GDC | March 21-25, 2022
San Francisco, CA

Age of Empires IV: Machine Learning Trials and Tribulations

#GDC22

Andrea Schiel – anschiel@microsoft.com Lead AI Architect (Worlds Edge Studio) She/Her
Peter Chan – peter.chan@relic.com Senior AI Developer (Relic) He/Him
Guy Leroy - t-gleroy@microsoft.com Associate ML Researcher (MSR) He/Him
Sam Devlin – sam.devlin@microsoft.com Principal ML Researcher (MSR) He/Him




WORLD'S EDGE
relic ENTERTAINMENT
Microsoft Research Cambridge

GDC

March 21-25, 2022
San Francisco, CA

Phil Wardlaw – Senior AI Developer – Combat Fitness
Matt Burgi – Automation Engineer – Combat Fitness
Jaroslaw Rzepecki – Senior Research Engineer - MSR
Dave Bignell – Research Engineer – MSR

Phil Wardlaw

Matt Burgi

Dave Bignell

Jaroslaw Rzepecki

Age of Empires IV AI Team

Darren Ward – lead
Jasbir Roopra – producer
Byron Chow – designer
Wayne Chen
Liz Gordon
Puya Dadgar
Andrea Schiel

Also:

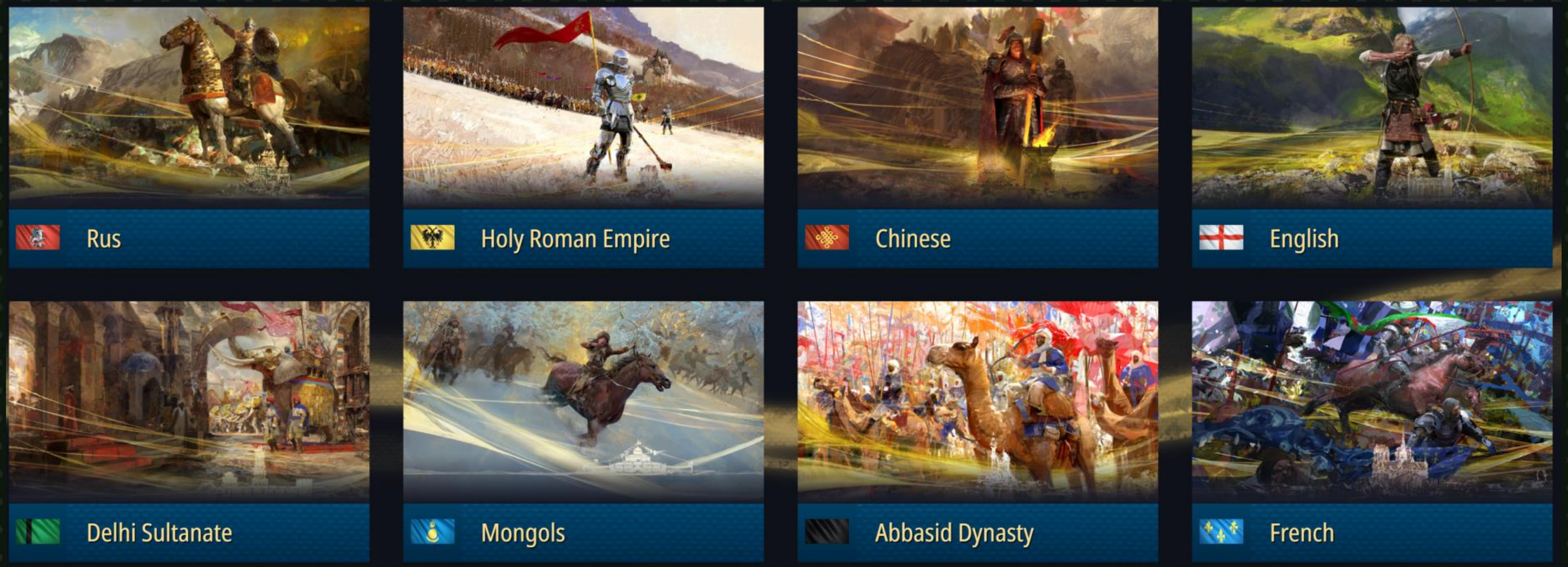
Diccon Yamanaka
Peter Chan
Phil Wardlaw
Warren Johnson

Wayne C

Andrea S

Problem Space

- 8 civilizations
- 380+ units and structures
- 130+ upgrades



Problem Space

- castles & walls
- siege mechanics
- 2 wonders per age
- 4 ages
- 3 victory types

Problem Space

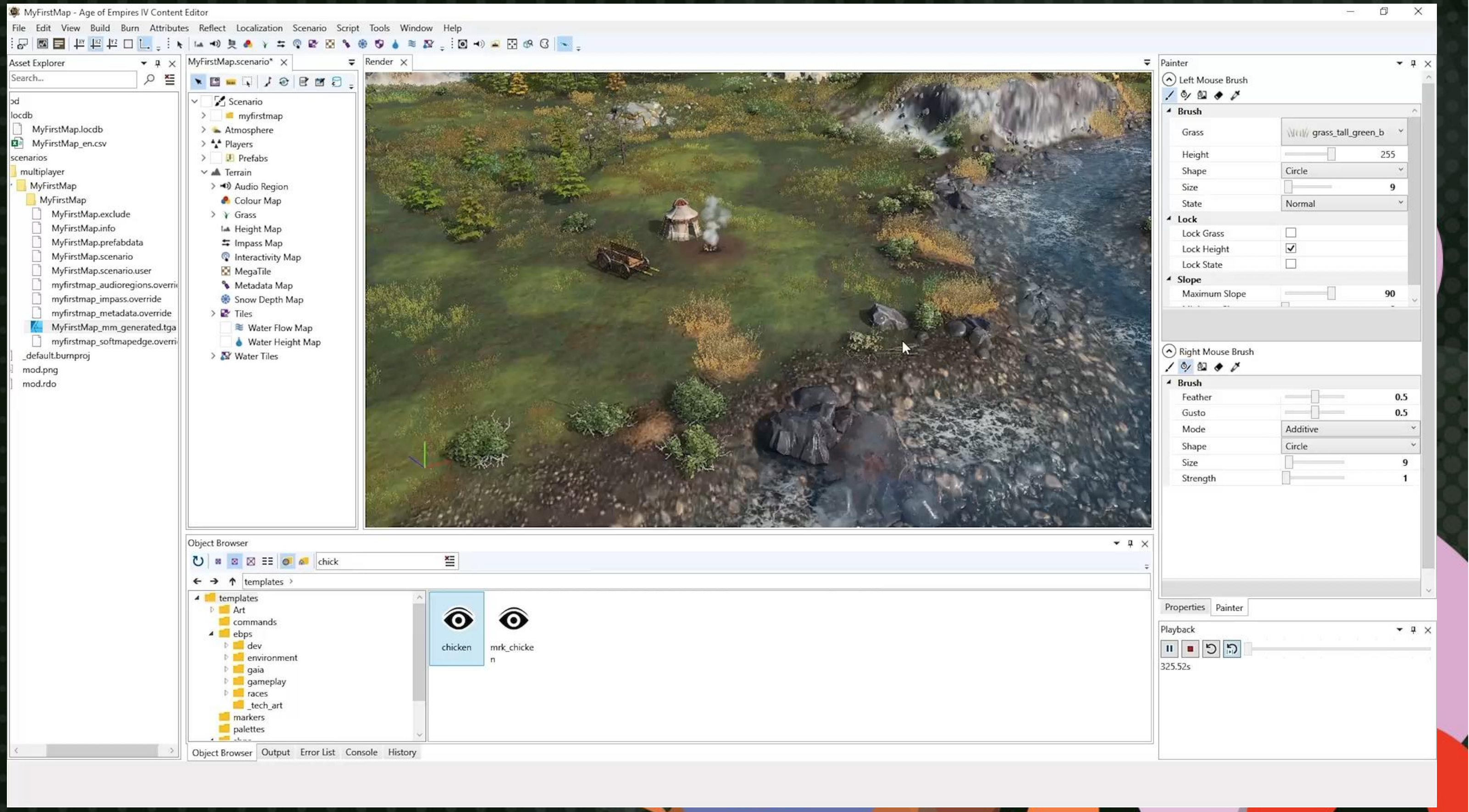
- naval combat
- 2 terrain types
- 4 key resources

Machine Learning

2 examples of DNNs (deep neural networks)
Different forms of supervised training:
- combat fitness uses labelled data
- navigation & combat uses RL (reinforcement learning)

Goals for DNN

- Modular/targeted approach – training to be done on a few machines, small compute
- Goal 1: determine what makes a good DNN/DRL problem
- Goal 2: determine how to train a model or policy during live game development
- Goal 3: performant at runtime (inference)
- Goal 4: fun not superhuman behavior



GDC | March 21-25, 2022
San Francisco, CA

Using Supervised Learning for Combat Fitness

#GDC22

Combat Fitness Agenda

- Problem Space
- Why Supervised Learning?
- Prototype
- Observations and Improvements
- Results and Takeaways

Combat Fitness Definition

```
float ComputeCF(const Army& teamA, const Army& teamB);
```

- Given two armies, should we fight or flight?
- Heuristic for many decisions
- Usually require supplements

1.0 = dominate
0.5 = even
0.0 = total lost

Combat Fitness Usage Examples

- Whether units should engage in combat
  - Should we initiate a fight?
  - When to fallback or retreat?
  - How much reinforcement to bring in?
- Utility calculation for unit production
- What upgrades to purchase

Combat Fitness Classic Approach

- Explicit formula to simulate damage model
- Requires data introspection
- Things to consider:
  - Unit health & Army size
  - Weapon attributes (range, AoE, etc)
  - Armor types
  - Upgrades
  - RNG…

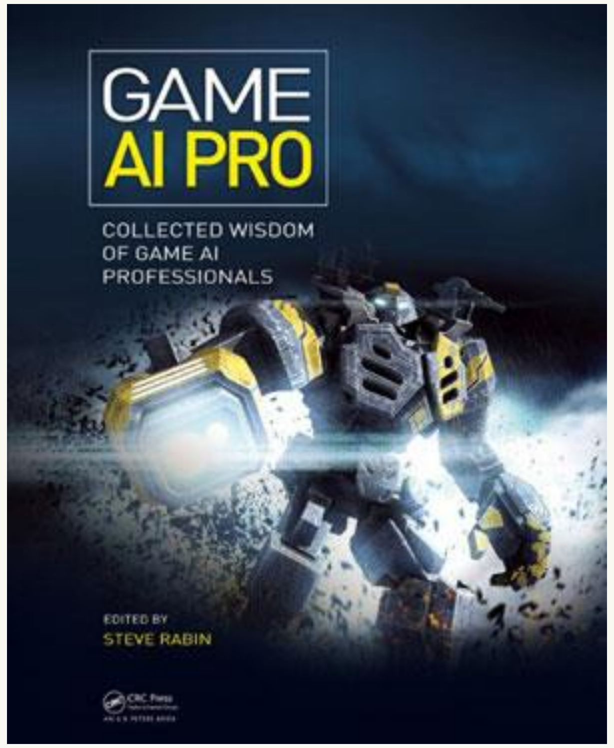
Combat Fitness Challenges

- Called a lot so it needs to be fast
- Hard to test, hard to maintain
- May require introspecting a lot of data during runtime
- Effectiveness of a unit may not be obvious from data
- Combinatorial explosion
  - 8 civilizations
  - 380+ units and structures
  - 130+ upgrades

Why Supervised Learning?

- We have a teacher (the game!)
- No need to handle any complexity during combat
- DNN model trained offline and can be automated
- Runtime inference is cheap
- It's been done before

Using Neural Networks to Control Agent Threat Response
By Michael Robbins



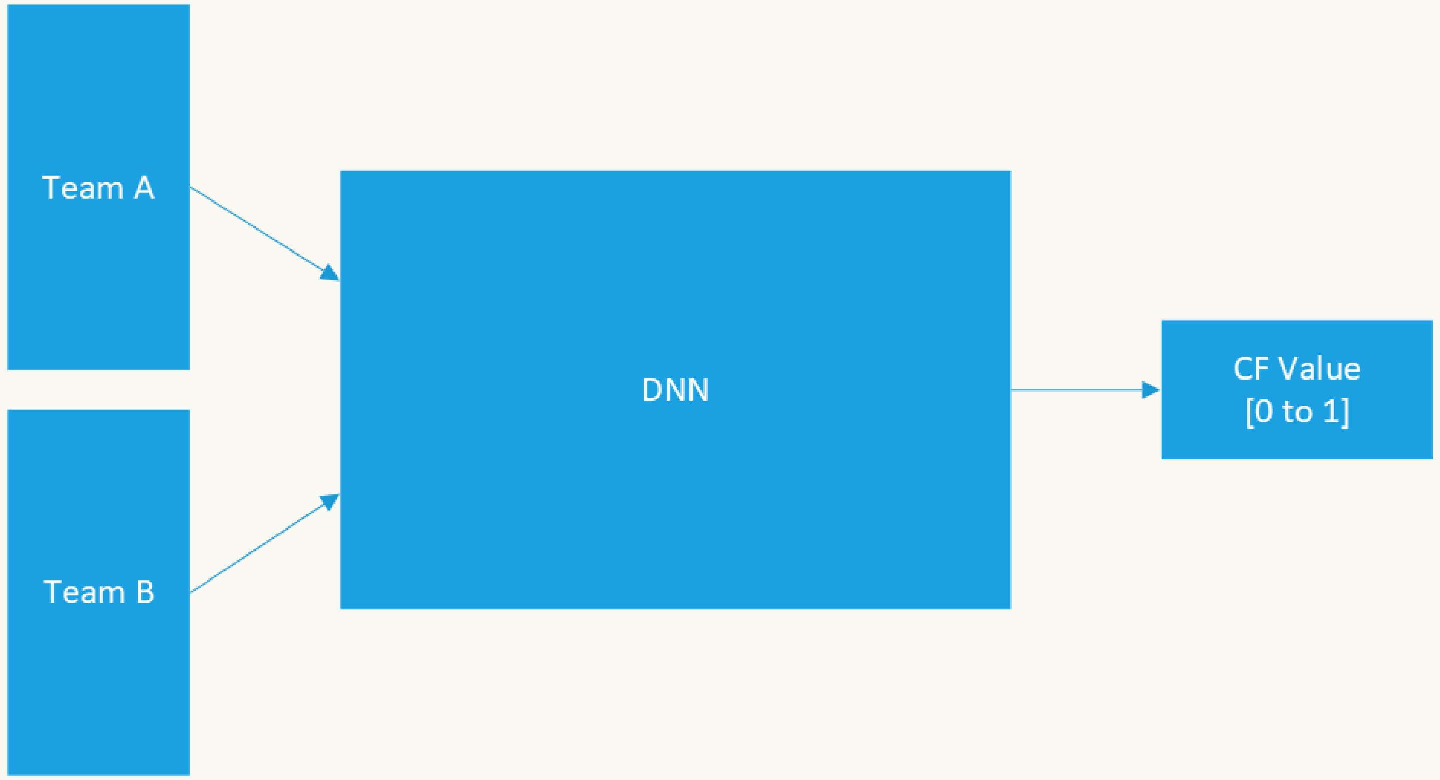
Prototype

- Setup a test scenario to generate fight data
  - Randomize unit type and count
  - Record initial and final health
- Experimented with different input features
- How well does it generalize?

Initial Model

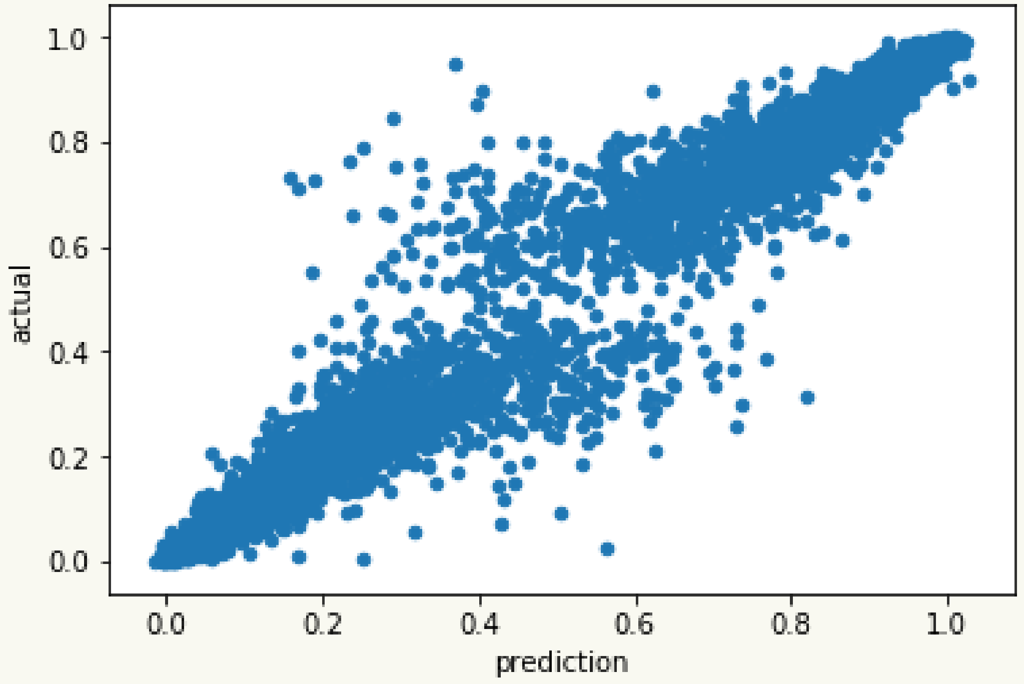
- Extract values summarizing English infantry units
- Dataset took a couple hours to generate

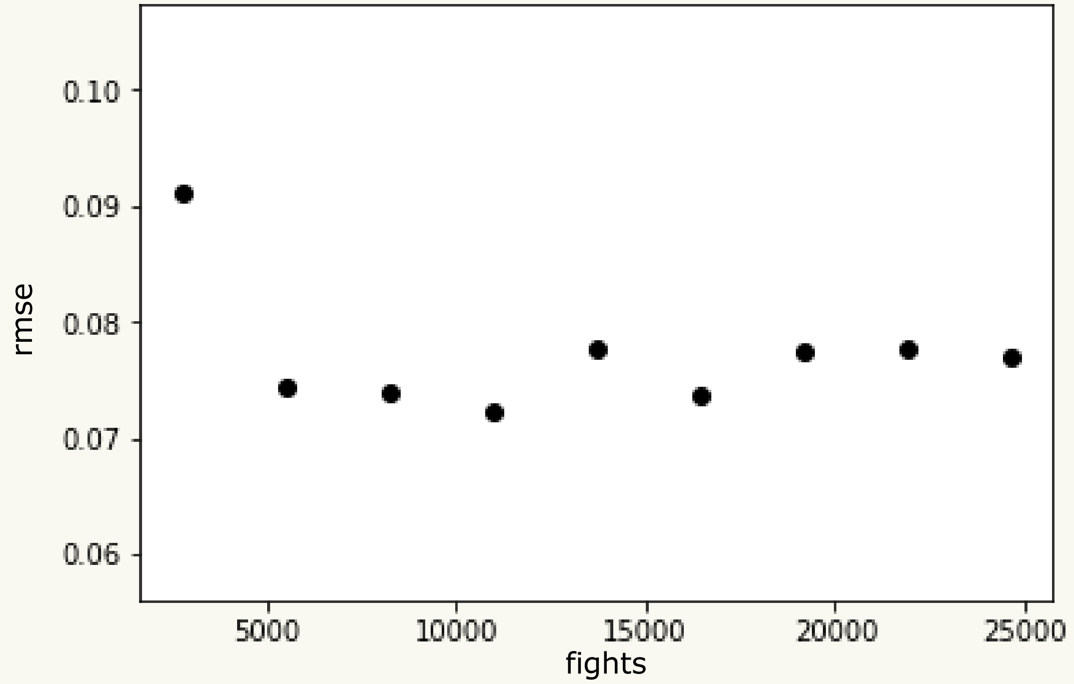
- Unit count
- Initial health
- Average move speed
- Total weapon damage
- Penetration
- Armor
- Firerate
- Etc



Initial Model – Not bad

- Model gave reasonable results
- Accuracy improves with more fight data





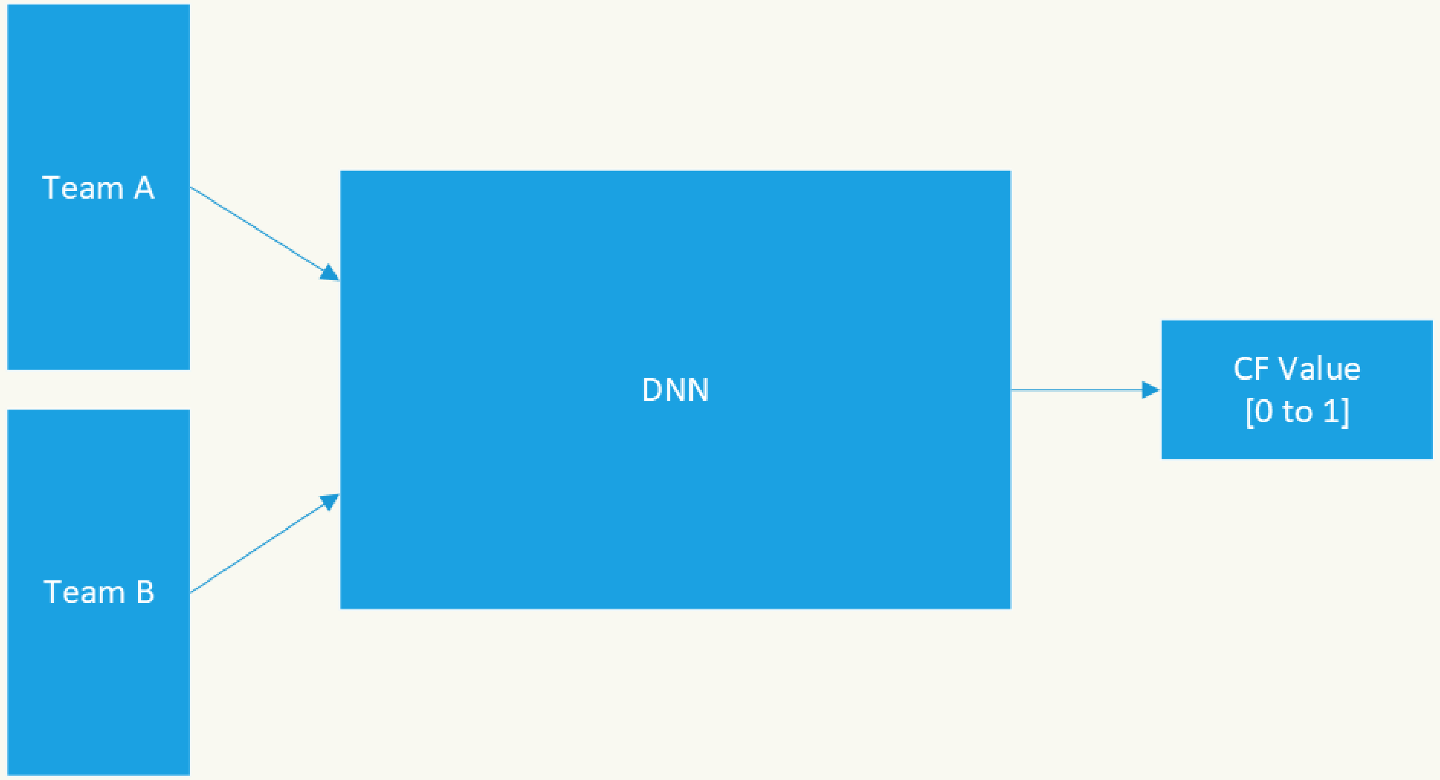
Initial Model – Limitations

- More work is required to consider other unit types
- Feature selection is tricky

Raw Unit Combinations

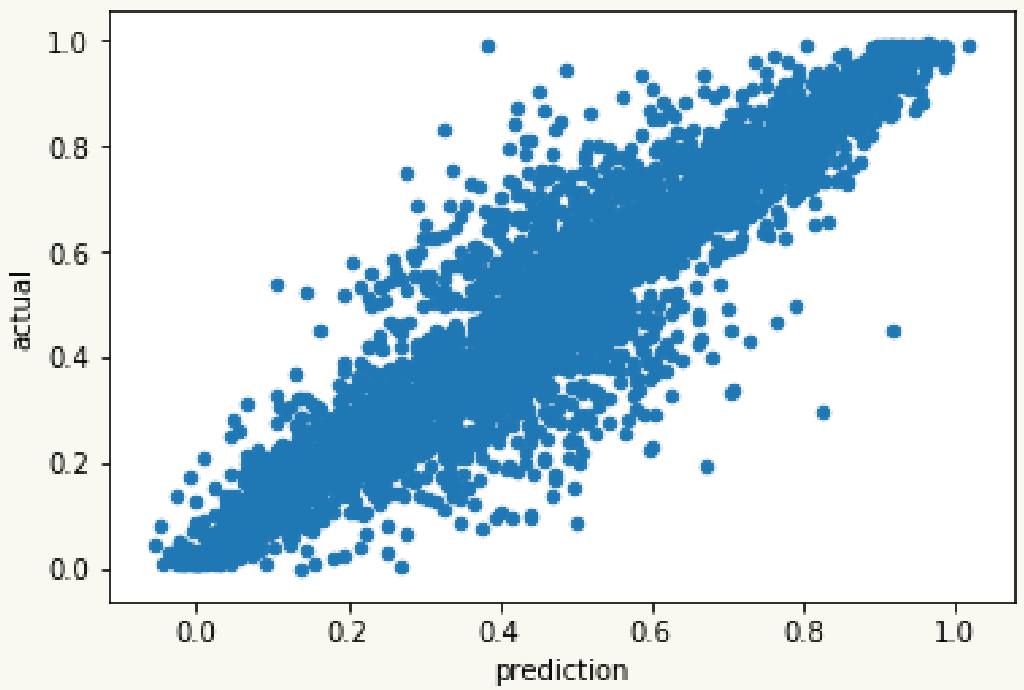
- Much simpler, can characterize all combat types

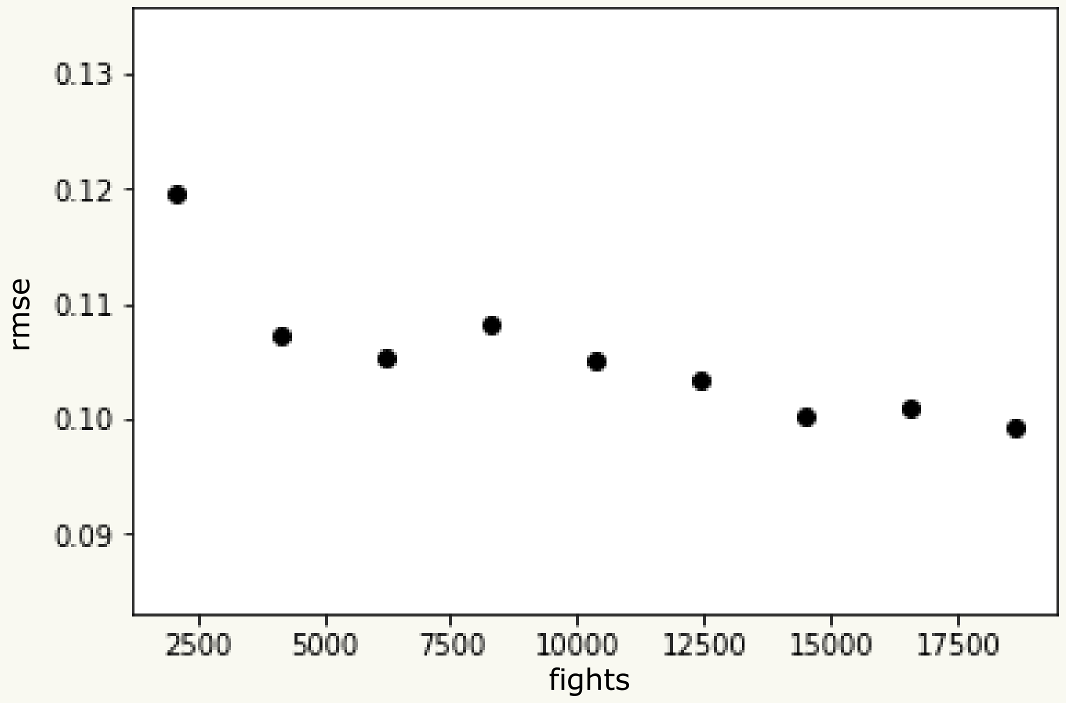
- `Number of units`
- `Initial health`
- `# unit type A`
- `# unit type B`
- `# unit type C`
- `...`



Raw Unit Combinations

- Requires a lot more training data to improve accuracy
- Took days to generate the fight data





What does it mean?

- Needs to train with all civs and combat types
- Potentially an infeasible large problem space to generate training data

Reduction of Features

- Can we reduce the feature dimension?
  - Faster training time
  - Faster prediction queries at runtime

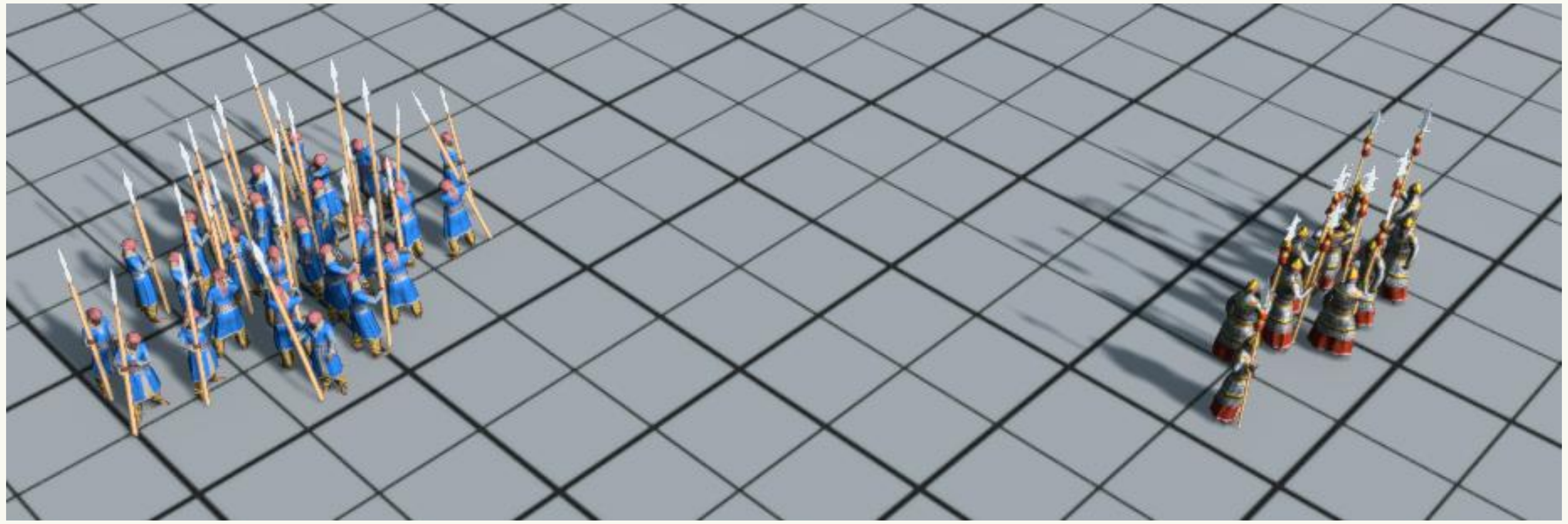
Yes we can!

Using Archetypes - Idea

- Group units with same combat mechanics
- Use the weakest unit in the group as the base unit with a score of 1
- Determine the relative strength ratio for each member to the base unit

Using Archetypes - Example

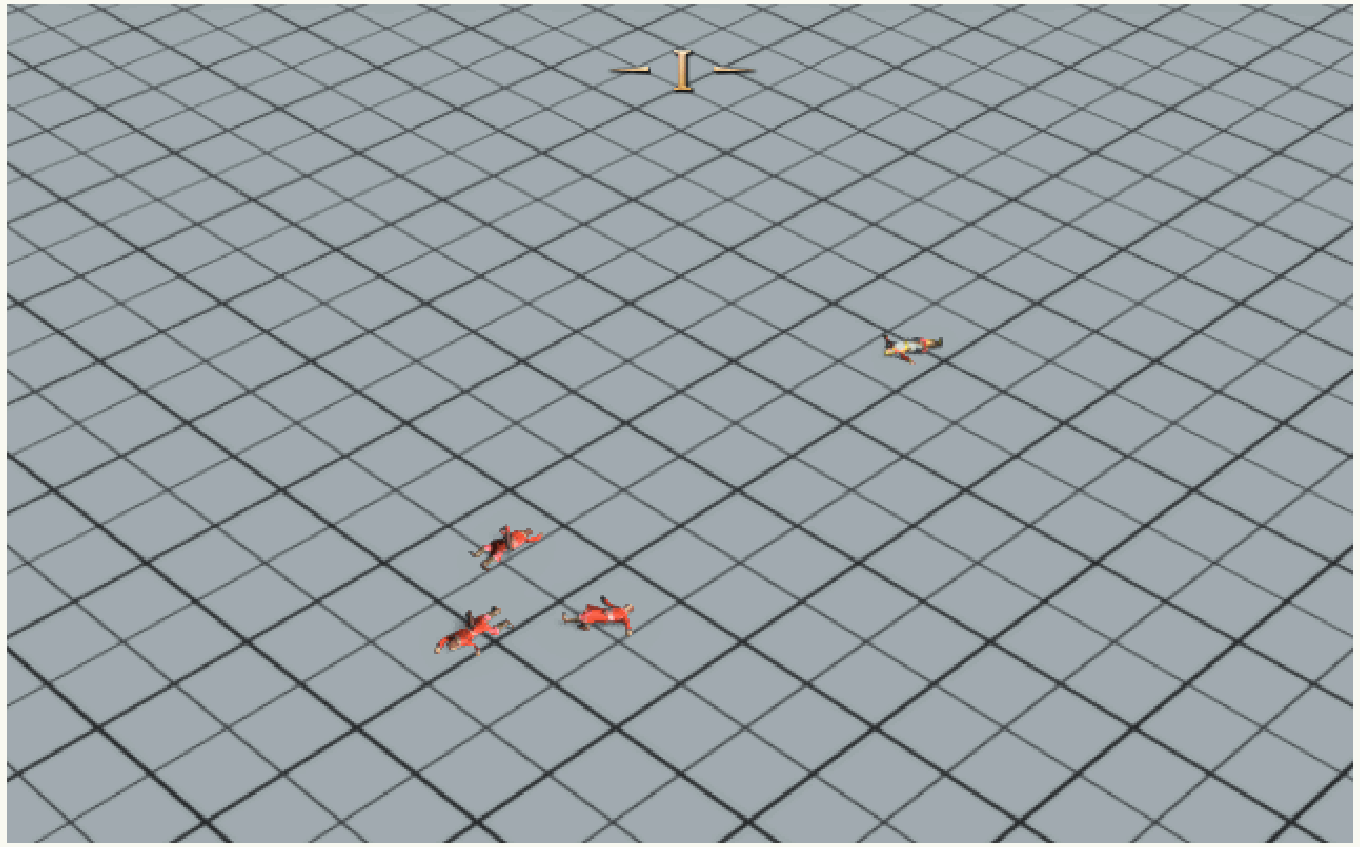
- 10 Imperial Age spearmen
- 27 Dark Age spearmen
- Imperial Age spearman score = 27 / 10 = 2.7

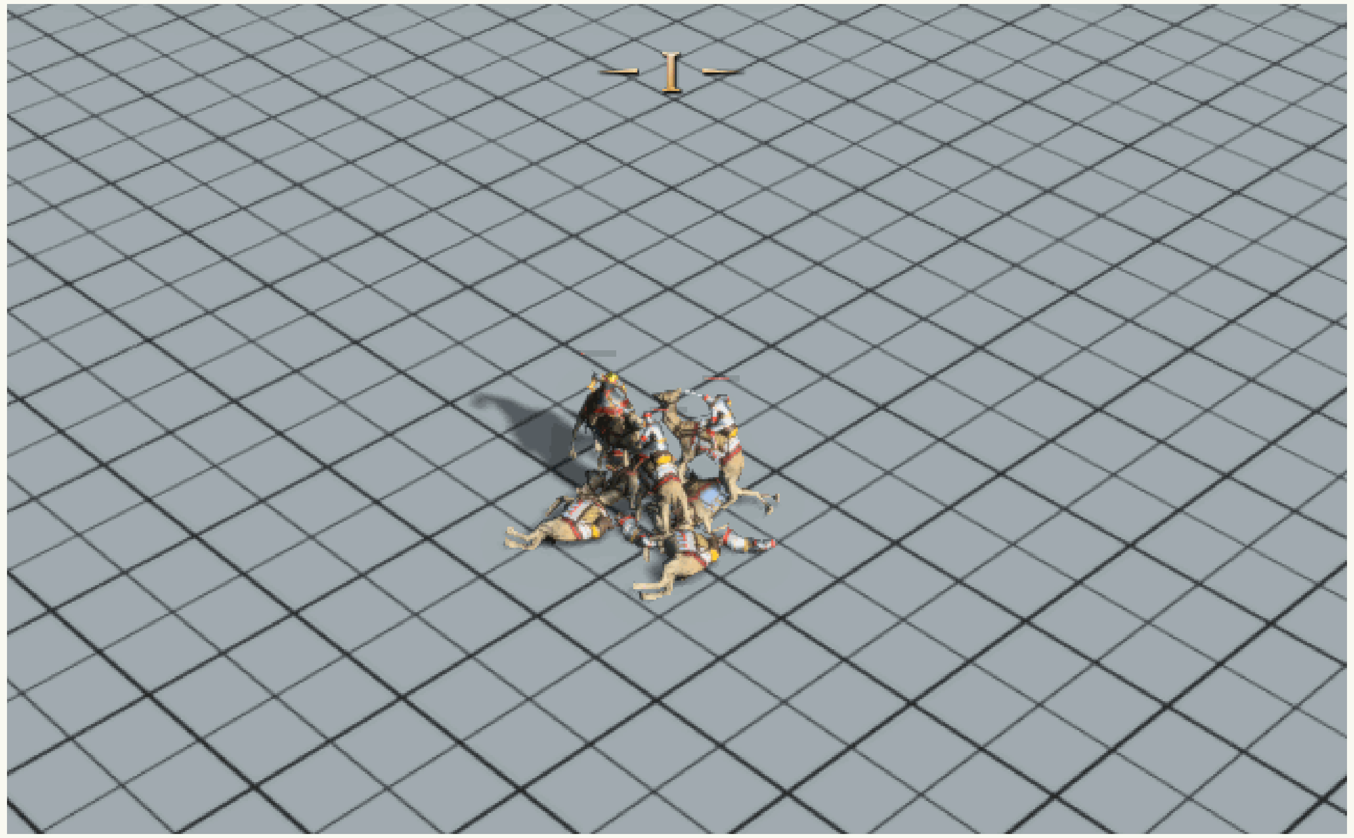


Ages
- Dark Age
- Feudal Age
- Castle Age
- Imperial Age

Archetype Training

- The process is automated





Using Member Scores

old model

unit_spearman_1_eng	unit_spearman_2_eng	unit_spearman_3_eng
1	5	3

new model

spearman
14.7

(1 x 1) + (5 x 1.45) + (3 x 2.15) = 14.7

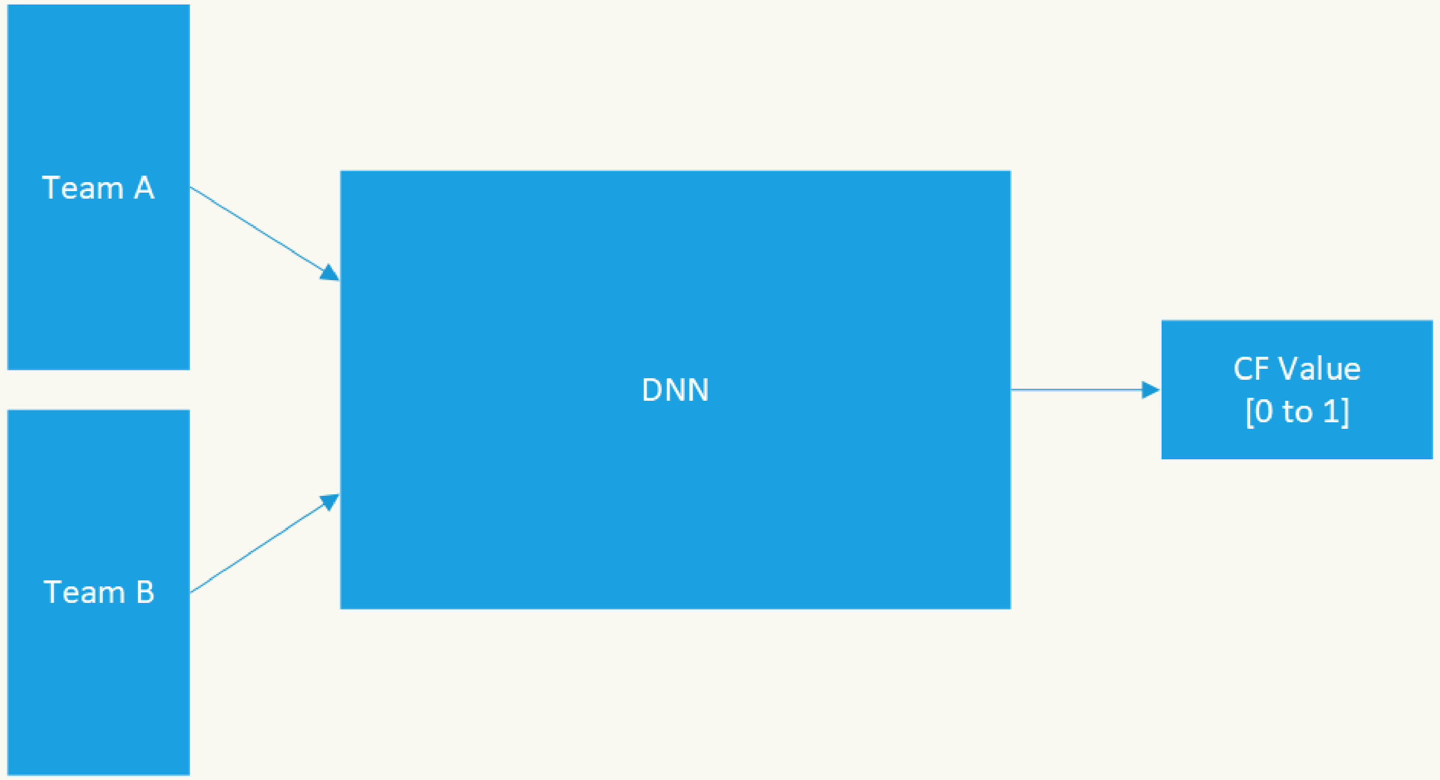
```
'Archetypes' = {
  'spearman':
  {
    'unit_spearman_1_eng': 1.0,
    'unit_spearman_2_eng': 1.45,
    'unit_spearman_3_eng': 2.15,
    'unit_spearman_4_eng': 2.7,
  },
  …
}
```

Combat Fitness Model

- Archetypes for infantry, siege units, naval
- Also have combat buildings and healers

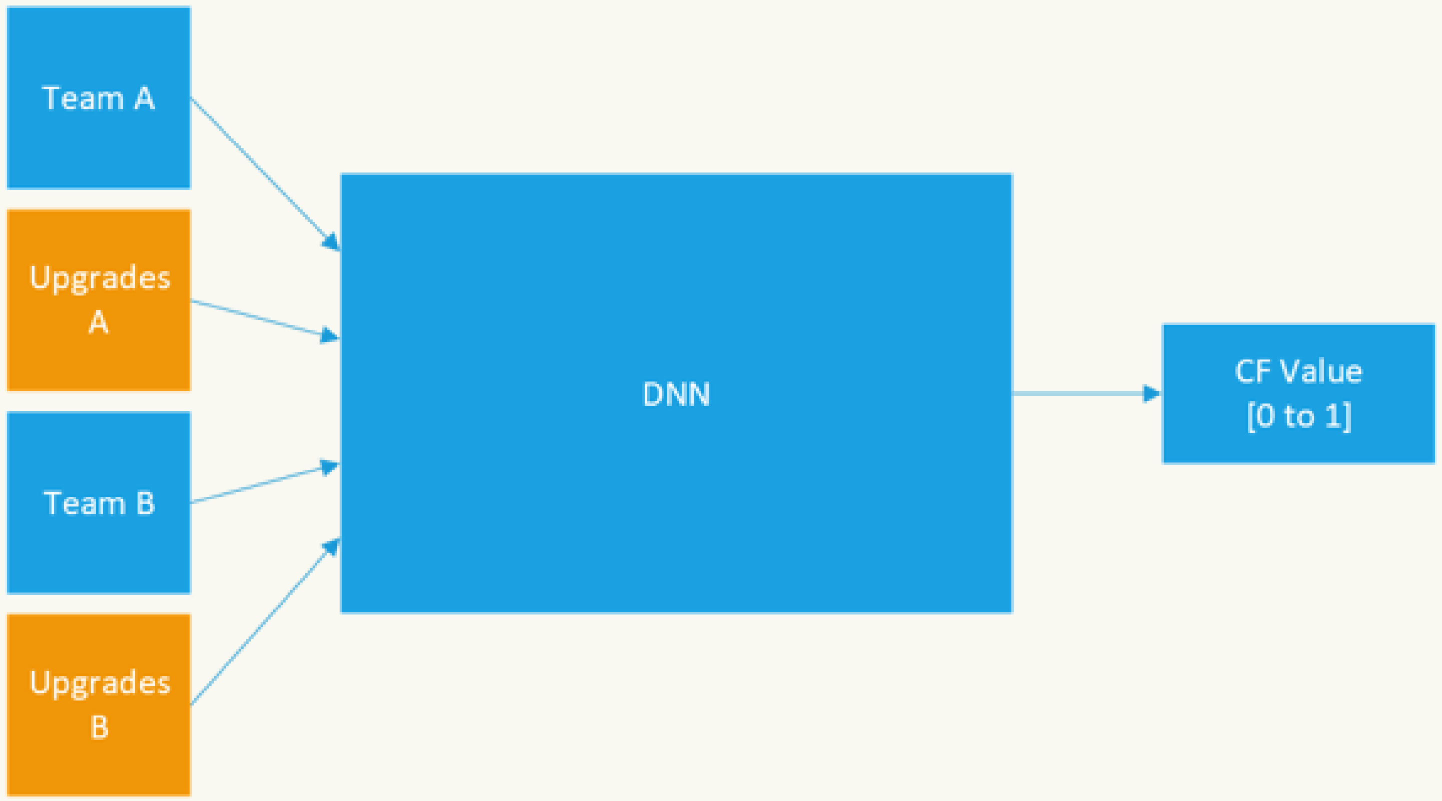
- Archetype score
- Health percent
- Unit count

(x41 Archetypes)



Layering in Upgrades

- How do upgrades affect combat?
- Add new input columns for each upgrade (0 or 1)
- Update fight data generation script to randomly add upgrades



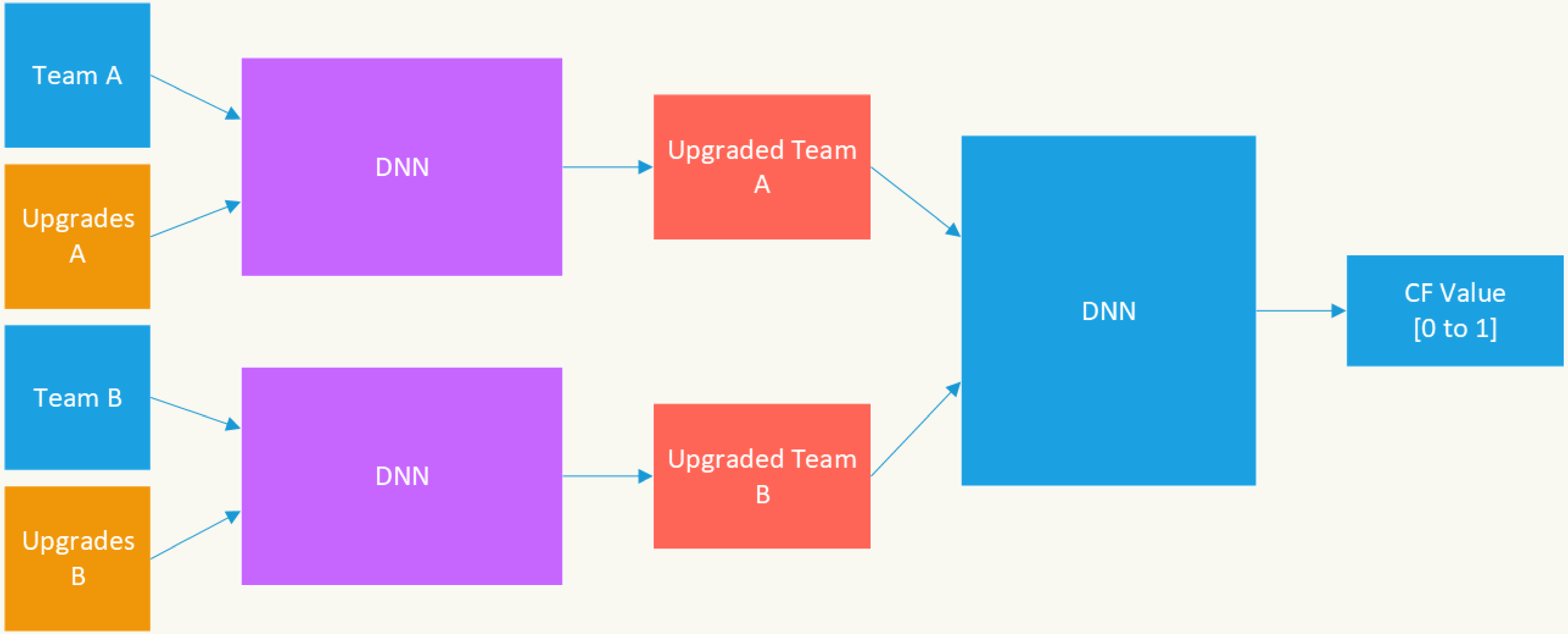
Upgrades - Problem

- Some upgrades improve units so subtly that random variations in combat overshadow their effectiveness
- With a fully connected layer, the model can associate improvements due to the presence of an upgrade to unrelated units

Upgrades - Solution

- Supply our own custom layer based on prior knowledge
- We know from game data what units each upgrade can affect
- Train a separate model to learn the effectiveness of an upgrade on certain units, ignoring unrelated ones

Upgrades - Solution



We have a winner

- We now have a solution for combat fitness
- How do we bring it to production?

Automation Goals

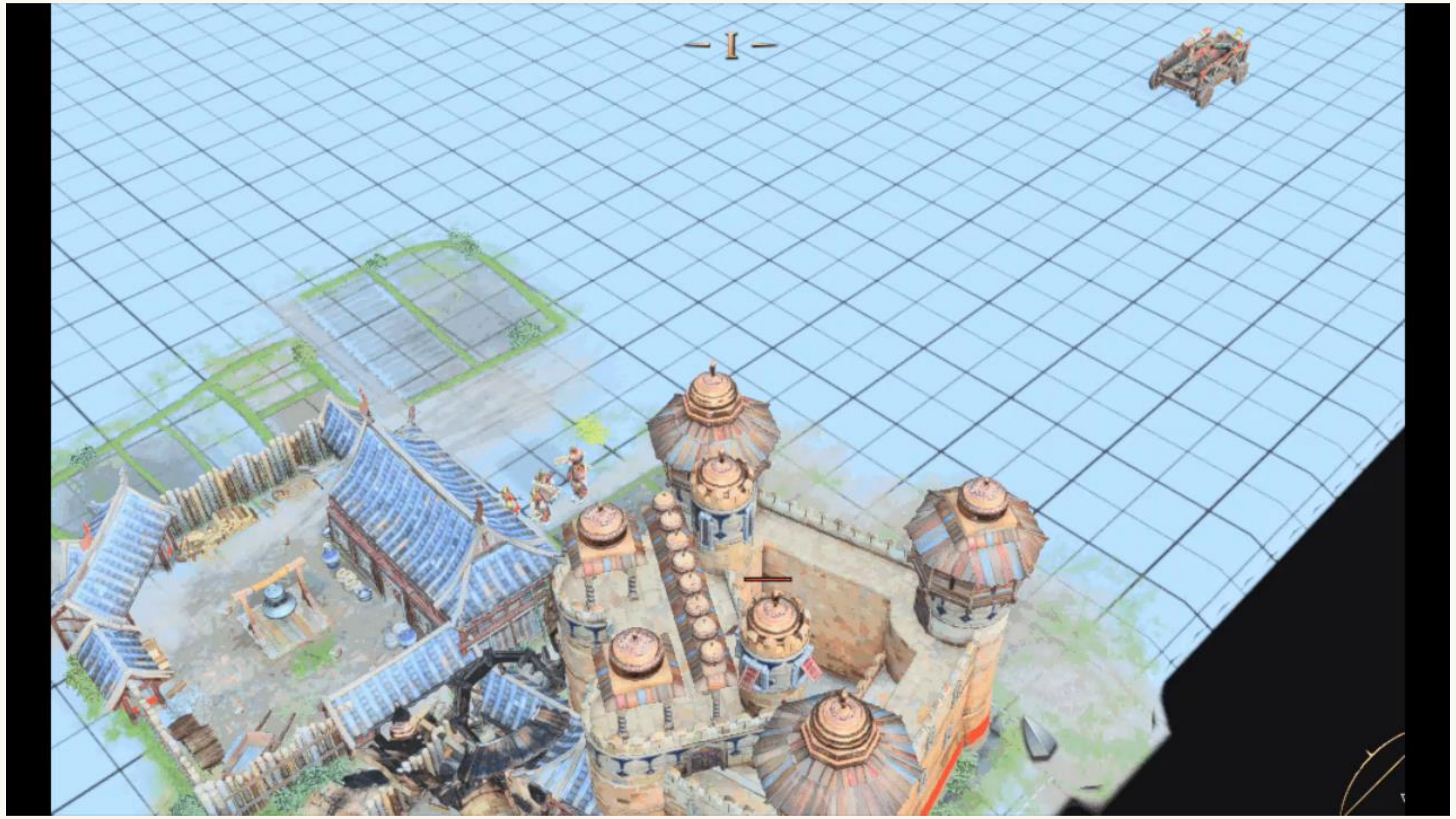
- On-going development and design changes and balancing require model update
- Two parts:
  - Archetype training automation
  - Combat data generation
- However, model training is still manual

Archetype Automation Settings

- Run archetype training in parallel
- Split up large archetypes into subgroups
- Take a couple hours to complete

Combat Automation Settings

- Unit count from 1v1 up to 40v40
- Can be single unit type or mixed
- Land, naval, structures
- With and without upgrades
- ~200000 fights in 8 hours



Troubleshooting Problems

- Single black box
- What happens when model is inaccurate?
- Model training is still manual
  - Spot check scenarios, can patch data and experiment
  - Data distribution (individual units vs archetypes)
  - Blind spots
- Lots of tests in place

Implementation Notes

- Used TensorFlow and Python/Jupyter Notebook
- Some hyperparameter tuning
- SavedModel converted to .tflite format
- Used TensorFlow Lite for the runtime (x4 speed improvements)

Results and Takeaways

- Successfully used SL for combat fitness
  - Improved runtime
  - Adaptive to ongoing changes
- Not quite fully automated
  - Problems need to be investigated manually
- Monitor everything
  - Data generation
  - Model accuracy
- Just a heuristic
  - Not always accurate
  - Requires supplements or safeguards

RL exploratory projects: Agenda

- Tribulation: Optimizing farm building
- Trial: Plausible naval battles
- Integration and engineering efforts

Core Reinforcement Learning Loop

What layer to override

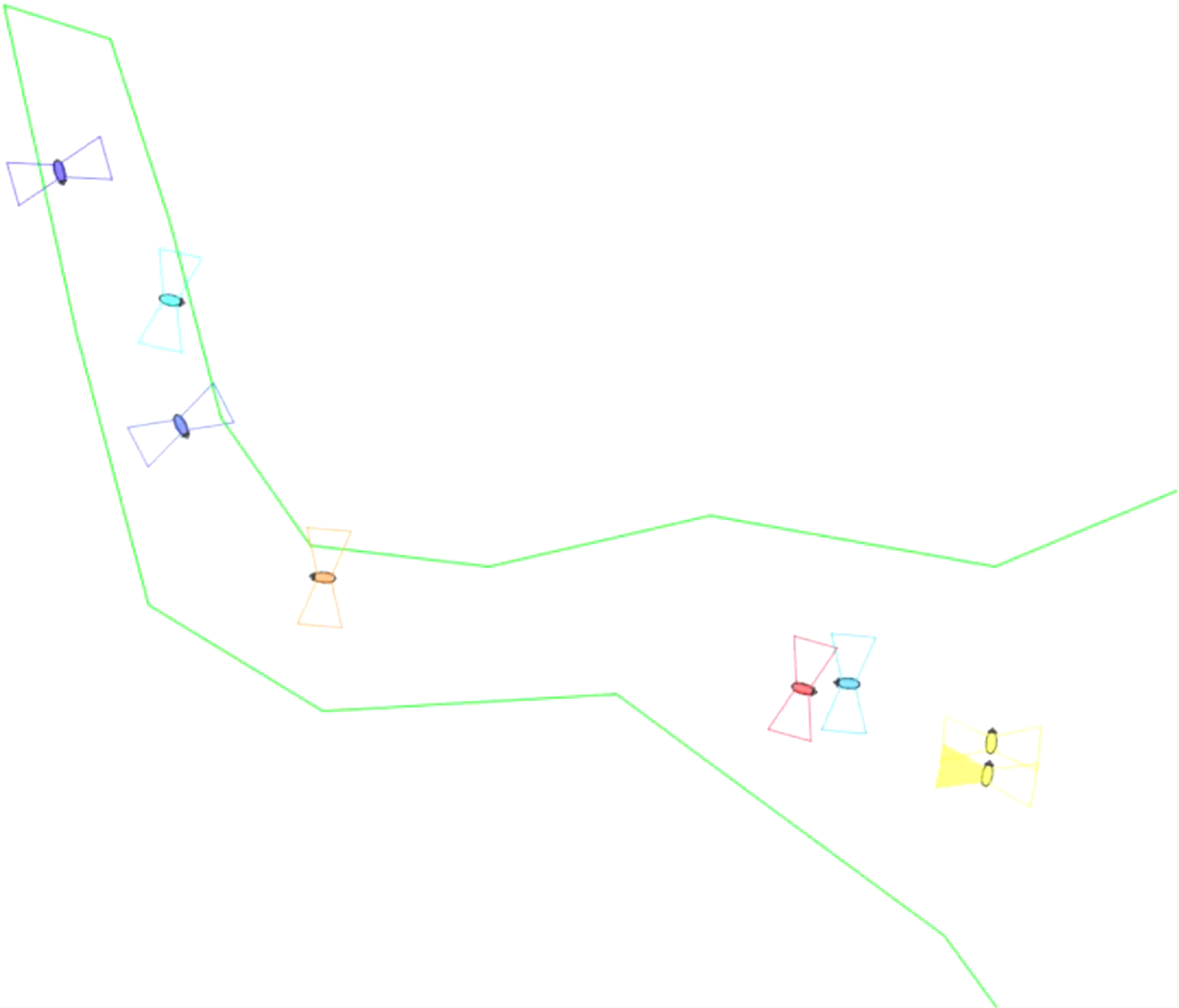
Engine-level actions

Human actions

High-level strategy & playstyle

Tribulation: Optimizing farm building

Prototype environment



RL ships

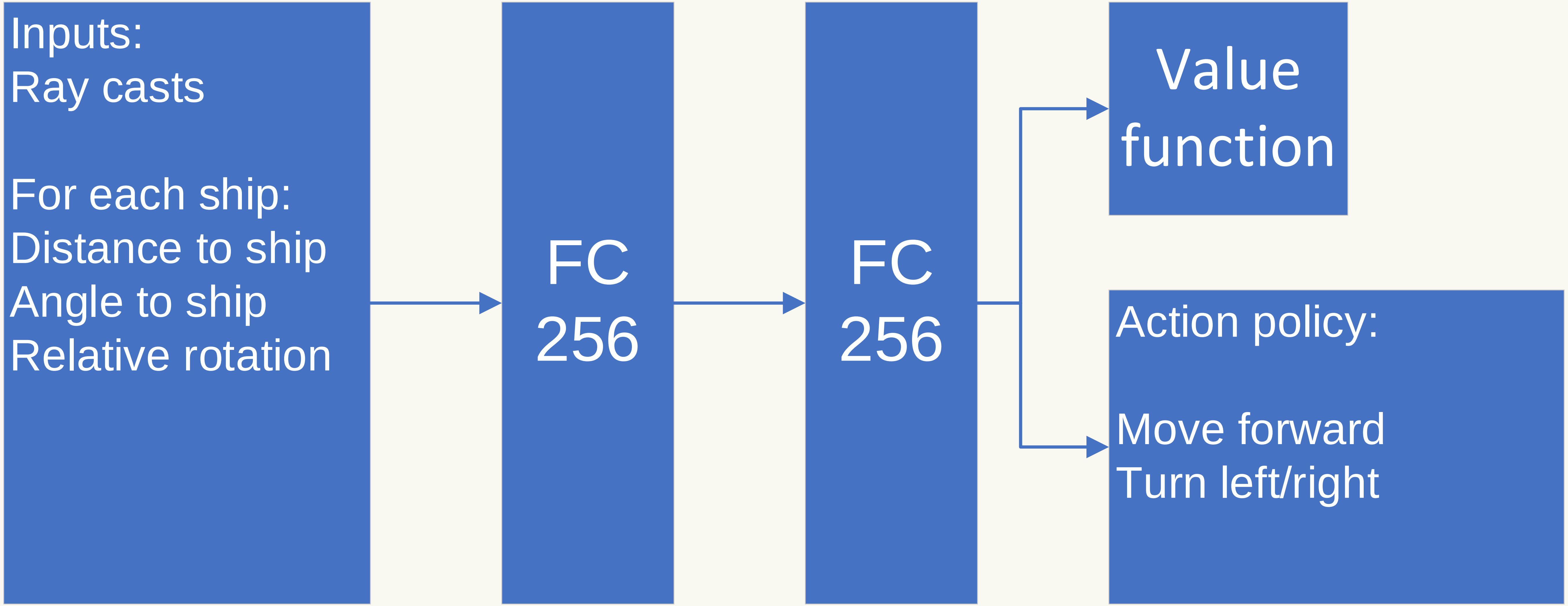
Scripted ships

Early training setup

- Start simple
  - Fully connected network
  - Proximal Policy Optimization with RLlib
  - Training on local machine

https://docs.ray.io/en/latest/rllib-algorithms.html#ppo

Model architecture for navigation



TANH activation

https://docs.ray.io/en/latest/rllib-models.html#default-model-config-settings

Trial: Human-like ship pathfinding

Built-in AI

Trial: Human-like ship pathfinding

RL ships

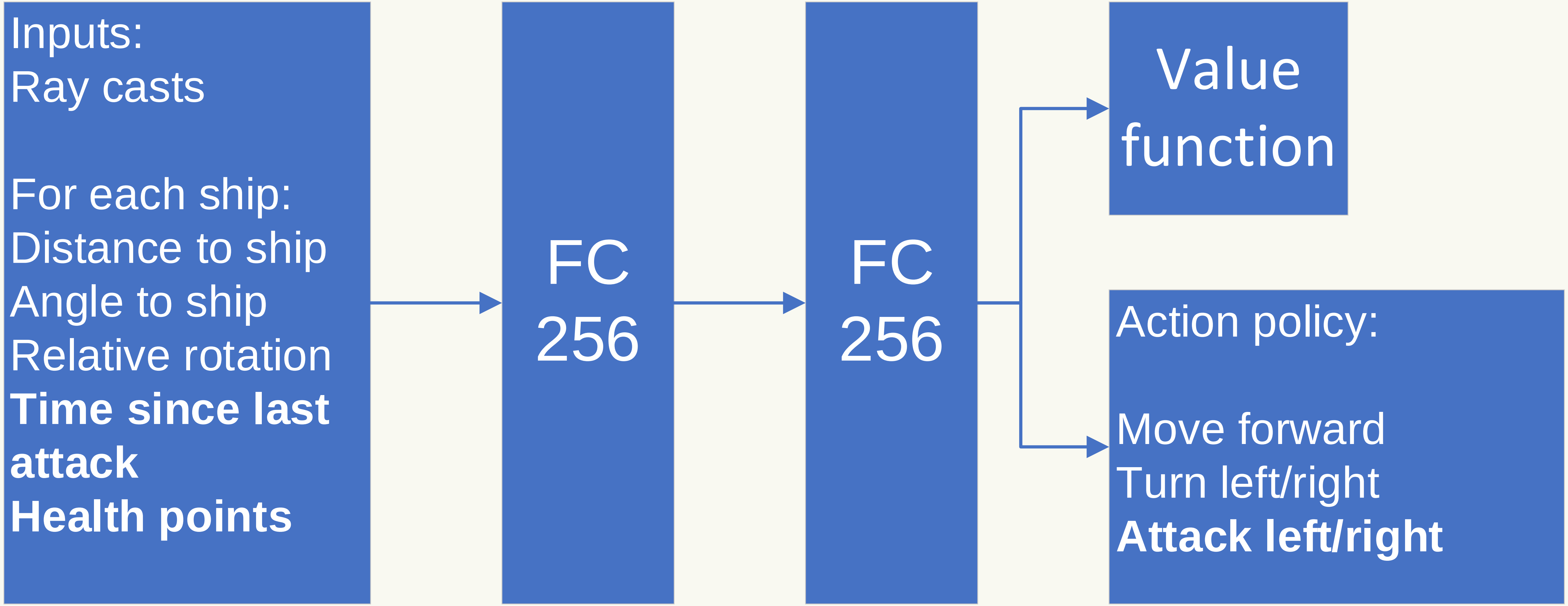
How to train faster & cheaper?

- Targeted model for narrow application with high impact
- Potential bottlenecks
  - Neural network training
  - Game samples collection

Speed up game sample collection

- Scale set of VMs to run the game (spot instances)
- Speed game up
  - Headless mode
  - Speed up task resets

Model architecture for combat



TANH activation

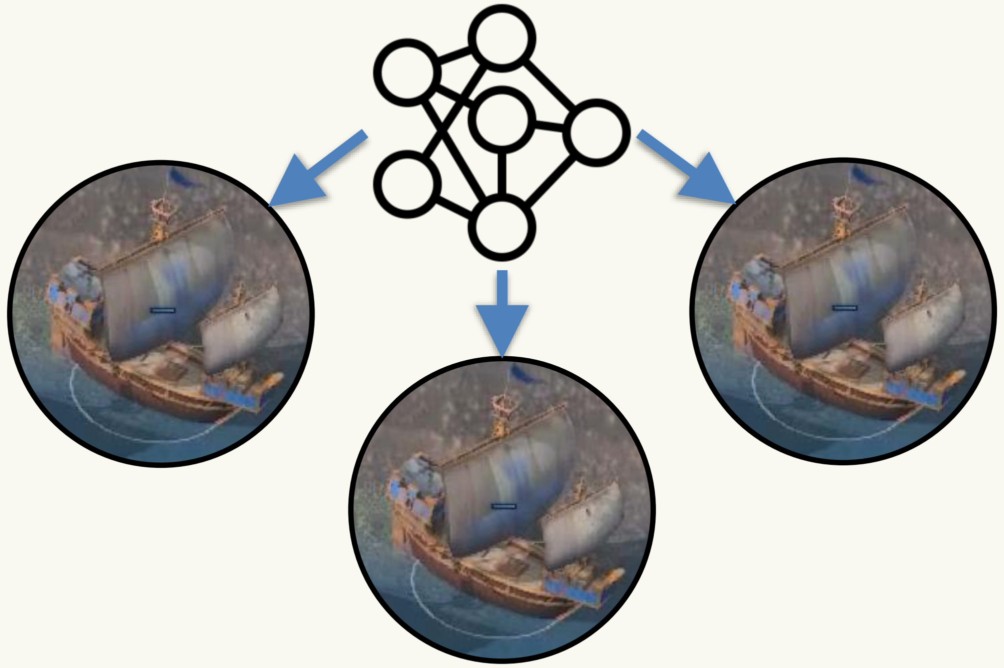
https://docs.ray.io/en/latest/rllib-models.html#default-model-config-settings

Trial: Ship to ship combat

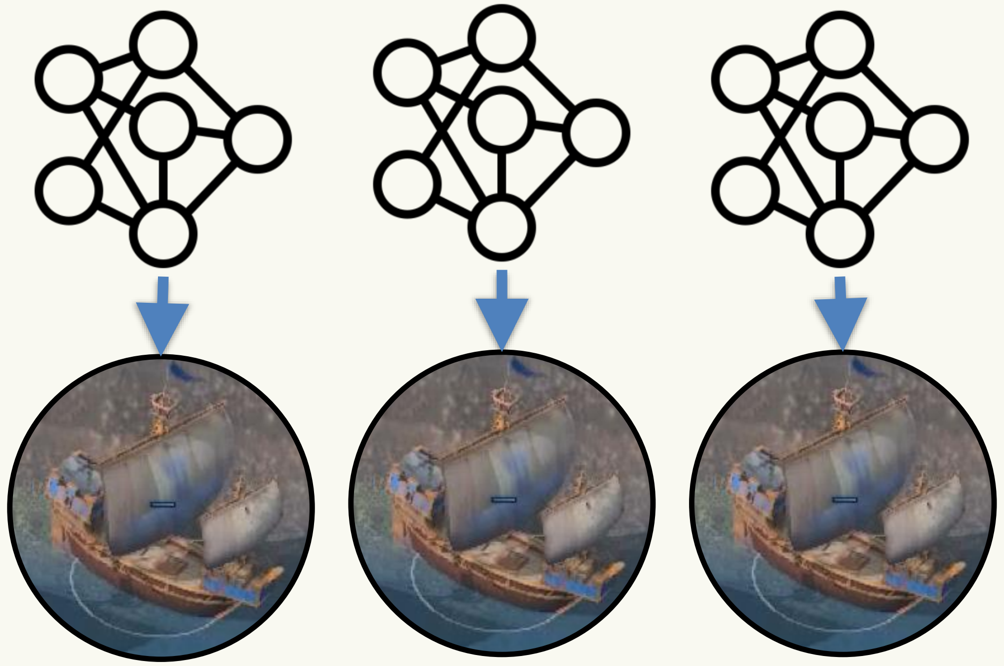
RL ship

Built-in AI ship

Multi-unit training paradigms

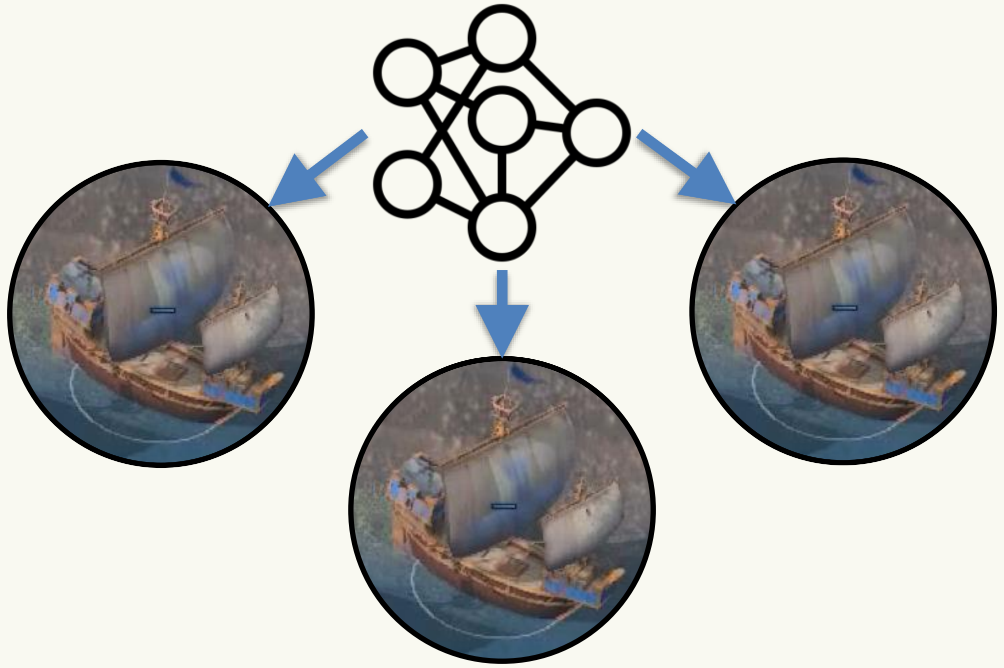


Single agent with
joint action space

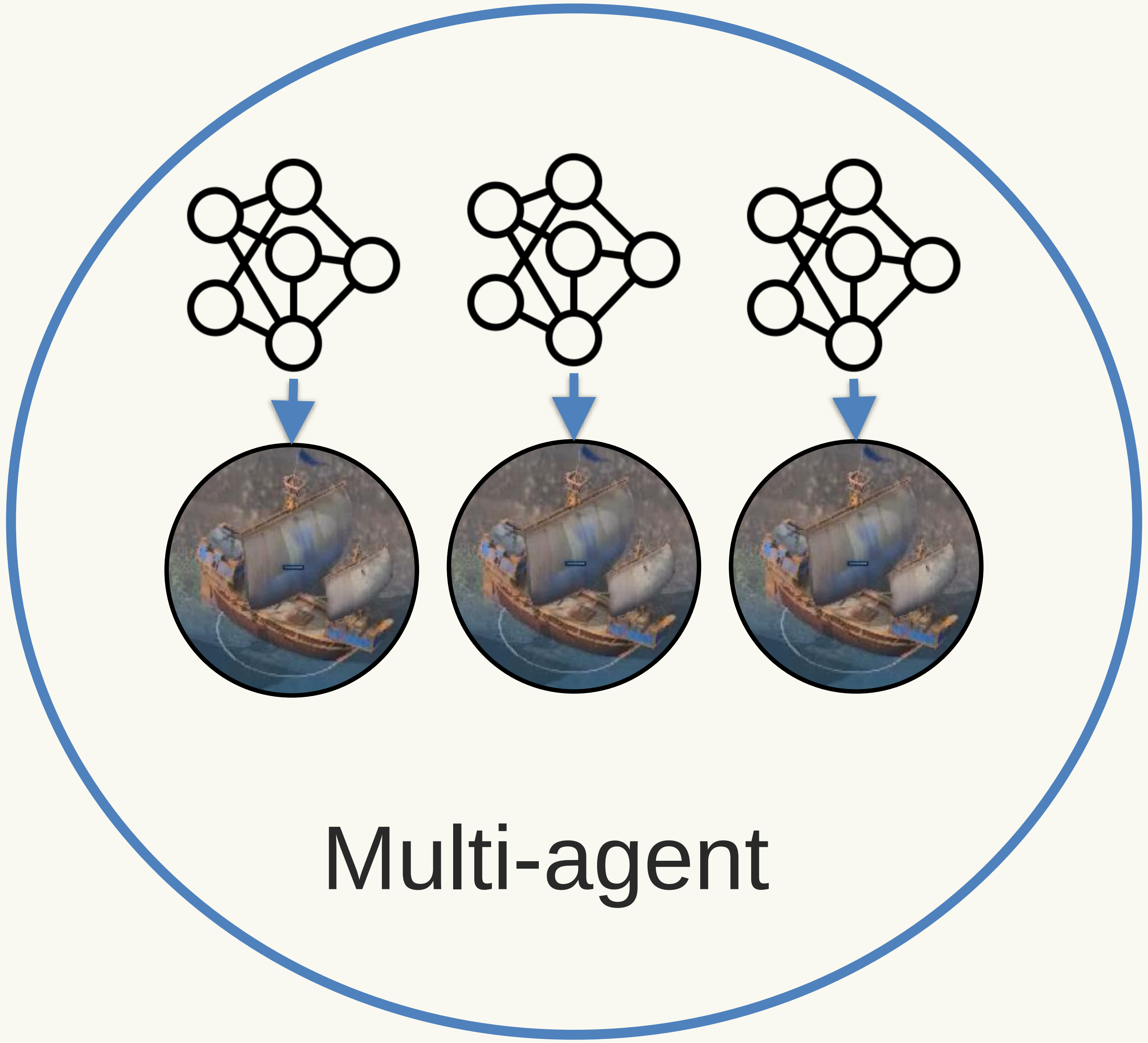


Multi-agent

Multi-unit training paradigms

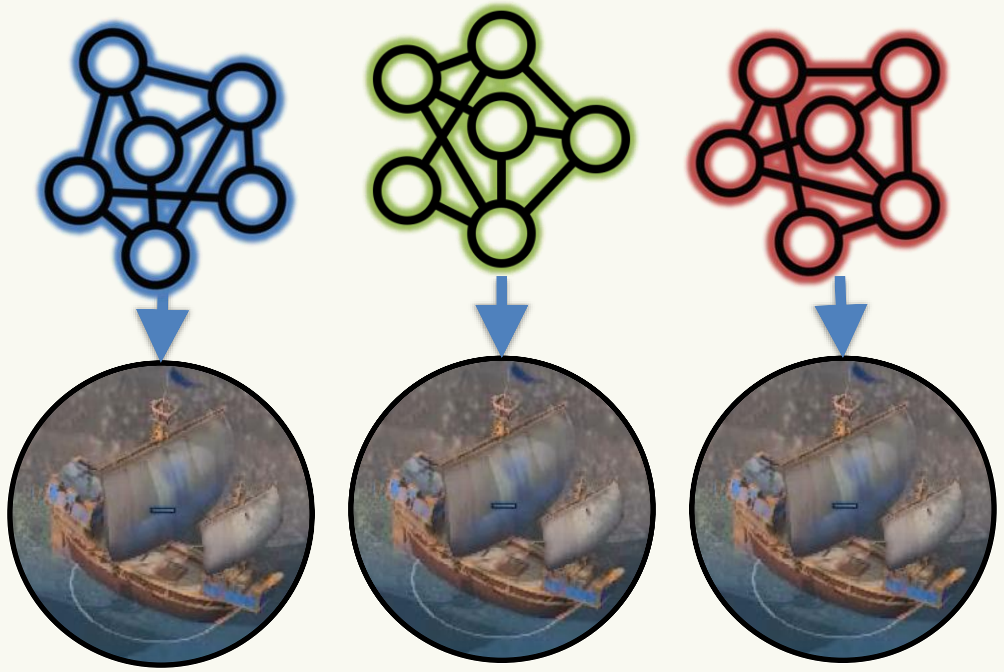


Single agent with
joint action space

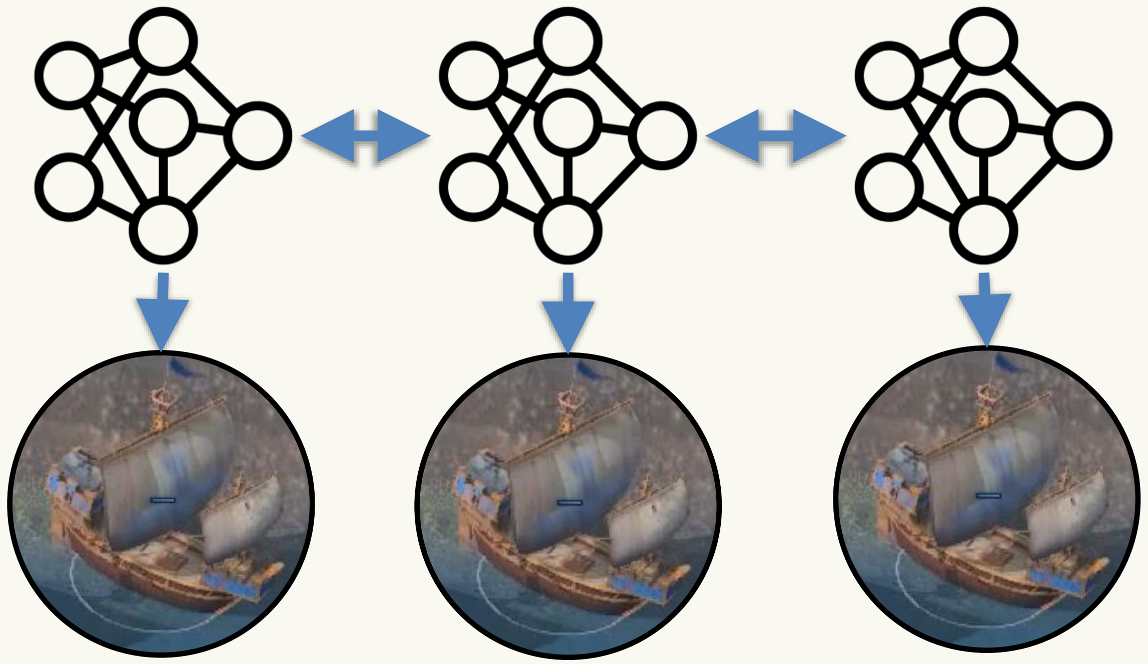


Multi-agent

Multi-unit training paradigms

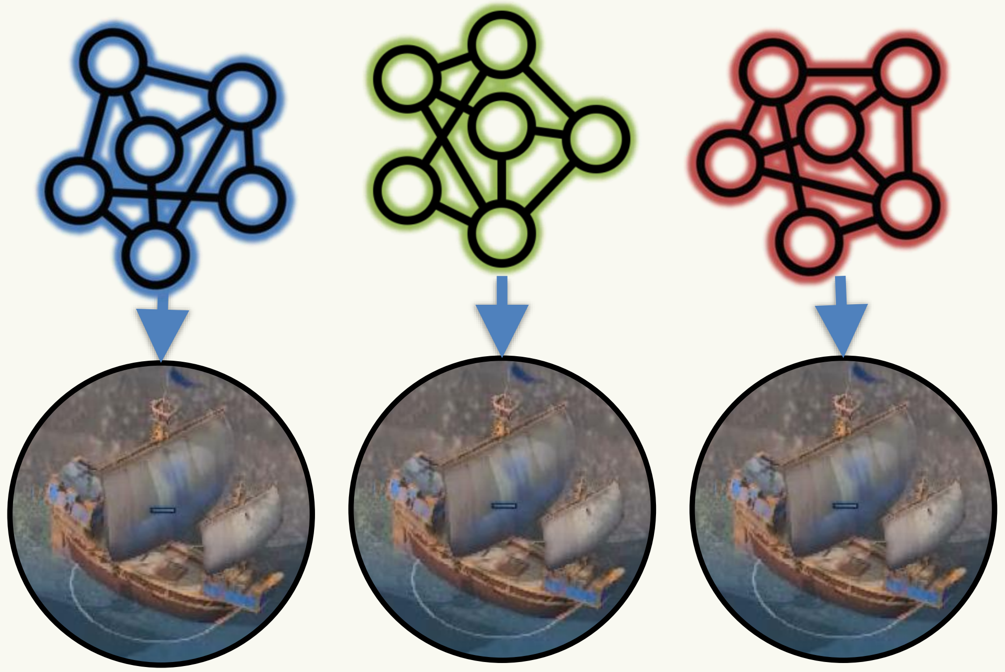


Multi-agent with
separate weights

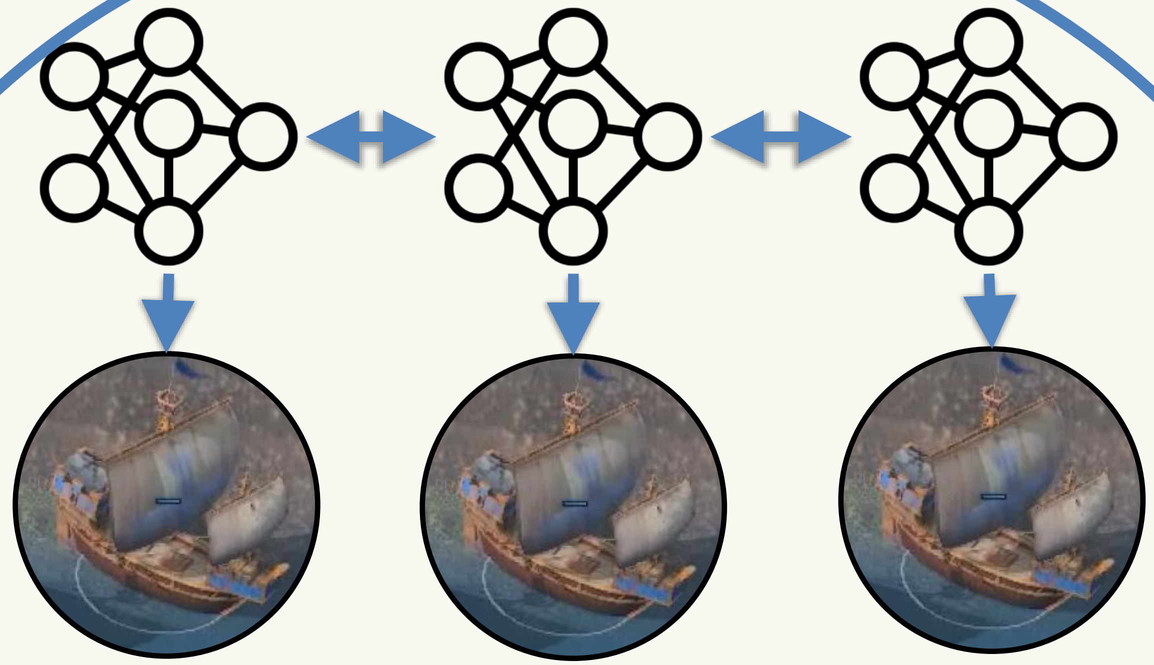


Multi-agent with
shared weights

Multi-unit training paradigms

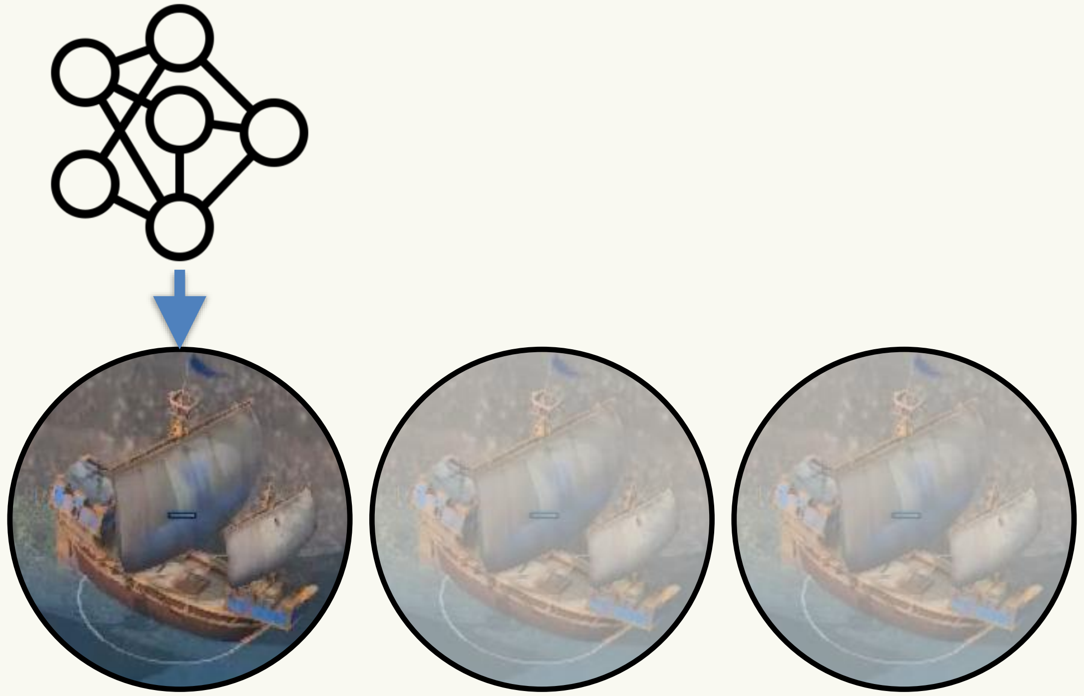


Multi-agent with separate weights



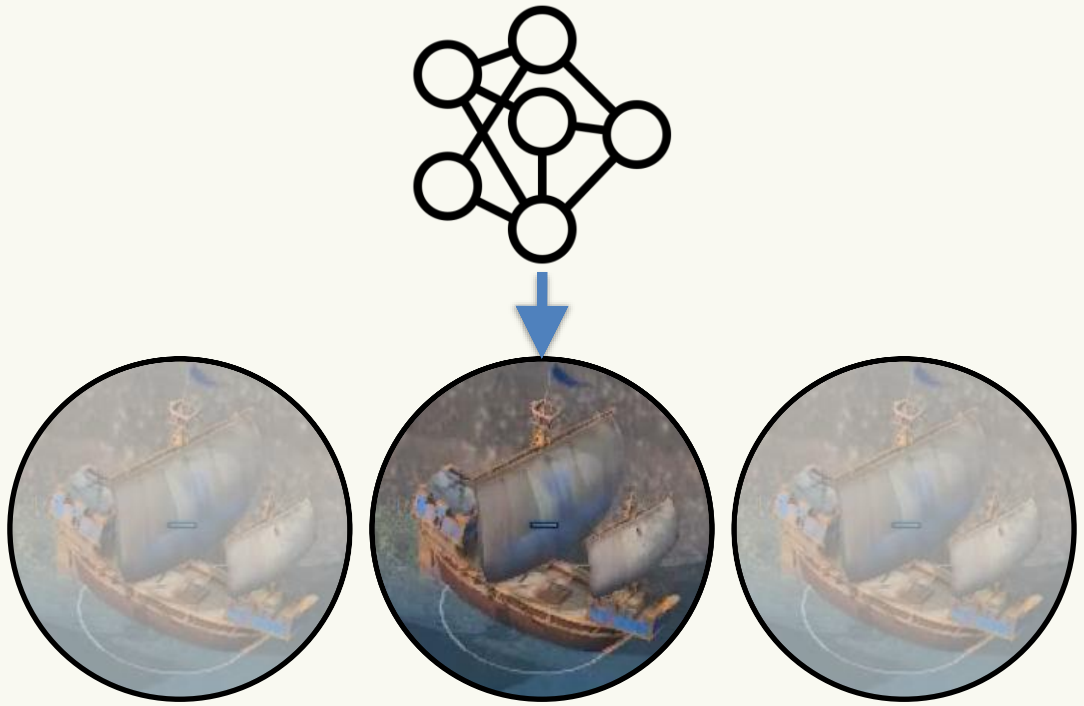
Multi-agent with shared weights

Multi-unit training paradigms



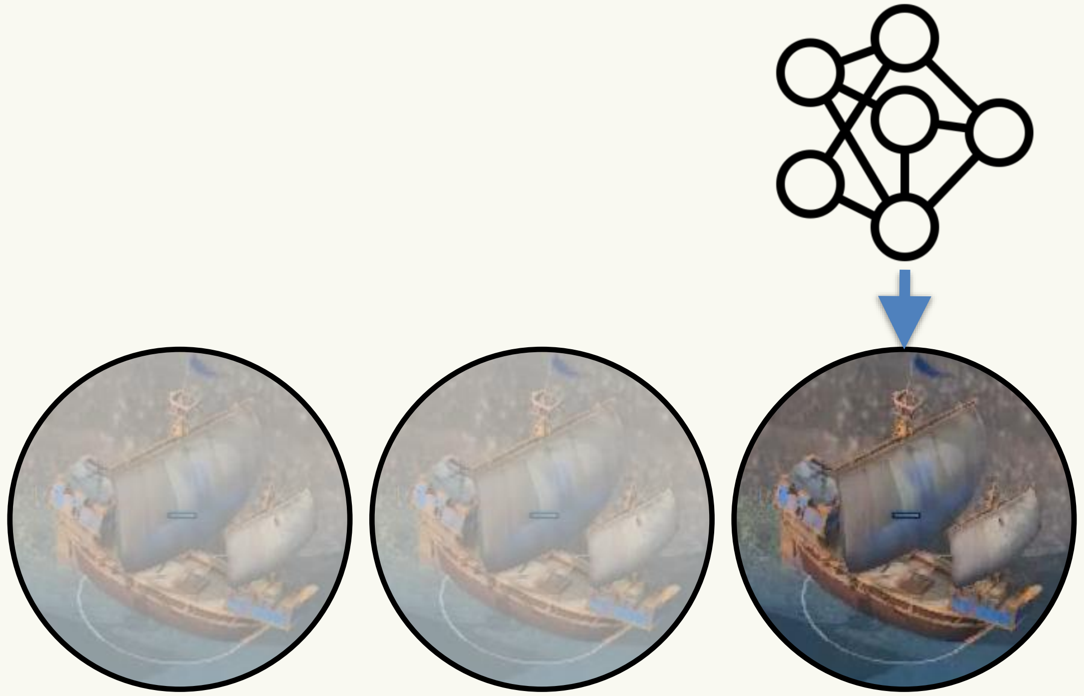
Weight freezing

Multi-unit training paradigms



Weight freezing

Multi-unit training paradigms



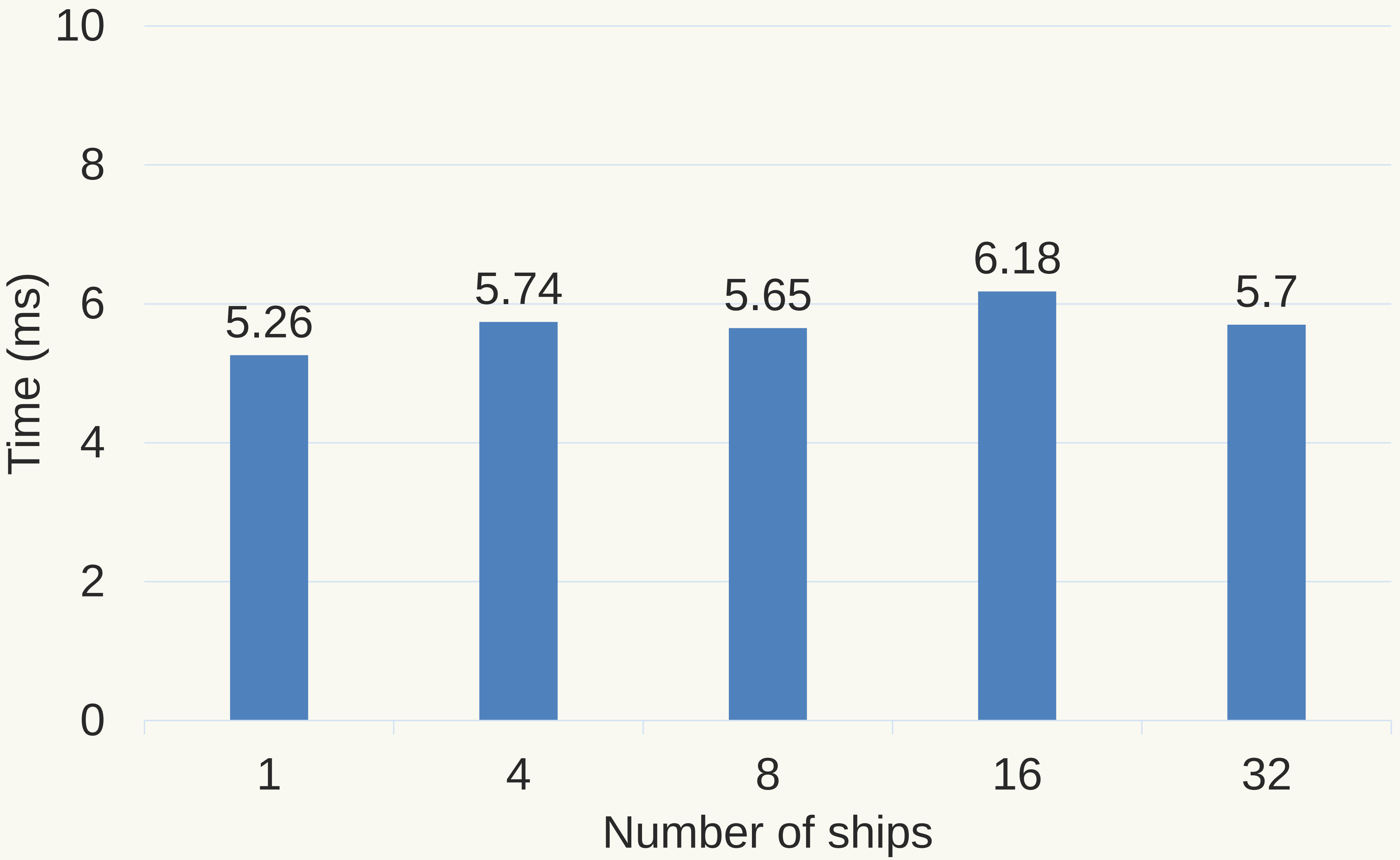
Weight freezing

Trial: 4v4 plausible naval battle

RL ships

Built-in AI ships

Policy inference time for all ships



Near-constant scaling

Inference not optimized, could expect from 5x to 80x speedup[1]

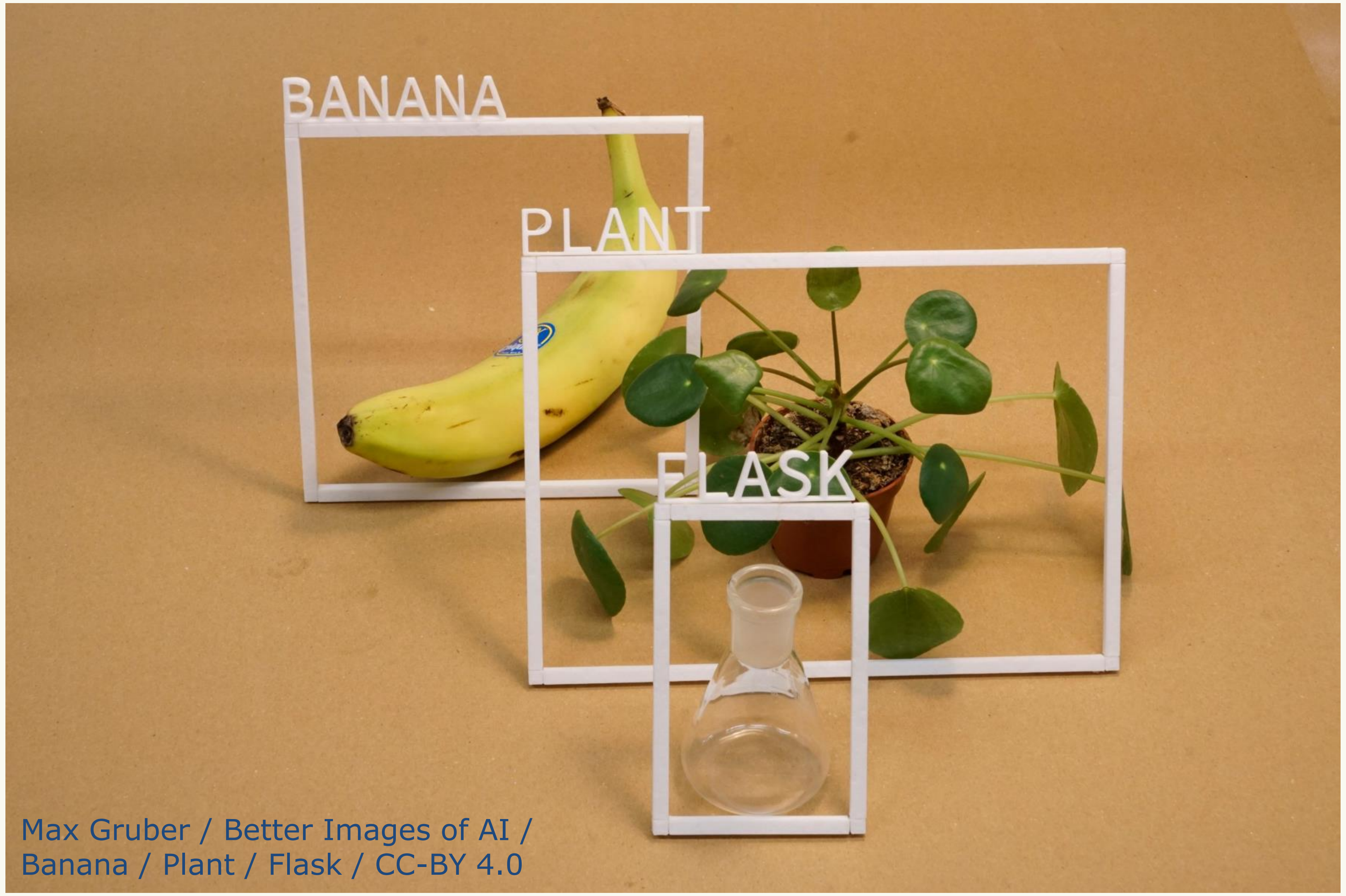
Single Intel Core i7-8650U CPU

[1]https://software.intel.com/content/www/us/en/develop/articles/tensorflow-optimizations-on-modern-intel-architecture.html

Designing A Modular AI

Combat Fitness	Farm Optimization	Multi-Unit Navigation + Combat
		
Supervised Learning	Utility System	Reinforcement Learning

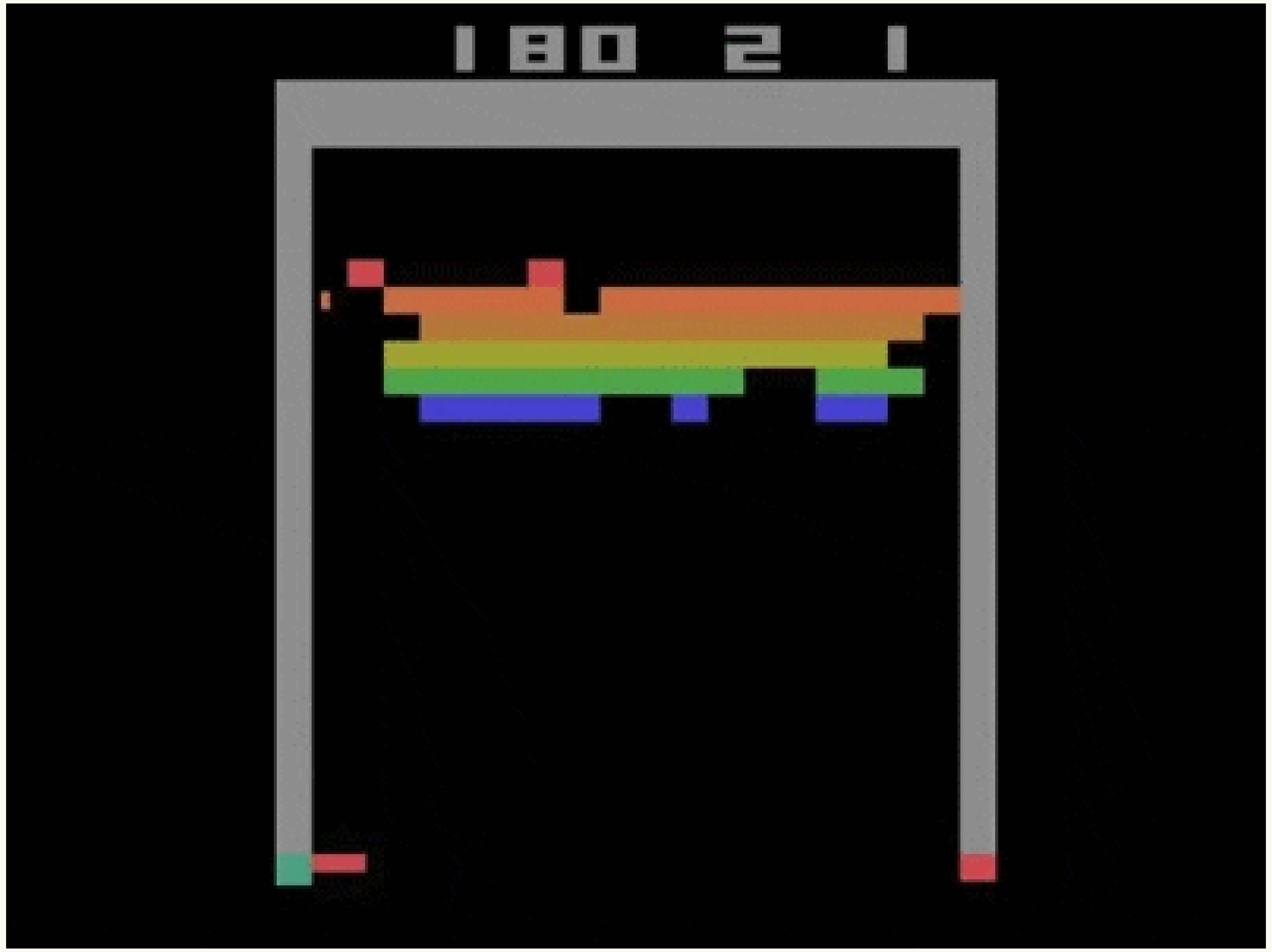
What makes a *good* supervised learning problem?



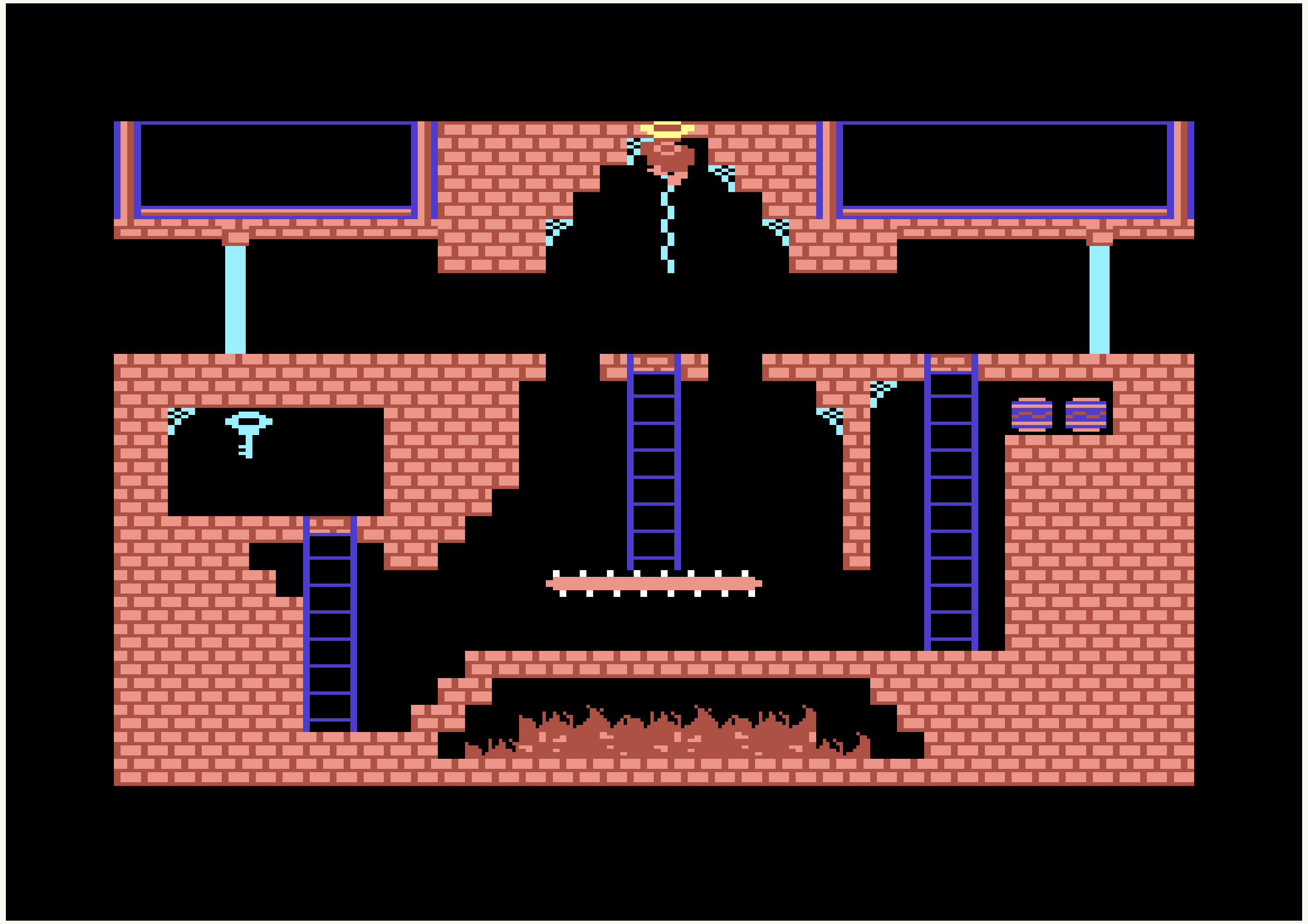
What makes a *good* supervised learning problem?

AGE OF EMPIRES IV

What makes a *good* RL problem?



Breakout



Montezuma's Revenge

Designing A Modular AI

Combat Fitness	Farm Optimization	Multi-Unit Navigation + Combat
		
Supervised Learning	Utility System	Reinforcement Learning

Pathing in 'Age of Empires IV': Flow Fields and Steering Behaviors
Frank Cheng -**Location:** Room 2010, West Hall
Date: Wednesday, March 23
Time: 10:30 am - 11:00 am

Give Your Players a Seat at the Table: Feedback Fundamentals
Emma Bridle & Savannah Harrison
Location: Room 2010, West Hall
Date: Wednesday, March 23
Time: 10:30 am - 11:00 am

The MAW: Safely Multithreading the Deterministic Gameplay of 'Age of Empires IV'
Joel Pritchett -**Location:** Room 2006, West Hall
Date: Thursday, March 24
Time: 2:00 pm - 2:30 pm

GDC | March 21-25, 2022
San Francisco, CA

Age of Empires IV: Machine Learning Trials and Tribulations

#GDC22