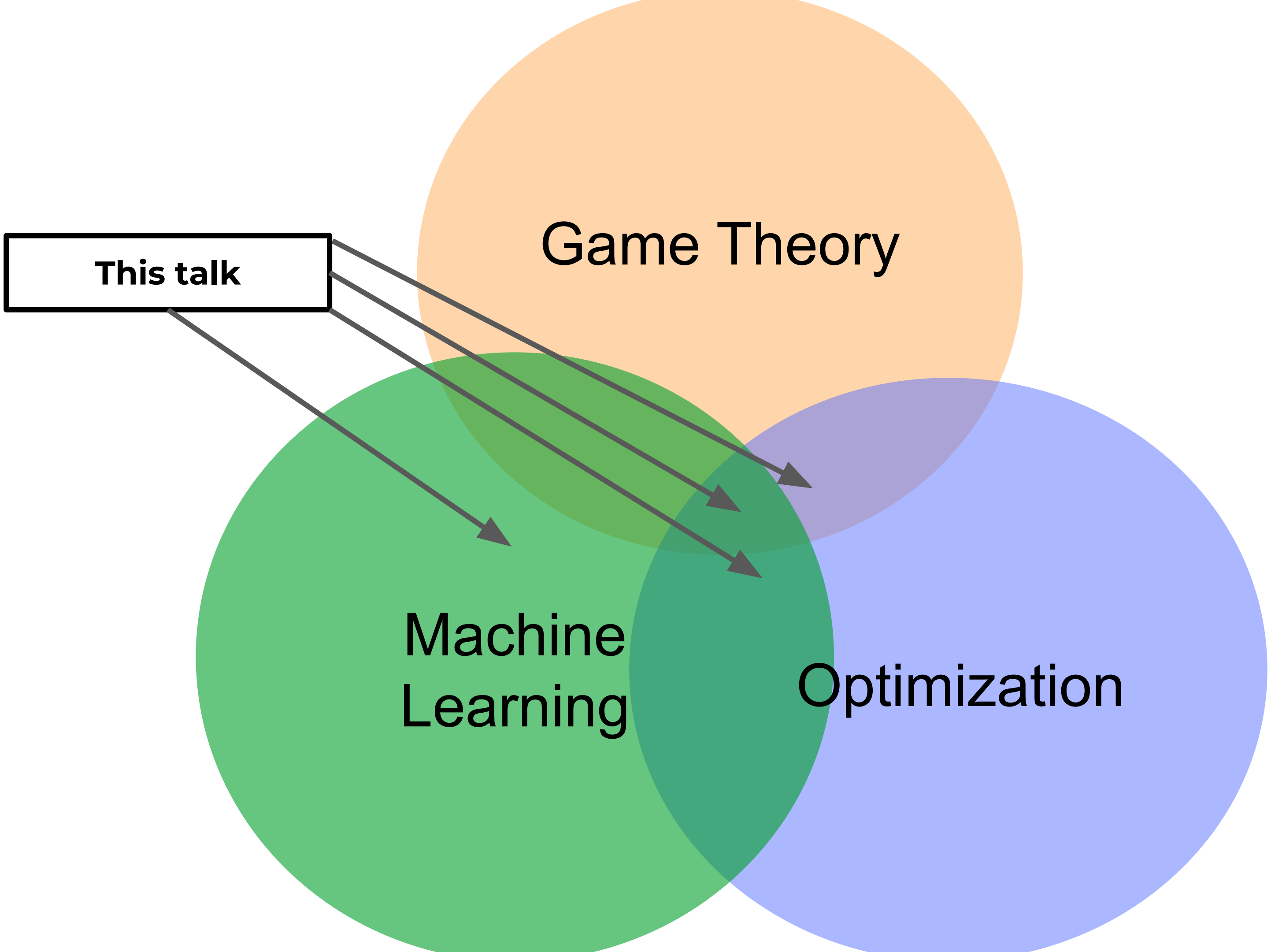
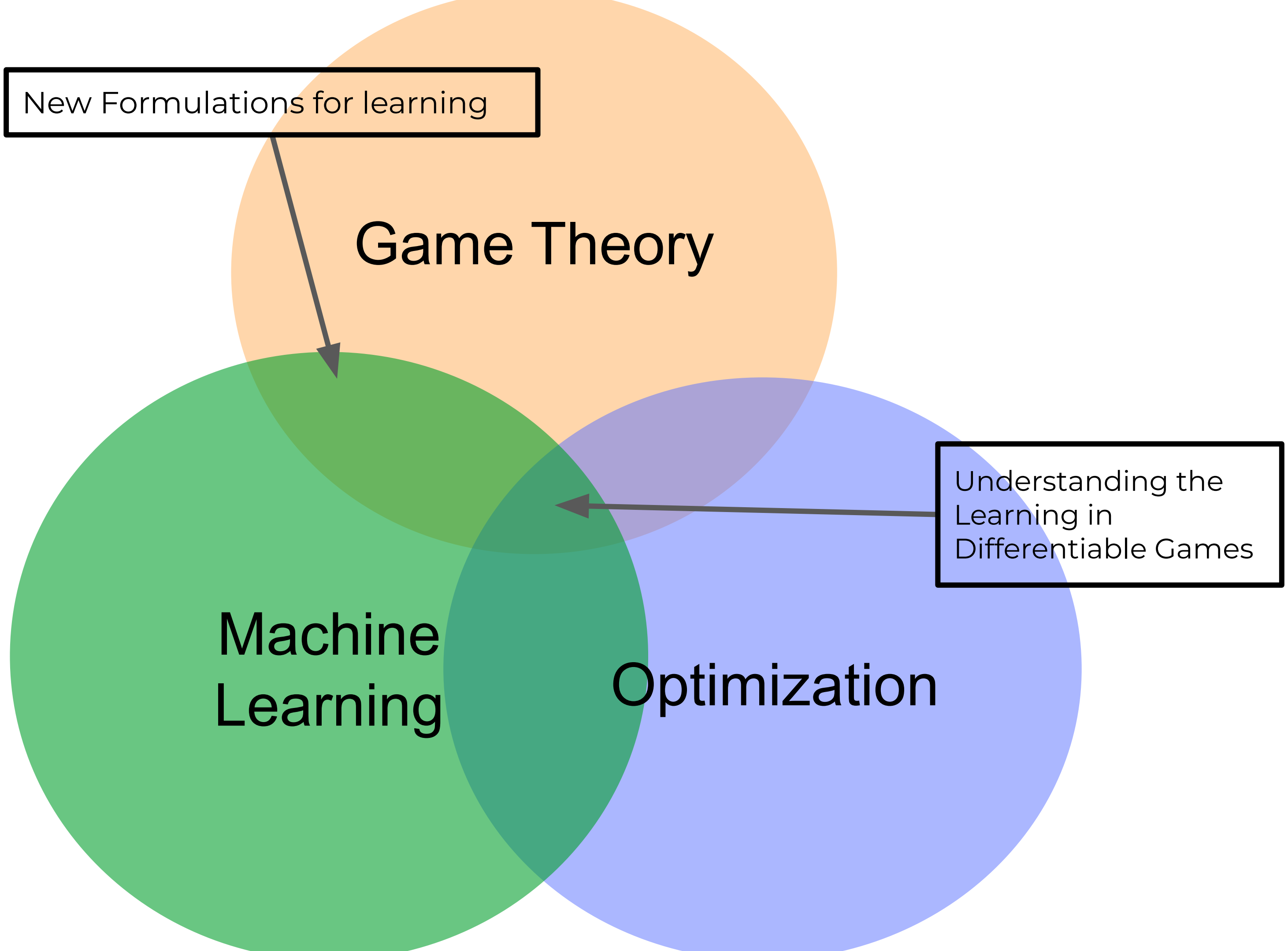
Differentiable Games in the Era of Machine Learning

Gauthier Gidel

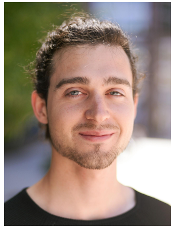
Mila and DIRO

Differentiable Games in the Era of Machine Learning


Game Theory
This talk
Machine Learning
Optimization


New Formulations for learning
Game Theory
Understanding the
Learning in
Differentiable Games
Machine
Learning
Optimization

Adversarial Example Games

Avishek Joey Bose*, **Gauthier Gidel***, **Hugo Berard***, Andre Cianflone, Pascal Vincent, Simon Lacoste-Julien, William L. Hamilton

Mila, McGill University, Université de Montréal, Facebook AI Research

Université de Montréal FACEBOOK AI RLLab McGill University CIFAR

Standard Adversarial Attack Setting:

$$\boxed{x' \in \text{argmax}_{x' \in \mathcal{X}} \, \ell(f(x'), y)\,, \quad \text{s.t.} \quad d(x, x') \leq \epsilon\,.}$$

Optimization Problem

Usually L_p norm.

We Need to know the function to optimize

- f : function to attack.
- $x \in \mathcal{X}$: input datapoint.
- $x' \in \mathcal{X}$: adversarial example.
- $y \in \mathcal{Y}$: true label.
- ℓ : loss function.

Standard Adversarial Attack Setting:

$$x' \in \mathrm{argmax}_{x' \in \mathcal{X}} \, \ell(f(x'), y) \,, \quad \text{s.t.} \quad d(x, x') \leq \epsilon \,.$$

Optimization Problem

Usually L_p norm.

- f : function to attack.
- ℓ : loss function

Threat model: what we assume to have access to.
(e.g. gradients, softmax values)

Whitebox threat model

Blackbox threat model

We Need to know the function to optimize

Intuitions

- Adversarial examples are **features**. [Ilyas et al. 2019]
- Adversarial examples **always exist** with Neural Nets
  [Bubeck, Cherapanamjeri, Gidel, Tachet des Combes 2021] [Daniely and Schacham 2020]

- These features can be learned.
- Modifying them can attack a whole class $\mathcal{F}$ function.

Conclusion: the generator can learn to detect and change these features **without querying** f_t $\implies$ **NoBox attack.**

A Realistic (and challenging) threat model: **No**n-interactive black**Box** (**NoBox**) threat model

- **Target model** f_t : we want to break that model.
- **Target examples** $\mathcal{D}$: the data we want to corrupt.
- **Model hypothesis class** $\mathcal{F}$: our knowledge on the target model. **New!**
- **Representative classifier** f_c : we assume we can optimize over the hypothesis class using that representative classifier. **New!**
- **A Reference Dataset** $\mathcal{D}_{ref}$: similar to the training set of f_t **New!**

IDEA: Optimize over $\mathcal{F}$ to get adversarial examples that can attack any function in $\mathcal{F}$

Adversarial Example Games Framework

Game Between:

- A generator that generate adversarial examples conditioned on (x,y):

$$(x', y) \sim p_g \Leftrightarrow x' = g(x, y, z)\,,\ (x, y) \sim \mathcal{D}\,,\ z \sim p_z \quad \textbf{with} \quad d(x', x) \leq \epsilon\,.$$

- A Classifier f_c that aims at getting robust against adversarial examples:

<u>Classification loss of an adversarial example of (x,y):</u>

$$\ell(\overbrace{f_c(\underbrace{g(x, y, z)}_{\text{Adversarial example}})}^{\text{Prediction by the classifier}}, \overset{\text{True label}}{y})$$

Adversarial Example Games Framework

Game Between:

- A generator that generate adversarial examples conditioned on (x,y):

$$(x', y) \sim p_g \Leftrightarrow x' = g(x, y, z)\,,\ (x, y) \sim \mathcal{D}\,,\ z \sim p_z \quad \textbf{with} \quad d(x', x) \leq \epsilon\,.$$

- A Classifier f_c that aims at getting robust against adversarial examples:

$$\boxed{\max_{g \in \mathcal{G}_\epsilon} \min_{f_c \in \mathcal{F}} \mathbb{E}_{(x,y)\sim\mathcal{D}, z\sim p_z}[\ell(f_c(g(x,y,z)), y)] =: \varphi(f_c, p_g)}$$

Attacking in the Wild: CIFAR 10

Target classifier we want to **attack.**

f_t

Architecture of classifier used to **train** attacker.

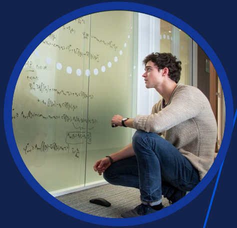
f_c

Source	Attack	VGG-16	RN-18	WR	DN-121	Inc-V3
	Clean	11.2 ± 0.9	13.1 ± 2.0	6.8 ± 0.7	11.2 ± 1.4	9.9 ± 1.3
RN-18	MI-Attack	63.9 ± 1.3	74.6 ± 0.4	63.1 ± 1.2	72.5 ± 1.3	67.9 ± 1.6
	DI-Attack	77.4 ± 1.7	90.2 ± 0.8	74.0 ± 1.0	87.1 ± 1.3	$\mathbf{85.8 \pm 0.8}$
	TID-Attack	21.6 ± 1.3	26.5 ± 2.2	14.0 ± 1.5	22.3 ± 1.6	19.8 ± 0.9
	SGM-Attack	68.4 ± 1.8	79.5 ± 0.5	64.3 ± 1.6	73.8 ± 1.0	70.6 ± 1.7
	AEG (Ours)	$\mathbf{89.0 \pm 2.1}$	$\mathbf{96.8 \pm 0.7}$	$\mathbf{80.9 \pm 2.4}$	$\mathbf{91.6 \pm 1.7}$	$\mathbf{87.2 \pm 1.6}$
DN-121	MI-Attack	54.3 ± 1.1	62.5 ± 0.9	56.3 ± 1.3	66.1 ± 1.5	65.0 ± 1.3
	DI-Attack	61.1 ± 1.9	69.1 ± 0.8	61.9 ± 1.1	77.1 ± 1.2	71.6 ± 1.6
	TID-Attack	21.7 ± 1.2	23.8 ± 1.5	14.0 ± 1.4	21.7 ± 1.1	19.3 ± 1.2
	SGM-Attack	51.6 ± 0.7	60.2 ± 1.3	52.6 ± 0.9	64.7 ± 1.6	61.4 ± 1.3
	AEG (Ours)	$\mathbf{90.5 \pm 1.6}$	$\mathbf{95.9 \pm 1.4}$	$\mathbf{80.3 \pm 2.3}$	$\mathbf{95.9 \pm 1.4}$	$\mathbf{90.6 \pm 2.4}$
VGG-16	MI-Attack	49.9 ± 0.1	50.0 ± 0.2	46.7 ± 0.4	50.4 ± 0.6	50.0 ± 0.3
	DI-Attack	65.1 ± 0.1	64.5 ± 0.2	58.8 ± 0.6	64.1 ± 0.3	60.9 ± 0.6
	TID-Attack	26.2 ± 0.6	24.0 ± 0.6	13.0 ± 0.2	20.8 ± 0.7	18.8 ± 0.2
	AEG (Ours)	$\mathbf{94.2 \pm 1.2}$	$\mathbf{93.7 \pm 1.6}$	$\mathbf{77.1 \pm 1.1}$	$\mathbf{92.3 \pm 1.7}$	$\mathbf{86.5 \pm 1.3}$

Table 2: Error rates on $\mathcal{D}$ for average NoBox architecture transfer attacks with $\epsilon = 0.03125$

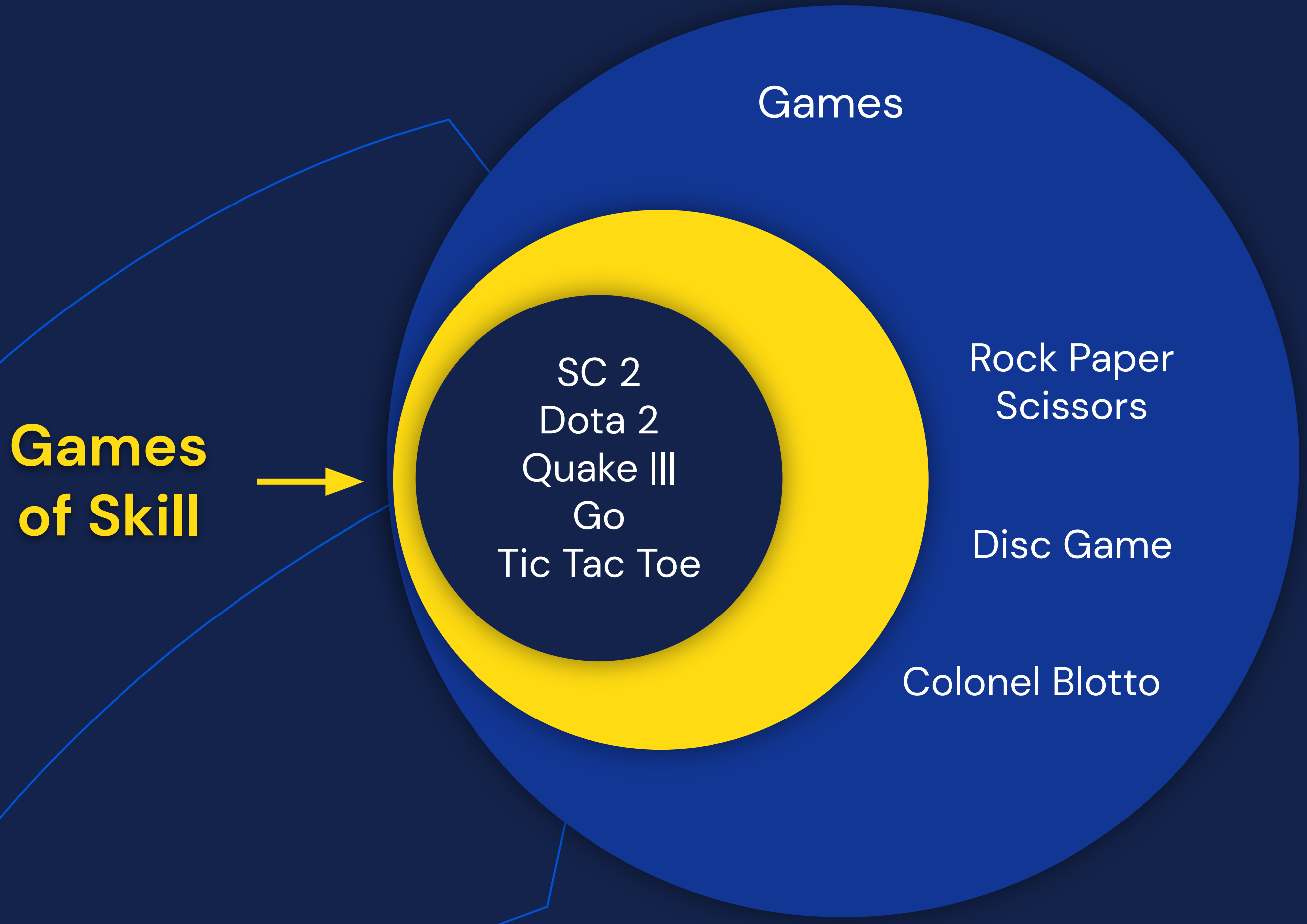
DeepMind

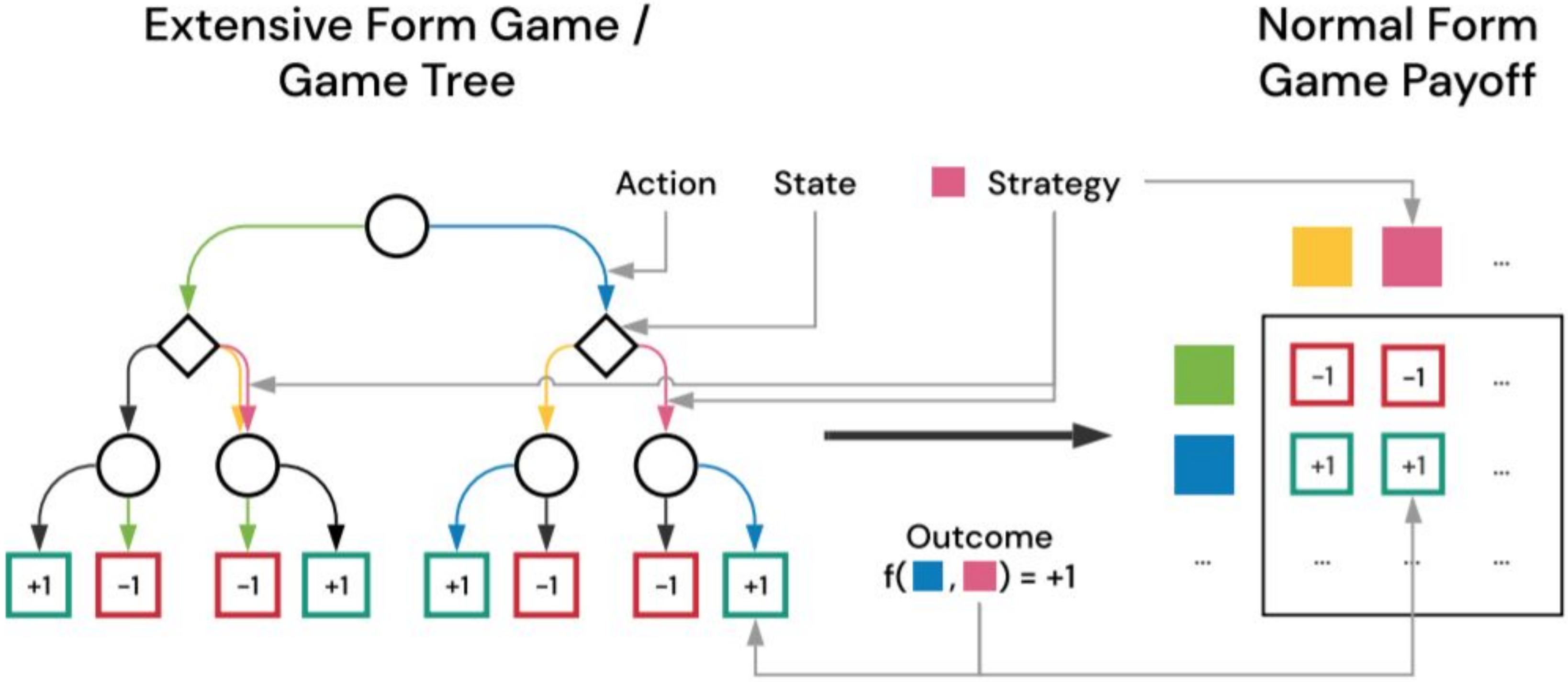
Real World Games look like Spinning Tops

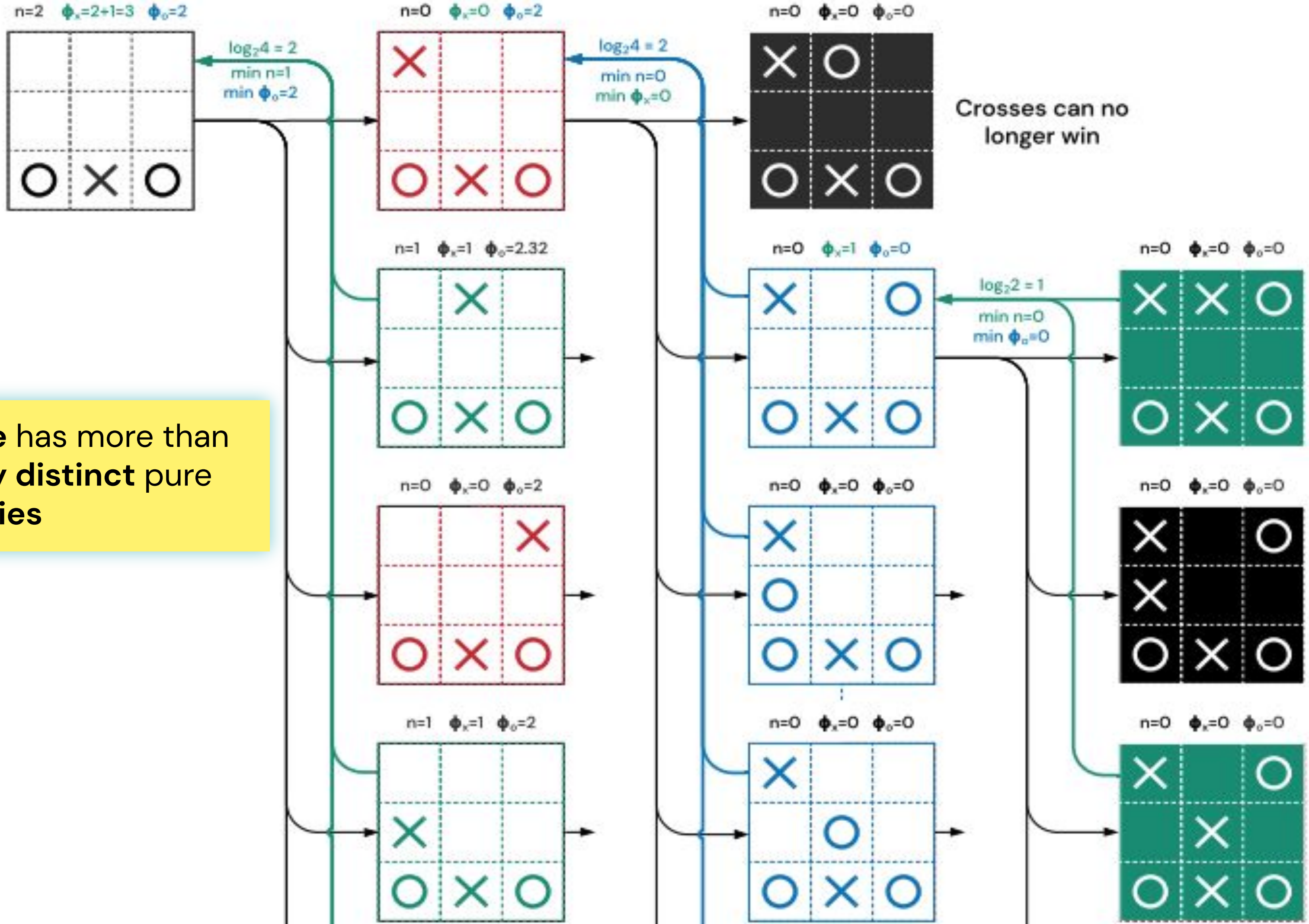
Real World Game

A competitive, two-player, symmetric zero-sum game, designed for human enjoyment, engagement and as a mean of challenging each others strategic thinking.

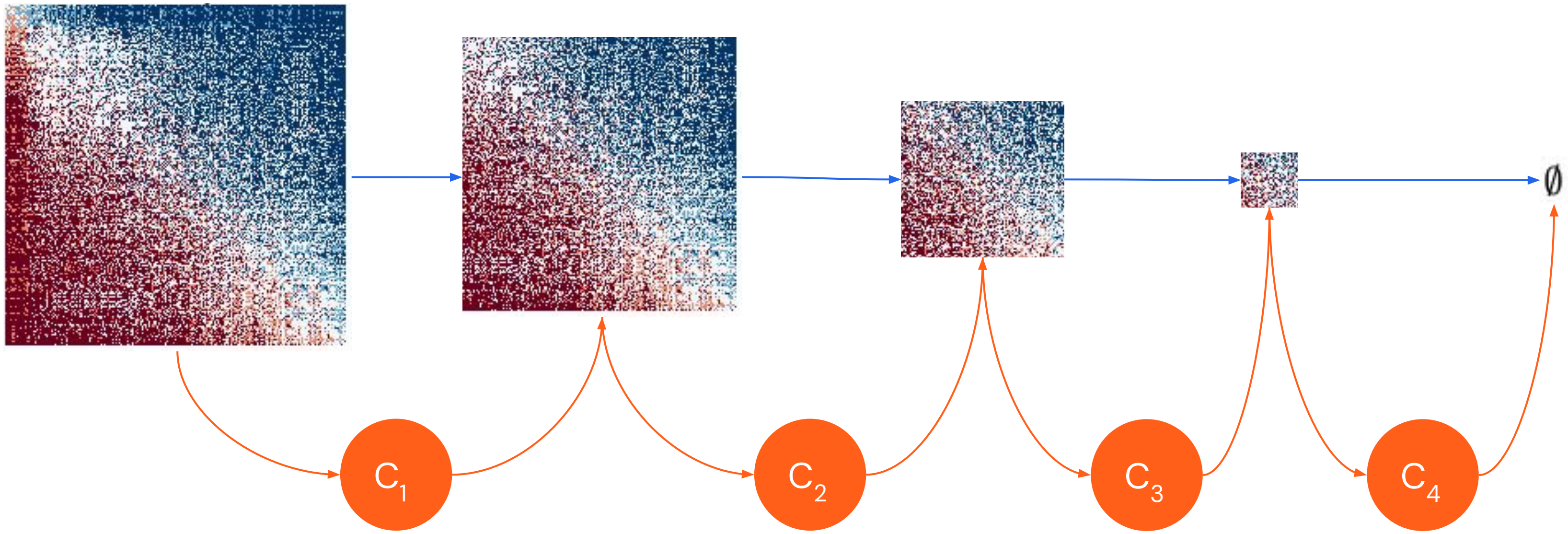

Games
Games of Skill
SC 2
Dota 2
Quake III
Go
Tic Tac Toe
Rock Paper Scissors
Disc Game
Colonel Blotto


Extensive Form Game / Game Tree
Normal Form Game Payoff
Action
State
Strategy
+1
-1
-1
+1
+1
-1
-1
+1
Outcome
f(,) = +1
...

Game of **Tic Tac Toe** has more than 10^{567} **behaviourally distinct** pure **strategies**

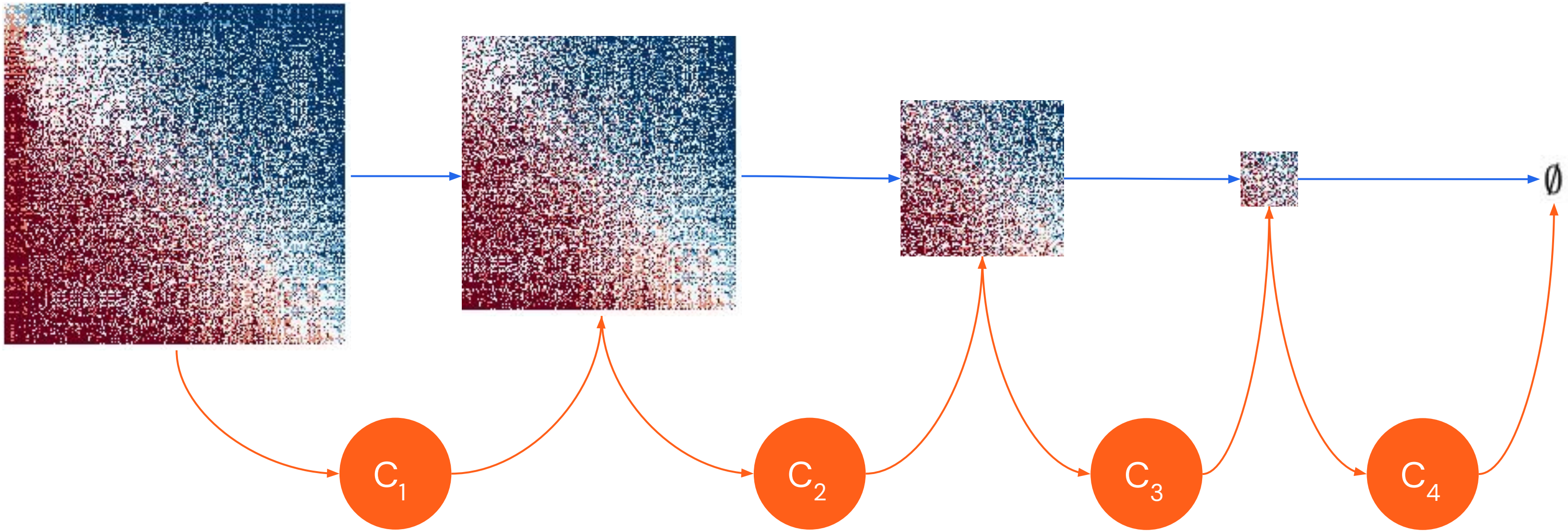


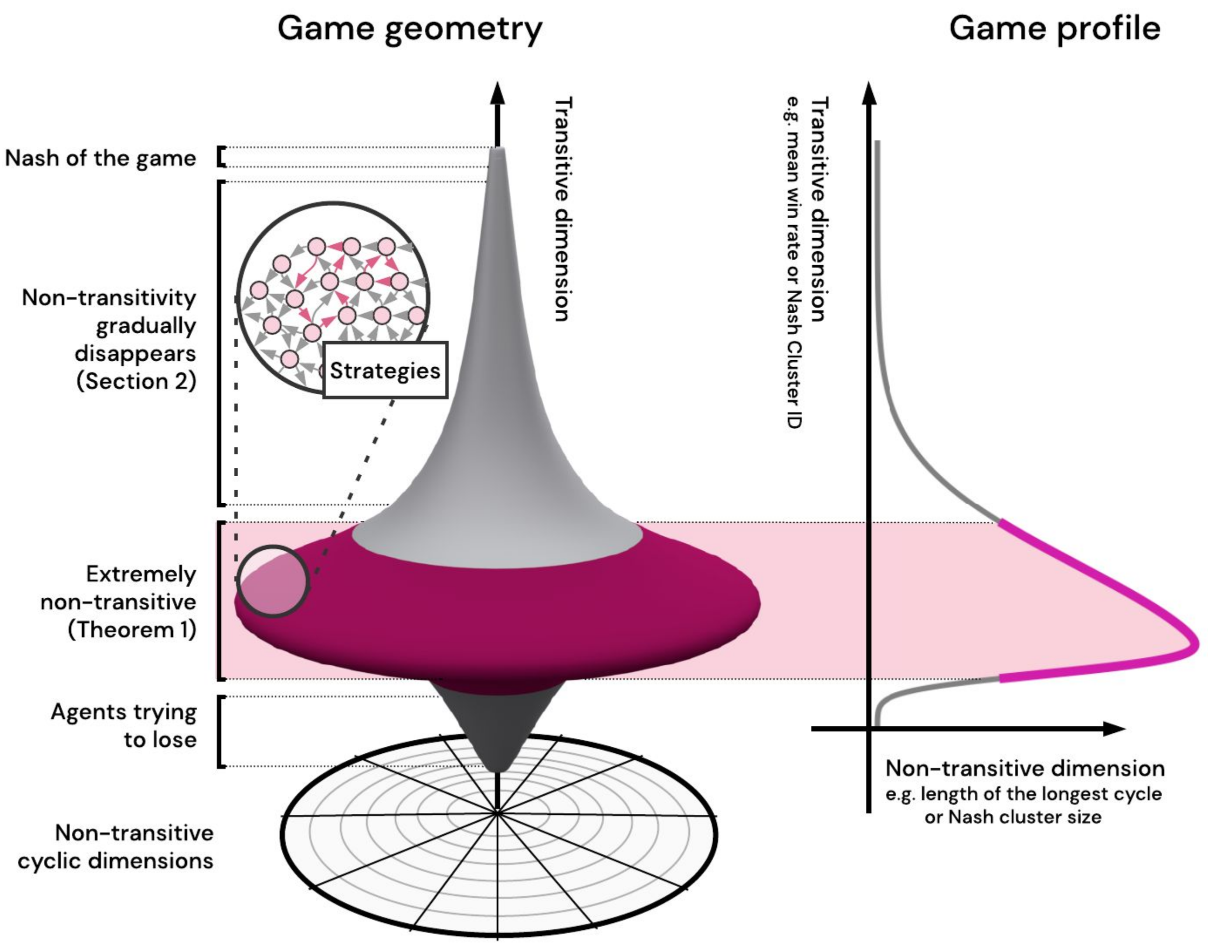
Definition 3. *Nash clustering* $\mathbf{C}$ *of the finite zero-sum symmetric game strategy* Π *set by setting for each* $i \geq 1$*:* $N_{i+1} = \mathrm{supp}(\mathrm{Nash}(\mathbf{P}|\Pi \setminus \bigcup_{j \leq i} N_j))$ *for* $N_0 = \emptyset$ *and* $\mathbf{C} = (N_j : j \in \mathbb{N} \wedge N_j \neq \emptyset)$.



Definition 3. *Nash clustering* $\mathbf{C}$ *of the finite zero-sum symmetric game strategy* Π *set by setting for each* $i \geq 1$*:* $N_{i+1} = \mathrm{supp}(\mathrm{Nash}(\mathbf{P}|\Pi \setminus \bigcup_{j \leq i} N_j))$ *for* $N_0 = \emptyset$ *and* $\mathbf{C} = (N_j : j \in \mathbb{N} \wedge N_j \neq \emptyset)$.

Theorem 2. *Nash clustering satisfies* $\mathrm{RPP}(\mathbf{C}_i, \mathbf{C}_j) \geq 0$ *for each* $j > i$.




Game geometry
Game profile
Nash of the game
Non-transitivity
gradually
disappears
(Section 2)
Extremely
non-transitive
(Theorem 1)
Agents trying
to lose
Non-transitive
cyclic dimensions
Strategies
Transitive dimension
Transitive dimension
e.g. mean win rate or Nash Cluster ID
Non-transitive dimension
e.g. length of the longest cycle
or Nash cluster size

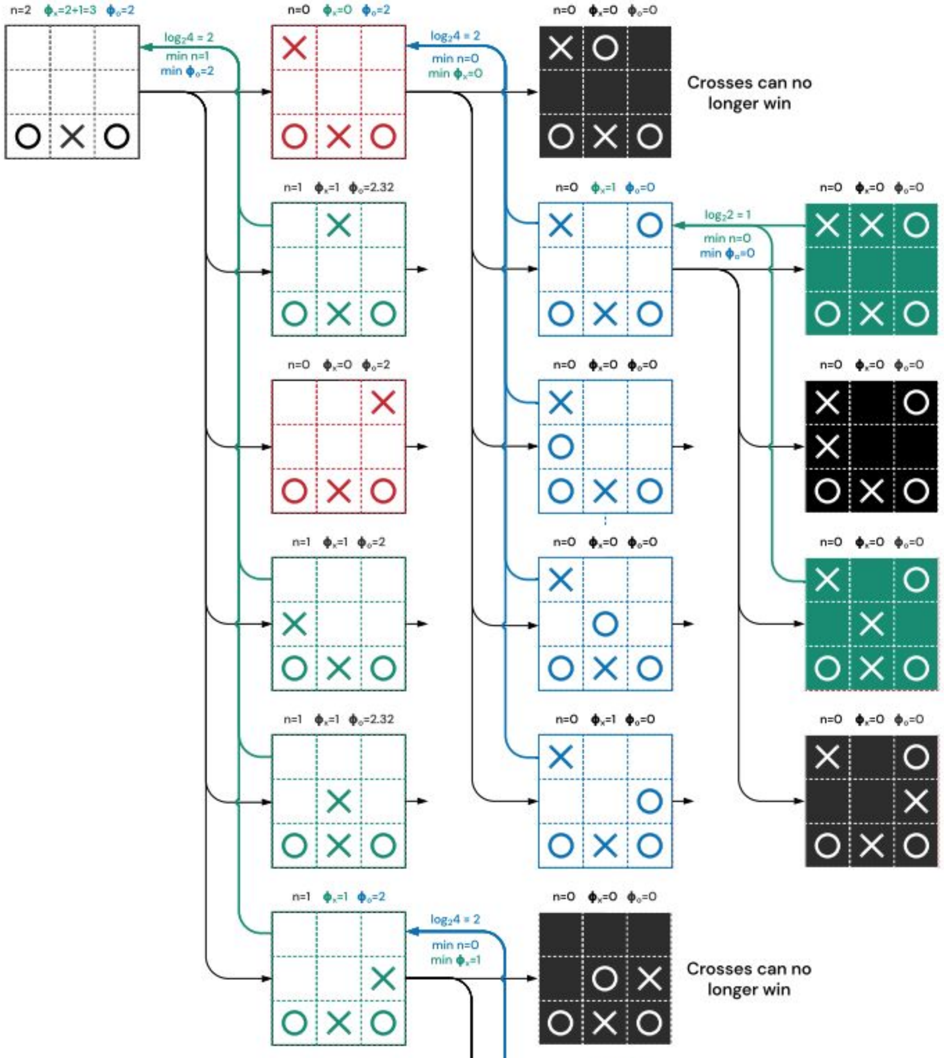
Empirical Verification

OpenSpiel [LINK]

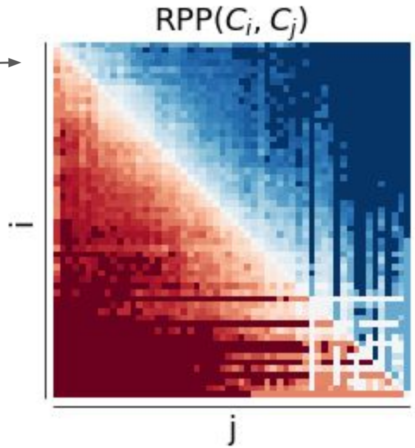
Game of **Tic Tac Toe** has more than **10^{567} behaviourally distinct** pure **strategies**

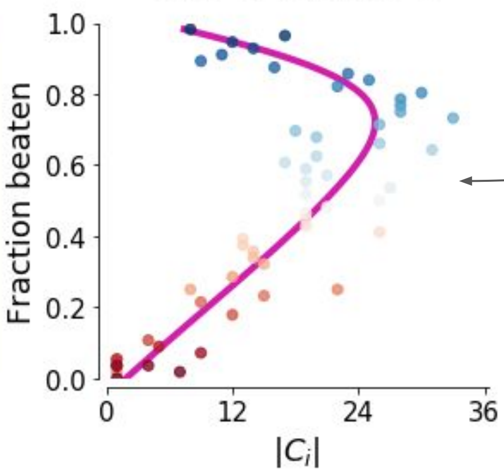
We rely on **empirical game theory** through sampling

An open question: can the analysis be done implicitly through the game tree traversal?

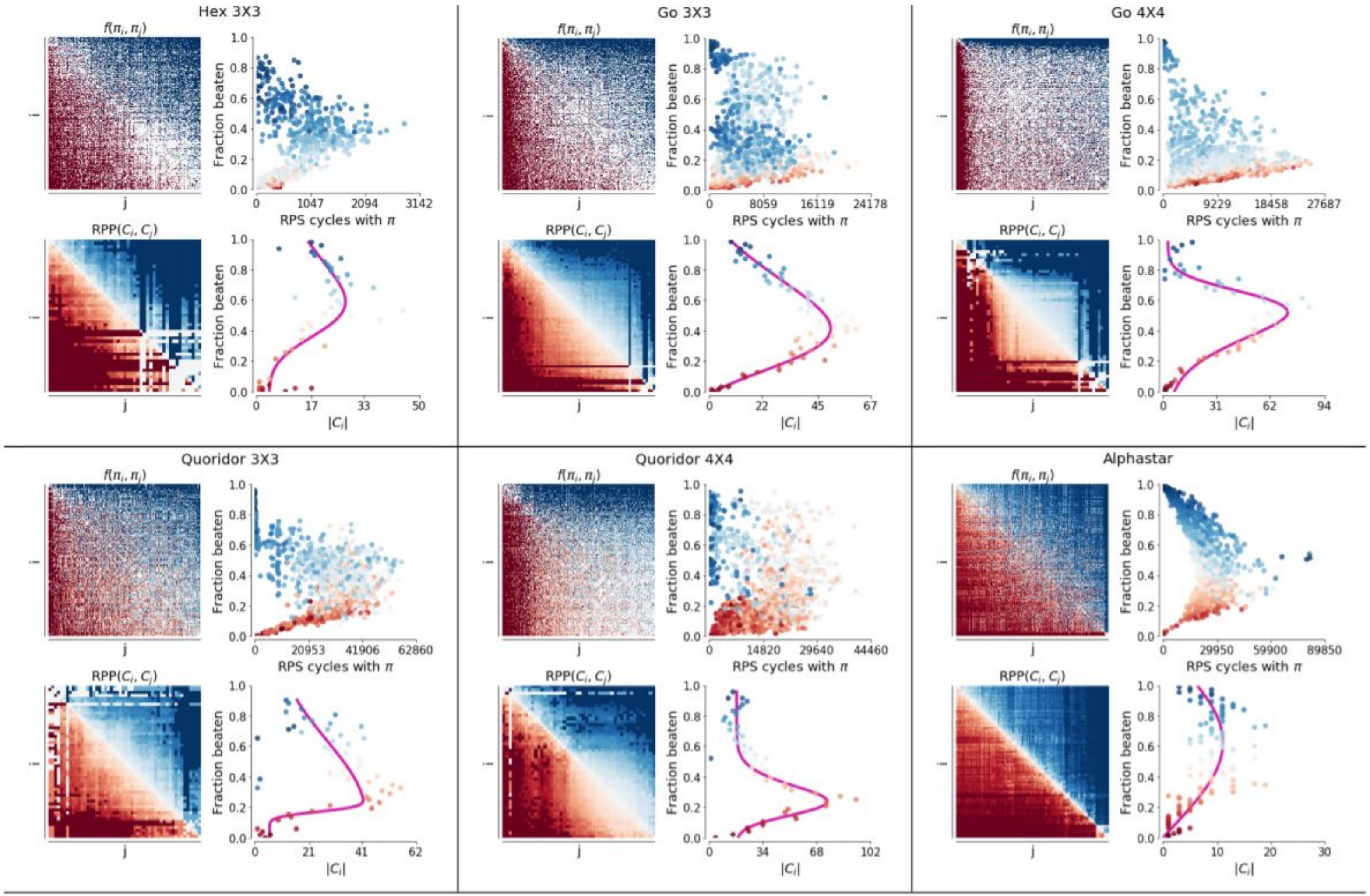


Nash clustering + RPP creates **transitive structure** (Theorem 2)





Sizes of Nash **clusters** denote “**non-transitivity**” at each level



Hex 3X3
Go 3X3
Go 4X4
Quoridor 3X3
Quoridor 4X4
Alphastar
$f(\pi_i, \pi_j)$
$RPP(C_i, C_j)$
i
j
Fraction beaten
RPS cycles with π
$|C_i|$
0 1047 2094 3142
0 8059 16119 24178
0 9229 18458 27687
0 20953 41906 62860
0 14820 29640 44460
0 29950 59900 89850
0 17 33 50
0 22 45 67
0 31 62 94
0 21 41 62
0 34 68 102
0 10 20 30

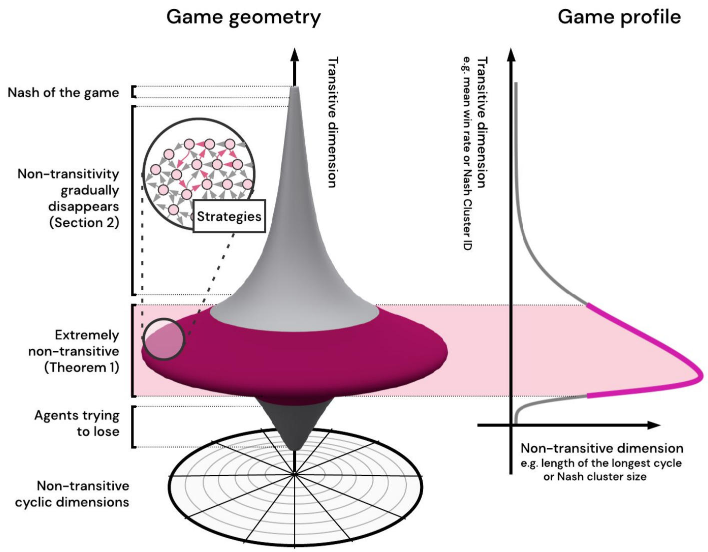
Conclusion:

Empirical and Theoretical evidence that in **real world game:**

- Huge number of strategies.
- But tiny number of **Good** strategies
- Spinning top shape.

  (The worst you get the more

  strategies there is)



Thank you !