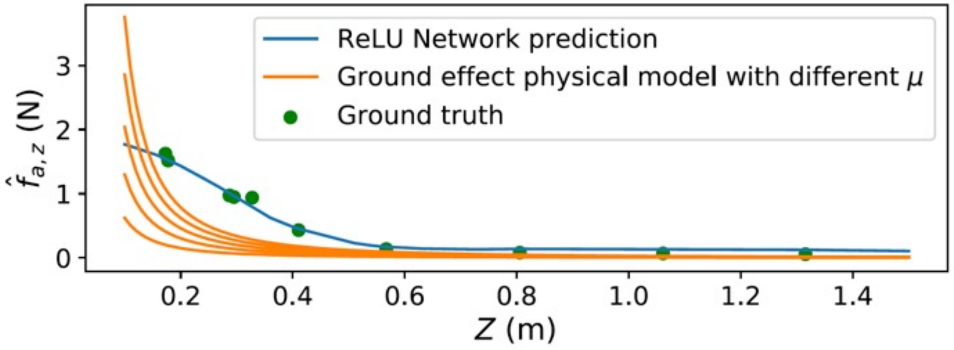
smoothness of the desired trajectory x_t^d

- Thus, we trained the $\hat{\mathbf{f}}_a$ model using Spectral Normalization (SN)
  - *SN of DNNs also leads to *good generalization* (stability in a learning-theoretic sense) [*Bartlett & Foster & Telgarsky, NeurIPS 2017*]

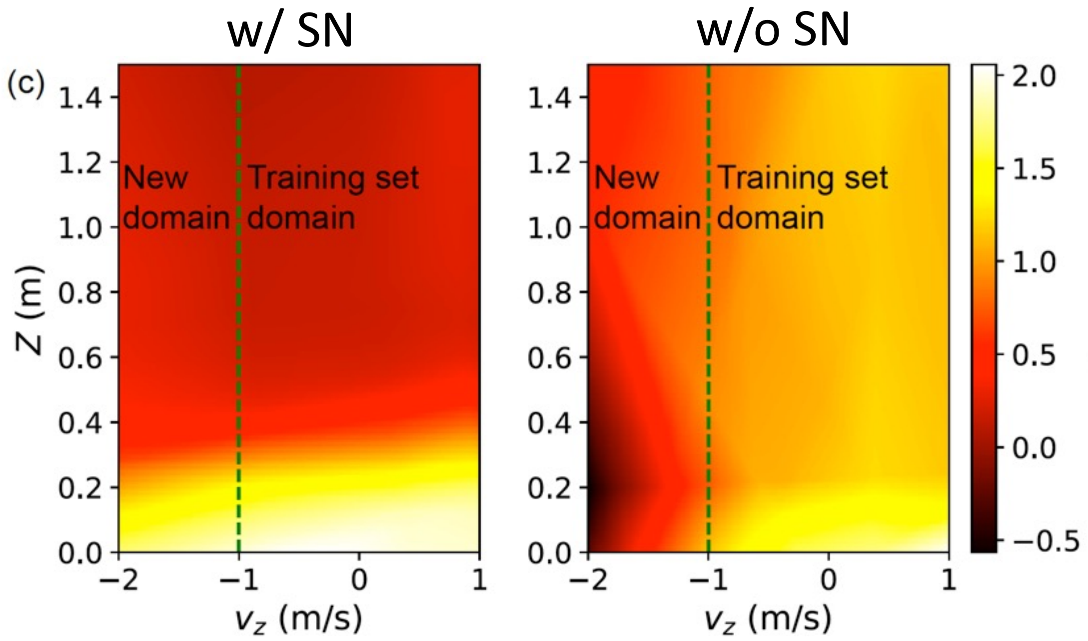
Training Result Visualization

- Training data: manually fly the drone for 5 minutes





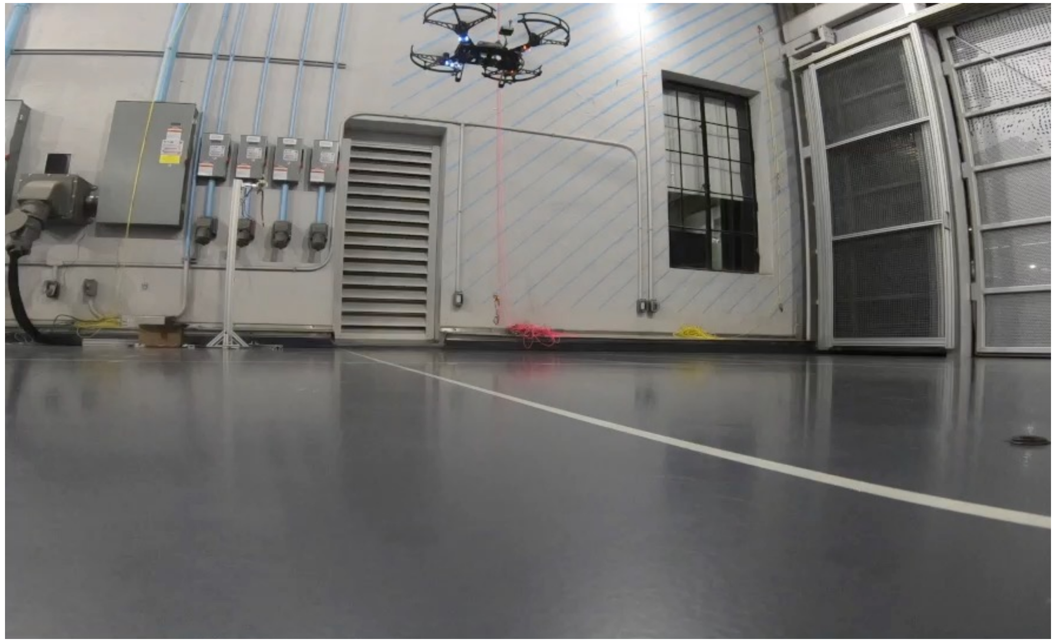
compare with an existing ground effect model



2D heatmap of the learned $\hat{\mathbf{f}}_a$

*Spectrally normalized DNNs can guarantee generalization (*Bartlett et al., NeurIPS 2020; Liu & **Shi** et al., L4DC 2020*)

Trajectory Tracking Performance



“table effect”: need to recollect data and retrain $\hat{\mathbf{f}}_a$

Neural-Lander Takeaways

stability guarantee

Physics

$\hat{\mathbf{f}}_a(x, u)$

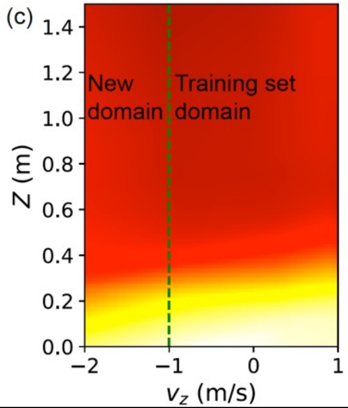
Mixed & Modularized Dynamical System

$u_t = \pi(x_t, \hat{\mathbf{f}}_a(x_t, u_{t-1}))$

spectral normalization

only 5min data

Generalizability



Guarantees

$$\|\mathbf{s}(t)\| \leq \|\mathbf{s}(t_0)\| \exp\left(-\frac{\lambda - L_a\rho}{m}(t - t_0)\right) + \frac{\epsilon_m}{\lambda - L_a\rho}$$

Smooth, Safe, Fast Landing

***Shi** & Shi & O'Connell et al., ICRA 2019*

Neural-Swarm

$$\dot{x} = f(x, u) + \boxed{g(x, u, t)}$$

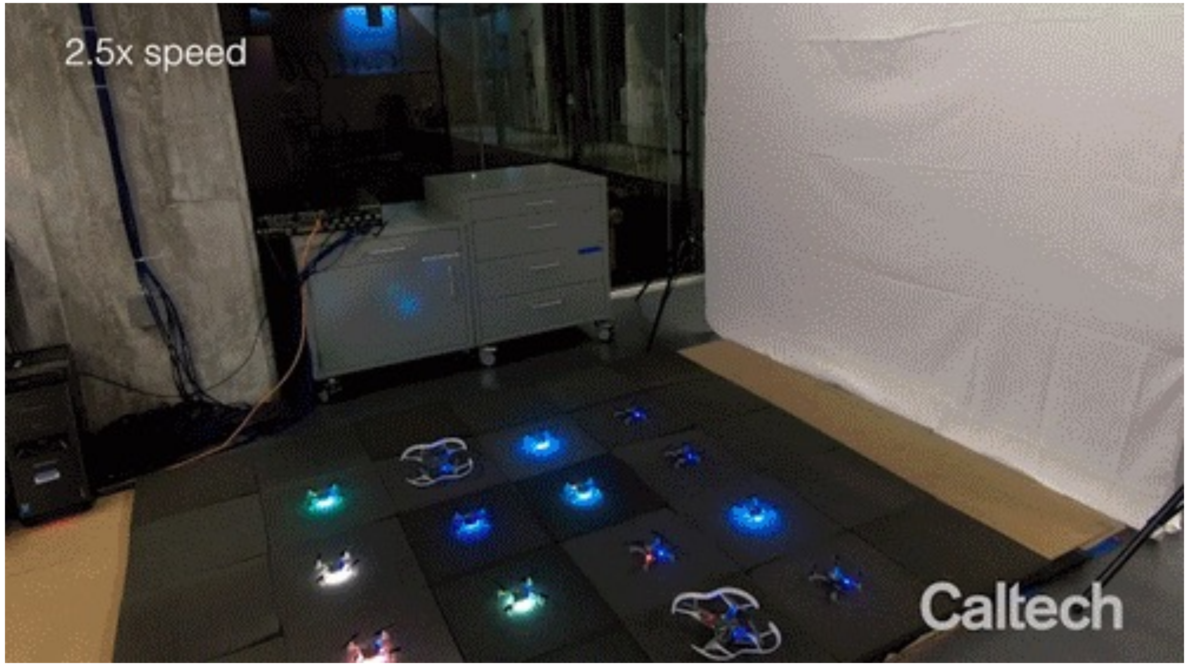
unknown

(Neural-Lander) learn g using normalized DNNs

(Neural-Swarm) learn g in heterogeneous swarms

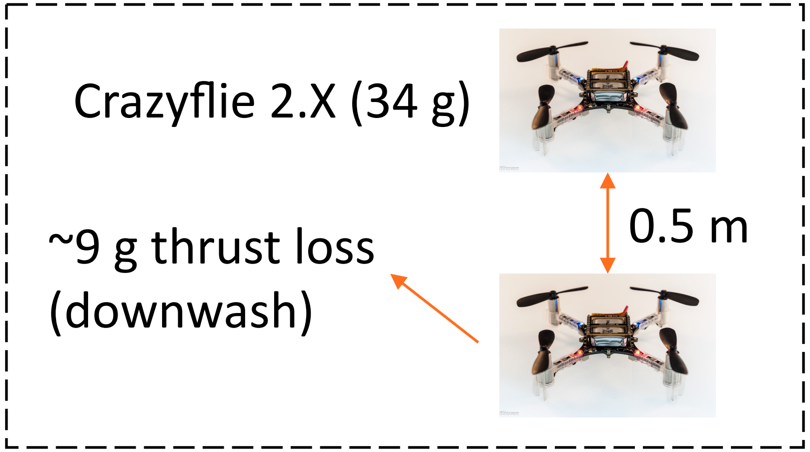
(Neural-Fly) meta-learn g and online fast adapt

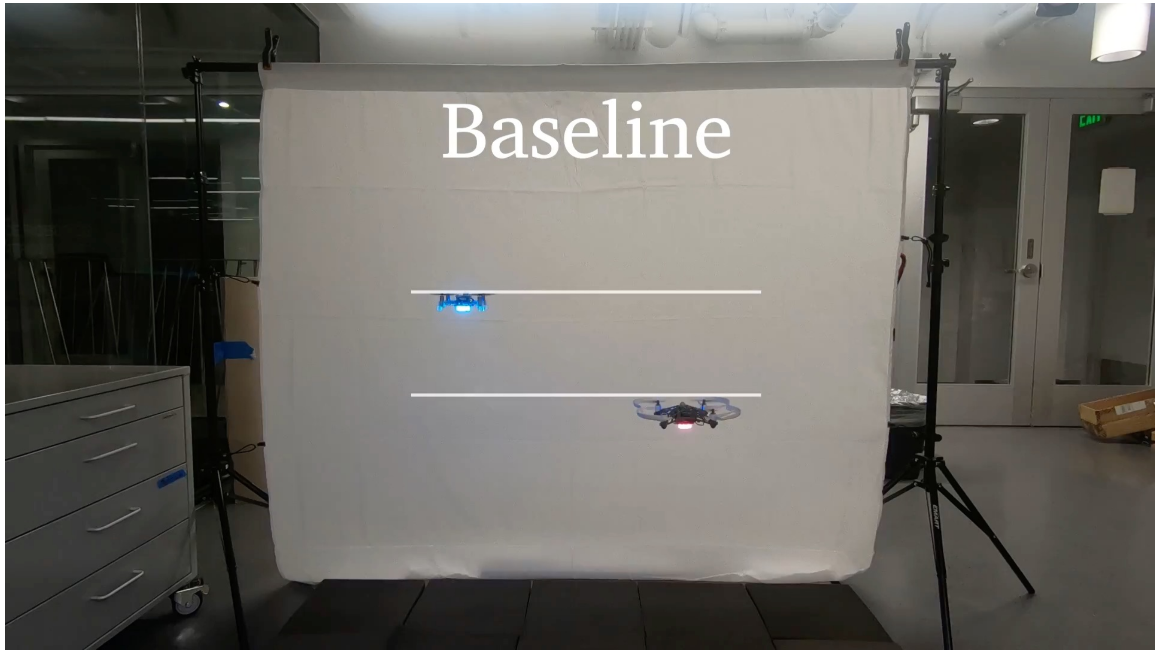
(Safe Exploration) safe learning g without experts



***Shi** et al., ICRA 2020;* ***Shi** et al., accepted by IEEE T-RO*

Motivations

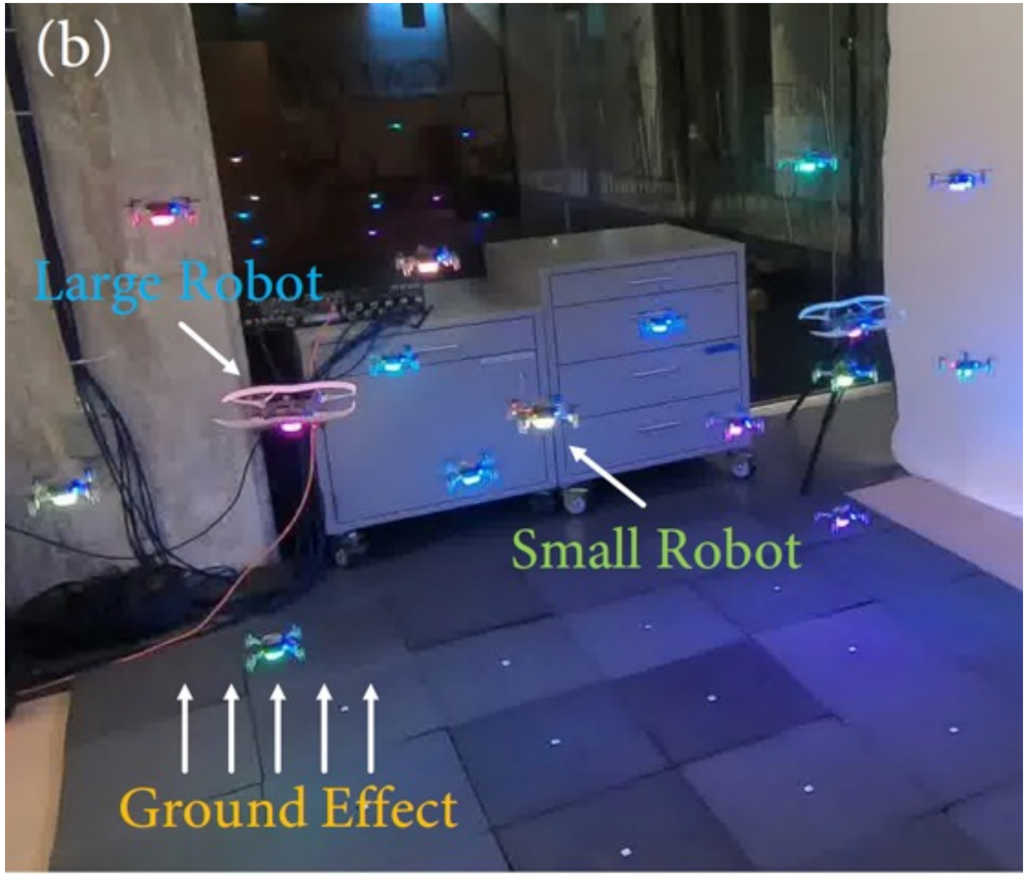




- Interaction matters
    - Prior works require **~60cm** safe vertical distance

- How can we model the interaction in:
    - two and more drones?
    - heterogeneous teams?

Models and Challenges



14 small robots and 2 large robots

nominal dynamics of type $I(i)$

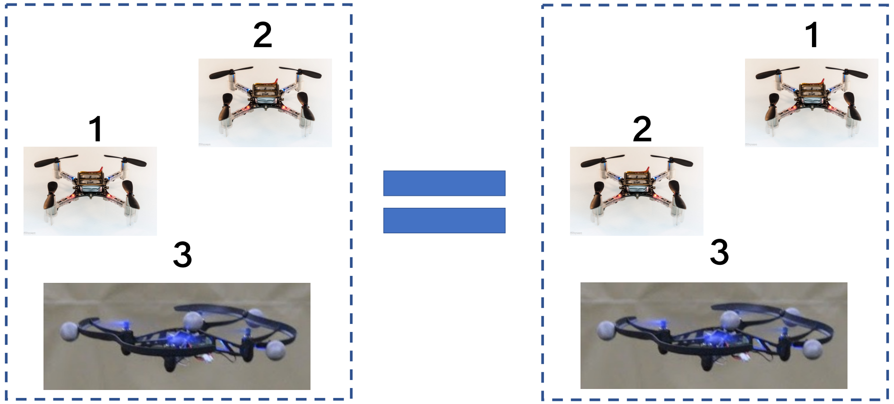
$$\dot{\mathbf{x}}^{(i)} = f_{\mathcal{I}(i)}(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}) + \begin{bmatrix} \mathbf{0} \\ \mathbf{f}_a^{(i)} \\ \mathbf{0} \\ \boldsymbol{\tau}_a^{(i)} \end{bmatrix}, \quad i = 1, \cdots, n$$

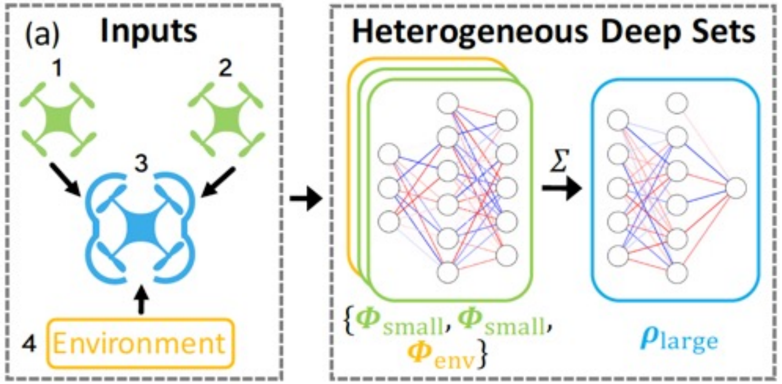
$$\mathbf{f}_a^{(i)} = \mathbf{f}_a^{\mathcal{I}(i)}(\mathcal{N}_{\text{type}-1}^{(i)}, \cdots, \mathcal{N}_{\text{type}-K}^{(i)})$$

$$\boldsymbol{\tau}_a^{(i)} = \boldsymbol{\tau}_a^{\mathcal{I}(i)}(\mathcal{N}_{\text{type}-1}^{(i)}, \cdots, \mathcal{N}_{\text{type}-K}^{(i)})$$

- Heterogeneity
- Generalization in the number of robots
  - What if we only have data from 1-3 robots?
- Limited computation power
  - Requires decentralization

Heterogeneous Permutation Invariance





- Heterogeneous permutation-invariant function: $h(x, y) = \max\{2x_1, 2x_2\} + \max\{y_1, y_2\}$
- We generalize the "Deep Sets" architecture:

Theorem (***Shi** et al., accepted by T-RO*; generalized *Zaheer et al., NeurIPS 2017*)
Any continuous, heterogeneous permutation-invariant function $[\hat{\mathbf{f}}_a; \hat{\boldsymbol{\tau}}_a]$ can be approximated by

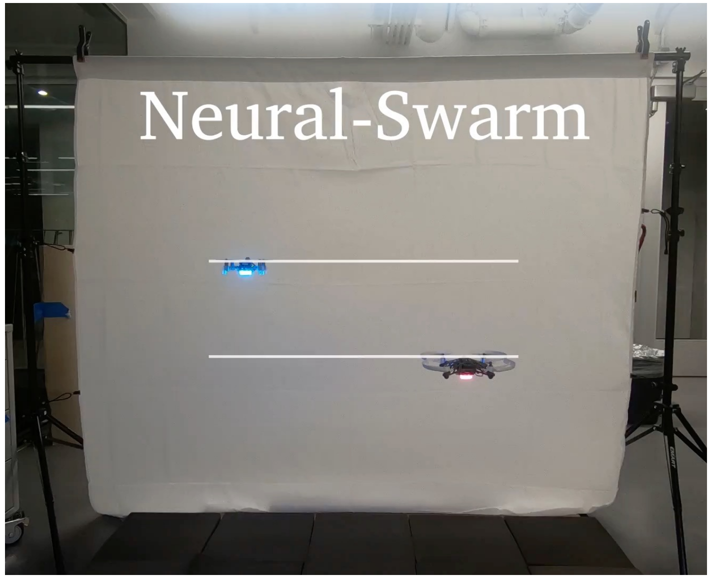
superposition in the latent space

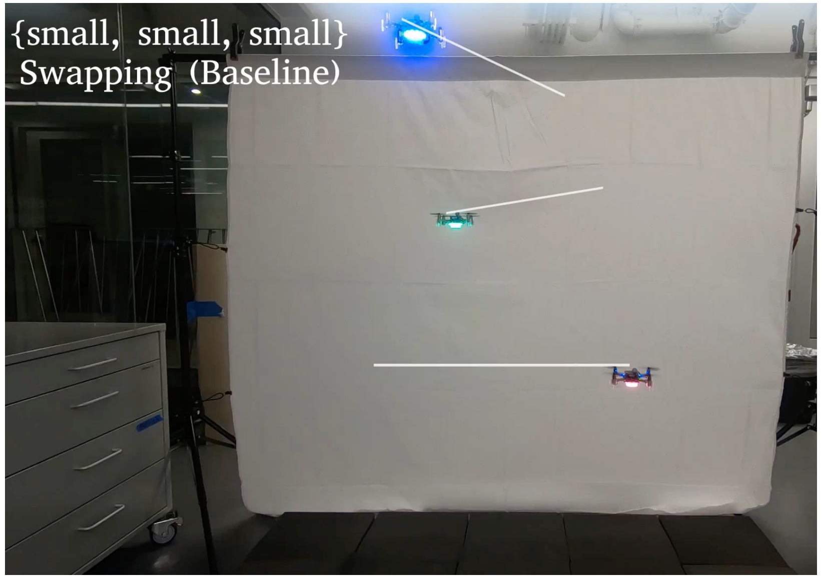
$$\begin{bmatrix} \mathbf{f}_a^{\mathcal{I}(i)}(\mathcal{N}^{(i)}_{\text{type}-1}, \cdots, \mathcal{N}^{(i)}_{\text{type}-K}) \\ \boldsymbol{\tau}_a^{\mathcal{I}(i)}(\mathcal{N}^{(i)}_{\text{type}-1}, \cdots, \mathcal{N}^{(i)}_{\text{type}-K}) \end{bmatrix} \approx \rho_{\mathcal{I}(i)}\left(\sum_k \sum_{\mathbf{x}^{(ij)} \in \mathcal{N}^{(i)}_{\text{type}-k}} \phi_{\mathcal{I}(j)}(\mathbf{x}^{(ij)})\right)$$

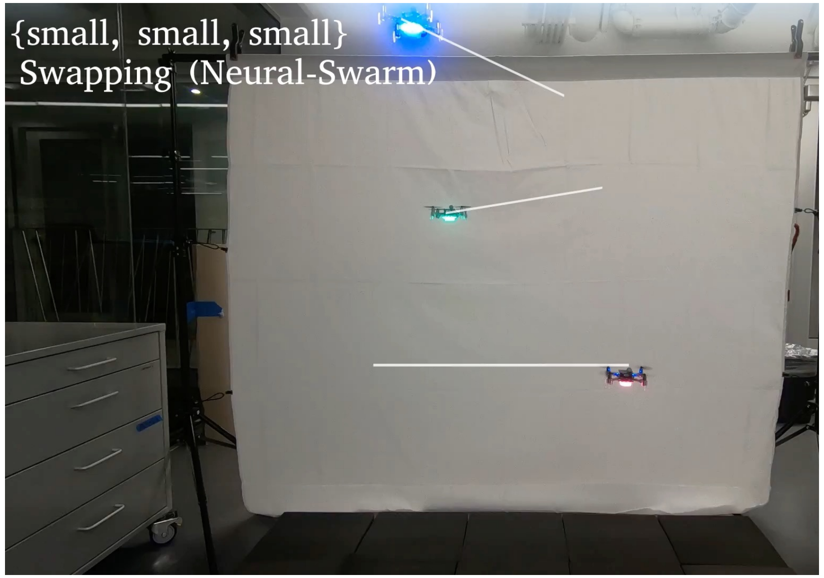
Interaction force on type $I(i)$ decode for type $I(i)$ encode type $I(j)$ neighbors

- We only need $2K$ neural networks for K types

Use-Case I: Control



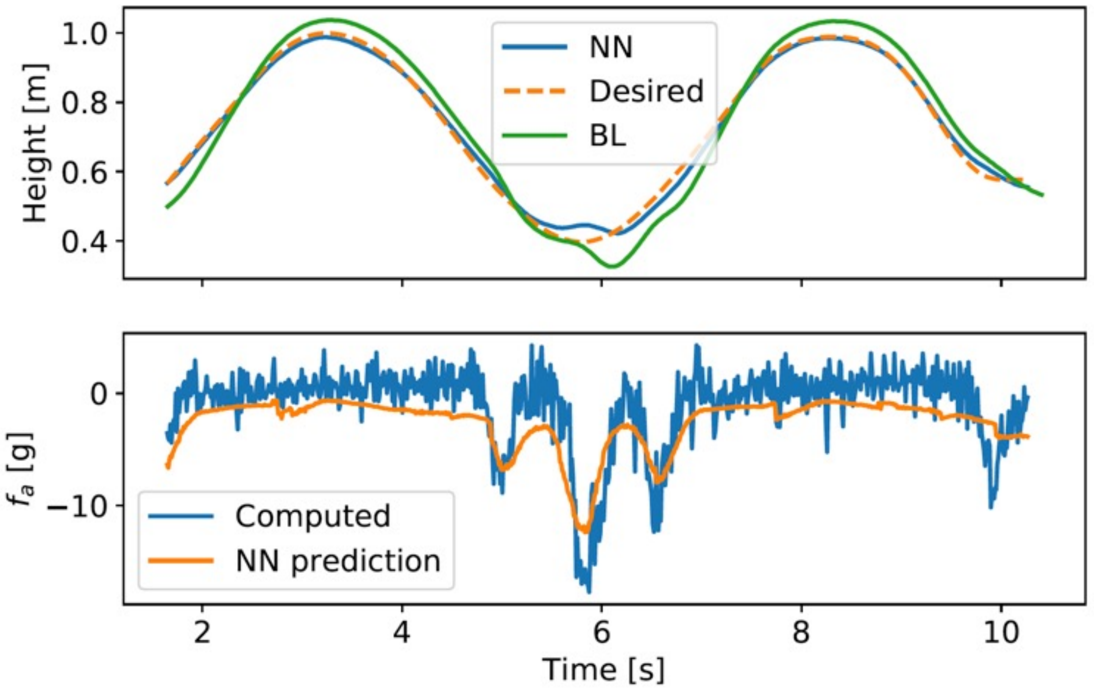




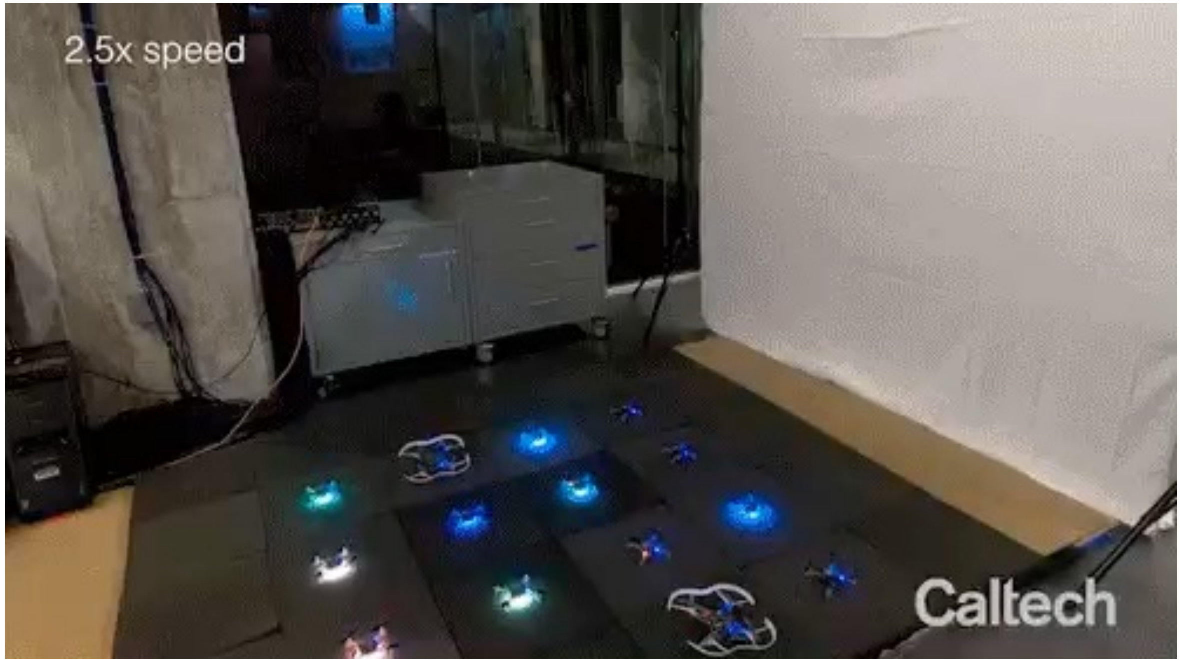
- Baseline (BL) is a SOTA nonlinear controller with delay compensation

Generalization

- We only collected 1-3 robots' data in training
- Can it generalize?



5-robot swapping task (2 larges, 3 smalls)



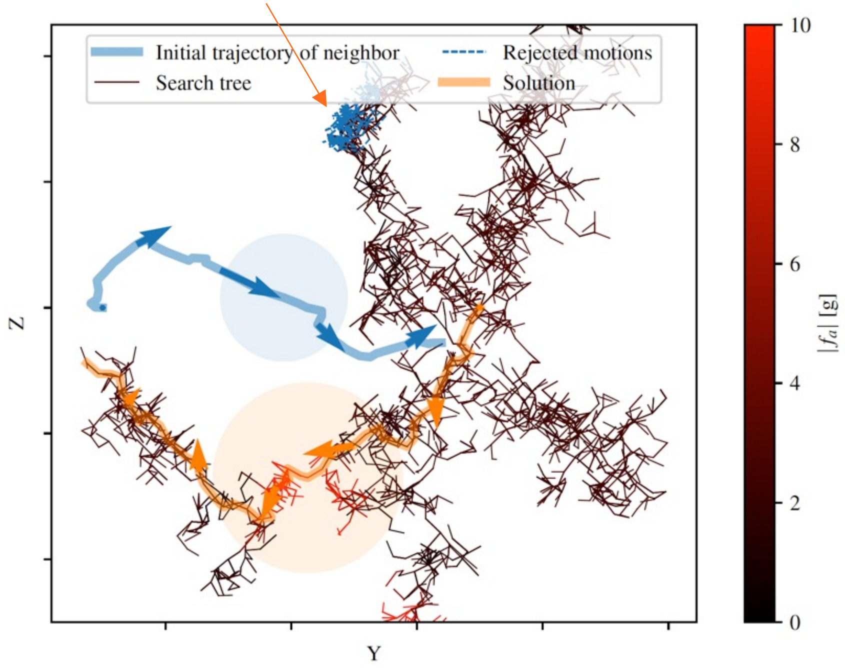
16-robot 3-ring tracking task (2 larges, 14 smalls)

- **24cm** minimum vertical distance
- Prior works: **~60cm** in a homogeneous team

Use-Case II: Interaction-Aware Motion Planning

- Integrate learned interactions with a **two-stage** planner
- Stage I: AO-RRT type sampling-based planning with learned interactions
- Stage II: Optimal-control-based planning using Sequential Convex Programming (SCP)

Rejected because introducing too large force



Stage I: AO-RRT type planning

- Fix the blue robot and plan for the orange

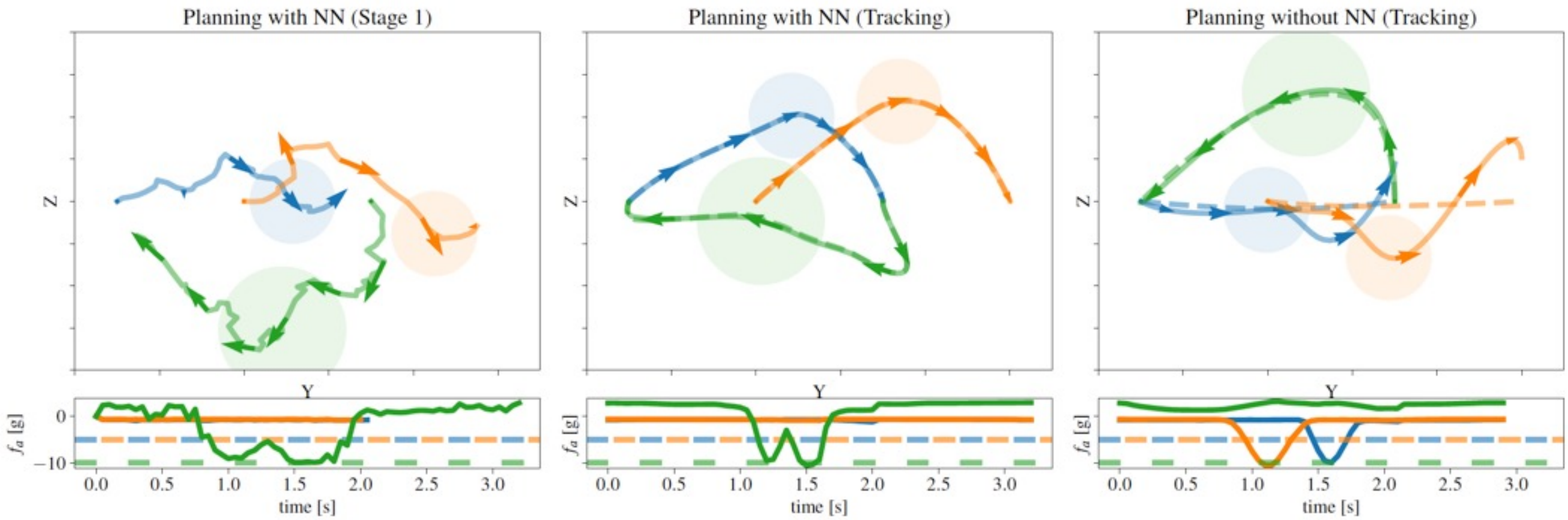
$$\min_{\mathbf{u}^{(i)},\mathbf{x}^{(i)},t_f} \sum_{i=1}^{N} \int_0^{t_f} \|\mathbf{u}^{(i)}(t)\| dt \tag{6}$$

$$\text{s.t.} \begin{cases} \text{robot dynamics (4)} & i \in [1,N] \\ \mathbf{u}^{(i)}(t) \in \mathcal{U}^{\mathcal{I}(i)};\ \mathbf{x}^{(i)}(t) \in \mathcal{X}^{\mathcal{I}(i)} & i \in [1,N] \\ \|\mathbf{p}^{(ij)}\| \geq r^{(\mathcal{I}(i)\mathcal{I}(j))} & i < j,\ j \in [2,N] \\ \boxed{\|\mathbf{f}_a^{(i)}\| \leq f_{a,\max}^{\mathcal{I}(i)};\ \|\boldsymbol{\tau}_a^{(i)}\| \leq \tau_{a,\max}^{\mathcal{I}(i)}} & i \in [1,N] \\ \mathbf{x}^{(i)}(0) = \mathbf{x}_s^{(i)};\ \mathbf{x}^{(i)}(t_f) = \mathbf{x}_f^{(i)} & i \in [1,N] \end{cases}$$

Stage II: optimal control

- Nonconvex so we use SCP
- We can explicitly control the interaction magnitude!

Use-Case II: Interaction-Aware Motion Planning

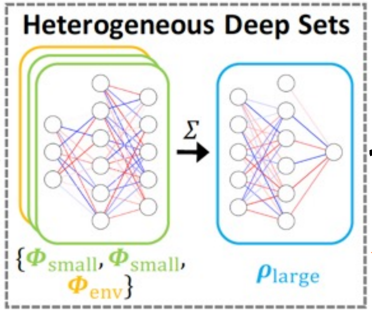


The large robot is at the bottom

The large robot is at the top; constraint violations

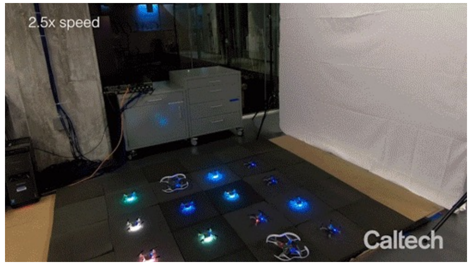
Neural-Swarm Takeaways

Physics



Mixed & Modularized Swarm

Decentralized Control
2-stage Planning



stability and safety guarantee

permutation invariance

Generalizability

1-3 robots → 5-16 robots

Compact & Efficient Representation

$$\rho_{\mathcal{I}(i)}\left(\sum_{k} \sum_{\mathbf{x}^{(ij)}\in\mathcal{N}^{(i)}_{\mathrm{type}-k}} \phi_{\mathcal{I}(j)}\left(\mathbf{x}^{(ij)}\right)\right)$$

⇨ Safe Close-Proximity Flight

***Shi** et al., ICRA 2020; **Shi** et al., accepted by IEEE T-RO*

Neural-Fly (Ongoing Work)

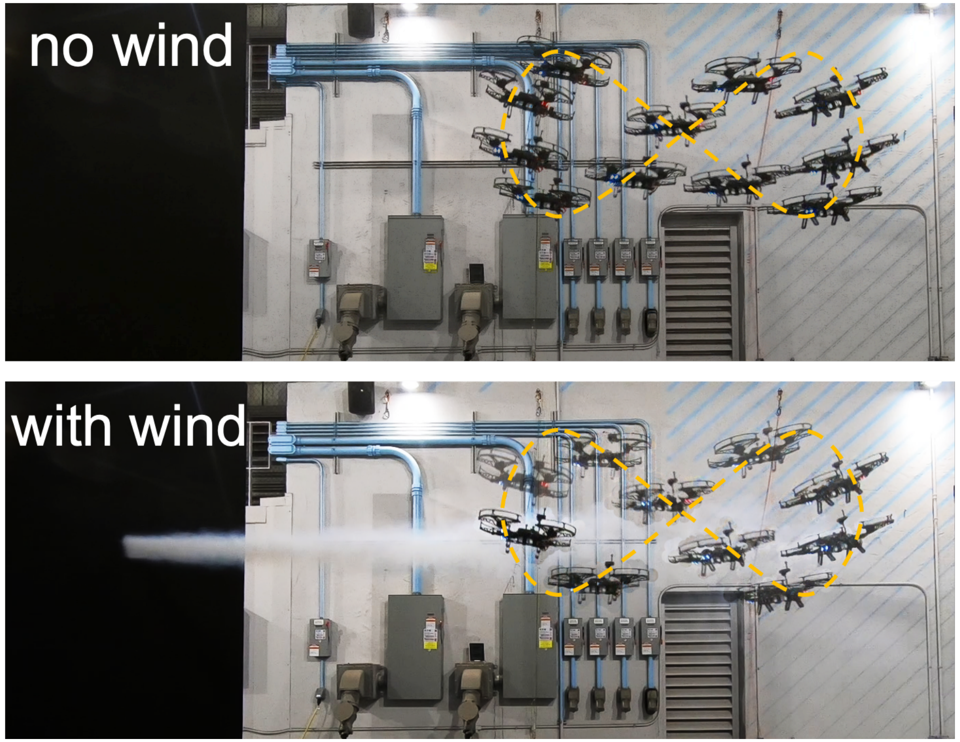
$$\dot{x} = f(x,u) + \underbrace{\boxed{g(x,u,t)}}_{\text{unknown}}$$

(Neural-Lander) learn g using normalized DNNs

(Neural-Swarm) learn g in heterogeneous swarms

(Neural-Fly) meta-learn g and online fast adapt

(Safe Exploration) safe learning g without experts



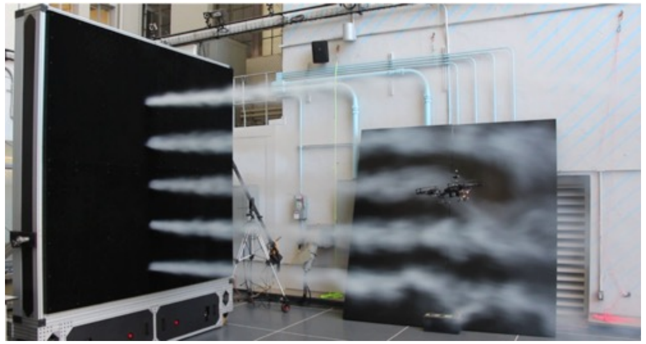
*O'Connell & **Shi** et al., ongoing work, preliminary version at arXiv*

Meta-Learning Meets Adaptive Control

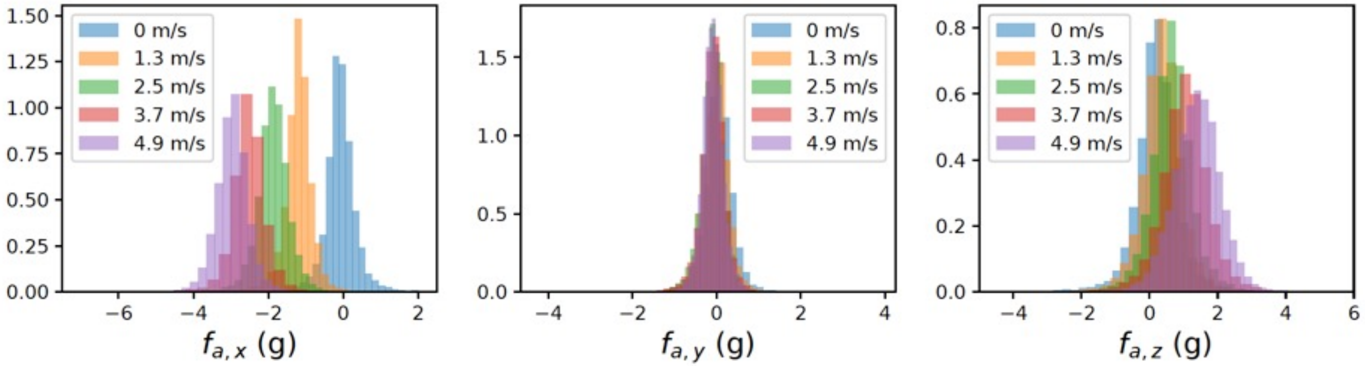
- The unknown residual term is governed by the environment $c(t)$:

$$\boxed{\dot{x} = f(x,u) + g(x, c(t))}$$

- Example: drone flying in different winds



Caltech CAST fan wall



histogram of g with different c

- Key idea:

$$\boxed{g(x,c) \approx \phi(x;\theta)a(c)}$$

a latent state to be online adapted (using adaptive control theory)

representations to be meta-learned

Meta-Learning Meets Adaptive Control

- With the meta-learned representation $\phi(x;\theta)$, adaptive control comes into play

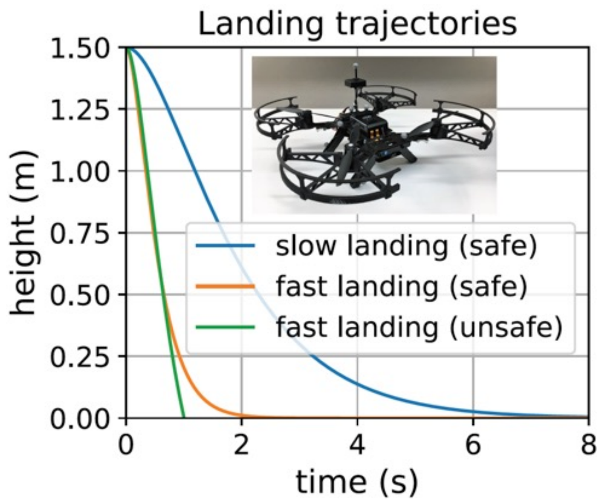
$$\boxed{a_{t+1} = \text{adapt}(a_t, \phi, x_t, \dots)}$$

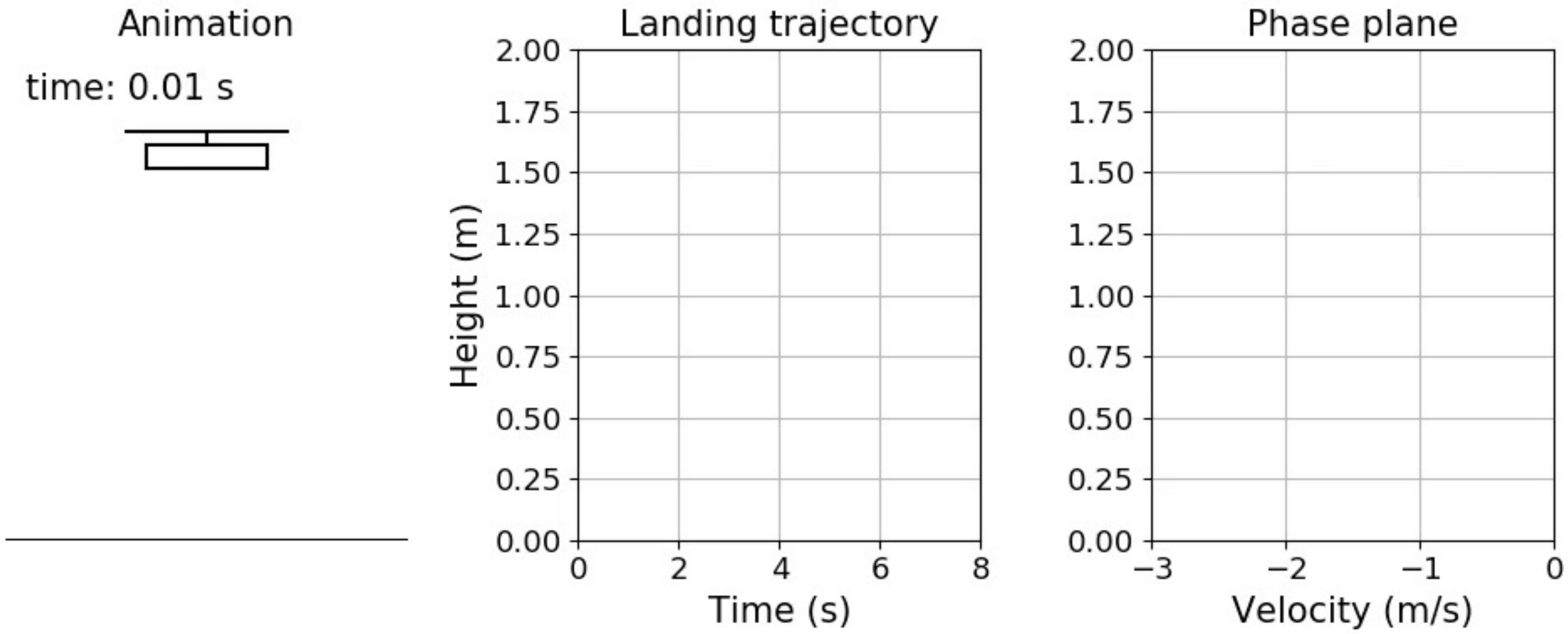
- Control stability and robustness can be guaranteed

Safe Exploration

$$\dot{x} = f(x,u) + \underbrace{\boxed{g(x,u,t)}}_{\text{unknown}}$$

- **(Neural-Lander)** learn g using normalized DNNs
- **(Neural-Swarm)** learn g in heterogeneous swarms
- **(Neural-Fly)** meta-learn g and online fast adapt
- **(Safe Exploration)** safe learning g without experts



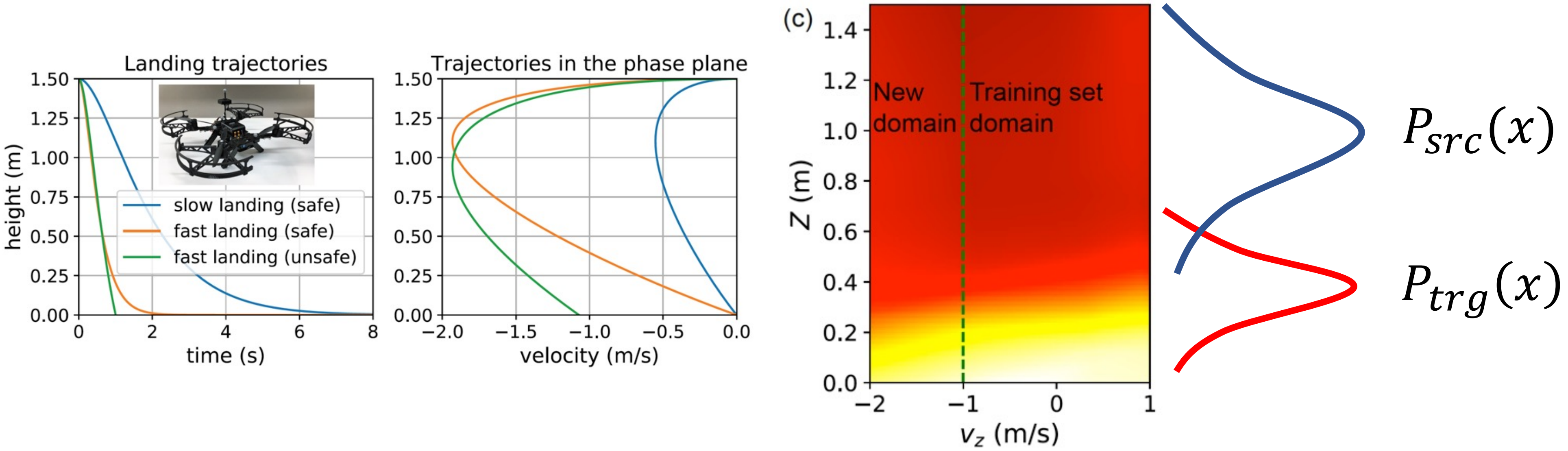


Liu et al., L4DC 2020, Nakka et al., RA-L 2020

Motivations and Challenges

- How to collect data without experts?
- Exploration v.s. exploitation: how to quantify uncertainty under domain shift?

covariate shift: $P(y|x)$ is fixed, but $P_{trg}(x) \neq P_{src}(x)$

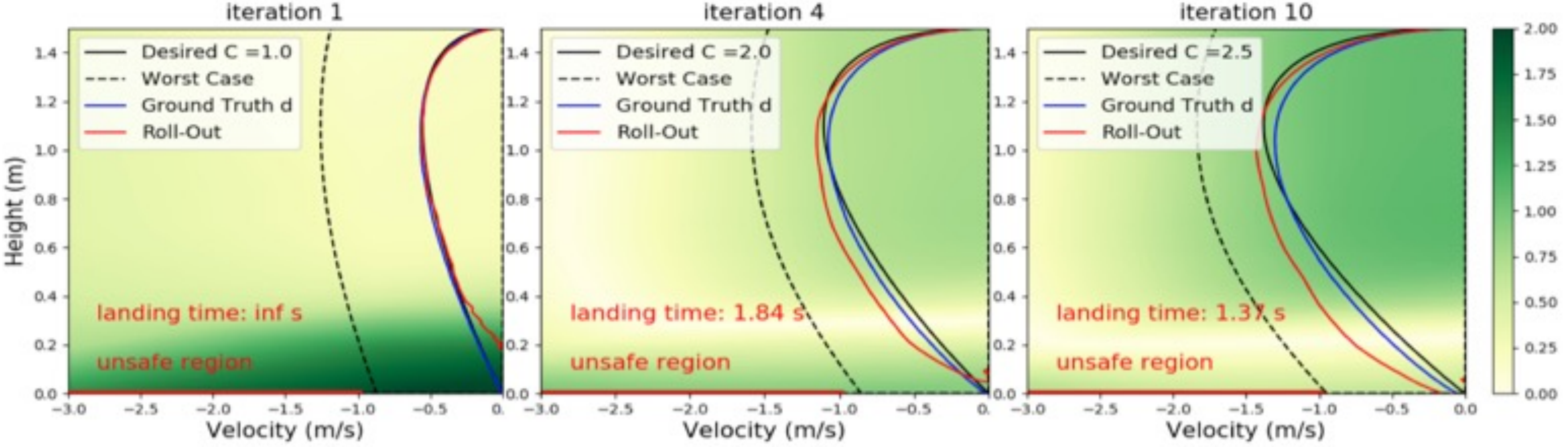


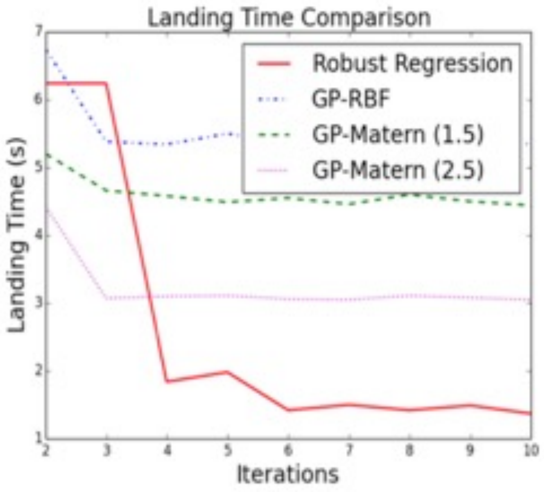
- Key idea: using **robust regression** to quantify uncertainty under covariate shift

Liu et al., L4DC 2020, Nakka et al., RA-L 2020

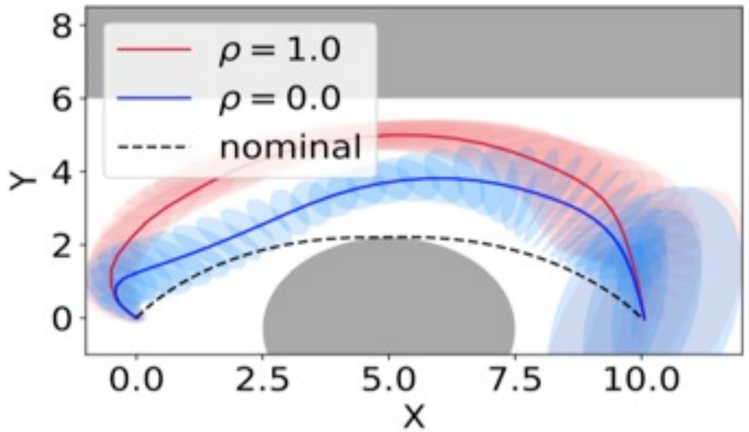
Results

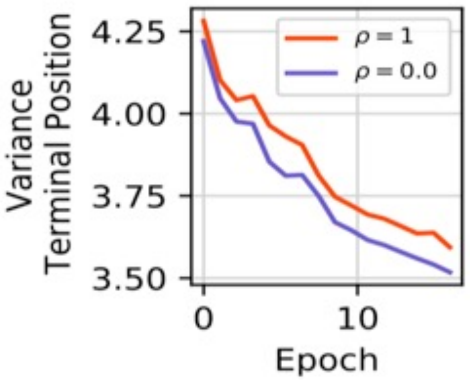
- Deterministic safety constraints and planning in a trajectory pool [*Liu et al., L4DC 2020*]

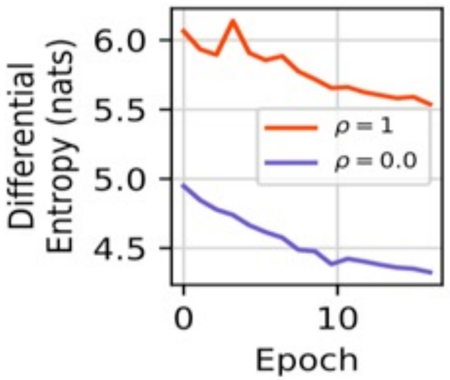


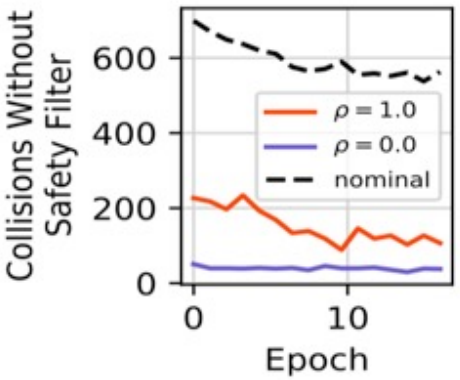


- Information-cost stochastic optimal control with chance-constraints [*Nakka et al., RA-L 2020*]









Recap for Topic I: Dynamics-Level Residual Learning

learning & control theoretical results integrate nicely

Physics

Data ↻ Learning

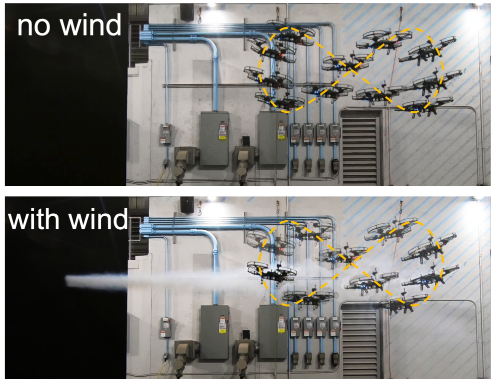
Mixed & Modularized Dynamical Systems

Control Synthesis Motion Planning

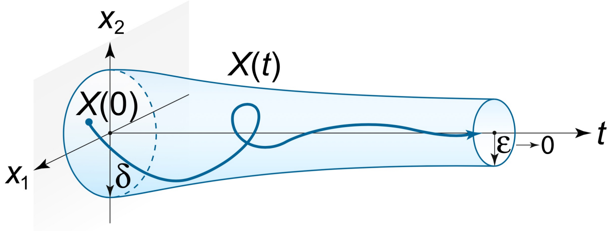
"control-semantic" regularization

- Key features we pursue:

Generalizability



Guarantees



Compact & Efficient Representation

New Capability in Robotics!

Some Other “Control-semantic” Regularization

- Learning stabilizable dynamics

Learning Stabilizable Nonlinear Dynamics
with Contraction-Based Regularization

Sumeet Singh[1], Spencer M. Richards[1], Vikas Sindhwani[2], Jean-Jacques E. Slotine[3], and Marco Pavone[1]

$$\min_{\hat{f}\in\mathcal{H}} \quad \sum_{i=1}^{N} \left\| \hat{f}(x_i, u_i) - \dot{x}_i \right\|_2^2 + \mu \|\hat{f}\|_{\mathcal{H}}^2$$

$$\text{s.t.} \quad \hat{f} \text{ is stabilizable,}$$

- Learning Lagrangian systems

Deep Lagrangian Networks:
Using Physics as Model Prior for Deep Learning

Michael Lutter, Christian Ritter & Jan Peters *

covering all (rigid) robotic systems

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \underbrace{\dot{\mathbf{H}}(\mathbf{q})\dot{\mathbf{q}} - \frac{1}{2}\left(\frac{\partial}{\partial \mathbf{q}}\left(\dot{\mathbf{q}}^T \mathbf{H}(\mathbf{q})\dot{\mathbf{q}}\right)\right)^T}_{:=\mathbf{c}(\mathbf{q},\dot{\mathbf{q}})} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}$$

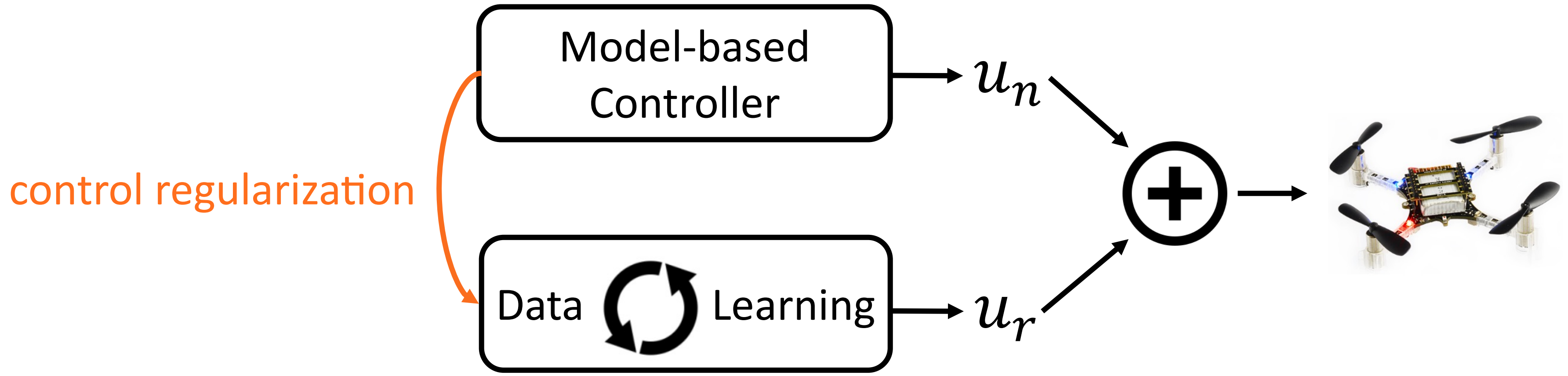
- Could be either hard constraints or regularizations

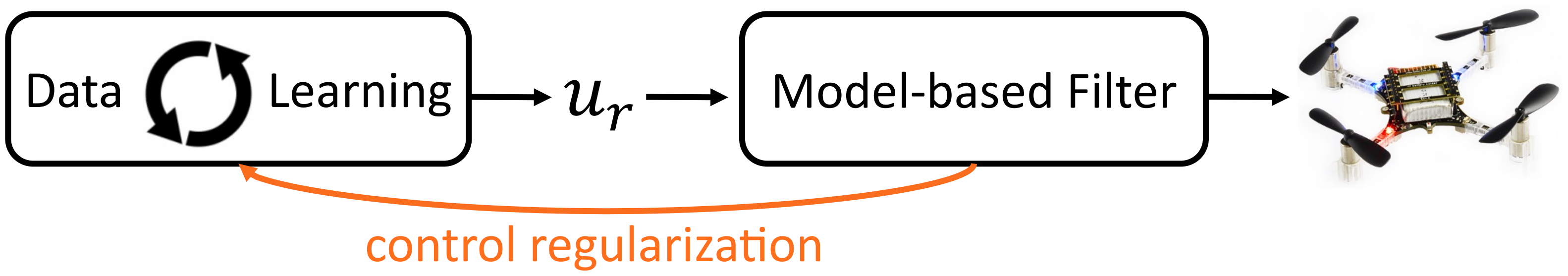
References in Topic I: Dynamics-Level Residual Learning

- Zeng, A., Song, S., Lee, J., Rodriguez, A., & Funkhouser, T. (2020). Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*
- G. Shi*, X. Shi*, M. O'Connell*, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, S.-J. Chung, "Neural Lander: Stable Drone Landing Control Using Learned Dynamics", *ICRA 2019*
- G. Shi, W. Hoenig, Y. Yue, S.-J. Chung, "Neural-Swarm: Decentralized Close-Proximity Multirotor Control Using Learned Interactions", *ICRA 2020*
- G. Shi, W. Hoenig, X. Shi, Y. Yue, S.-J. Chung, "Neural-Swarm2: Planning and Control of Heterogeneous Multirotor Swarms Using Learned Interactions", accepted by *T-RO, 2021*
- M. O'Connell*, G. Shi, X. Shi, S.-J. Chung, "Meta-Learning-Based Robust Adaptive Flight Control Under Uncertain Wind Conditions", *arXiv preprint*
- A. Liu, G. Shi, S.-J. Chung, A. Anandkumar, Y. Yue, "Robust Regression for Safe Exploration in Control", *L4DC 2020*
- Y. K. Nakka, A. Liu, G. Shi, A. Anandkumar, Y. Yue, S.-J. Chung, "Chance-Constrained Trajectory Optimization for Safe Exploration and Learning of Nonlinear Systems", *RA-L 2020*
- P. Bartlett, D. Foster, D. Telgarsky, "Spectrally-normalized Margin Bounds for Neural Networks", NeurIPS 2017
- M. Zaheer, S. Kottur, S. Ravanbhakhsh, B. Póczos, R. Salakhutdinov, A. J. Smola, "Deep Sets", *NeurIPS 2017*
- C. Finn, P. Abbeel, S. Levine, "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks", *ICML 2017*
- H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein, "Visualizing the Loss Landscape of Neural Nets", *NeurIPS 2018*

Topic II: Action-level Residual Learning

- Two popular pipelines: 1) superposition and 2) filtering
- Some materials from http://www.yisongyue.com/talks/safety_critical_learning.pdf





Example 1: Deep RL + Hand-engineered Controller

- In the training process, π_H is "encoded" in the dynamics

Residual Reinforcement Learning for Robot Control

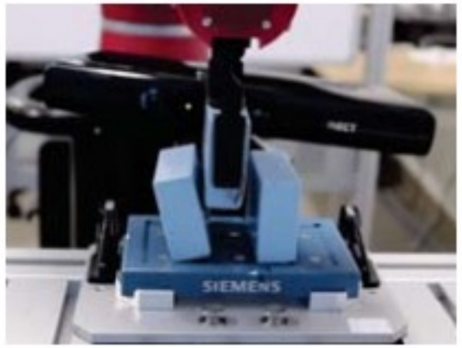
Tobias Johannink*[1,3], Shikhar Bahl*[2], Ashvin Nair*[2], Jianlan Luo[1,2], Avinash Kumar[1], Matthias Loskyll[1], Juan Aparicio Ojea[1], Eugen Solowjow[1], Sergey Levine[2]

Algorithm 1 Residual reinforcement learning

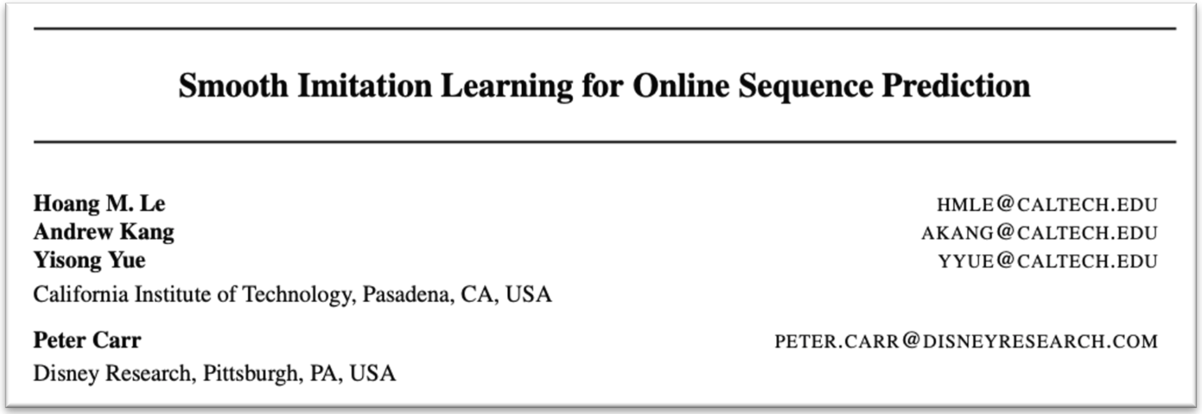
Require: policy π_θ, hand-engineered controller π_{H}.

for $n = 0, \ldots, N-1$ episodes **do**
  Initialize random process $\mathcal{N}$ for exploration
  Sample initial state $s_0 \sim E$.
  for $t = 0, \ldots, H-1$ steps **do**
    Get policy action $u_t = \pi_\theta(s_t) + \mathcal{N}_t$.
    Get action to execute $u'_t = u_t + \pi_{\mathrm{H}}(s_t)$.
    Get next state $s_{t+1} \sim p(\cdot \mid s_t, u'_t)$.
    Store (s_t, u_t, s_{t+1}) into replay buffer $\mathcal{R}$.
    Sample set of transitions $(s, u, s') \sim \mathcal{R}$.
    Optimize θ using RL with sampled transitions.
  end for
end for

hand-engineered controller



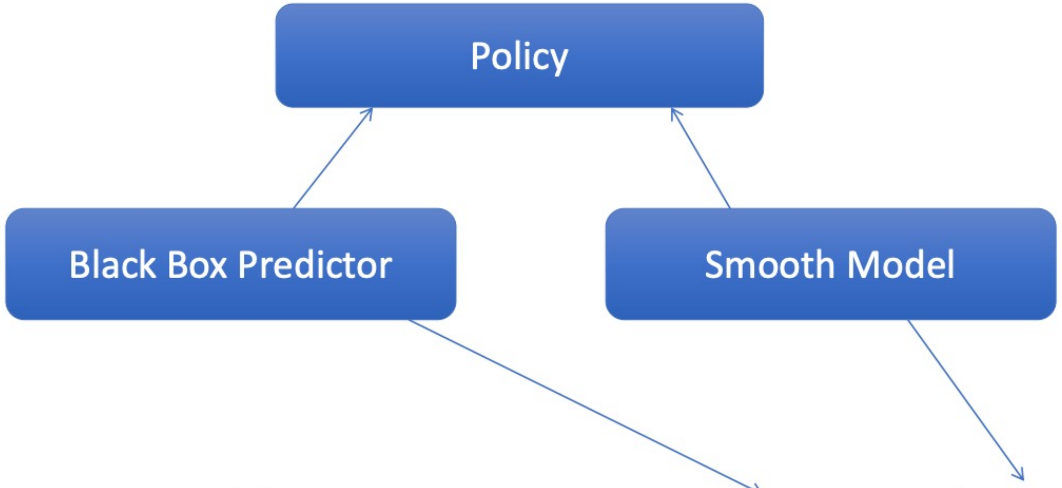
Example 2: Smooth Imitation Learning



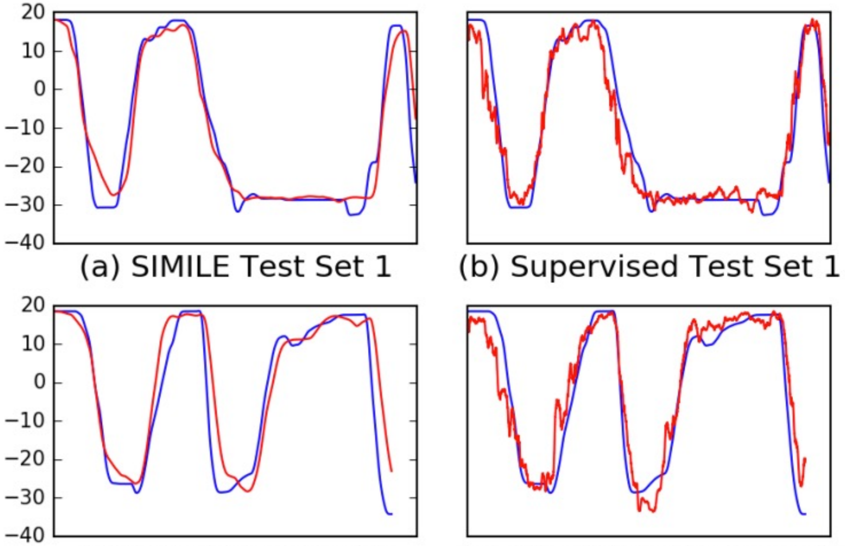
Smooth Imitation Learning for Online Sequence Prediction

Hoang M. Le HMLE@CALTECH.EDU
Andrew Kang AKANG@CALTECH.EDU
Yisong Yue YYUE@CALTECH.EDU
California Institute of Technology, Pasadena, CA, USA

Peter Carr PETER.CARR@DISNEYRESEARCH.COM
Disney Research, Pittsburgh, PA, USA



$$argmin_{h=(f,g)} L(h) \quad \text{s.t.} \quad h(s) = argmin_{a'}(f(s) - a')^2 + \lambda(g(s) - a')^2$$
$$= \frac{f(s)+\lambda g(s)}{1+\lambda}$$



(a) SIMILE Test Set 1 (b) Supervised Test Set 1

(c) SIMILE Test Set 2 (d) Supervised Test Set 2

Example 3: Control Regularization Reduces Variance in RL

Control Regularization for Reduced Variance Reinforcement Learning

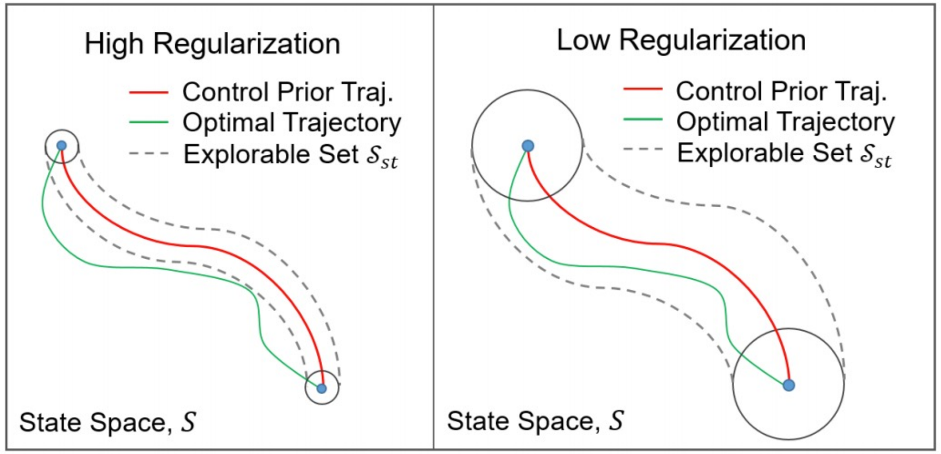
Richard Cheng[1] Abhinav Verma[2] Gábor Orosz[3] Swarat Chaudhuri[2] Yisong Yue[1] Joel W. Burdick[1]

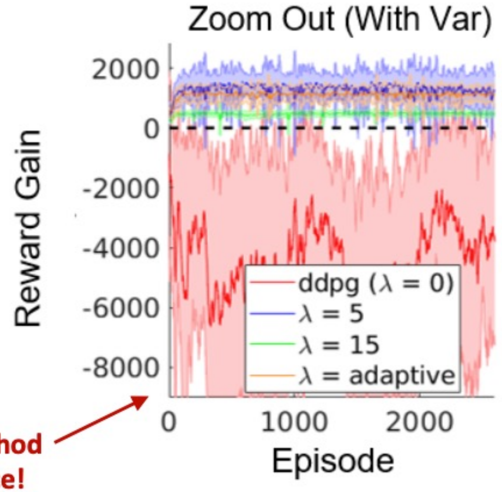
$$u_k(s) = \frac{1}{1+\lambda} u_{\theta_k}(s) + \frac{\lambda}{1+\lambda} u_{prior}(s)$$

- Theorem (informal):
  - Variance of policy gradient decreases by factor of: $\left(\frac{1}{1+\lambda}\right)^2$

Implies much faster learning!

  - Bias converges to: $\left(\frac{\lambda}{1+\lambda}\right) D_{TV}(h^*, g)$





Baseline RL Method has High Variance!

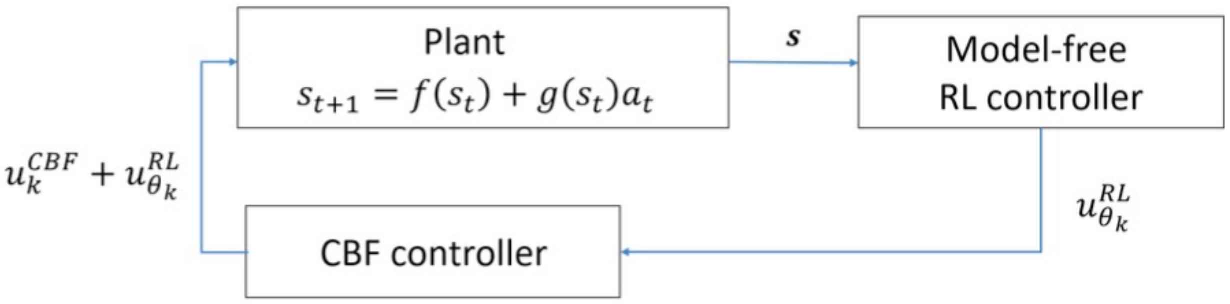
Example 4: Model-based Controller as a "Filter"

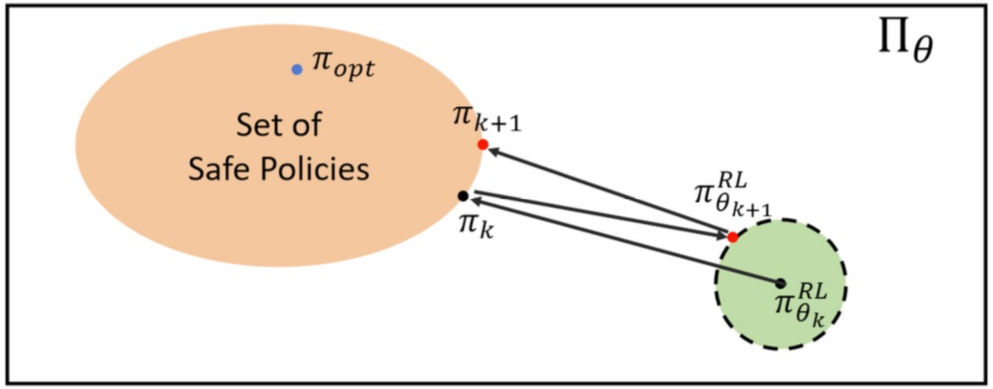
End-to-End Safe Reinforcement Learning through Barrier Functions for Safety-Critical Continuous Control Tasks

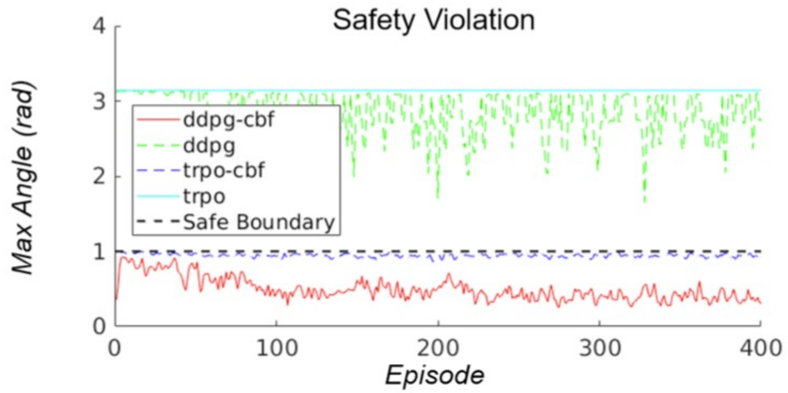
Richard Cheng,[1] Gábor Orosz,[2] Richard M. Murray,[1] Joel W. Burdick[1]
[1]California Institute of Technology, [2]University of Michigan, Ann Arbor

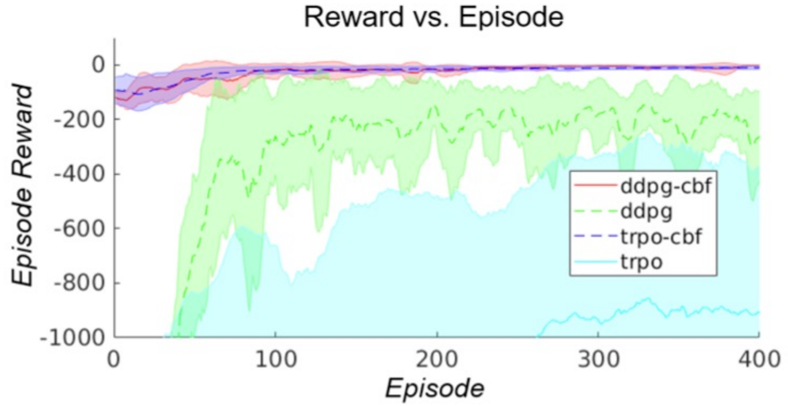
$$(a_t, \epsilon) = \underset{a_t, \epsilon}{\operatorname{argmin}} \|a_t\|_2 + K_\epsilon \epsilon$$
$$\text{s.t. } p^T f(s_t) + p^T g(s_t)\left(u^{RL}_{\theta_k}(s_t) + a_t\right) + p^T \mu_d(s_t)$$
$$- k_\delta |p|^T \sigma_d(s_t) + q \geq (1-\eta) h(s_t) - \epsilon$$
$$a^i_{low} \leq a^i_t + u^{RL(i)}_{\theta_k}(s_t) \leq a^i_{high} \text{ for } i = 1, \ldots, M$$

CBF safety constraints









Topic III: Program-level Residual Learning

- The “program” design is on a case-by-case basis
- Algorithm design and analysis are not as clear as the dynamics-level and action-level residual learning
- A (very) general framework:

(Partially) Unknown Systems
$$\dot{x} = f(x,u) + g(x,u,t)$$

→

Some Program for Control

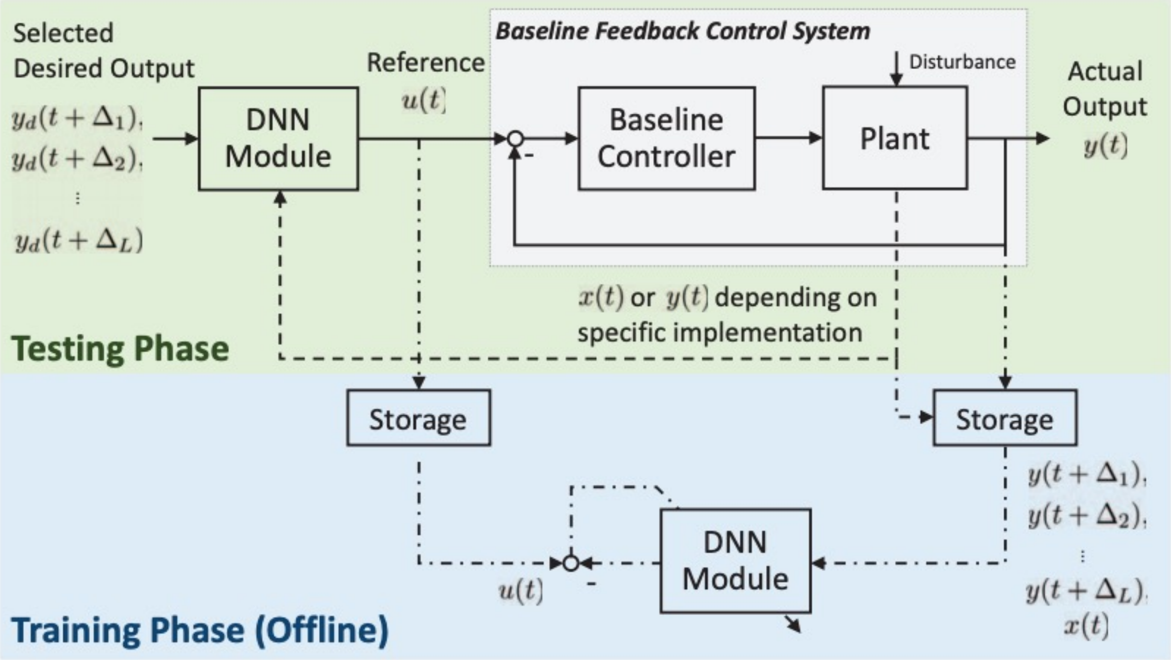
Data ⟳ Learning

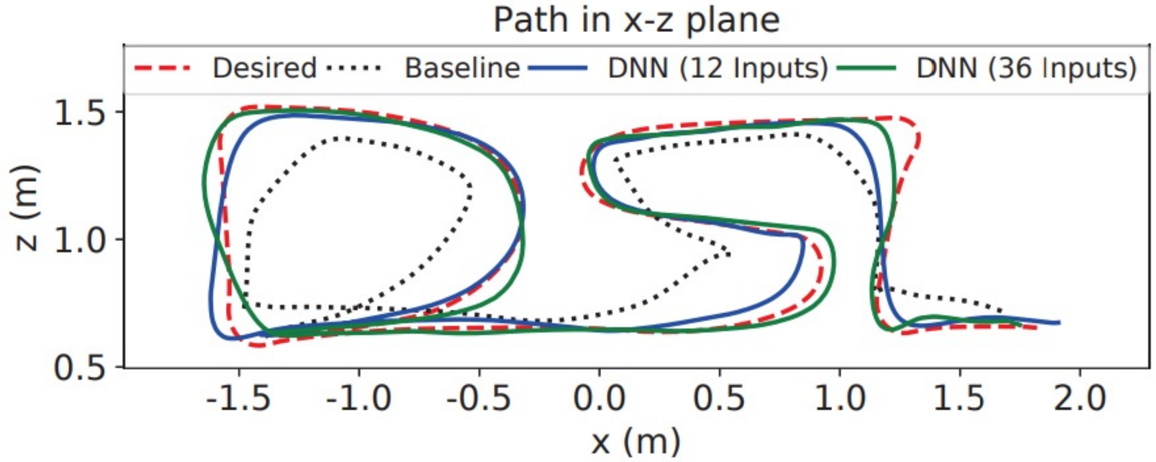
→

Example 1: DNN to “Adapt” the Reference Signal

Design of Deep Neural Networks as Add-on Blocks for Improving Impromptu Trajectory Tracking

Siqi Zhou, Mohamed K. Helwa, and Angela P. Schoellig





Example 2: Learning Control Lyapunov Function Residual

A Control Lyapunov Perspective on Episodic Learning via Projection to State Stability

Andrew J. Taylor[1], Victor D. Dorobantu[1], Meera Krishnamoorthy,
Hoang M. Le, Yisong Yue, and Aaron D. Ames

- In control we only need to make sure $\mathcal{U}(\mathbf{x}) = \{\mathbf{u} \in \mathcal{U} : \hat{\dot{V}}(\mathbf{x}, \mathbf{u}) \leq -\alpha(\|\mathbf{x}\|)\},$

$$\dot{\mathbf{x}} = \hat{\mathbf{f}}(\mathbf{x}) + \hat{\mathbf{g}}(\mathbf{x})\mathbf{u} + \overbrace{(\underbrace{\mathbf{g}(\mathbf{x}) - \hat{\mathbf{g}}(\mathbf{x})}_{\boxed{\mathbf{A}(\mathbf{x})}})\mathbf{u} + \underbrace{\mathbf{f}(\mathbf{x}) - \hat{\mathbf{f}}(\mathbf{x})}_{\boxed{\mathbf{b}(\mathbf{x})}}}^{\mathbf{d}}, \quad (23)$$

Residual dynamics

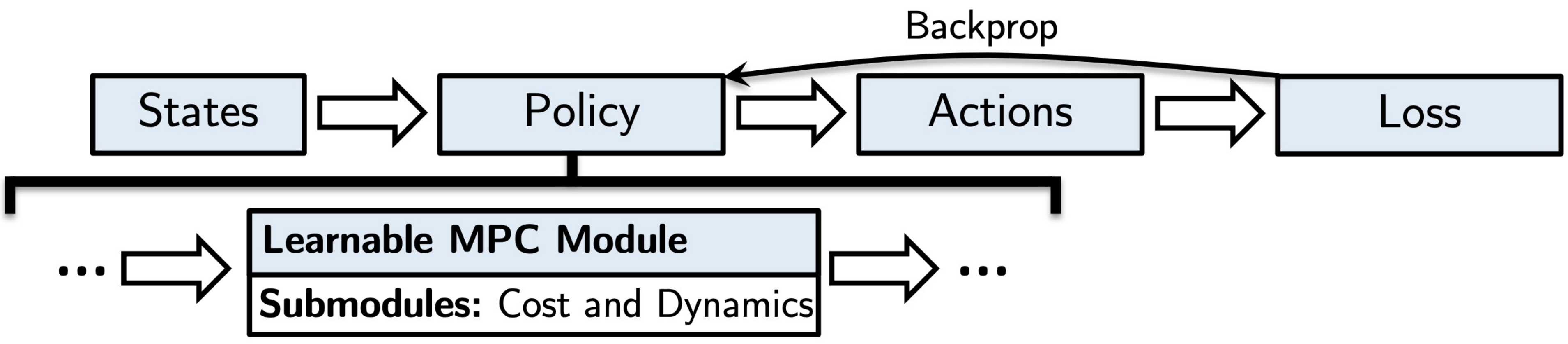
$$\begin{aligned}\dot{V}(\mathbf{x}, \mathbf{u}, \mathbf{d}) = & \overbrace{(\hat{\mathbf{f}}(\mathbf{x}) + \hat{\mathbf{g}}(\mathbf{x})\mathbf{u})^\top \nabla V(\mathbf{x})}^{\hat{\dot{V}}(\mathbf{x},\mathbf{u})} \\ & + (\underbrace{\mathbf{A}(\mathbf{x})^\top \nabla V(\mathbf{x})}_{\boxed{\mathbf{a}(\mathbf{x})}})^\top \mathbf{u} + \underbrace{\mathbf{b}(\mathbf{x})^\top \nabla V(\mathbf{x})}_{\boxed{b(\mathbf{x})}}, \quad (24)\end{aligned}$$

If we have a control Lyapunov function V, we can just learn the projection

Example 3: Differentiable MPC

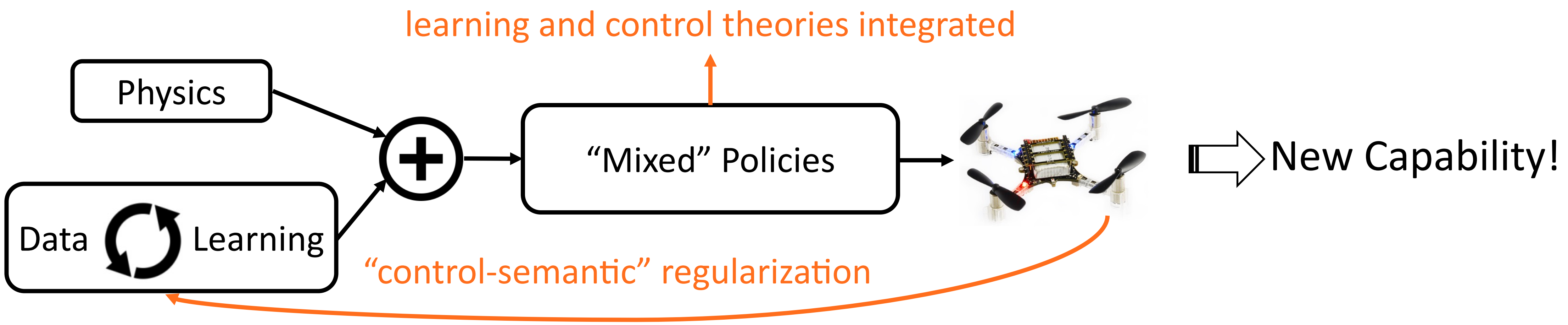
Differentiable MPC for End-to-end Planning and Control

Brandon Amos[1] **Ivan Dario Jimenez Rodriguez**[2] **Jacob Sacks**[2]
Byron Boots[2] **J. Zico Kolter**[13]
[1]Carnegie Mellon University [2]Georgia Tech [3]Bosch Center for AI



$$\tau^\star_{1:T} = \underset{\tau_{1:T}}{\operatorname{argmin}} \sum_t C_{\theta,t}(\tau_t) \quad \text{subject to} \quad x_1 = x_{\text{init}},\ x_{t+1} = f_\theta(\tau_t),\ \underline{u} \le u \le \overline{u},$$

Summary



- Topic I: dynamics-level residual learning
- Topic II: action-level residual learning
- Topic III: program-level residual learning

Some directions:

- Trade-offs (e.g., sampling complexity)?
- Combine control and learning theory (e.g., generalization)

“1+1>2”

approximation error of DNN ($\|\hat{\mathbf{f}}_a - \mathbf{f}_a\|$)

$$\frac{\epsilon_m}{\lambda - L_a \rho}$$