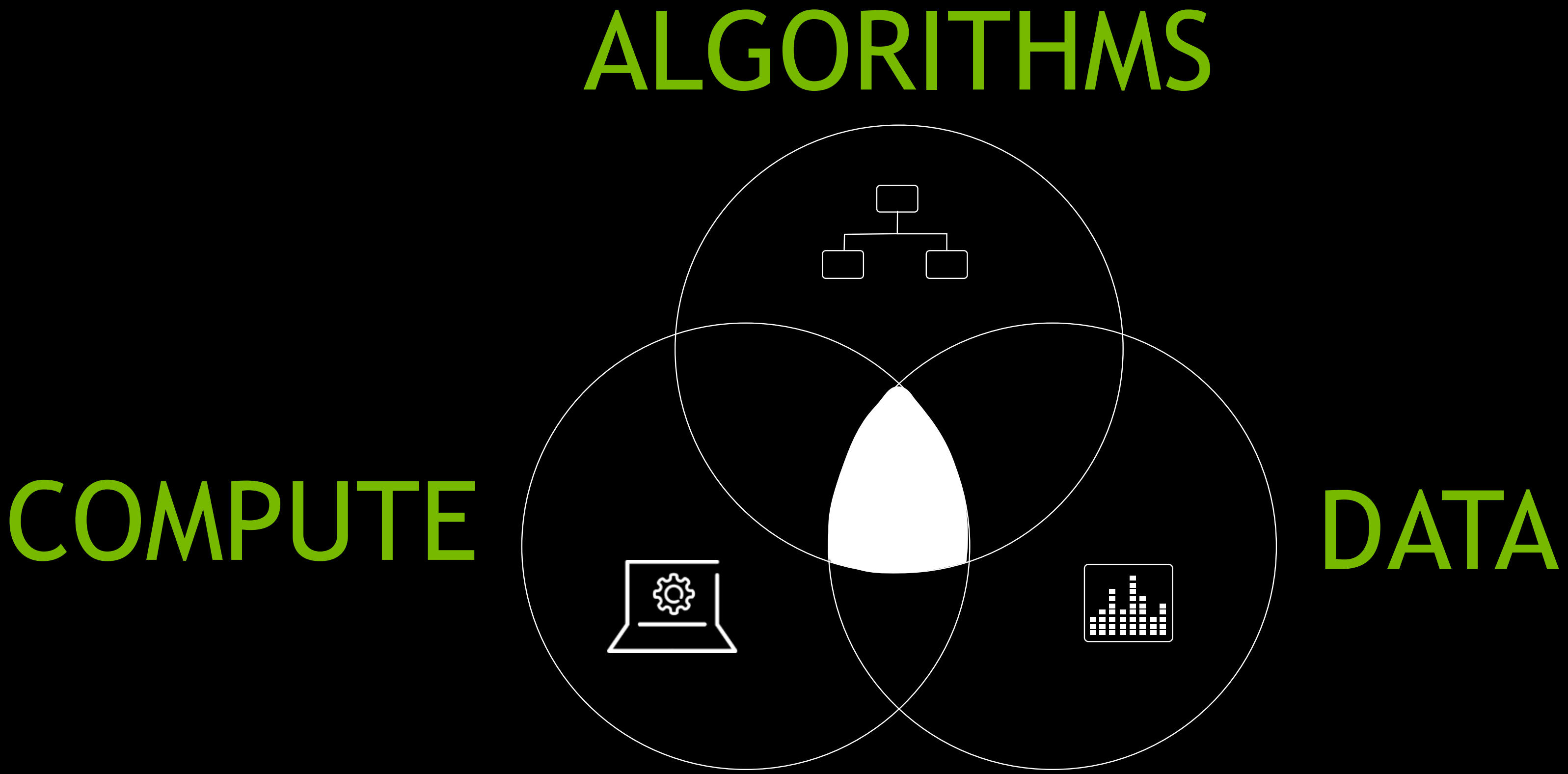
Anima Anandkumar

Caltech

NVIDIA

INFUSING PHYSICS + STRUCTURE INTO MACHINE LEARNING

TRINITY OF AI/ML



PHYSICS-INFUSED LEARNING

Learning = Data + Priors

Learning = Computational Reasoning over Data & Priors

How to use Physics as a (new) form of Prior?

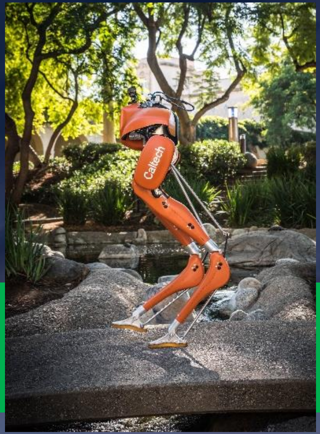
- Learnability
- Generalization

What are some of the hardest challenges for AI?

“ **Mens sana in corpore sano.**”

Juvenal in Satire X.

ROBOTICS MOONSHOTS

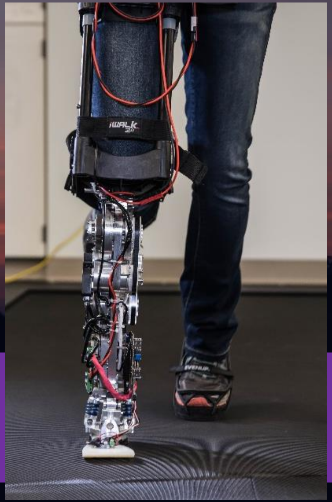


Explorers
Planetary, Underwater, and Space Explorers

Guardians
Dynamic Event Monitors and First Responders

Transformers
Swarms of Robots Transforming Shapes and Functions

Transporters
Robotic Flying Ambulances and Delivery Drones

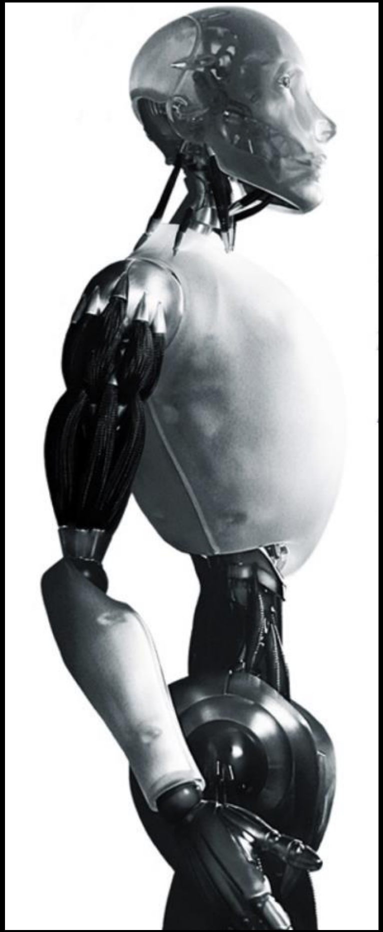


Partners
Robots Helping and Entertaining People

MIND & BODY
NEXT-GENERATION AI

Instinctive:
Fine-grained reactive Control

Behavioral:
Sense and react to human



Deliberative:
Making and adapting plans

Multi-Agent:
Acting for the greater good

PHYSICS-INFUSED LEARNING FOR ROBOTICS AND CONTROL

Learning = Data + Priors

Learning = Computational Reasoning over Data & Priors

How to use Physics as a (new) form of Prior?

- Learnability
- Generalization

BASELINE: MODEL-BASED CONTROL (NO LEARNING)

New State

Current Action (aka control input)

$$s_{t+1} = F(s_t, u_t) + \epsilon$$

Current State

Unmodeled Disturbance / Error

(Value Iteration is also contraction mapping)

Robust Control (fancy contraction mappings)
- Stability guarantees (e.g., Lyapunov)
- Precision/optimality depends on error

LEARNING RESIDUAL DYNAMICS FOR DRONE LANDING

f = nominal dynamics
$\tilde{f}$ = learned dynamics

New State

Current Action (aka control input)

$$s_{t+1} = f(s_t, a_t) + \tilde{f}(s_t, a_t) + \epsilon$$

Current State

Unmodeled Disturbance

Use existing control methods to generate actions

- Provably robust (even using deep learning)
- Requires $\tilde{f}$ Lipschitz & bounded error

CONTROL SYSTEM FORMULATION

Learn the Residual
(function of state and control input)

- Dynamics:

$$\begin{cases} \dot{\mathbf{p}} = \mathbf{v}, & m\dot{\mathbf{v}} = m\mathbf{g} + R\mathbf{f}_u + \mathbf{f}_a \\ \dot{R} = RS(\boldsymbol{\omega}), & J\dot{\boldsymbol{\omega}} = J\boldsymbol{\omega} \times \boldsymbol{\omega} + \boldsymbol{\tau}_u + \boldsymbol{\tau}_a \end{cases}$$

- Control:

$$\mathbf{f}_u = [0, 0, T]^\top$$

$$\boldsymbol{\tau}_u = [\tau_x, \tau_y, \tau_z]^\top$$

$$\begin{bmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} c_T & c_T & c_T & c_T \\ 0 & c_T l_{\text{arm}} & 0 & -c_T l_{\text{arm}} \\ -c_T l_{\text{arm}} & 0 & c_T l_{\text{arm}} & 0 \\ -c_Q & c_Q & -c_Q & c_Q \end{bmatrix} \begin{bmatrix} n_1^2 \\ n_2^2 \\ n_3^2 \\ n_4^2 \end{bmatrix}$$

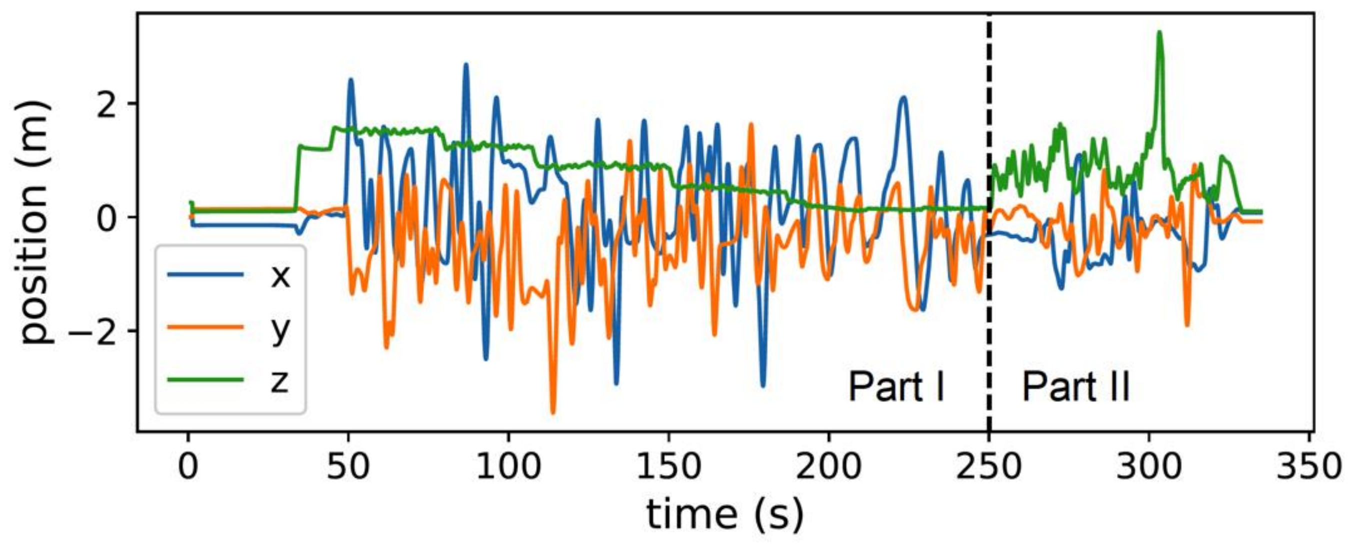
- Unknown forces & moments:

$$\mathbf{f}_a = [f_{a,x}, f_{a,y}, f_{a,z}]^\top$$

$$\boldsymbol{\tau}_a = [\tau_{a,x}, \tau_{a,y}, \tau_{a,z}]^\top$$

Learn the Residual

DATA COLLECTION (MANUAL EXPLORATION)



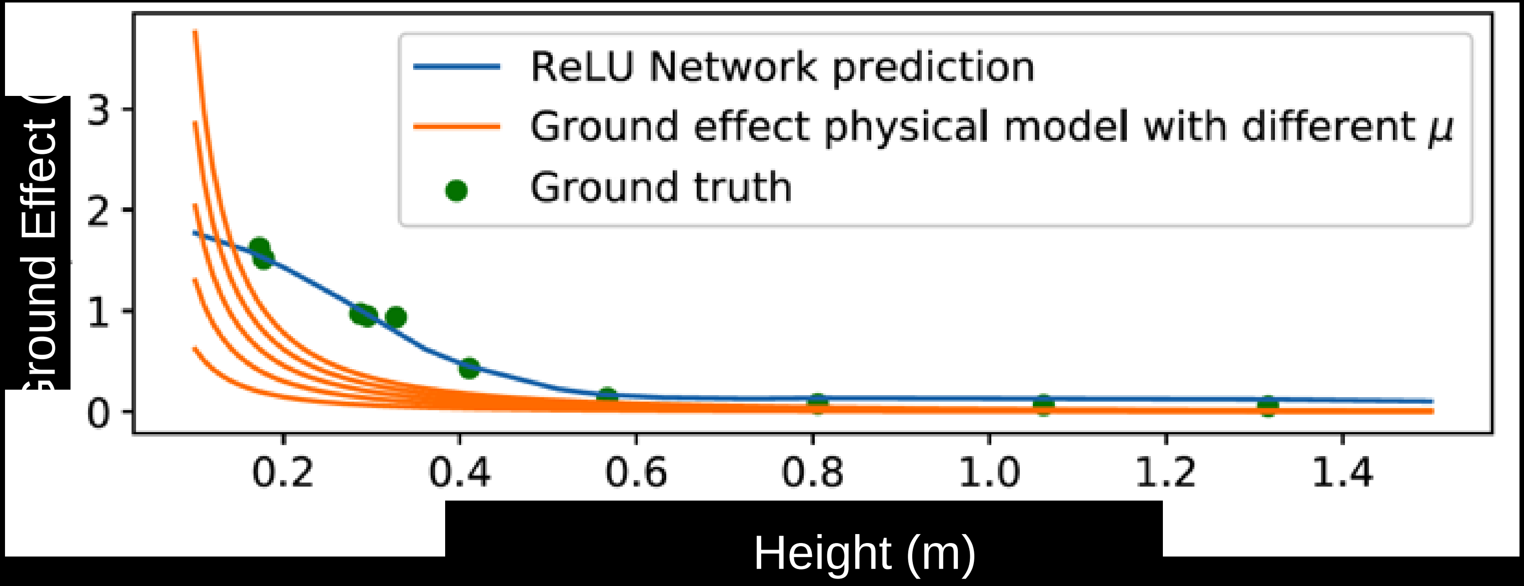
Spectral Normalization:
Ensures $\widetilde{F}$ is Lipshitz
[Bartlett NeurIPS 2017]
[Miyato et al., ICLR 2018]

Spectral-Normalized
4-Layer Feed-Forward

- Learn ground effect: $\tilde{F}(s,u)$ $\rightarrow$ $\mathbf{f}_a = [f_{a,x}, f_{a,y}, f_{a,z}]^\top$
- (s,u): height, velocity, attitude and four control inputs

Ongoing Research:
Safe Exploration

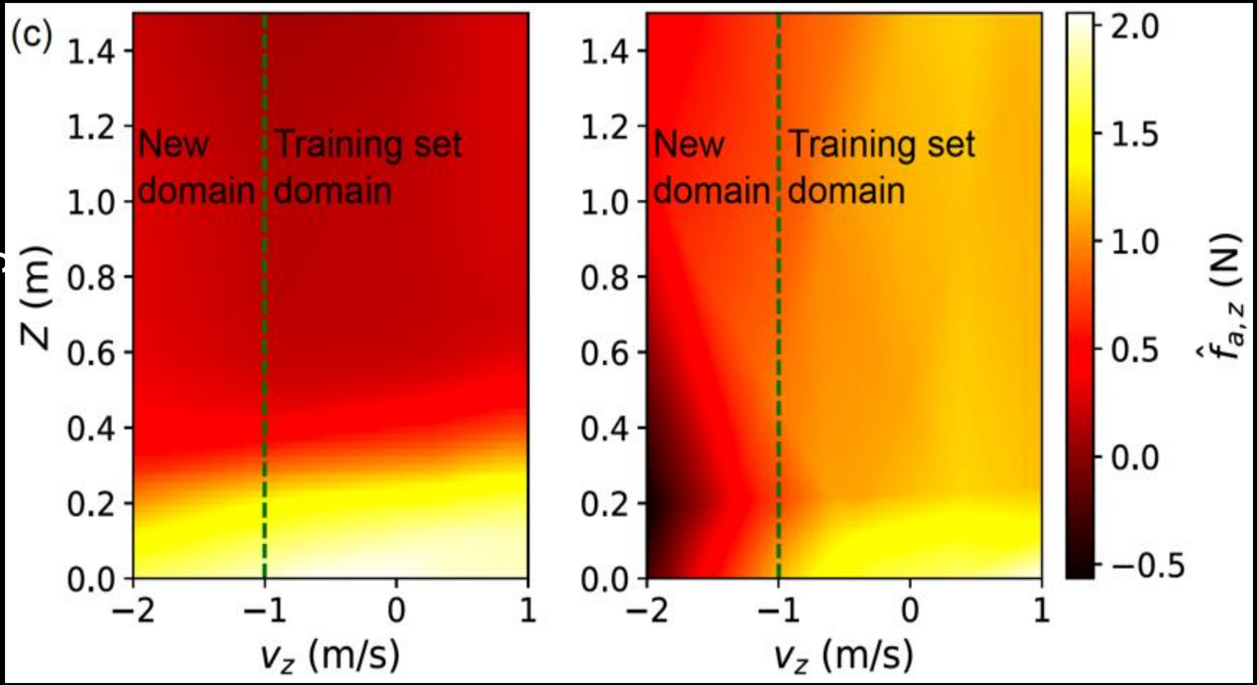
PREDICTION RESULTS



GENERALIZATION PERFORMANCE ON DRONE

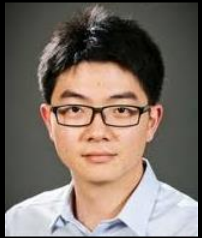
Spectral Regularized NN Conventional NN

Height



Vertical Velocity

Guanya Shi



Xichen Shi

Michael O'Connell

Yisong Yue

Rose Yu

Kamyar Azizzadenesheli

Soon-Jo Chung

Neural Lander: Stable Drone Landing Control using Learned Dynamics, ICRA 2019

CONTROLLER DESIGN (SIMPLIFIED)

Nonlinear Feedback Linearization:

$$u_{nominal} = K_s \eta \qquad \eta = \begin{bmatrix} p - p^* \\ v - v^* \end{bmatrix}$$

Desired Trajectory
(tracking error)

Feedback Linearization (PD control)

Cancel out ground effect $\tilde{F}(s, u_{old})$: $u = u_{nominal} + u_{residual}$

Requires Lipschitz & small time delay

STABILITY GUARANTEES

Assumptions:

Desired states along position trajectory bounded

Control updates faster than state dynamics

Learning error bounded (new): Bounded Lipschitz (through spectral normalization of layers

Stability Guarantee:(simplified)

control gain

Time delay

Unmodeled disturbance

$$\|\eta(t)\| \leq \|\eta(0)\| \exp\left\{\frac{\lambda \quad - \tilde{L}\rho}{C} t\right\} + \frac{\epsilon}{\lambda \quad - \tilde{L}\rho}$$

Lipschitz of NN

$$\Rightarrow \|\eta(t)\| \to \frac{\epsilon}{\lambda_{min}(K) - \tilde{L}\rho}$$

Exponentially fast

CAST @ CALTECH

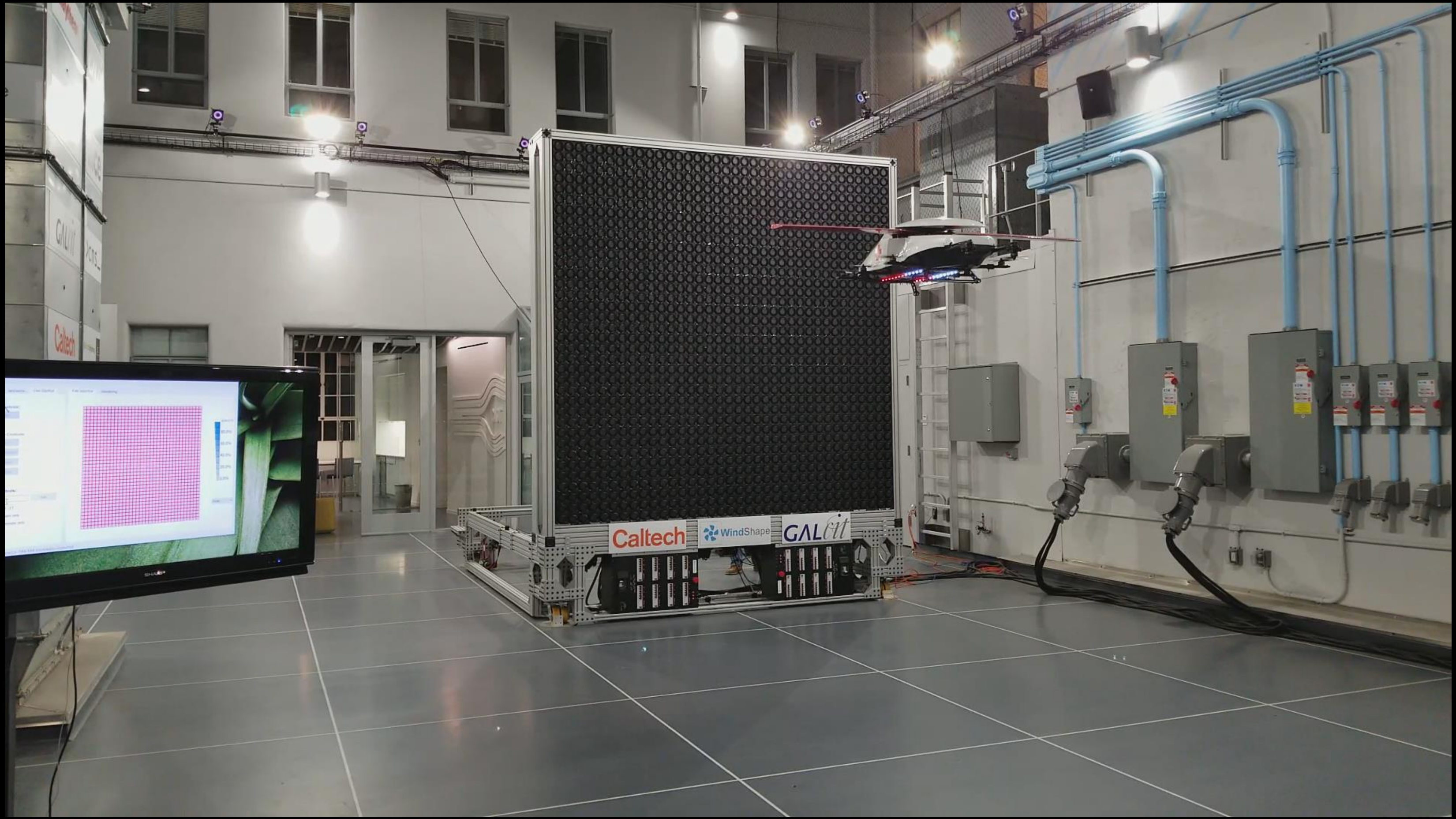
LEARNING TO LAND

3D Landing Performance

TESTING TRAJECTORY TRACKING

Move around a circle super close to the ground

CAST @ CALTECH
DRONE WIND TESTING LAB



TAKEAWAYS

Control methods => analytic guarantees

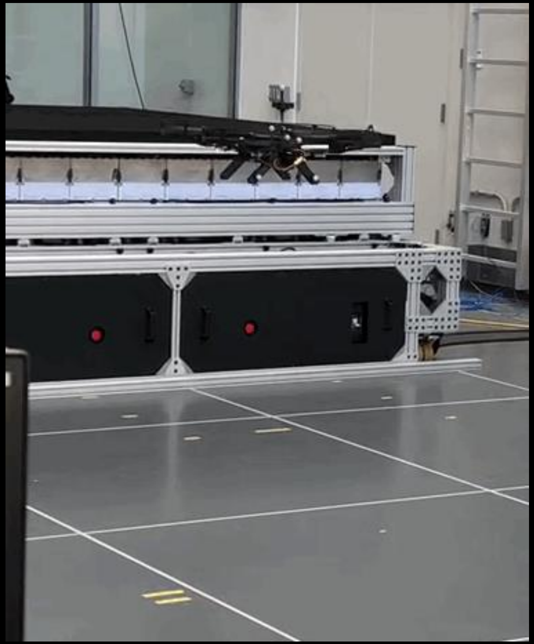
(side guarantees)

Blend w/ learning => improve precision/flexibility

Preserve side guarantees (possibly relaxed)

Sometimes interpret as functional regularization

(speeds up learning)



Blending Data Driven Learning with Symbolic Reasoning

AGE-OLD DEBATE IN AI

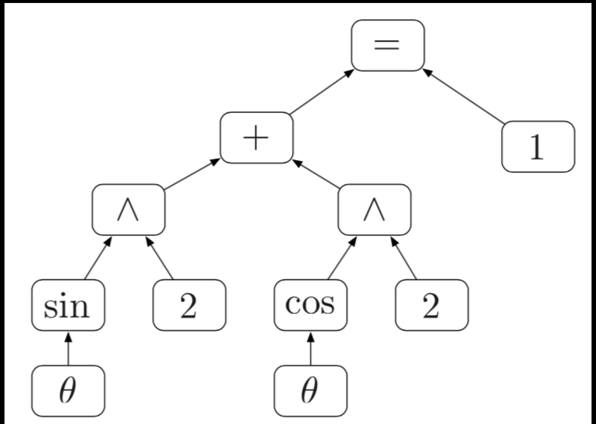
Symbols vs. Representations

Symbolic reasoning:

- Humans have impressive ability at symbolic reasoning
- Compositional: can draw complex inferences from simple axioms.

Representation learning:

- Data driven: Do not need to know the base concepts
- Black box and not compositional: cannot easily combine and create more complex systems.



$$\sin^2(\theta) + \cos^2(\theta) = 1$$

Forough Arabshahi

Sameer Singh

Combining Symbolic Expressions & Black-box Function Evaluations in Neural Programs, ICLR 2018

SYMBOLIC + NUMERICAL INPUT

Goal: Learn a domain of functions (sin, cos, log…)

Training on numerical input-output does not generalize.

Data Augmentation with Symbolic Expressions

Efficiently encode relationships between functions.

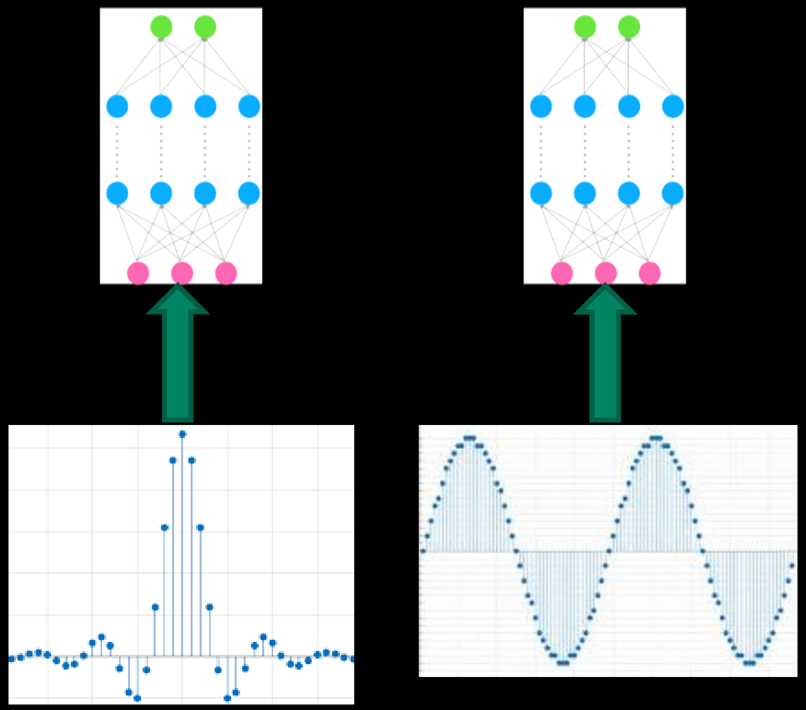
Solution:

Design networks to use both

symbolic + numeric

Leverage the observed structure of the data

Hierarchical expressions



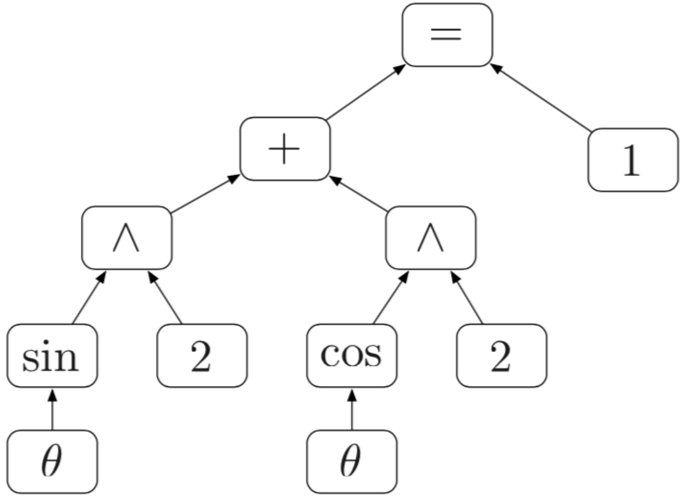
TASKS CONSIDERED

- Mathematical equation verification
  - $\sin^2\theta + \cos^2\theta = 1$???
- Mathematical question answering
  - $\sin^2\theta + \quad ^2 = 1$
- Solving differential equations

$$\frac{d^2 f(x)}{dx^2} + 4f(x) = \sin(2x)$$

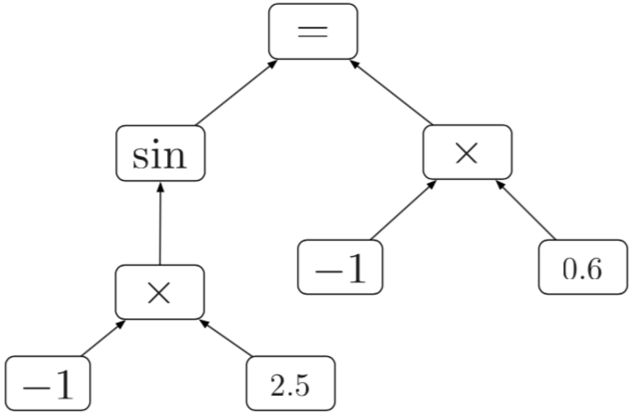
$$f(x) : \frac{1}{8}\sin(2x) - \frac{x}{4}\cos(2x)$$

EXPLOITING HIERARCHICAL REPRESENTATIONS



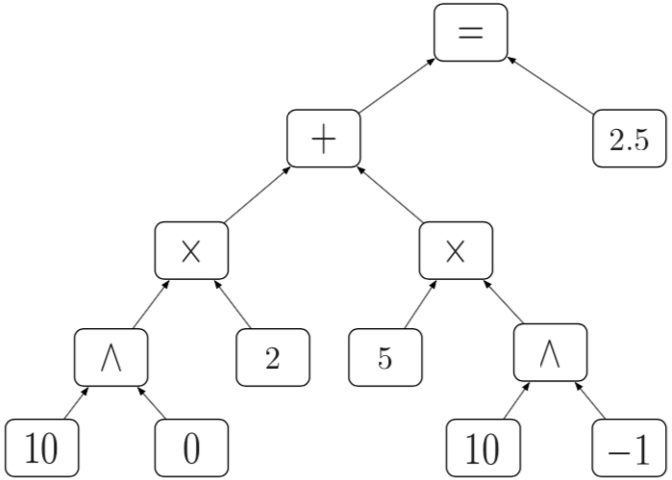
$$\sin^2(\theta) + \cos^2(\theta) = 1$$

Symbolic expression



$$\sin(-2.5) = -0.6$$

Function Evaluation Data Point



decimal tree for 2.5

Number Encoding Data Point

REPRESENTING MATHEMATICAL EQUATIONS

- Grammar rules

$$
\begin{aligned}
I &\rightarrow \; =(E, E), \; \neq(E, E) \\
E &\rightarrow T, F_1(E), F_2(E, E) \\
F_1 &\rightarrow \; \sin, \cos, \tan, \ldots \\
F_2 &\rightarrow +, \wedge, \times, \mathrm{diff}, \ldots \\
T &\rightarrow \; -1, 0, 1, 2, \pi, x, y, \ldots, \\
&\quad \text{floating point numbers of precision 2}
\end{aligned}
$$

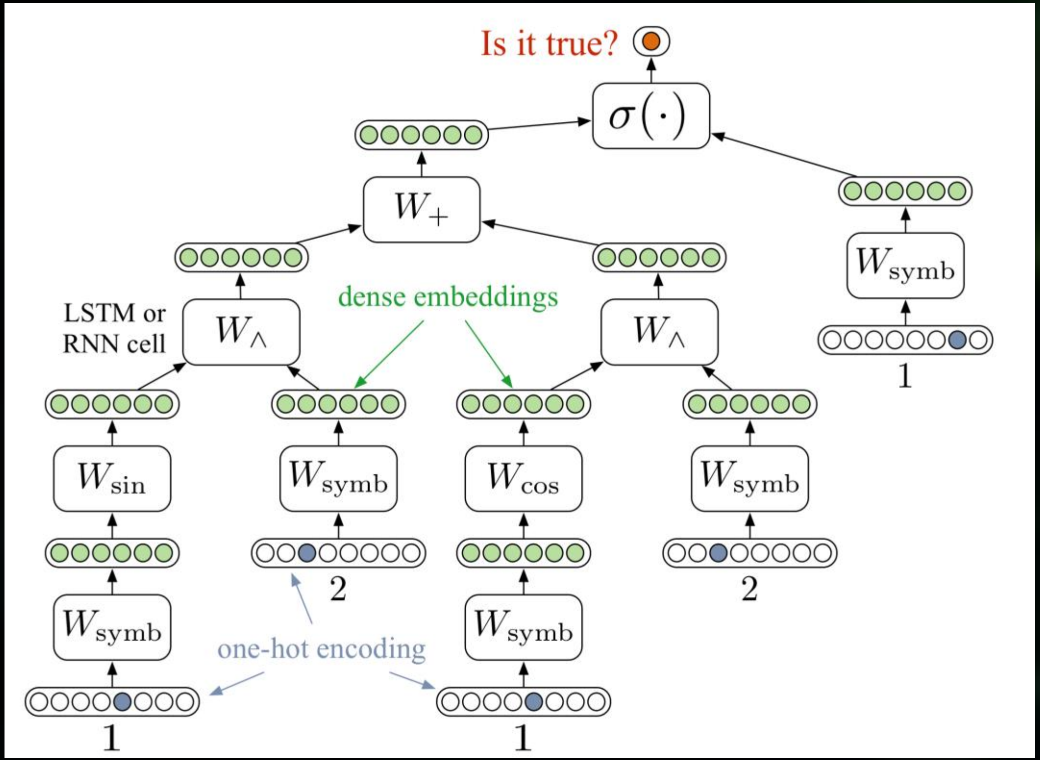
DOMAIN

Unary functions, F_1				
sin	cos	csc	sec	tan
cot	arcsin	arccos	arccsc	arcsec
arctan	arccot	sinh	cosh	csch
sech	tanh	coth	arsinh	arcosh
arcsch	arsech	artanh	arcoth	exp

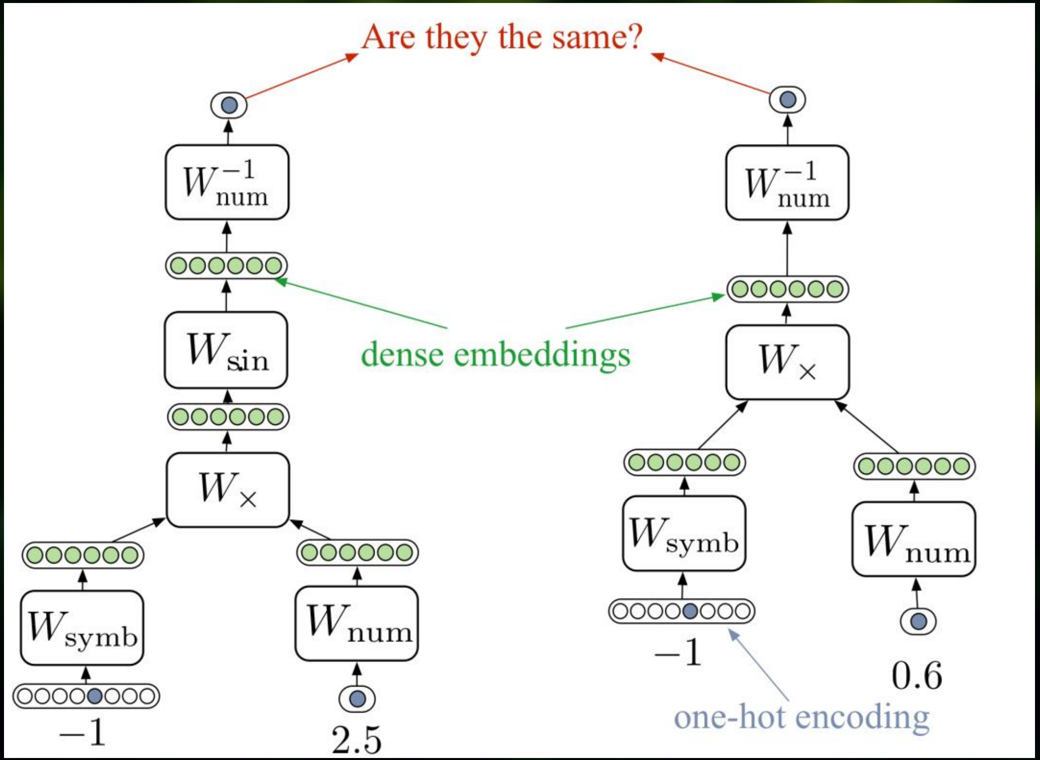
Terminal, T	
0	1
2	3
4	10
0.5	-1
0.4	0.7
π	x

Binary, F_2
$+$
$\times$
$\wedge$
diff

TREE-LSTM FOR CAPTURING HIERARCHIES



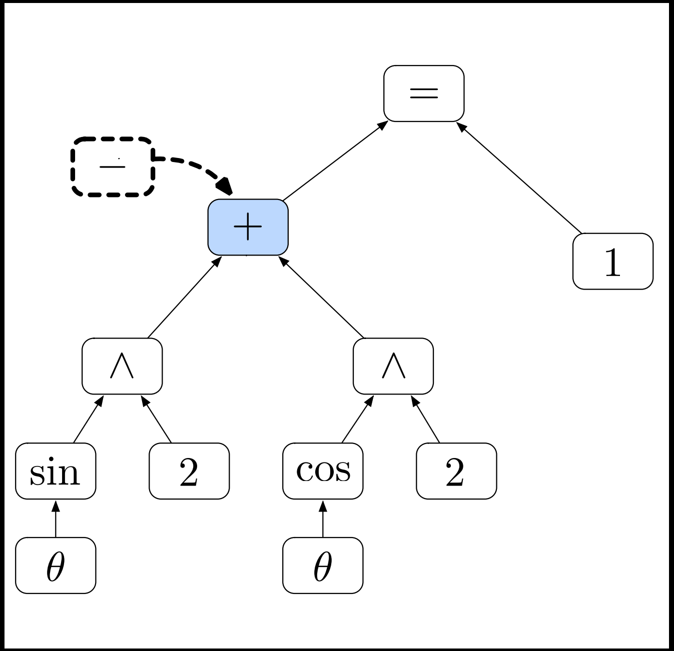
$$\sin^2(\theta) + \cos^2(\theta) = 1$$



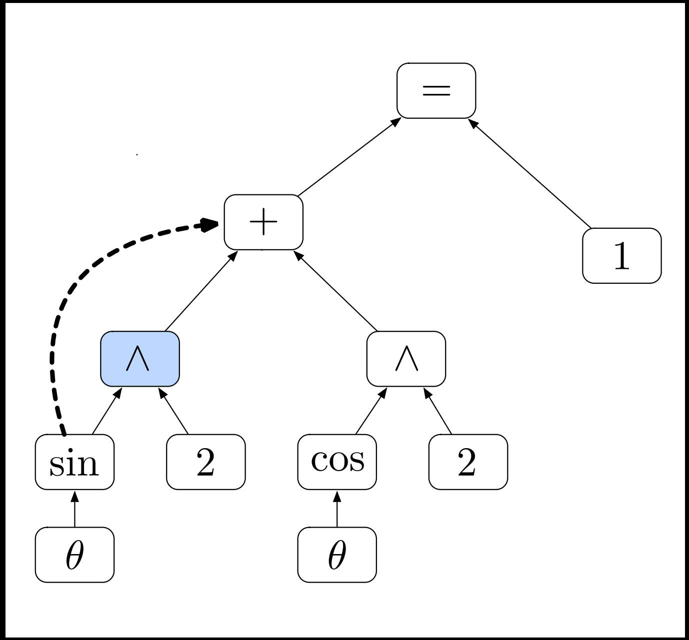
$$\sin(-2.5) = -0.6$$

DATASET GENERATION

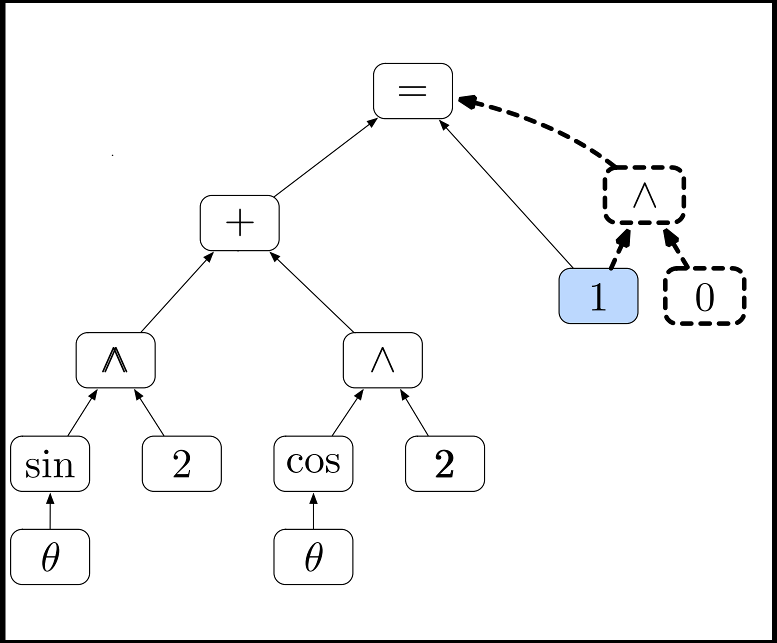
- Random local changes



Replace Node



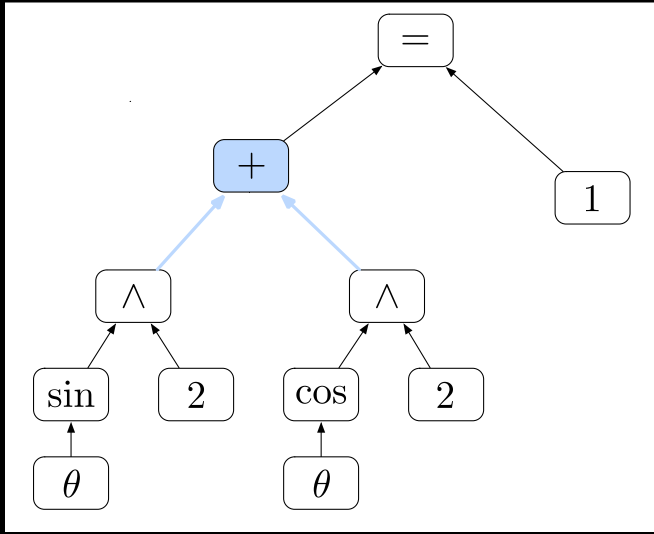
Shrink Node



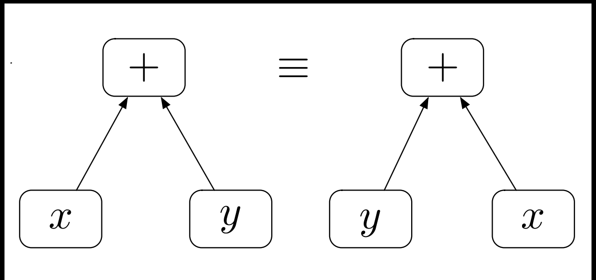
Expand Node

DATASET GENERATION

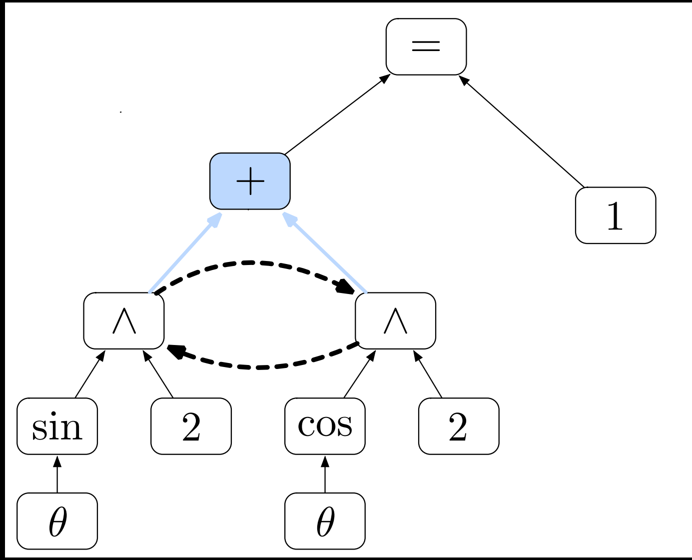
- Sub-tree matching



Choose Node



Dictionary key-value pair



Replace with value’s pattern

EQUATION VERIFICATION

	Majority Class	Sympy	LSTM : sym	TreeLSTM : sym	TreeLSTM:sym+num
Generalization	50.24%	81.74%	81.71%	95.18%	97.20%
Extrapolation	44.78%	71.93%	76.40%	93.27%	96.17%

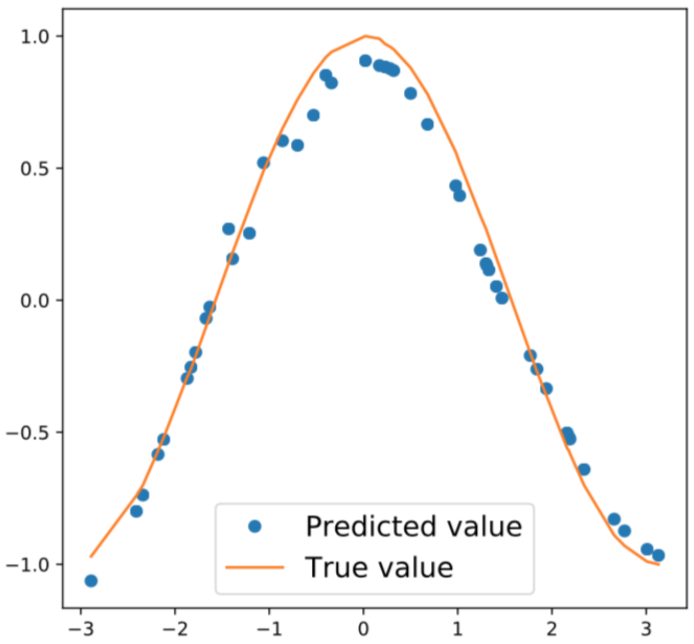
EQUATION COMPLETION

$$4^{\tanh(0)} = \blacksquare^{x}$$

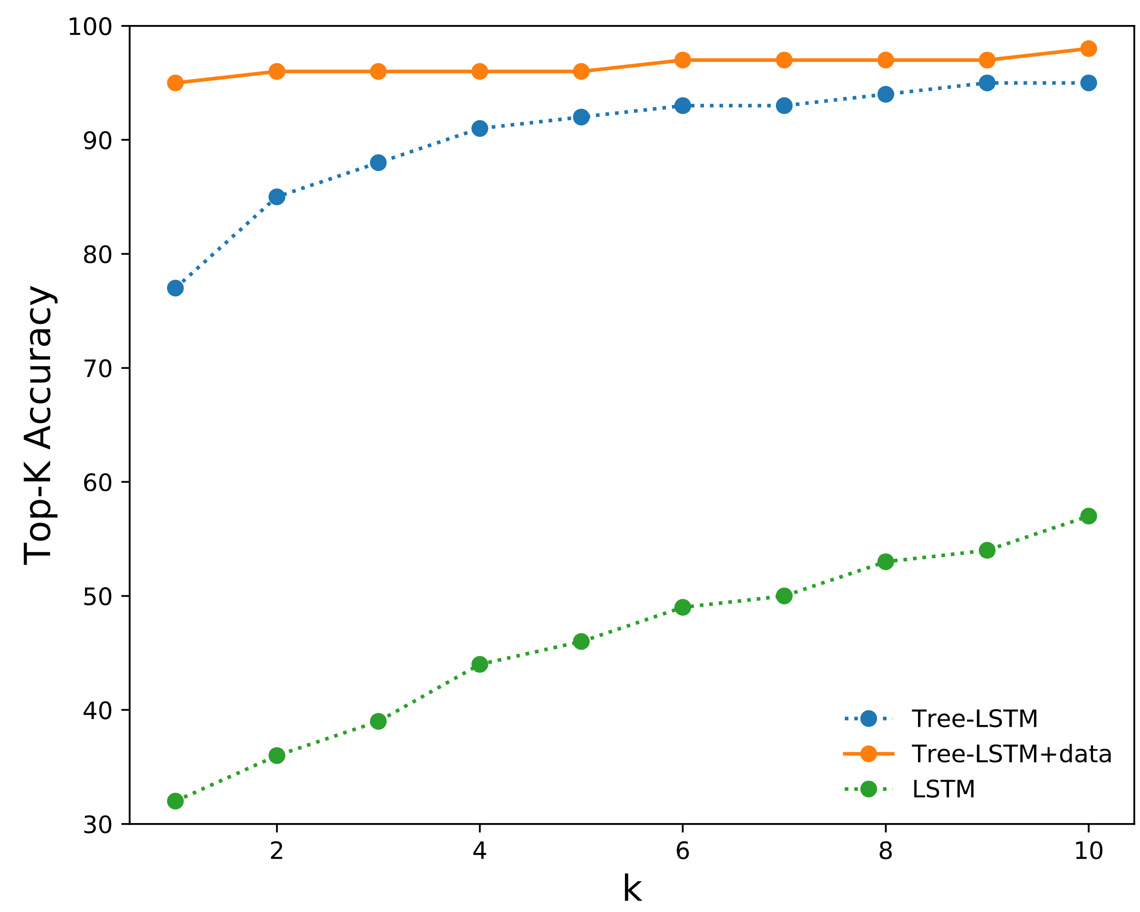
pred	prob
-2^0	**0.9999**
1^0	**0.9999**
7^0	**0.9999**
-3^0	**0.9999**

$$\cos(-\blacksquare) = -0.57$$

pred	modelErr	trueErr
3	$1.8e{-5}$	$1.7e{-1}$
2.17	$1.9e{-5}$	$9.9e{-5}$
2.16	$2.6e{-5}$	$3.9e{-4}$
2.18	$\mathbf{1.9e{-4}}$	**0**



EQUATION COMPLETION



TAKE-AWAYS

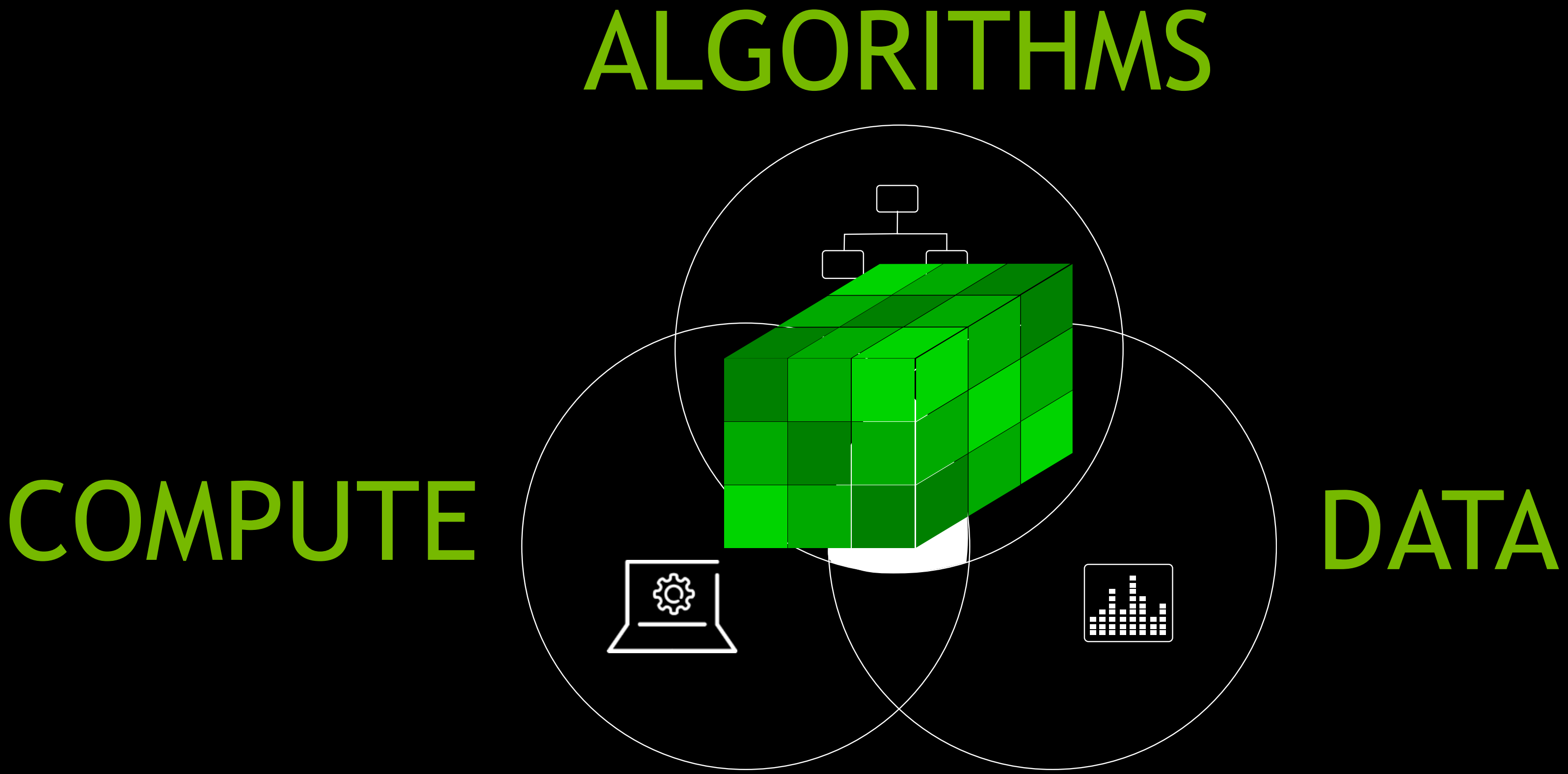
Vastly Improved numerical evaluation: 90% over function-fitting baseline.

Generalization to verifying symbolic equations of higher depth

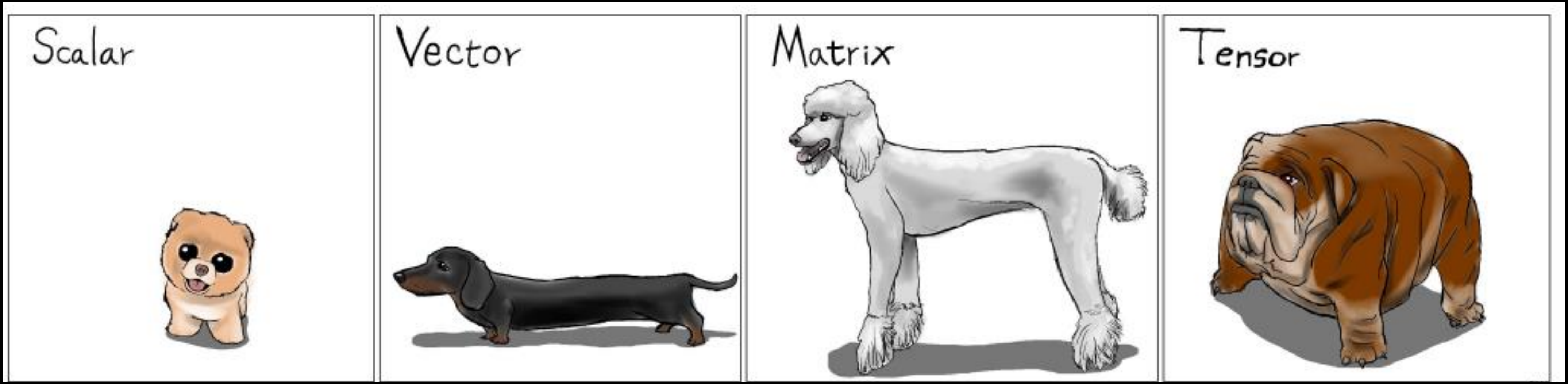
LSTM: Symbolic	TreeLSTM: Symbolic	TreeLSTM: symbolic + numeric
76.40 %	93.27 %	**96.17 %**

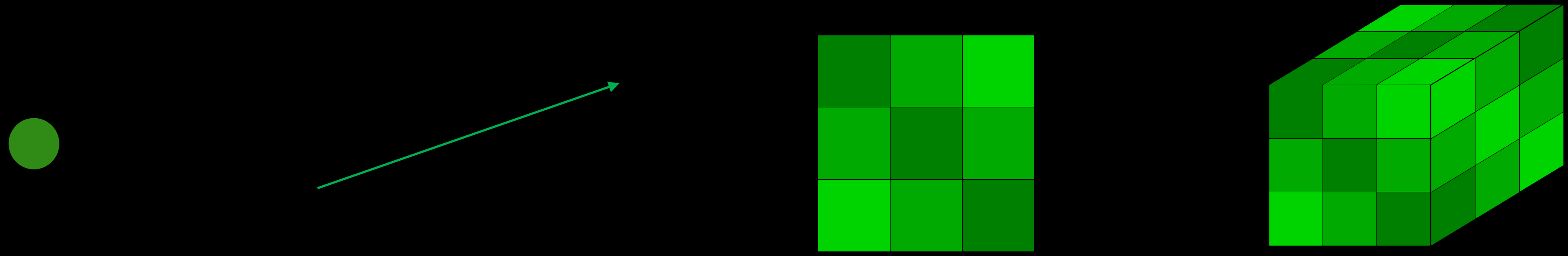
Combining symbolic + numerical data helps in better generalization for both tasks: symbolic and numerical evaluation.

TENSORS PLAY A CENTRAL ROLE



TENSOR : EXTENSION OF MATRIX





TENSORS FOR DATA
ENCODE MULTI-DIMENSIONALITY

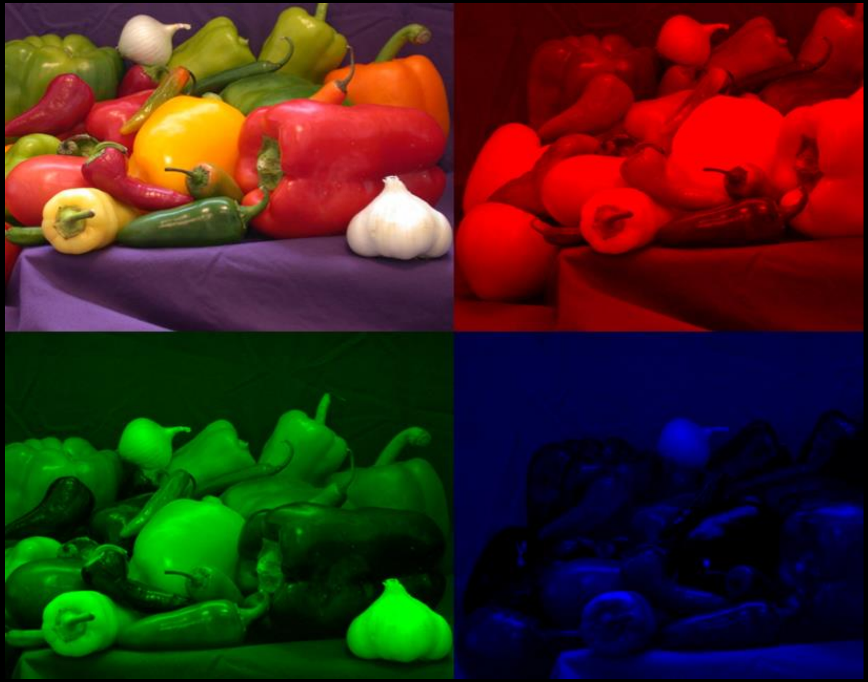
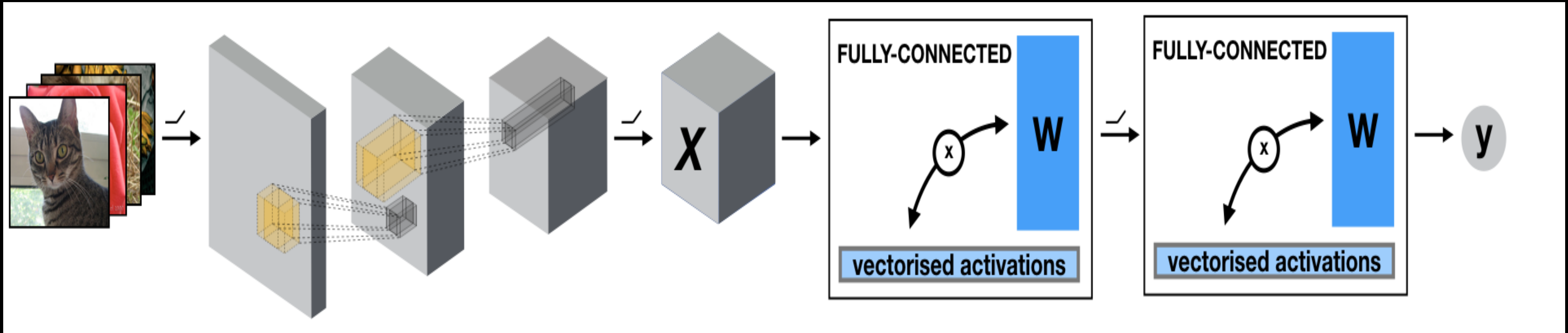


Image: 3 dimensions
Width * Height * Channels

Video: 4 dimensions
Width * Height * Channels * Time

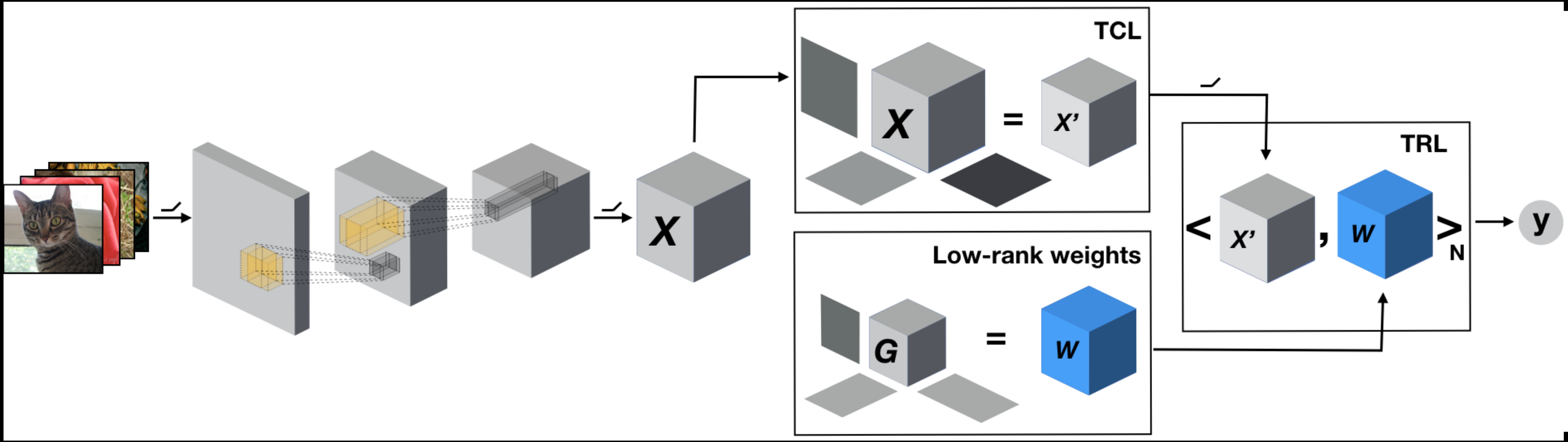
TENSORS FOR MODELS

STANDARD CNN USE LINEAR ALGEBRA



TENSORS FOR MODELS

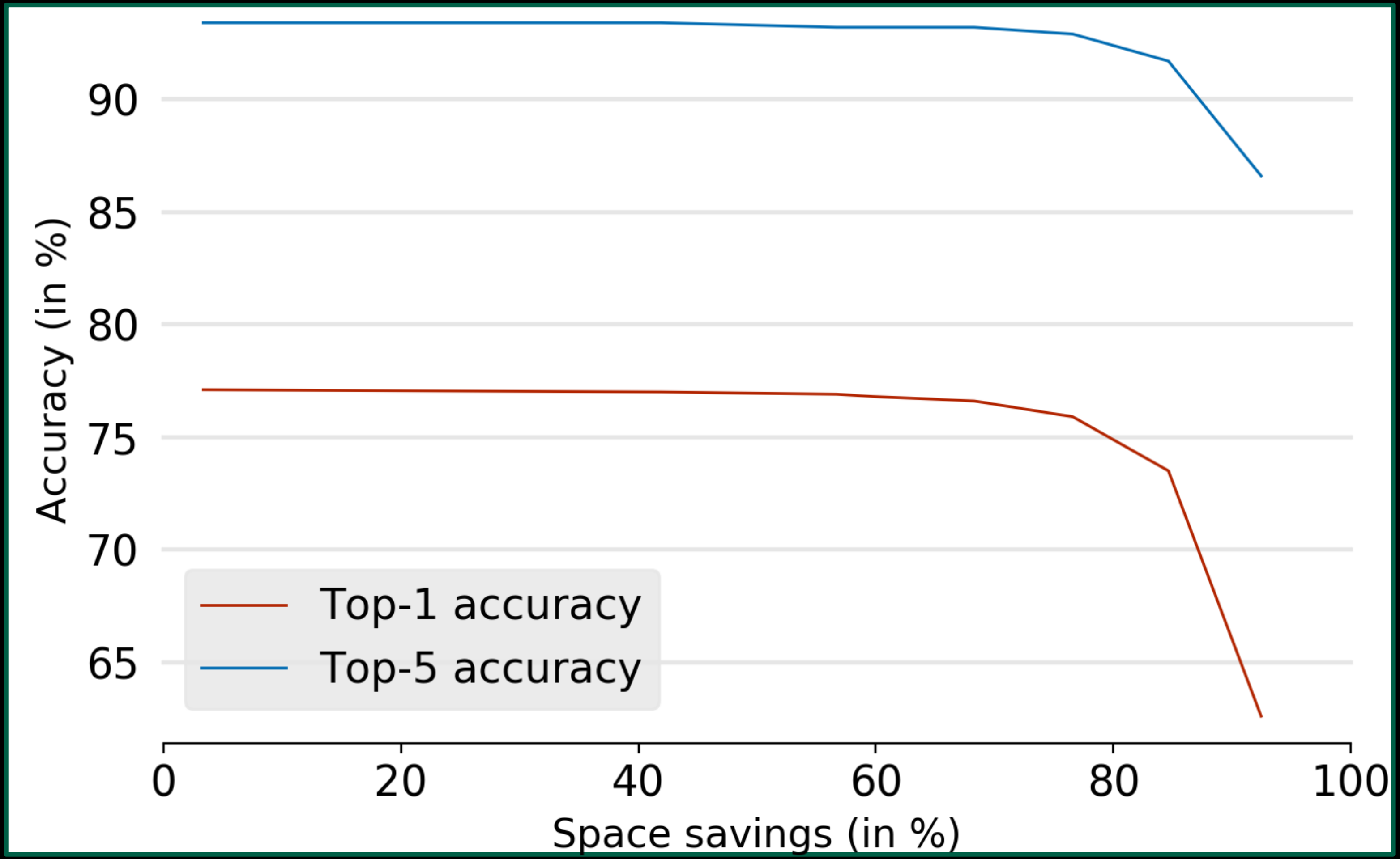
TENSORIZED NEURAL NETWORKS



Jean Kossaifi, Zack Chase Lipton, Aran Khanna, Tommaso Furlanello, **A**

Jupyters notebook: https://github.com/JeanKossaifi/tensorly-notebooks

SPACE SAVING IN DEEP TENSORIZED NETWORKS



TENSORLY: HIGH-LEVEL API FOR TENSOR ALGEBRA

Tensor decomposition | Tensor regression | Tensors + Deep

Basic tensor operations

Unified backend

NumPy | SciPy | mxnet | PYTORCH | TensorFlow

- Python programming
- User-friendly API
- Multiple backends: flexible + scalable
- Example notebooks

Jean Kossaifi

TENSORLY WITH PYTORCH BACKEND

```
import tensorly as tl
from tensorly.random import tucker_tensor

tl.set_backend('pytorch')
core, factors = tucker_tensor((5, 5, 5),
                              rank=(3, 3, 3))
core = Variable(core, requires_grad=True)
factors = [Variable(f, requires_grad=True) for f in factors]

optimiser = torch.optim.Adam([core]+factors, lr=lr)

for i in range(1, n_iter):
    optimiser.zero_grad()
    rec = tucker_to_tensor(core, factors)
    loss = (rec - tensor).pow(2).sum()
    for f in factors:
        loss = loss + 0.01*f.pow(2).sum()

    loss.backward()
    optimiser.step()
```

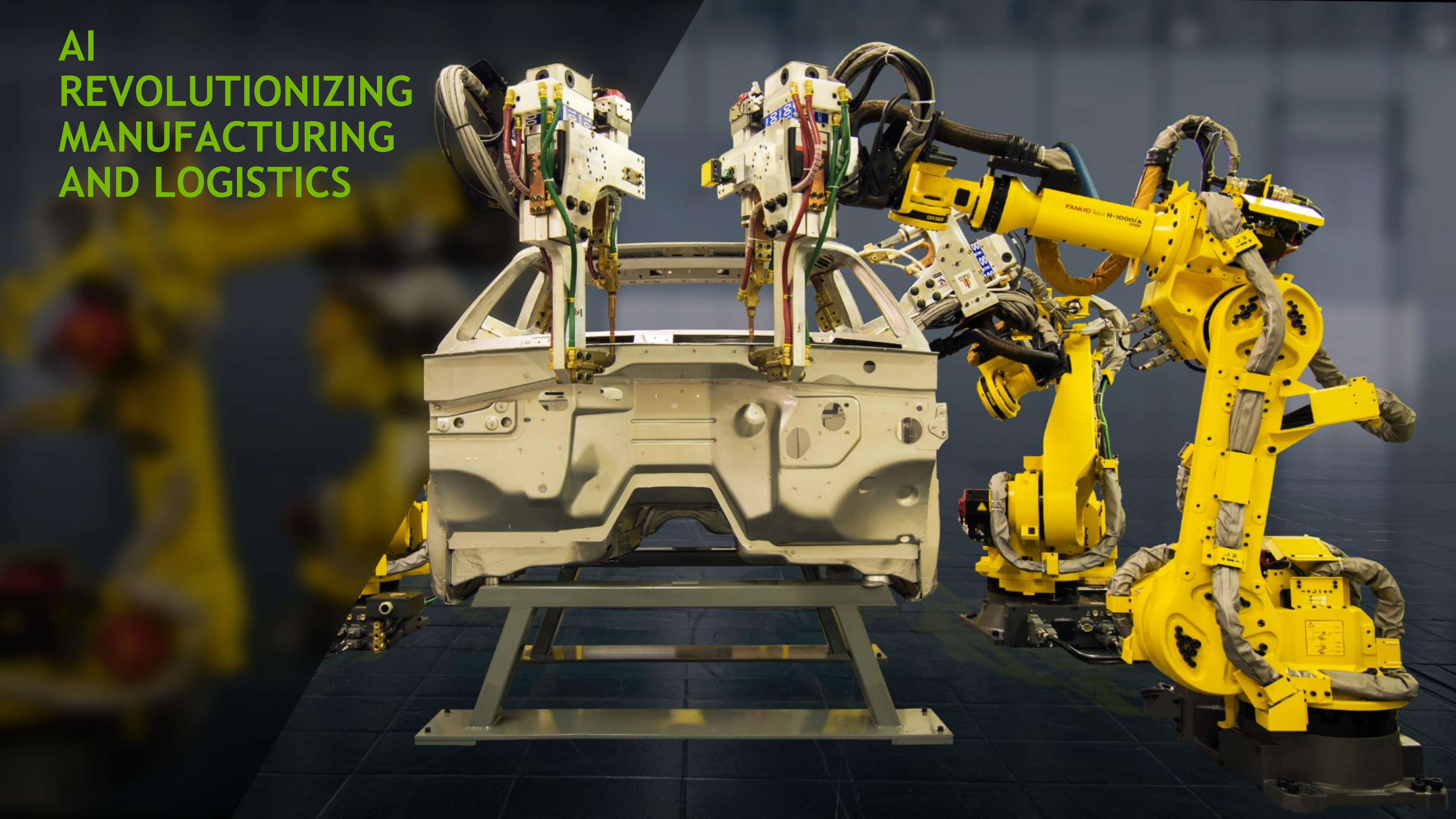
Set Pytorch backend

Tucker Tensor form

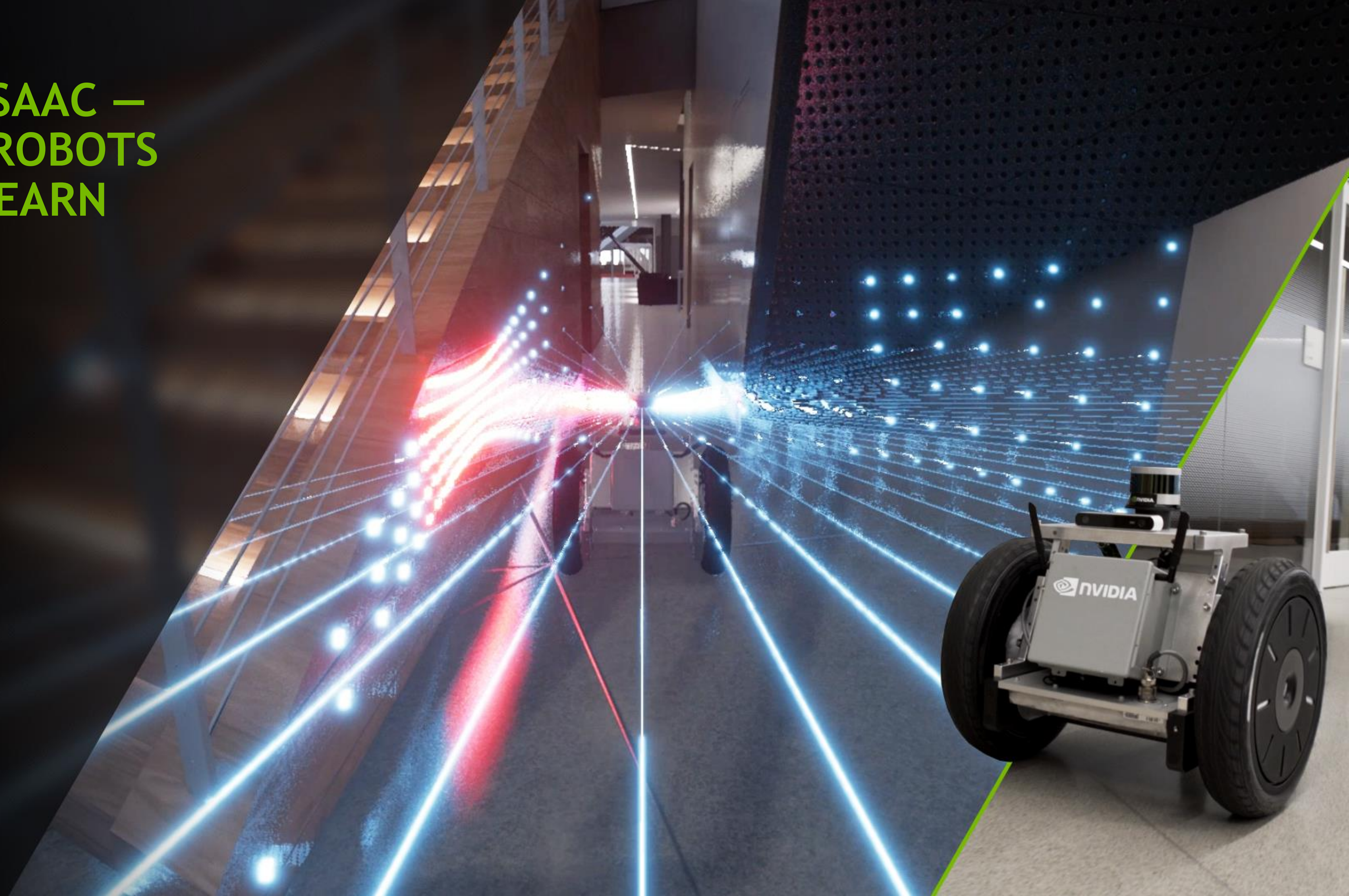
Attach gradients

Set optimizer

AI REVOLUTIONIZING MANUFACTURING AND LOGISTICS



NVIDIA ISAAC — WHERE ROBOTS GO TO LEARN



NVIDIA DRIVE
FROM TRAINING TO SAFETY

1. COLLECT & PROCESS DATA

2. TRAIN MODELS

3. SIMULATE

4. DRIVE

TAKEAWAYS

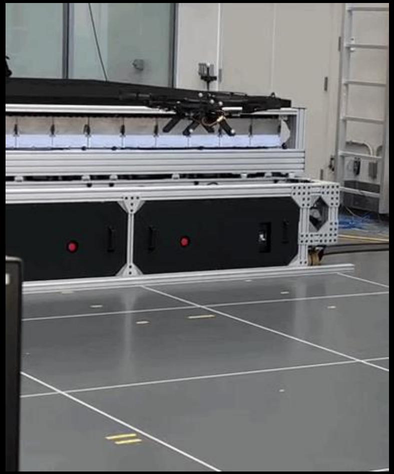
End-to-end learning from scratch is impossible in most settings

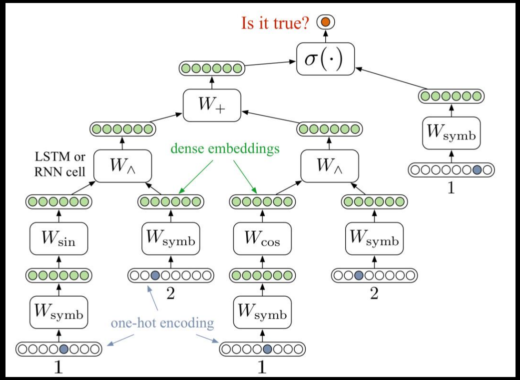
Blend DL w/ prior knowledge => improve data efficiency, generalization, model size

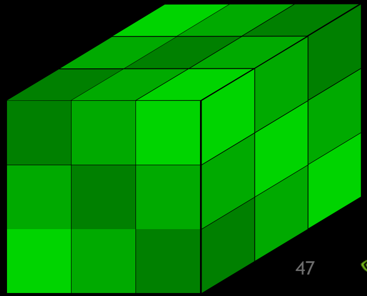
Obtain side guarantees like stability + safety,

Outstanding challenge (application dependent):

what is right blend of prior knowledge vs data?







Thank you