Deep Reinforcement Learning and the Type IIA Landscape

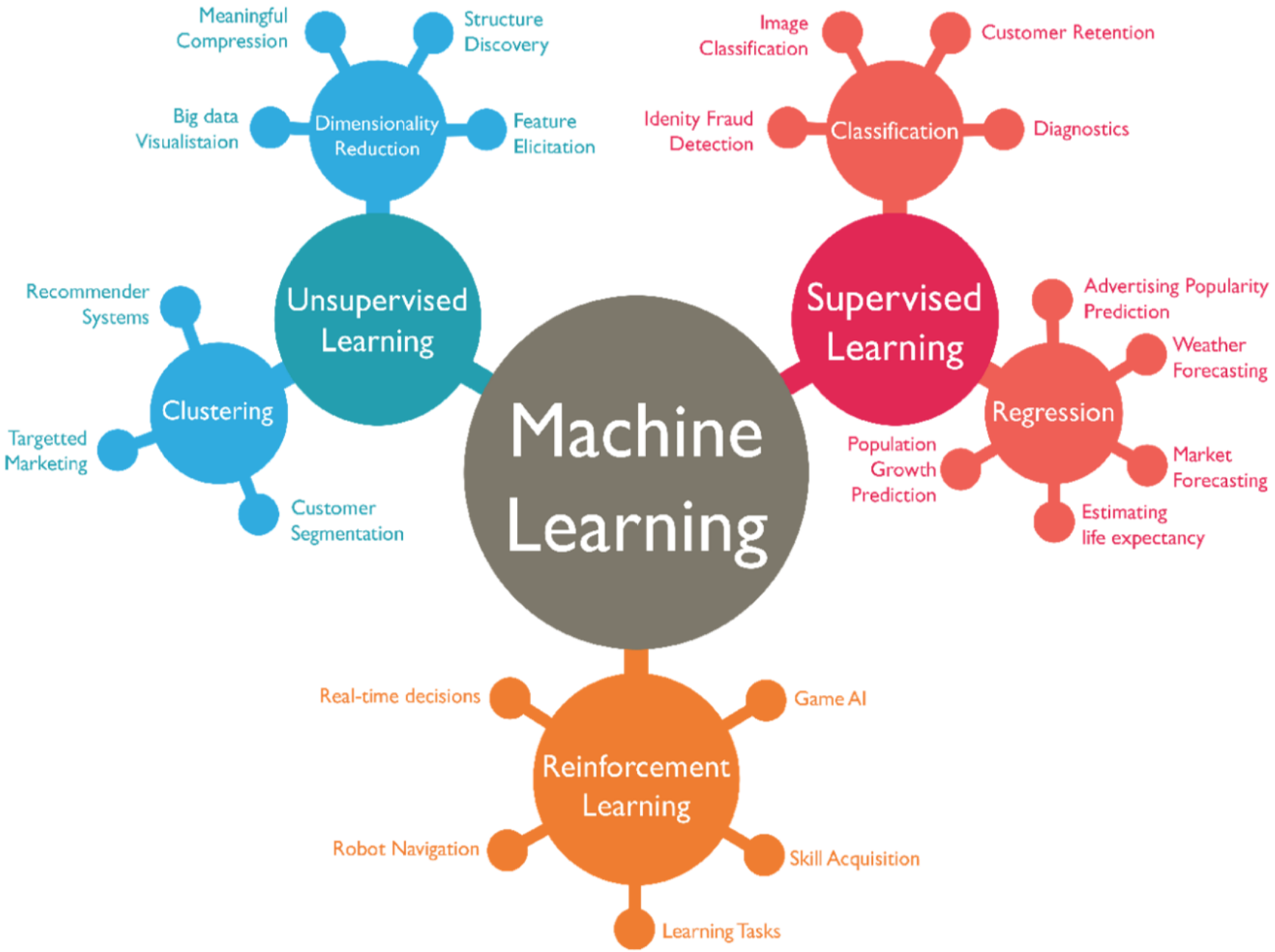
Brent D. Nelson

String Phenomenology 2019, CERN

hep-th/1903.11616, with J. Halverson and F. Ruehle
JHEP 1906 (2019) 3

Northeastern University

The Machine Learning Landscape



from amazonaws.com

Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih **Koray Kavukcuoglu** **David Silver** **Alex Graves** **Ioannis Antonoglou**

Daan Wierstra **Martin Riedmiller**

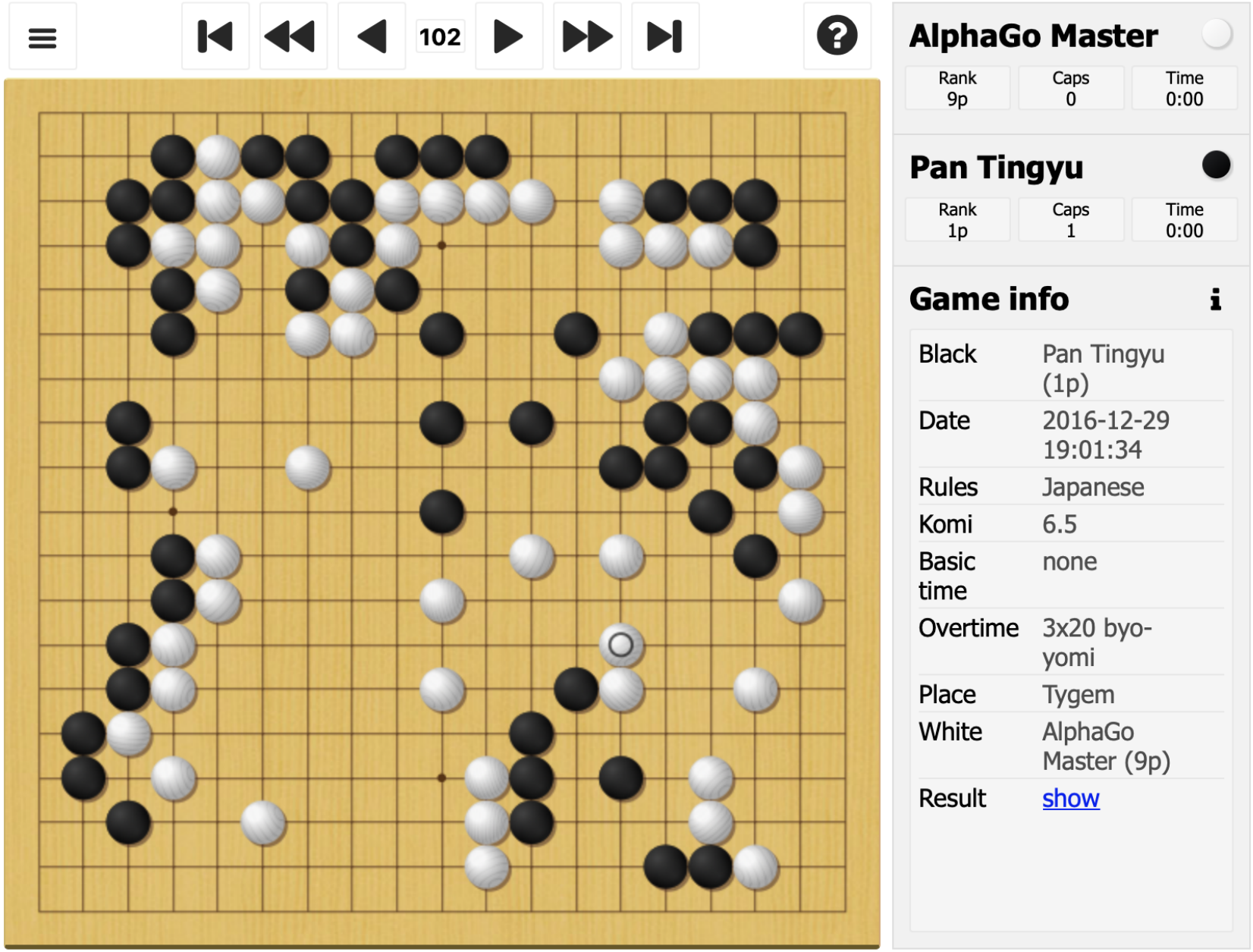
DeepMind Technologies

Breakout and Space Invaders, 2 of the 49 Atari games used in the paper

Reinforcement Learning: Born to Play!

Reinforcement Learning: Born to Play!

Choose Game : Pan Tingyu (2016.12.29)

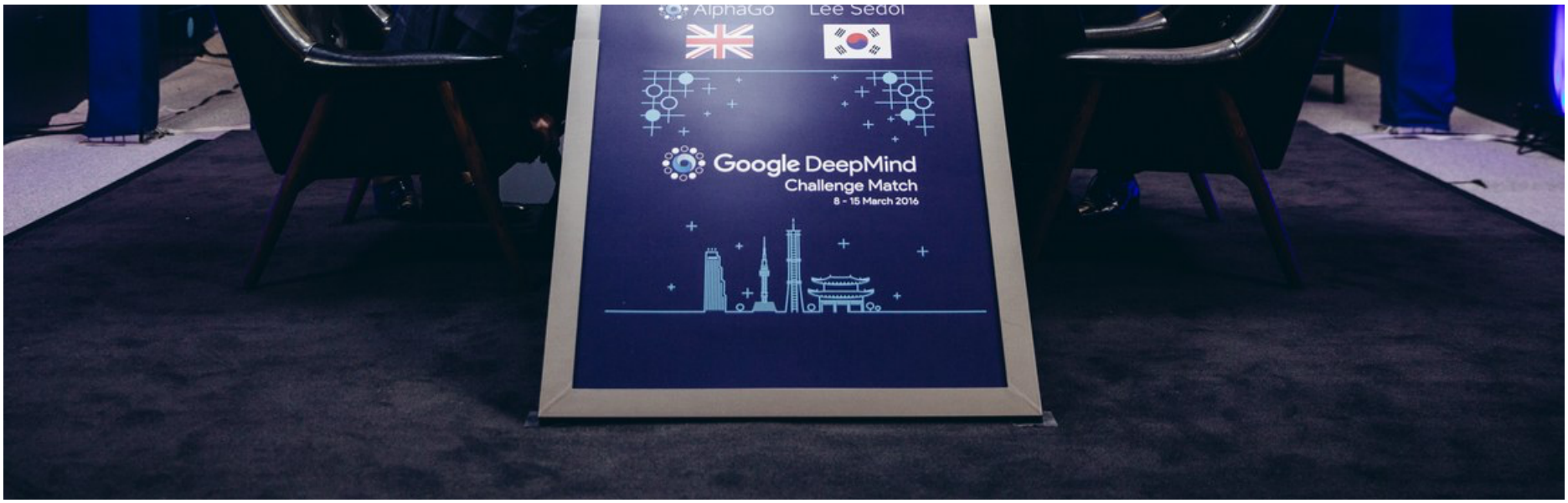


AlphaGo vs. Lee Sedol

AlphaGo vs. Lee Sedol

CADE METZ BUSINESS 03.14.16 02:40 PM

WHY THE FINAL GAME BETWEEN ALPHAGO AND LEE SEDOL IS SUCH A BIG DEAL FOR HUMANITY



AlphaGo vs. Lee Sedol

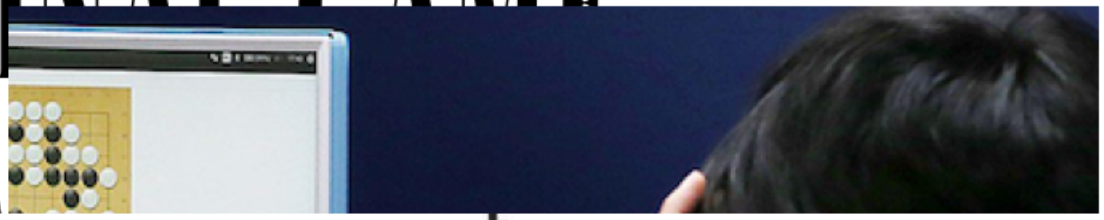
CADE METZ BUSINESS 03.14.16 02:40 PM

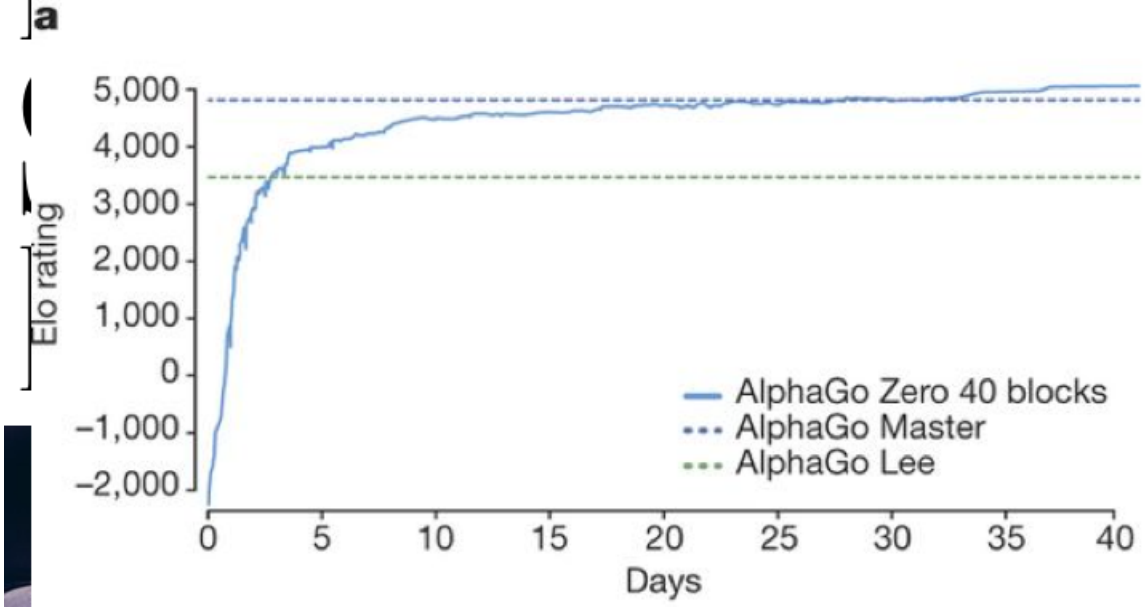
WHY THE FINAL GAME BETWEEN A SEDOL IS SU R HUMANITY

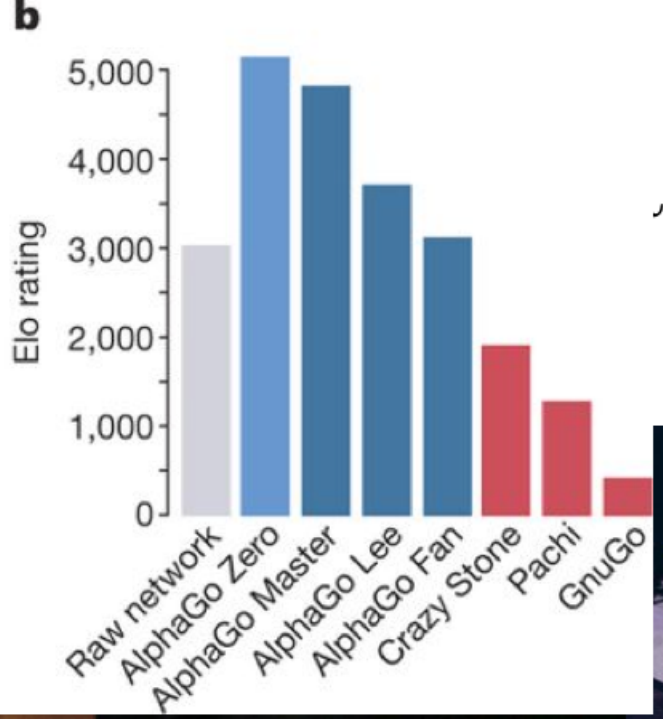
AlphaGo vs. ..... AlphaGo ?!?

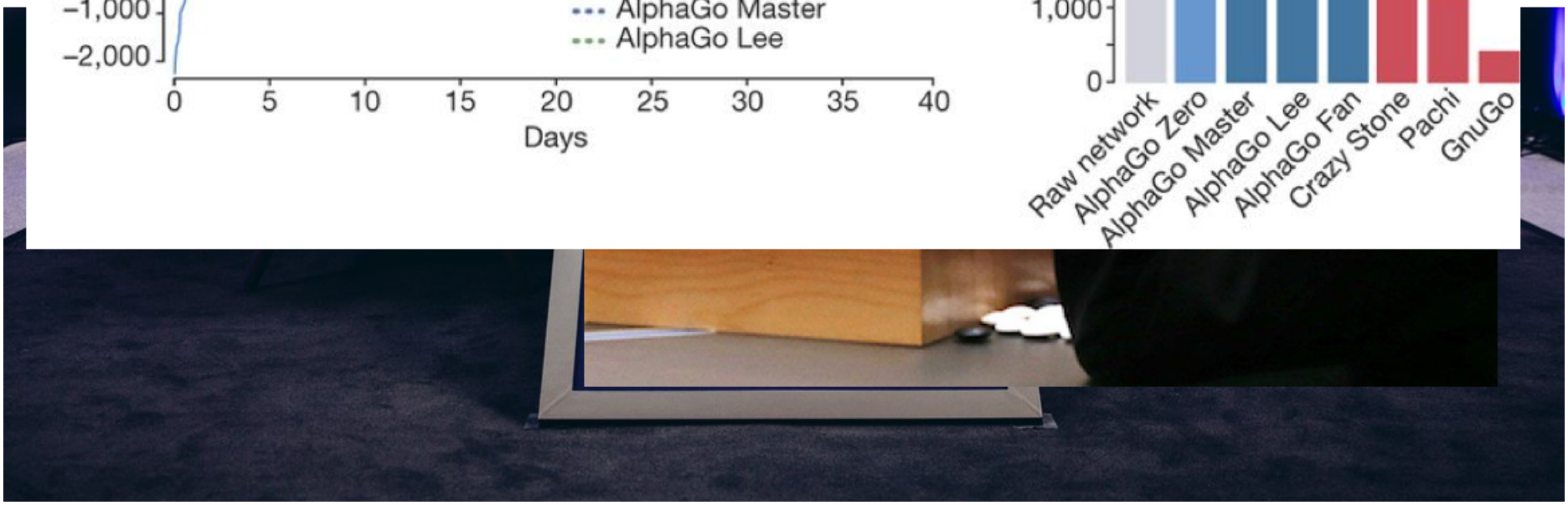
CADE METZ BUSINESS 03.14.16 02:40 PM

WHY THE FINAL GAME BETWEEN A









A Physics Problem to “Game-ify”?

A Physics Problem to "Game-ify"?

High Energy Physics - Theory

One in a Billion: MSSM-like D-Brane Statistics

Florian Gmeiner, Ralph Blumenhagen, Gabriele Honecker, Dieter Lust, Timo Weigand

(Submitted on 20 Oct 2005 (v1), last revised 22 Dec 2005 (this version, v3))

Continuing our recent work hep-th/0411173, we study the statistics of four-dimensional, supersymmetric intersecting D-brane models in a toroidal orientifold background. We have performed a vast computer survey of solutions to the stringy consistency conditions and present their statistical implications with special emphasis on the frequency of Standard Model features. Among the topics we discuss are the implications of the K-theory constraints, statistical correlations among physical quantities and an investigation of the various statistical suppression factors arising once certain Standard Model features are required. We estimate the frequency of an MSSM like gauge group with three generations to be one in a billion.

Comments:	36 pages, 12 figures; v2: typos corrected, one ref. added; v3: minor changes, version to appear in JHEP
Subjects:	**High Energy Physics - Theory (hep-th)**; High Energy Physics - Phenomenology (hep-ph)
Journal reference:	JHEP 0601:004,2006
DOI:	10.1088/1126-6708/2006/01/004
Report number:	MPP-2005-122, LMU-ASC 68/05
Cite as:	**arXiv:hep-th/0510170**
	(or **arXiv:hep-th/0510170v3** for this version)

A Physics Problem to "Game-ify"?

Search...

Help | Advanced

-Brane Statistics

Honecker, Dieter Lust, Timo Weigand

(this version, v3))

study the statistics of four-dimensional, supersymmetric intersecting D-brane
ve performed a vast computer survey of solutions to the stringy consistency
s with special emphasis on the frequency of Standard Model features. Among the
ory constraints, statistical correlations among physical quantities and an
factors arising once certain Standard Model features are required. We estimate the
frequency of an MSSM like gauge group with three generations to be one in a billion.

Comments:	36 pages, 12 figures; v2: typos corrected, one ref. added; v3: minor changes, version to appear in JHEP
Subjects:	**High Energy Physics - Theory (hep-th)**; High Energy Physics - Phenomenology (hep-ph)
Journal reference:	JHEP 0601:004,2006
DOI:	10.1088/1126-6708/2006/01/004
Report number:	MPP-2005-122, LMU-ASC 68/05
Cite as:	**arXiv:hep-th/0510170**
	(or **arXiv:hep-th/0510170v3** for this version)

A Physics Problem to "Game-ify"?

Search...

Help | Advanced

-Brane Statistics

Honecker, Dieter Lust, Timo Weigand

(this version, v3))

study the statistics of four-dimensional, supersymmetric intersecting D-brane
ve performed a vast computer survey of solutions to the stringy consistency
s with special emphasis on the frequency of Standard Model features. Among the
ory constraints, statistical correlations among physical quantities and an
factors arising once certain Standard Model features are required. We estimate the
frequency of an MSSM like gauge group with three generations to be one in a billion.

Comments:	36 pages, 12 figures; v2: typos corrected, one ref. added; v3: minor changes, version to appear in JHEP
Subjects:	**High Energy Physics - Theory (hep-th)**; High Energy Physics - Phenomenology (hep-ph)
Journal reference:	JHEP 0601:004,2006
DOI:	10.1088/1126-6708/2006/01/004
Report number:	MPP-2005-122, LMU-ASC 68/05
Cite as:	**arXiv:hep-th/0510170**
	(or **arXiv:hep-th/0510170v3** for this version)

A Physics Problem to "Game-ify"?

Search...

Help | Advanced

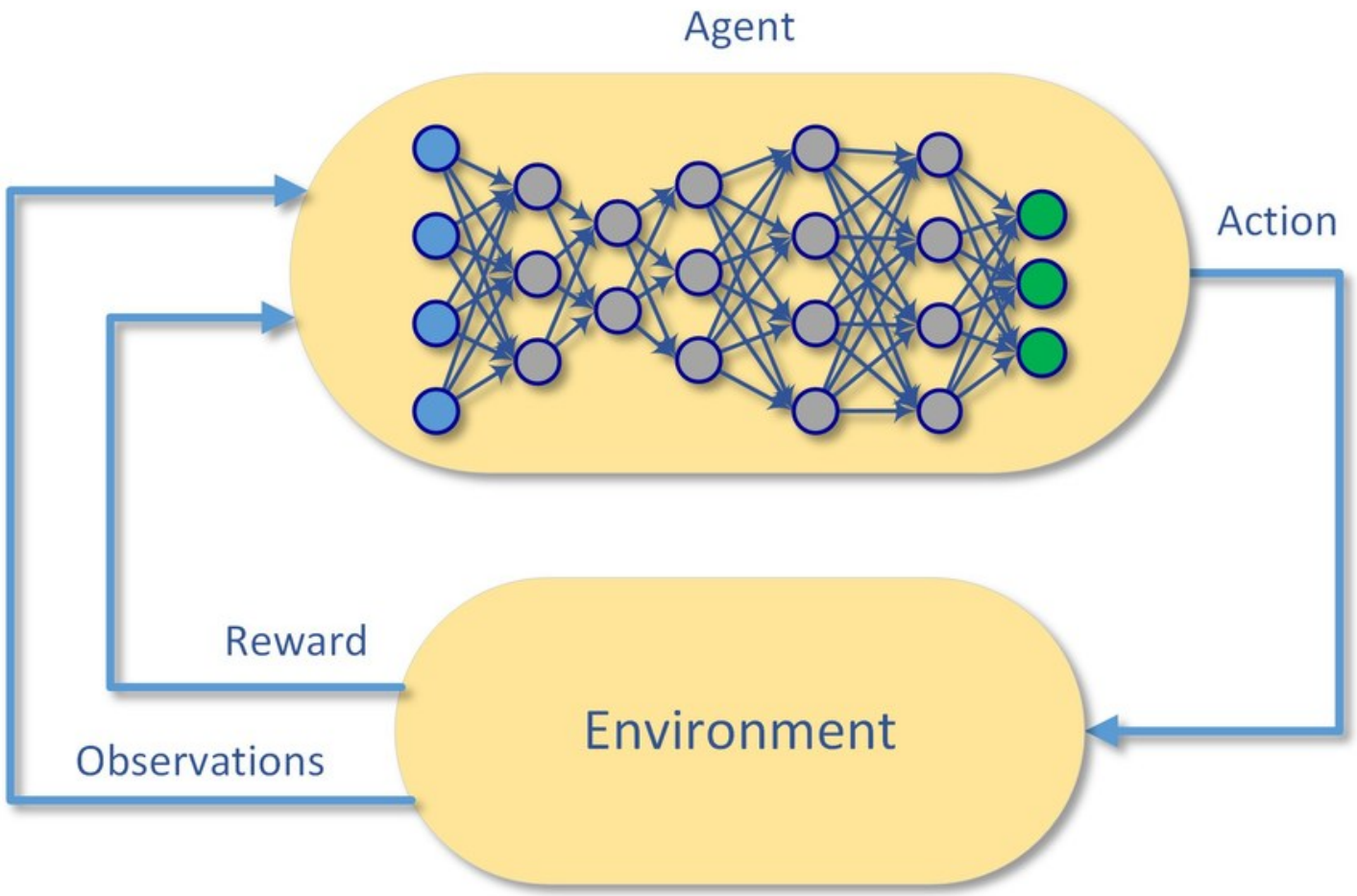
frequency of an MSSM like gauge group with three generations to be one in a billion.

Comments:	36 pages, 12 figures; v2: typos corrected, one ref. added; v3: minor changes, version to appear in JHEP
Subjects:	**High Energy Physics - Theory (hep-th)**; High Energy Physics - Phenomenology (hep-ph)
Journal reference:	JHEP 0601:004,2006
DOI:	10.1088/1126-6708/2006/01/004
Report number:	MPP-2005-122, LMU-ASC 68/05
Cite as:	arXiv:hep-th/0510170
	(or arXiv:hep-th/0510170v3 for this version)

Outline

1. What is Reinforcement Learning?
2. The Type-IIA “Game”
3. The Stacking Agent
4. Designing Rewards
5. It Learns!
6. Results

Reinforcement Learning (RL) in a Nutshell



- Use an intelligent agent to explore a state space (environment), modifying its exploration strategy via feedback (rewards)
- Reinforcement: the agent strives to find an optimal policy for action, after receiving multiple penalties/rewards

What is Reinforcement Learning?

⇒ Reinforcement learning takes place in an environment

- An agent perceives a subset of the environment data known as a state s
- Based on a policy π, the agent takes an action a that moves the system to a different state s'
- The agent receives a reward R based on the fitness of s'

⇒ Rewards may be accumulated as subsequent actions are taken, perhaps weighted by a discount factor γ

- The accumulated discounted reward is called the return $G(s)$
- The return depends on the state, and there are many possible returns for a given state based on the subsequent trajectory through state space

⇒ *The goal of an agent is to maximize the total future accumulated reward*

Value Functions

⇒ Simple, finite problems can often be solved exactly, and an optimal policy π_* determined analytically from the outset

⇒ For realistic problems, state space may remain largely unexplored – optimal policy will be approximate, arrived at iteratively, through exploration

⇒ Two key functions determine updates to the future policy of the agent:

State Value Function $v(s)$ | Action Value Function $v(s)$

$$v(s) = \mathbb{E}[G_t|S_t = s]. \qquad q(s,a) = \mathbb{E}[G_t|S_t = s, A_t = a].$$

- It is important to distinguish value from reward, as $v(s)$ captures the long-term value of being in s, not the short-term reward
- Both may be indexed by a subscript π if the trajectories through state space are determined by a policy π, i.e., $v_\pi(s)$ and $q_\pi(s,a)$

⇒ Finally, we define the advantage function

$$A_\pi(s,a) = q_\pi(s,a) - v_\pi(s),$$

which is an estimate of the advantage of taking the action a in the state s relative to the value of simply being in the state, as measured by $v_\pi(s)$.

Modifying the Agent's Policy

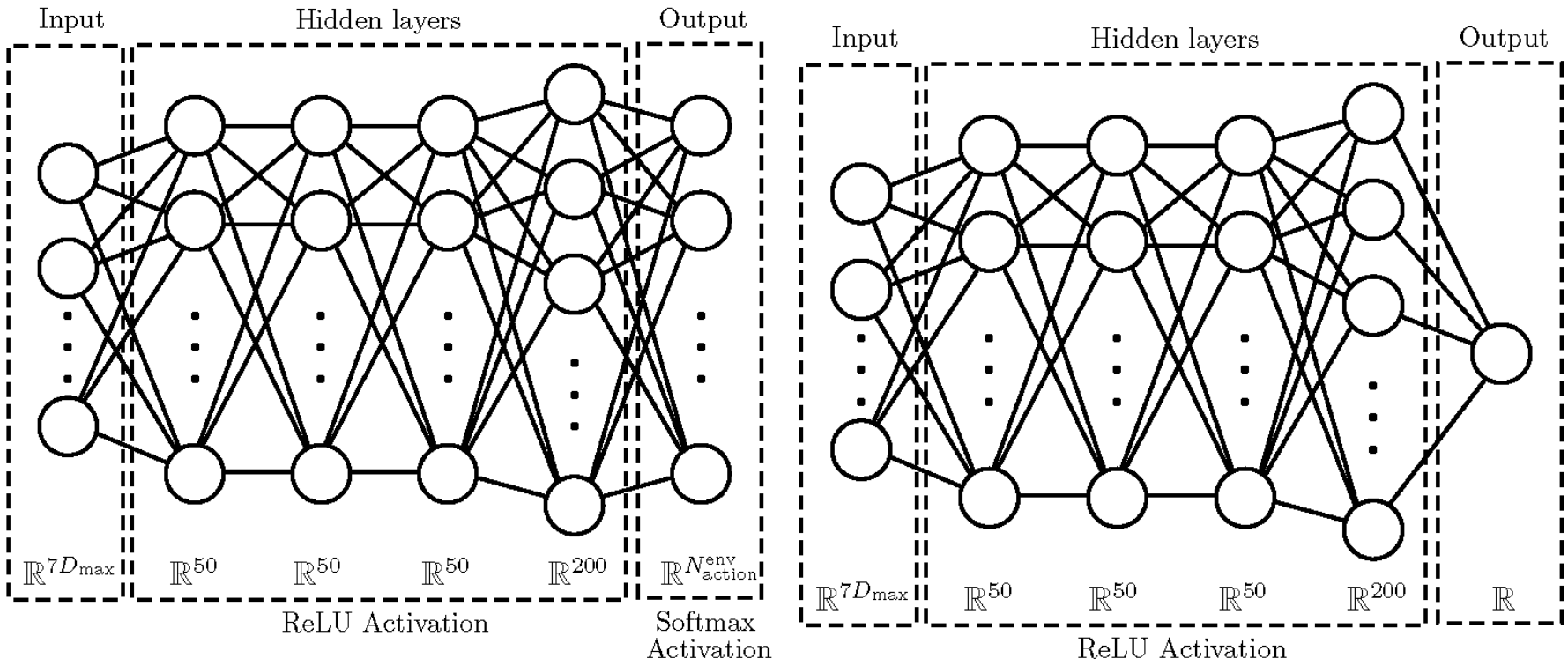
$\Rightarrow$ The previous value functions are modified each time the agent completes an episode, or "game"

- Episodes conclude when a terminal state is reached ("winning the game"), or a finite number of actions is accomplished
- Many episodes (games) are played and an average of returns (for a given state) over episodes is computed

$\Rightarrow$ Value functions are at the heart of RL, but difficult/computationally involved to compute

- In our environment, very large number of states – thus only a small subset of states will be sampled
- Must use function approximators – in "deep RL", these function approximators are deep neural networks (NNs)
- The NNs will parameterize these functions across all state space, in terms of a series of weights w (for the state) and θ (for the policy)

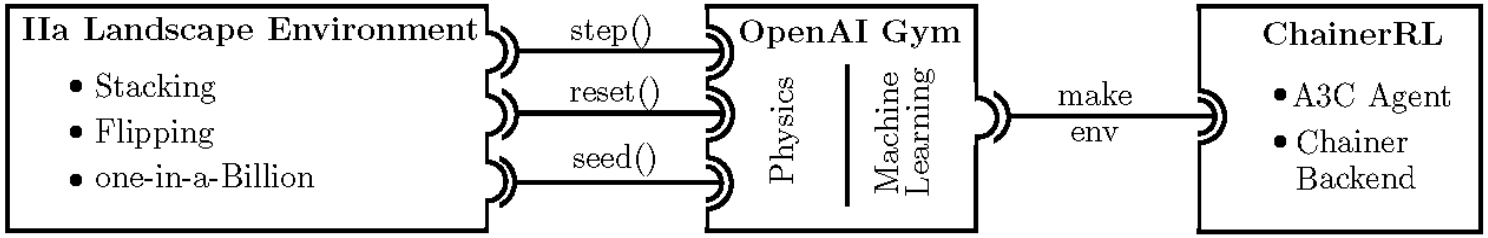
Deep Reinforcement Learning



$\Rightarrow$ Left: policy neural network. Right: state value neural network

- Both take as input the current state of the agent: a vector in $\mathbb{R}^{7D_{\max}}$
- Using stochastic gradient decent, weights are adjusted within each game as the agent uses its growing knowledge to improve its policy $\pi(s)$ within the episode

A3C Implementation



⇒ Our deep RL achitecture is the Asynchronous Advantage Actor-Critic (A3C) method, developed by Google's *DeepMind* group in 2016

- **Advantage**: the advantage function (described earlier) serves as the gradient the agent will follow as it traverses the state space
- **Actor-Critic**: we simultaneously use a function approximator for both the action-value function and the policy. the critic updates the action-value function approximator by adjusting the weights w, and the actor updates the policy weights θ in the direction suggested by the action-value function, that is, by the critic
- **Asynchronous**: many agents are run in parallel and updates are performed on neural networks as the ensemble of agents experience their environments

Deep RL and the Type IIA Game

$\Rightarrow$ Game setting: $T^6/(\mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_{2,0})$ orientifold compactifications of Type IIA

- The $\mathbb{Z}_2 \times \mathbb{Z}_2$ is the orbifold action; $\mathbb{Z}_{2,0}$ is the orientifold action
- Involution on $\mathbb{Z}_{2,0}$ gives an O6-plane whose RR charge is canceled by D6-branes

$\Rightarrow$ Assume the O6-plane and D6-branes wrap factorizable three cycles (one-cycles on each of the three T^2 factors comprising T^6)

- Each one-cycle specified by a vector in $\mathbb{Z}^2$ (the wrapping numbers): $(n_i,\, m_i)$, for $i = 1, 2, 3$
- For a product of three two-tori we define a three-cycle by

$$\pi_a = (n_1^a, m_1^a, n_2^a, m_2^a, n_3^a, m_3^a)$$

$\Rightarrow$ To move towards the RL implementation, let's define the following:

- A *plane* is a vector $(n_1, m_1, n_2, m_2, n_3, m_3) \in \mathbb{Z}^6$ that represents the O6-plane
- A *stack* is a vector $(N, n_1, m_1, n_2, m_2, n_3, m_3) \in \mathbb{Z}^7$ that represents a D6 stack

Deep RL and the Type IIA Game

$\Rightarrow$ Two complex structure assignments compatible with $\mathbb{Z}_2 \times \mathbb{Z}_2$

- Rectangular torus ($b_i = 0$) vs. tilted torus ($b_i = 1/2$)
- Allow for both by defining a wrapping number $\tilde{m}_i = m_i + b_i n_i$

$\Rightarrow$ Some notation to define: $\hat{b} = \left(\prod_{i=1}^{3}(1 - b_i)\right)^{-1}$

$$\hat{X}^0 = \hat{b} n_1 n_2 n_3 \,, \qquad \hat{X}^i = -\hat{b} n_i \tilde{m}_j \tilde{m}_k \,,$$
$$\hat{Y}^0 = \hat{b} \tilde{m}_1 \tilde{m}_2 \tilde{m}_3 \,, \qquad \hat{Y}^i = -\hat{b} \tilde{m}_i n_j n_k \,,$$

for $i, j, k \in \{1, 2, 3\}$ cyclic.

$\Rightarrow$ Finally, let $R_1^{(i)}$ and $R_2^{(i)}$ be the radii of the i^{th} torus and define products

$$U_0 = R_1^{(1)} R_1^{(2)} R_1^{(3)} \,, \qquad U_i = R_1^{(i)} R_2^{(j)} R_2^{(k)} \,,$$

with $i, j, k \in \{1, 2, 3\}$ cyclic.

- A *state* s is a set $s = (b_1, b_2, b_3, U_0, U_1, U_2, U_3, O, \mathcal{D})$, where $b_i \in \{0, \frac{1}{2}\}$, $U_0, U_i \in \mathbb{R}_+$, O is a plane, and $\mathcal{D}$ is a set of stacks. The set of states is denoted $\mathcal{S}$.

Deep RL and the Type IIA Game

⇒ Game rules: these D6-branes must satisfy tadpole cancellation, K-theory constraints and supersymmetry conditions (TCKS)

Tadpole Cancellation

$$\sum_a N_a \hat{X}^0_a = 8\hat{b}, \qquad \sum_a N_a \hat{X}^i_a = \frac{8}{1-b_i}, \qquad i \in \{1,2,3\}\,.$$

K-theory

$$\sum_a N_a \hat{Y}^0_a \equiv 0 \text{ mod } 2\,, \qquad (1-b_j)(1-b_k)\sum_a N_a \hat{Y}^i_a \equiv 0 \text{ mod } 2\,,$$

for $i,j,k \in \{1,2,3\}$ cyclic.

Unbroken supersymmetry: there exists (positive) radii $R_1^{(i)}$ and $R_2^{(i)}$ for the i^{th} torus such that

$$\sum_{I=0}^{3} \frac{\hat{Y}^I_a}{U_I} = 0, \qquad \sum_{I=0}^{3} \hat{X}^I_a U_I > 0\,.$$

Type IIA Game Objectives

$\Rightarrow$ Different sets of D6-branes represent different field theory outcomes

- Well-known relationships between O and $\mathcal{D}$ reveal the gauge group structure
- Similarly, well-known relationships amongst the stacks in $\mathcal{D}$ reveal the matter content

$\Rightarrow$ For the case of hypercharge, we seek only the existence of (at least) one massless $U(1)$

- Let the m_i^a be integers characterizing the unitary brane stacks
- Check for the vanishing of the following kernel, determining the generators of $U(1)$ factors

$$T_i = \text{ker}(N^a m_i^a)\,, \quad i = 1,2,3\,, \quad a = 1,\ldots,\text{number of U stacks}$$

$\Rightarrow$ Thus the spectrum can be represented as a function of the state $s \in \mathcal{S}$: $\mathcal{P}(s)$

$\Rightarrow$ An *action* a is a map $a : \mathcal{S} \to \mathcal{S}$ that changes the set of stacks $\mathcal{D}$. That is, we will hold O and the b_i fixed for a given 'game'

Douglas-Taylor: The 'Stacking' Environment

$\Rightarrow$ Take advantage of Douglas & Taylor: classify SUSY-consistent brane choices

M.R. Douglas, W. Taylor **JHEP 01** (2007) 031; hep-th/0606109

- **A-stacks**: A-branes have four non-vanishing tadpoles $\hat{X}^I$; four possible sets of signs on the triplets of n and m values
- **B-stacks**: B-branes have two vanishing and two non-vanishing tadpoles; 48 sign possibilities for winding numbers
- **C-stacks**: C-branes have one non-vanishing tadpole (three vanishing winding numbers); 16 sign possibilities for remaining winding numbers
  - ⋆ *C-branes have no effect on the SUSY condition!*

$\Rightarrow$ What is the size of state space?

- Let D_A, D_B, and D_C be the number of A, B, C-stacks that are considered
- Let N_A, N_B, and N_C be the max number of branes in any A, B, C-stack
- Let d_A and d_B be upper bound on the absolute value of any winding number for an A-stack or B-stack

Maximally conservative case: $N_A = N_B = N_C = 3$ and $D_A + D_B + D_C = 4$,

$d_A = d_B$	1	2	3	4	5
	7.4×10^7	3.6×10^{13}	1.5×10^{17}	6.2×10^{19}	6.9×10^{21}

Action Space of Stacking Environment

$\Rightarrow$ Allowed actions of the agent in the stacking environment:

- Each of the stacks (up to D_{max}) can be taken as an A-,B-, or C-brane stack
- The agent may change the number of branes N_a in each stack up to N_{max}
- If the agent sets N_a of any stack to zero, this brane stack is completely removed and an entirely new stack can be added

$\Rightarrow$ Let number of distinct A,B,C branes be μ_A, μ_B, μ_C, the cardinality $N_{\text{action}}^{\text{stack}}$ of the action space is

$$N_{\text{action}}^{\text{stacking}} = D_{\text{max}} + D_{\text{max}} + (\mu_A + \mu_B + \mu_C)$$

$\Rightarrow$ Define three 'distance' measures with respect to goals:

- Tadpole Condition Δ_{TC}

$$\Delta_{\text{TC}} := \sum_a \left(\left| N_a \hat{X}_a^0 - 8\hat{b} \right| + \sum_i \left| N_a \hat{X}_a^i (1 - b_i) - 8 \right| \right)$$

- Δ_{GG} and Δ_{EX} measures presence/absence of SM gauge group and presence/absence of SM representations, respectively

Designing Rewards

⇒ Rewards are given sequentially, depending on the 'game' we are playing

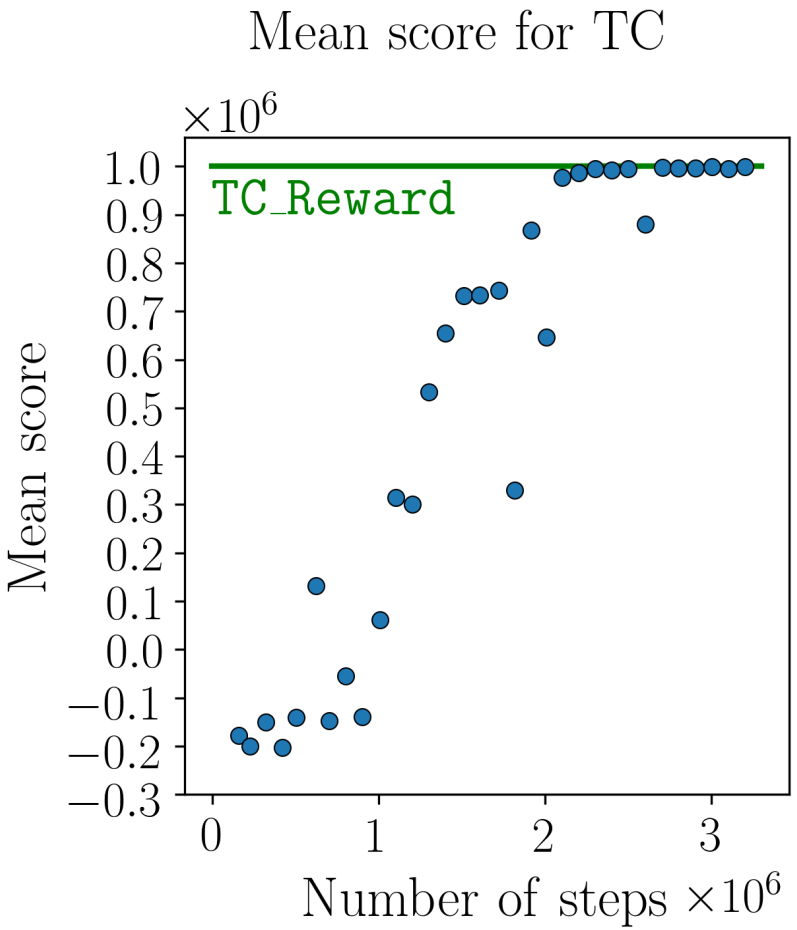
$$R^{\mathsf{C}} = \begin{cases} 0 & \text{if } 0 < \Delta_{\mathsf{TC}} \leq 8 \\ -\Delta_{\mathsf{TC}} \times \texttt{tadpoleDistanceMultipler} & \text{if } \Delta_{\mathsf{TC}} > 8 \\ \texttt{TC\_Reward} & \text{if TC, i.e. } \Delta_{\mathsf{TC}} = 0 \\ \texttt{TCK\_Reward} & \text{if TCK} \\ \texttt{TCKS\_Reward} & \text{if TCKS} \end{cases} .$$

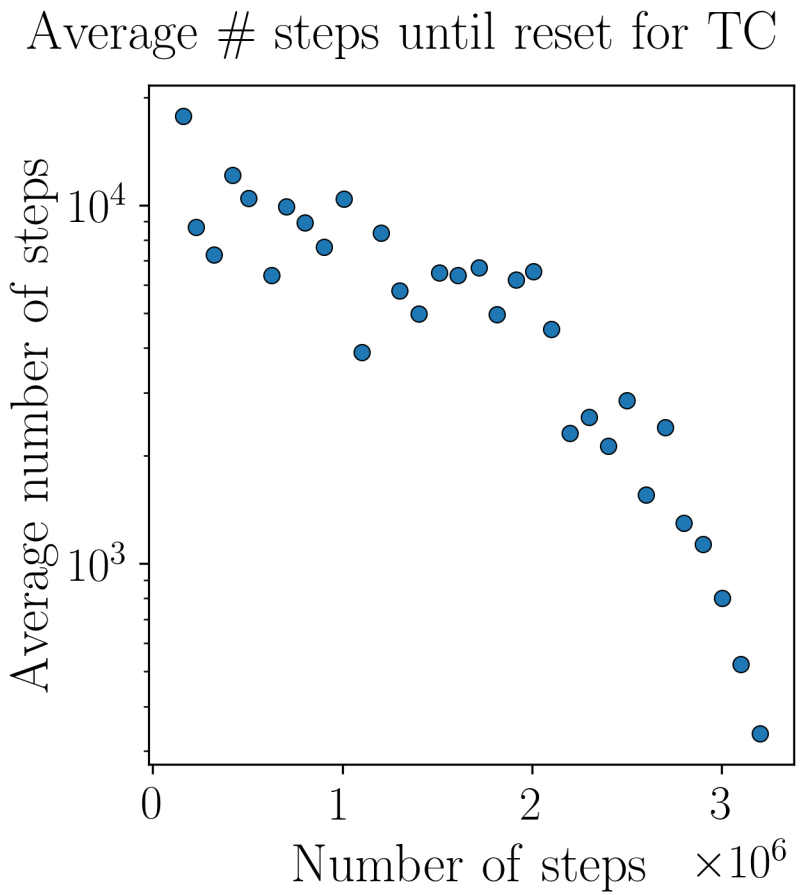
$$R^{\mathsf{SM}} = \begin{cases} -\Delta_{\mathsf{GG}} \times \texttt{missingGroupFactorDistance} & \text{if } \Delta_{\mathsf{GG}} \neq 0 \\ \texttt{SMlike\_Reward} - \Delta_{\mathsf{EX}} \times \texttt{missingParticleDistance} & \text{if } \Delta_{\mathsf{GG}} = 0, \Delta_{\mathsf{EX}} \neq 0 \\ \texttt{SM\_Reward} & \text{if } \Delta_{\mathsf{GG}} = \Delta_{\mathsf{EX}} = 0 \end{cases}$$

⇒ Typical 'points' values are as follows:

Property	Value
tadpoleDistanceMultiplier	1
TC_Reward	10^6
TCK_Reward	10^8
TCKS_Reward	10^9
missingGroupFactorDistance	10^4
missingParticleDistance	10^4
SMlike_Reward	10^{11}
SM_Reward	10^{13}

Evidence of Learning

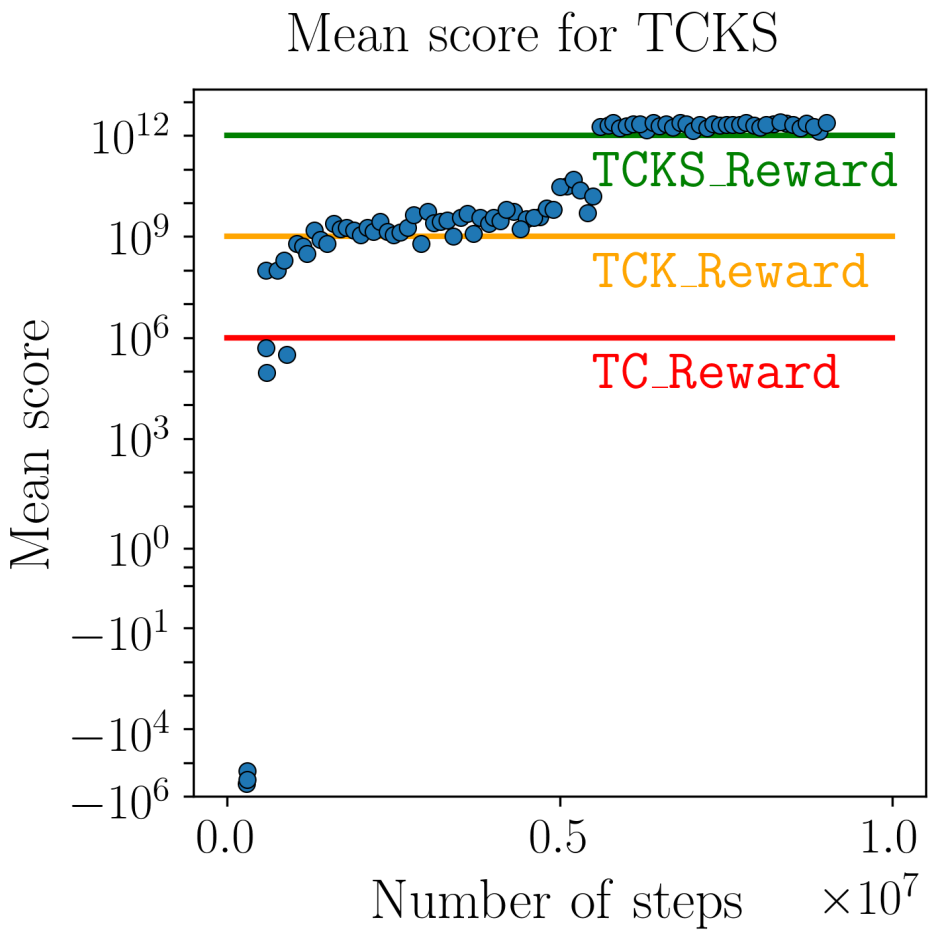




⇒ This game checks only TADPOLE CONSISTENCY

- Performance sampled 10 times after every 10^5 steps in training
- Consistent satisfaction sets in at about 2×10^6 steps
- Learning continues: average number of steps to 'win' continues to decline

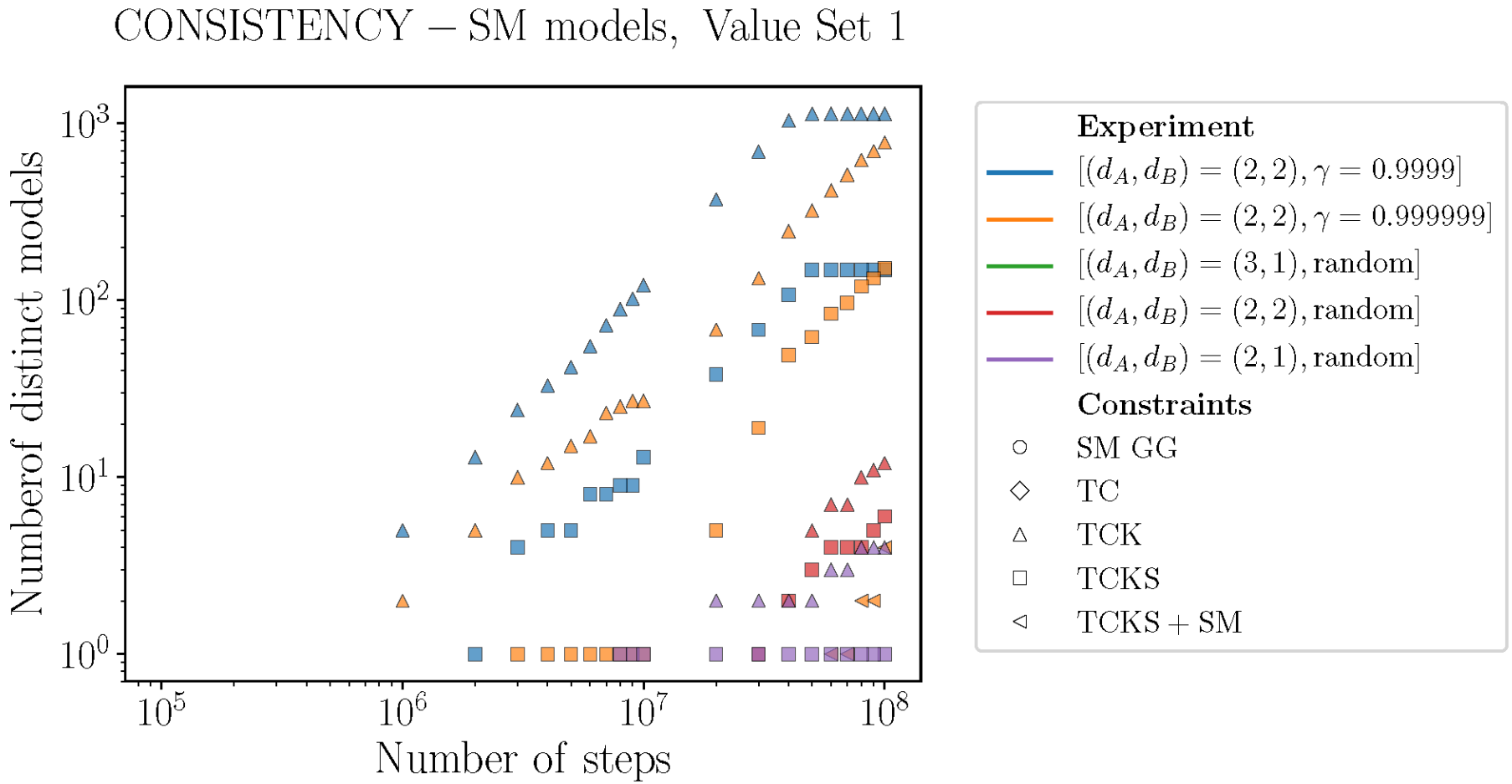
Evidence of Learning



$\Rightarrow$ This game seeks to satisfy all mathematical consistency conditions

Property	Value
TC_Reward	10^{6}
TCK_Reward	10^{9}
TCKS_Reward	10^{12}

Result 1: Far Outperforms Random Sampling



$\Rightarrow$ Comparison of RL agent (blue, orange) versus random walks (red, violet)

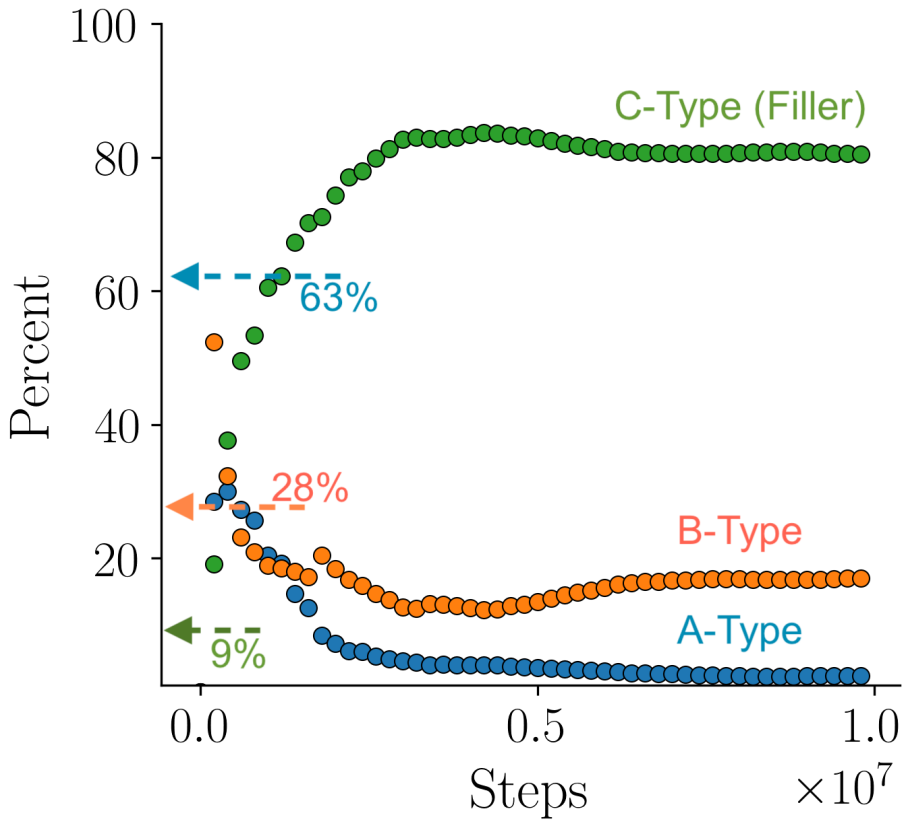
- RL agents conquer TCK at 10^6 steps, SUSY at 2×10^6; random walker achieves both at $\sim 10^7$ steps
- After 10^8 steps, RL agents find $\mathcal{O}(200)$ more TCK models, and $\mathcal{O}(50)$ more TCKS models, than random walker

Result 2: Learns Human-derived Strategy

- Change game: now require SUSY and TCK *simultaneously* (below)
- Note how quickly agent moves away from "random" mix of A,B,C-branes
- Machine has learned the technique of "filler branes": C-branes which do not contribute to SUSY condition at all, but useful for satisfying tadpole cancellation conditions

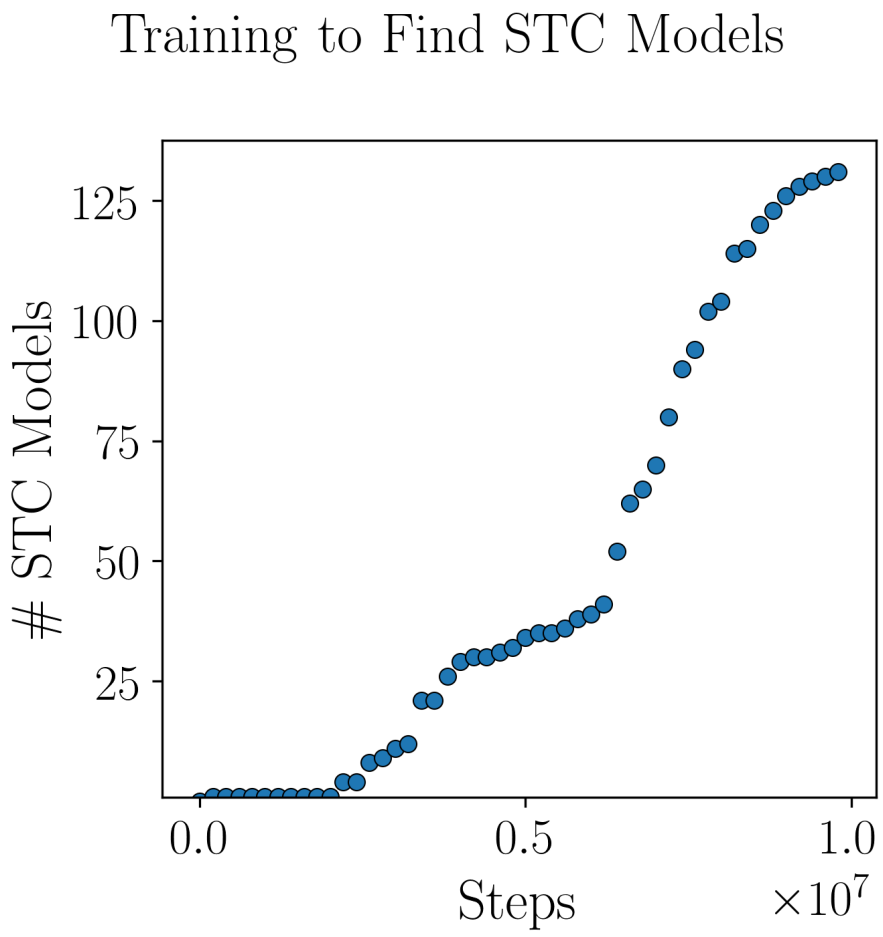
c.f. Cvetic, Li, Liu **NPB 698** (2004) 163; hep-th/0403061

$(d_A, d_B) = (2, 1)$ and $|\mathcal{D}| = 10$



$$R^{\text{STC}} = \begin{cases} 0 & \text{if } 0 < \Delta_{\text{TC}} \leq 8 \\ -\Delta_{\text{TC}} \times \texttt{tadpoleDistanceMultipler} & \text{if } \Delta_{\text{TC}} > 8 \\ \texttt{TC\_Reward} & \text{if TC, i.e. } \Delta_{\text{TC}} = 0 \\ \texttt{S\_Reward} & \text{if S} \\ \texttt{STC\_Reward} & \text{if STC} \end{cases} .$$

Result 3: Learns New Strategy, 2x More Efficient



Performance of "filler brane" strategy after 24 hours

- 125 TCKS models in $\mathcal{O}(10^7)$ steps

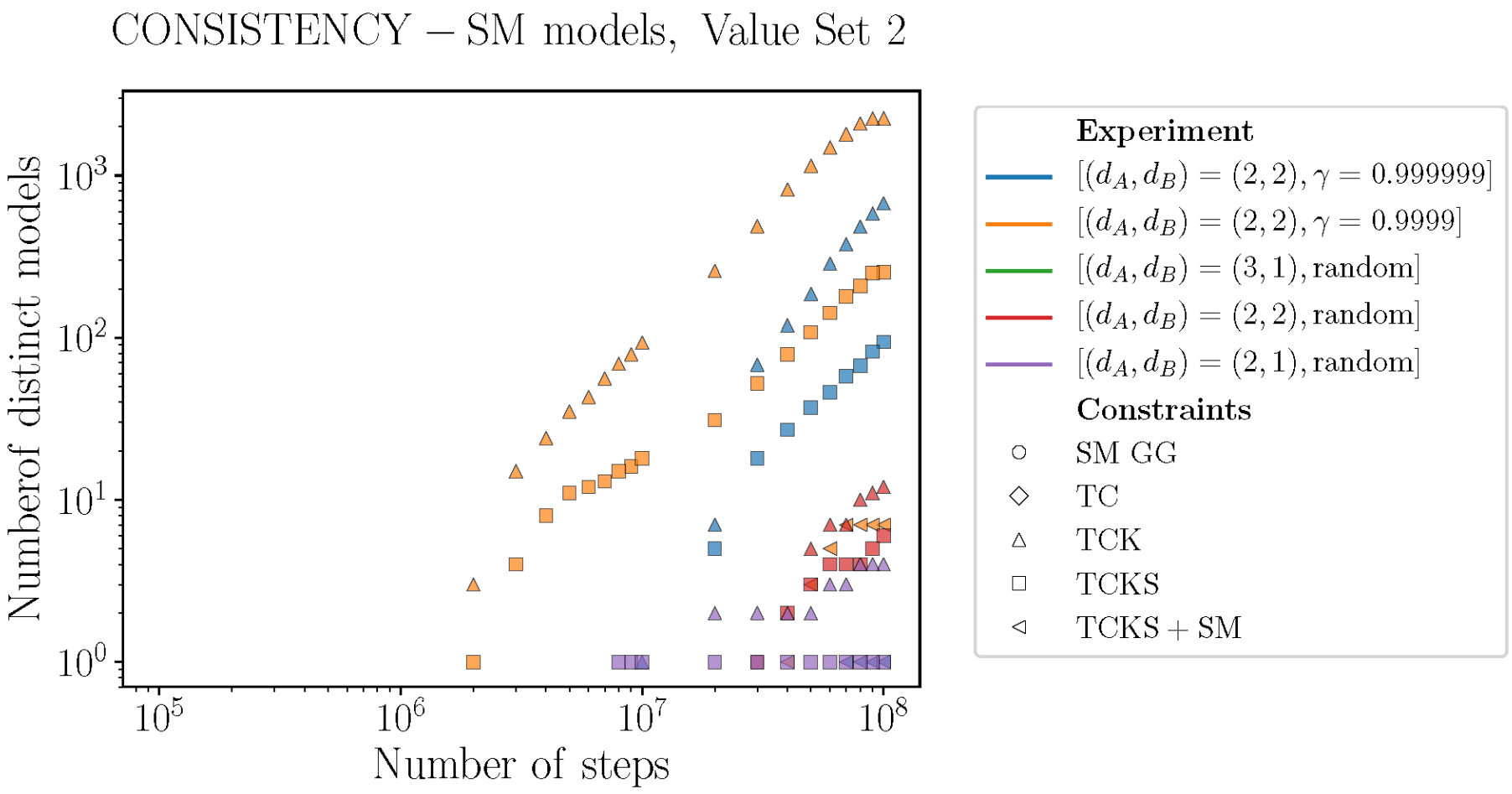
$\Rightarrow$ Performance of best RL agent in stacking environment

- Over 200 TCKS models discovered in 24 hours
- More steps (10^8), but more efficient (SUSY not checked at each step)

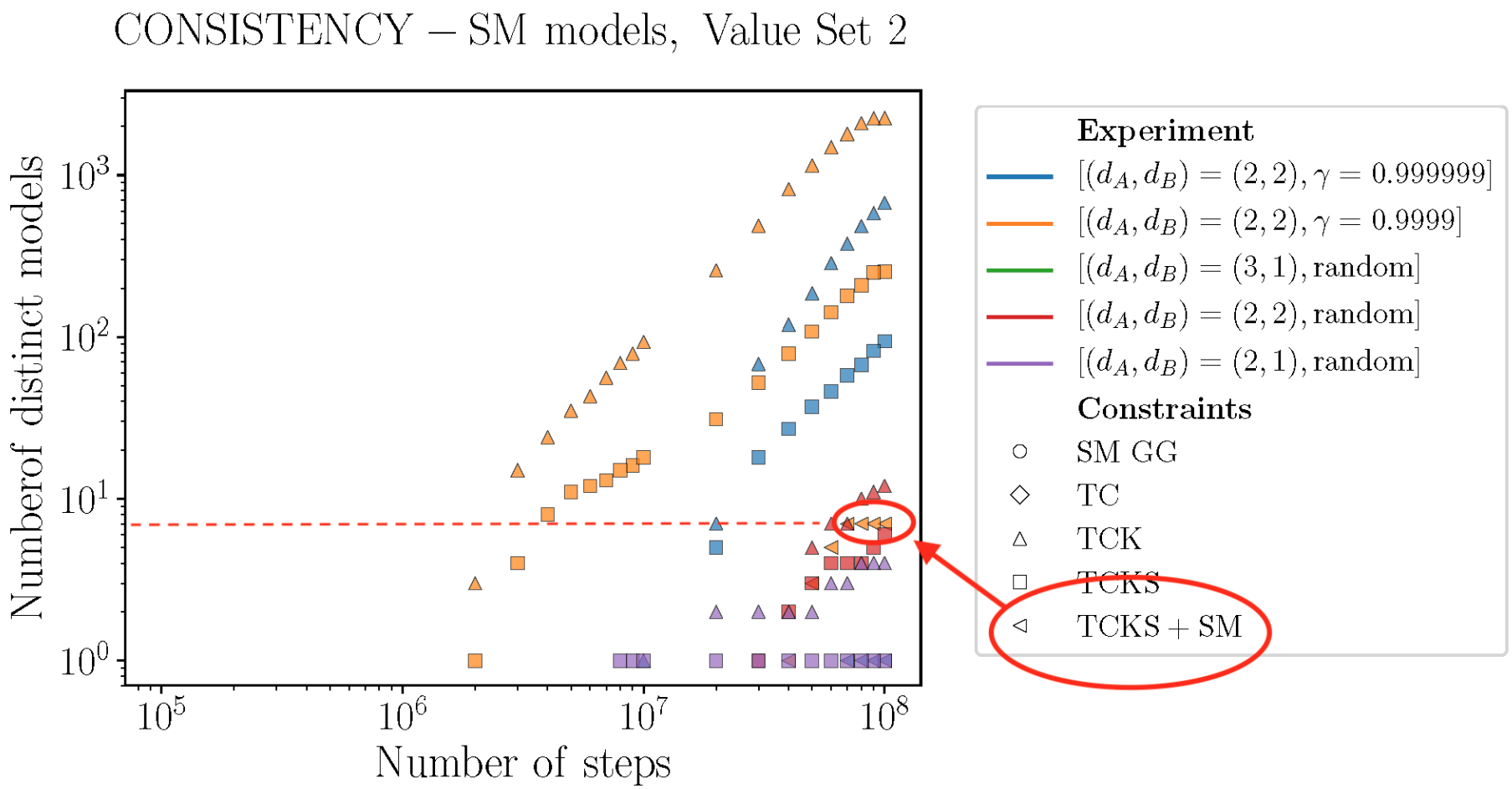
$\Rightarrow$ RL agent checks tadpole conditions first – *cannot* be utilizing the filler brane strategy

- Checking tadpole cancellation first saves time from the costly evaluation of the SUSY conditions
- Conclusion: RL agent has learned a new strategy for finding consistent string models that is about twice as efficient per unit time as the filler brane strategy

Did We Beat the Humans?



Did We Beat the Humans?



⇒ TCKS + SM here means SM gauge group: $SU(3) \times SU(2)$ and at least one massless $U(1)$

⇒ Do find cases with net number of three generations of quarks; however, all cases involve $\mathcal{O}(10)$ exotics

Summary

⇒ Advantages of reinforcement learning as a tool:

1. Allows for coarse graining of model (state) space: can use NNs as function approximators for both the policy and value ('reward') function. Crucial for large state spaces!
2. May allow for search strategies that have never been tried before, or improve currently existing strategies
3. Can be somewhat model-free – plugging in different environments for same AC3 – means less re-programming once a good architecture is designed ('plug and play')

⇒ Our key results:

1. RL agents improve on random scans by two orders of magnitude
2. RL agents can learn human-inspired strategies, and devise highly effective strategies of their own
3. For the Type IIA landscape game, humans are still the masters..... *for now!*

An Invitation

String Phenomenology, June 2020

Northeastern University

THANK YOU!

Backup Slides

Formal Definitions of RL Objects

- A **state** s (or S_t) represents what the agent measures from the environment, set of states is $\mathcal{S}$.
- An **action** a (or A_t) to move from one state to another. $\mathcal{A}$ is the abstract set of actions, and $\mathcal{A}(s)$ is the set of actions possible in the state s
- A **policy** is a map from states to actions, $\pi : \mathcal{S} \to \mathcal{A}$. A *deterministic policy* $\pi(s)$ picks a unique action a for each state s, and a *stochastic policy* $\pi(a|s)$ is the probability of the agent selecting action a given that it is in state s.
- The **reward** $R_t \in \mathbb{R}$ at a given time t depends on the state S_t. The goal of an agent is to maximize the total future accumulated reward
- The **return** measures accumulated rewards from time t,

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

where $\gamma \in [0, 1]$ is the *discount factor* and the sum truncates for an episodic task.

Markov Decision Processes

$\Rightarrow$ The method for finding the optimal versions of value functions (i.e. the optimal policy π_*), is that of a Markov Decision Process (MDP)

$\Rightarrow$ The MDP describes the interactions of the agent with the environment, and when the MDP is solved the agent has optimal behavior

- Dynamic programming: if you have a set of stochastic policies $\pi(a|s)$, and a set of state transition probabilities $p(s'|s,a)$ (the probability of transition to a state s' given s and an action a), then one can compute $v_\pi(s)$ via recursion, **assuming** you can sum over **all** states s'.
- Monte Carlo: the agent plays a large number N of episodes and gathers returns from the states of the episode. Then the value function may be approximated by

$$v(s) = \mathbb{E}[G(t)|S(t) = s] \simeq \frac{1}{N}\sum_{i=1}^{N} G_i(s)$$

- Temporal difference learning: like MC, agents in TD learn directly from raw experience without a model of the environment's dynamics, as required for DP. On the other hand, TD methods update estimates based on learning, as in DP, rather than waiting for the final outcome at the end of an episode, as in MC. Therefore TD methods may be applied with each action of the agent.

Solving the SUSY Condition

⇒ For computational efficiency, want to turn system of equality/inequalities into a linear algebra problem

⇒ Rewrite SUSY condition as

$$\frac{1}{U_0}\left(\hat{Y}_a^0 + \hat{Y}_a^1\frac{U_0}{U_1} + \hat{Y}_a^2\frac{U_0}{U_2} + \hat{Y}_a^3\frac{U_0}{U_3}\right) = 0,$$

$$A_a + B_a\, j + C_a\, k + D_a\, l = 0, \qquad \forall a.$$

where we defined $j = \frac{U_0}{U_1}\,, \qquad k = \frac{U_0}{U_2}\,, \qquad l = \frac{U_0}{U_3}$

⇒ This defines a particular hyperplane in moduli space, leading to the following algorithm:

1. Find a triple of brane stacks such that the three hypersurface equations define a matrix equation with full rank
2. Invert that matrix to solve uniquely for the (j, k, l) consistent with supersymmetry
3. Check for those specific (j, k, l) whether the rest of the supersymmetry conditions are satisfied

Determination of the Gauge Group

The overall gauge group is given by

$$G = \bigotimes_{a=1}^{|\mathcal{D}|} G_a \,,$$

where $|\mathcal{D}|$ is the number of D6 brane stacks and G_a is a non-Abelian Lie group whose type is determined by the intersection of the brane stack with the orientifold plane.

- $G_a = \mathrm{U}(N_a)$ if π_a and π_{O6} are in general position,
- $G_a = \mathrm{SO}(2N_a)$ if π_a is on top of π_{O6},
- $G_a = \mathrm{USp}(N_a)$ if π_a is orthogonal to π_{O6}.

$\Rightarrow$ Generators T_i of the massless U(1)s are given by kernel of $3 \times K$ matrix

$$T_i = \ker(N^a m_i^a) \,, \quad i = 1,2,3 \,, \quad a = 1, \ldots, \text{number of U stacks} \,,$$

where K is the number of brane stacks with unitary gauge group and the m_i^a are integers characterizing the unitary brane stacks.

Determination of Matter Representations

$\Rightarrow$ Intersection of cycles defined by

$$I_{ab} = \prod_{i=1}^{3} (n_i^a m_i^b - n_i^b m_i^a)$$

$\Rightarrow$ Matter representation determination:

- Bifundamental matter $(\Box_a, \overline{\Box}_b)$ may arise at the intersection of D6-branes on π_a and π_b, with chiral index $\chi(\Box_a, \overline{\Box}_b) = \pi_a \cdot \pi_b \in \mathbb{Z}$, where $\Box$ and $\overline{\Box}$ denote the fundamental and anti-fundamental representation of the associated stack. Similarly, $\chi(\Box_a, \Box_b) = \pi_a \cdot \pi_b' \in \mathbb{Z}$.

- Matter in the two-fold symmetrized representation $(\begin{smallmatrix}\Box\Box\end{smallmatrix})_a$ may arise at the intersection of a D6-brane with the orientifold brane, with chiral index $\chi(\begin{smallmatrix}\Box\Box\end{smallmatrix})_a = \frac{1}{2}(\pi_a \cdot \pi_a' - \pi_a \cdot \pi_{O6}) \in \mathbb{Z}$.

- Matter in the two-fold anti-symmetrized representation $(\begin{smallmatrix}\Box\\\Box\end{smallmatrix})_a$ may arise at the intersection of a D6-brane with the orientifold brane, with chiral index $\chi(\begin{smallmatrix}\Box\\\Box\end{smallmatrix})_a = \frac{1}{2}(\pi_a \cdot \pi_a' + \pi_a \cdot \pi_{O6}) \in \mathbb{Z}$.

Game Variant: The Flipping Environment

⇒ Agents can increase/decrease the number of branes in any given stack, or increment/decrement any winding numbers in any of the stacks by one unit

- $\mathcal{D} \to \mathcal{D}'$ by selecting a single stack $d_a \in \mathcal{D}$ and adjusting the number of branes in this stack, $N_a \to N_a \pm 1$, or increasing/decreasing a single winding number n_i^a or m_i^a by one unit
- Truncate state space: put a cap on maximum values of $|m|$, n, N
- Depending on the tilting of the torus and the winding number, this increase might be half-integer or integer
- For the latter, if co-prime condition violated, keep incrementing (with no penalty) until satisfied

⇒ Number of the actions $N_{\text{action}}^{\text{flipping}}$ of the flipping environment

$$N_{\text{action}}^{\text{flipping}} = D_{\text{max}} + D_{\text{max}} + 6D_{\text{max}} + 6D_{\text{max}}$$

⇒ Game initialized from random but fixed set of winding configurations for each of the D_{max} states, and populate each stack with a random but fixed number of branes N_a

Game Variant: One-in-a-Billion

⇒ Like a "cheat code" for the game: set $D_{\text{max}} = 4$ and fix the numbers of branes per stack to $N_a = (3, 2, 1, 1)$

- We either change winding numbers only (flipping), or we change the entire nature of the stack (from the A,B,C list) (stacking)
- Number of possible actions
  - ⋆ Stacking ($D_{\text{max}} = 4$): μ counts number of distinct branes of each type:

$$N_{\text{action}}^{\text{1:B-stacking}} = 4(\mu_A + \mu_B + \mu_C)$$

  - ⋆ Flipping ($D_{\text{max}} = 4$): each stack has 6 winding numbers that can be decreased or increased

$$N_{\text{action}}^{\text{1:B-flipping}} = 4 \times 6 \times 2 = 48$$

Are the Solutions Different?