TRINITY OF AI/ML

ALGORITHMS

COMPUTE

DATA
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.
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 = Computational Reasoning over Data & Priors

How to use Physics as a (new) form of Prior?
  • Learnability
  • Generalization
BASELINE: MODEL-BASED CONTROL (NO LEARNING)

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

New State \rightarrow Current Action (aka control input) \rightarrow Unmodeled Disturbance / Error

Current State

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

(Value Iteration is also contraction mapping)
LEARNING RESIDUAL DYNAMICS FOR DRONE LANDING

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

New State \rightarrow Current Action (aka control input) \rightarrow Current State \rightarrow Unmodeled Disturbance

Use existing control methods to generate actions
- Provably robust (even using deep learning)
- Requires \( \tilde{f} \) Lipschitz & bounded error
CONTROL SYSTEM FORMULATION

- **Dynamics:**
  \[
  \dot{\mathbf{p}} = \mathbf{v}, \quad m\dot{\mathbf{v}} = mg + R\mathbf{f}_u + \mathbf{f}_a
  \]
  \[
  \dot{\mathbf{R}} = RS(\omega), \quad J\dot{\omega} = J\omega \times \omega + \tau_u + \tau_a
  \]

- **Control:**
  \[
  \mathbf{f}_u = [0, 0, T]^T
  \]
  \[
  \tau_u = [\tau_x, \tau_y, \tau_z]^T
  \]

\[
\begin{bmatrix}
T \\
\tau_x \\
\tau_y \\
\tau_z
\end{bmatrix} = \begin{bmatrix}
cT & cT & cT & cT \\
0 & -cTl_{\text{arm}} & 0 & -cTl_{\text{arm}} \\
-cTl_{\text{arm}} & 0 & cTl_{\text{arm}} & 0 \\
-cQ & cQ & -cQ & cQ
\end{bmatrix} \begin{bmatrix}
n_1^2 \\
n_2^2 \\
n_3^2 \\
n_4^2
\end{bmatrix}
\]

- **Unknown forces & moments:**
  \[
  \mathbf{f}_a = [f_{a,x}, f_{a,y}, f_{a,z}]^T
  \]
  \[
  \tau_a = [\tau_{a,x}, \tau_{a,y}, \tau_{a,z}]^T
  \]

Learn the Residual (function of state and control input)
DATA COLLECTION (MANUAL EXPLORATION)

- 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

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

Ongoing Research: Safe Exploration

Spectral-Normalized 4-Layer Feed-Forward
PREDICTION RESULTS

![Graph showing prediction results for height and ground effect](image)

- **Ground Effect (N)**
- **Height (m)**

- Blue line: ReLU Network prediction
- Orange line: Ground effect physical model with different $\mu$
- Green dots: Ground truth
GENERALIZATION PERFORMANCE ON DRONE

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

\[ u_{\text{nominal}} = K_s \eta \]

\[ \eta = \begin{bmatrix} p - p^* \\ v - v^* \end{bmatrix} \]

Desired Trajectory (tracking error)

Feedback Linearization (PD control)

Cancel out ground effect \( \tilde{F}(s, u_{\text{old}}): \)

\[ u = u_{\text{nominal}} + u_{\text{residual}} \]

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)

\[
\|\eta(t)\| \leq \|\eta(0)\| \exp \left\{ \frac{\lambda}{c} - \frac{L \rho}{c} t \right\} + \frac{\epsilon}{\lambda - L \rho}
\]

\[\Rightarrow \|\eta(t)\| \rightarrow \frac{\epsilon}{\lambda_{\text{min}}(K) - L \rho}\]

Exponentially fast
CAST @ CALTECH
LEARNING TO LAND

3D Landing Performance
TESTING TRAJECTORY TRACKING

Move around a circle super close to the ground
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.

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

Forough Arabshahi
Sameer Singh
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
  ○ \( \sin^2 \theta + \theta^2 = 1 \)

◎ 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

\[ \sin^2(\theta) + \cos^2(\theta) = 1 \]

\[ \sin(-2.5) = -0.6 \]

decimal tree for 2.5

Symbolic expression | Function Evaluation Data Point | Number Encoding Data Point
Grammar rules

\[ I \rightarrow = (E, E), \neq (E, E) \]
\[ E \rightarrow T, F_1(E), F_2(E, E) \]
\[ F_1 \rightarrow \sin, \cos, \tan, \ldots \]
\[ F_2 \rightarrow +, \land, \times, \text{diff}, \ldots \]
\[ T \rightarrow -1, 0, 1, 2, \pi, x, y, \ldots, \]

floating point numbers of precision 2
<table>
<thead>
<tr>
<th><strong>Unary functions, ( F_1 )</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sin )</td>
<td>( \cos )</td>
</tr>
<tr>
<td>( \cot )</td>
<td>( \arcsin )</td>
</tr>
<tr>
<td>( \arctan )</td>
<td>( \arccot )</td>
</tr>
<tr>
<td>( \sech )</td>
<td>( \tanh )</td>
</tr>
<tr>
<td>( \text{arcsch} )</td>
<td>( \text{arsech} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Terminal, ( T )</strong></th>
<th><strong>Binary, ( F_2 )</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>0.5</td>
<td>-1</td>
</tr>
<tr>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>( \pi )</td>
<td>( x )</td>
</tr>
</tbody>
</table>
\[ \sin^2(\theta) + \cos^2(\theta) = 1 \]

\[ \sin(-2.5) = -0.6 \]
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

<table>
<thead>
<tr>
<th></th>
<th>Generalization</th>
<th>Extrapolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Class</td>
<td>50.24%</td>
<td>44.78%</td>
</tr>
<tr>
<td>Sympy</td>
<td>81.74%</td>
<td>71.93%</td>
</tr>
<tr>
<td>LSTM : sym</td>
<td>81.71%</td>
<td>76.40%</td>
</tr>
<tr>
<td>TreeLSTM : sym</td>
<td>95.18%</td>
<td>93.27%</td>
</tr>
<tr>
<td>TreeLSTM:sym+num</td>
<td>97.20%</td>
<td>96.17%</td>
</tr>
</tbody>
</table>
\[ 4 \tanh(0) = \boxed{x} \]

<table>
<thead>
<tr>
<th>pred</th>
<th>prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-2^0)</td>
<td>0.9999</td>
</tr>
<tr>
<td>(1^0)</td>
<td>0.9999</td>
</tr>
<tr>
<td>(7^0)</td>
<td>0.9999</td>
</tr>
<tr>
<td>(-3^0)</td>
<td>0.9999</td>
</tr>
</tbody>
</table>

\[ \cos(\boxed{-\text{value}}) = -0.57 \]

<table>
<thead>
<tr>
<th>pred</th>
<th>modelErr</th>
<th>trueErr</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1.8e-5</td>
<td>1.7e-1</td>
</tr>
<tr>
<td>2.17</td>
<td>1.9e-5</td>
<td>9.9e-5</td>
</tr>
<tr>
<td>2.16</td>
<td>2.6e-5</td>
<td>3.9e-4</td>
</tr>
<tr>
<td>2.18</td>
<td>1.9e-4</td>
<td>0</td>
</tr>
</tbody>
</table>
EQUATION COMPLETION

![Graph showing top-k accuracy for different models as a function of k. The graph compares Tree-LSTM, Tree-LSTM+data, and LSTM models.]
TAKE-AWAYS

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

Generalization to verifying symbolic equations of higher depth

<table>
<thead>
<tr>
<th>LSTM: Symbolic</th>
<th>TreeLSTM: Symbolic</th>
<th>TreeLSTM: symbolic + numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>76.40 %</td>
<td>93.27 %</td>
<td>96.17 %</td>
</tr>
</tbody>
</table>

Combining symbolic + numerical data helps in better generalization for both tasks: symbolic and numerical evaluation.
TENSORS PLAY A CENTRAL ROLE

ALGORITHMS

COMPUTE

DATA
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

Jupyter notebook: https://github.com/JeanKossaifi/tensorly-notebooks
SPACE SAVING IN DEEP TENSORIZED NETWORKS

The diagram shows the relationship between space savings and accuracy for different network configurations. The red line represents Top-1 accuracy, which remains relatively stable up to 50% space savings and then decreases sharply. The blue line represents Top-5 accuracy, which also remains stable up to 50% space savings but drops more gradually compared to Top-1 accuracy.
Tensor decomposition

Tensor regression

Tensors + Deep

Basic tensor operations

Unified backend

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

Jean Kossaifi
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()
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

Cars  Pedestrians  Path
Lanes  Signs  Lights

3. SIMULATE

4. DRIVE

Cars  Pedestrians  Path
Lanes  Signs  Lights
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