

# What can we learn from a **billion** agents?

Ayush Chopra  
MIT

[github.com/AgentTorch/AgentTorch](https://github.com/AgentTorch/AgentTorch)

## AI economist with 4 agents



Zheng et al (2021)

## AI economist with 4 agents



Zheng et al (2021)

## Smallville with 25 agents



Park et al (2023)

## AI economist with 4 agents



Zheng et al (2021)

## Smallville with 25 agents



Park et al (2023)

## Minecraft with 1000 agents



Yang et al (2024)

## AI economist with 4 agents



Zheng et al (2021)

## Smallville with 25 agents



Park et al (2023)

## Minecraft with 1000 agents



Yang et al (2024)

## Deepmind framework for generative social simulations



### Concordia

A library for generative social simulation

python 3.11 | 3.12 pyPI v1.7.0 pypi-test passing test-concordia passing test-examples passing

[Concordia Tech Report](#)

Vezhnevets et al (2023)





high-resolution agents + population-scale interactions



END THE  
LOCKDOWN

VACCINATE  
HIMSELF!

LOCKDOWN!

DON'T  
BE A  
LAB RA

COVID=  
SCAM

THIS SIGN  
TESTED POSITIVE  
FOR COVID-19

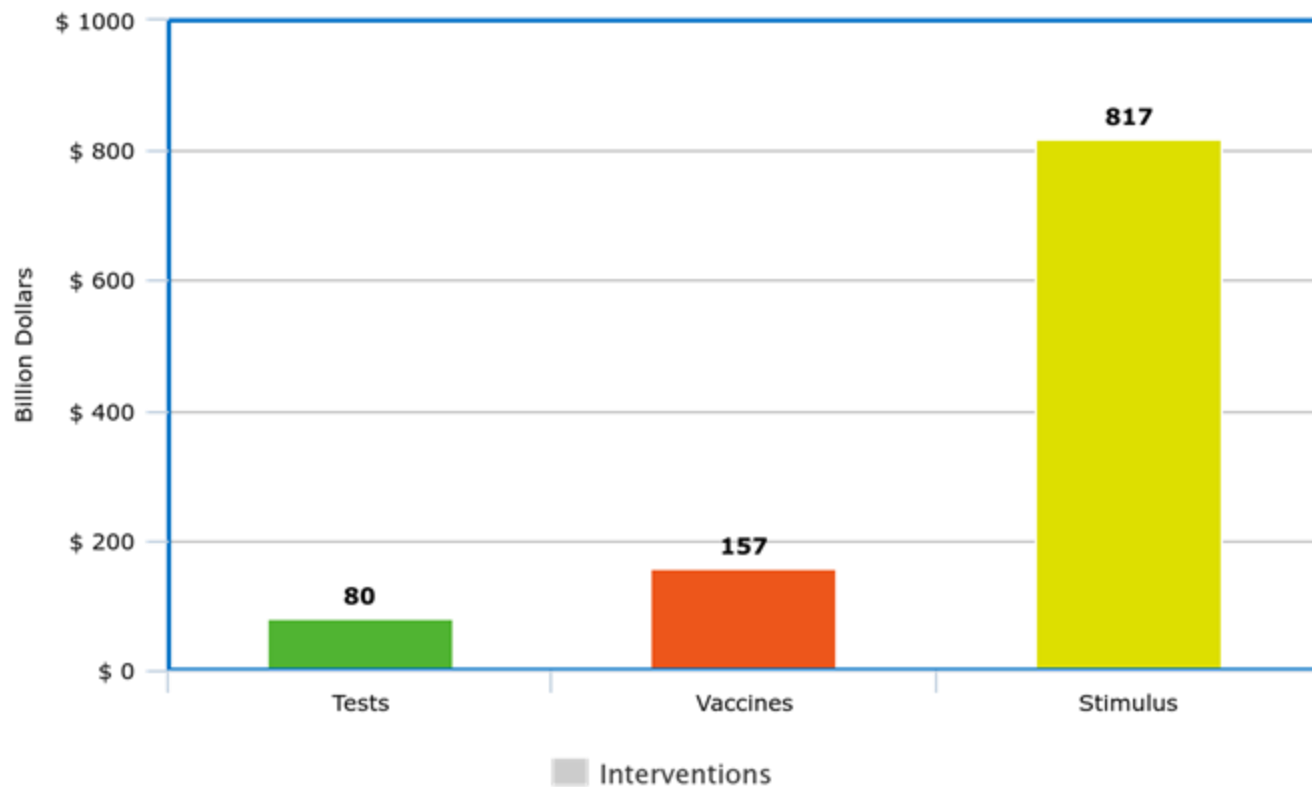
SAVE CALIFORNIA  
RECALL GAVIN NEWSOM  
SIGN HERE

NO  
VENDING

58



## Trillion-Dollar Pandemic





# "What kind of a test do you want?"

10% false-positive  
3 day result

40% false positive  
1 day result



# Large Population Models

[lpm.media.mit.edu/platform](http://lpm.media.mit.edu/platform)

**model millions of interacting agents to capture collective dynamics**



5 million citizens





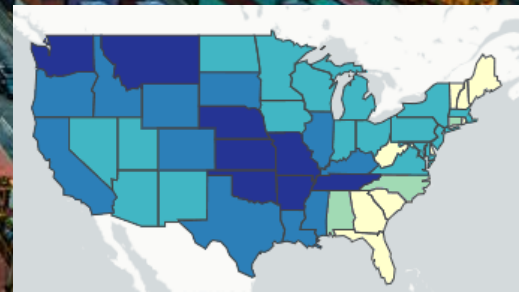
30 million households







\$ billion supply chains



# Outline: What can we learn from a billion agents?

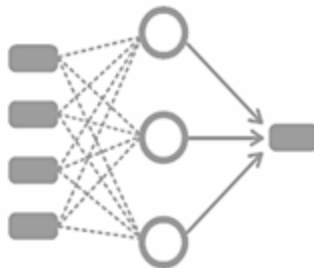
- Motivation and Impact
- Introduction
  - Large Population Models (LPMs)
  - The AgentTorch framework
- Case Study
  - Modeling disease spread over 8.4 million agents
- Conclusion

# Challenges with agent-based simulations



Expressive agents

Adaptive  
Heterogeneous



Dynamic interactions

Multi-scale  
Stochastic



Synchronized analysis

Decentralized  
Sensitive

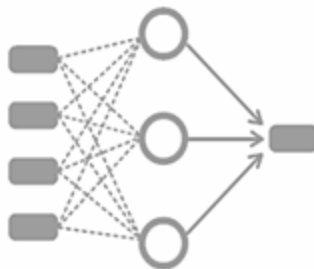
# Large Population Models



**Expressive agents**

Adaptive  
Heterogeneous

**LLM Archetypes**



**Dynamic interactions**

Multi-scale  
Stochastic



**Synchronized analysis**

Decentralized  
Sensitive



# Large Population Models

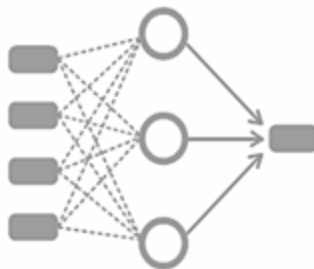


**Expressive agents**

Adaptive  
Heterogeneous

**LLM Archetypes**

AAMAS'23; arxiv'24



**Dynamic interactions**

Multi-scale  
Stochastic

**Differentiability**

BMJ'21; AAMAS'23;  
AAMAS'24 (Best paper runner-up)



**Synchronized analysis**

Decentralized  
Sensitive

# Large Population Models

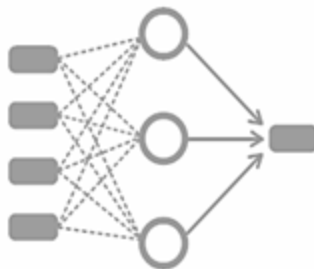


**Expressive agents**

Adaptive  
Heterogeneous

**LLM Archetypes**

AAMAS'23; arxiv'24



**Dynamic interactions**

Multi-scale  
Stochastic

**Differentiability**

BMJ'21; AAMAS'23;  
AAMAS'24 (Best paper runner-up)



**Synchronized analysis**

Decentralized  
Sensitive

**Secure MPC**

ICML-W'22 (Best paper);  
AAMAS'24; Nature Medicine'25

# Simulate an entire country on commodity hardware

Execute upto 300,000 interactions/sec and scale to 60 million agents/GPU

Method	Simulation	Calibration	Analysis
Conventional ABM*	50 hours	100,000 hours	5,000 hours
<b>LPM</b>	5 minutes	12 hours	10 seconds

# Simulate an entire country on commodity hardware

Execute upto 300,000 interactions/sec and scale to 60 million agents/GPU

Method	Simulation	Calibration	Analysis
Conventional ABM*	50 hours	100,000 hours	5,000 hours
<b>LPM</b>	5 minutes	12 hours	10 seconds

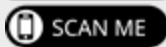
+ 600x

+ 8300x

+ 5000x

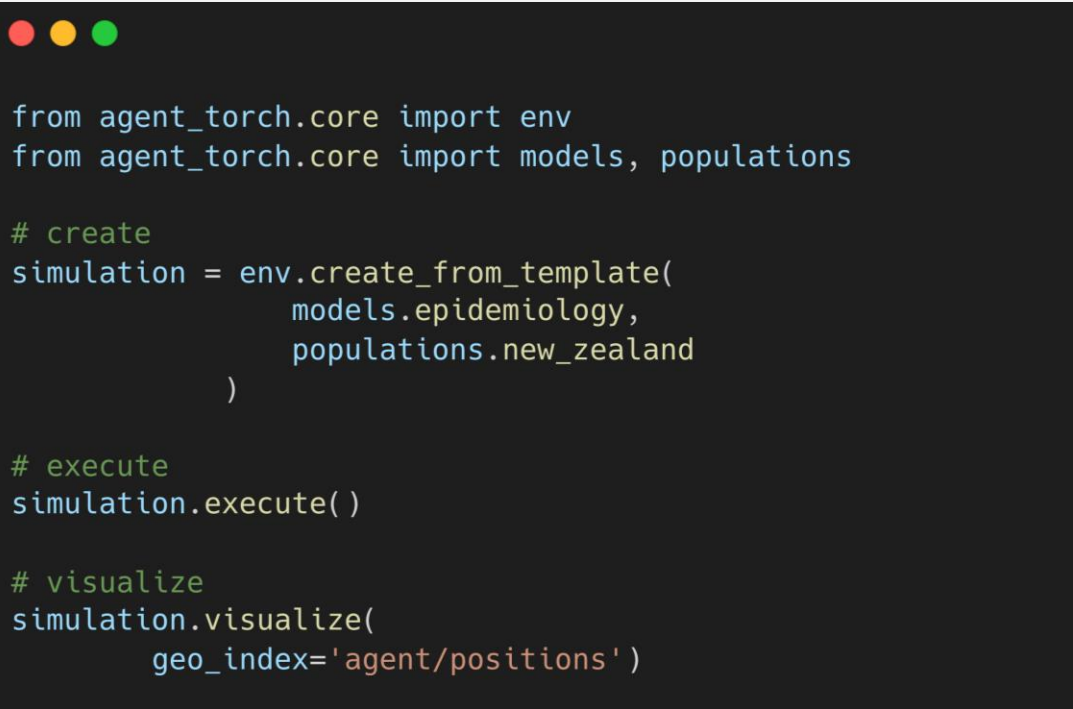
[github.com/AgentTorch/AgentTorch](https://github.com/AgentTorch/AgentTorch)

`pip install agent-torch`





# Build end-to-end workflows in 3 lines of code



```
from agent_torch.core import env
from agent_torch.core import models, populations

# create
simulation = env.create_from_template(
    models.epidemiology,
    populations.new_zealand
)

# execute
simulation.execute()

# visualize
simulation.visualize(
    geo_index='agent/positions')
```

Expressive templates

Custom populations

High-res visualizations

# Build end-to-end workflows in 3 lines of code

```
from agent_torch.core import env
from agent_torch.core import models, populations

# create
simulation = env.create_from_template(
    models.epidemiology,
    populations.new_zealand
)

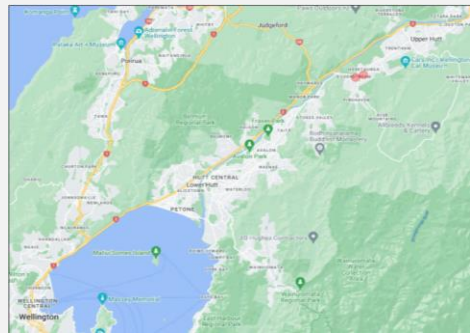
# execute
simulation.execute()

# visualize
simulation.visualize(
    geo_index='agent/positions')
```

Expressive templates

Custom populations

High-res visualizations



# Comparison with other frameworks

Feature	AgentTorch	Concordia	Flame	Mesa
GPU Execution	✓	✓	✓	✗
Million-agent Populations	✓	✗	✓	⚠
Differentiable Environments	✓	✗	✗	✗
Mechanistic Environments	✓	✗	✓	✓
LLM Integration	✓	✓	✗	✗
Neural Composition	✓	⚠	✗	✗

# Outline: What can we learn from a billion agents?

- Motivation and Impact
- Introduction
- Case Study:
  - Problem Formulation: modeling disease spread over 8.4 million agents
  - Part 1: Population Prompting via Archetypes
  - Part 2: Gradient-assisted simulation via Differentiable Environments
  - Part 3: Decentralized analysis via Secret Sharing
- Conclusion



END THE  
LOCKDOWN

VACCINATE  
HIMSELF!

LOCKDOWN!

DON'T  
BE A  
LAB RA

COVID=  
SCAM

THIS SIGN  
TESTED POSITIVE  
FOR COVID-19

SAVE CALIFORNIA  
RECALL GAVIN NEWSOM  
SIGN HERE

NA  
VENTA

LT  
LM  
GAS  
COVE  
58

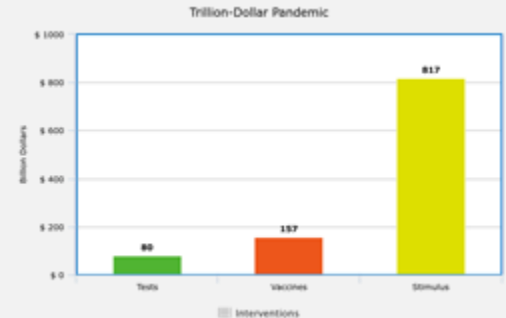
## New York City

- 5 boroughs
- 8.4 million people
- 3.3 million households
- 200,000 small businesses

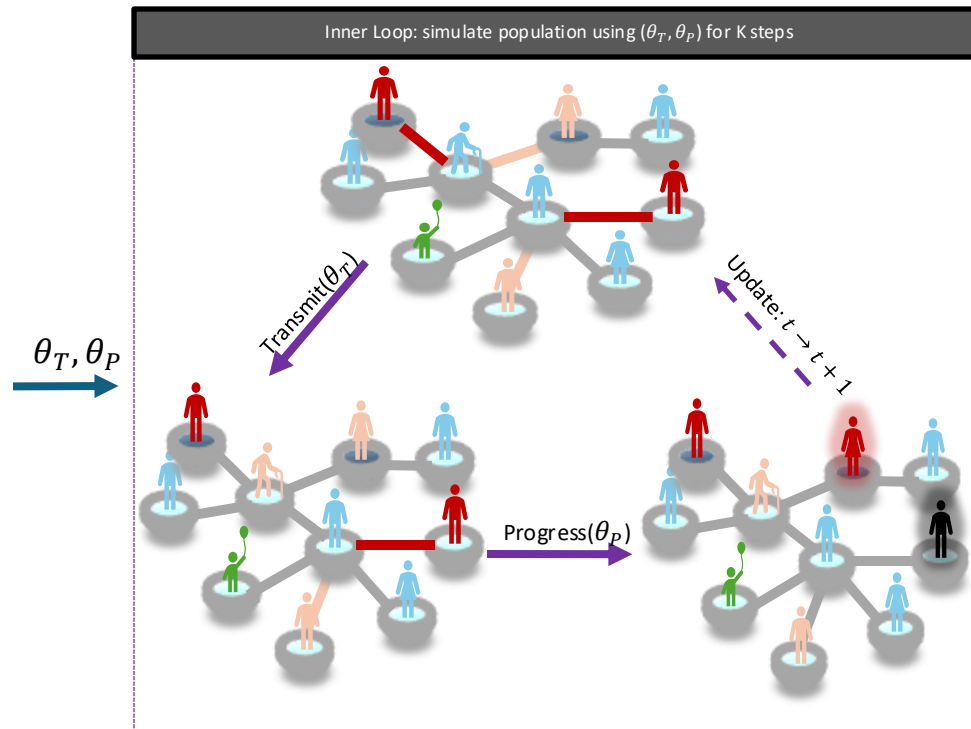


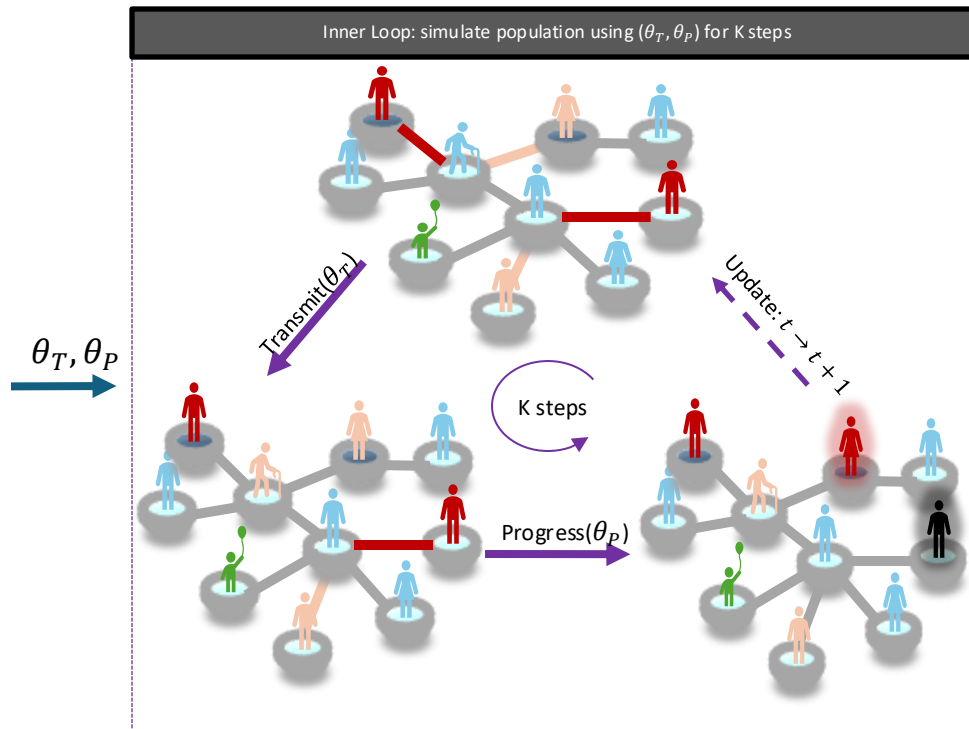
2020-2022

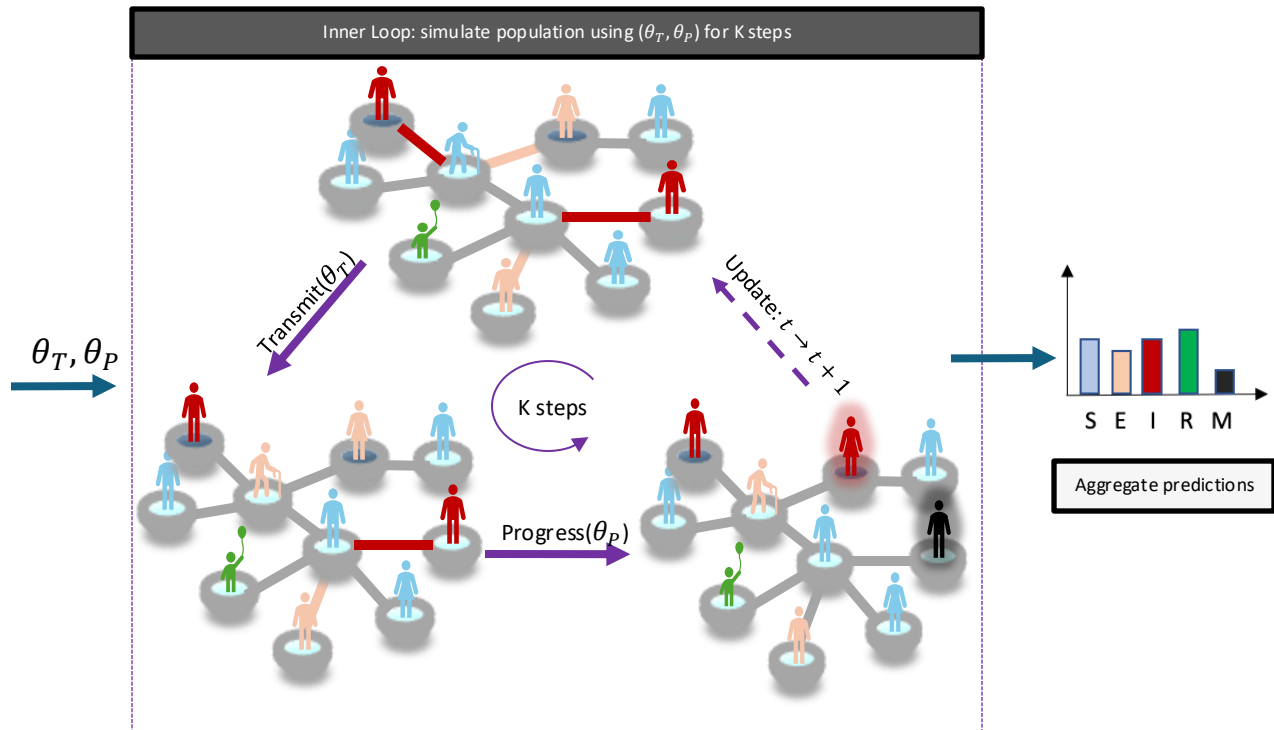
# What if we give \$500 stimulus payments?











$$\mathbf{z}_i(t+1) = f\left(\mathbf{z}_i(t), \bigoplus_{j \in \mathcal{N}_i(t)} M_{ij}(t), \boldsymbol{\theta}, \ell(\cdot | \mathbf{z}_i(t))\right)$$

$$\mathbf{z}_i(t+1) = f \left( \mathbf{z}_i(t), \bigoplus_{j \in \mathcal{N}_i(t)} M_{ij}(t), \boldsymbol{\theta}, \ell(\cdot | \mathbf{z}_i(t)) \right)$$

Diagram illustrating the components of the state transition function  $f$ :

- New state of agent**:  $\mathbf{z}_i(t+1)$
- Current state of agent**:  $\mathbf{z}_i(t)$
- neighborhood influence**:  $\bigoplus_{j \in \mathcal{N}_i(t)} M_{ij}(t)$
- environmental influence**:  $\boldsymbol{\theta}$
- Individual agency**:  $\ell(\cdot | \mathbf{z}_i(t))$

“Spread factor” of the variant  
environmental influence

- Disease status  
- Age, Income, Occupation  
Current state of agent

$$\mathbf{z}_i(t+1) = f \left( \mathbf{z}_i(t), \bigoplus_{j \in \mathcal{N}_i(t)} M_{ij}(t), \boldsymbol{\theta}, \ell(\cdot | \mathbf{z}_i(t)) \right)$$

New state of agent

- New disease status
- Age, Income, Occupation

neighborhood influence

- Time since infection
- Vaccination status


Individual agency

Willingness to interact

## Q1: model individual agency?

$$\mathbf{z}_i(t+1) = f \left( \mathbf{z}_i(t), \bigoplus_{j \in \mathcal{N}_i(t)} M_{ij}(t), \boldsymbol{\theta}, \ell(\cdot | \mathbf{z}_i(t)) \right)$$

- Varies with agent state
- Need to be sampled online



LLMs to prompt 8.4 millions agents in NYC?



# Large Population Models



**Expressive agents**

Adaptive  
Heterogeneous

**LLM Archetypes**

AAMAS'23; arxiv'24



**Dynamic interactions**

Multi-scale  
Stochastic

**Differentiability**

BMJ'21; AAMAS'23;  
AAMAS'24 (Best paper runner-up)



**Data-driven analysis**

Decentralized  
Sensitive

**Secure MPC**

ICML-W'22 (Best paper);  
AAMAS'24; Nature Medicine'25



Expressive agents

Adaptive  
Heterogeneous

LLM Archetypes

## Scale LLMs to prompt millions of agents

```
from agent_torch.core import Archetype, Behavior
from agent_torch.populations import NYC

# Create an object of Archetype class
# n_arch estimates a predictive posterior over outcomes
archetype = Archetype(n_arch=7)

# Create an object of Behavior class
work_behavior = Behavior(archetype=archetype.llm(prompt),
                        region=NYC)

will_work = work_behavior.sample()
```

# Agent Prompt

## User Prompt

You are a {gender} of age {age}, living in the {location} region. You work in {occupation } industry with monthly income of {income }.

The number of new cases in your neighborhood is {cases}, which is a {change}% change from the previous month. It has been {duration} months since the start of the pandemic.

This month, you have received a stimulus payment of {payment} to support your living expenses.

Given these factors, do you choose to isolate at home? (isolation behavior)

"There isn't enough information" and "It is unclear" are not acceptable answers. Give a "Yes" or "No" answer, followed by a period. Give one sentence explaining your choice.

*#archetypes = age x gender x regions x occupation x income-levels*

# Specify prompt-specific archetypes



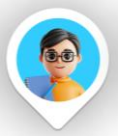
8.4 million agents in NYC



~2000 representative archetypes

prompt dimensions  
data resolution (e.g. census)

# Population Prompting via Archetypes



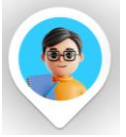
Prompt

•  
•  
•



Archetypes

# Population Prompting via Archetypes



Prompt

Aggregate

.  
. .  
. .

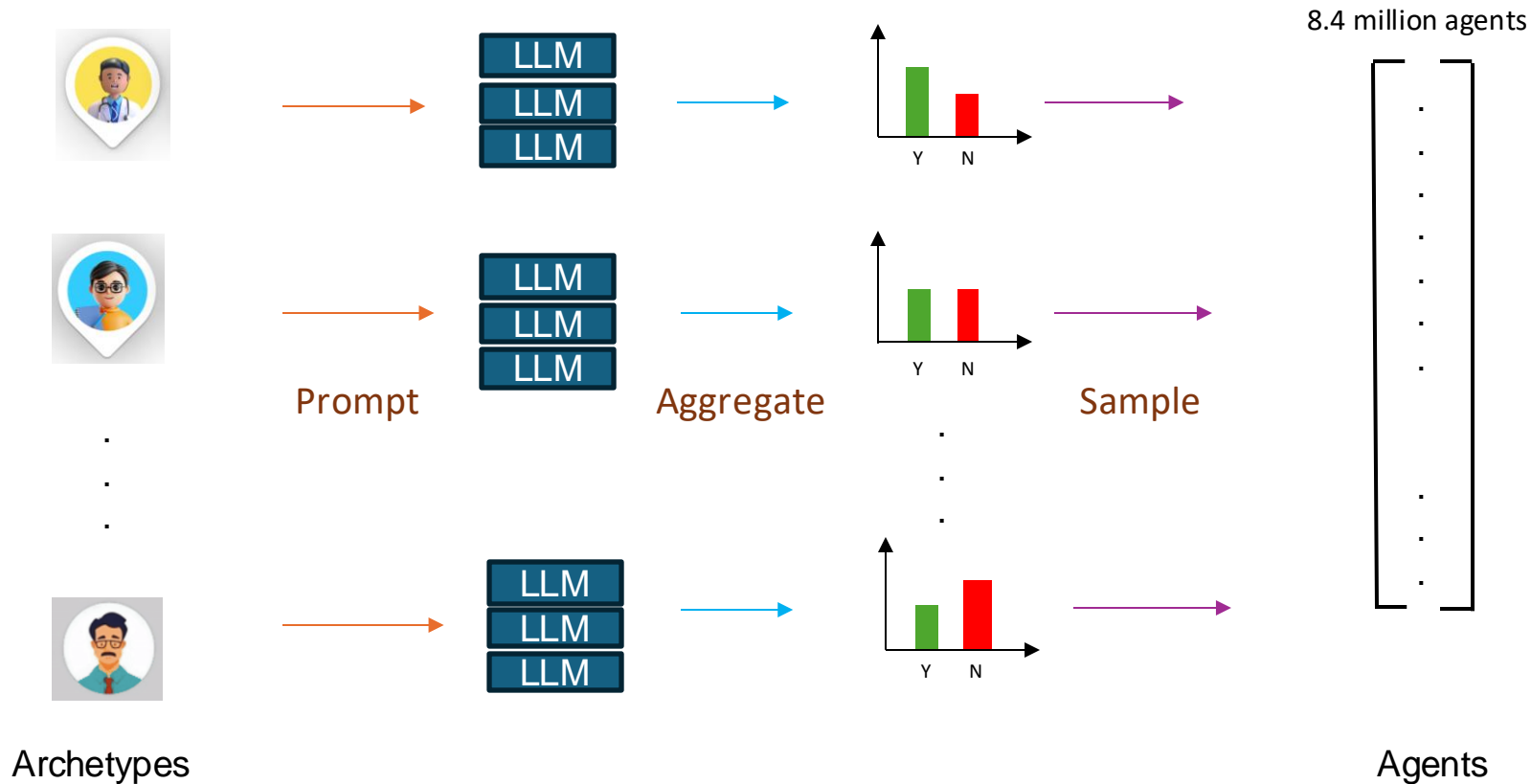
.  
. .  
. .



Archetypes



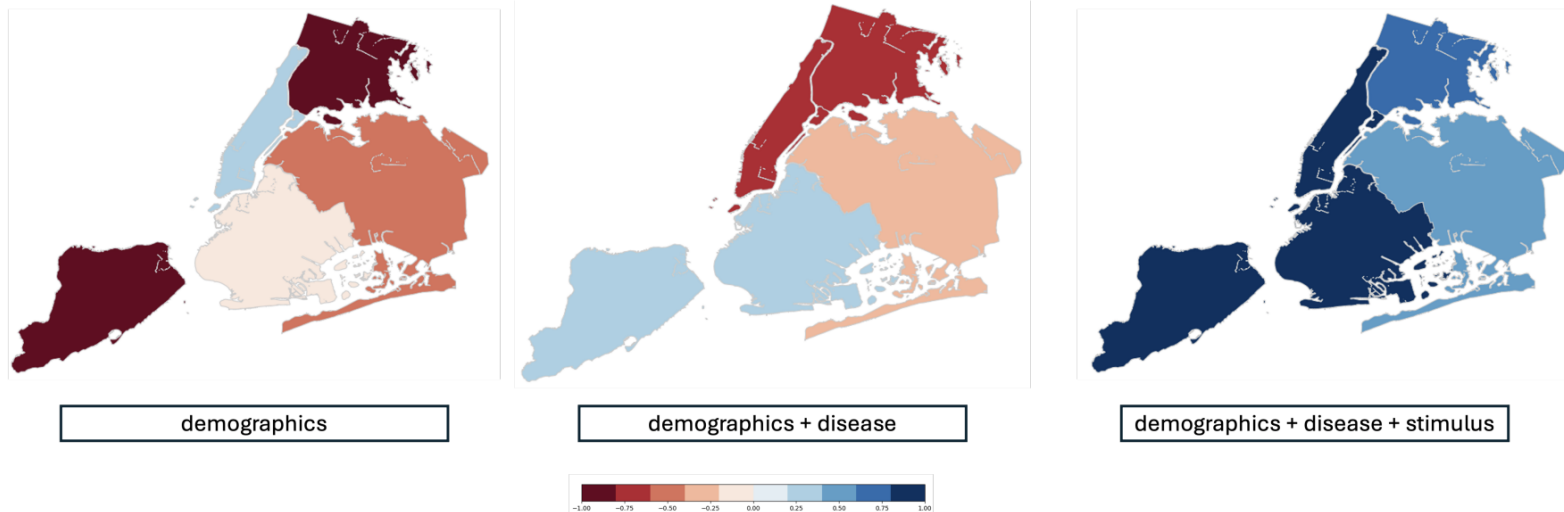
# Population Prompting via Archetypes



# High variance individual agents produce reliable census-level estimates

Prompt virtual population of 8.4 million agents

Measure correlation with labor force participation census

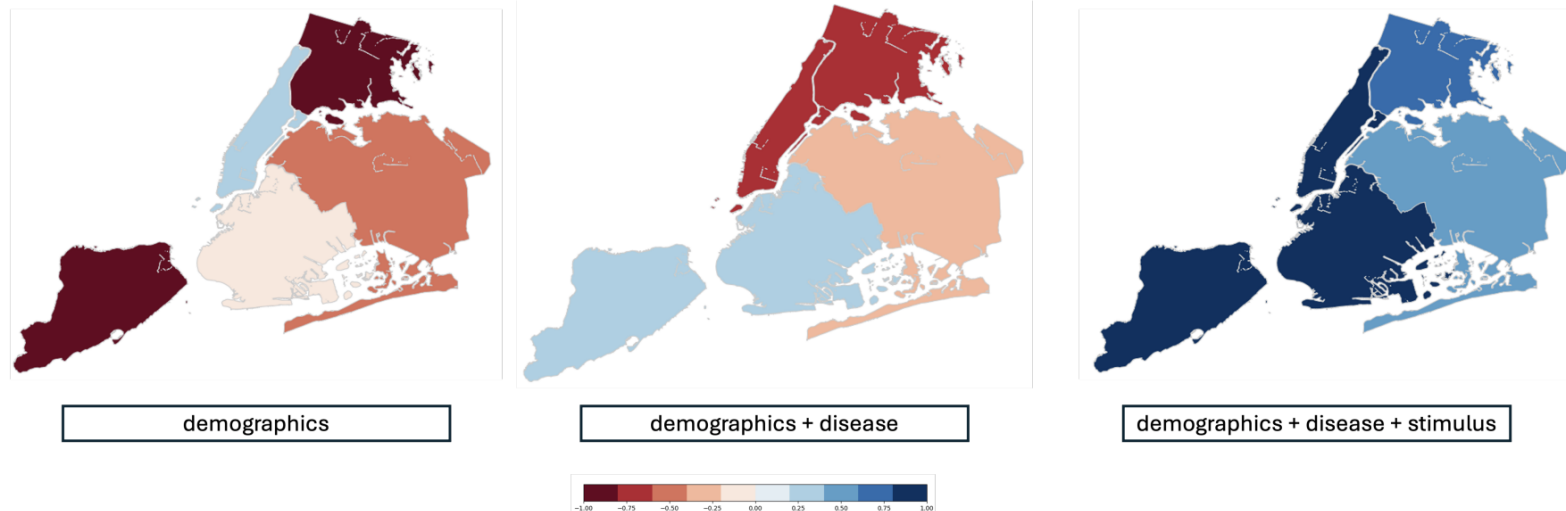


Prompt axes:

# High variance individual agents produce reliable census-level estimates

Prompt virtual population of 8.4 million agents

Measure correlation with labor force participation census



**Rebuild the census, in simulation!?**

$$\mathbf{z}_i(t+1) = f \left( \mathbf{z}_i(t), \bigoplus_{j \in \mathcal{N}_i(t)} M_{ij}(t), \boldsymbol{\theta}, \ell(\cdot | \mathbf{z}_i(t)) \right)$$

## Q2: simulate and calibrate efficiently?

Simulation involves repeated execution of  $f$


Calibration involves estimating  $\theta$

$$\mathbf{z}_i(t+1) = f \left( \mathbf{z}_i(t), \bigoplus_{j \in \mathcal{N}_i(t)} M_{ij}(t), \theta, \ell(\cdot | \mathbf{z}_i(t)) \right)$$

## Q2: simulate and calibrate efficiently?

Simulation involves repeated execution of  $f$

Calibration involves estimating  $\theta$

$$\mathbf{z}_i(t+1) = f \left( \mathbf{z}_i(t), \bigoplus_{j \in \mathcal{N}_i(t)} M_{ij}(t), \theta, \ell(\cdot | \mathbf{z}_i(t)) \right)$$


- neural or mechanistic
- Stochastic and multiple “sub-steps”

$f$  vectorized and differentiable?



# Large Population Models

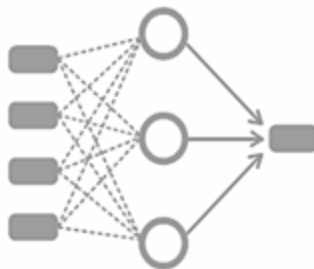


Expressive agents

Adaptive  
Heterogeneous

LLM Archetypes

AAMAS'23; arxiv'24



Dynamic interactions

Multi-scale  
Stochastic

Differentiability

BMJ'21; AAMAS'23;  
AAMAS'24 (Best paper runner-up)



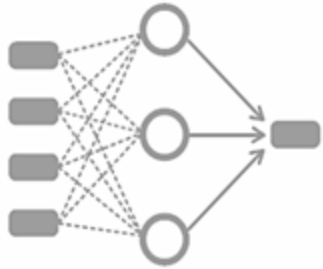
Data-driven analysis

Decentralized  
Sensitive

Secure MPC

ICML-W'22 (Best paper);  
AAMAS'24; Nature Medicine'25

# Compose end-to-end with NNs



Dynamic interactions

Multi-scale  
Stochastic

Differentiability



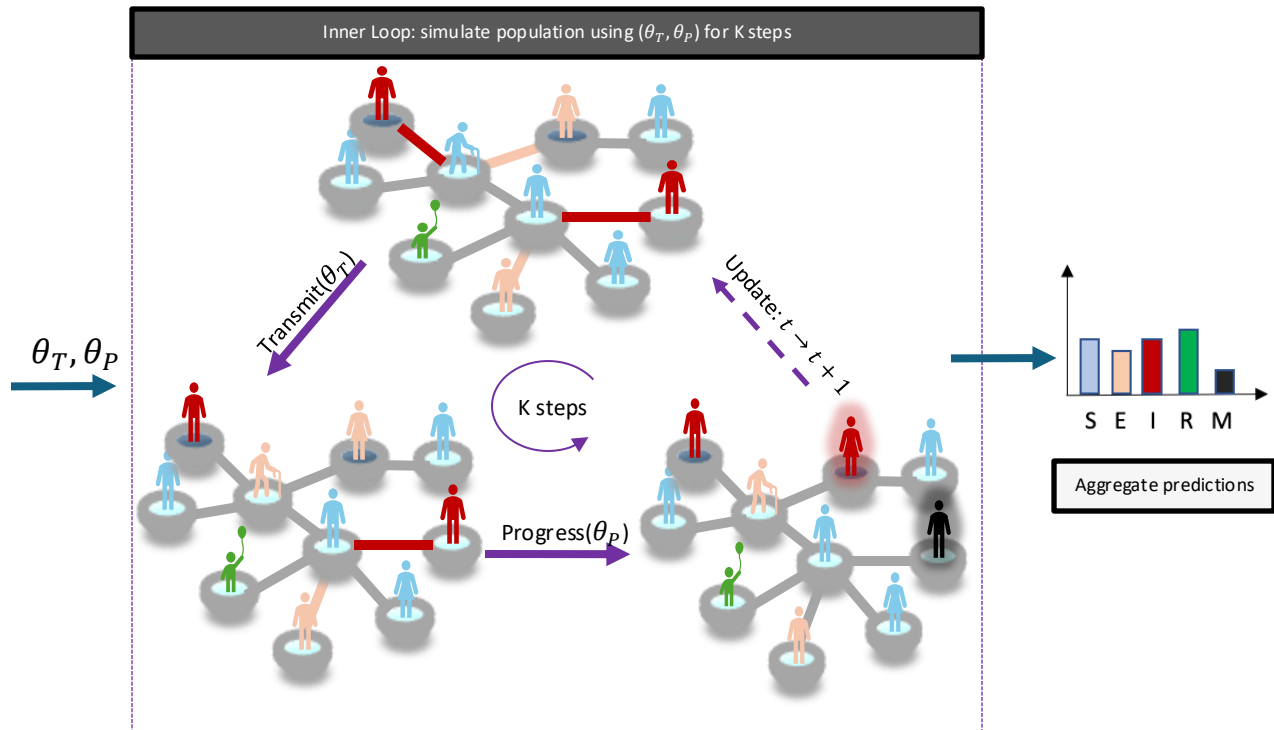
```
# agent_"torch" works seamlessly with the pytorch API
from torch.optim import SGD

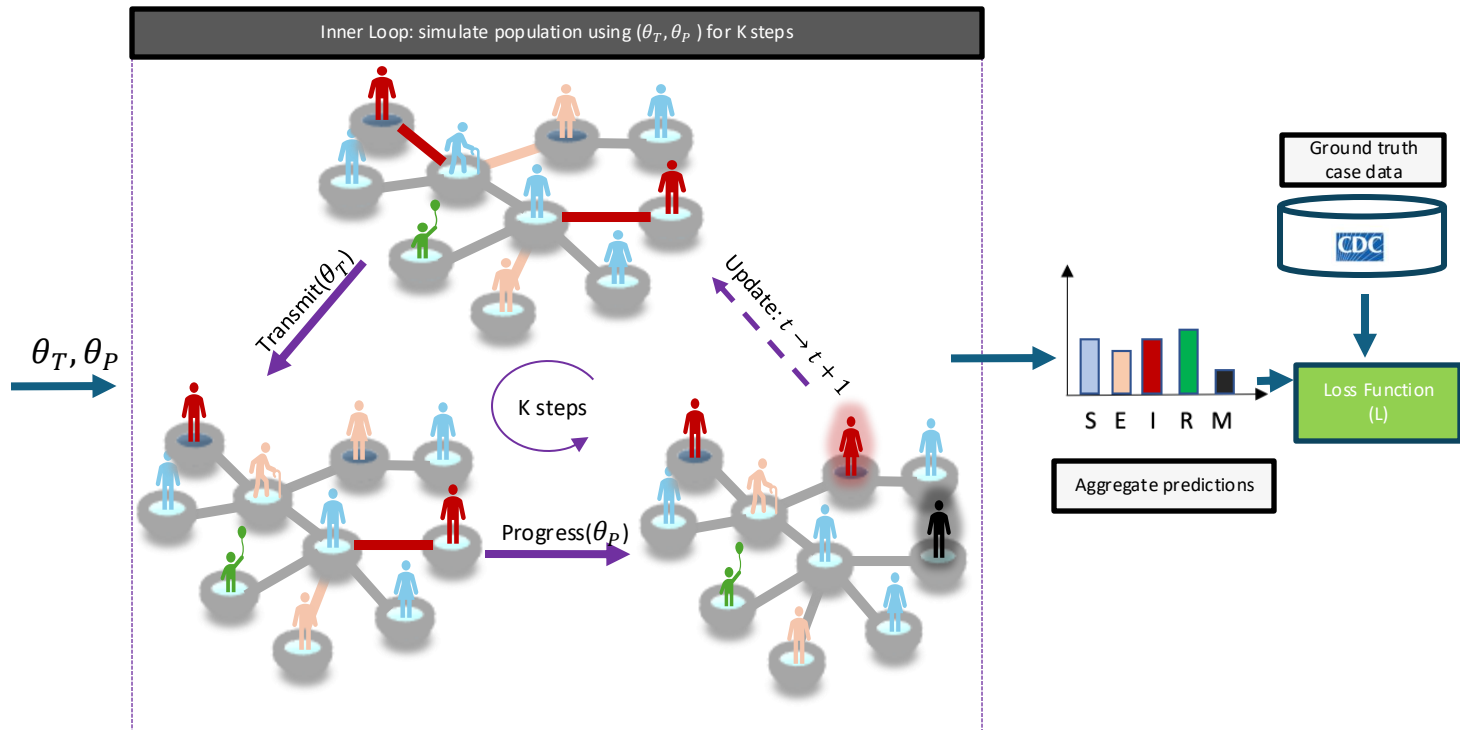
nn = compose_nn()

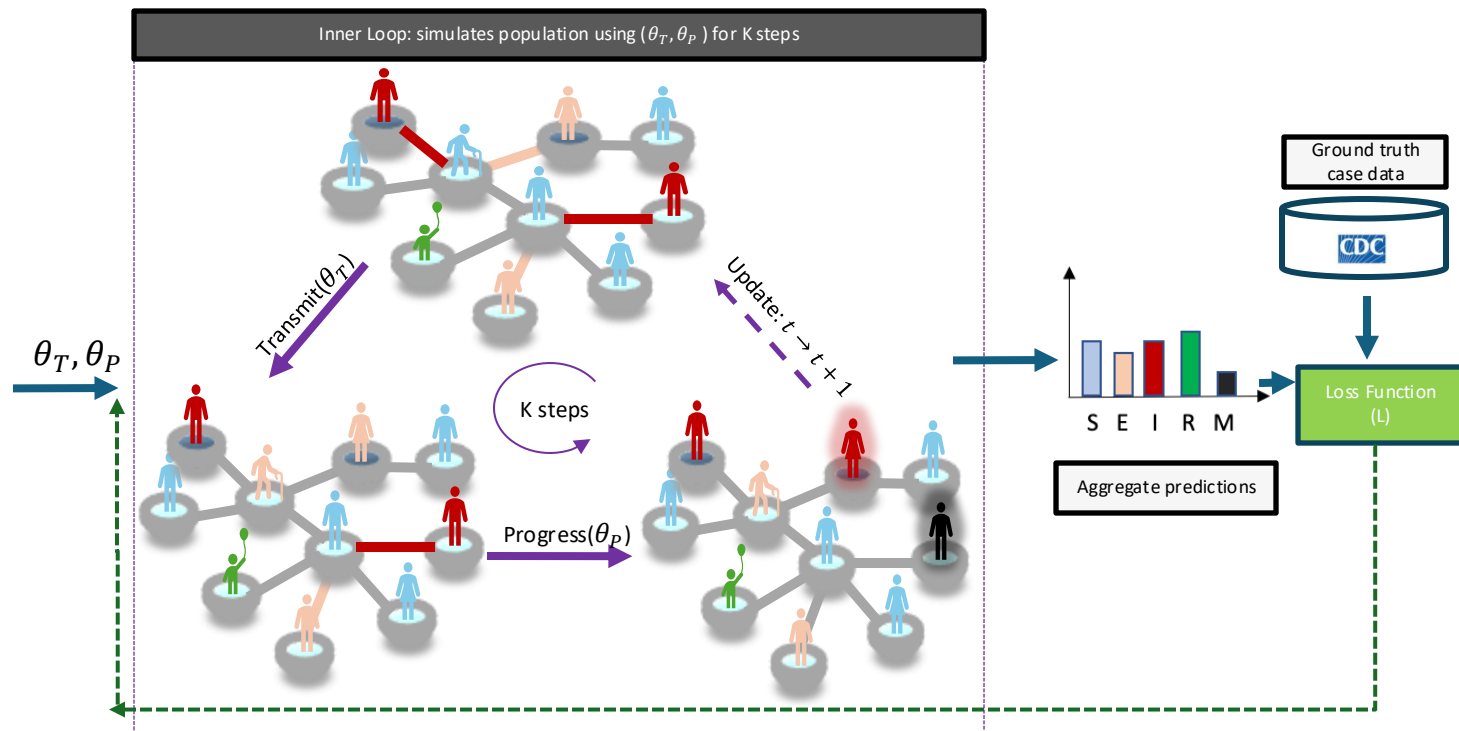
for i in range(n_epochs):
    parameters = compose_nn.forward()

    simulation.step(params=parameters)
    cases, employment = simulation.predict()

    simulation.optimize(SGD)
    simulation.reset()
```

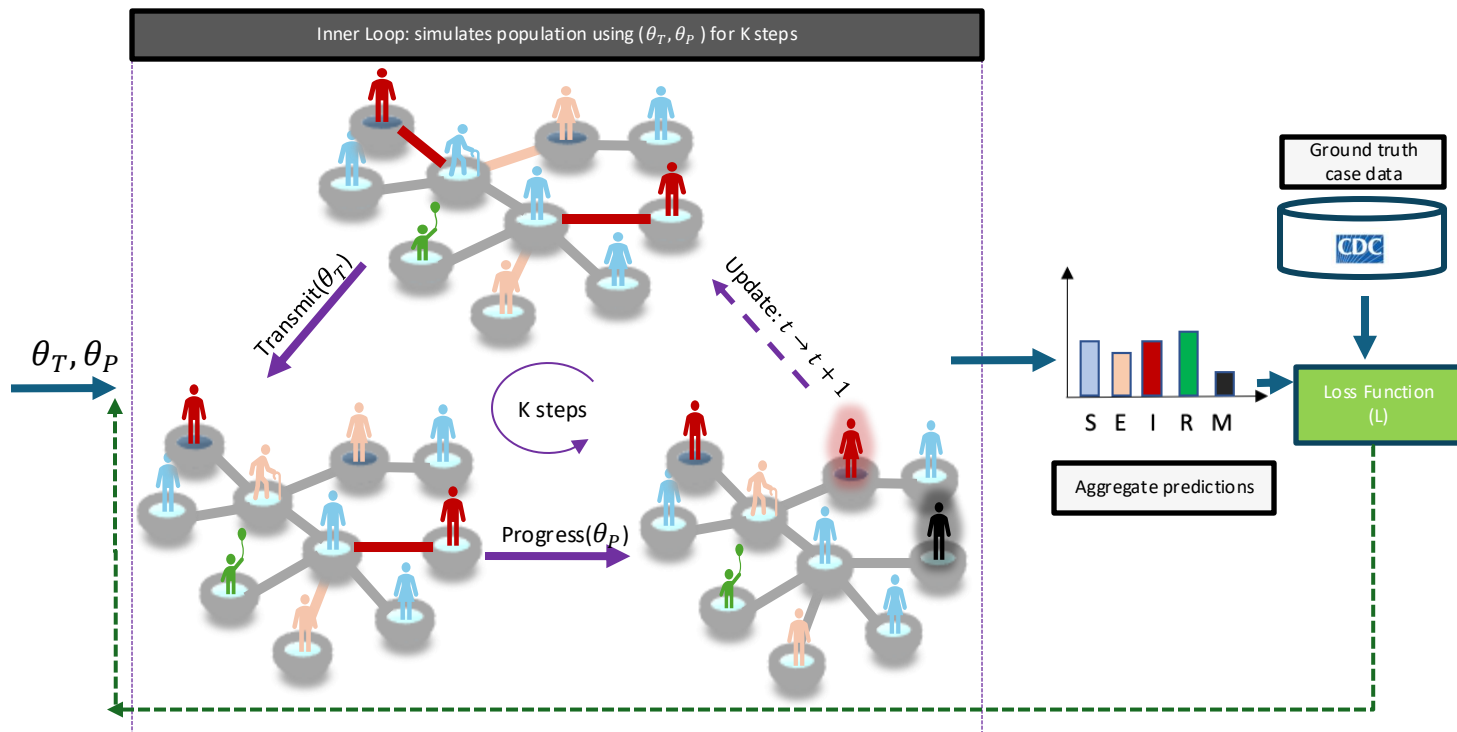






$$\theta_T = \theta_T - \alpha \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \theta_T}$$

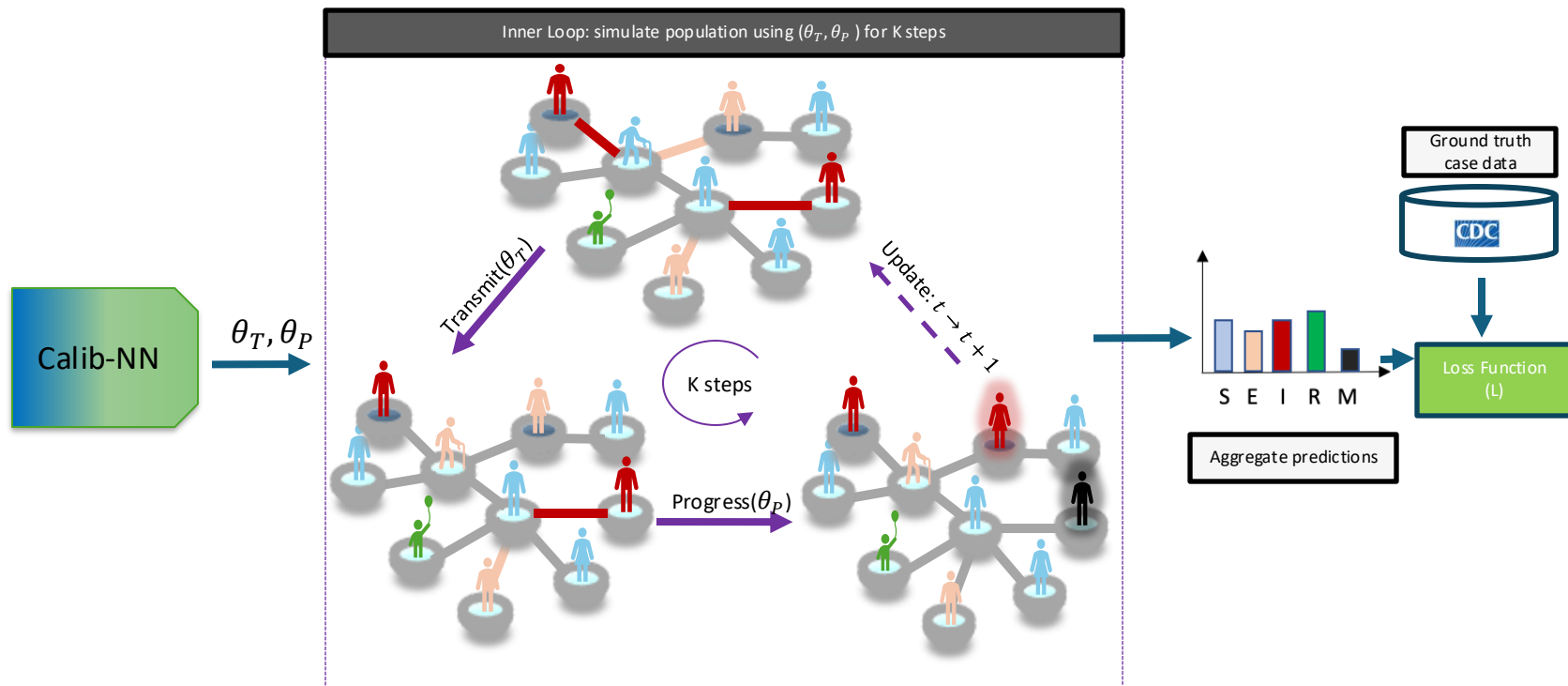
$$\theta_P = \theta_P - \alpha \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \theta_P}$$



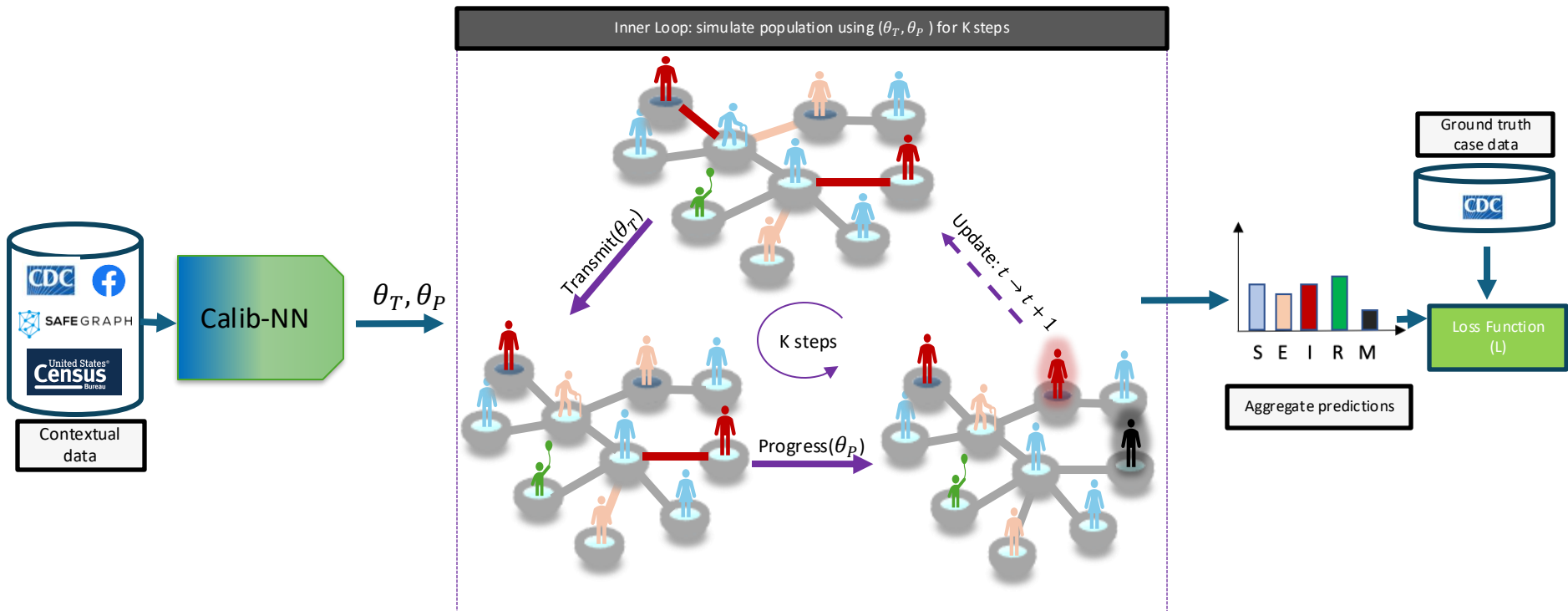
$$\theta_T = \theta_T - \alpha \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \theta_T}$$

$$\theta_P = \theta_P - \alpha \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \theta_P}$$

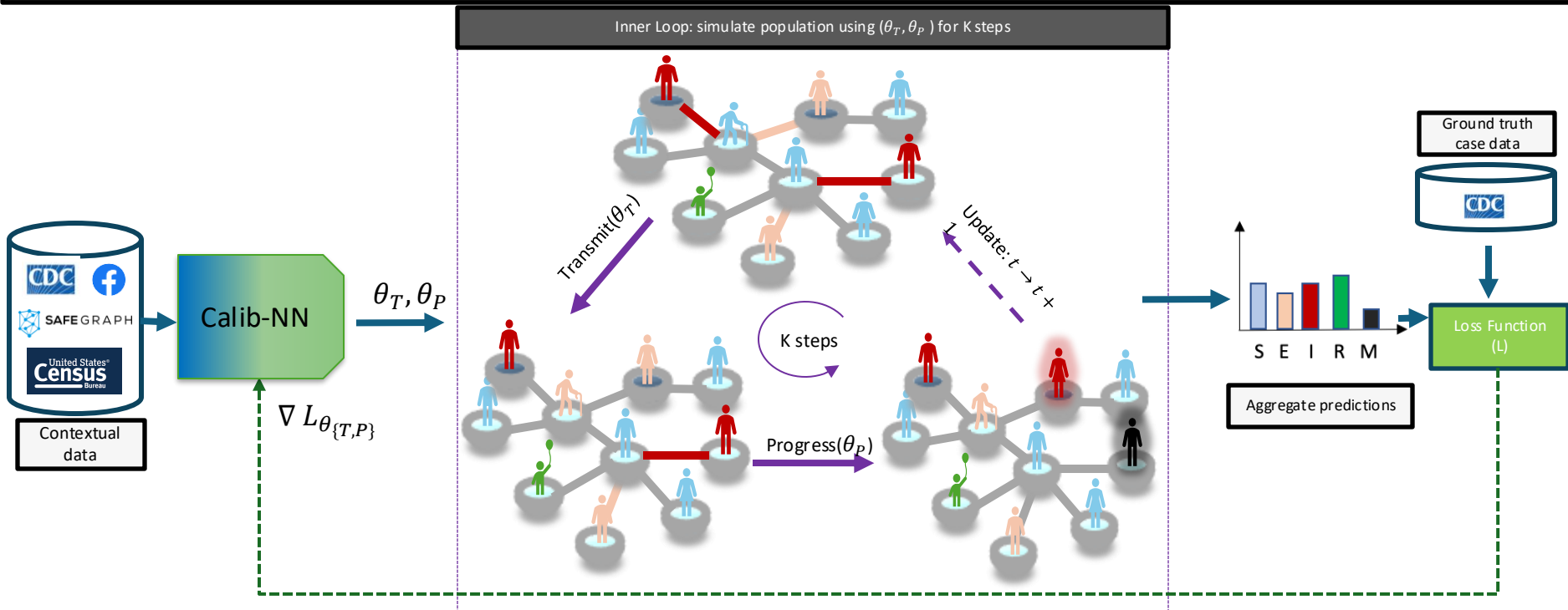
**Mode 1:** Calibrate **parameters** with gradient descent (c-GRADABM)





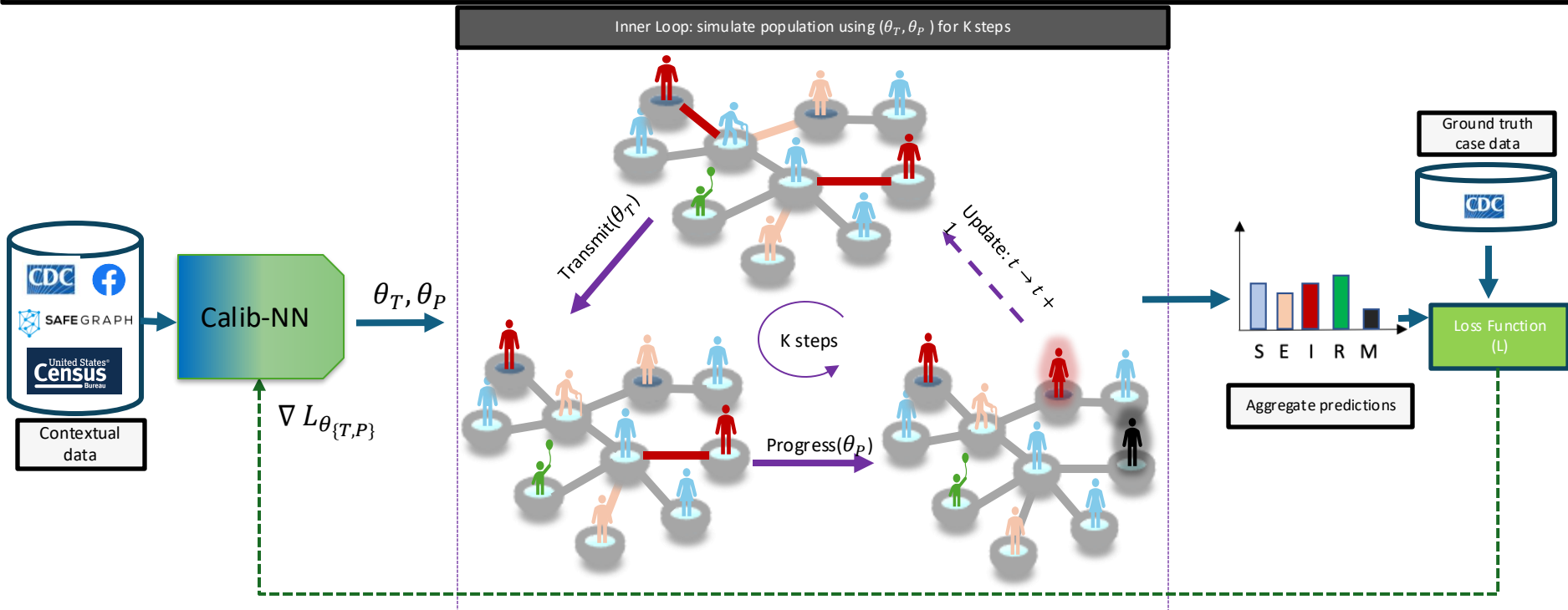


Outer Loop: Calib-NN predict infection parameters  $(\theta_T, \theta_P)$  for population simulation and is optimized using end-to-end gradient flow



$$\phi = \phi - \alpha \frac{\partial \mathcal{L}(\hat{y}, y; (\theta_T^t, \theta_P^t))}{\partial \phi},$$

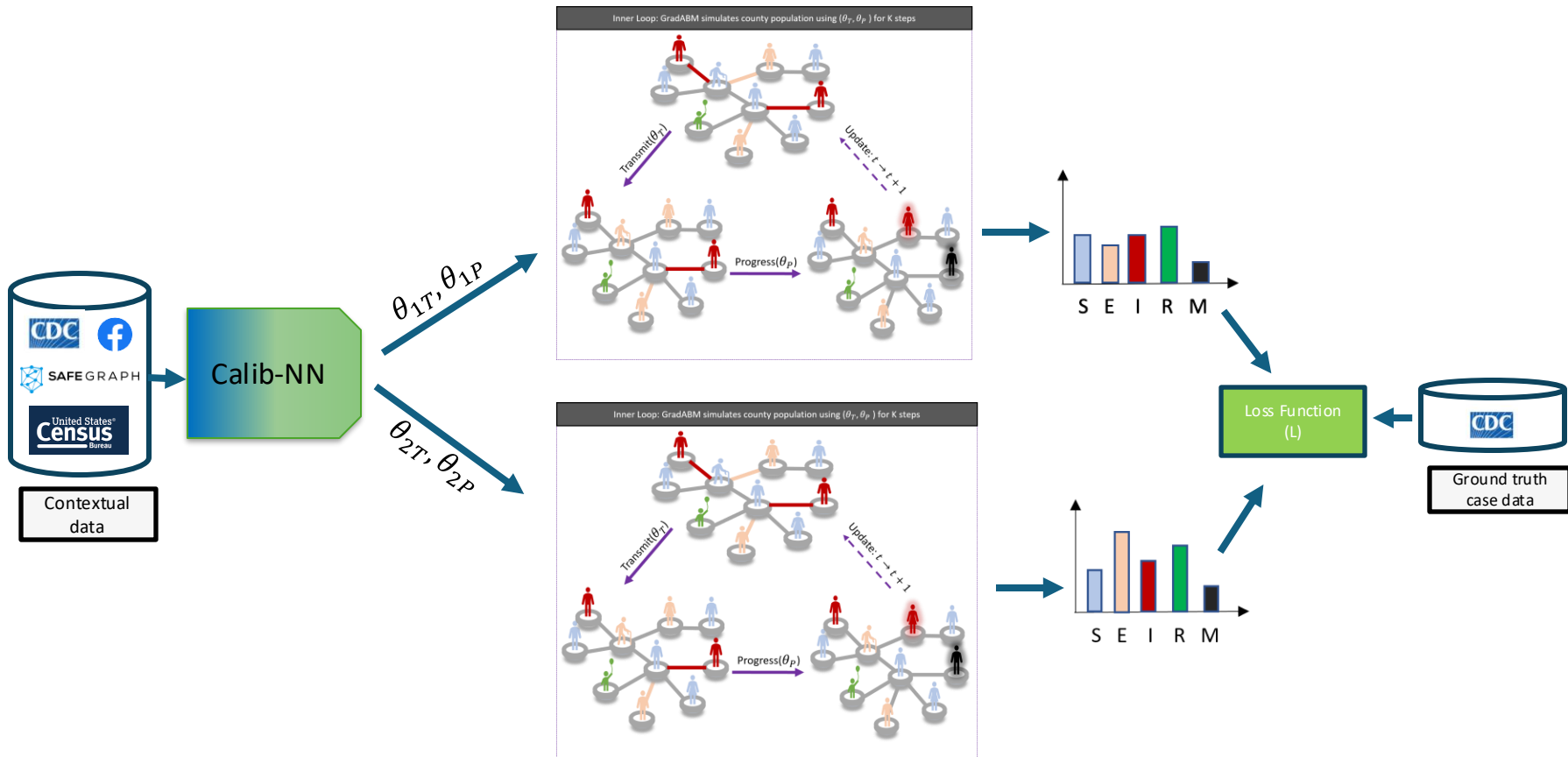
Outer Loop: Calib-NN predict infection parameters  $(\theta_T, \theta_P)$  for population simulation and is optimized using end-to-end gradient flow



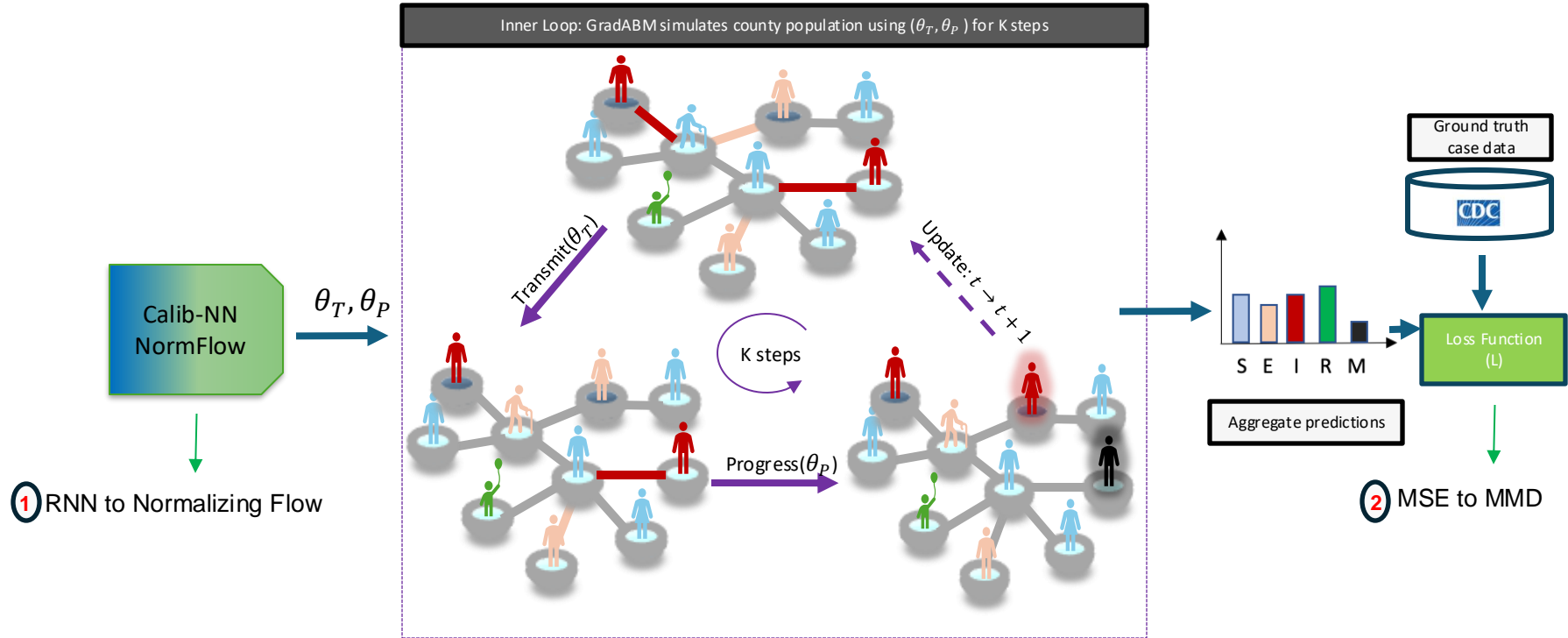
**Mode 2:** Calibrate generator function with gradient descent (dc-GRADABM)

$$\phi = \phi - \alpha \frac{\partial \mathcal{L}(\hat{y}, y; (\theta_T^t, \theta_P^t))}{\partial \phi},$$

# Multi-task learning over simulation environments, reduce overfitting



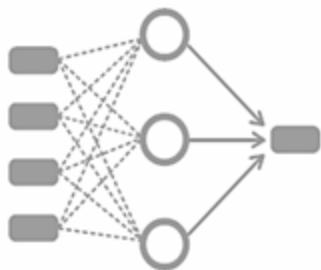
# Estimate posteriors over simulation parameters



[github.com/AgentTorch/AgentTorch](https://github.com/AgentTorch/AgentTorch)

`pip install agent-torch`





Dynamic interactions

Multi-scale  
Stochastic

Differentiability

## Compose with ODEs

```
from chirho.dynamical.handlers.solver import TorchDiffEq
from chirho.dynamical.ops import simulate

simulation = envs.create(
    model=epidemiology,
    populations=NYC,
    archetype={'immune_dynamics':
               archetype.ode(chiro_ode, eval_times)
    })

simulation.execute()
```

$$\mathbf{z}_i(t+1) = f\left(\mathbf{z}_i(t), \bigoplus_{j \in \mathcal{N}_i(t)} M_{ij}(t), \boldsymbol{\theta}, \ell(\cdot | \mathbf{z}_i(t))\right)$$



### Q3: what about the data?

coarse-grained, noisy and stale data  
limited granularity from privacy not scarcity?

$$\mathbf{z}_i(t+1) = f \left( \mathbf{z}_i(t), \bigoplus_{j \in \mathcal{N}_i(t)} M_{ij}(t), \boldsymbol{\theta}, \ell(\cdot | \mathbf{z}_i(t)) \right)$$

- Interaction traces
- Vaccination, disease status

### Q3: what about the data?

coarse-grained, noisy and stale data  
limited granularity from privacy not scarcity?

$$\mathbf{z}_i(t+1) = f \left( \mathbf{z}_i(t), \bigoplus_{j \in \mathcal{N}_i(t)} M_{ij}(t), \boldsymbol{\theta}, \ell(\cdot | \mathbf{z}_i(t)) \right)$$

- Interaction traces
- Vaccination, disease status

decentralize the simulation?

# Large Population Models



Expressive agents

Adaptive  
Heterogeneous

LLM Archetypes

AAMAS'23; arxiv'24



Dynamic interactions

Multi-scale  
Stochastic

Differentiability

BMJ'21; AAMAS'23;  
AAMAS'24 (Best paper runner-up)



Data-driven analysis

Decentralized  
Sensitive

Secure MPC

ICML-W'22 (Best paper);  
AAMAS'24; Nature Medicine'25



Data-driven analysis

Decentralized  
Sensitive

Secure MPC

## Backprop through the “real-world”

```
from agent_torch.core import decentralize

# map simulation to real protocol
real_sim = decentralize(
    simulation,
    protocol="ble_contact_tracing")

real_sim.deploy()

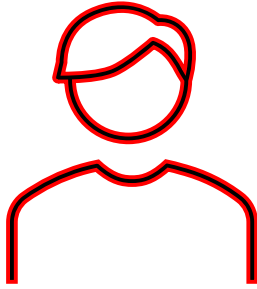
real_sim.sync()
```

Compose with Physical protocols

Estimate simulation gradients via MPC

aside: what is secure multi-party computation?

Alice



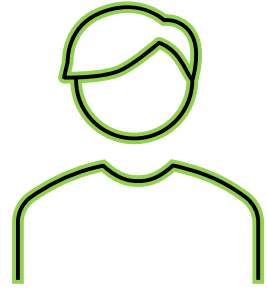
2

Bob



3

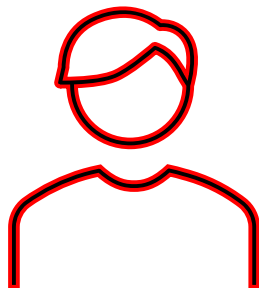
Charlie



5

n=11

Alice



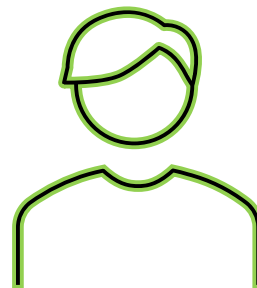
2

Bob



3

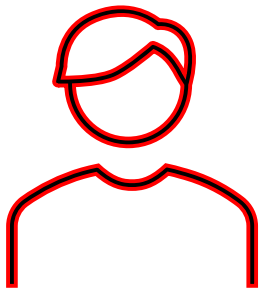
Charlie



5

n=11

Alice



2

7 + 5 + 1

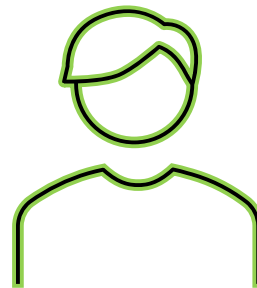
Bob



3

2 + 0 + 1

Charlie



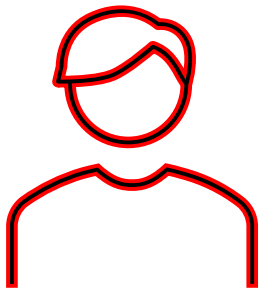
5

3 + 1 + 1



n=11

Alice



2

7 + 5 + 1

7 + 2 + 3

Bob

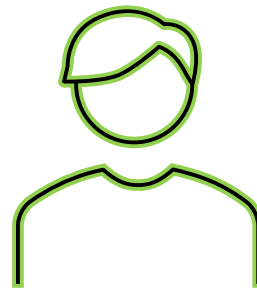


3

2 + 0 + 1

5 + 0 + 1

Charlie



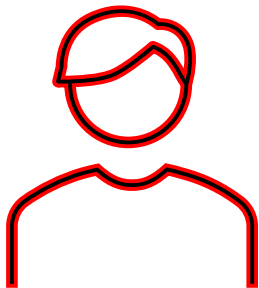
5

3 + 1 + 1

1 + 1 + 1

n=11

Alice



2

7 + 5 + 1

7 + 2 + 3

1

Bob



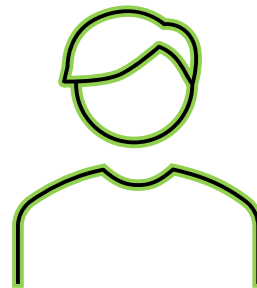
3

2 + 0 + 1

5 + 0 + 1

6

Charlie



5

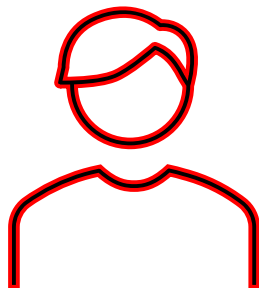
3 + 1 + 1

1 + 1 + 1

3

n=11

Alice



2

$$7 + 5 + 1$$

$$7 + 2 + 3$$

1

$$1 + 6 + 3$$

Bob



3

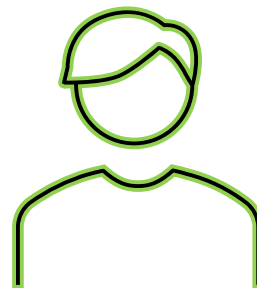
$$2 + 0 + 1$$

$$5 + 0 + 1$$

6

$$1 + 6 + 3$$

Charlie



5

$$3 + 1 + 1$$

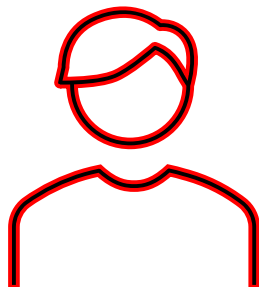
$$1 + 1 + 1$$

3

$$1 + 6 + 3$$

n=11

Alice



2

$$7 + 5 + 1$$

$$7 + 2 + 3$$

1

answer

10

Bob



3

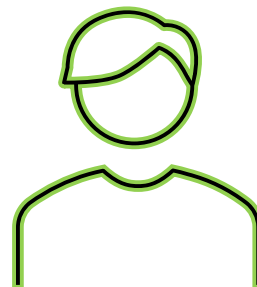
$$2 + 0 + 1$$

$$5 + 0 + 1$$

6

10

Charlie



5

$$3 + 1 + 1$$


$$1 + 1 + 1$$

3

10

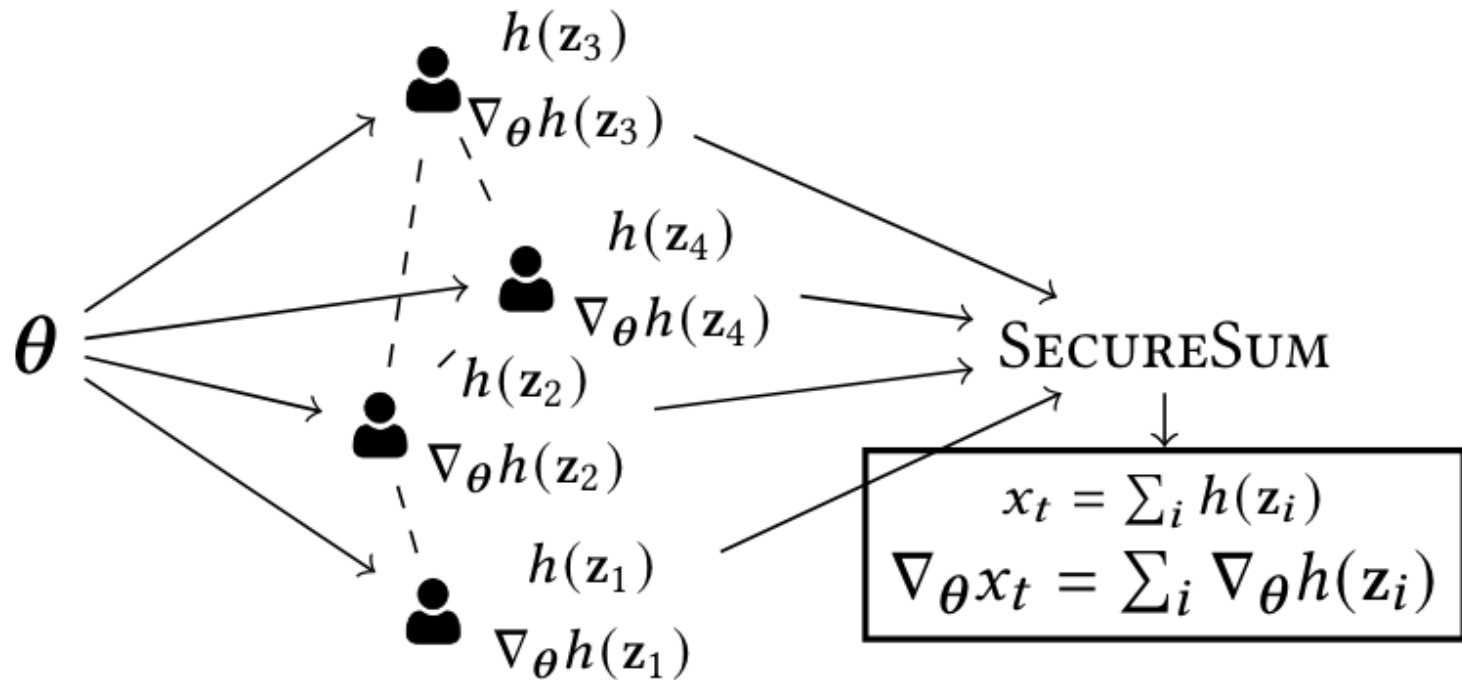
$$\mathbf{z}_i(t+1) = f \left( \mathbf{z}_i(t), \bigoplus_{j \in \mathcal{N}_i(t)} M_{ij}(t), \boldsymbol{\theta}, \ell(\cdot | \mathbf{z}_i(t)) \right)$$

additive secret sharing

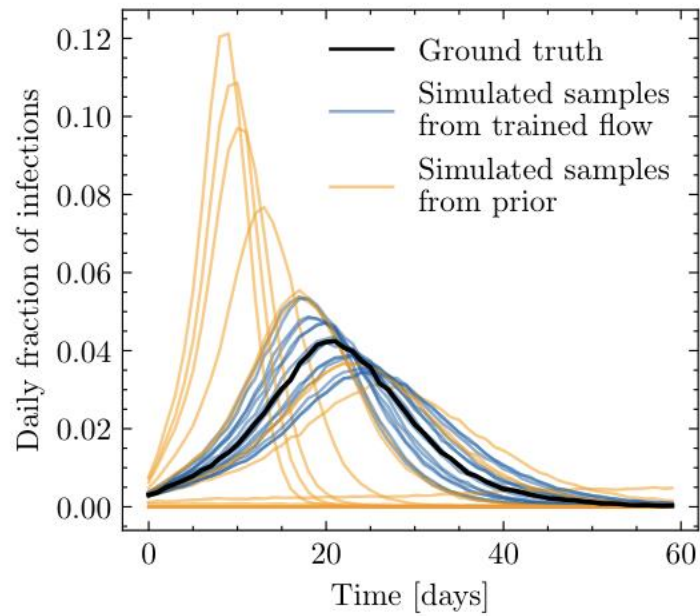
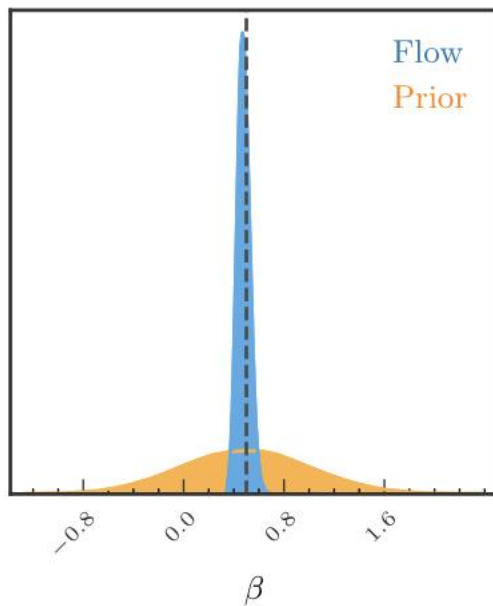


securely estimate  $z_i(t+1)$  and  $dz/d\theta$  on agent device

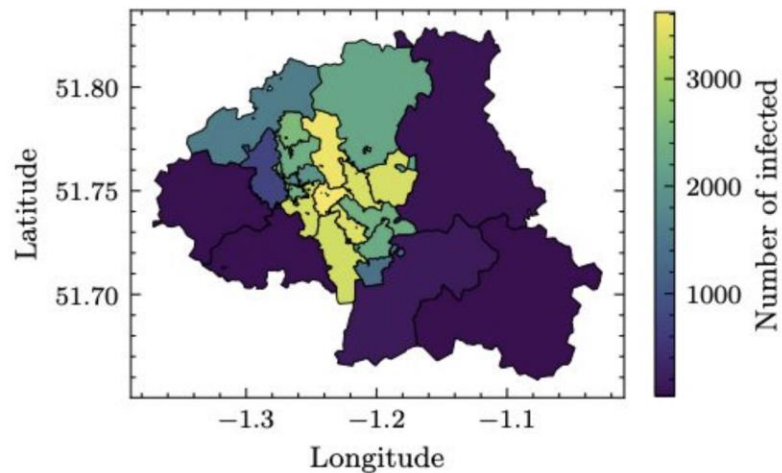
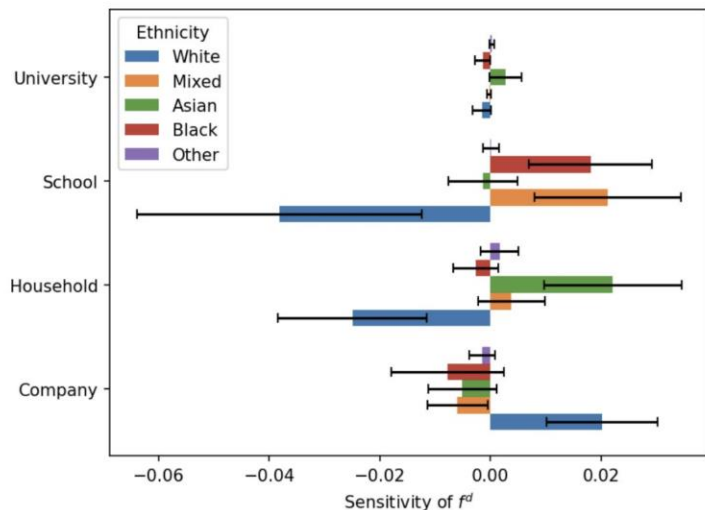
## Aggregate Message and Calibration Gradient



Calibrate disease parameters without leaking an agent's state or interaction trace



Analyze dynamics without leaking individual disease, demographic or geo-location







Data-driven analysis

Decentralized  
Sensitive

Secure MPC

## Backprop through the “real-world”

```
from agent_torch.core import decentralize

# map simulation to real protocol
real_sim = decentralize(
    simulation,
    protocol="ble_contact_tracing")

real_sim.deploy()

real_sim.sync()
```

Compose with Physical protocols

Estimate simulation gradients via MPC

# Large Population Models

[lpm.media.mit.edu/research.pdf](http://lpm.media.mit.edu/research.pdf)

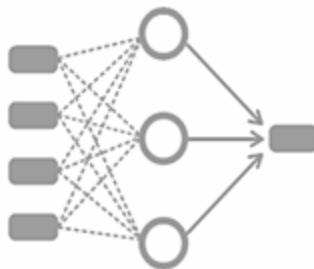


**Expressive agents**

Adaptive  
Heterogeneous

**LLM Archetypes**

AAMAS'23; arxiv'24



**Dynamic interactions**

Multi-scale  
Stochastic

**Differentiability**

BMJ'21; AAMAS'23;  
AAMAS'24 (Best paper runner-up)



**Data-driven analysis**

Decentralized  
Sensitive

**Secure MPC**

ICML-W'22 (Best paper);  
AAMAS'24; Nature Medicine'25

# Conclusion: Power of Large Population Models

- Scale and Speed: LPMs simulate millions of agents in seconds on standard hardware, enabling unprecedented insights into complex systems.
- Expressive Agents: By leveraging LLMs as behavioral archetypes, LPMs capture nuanced individual behaviors at scale.
- Closing Sim2Real Gap: Secure multi-party computation allows LPMs to incorporate real-world data without compromising privacy.
- Diverse Applications: From pandemic response to energy adoption and supply chain management, LPMs are already making global impact.
- Open-Source Accessibility: AgentTorch democratizes LPM technology, making it available to researchers and policymakers worldwide.

[github.com/AgentTorch/AgentTorch](https://github.com/AgentTorch/AgentTorch)

[lpm.media.mit.edu/join](http://lpm.media.mit.edu/join)

[github.com/AgentTorch/AgentTorch](https://github.com/AgentTorch/AgentTorch)

`pip install agent-torch`

