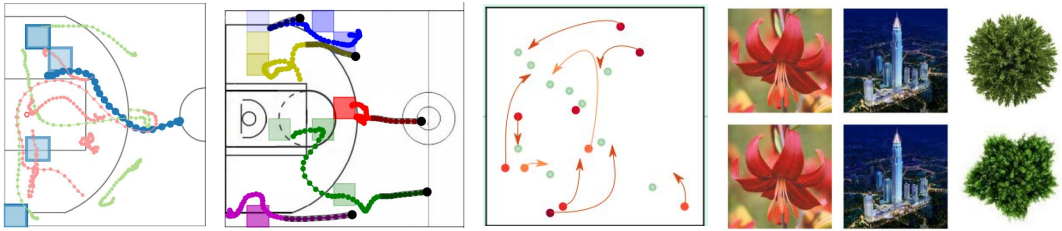
April 23, 2018



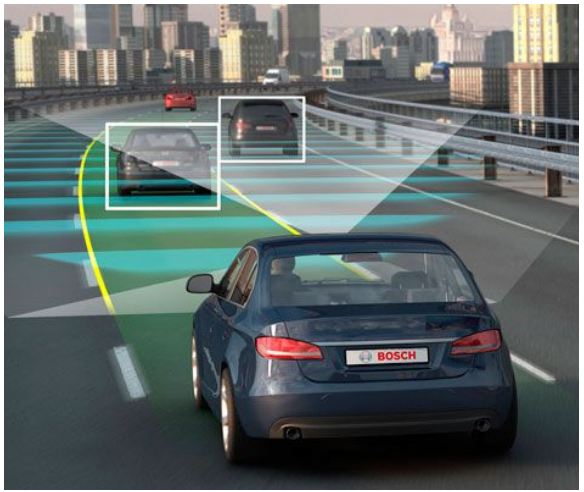
Exploiting Structure for Scalable and Robust Deep Learning

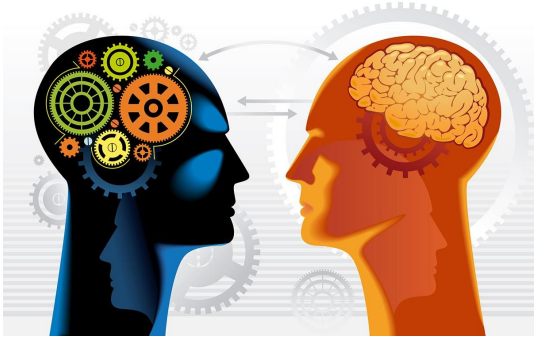
Stephan Zheng

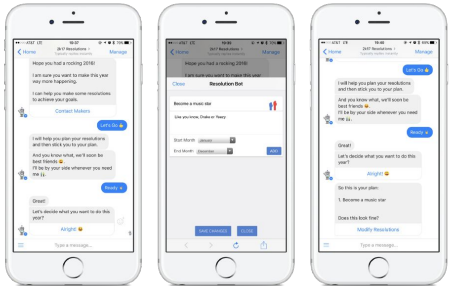
stephan@caltech.edu
www.stephanzheng.com

Caltech

Multi-agent decision-making







Markov Decision Process

Perception s
- vision
- audio
- ...

Reward r

Action a
- move
- shoot at basket
- ...

Policy $P^{\pi}(a \mid s)$

How do we learn optimal decision-making?

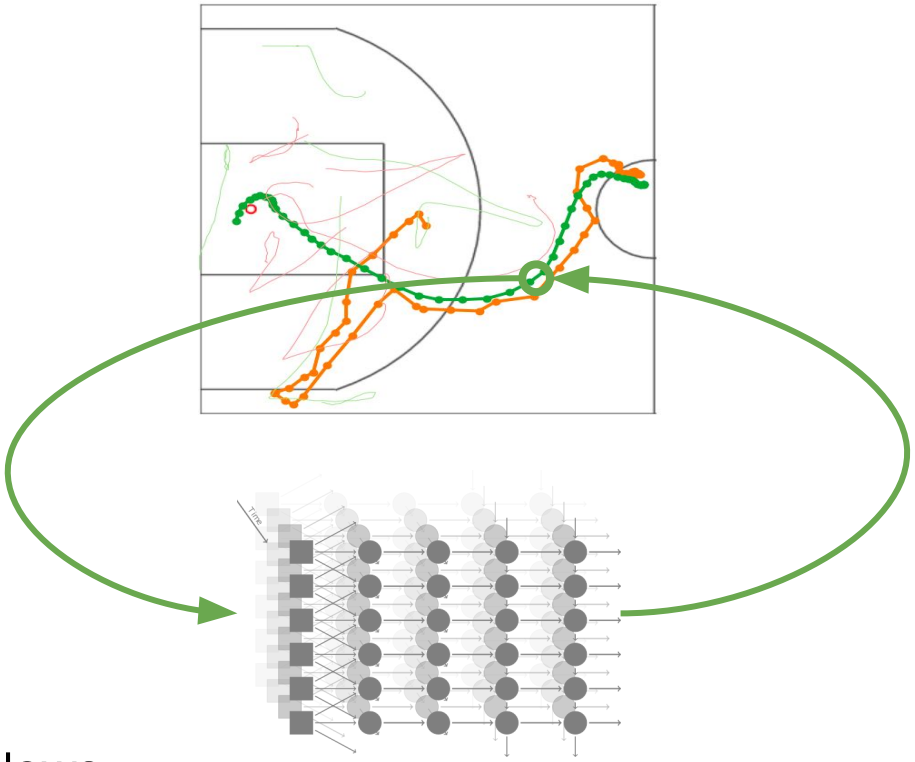
How do we measure **reward** $R = \sum_t r_t$ and optimize **policy** $P^{\pi}(a \mid s; \theta)$?

$$\max_{\pi} \mathbb{E}_{\pi}[R(\mathbf{s}_1 \ldots \mathbf{s}_T, \mathbf{a}_1 \ldots \mathbf{a}_T)]$$

Imitation learning: supervised learning from expert demonstrations

Reinforcement learning: get training data using trial-and-error (exploration)

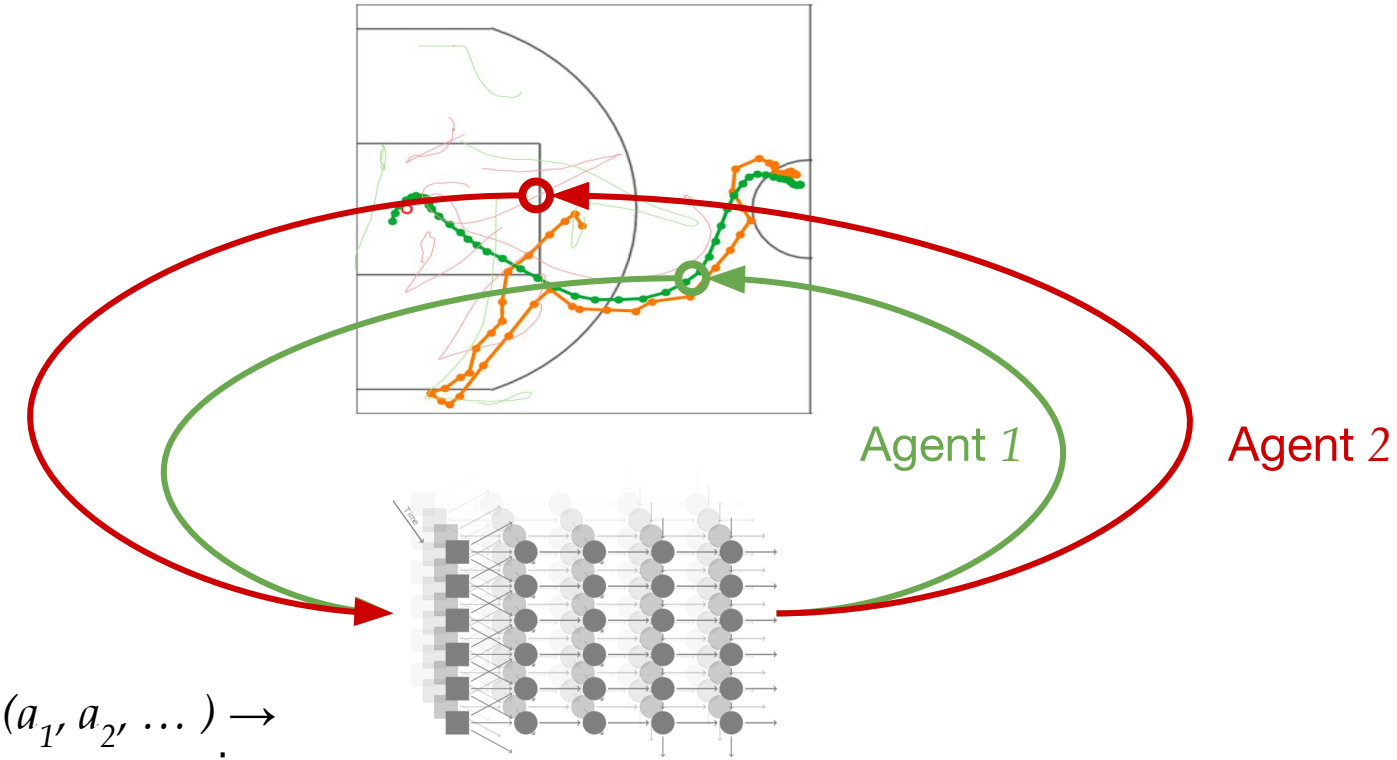
Scalability: long-term planning



$P^{\pi}(a \mid s)$

$T \gg 1$ iterations →
exponential # futures allows
arbitrary drift from data

Scalability: high-dimensional actions



Joint action $\boldsymbol{a} = (a_1, a_2, \dots) \rightarrow$ exponentially large space is hard to explore

Robustness: high-dimensional input

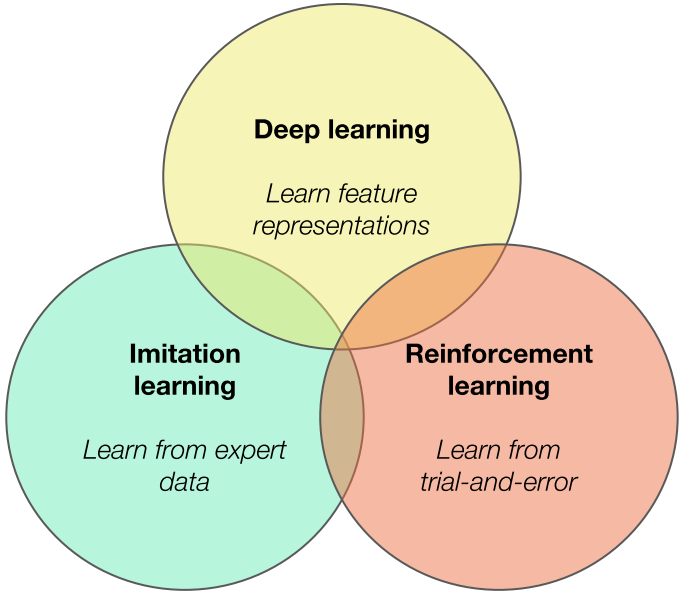
- High-dimensional input $s \rightarrow$ model is vulnerable to adversarial perturbations
- General issue in deep learning.

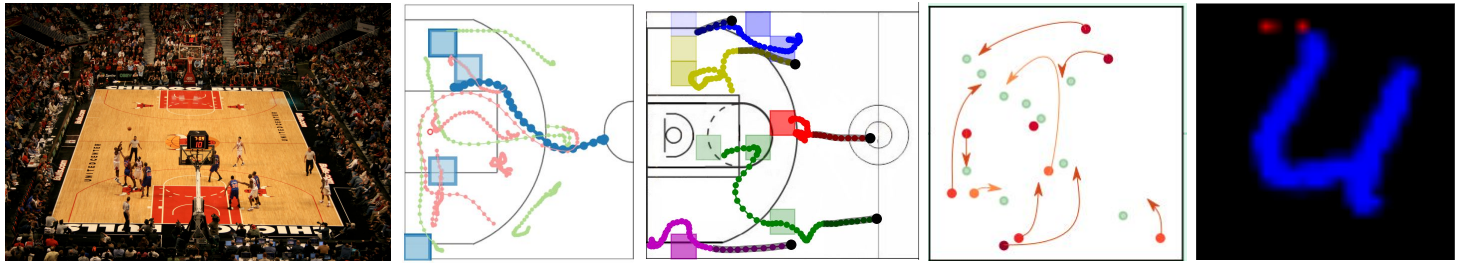
Input $s \in \mathbb{R}^{\text{num-pixels}}$

“ball”

“cake”

Efficient and robust multi-agent decision-making





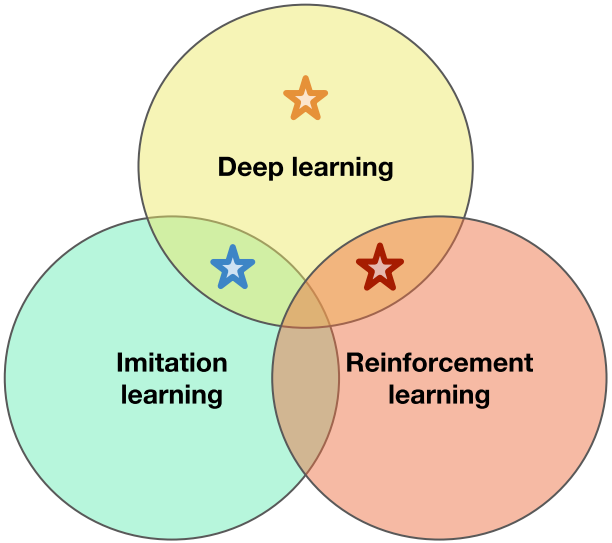
Thesis

Deep learning is a powerful tool for multi-agent decision-making:

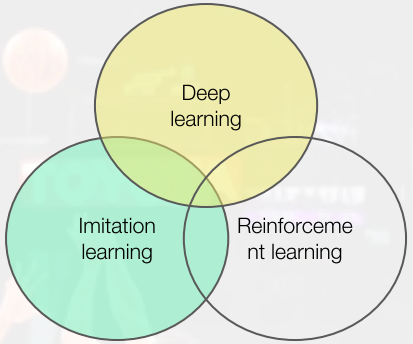
- exploiting **hierarchical** and **spatiotemporal** structure can significantly improve scalability and expressiveness, and

- deep models can be made more robust using **stochastic data augmentation** and appropriate **learning objectives**.

Exploiting Structure for Scalable and Robust Deep Learning



- **Long-term planning**
  - Generating Long-term Trajectories Using Deep Hierarchical Networks
  - Generative Multi-Agent Behavioral Cloning
- **Structured Exploration via Hierarchical Policies**
- **Improving the Robustness of Neural Networks**
- **Vignettes**



Generating Long-term Trajectories using Deep Hierarchical Networks

Controlling the complexity of long-term planning

Behavioral Cloning

Goal: Learn $P^{\pi}(a \mid s)$ that imitates human experts

Data:

- Ball + offensive + defensive players
- Long high-resolution sequences ($T \sim 50$ - 200)
- Complex multi-agent dynamics, non-iid states.
- **Sparse in the space of all possible paths.**

Behavior of multiple agents can be analyzed at many levels.

- Realistic intent and movement curvature of **single player.**
- We will consider multi-agent setting later.

$$s_t^i = (x_t^i, y_t^i)$$

$$a_t^i = (v_{x,t}^i, v_{y,t}^i, \text{"pass / shoot / no-op"})$$

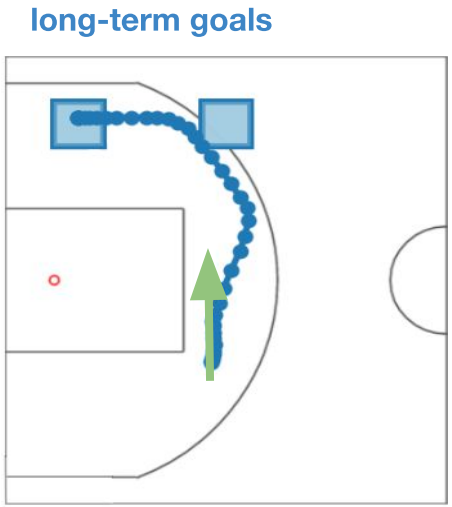
80,000 plays -- https://www.stats.com/data-science/

Generating Long-term Trajectories using Deep Hierarchical Networks, *SZ, Y. Yue, NIPS '16.*

Efficient Behavioral Cloning using Long-term Goals

Idea:

- Decompose behavior into **short-term actions** + **long-term goals**
- Condition policy on ‘future’
- Goals are constraints → dimension reduction
- Ameliorate drift of imperfect model away from the data.

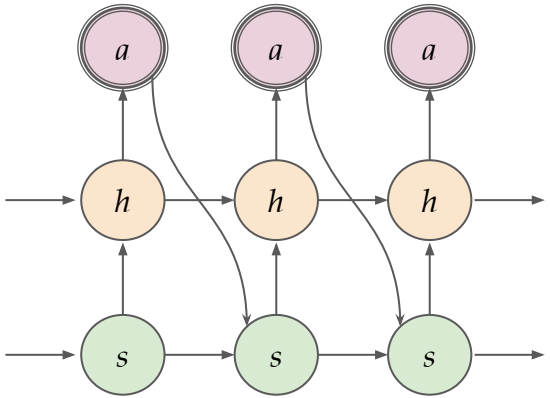


Hidden state policy models

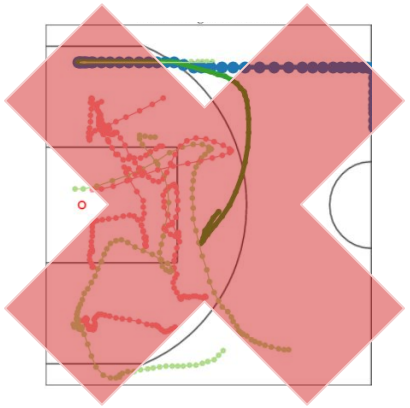
Common: try RNN, LSTM, … with deterministic **state h**

$$a_t \sim P(a_t|h_t)$$
$$h_{t+1} = f(Wh_t + Us_t + b)$$

But these models fail for long sequences!



- Hidden states fail to encode long-term dependencies
- Non-hierarchical policies are **myopic** → fail to generate realistic behavior (“lack intent”).
- **Baseline:** unrealistic straight lines, unrealistic goal

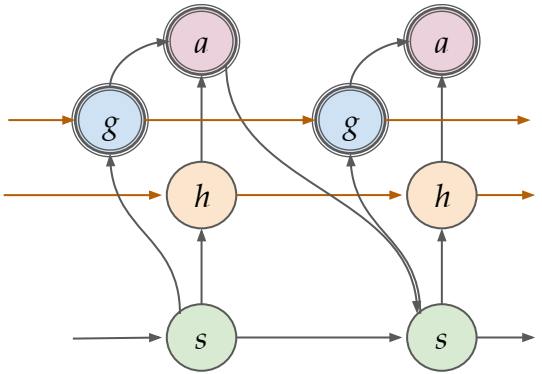


Generating Long-term Trajectories using Deep Hierarchical Networks, *SZ, Y. Yue, NIPS ‘16.*

Hierarchical policy

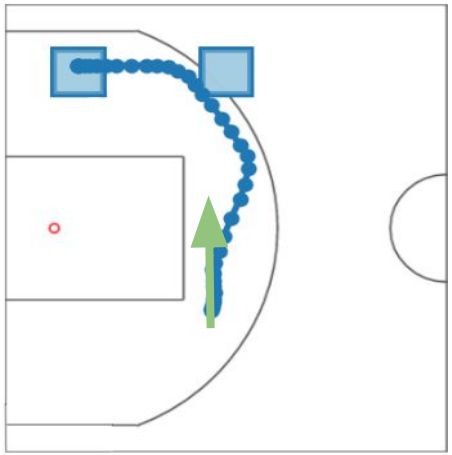
Single-agent hierarchical policy

- Hierarchy of hidden-states: **short-term** vs **long-term** behavior
- Higher levels encode longer temporal dependencies → **predict long-term goals** g



Hierarchical policy

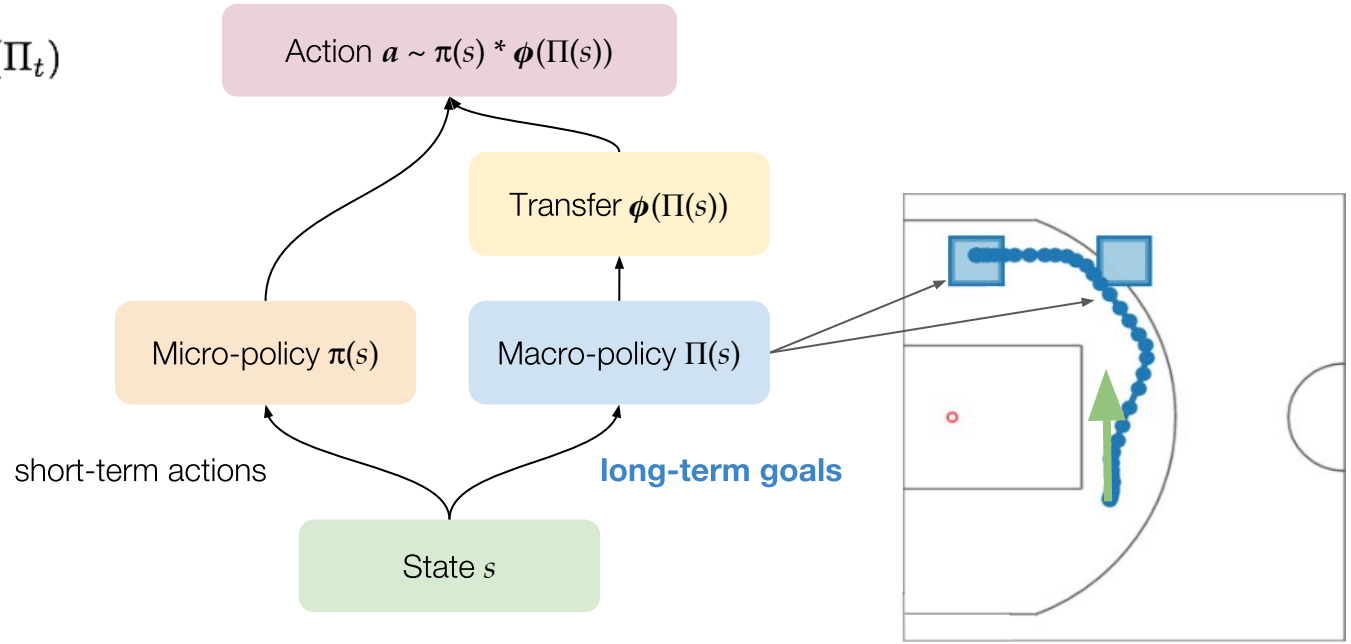
- **Goal ("intent")** = parameterized **region** of state-space
- Use player stationary points as labels to train goal-predictor.
- Capture low-dimensional structure of player behavior!



Generating Long-term Trajectories using Deep Hierarchical Networks, *SZ, Y. Yue, NIPS '16.*

Hierarchical policy

$$\pi(s_t), h_t = \texttt{lstm}(\texttt{conv}(s_t), h_{t-1})$$
$$\Pi(s_t), H_t = \texttt{lstm}(\texttt{conv}(s_t), H_{t-1})$$
$$\phi(\Pi_t) = \texttt{fc}(\Pi(s_t))$$
$$a_t \sim \pi(s_t) \odot \phi(\Pi_t)$$

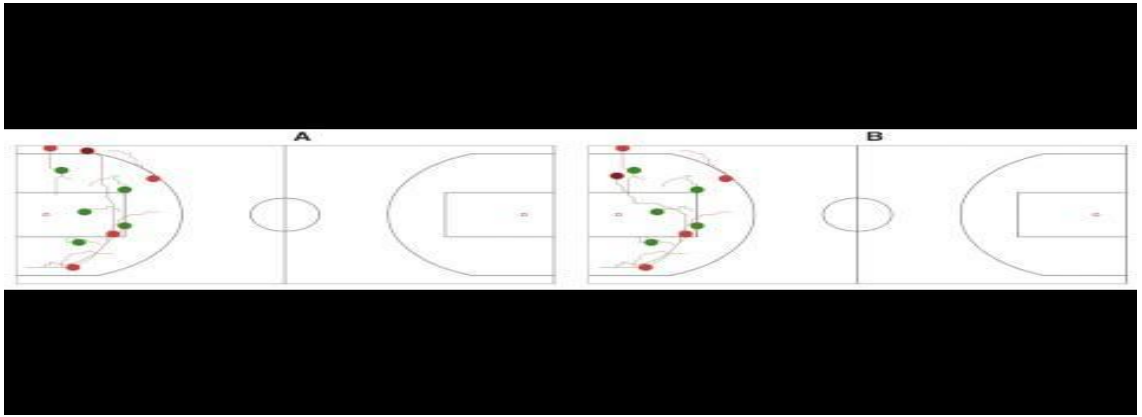


Generating Long-term Trajectories using Deep Hierarchical Networks, *SZ, Y. Yue, NIPS '16.*

User study: fooling sports analysts

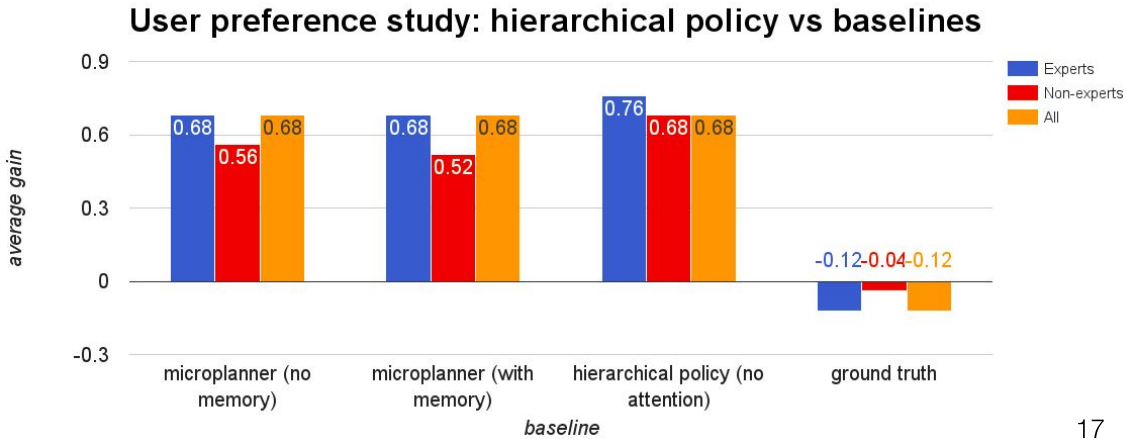
“Which track looks more real?”

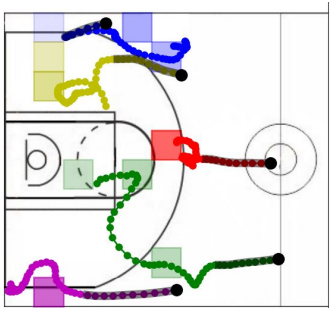
- 7 professional sports analysts
- 8 knowledgeable non-experts
- 25 test cases
- 4 models

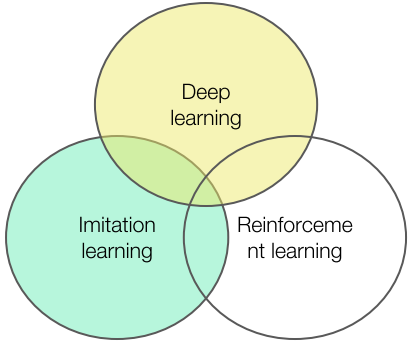


Realism → multi-modality

- Humans cannot distinguish between “left-around” and “right-around”
- Both options are realistic.







Generative multi-agent behavioral cloning

Controlling the complexity of multi-agent coordination + long-term planning

Hierarchical Multi-agent Generative Models

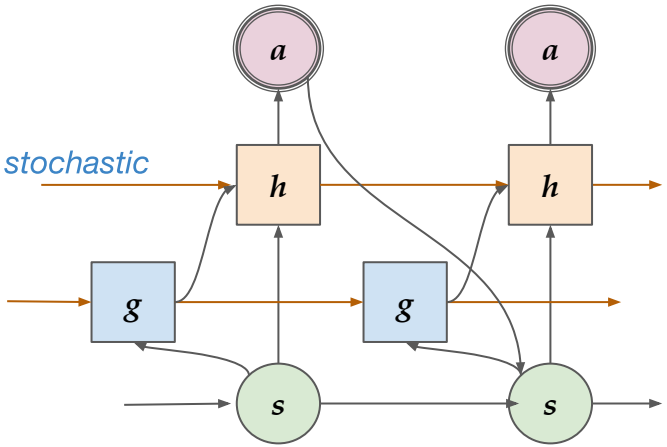
Learning realistic multi-agent policies is challenging.

- Generate trajectories of 5 agents jointly
- Model long-term coordination
- How do we compact represent multi-modality in data?

Hierarchical Multi-agent Generative Models

Learning a hierarchical multi-agent policy.

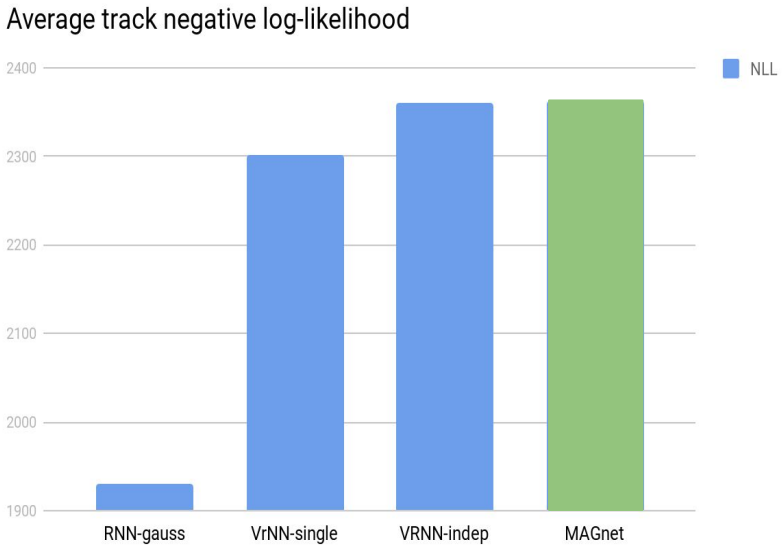
- Perfect communication: each agent sees goals of all agents
- Hierarchical **stochastic** policy → compact multi-modality
- Optimizing MLE is intractable → maximize Evidence Lower Bound (ELBO)

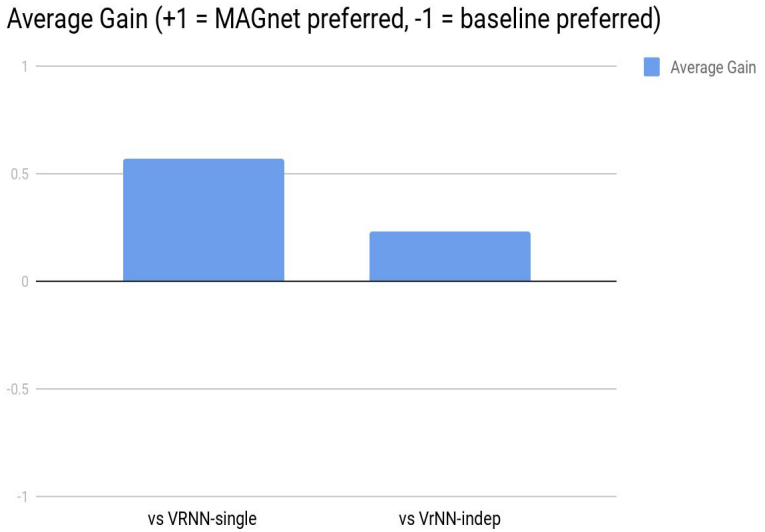


$$\max_{\theta} \log p(\mathbf{x}_{\leq T}) = \max_{\theta} \log \prod_{t=1}^{T} p(\mathbf{x}_t \mid \mathbf{x}_{<t}) \geq \max_{\theta,\phi} \texttt{ELBO}(\theta, \phi)$$

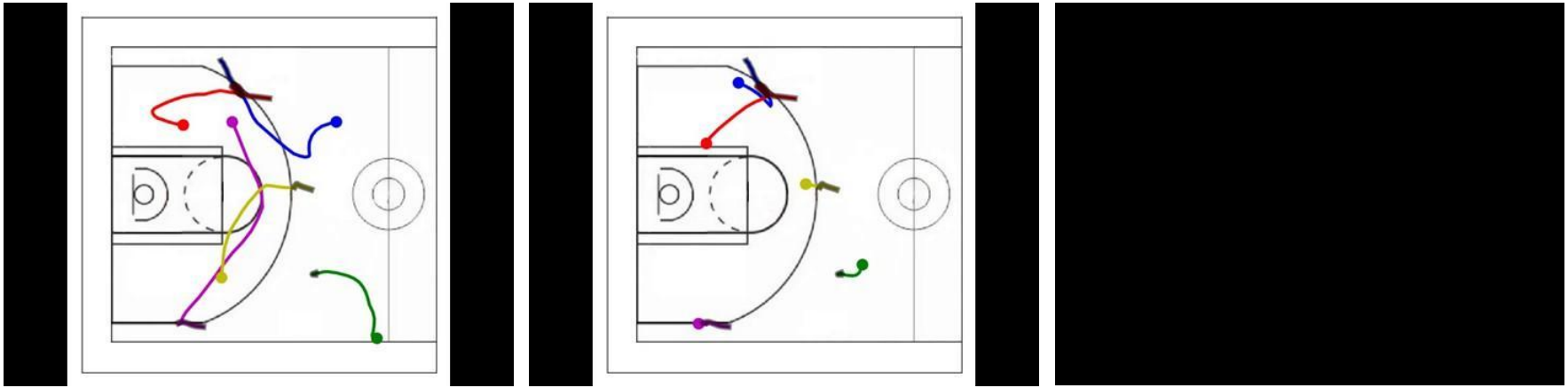
Quantitative Performance

- Generating realistic 5-agent trajectories significantly harder, for same # data
- Our model is significantly preferred over baselines
- Our model is not yet competitive with ground truth in all situations…





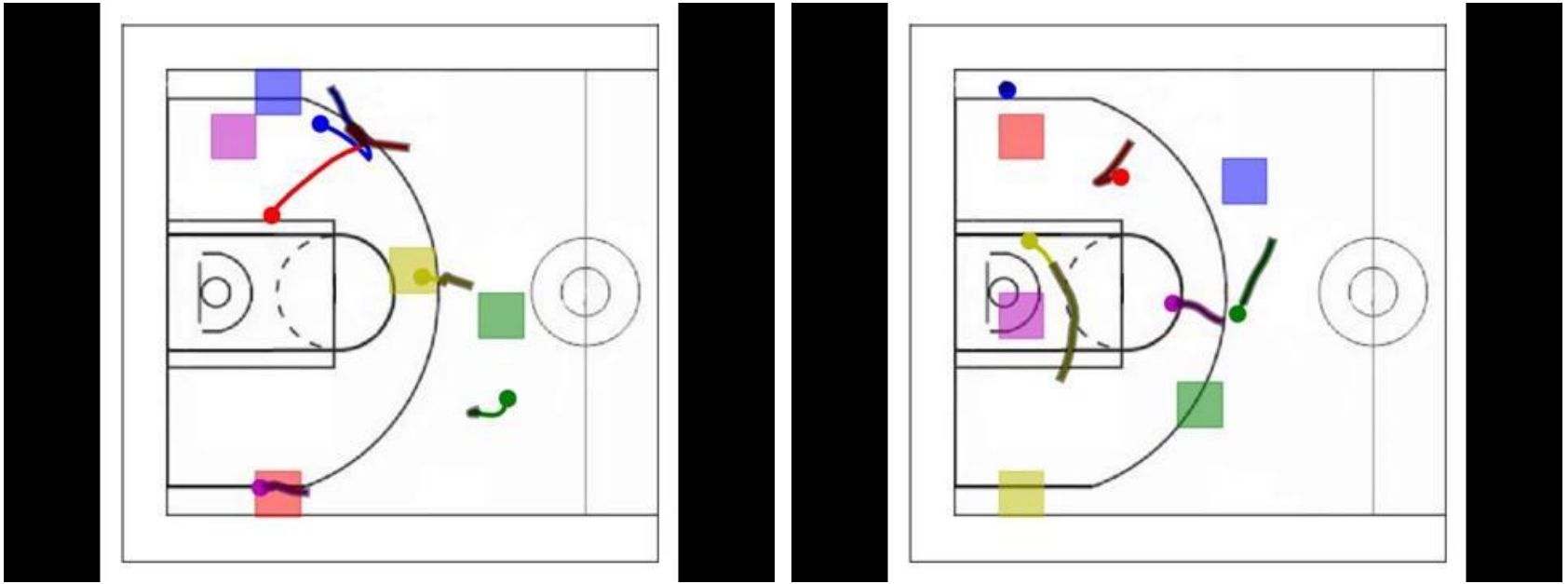
Qualitative Performance



Focus: long-term goals + team formations (offense only).

www.basketball-ai.com

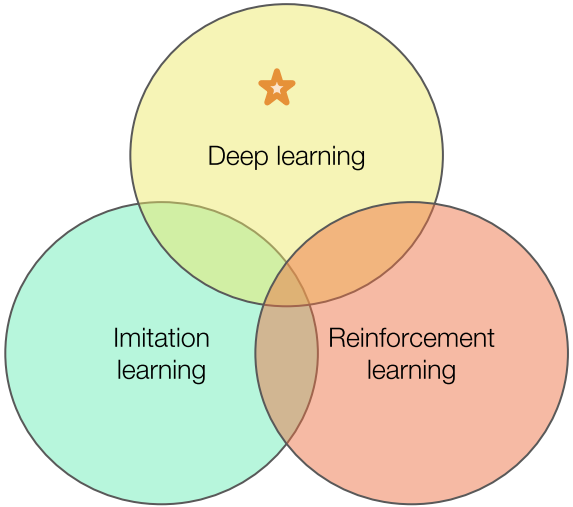
Multi-agent Macro-goals

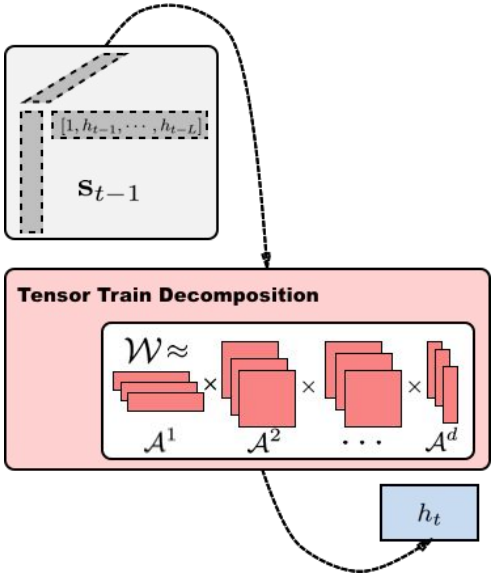


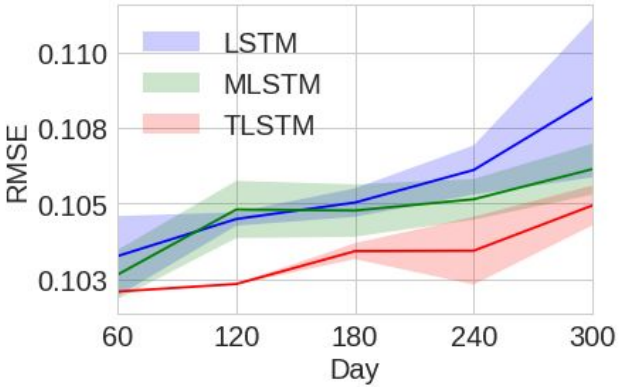
Macro-planners predict goals, agents move towards them.

Goals form realistic team formations.

Long-term Forecasting using Tensor-Train RNNs





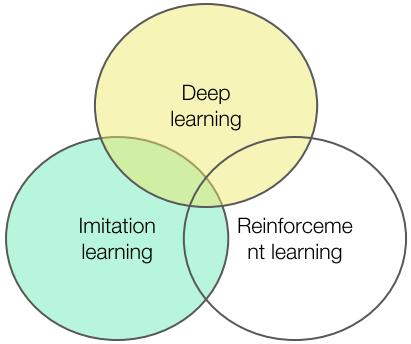


Forecasting model $s_{t+1} = f(s_t, h_t; \theta)$

Using lag-k, order-d polynomial transition functions improves long-term forecasting on real data and enjoys theoretical approximation guarantees.

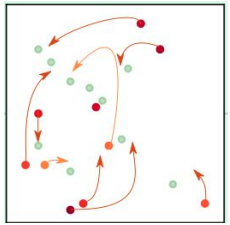
+Rose Yu, Yisong Yue, Anima Anandkumar

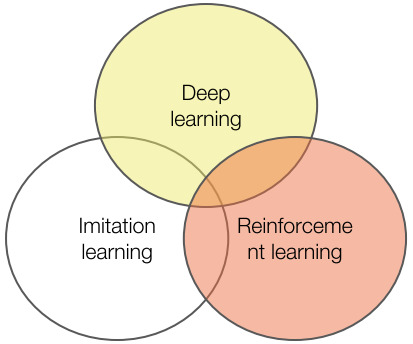
Section Summary



- Hierarchical policies learn ***long-term planning via macro-goals*** and ***coordination***.
- Structure in neural networks greatly improves efficiency and expressiveness

- Learn structured inference models?
- Can we learn hierarchical structure in an **unsupervised** fashion?





Structured Exploration via Hierarchical Policies

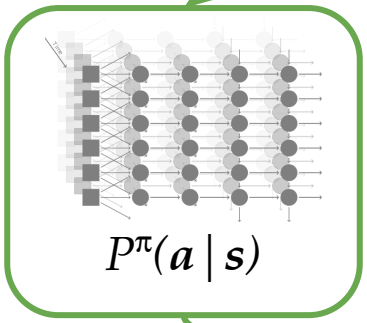
Controlling complexity in large state-action spaces (e.g. coordination games)

Structured Exploration via Hierarchical Variational Policy Networks, ***SZ**, Y. Yue, in submission.*

Reinforcement Learning in Large State-Action Spaces

- Multi-agent RL has exponentially large state-action spaces
- Policy needs to gather training data → exploration of state-action space
- But sampling from high-dimensional spaces is inefficient!

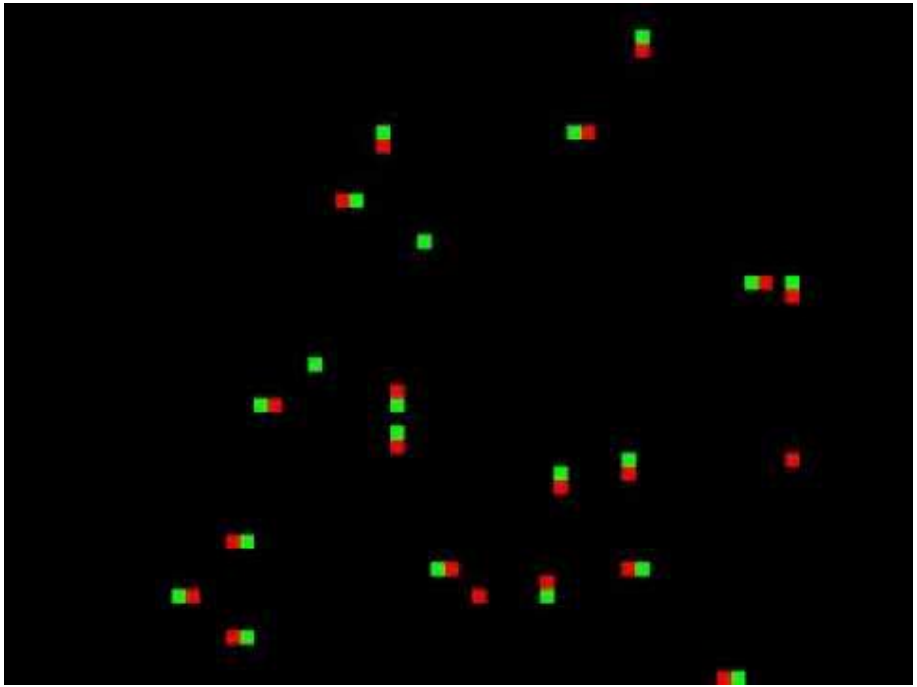
Observation s, reward r

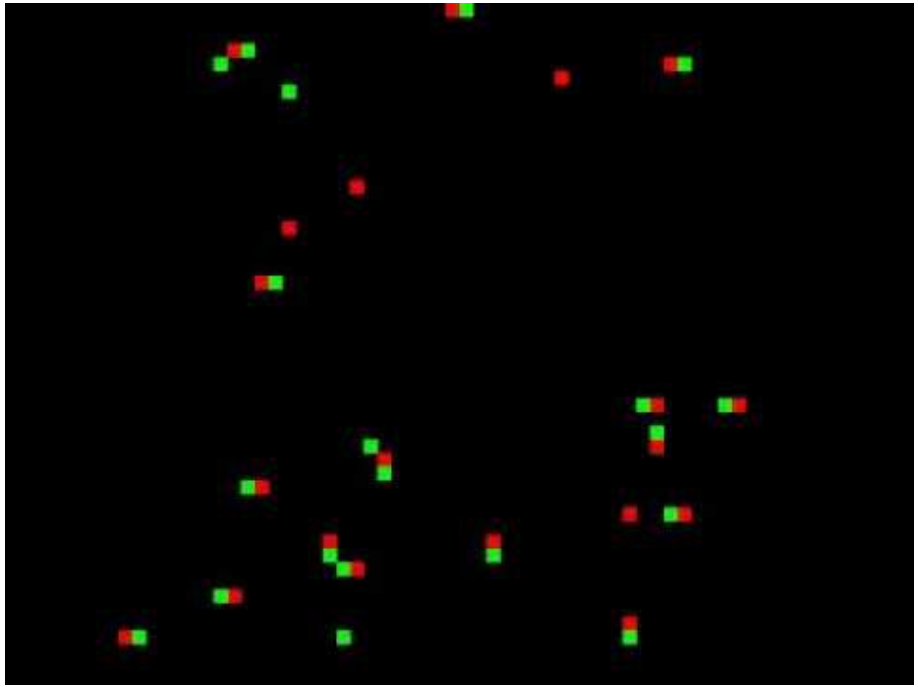


Action a

$$i = 1 \dots N \text{ agents}$$
$$\mathbf{s}_t = (s_t^1, \dots, s_t^N), \quad s^i = (x_t^i, y_t^i)$$
$$\mathbf{a}_t = (a_t^1, \dots, a_t^N), \quad a^i = (v_{x,t}^i, v_{y,t}^i)$$
$$\pi : \mathbf{s}_t \mapsto \mathbf{a}_t$$

Multi-agent Complexity





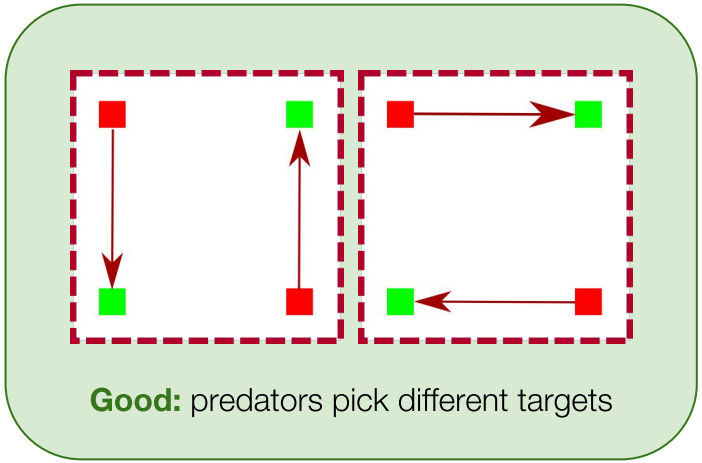
Structured exploration

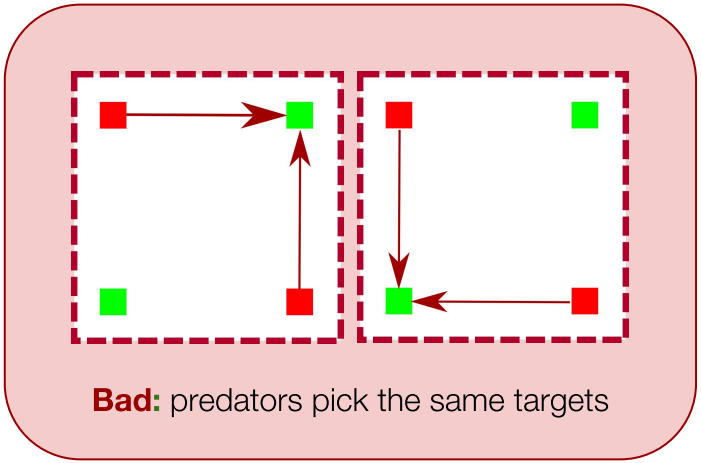
Idea: often joint action / policy space A has **structure** induced by reward

- **High-level analogy**: an optimal matrix policy could be low-rank
- Efficient exploration → bias exploration towards low-dimensional subspace

Example: learn a multi-agent controller in predator-prey game

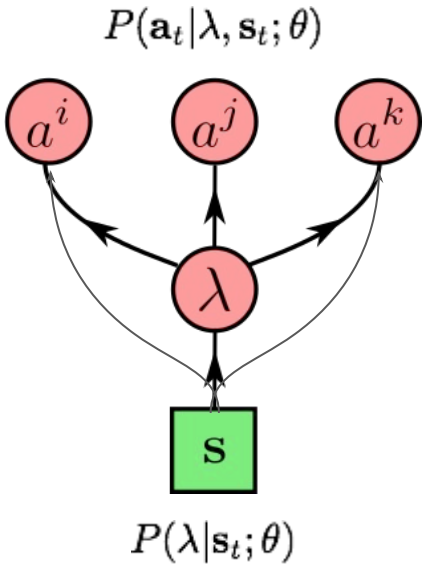
- Policies that **coordinate well** and **that do not**.
- Good coordination = predators pick different targets





Multi-Agent Coordinated Exploration

Idea: combine **deep graphical models** + **reinforcement learning**



- Learn structure of joint action space.
- Stochastic latent variable λ that encodes coordination types
- Multi-agent policy with "correlation device" factorizes as:

$$P(\mathbf{a}_t|\mathbf{s}_t) = \int d\lambda_t \prod_{i=1}^{N} P(a_t^i|\lambda_t, \mathbf{s}_t) P(\lambda_t|\mathbf{s}_t)$$

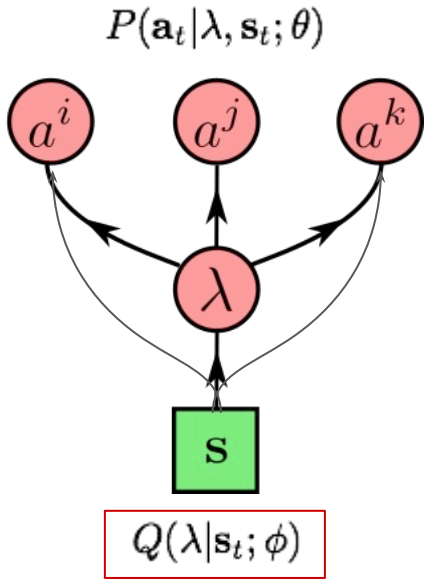
- Explore by sampling λ and $a \rightarrow$ **correlated joint actions**

Problem: integrating over λ is intractable $\rightarrow$ learning $P(a \mid \lambda, s)$ is hard

Variational RL

Solution: RL as probabilistic inference

- Maximizing reward = maximizing log-likelihood.
- Maximize Evidence Lower Bound (ELBO).
- Choose $Q(\lambda \mid s; \phi)$ as factorized Gaussian, with **learned** parameters.



$$P(\mathcal{O} = 1|\tau) \propto \exp R(\tau)$$

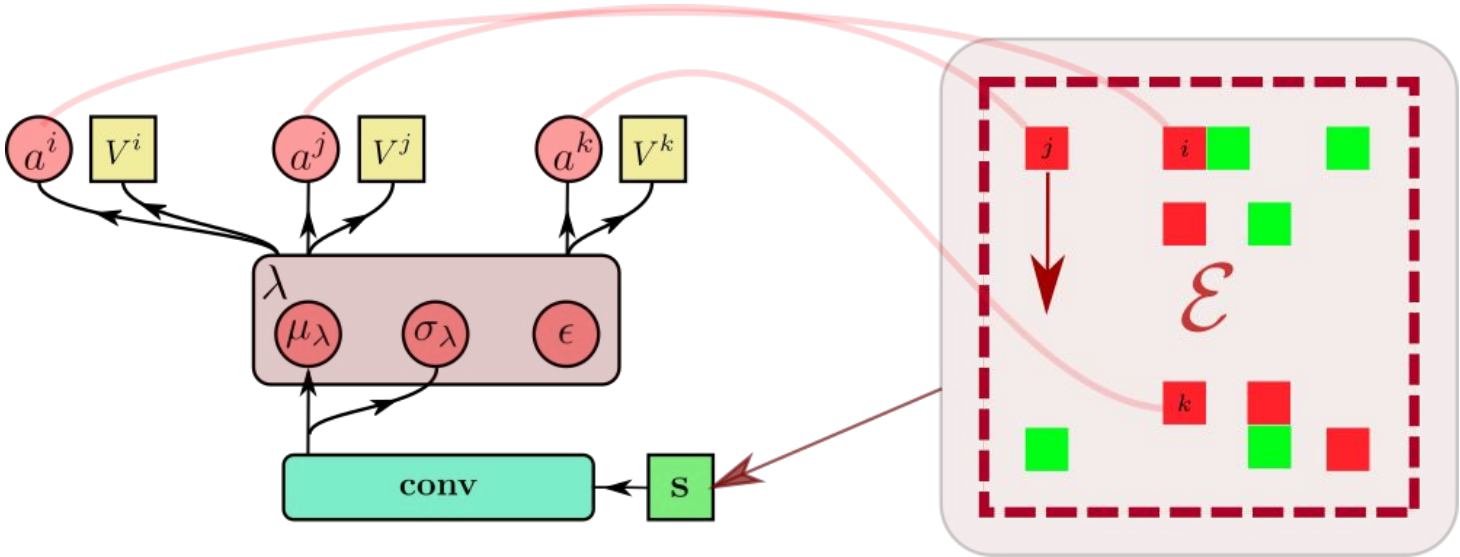
$$\max_\theta \mathbb{E}_\pi[R(\tau)] = \max_\theta P(\mathcal{O}|\theta)$$

$$\max_\theta \log P(\mathcal{O}|\theta) \geq \max_\theta \underbrace{\int d\tau d\lambda Q(\tau, \lambda|\phi) \log \frac{\exp R(\tau) \cdot P(\tau, \lambda|\theta)}{Q(\tau, \lambda|\phi)}}_{\texttt{ELBO}(\theta, \phi)}$$

Multi-Agent Coordinated Exploration

- End-to-end actor-critic: learn **policy** π and **value** function V.
- Model computes $\boldsymbol{\mu}, \boldsymbol{\sigma}$ of $Q(\hat{\lambda} \mid s; \phi)$.

$$\nabla_\theta \texttt{ELBO}(\theta, \phi) = \mathbb{E}_{s,a,\lambda}\left[\sum_t \nabla_\theta \log P(\mathbf{a}_t|\mathbf{s}_t, \lambda_t; \theta) \cdot \exp R(\tau) \,\middle|\, \lambda \sim Q(\lambda|\mathbf{s}_t; \phi)\right]$$



Experimental Results: Hare-Hunters

Multi-agent Predator-Prey in 30 x 30 gridworld

- Hare-hunters: 1 predator can capture 1 prey.
- Very sparse reward.

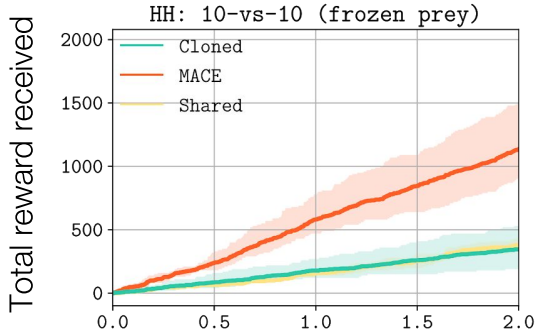
$$R^i = \begin{cases} 1, & \text{if } \textit{all} \text{ prey are inactive before the time limit } T \\ 0, & \text{otherwise} \end{cases}$$

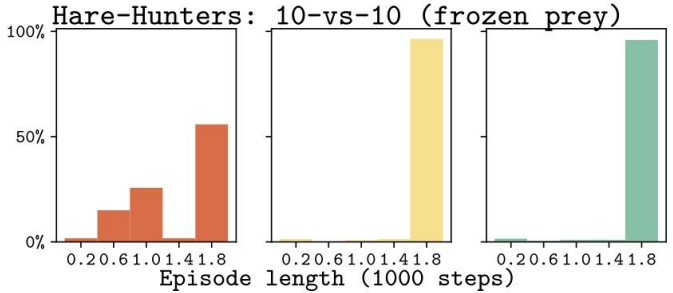
Baselines:

- *Shared*: deterministic latent variable
- *Cloned:* agents use identical policies

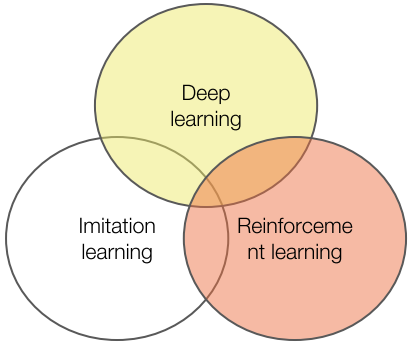
In games with **up to 20 agents**, hierarchical policy

- solves game faster over time
- accumulates rewards significantly (2x) faster



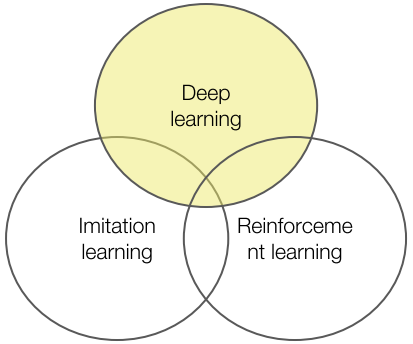


Section Summary



- Hierarchical policies can explore more efficiently by restricting to low-dimensional subspace
- Low-dimensional subspace is semantic → “good coordination”.

- Temporally extended graphical model policies?
- Learn the right structure automatically?



Improving Neural Network Robustness via Stability Training

SZ, Y. Song, T. Leung, I. Goodfellow, CVPR '16.

Improving Robustness via Stability Training

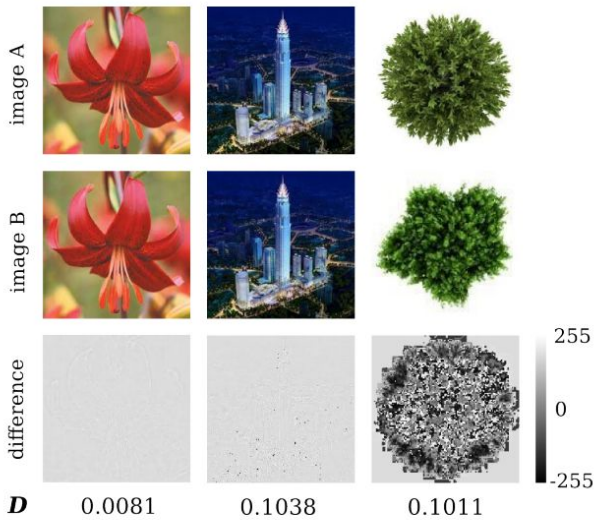
Robustness in deep learning is a fundamental issue.

$$\forall x' : d(x, x') < \delta \Leftrightarrow D(f(x), f(x')) < \epsilon.$$

Vulnerability in computer vision:

Perturbations in visual input can distort neural networks

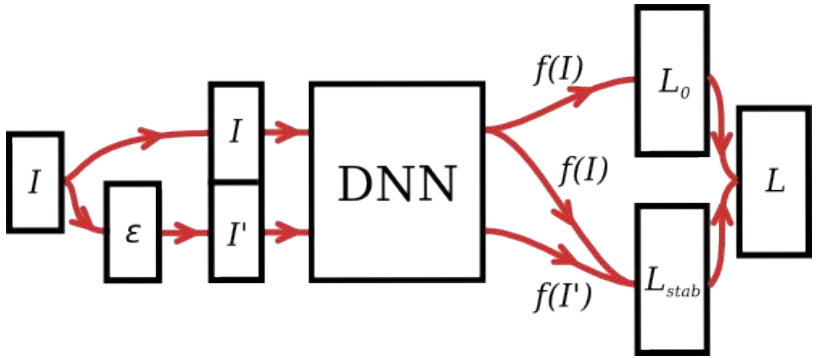
- Natural: compression, cropping, re-scaling
- Crafted (imperceptible) adversarial attacks



State-of-the-art image network thinks dissimilar pair is more similar than almost identical pair

Improving Robustness via Stability Training

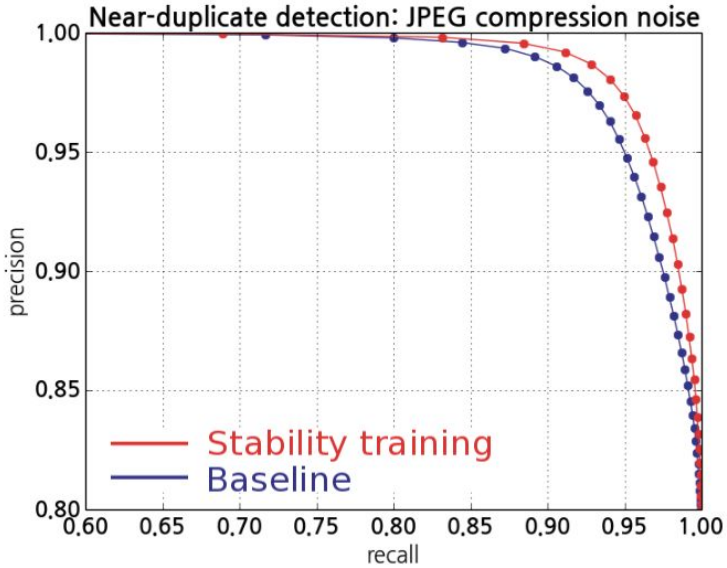
- Stochastic data augmentation
- Force network to behave similarly on perturbed input, even if network is wrong
- Effective although f is non-convex + simple to implement → deployed in Google Image Search

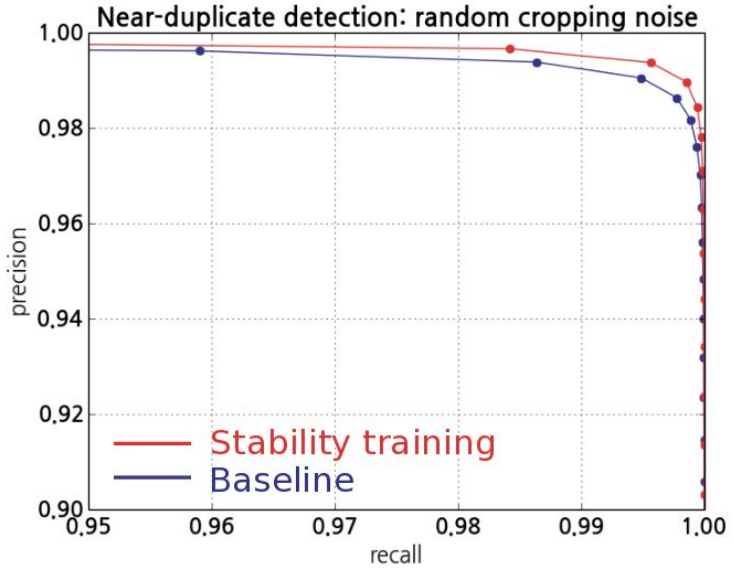


$$L(x, x'; \theta) = L_0(x; \theta) + \alpha L_{stab}(x, x'; \theta)$$
$$L_{stab}(x, x'; \theta) = D(f(x), f(x'))$$

Improving Robustness via Stability Training

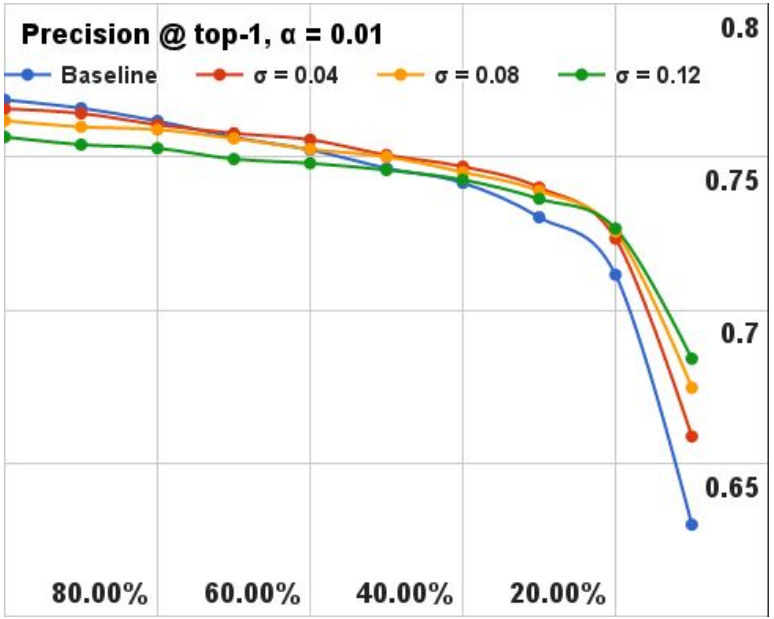
Stability training improves **near-duplicate detection** precision-recall performance by 2-3% on corrupted images

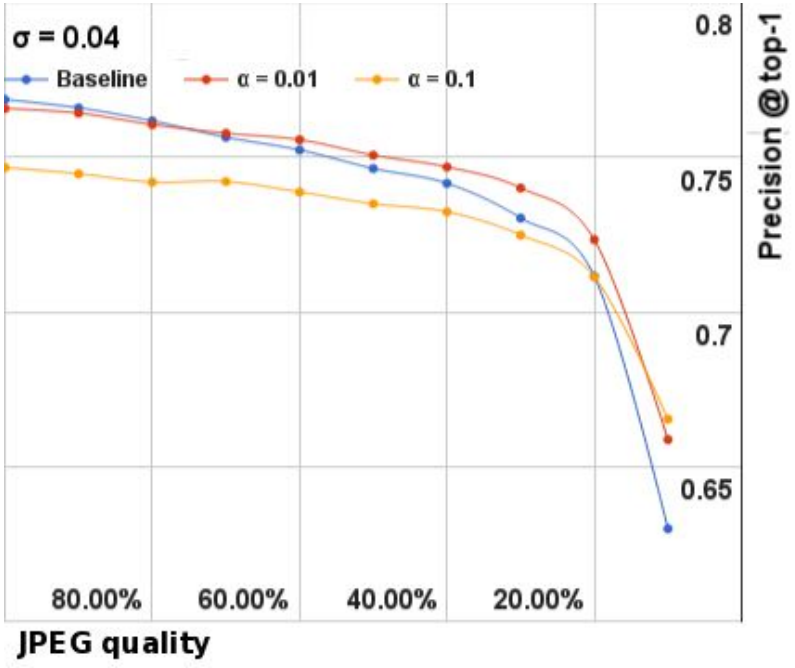




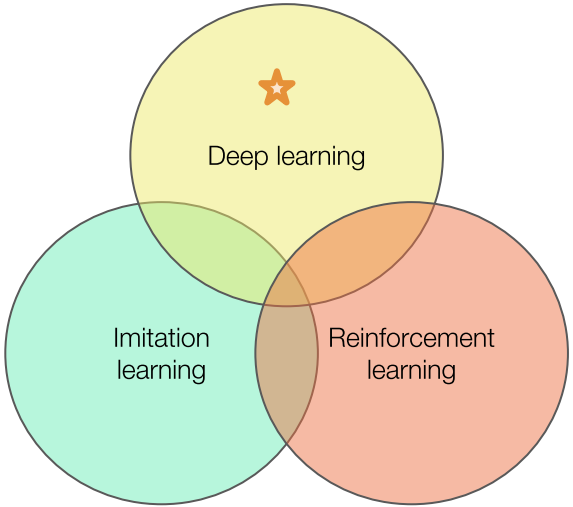
Improving Robustness via Stability Training

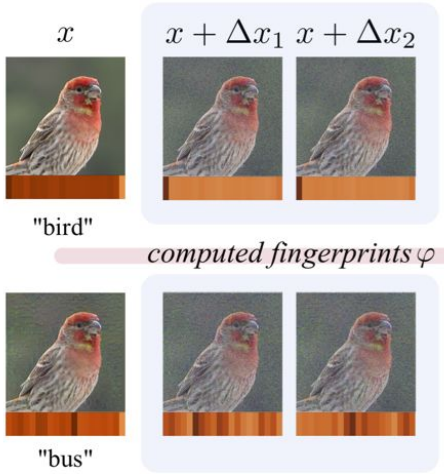
Stability training improves **classification** performance by 2-3% on corrupted images

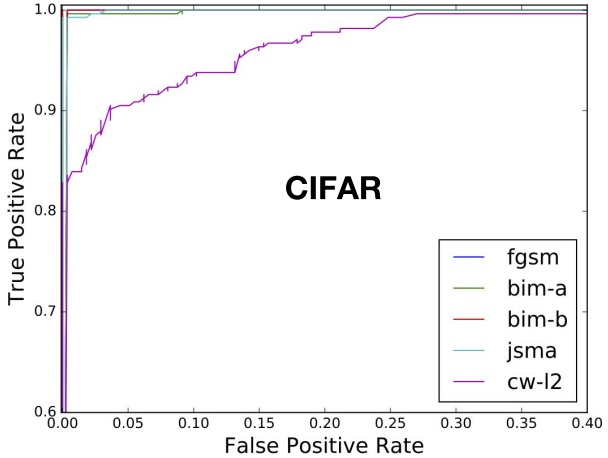




Detecting Adversarials using Neural Fingerprinting





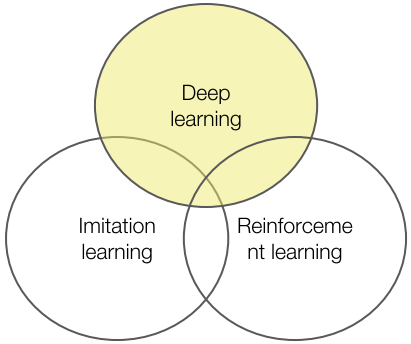


Neural fingerprints detect adversarial examples at ~99% AUC-ROC on MNIST, CIFAR, Miniimagenet-20.

Enjoys theoretical guarantees for linear models.

+*Sumanth Dathathri, Richard Murray*

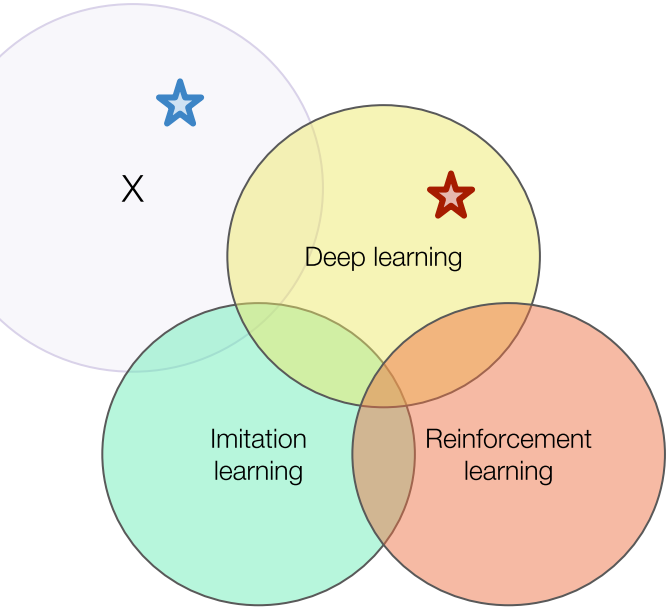
Section Summary

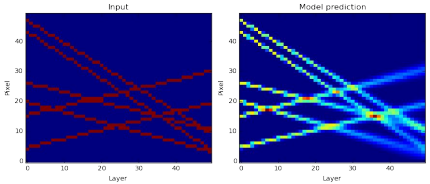


- Stochastic data augmentation and stability objective → more robust neural networks
- Adversarial examples can be efficiently detected using Neural Fingerprints

- Can we guarantee robustness?
- Can we characterize geometry of neural networks?

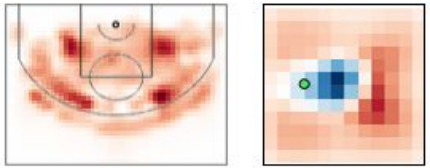
Vignettes





HEP.TrkX: Graph-CNNs for High-energy Particle Tracking

+Steven Farrell, Maria Spiropoulou et al. *(ACAT '17)*



Multi-resolution Tensor Learning for Large-scale Spatial Data

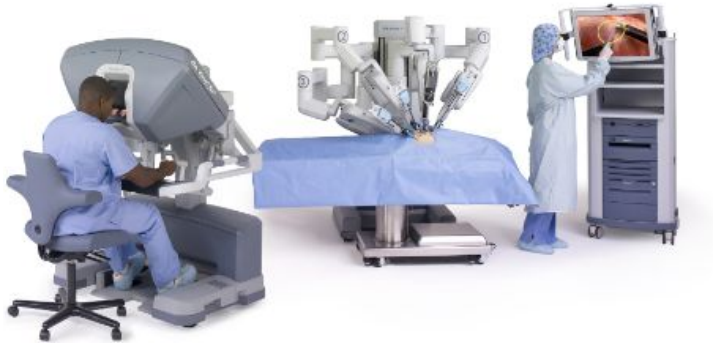
+Rose Yu, Yisong Yue *(in submission)*

Towards efficient, robust and safe AI

Structured machine learning for high-stakes multi-agent applications.

- Coordination, communication, safety
- Safe exploration
- Model-based learning
- Learning safe reward functions (mechanism design)
- Adversarial policies: learning Nash, correlated equilibria

Learning informative communication in adversarial games



Robotic surgery; multi-doctor patient treatment

Acknowledgements

Yisong Yue
www.yisongyue.com

Caltech

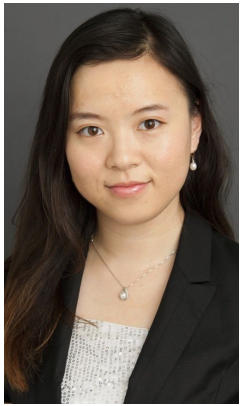
Eric Zhan

Patrick Lucey
www.patricklucey.com

STATS

Anima Anandkumar

Caltech



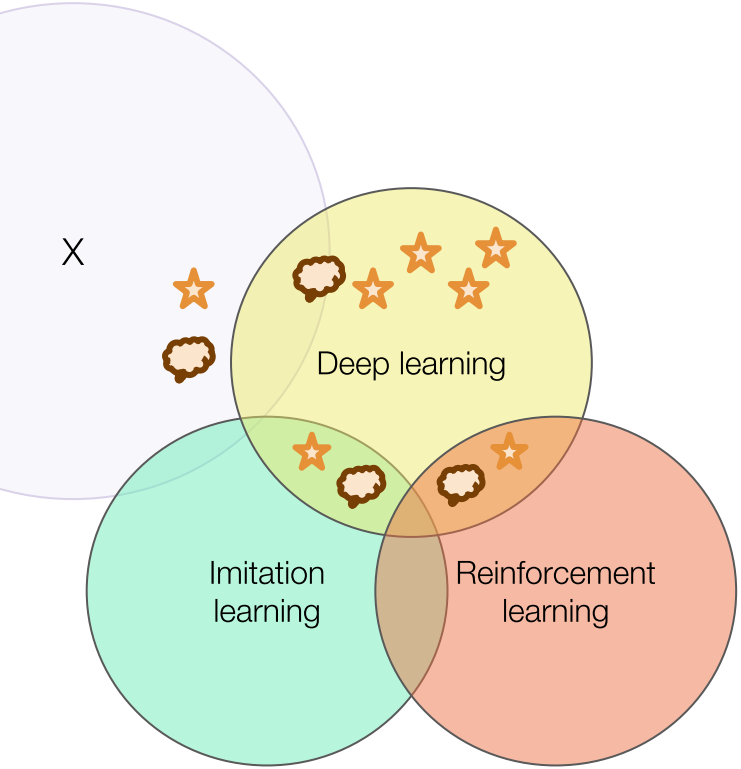
Rose Yu
www.roseyu.com

Yang Song

Ian Goodfellow

Google

Thank you! Questions?



Improving Neural Network Robustness via Stability Training, *CVPR '15*

Generating Long-term Trajectories Using Deep Hierarchical Networks, *NIPS '16*

HEP.TrkX: high-energy particle tracking using deep learning, *CtD '17, NIPS workshop '17, ACAT '17*

Multi-resolution Tensor Learning for Large-scale Spatial Data, *in submission*

Long-term Forecasting using Tensor-Train RNNs, *in submission*

Structured Exploration via Hierarchical Variational Policy Networks, *in submission*

Generative Multi-agent Behavioral Cloning, *in submission*

Detecting Adversarial Examples via Neural Fingerprinting, *in submission*

MAGnet: Generating Long-Term Multi-Agent Trajectories, *NIPS workshop '17*

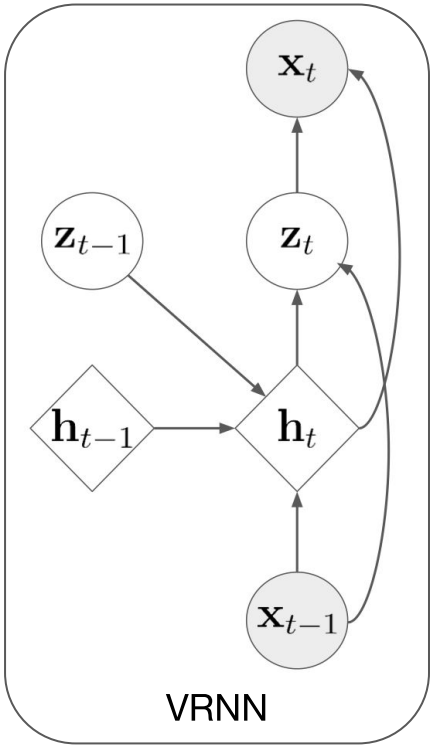
Measuring the robustness of neural networks via minimal adversarial examples, *NIPS workshop '17*

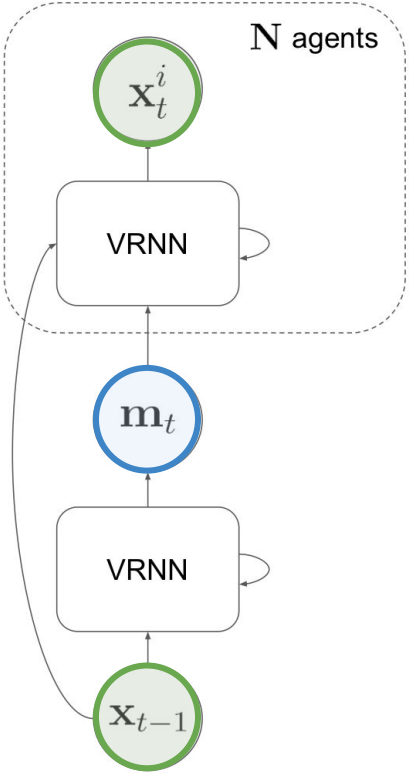
Multi-Agent Counterfactual Regret Minimization for Partial-Information Collaborative Games, *NIPS workshop '17*

Extra slides

Model details

- Conditional distributions instantiated with Variational Recurrent Neural Networks [Chung ‘16]
- Conditions only on history
- Approximate inference needs care

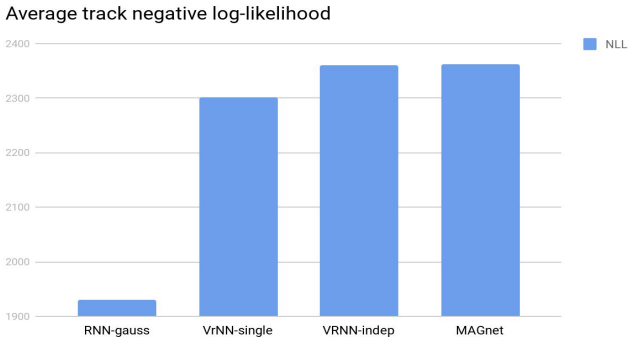


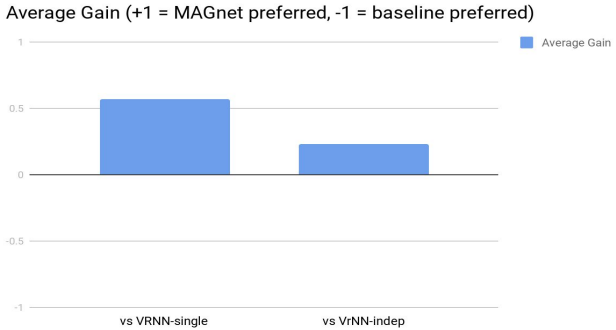


$$E_{q(\boldsymbol{z}_{\leq T}|\boldsymbol{x}_{\leq T})}\left[\sum_{t=1}^{T} -\mathrm{KL}\big(q(\boldsymbol{z}_t \mid \boldsymbol{x}_{\leq t}, \boldsymbol{z}_{<t}) \| p(\boldsymbol{z}_t \mid \boldsymbol{x}_{<t}, \boldsymbol{z}_{<t})\big) + \log p(\boldsymbol{x}_t \mid \boldsymbol{z}_{\leq t}, \boldsymbol{x}_{<t})\right]$$

Quantitative Performance

- Generating realistic 5-agent trajectories significantly harder, for same # data
- Our model is significantly preferred over baselines
- Our model is not yet competitive with ground truth in all situations…





Model	Log-Likelihood	# Parameters
RNN-gauss	1931	7,620,820
VRNN-single	$\geq$ 2302	8,523,140
VRNN-indep	$\geq$ 2360	4,367,340
Ours	$\geq$ **2362**	4,372,190

Table 1. We report the average log-likelihood per sequence in the test set as well as the number of trainable parameters for each model. "$\geq$" indicates a lower bound on the log-likelihood.

Model Comparison	Win/Tie/Loss	Avg Gain
vs. VRNN-single	25/0/0	0.57
vs. VRNN-indep	15/4/6	0.23

Table 2. Preference study results. We asked 14 professional sports analysts to judge the relative quality of the generated rollouts. Judges are shown 50 comparisons that animate one rollout from our model and another from a baseline. Win/Tie/Loss indicates how often our model is preferred over baselines. Gain scores are computed by scoring +1 when our model is preferred and -1 otherwise. The average gain is computed over the total number of comparisons (25 per baseline) and judges. Our results are 98% significant using a one-sample t-test.

Variational RL

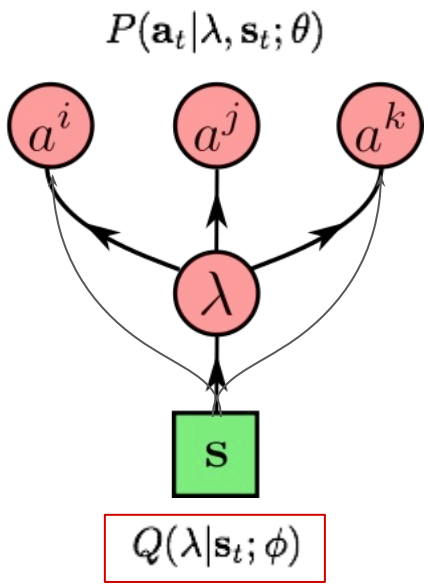
$$P(\mathcal{O}=1|\tau) \propto \exp R(\tau)$$

$$\max_\theta \mathbb{E}_\pi[R(\tau)] = \max_\theta P(\mathcal{O}|\theta) = \max_\theta \int d\tau \exp R(\tau) \cdot P(\tau|\theta)$$

$$\max_\theta \log P(\mathcal{O}|\theta) = \max_\theta \log \int d\tau d\lambda \exp R(\tau) \cdot P(\tau, \lambda|\theta)$$

$$\geq \max_\theta \underbrace{\int d\tau d\lambda Q(\tau,\lambda|\phi) \log \frac{\exp R(\tau) \cdot P(\tau,\lambda|\theta)}{Q(\tau,\lambda|\phi)}}_{\texttt{ELBO}(\theta,\phi)}$$

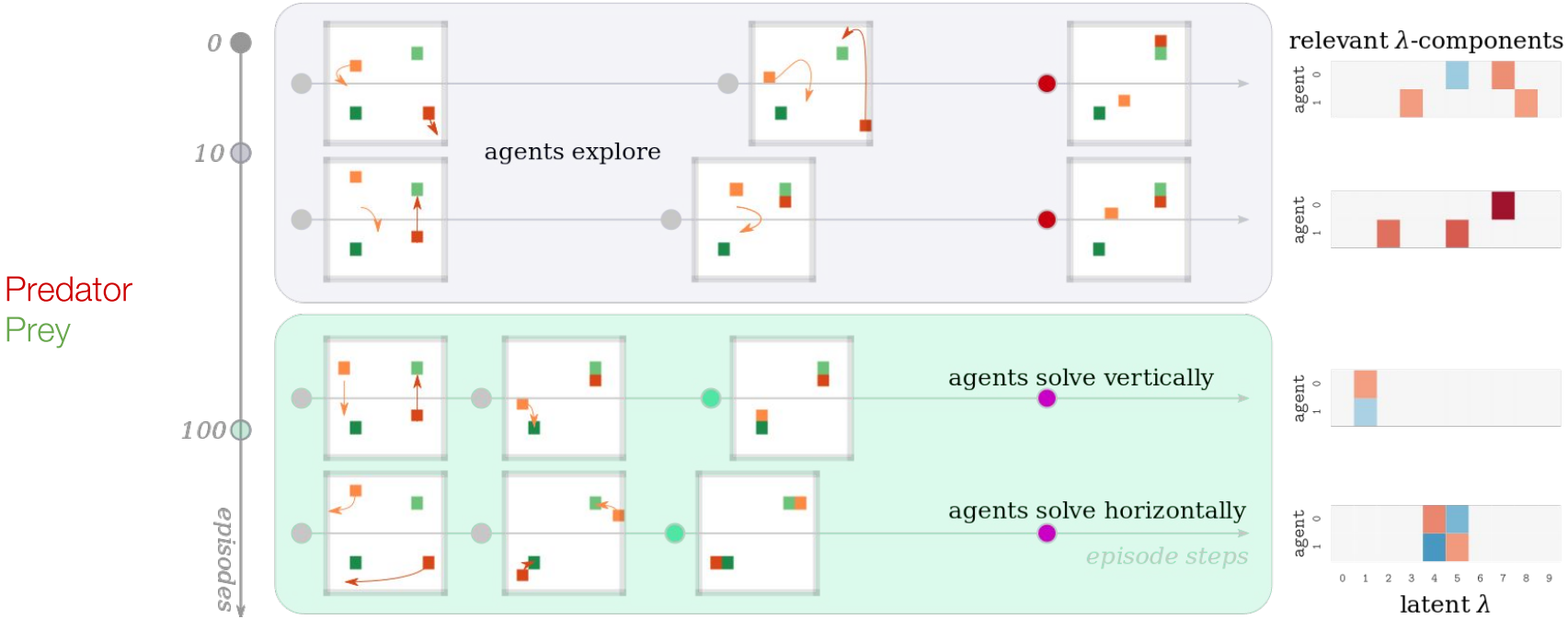
$$\nabla_\theta \texttt{ELBO}(\theta,\phi) = \mathbb{E}_{s,a,\lambda}\left[\sum_t \nabla_\theta \log P(\mathbf{a}_t|\mathbf{s}_t,\lambda_t;\theta) \cdot R(\tau) \middle| \lambda \sim Q(\lambda|\mathbf{s}_t;\phi)\right]$$

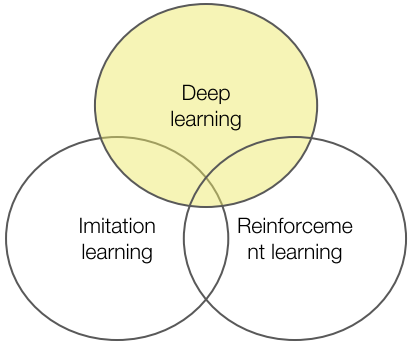


Samples: 2v2

Predators hunting prey with time limit

- Policy displays 2 types of solutions.
- Hierarchical policy learns faster how to solve game.
- Latent factors are meaningful: they correlate with qualitative different behavior (2-sided t-test α = 10%).



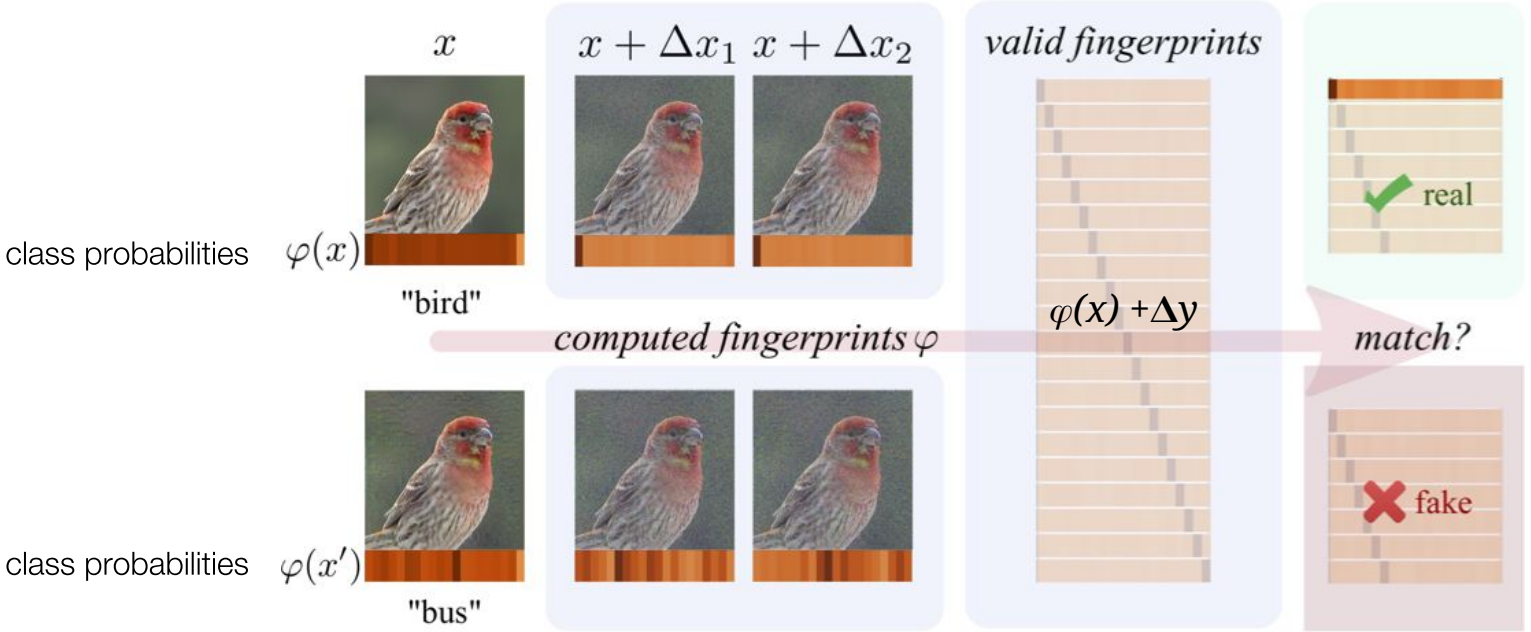


Detecting Adversarial Examples via Neural Fingerprinting

Neural Fingerprinting

+Sumanth Dathathri, Richard Murray, Yisong Yue

- Framework for detection and prediction
- Use **(secret) fingerprints** ***(Δx, Δy)*** to detect adversarial examples x'
- Train network to behave according to fingerprints on real examples
- At test-time: check if model behaves as expected on new inputs



Neural Fingerprinting

- Theoretical guarantees for linear classifiers
- Open question how to generalize to nonlinear decision boundaries

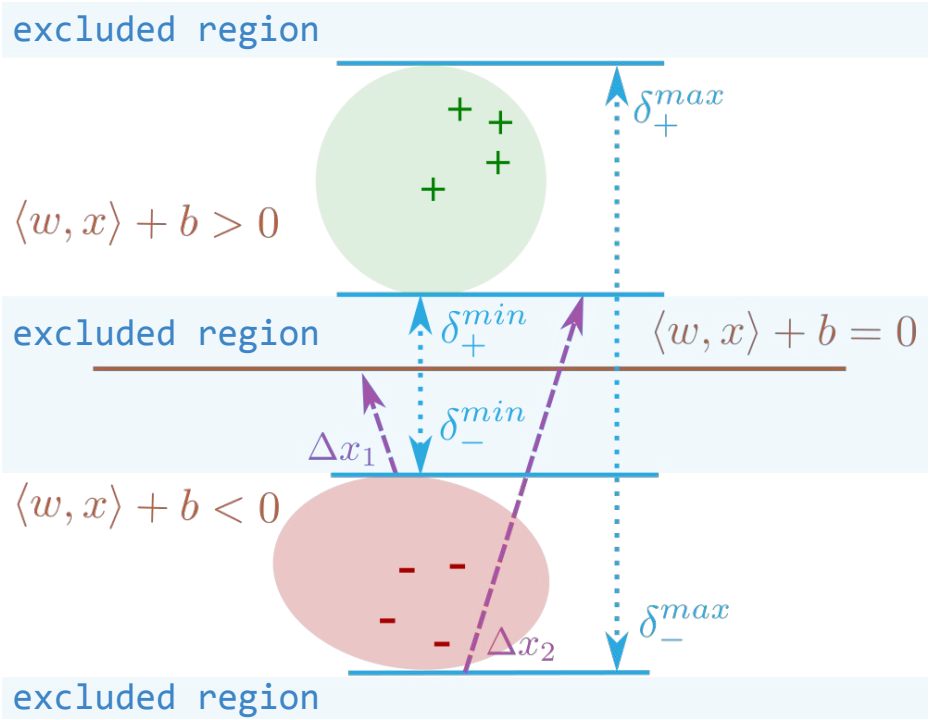
Algorithm 1 *NeuralFP*

Input: example x, comparison function D (see Eqn 9).
Input: threshold $\tau > 0$.
Input: (secret) $\left\{\left(\Delta x^i, \Delta y^{i,j}\right)\right\}_{i=1\ldots N, j=1\ldots K}$.
Output: `accept` / `reject`.
if $\exists j : D(x, f, \xi^{i,j}) \leq \tau$ **then**
    Return: `accept` # *x is real*
else
    Return: `reject` # *x is fake*
end if

$$D(x, f, \xi^{\cdot,j}) = \frac{1}{N}\sum_{i=1}^{N} \|F(x, \Delta x^i) - \Delta y^{i,j}\|_2$$

$$F(x, \Delta x^i) = \varphi(x + \Delta x^i) - \varphi(x),$$

$$\varphi(x) = \frac{h(x)}{\|h(x)\|},$$

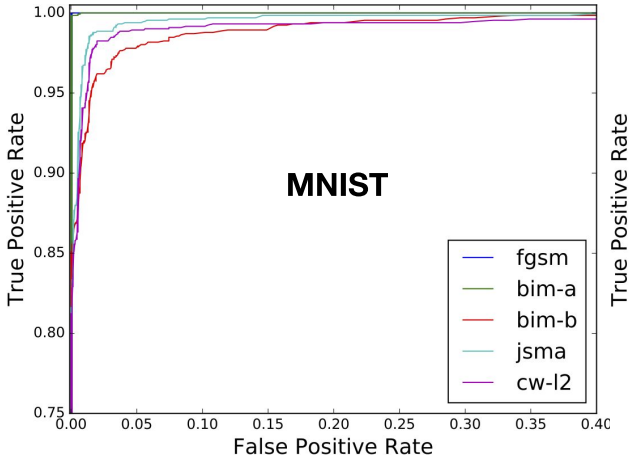


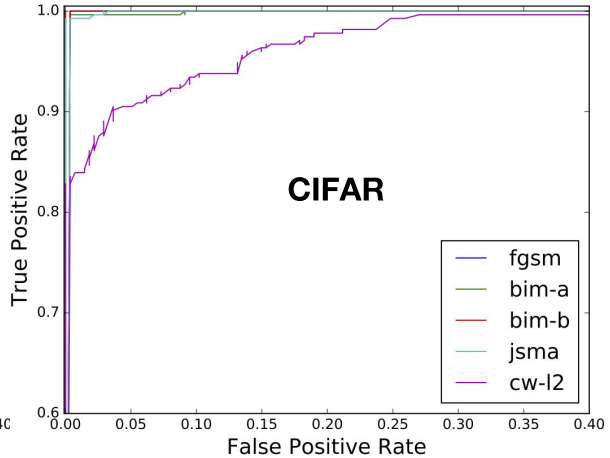
Neural Fingerprinting

Achieve near-perfect AUC-ROC against state-of-the-art attacks, for MNIST, CIFAR, MiniImagenet-20

Detecting Adversarial Examples via Neural Fingerprinting

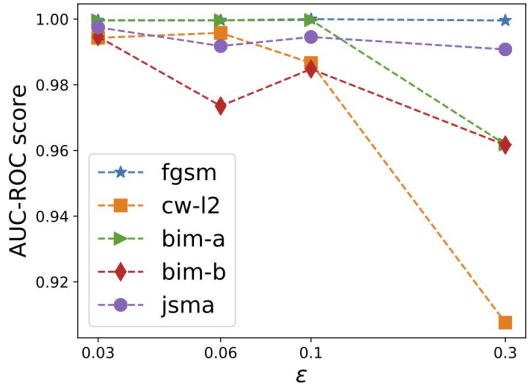
Data	Method	Defense type	FGM	JSMA	BIM-a	BIM-b	CW-L_2
MNIST	LID	Blackbox	99.68	96.36	99.05	99.72	98.66
	LID	Whitebox	99.68	98.67	99.61	99.90	99.55
	NeuralFP	Blackbox	**100.0**	**99.97**	**99.94**	**99.98**	**99.74**
CIFAR-10	LID	Blackbox	82.38	89.93	82.51	91.61	93.32
	LID	Whitebox	82.38	95.87	82.30	99.78	**98.94**
	NeuralFP	Blackbox	**99.96**	**99.91**	**99.91**	**99.95**	98.87
MiniImagenet-20	*NeuralFP*	Blackbox	99.96	-	-	99.68	-

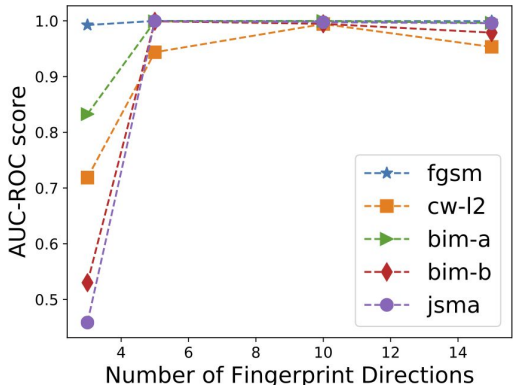




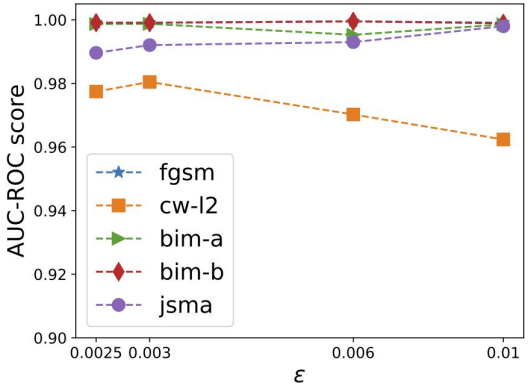
Neural Fingerprinting

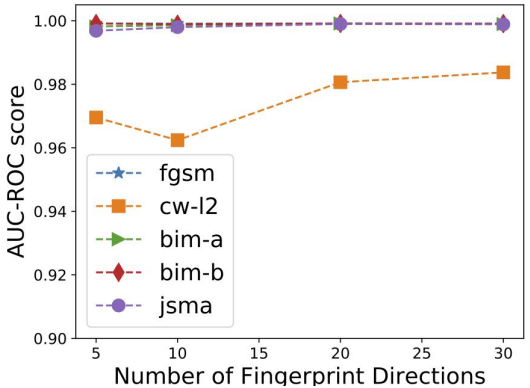
Performance is robust across hyperparameters



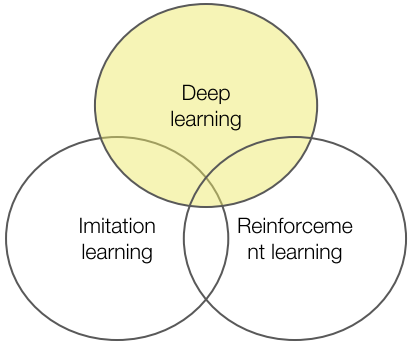


MNIST



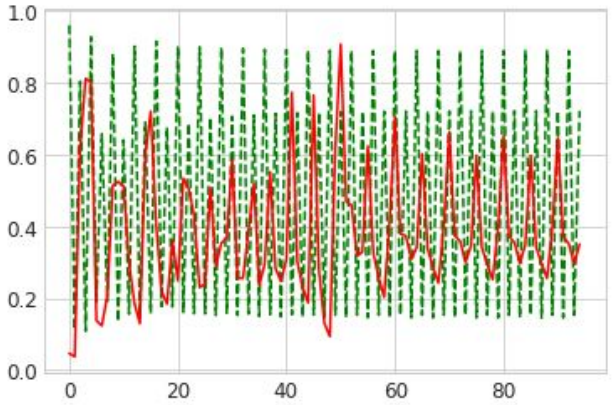


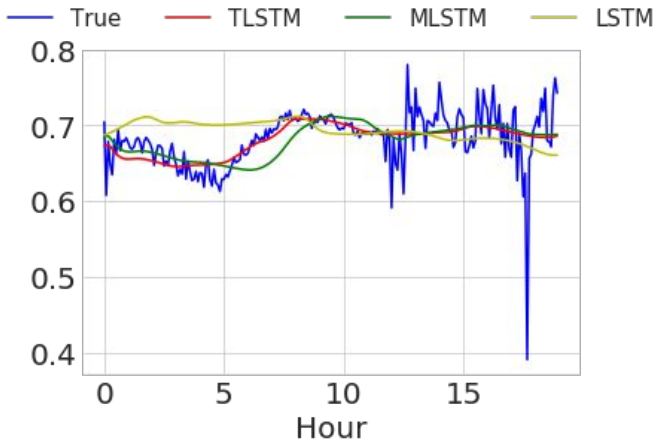
CIFAR



Long-term Forecasting using Tensor-train RNNs

Long-term Forecasting using Tensor-Train RNNs





$$x_{t+1} = \left(c^{-2} + (x_t + w)^2\right)^{-1}, \quad c, w \in [0, 1],$$

- Auto-regressive sequence prediction → error propagation makes long-term forecasting hard

Long-term Forecasting using Tensor-Train RNNs

Use higher-order structure in RNN, LSTM

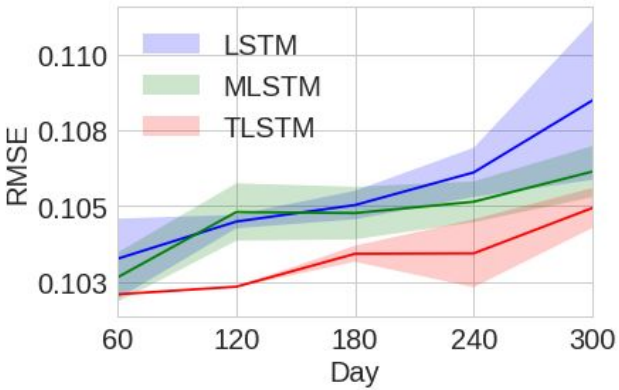
- k previous hidden states
- order-d polynomial interactions

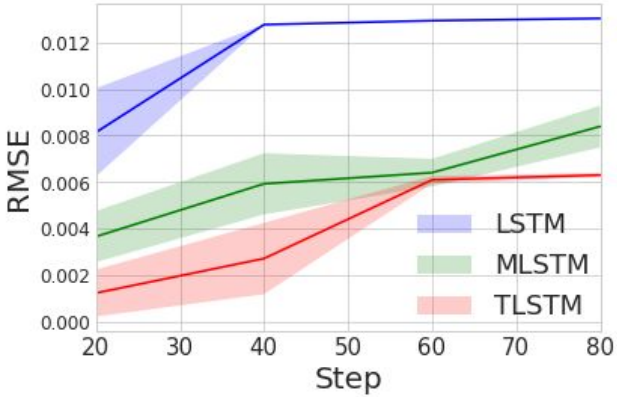
$$\mathbf{h}_{t;\alpha} = f(W_{\alpha}^{hx}\mathbf{x}_t + \sum_{i_1,\cdots,i_p} \mathcal{W}_{\alpha i_1 \cdots i_P} \underbrace{\mathbf{s}_{t-1;i_1} \otimes \cdots \otimes \mathbf{s}_{t-1;i_p}}_{P})$$

- tensor-train decomposition of weight tensor

$$\mathcal{W}_{i_1 \cdots i_P} = \sum_{\alpha_1 \cdots \alpha_{P-1}} \mathcal{A}^1_{\alpha_0 i_1 \alpha_1} \mathcal{A}^2_{\alpha_1 i_2 \alpha_2} \cdots \mathcal{A}^P_{\alpha_{P-1} i_P \alpha_P}$$

Long-term Forecasting using Tensor-Train RNNs





- TT-RNN gives state-of-the-art performance on long-term forecasting

Long-term Forecasting using Tensor-Train RNNs

Let the state transition function $f \in \mathcal{H}_{\mu}^{k}$ be a Hölder continuous function defined on a input domain $\mathbf{I} = I_1 \times \cdots \times I_d$, with bounded derivatives up to order k and finite Fourier magnitude distribution C_f. Then a single layer Tensor Train RNN can approximate f with an estimation error of ϵ using with h hidden units:

$$h \leq \frac{C_f^2}{\epsilon}(d-1)\frac{(r+1)^{-(k-1)}}{(k-1)} + \frac{C_f^2}{\epsilon}C(k)p^{-k}$$