Look At It This Way: Relational Structural Alignment in Multi-agent Emergent Language

Washington Garcia (UF)

Kevin Butler (UF)

Scott Clouse (AFRL/ACT3)

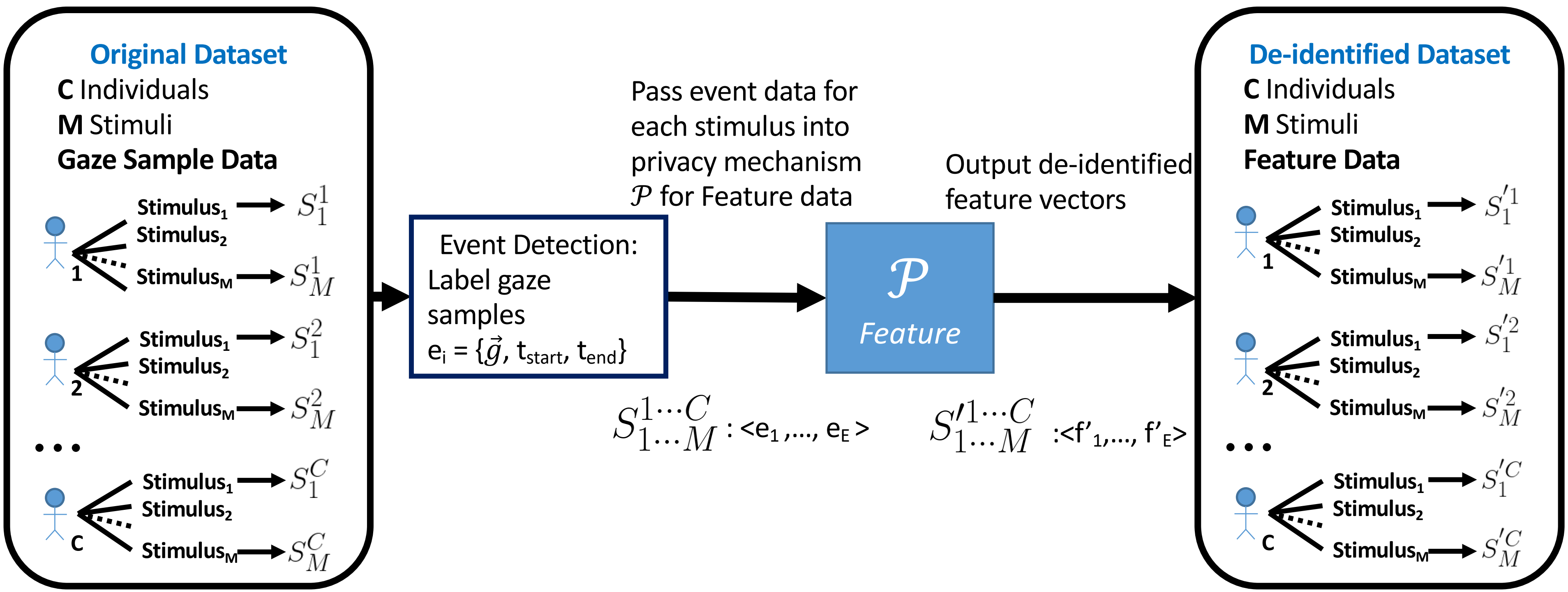
- Topics:
  - Privacy-preserving eye-tracking (ETRA'22)
  - FHE for autonomous agents (work in progress)
  - Multi-agent emergent language (submitted to NAACL'22)

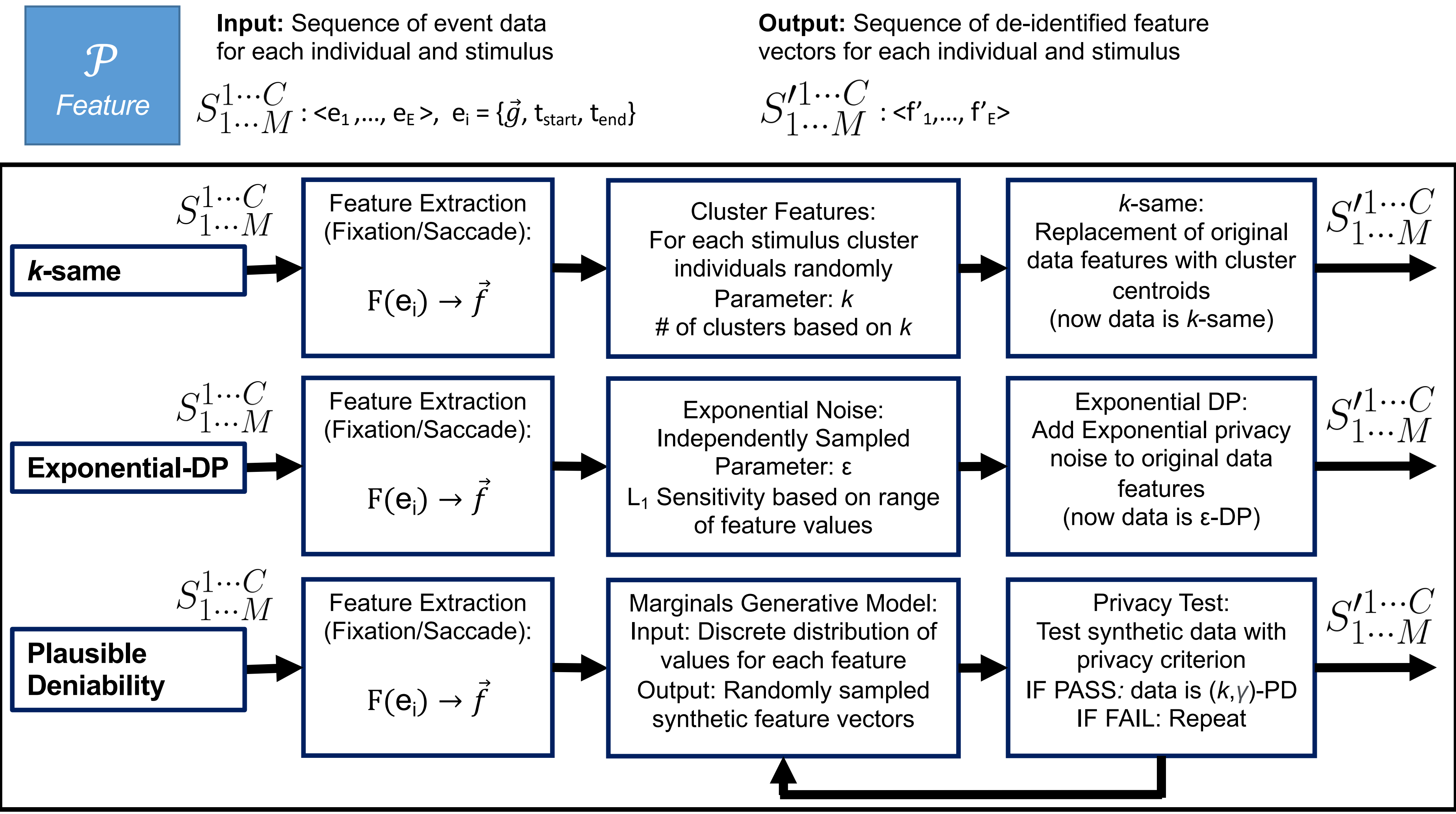
- Collaborators (UF unless otherwise noted):
  - Caroline Fedele, Aaditya Prakash, Brendan David-John, Eakta Jain, Scott Clouse (AFRL/ACT3)

Human-Machine Interaction

- Current state of deployed UASs currently involve significant human interaction (<=L3 autonomy)
- Autonomous systems will potentially learn from simulation data informed by human interaction
- Augmented reality (AR) systems can assist near-term operations while virtual reality (VR) simulators are standard for training
- What risks to privacy are incurred in these systems?

Feature Pipeline





$\mathcal{P}$
Feature
Input: Sequence of event data for each individual and stimulus
$S_{1\cdots M}^{1\cdots C}$: <e$_1$,..., e$_E$>, e$_i$ = {$\vec{g}$, t$_{start}$, t$_{end}$}
Output: Sequence of de-identified feature vectors for each individual and stimulus
$S'^{1\cdots C}_{1\cdots M}$: <f'$_1$,..., f'$_E$>
$S_{1\cdots M}^{1\cdots C}$
***k*-same**
Feature Extraction (Fixation/Saccade):
$F(e_i) \rightarrow \vec{f}$
Cluster Features:
For each stimulus cluster individuals randomly
Parameter: *k*
# of clusters based on *k*
k-same:
Replacement of original data features with cluster centroids
(now data is *k*-same)
$S'^{1\cdots C}_{1\cdots M}$
$S_{1\cdots M}^{1\cdots C}$
Exponential-DP
Feature Extraction (Fixation/Saccade):
$F(e_i) \rightarrow \vec{f}$
Exponential Noise:
Independently Sampled
Parameter: ε
L_1 Sensitivity based on range of feature values
Exponential DP:
Add Exponential privacy noise to original data features
(now data is ε-DP)
$S'^{1\cdots C}_{1\cdots M}$
$S_{1\cdots M}^{1\cdots C}$
Plausible Deniability
Feature Extraction (Fixation/Saccade):
$F(e_i) \rightarrow \vec{f}$
Marginals Generative Model:
Input: Discrete distribution of values for each feature
Output: Randomly sampled synthetic feature vectors
Privacy Test:
Test synthetic data with privacy criterion
IF PASS: data is (*k*,*γ*)-PD
IF FAIL: Repeat
$S'^{1\cdots C}_{1\cdots M}$

Original Marginals

Dataset: Feature Distributions

Sample Features Independently

S Synthetic Feature Vectors

Synthetic Data

Test Each Feature Vector

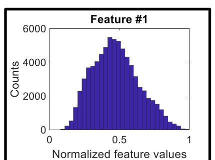
Real Dataset

Privacy Test: k,γ

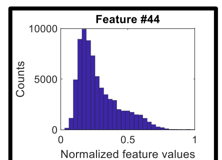
PASS

Synthetic k,γ-PD Dataset

Re-sample Synthetic Dataset: Preserve # of Vectors per Individual per stimulus based on Real Dataset



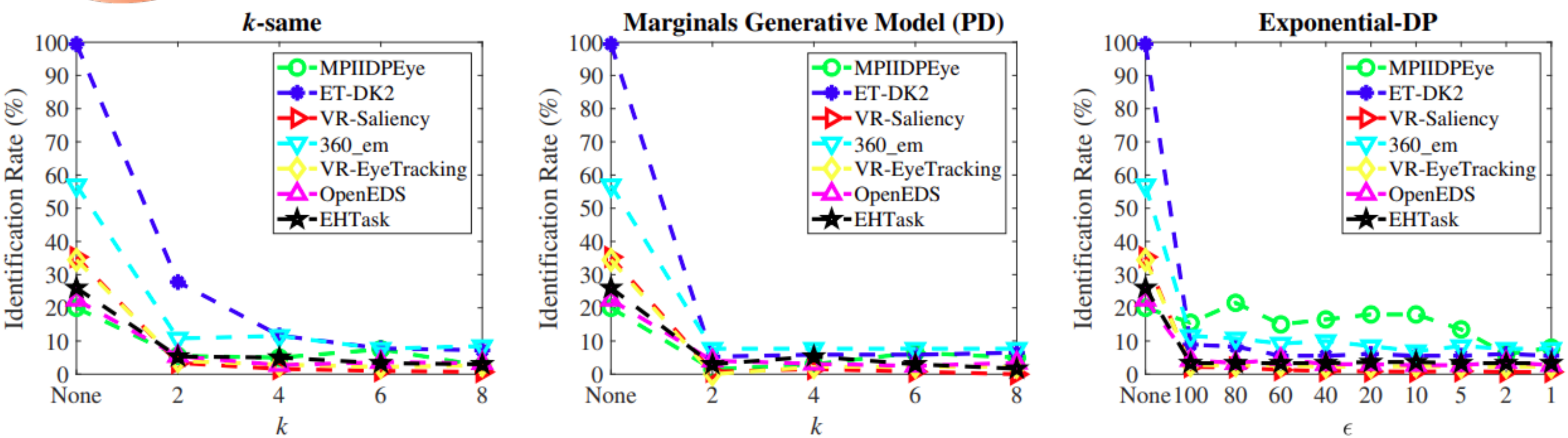
...



Uniformly Sample Synthetic Rows from Feature Distributions

Synthetic	f_1	f_2	f_3	...	f_F
$\vec{x}_{m,p,1}$	~	~	~	~	~
$\vec{x}_{m,p,2}$	~	~	~	~	~
$\vec{x}_{m,p,3}$	~	~	~	~	~
...	~	~	~	~	~
$\vec{x}_{m,p,i}$	~	~	~	~	~

Evaluation



- Generative model and exponential-DP rapidly reduce identification rate
- Utility of k-same is higher than both of these mechanisms
- Plausible deniability of marginals generative model provides formal guarantees about contributing inputs
- Next step opportunities: applying synthetic data from generative modeling in agent environments

Fully Homomorphic Encryption in Multi-Agent Environments

Caroline Fedele and Kevin Butler

Fully homomorphic encryption (FHE)

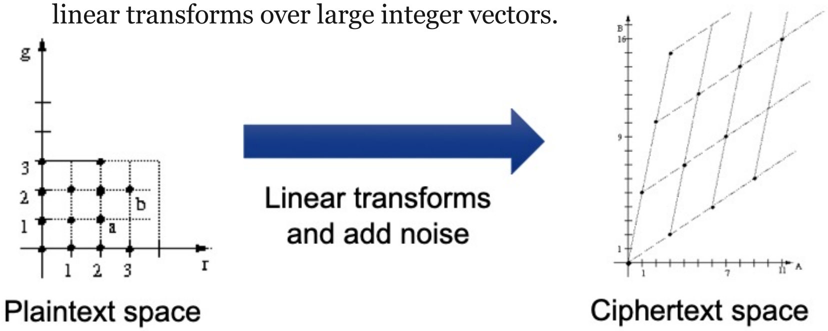
- Assuring **data privacy** during computation
  - allows for arbitrary operations on encrypted data
  - Lattice-based cryptosystem (LBC)
    - Hard problems used: closest/shortest vector problems, learning with errors (LWE)
    - Current schemes are IND-CPA secure
- Applications: cloud computing, machine learning, evaluation of private data (medical, financial, personal, IoT, etc.)

Recent Advancements:

- key/modulus switching – minimizes noise without requiring secret key or bootstrapping (BGV schemes)
- Improved FHE performance – approximate/continuous space operations (CKKS scheme)
  - uses parallel processing on GPUs, FPGAs, ASICs

Lattice Cryptography

Encryption/decryption are primarily composed of linear transforms over large integer vectors.



FHE vs. PHE

Partially HE (PHE)

- Allows arbitrary number of addition OR multiplication operations
- Efficient for specific/singular applications, lower overhead than FHE
  - handles limited class of low-deg polynomial functions/circuits
  - Pallier (addition), El Gamal/RSA (multiplication)
- Not post-quantum
  - Hard problems: DCRA, discrete logarithm, integer factorization respectively… all vulnerable to quantum and classical attacks
- Overall greater overhead if needed for various applications/data types, implement multiple different SHE/PHE schemes
- Not inherently boostrappable, no easy way to handle decryption function w/o exposing secret key

FHE

- Allows arbitrary number of addition AND multiplication operations
- Efficiency has grown significantly in recent years
- Modified SHE where decryption function is reduced enough for bootstrapping
  - bootstrapping, modulus switching
    - Necessary for large circuits/complex functions to reduce noise accumulated
    - Does not require secret key knowledge
- Post-quantum lattice-based crypto (LBC)
  - Hard problems: closest/shortest vector problem (CVP/SVP) and learning with errors (LWE)
- Allows for complex functions/operations needed in algorithms such as Dijkstra, ML and cloud computing operations

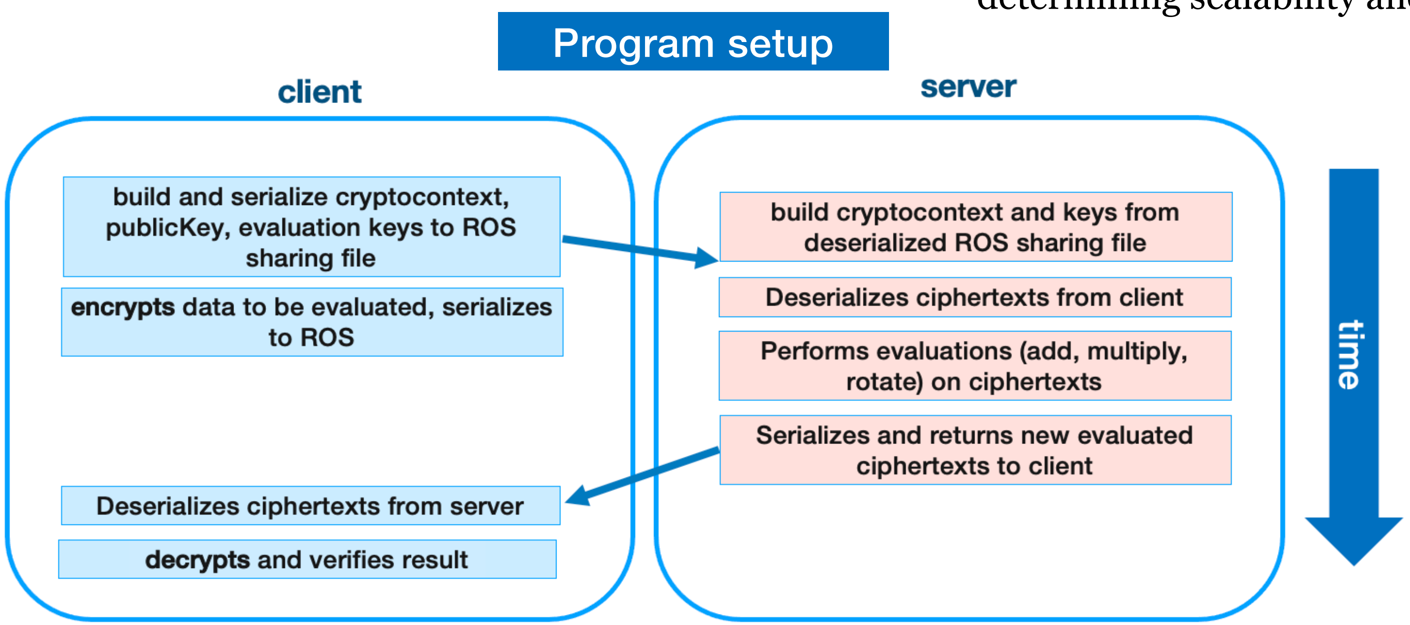
Goals and Methods

Goal: Implement FHE in contested environments, i.e. UAVs, satellites, other limited-resource systems

- secure robotic operations by integrating FHE into ROS operations
- secure outsourcing of crypto/computationally expensive tasks

Method:

- Palisade crypto toolkit for FHE, integrated with ROS software tested on Nvidia AI development boards
    - Server/client program testing various evaluations in two FHE schemes, determining scalability and overhead



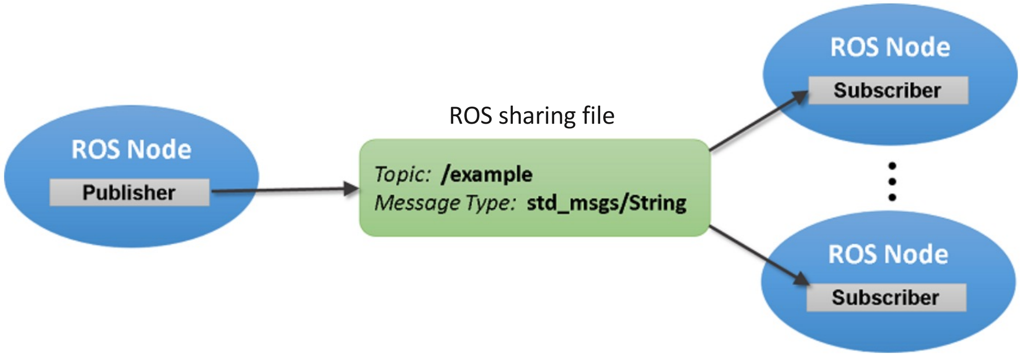
UF UNIVERSITY of FLORIDA · Duke UNIVERSITY · TEXAS The University of Texas at Austin · UC SANTA CRUZ

Implementation

Code sample (c++)

```
// SETUP: Client Node Activity
std::shared_ptr<rclcpp::Node> node = rclcpp::Node::make_shared("client");
rclcpp::Client<communication::srv::Arr>::SharedPtr client =
    node->create_client<communication::srv::Arr>("arr");

auto request = std::make_shared<communication::srv::Arr::Request>();

// STEP 1: Set CryptoContext
// Set the main parameters
int plaintextModulus = 65537;
double sigma = 3.2;
SecurityLevel securityLevel = HEStd_128_classic;
uint32_t depth = 2;
CFactory::ReleaseAllContexts();
// Instantiate the crypto context
CryptoContext<DCRTPoly> cryptoContext =
    CryptoContextFactory<DCRTPoly>::genCryptoContextBFVrns(
      plaintextModulus, securityLevel, sigma, 0, depth, 0, OPTIMIZED);

// Enable features that you wish to use
cryptoContext->Enable(ENCRYPTION);
cryptoContext->Enable(SHE);

// Sample Program: Step 3: Encryption
// First plaintext vector is encoded
std::vector<int64_t> vectorOfInts1 = {10, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
Plaintext plaintext1 = cryptoContext->MakePackedPlaintext(vectorOfInts1);
// Second plaintext vector is encoded
std::vector<int64_t> vectorOfInts2 = {3, 2, 1, 6, 15, 1, 2, 8, 9, 2, 0, 4};
Plaintext plaintext2 = cryptoContext->MakePackedPlaintext(vectorOfInts2);

std::cout << "Plaintext #1: " << plaintext1 << std::endl;
std::cout << "Plaintext #2: " << plaintext2 << std::endl;

// The encoded vectors are encrypted
auto ciphertext1 = cryptoContext->Encrypt(keyPair.publicKey, plaintext1);
auto ciphertext2 = cryptoContext->Encrypt(keyPair.publicKey, plaintext2);
std::cout << "The plaintexts have been encrypted." << std::endl;
```

ROS Publisher Subscriber Model



MathWorks

Overhead of objects in code

Object	Binary size (bytes)
plaintext	96
cryptocontext	2687
public key	396221
evalmult key	203295
ciphertext 1	396249
ciphertext 2	396249

UF UNIVERSITY of FLORIDA · Duke UNIVERSITY · TEXAS The University of Texas at Austin · UC SANTA CRUZ

Future Work

More computations

- Dijkstra's search algorithm
- optimize scheme used for specific UAV operations/data types

Outsource computationally-expensive tasks

- heavy cryptographic components
- particular data types, e.g. images

Testing platforms

- test programs on UAVs and other resource-constrained systems, e.g. satellites

Note: Quadratic optimization with PHE

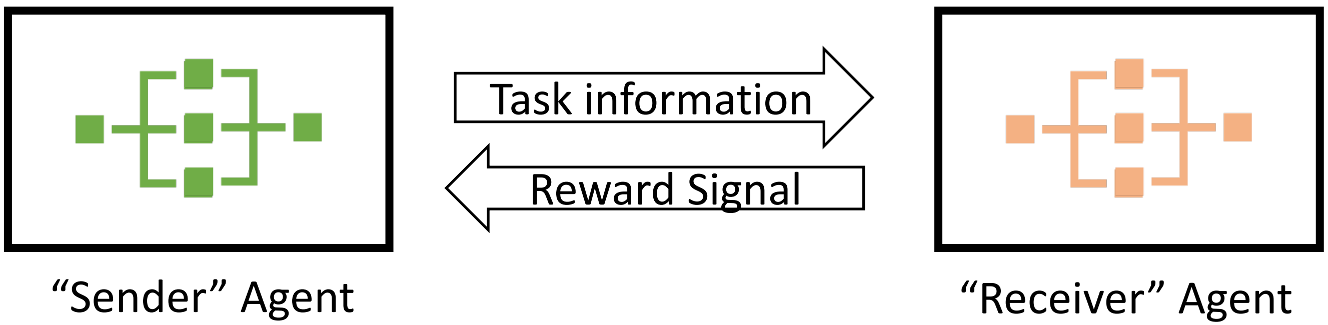
- Shoukry et al. (CDC 2016) developed a solution for a class of convex optimization problems using partially homomorphic encryption
  - Argument: FHE too slow to be practical
- Demonstrated using PHE to solve quadratic programs with linear inequality constraints
  - Paillier cryptosystem (addition only is possible)
- Open question: what types of optimization problems require both addition and multiplication and could benefit from advances in FHE?

Look At It This Way: Relational Structural Alignment in Multi-agent Emergent Language

Washington Garcia (UF)
Kevin Butler (UF)
Scott Clouse (AFRL/ACT3)

Motivation

Heterogenous multi-agent systems are capable of “language emergence” while learning to perform a task.

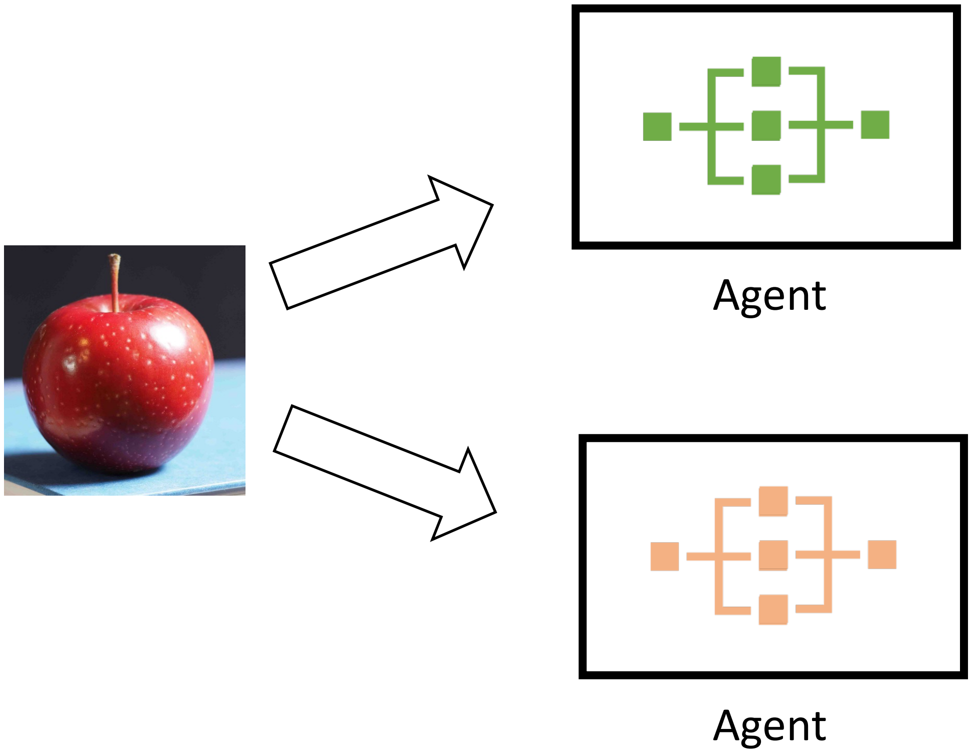


In this scheme, the communication channel acts as a “shim layer” between otherwise incompatible agents.

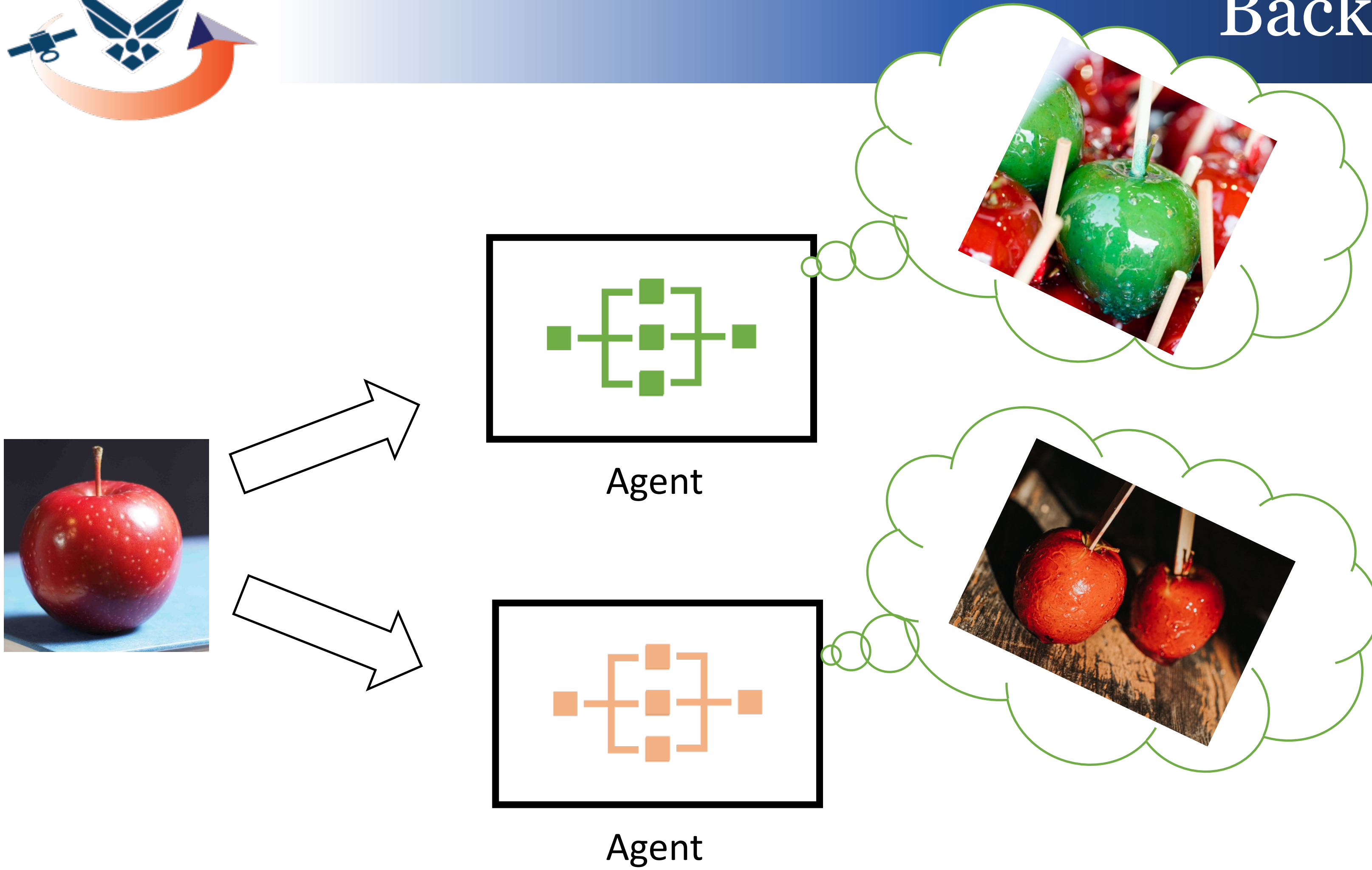
Main problem: learned communication can only be grounded to the environment

Goal: Ground messages to environment *and* agent knowledge

Background

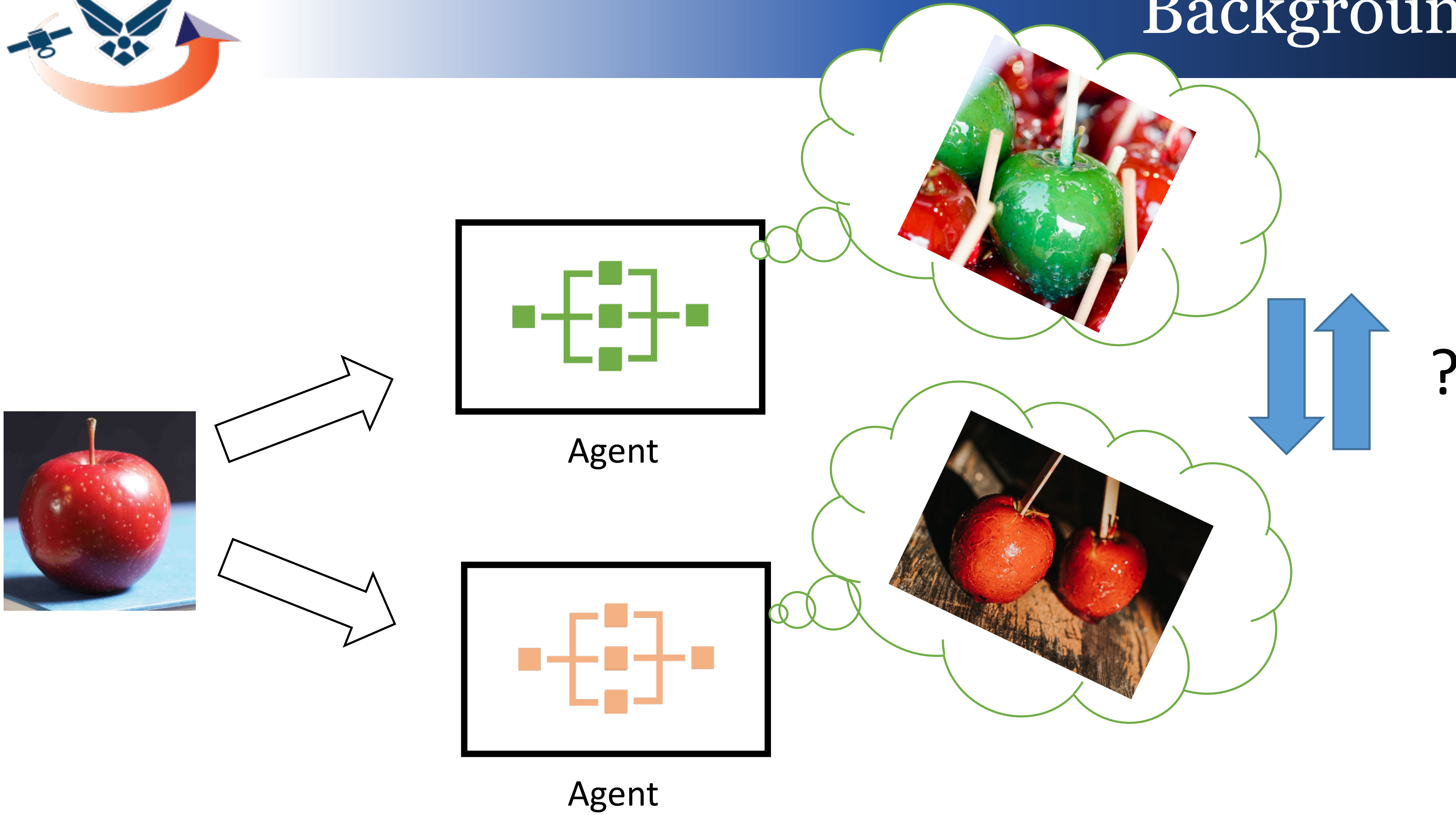


Background



Takeaway: Heterogenous agents = different reasoning process

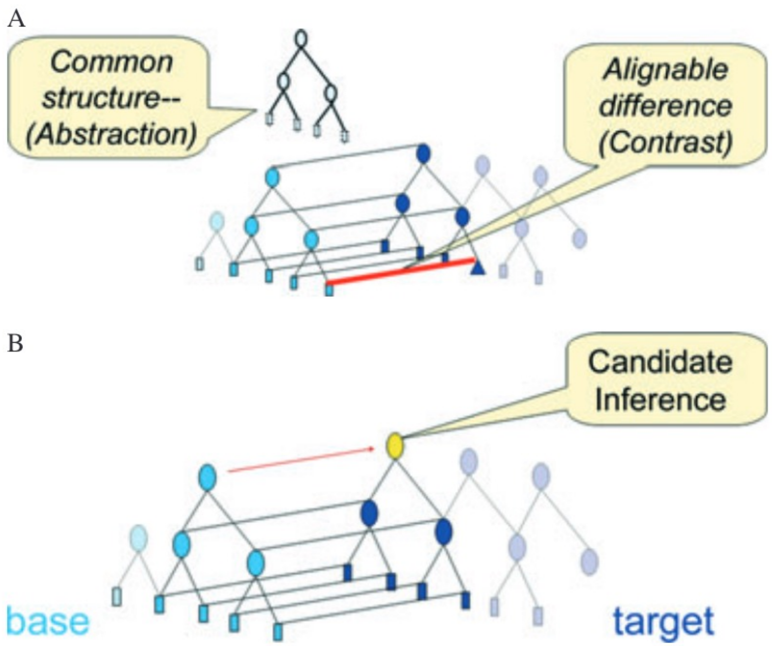
Background



Takeaway: Heterogenous agents = different reasoning process

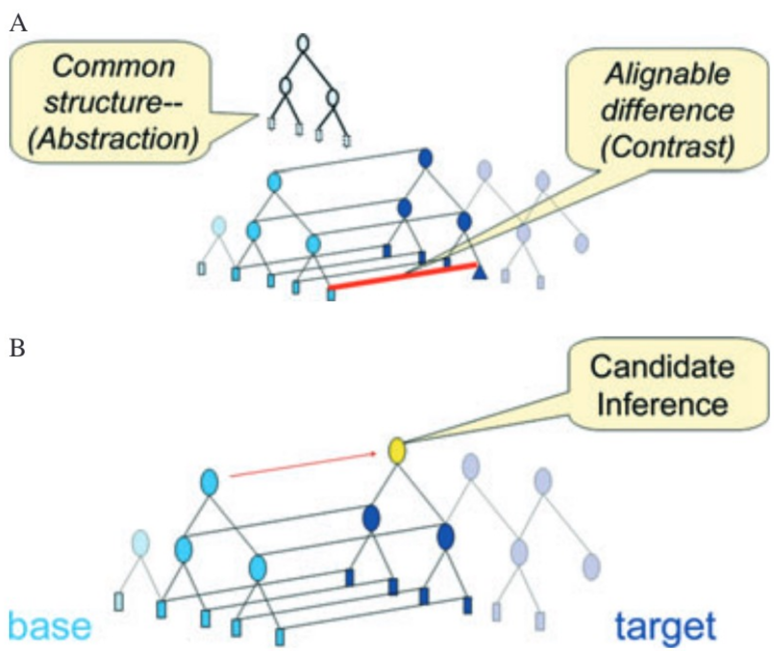
Is there cognitive theory view of this problem?

Gentner (2010) - Language supports relational cognition (analogical processing):



Is there cognitive theory view of this problem?

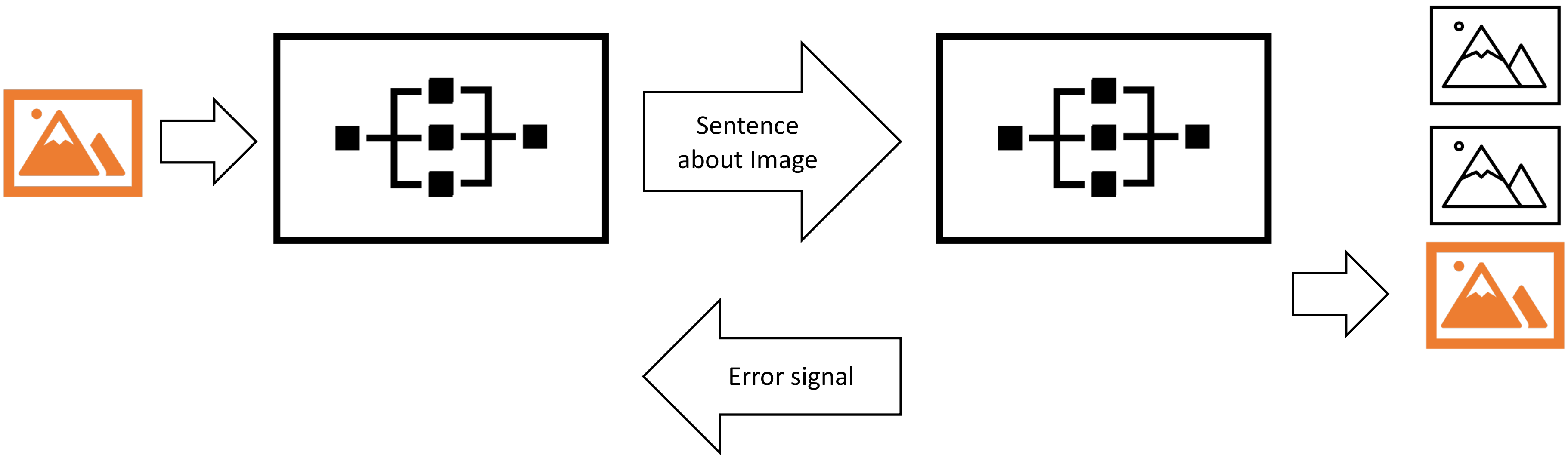
Gentner (2010) - Language supports relational cognition (analogical processing):



#1: How to build structurally-consistent representations?

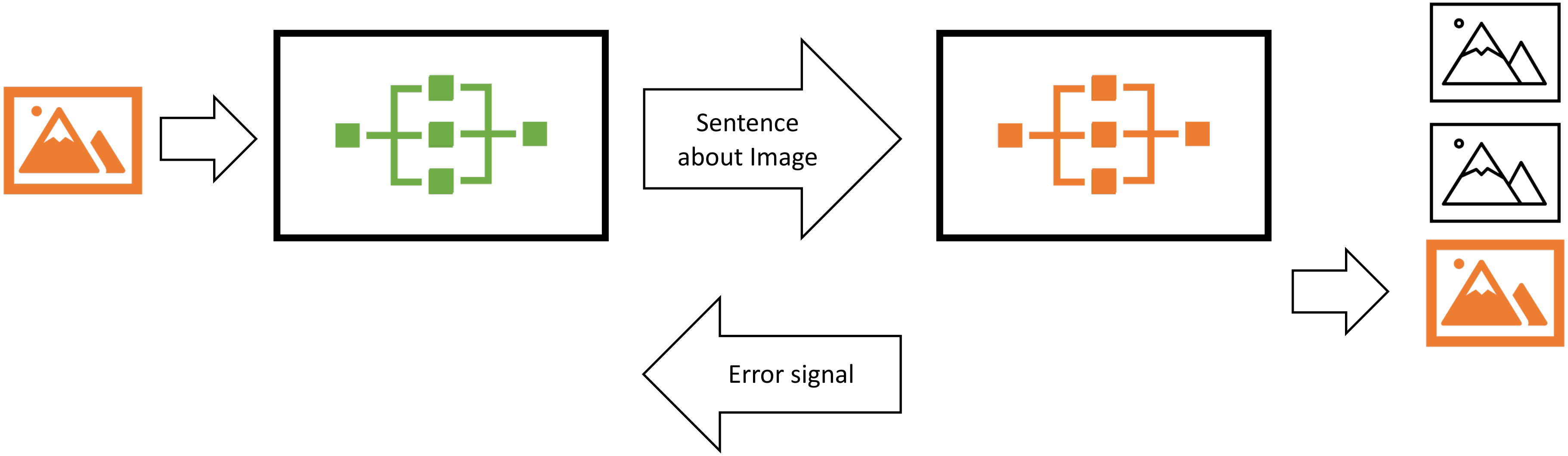
#2: How to align representations over a channel?

We consider two agents playing a Lewis Signaling Game:



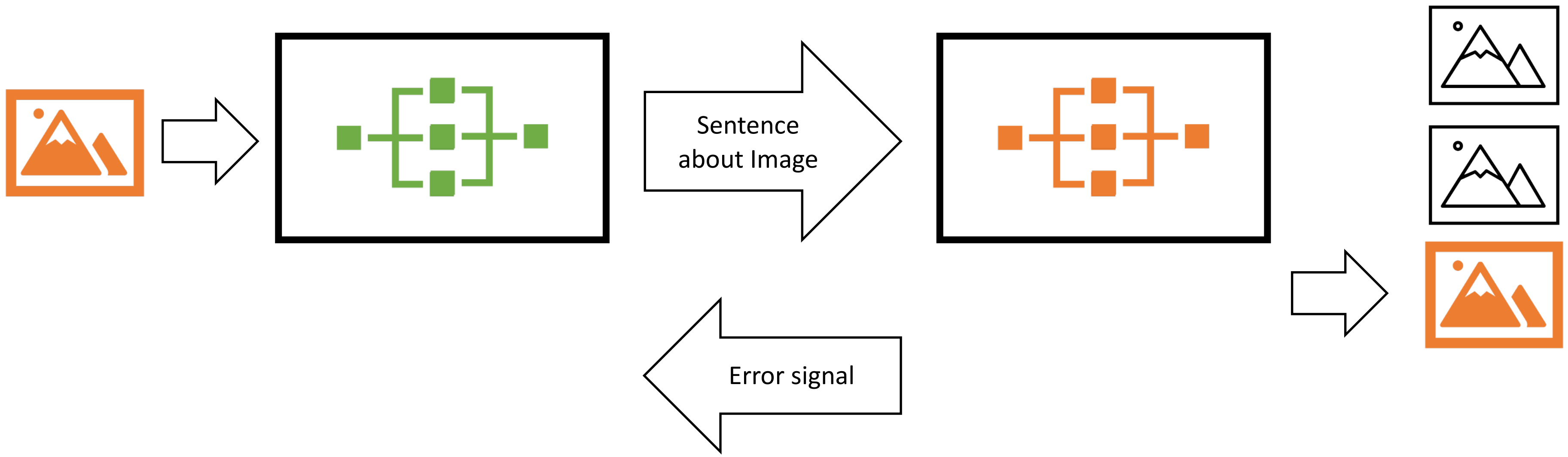
Approach

We consider two agents playing a Lewis Signaling Game:



Approach

We consider two agents playing a Lewis Signaling Game:



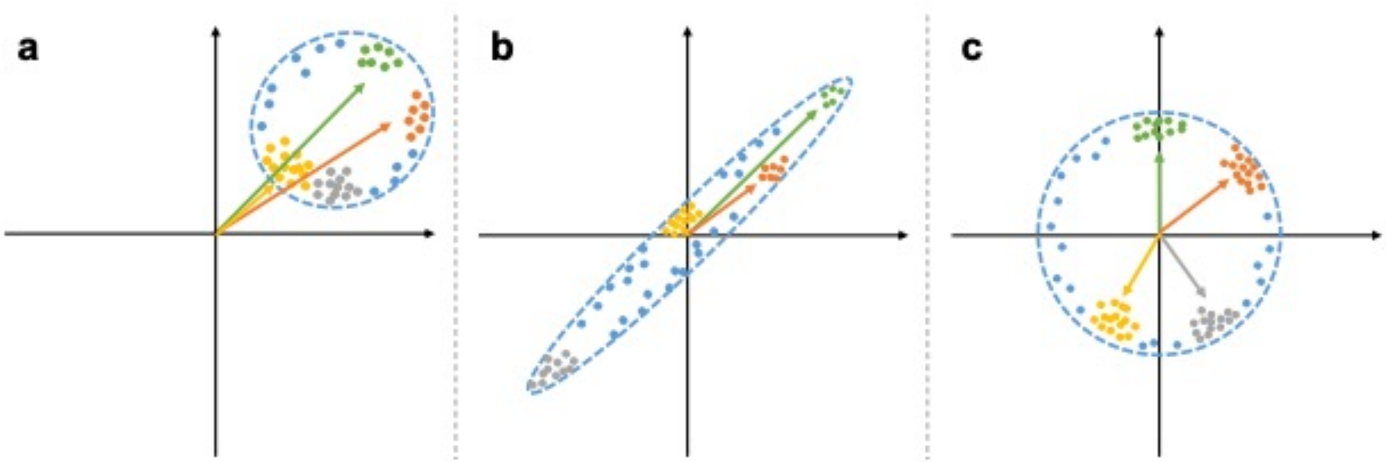
#1: Get a structure consistency from *disentangled representations*

Disentangled representations

Disentangled representations (DR) enable tuning the reasoning process.

DR generally split the learning domain into *k* concept classes (which can be different from dataset classes).

-> learn latent representation with concept separation



Concept Whitening for Interpretable Image Recognition (Chen et al. 2020)

Approach

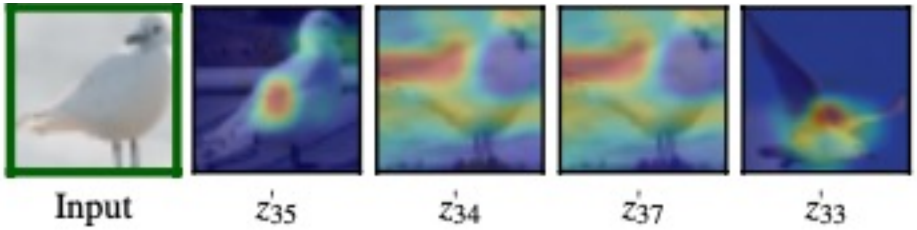
Disentangled representations (DR) enable tuning the reasoning process.

DR generally split the learning domain into *k* concept classes (which can be different from dataset classes).

-> learn latent representation with concept separation

ProtoPNet (Chen et al. 2018)

- **Unsupervised** disentanglement - using prototypical image patches from the data to represent concepts (denoted **z**).

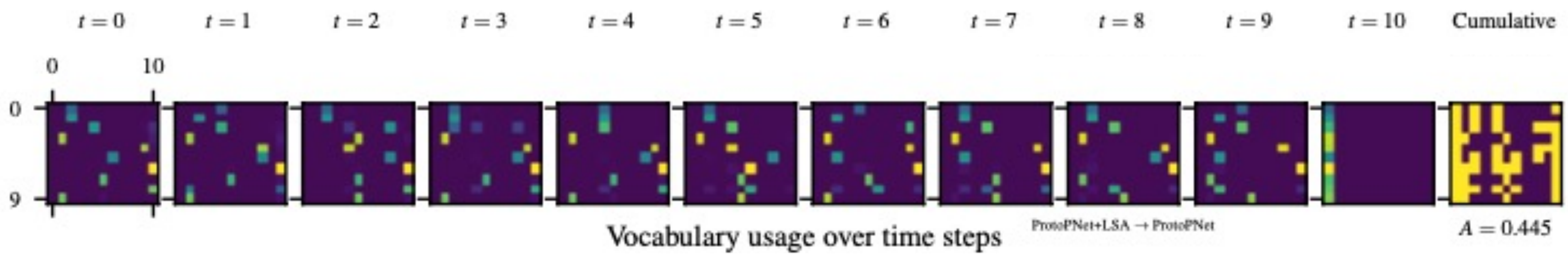


Approach

- Now for #2 (bulk of this talk): How can we compare/contrast learning structures through language (i.e., over the channel)?
- Difficult problem. Without supervision, the channel completely mixes the sender structure:

Approach

- Now for #2 (bulk of this talk): How can we compare/contrast learning structures through language (i.e., over the channel)?
- Difficult problem. Without supervision, the channel completely mixes the sender structure:



Approach

- Now for #2 (bulk of this talk): How can we compare/contrast learning structures through language (i.e., over the channel)?
- Difficult problem. Without supervision, the channel completely mixes the sender structure.
- Since we lose the mapping between sender agent's structure and the language, what if we make the agents learn it?
  - Refer to this as **reification** process
- Using the language-structure mapping, we can then ask receiver agent to perform relational inference against its own structure.
  - Multi-task learning

Multi-task learning (MTL)

- Sender solves two joint tasks:
  1. Learn to embed their top-1 activate structure ($\mathbf{z}^S$) in the message
  2. Learn to describe the target objects

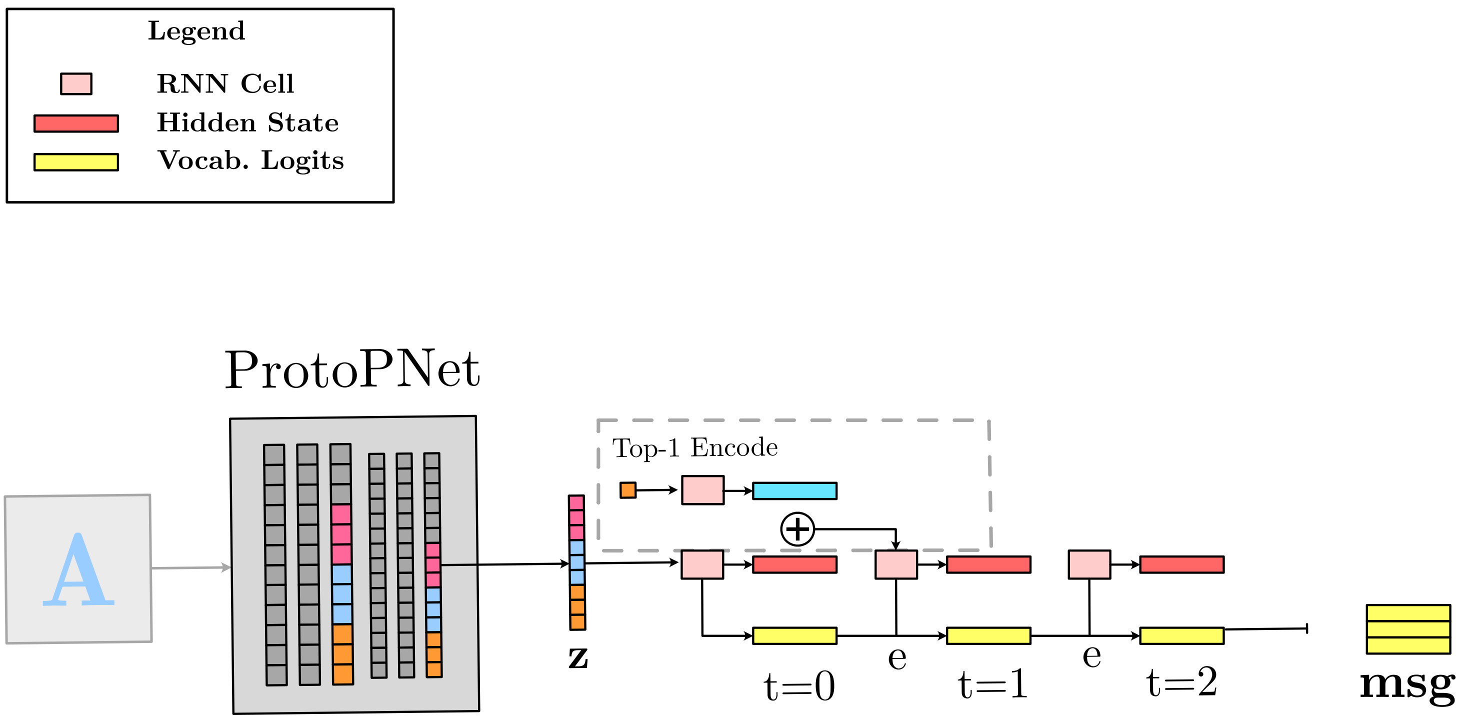
- Receiver solves two joint tasks:
  1. Learn to reconstruct the sender's top-1 structure (*rec*($\mathbf{z}^S$)) from the message (*reconstruction loss*)

$$\mathcal{L}_{rec}(\mathbf{z}^S, rec(\mathbf{z}^S)) = \frac{1}{L}\sum_{l=1}^{L} |\mathbf{z}^S_{(l)} - rec(\mathbf{z}^S_{(l)})|$$

  2. Learn to signal the correct target object (*classification loss*)

$$\mathcal{L}_{cls}(\mathbf{t}) = -\sum_{l=1}^{L} \alpha \log p(y_{(l)} = \mathbf{t} \mid msg_{(l)}),$$

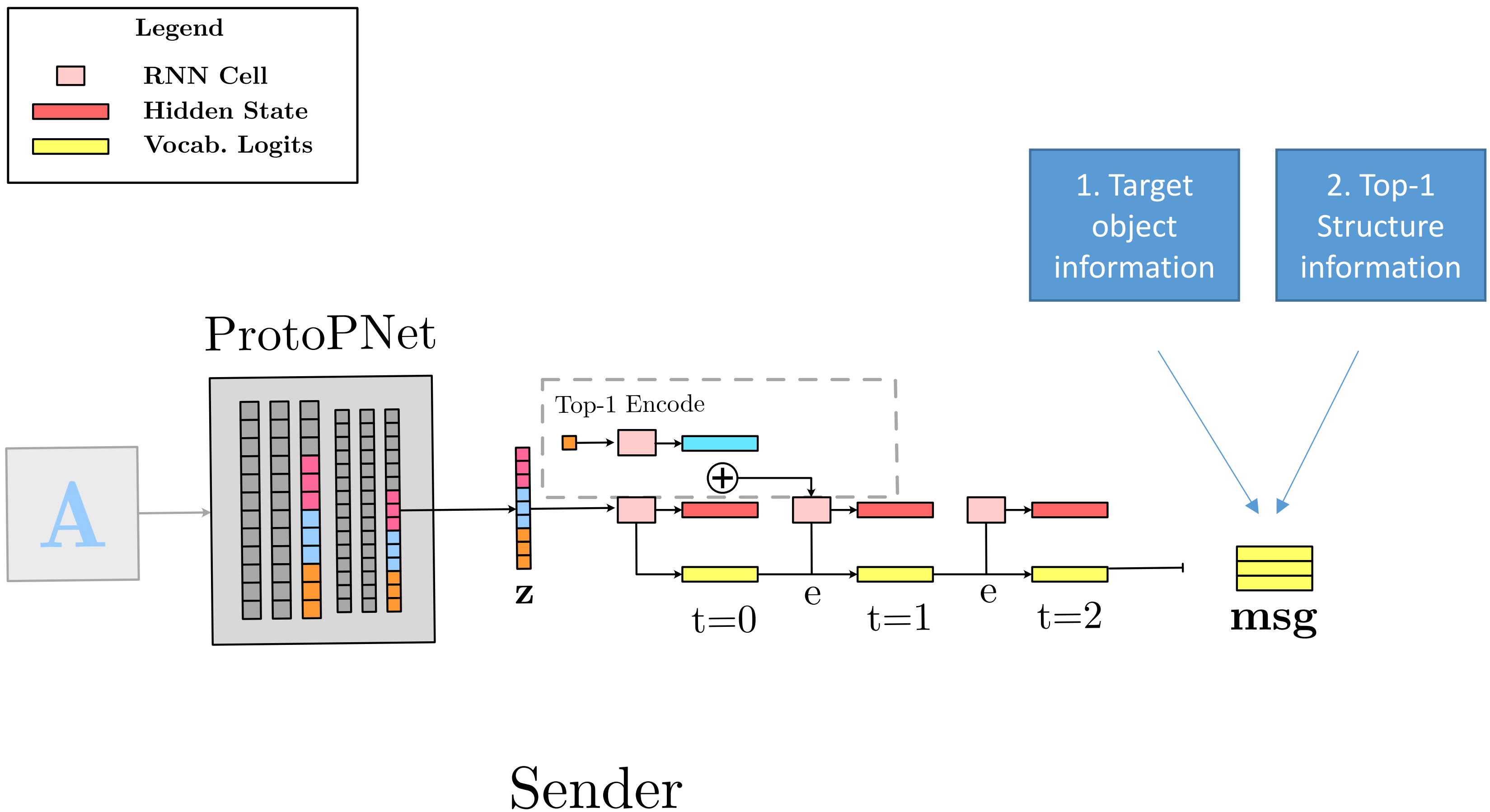
$$\boxed{\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{rec}}$$

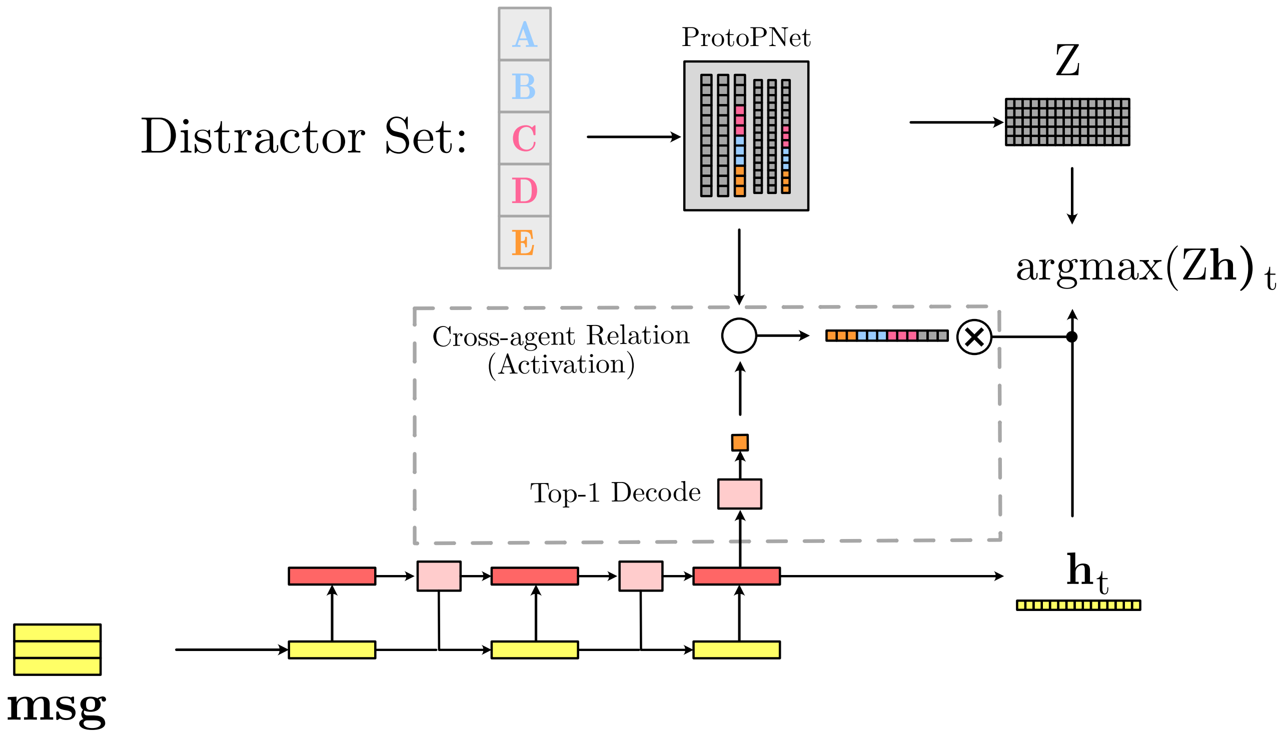


Legend
RNN Cell
Hidden State
Vocab. Logits
ProtoPNet
A
Top-1 Encode
z
e
e
t=0
t=1
t=2
msg

Sender

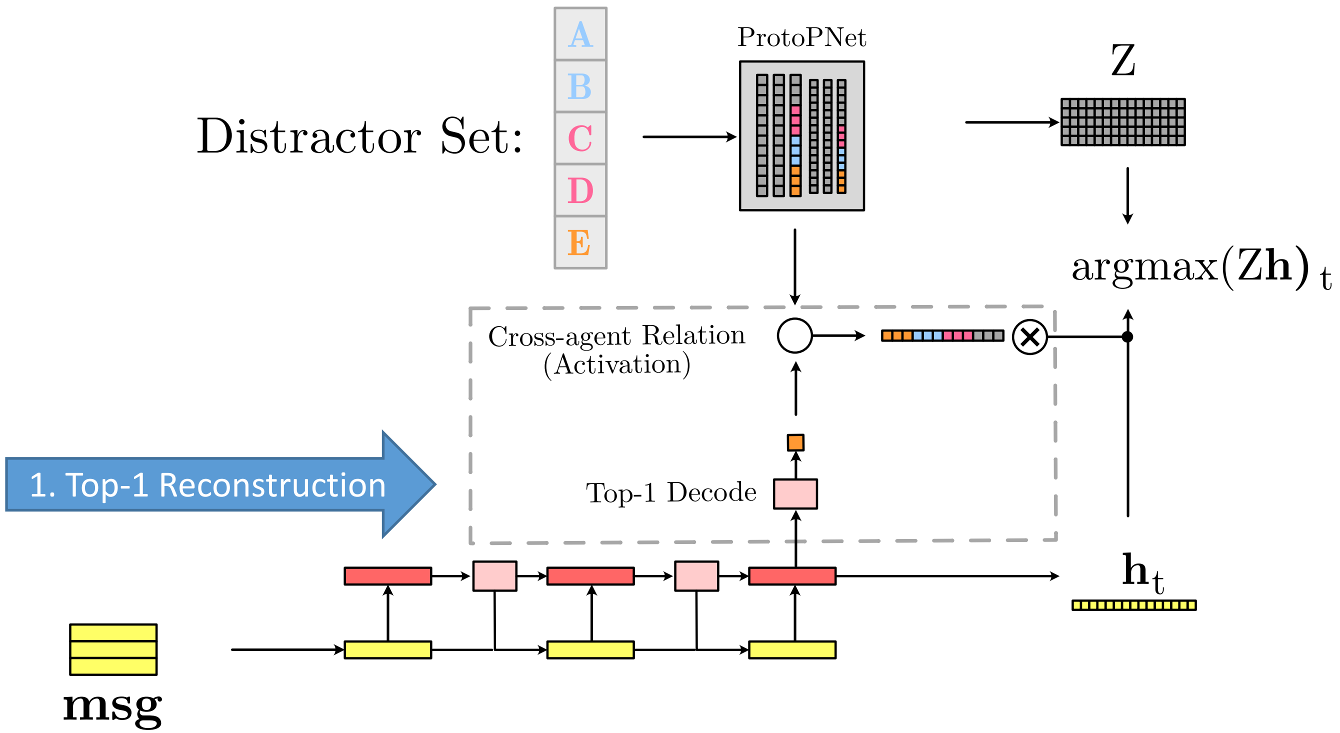
Architecture



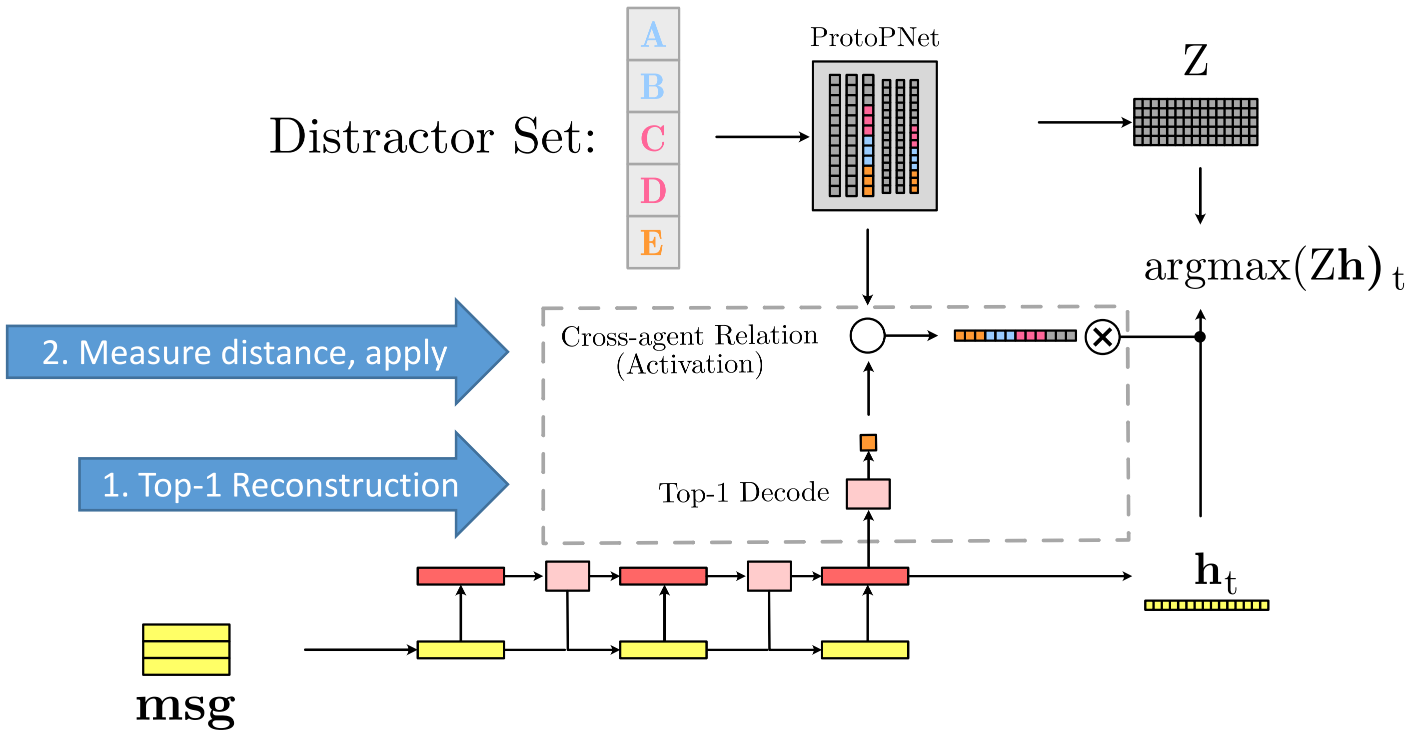

Distractor Set:
A
B
C
D
E
ProtoPNet
Z
argmax(Zh) t
Cross-agent Relation
(Activation)
Top-1 Decode
h t
msg

Receiver

Architecture



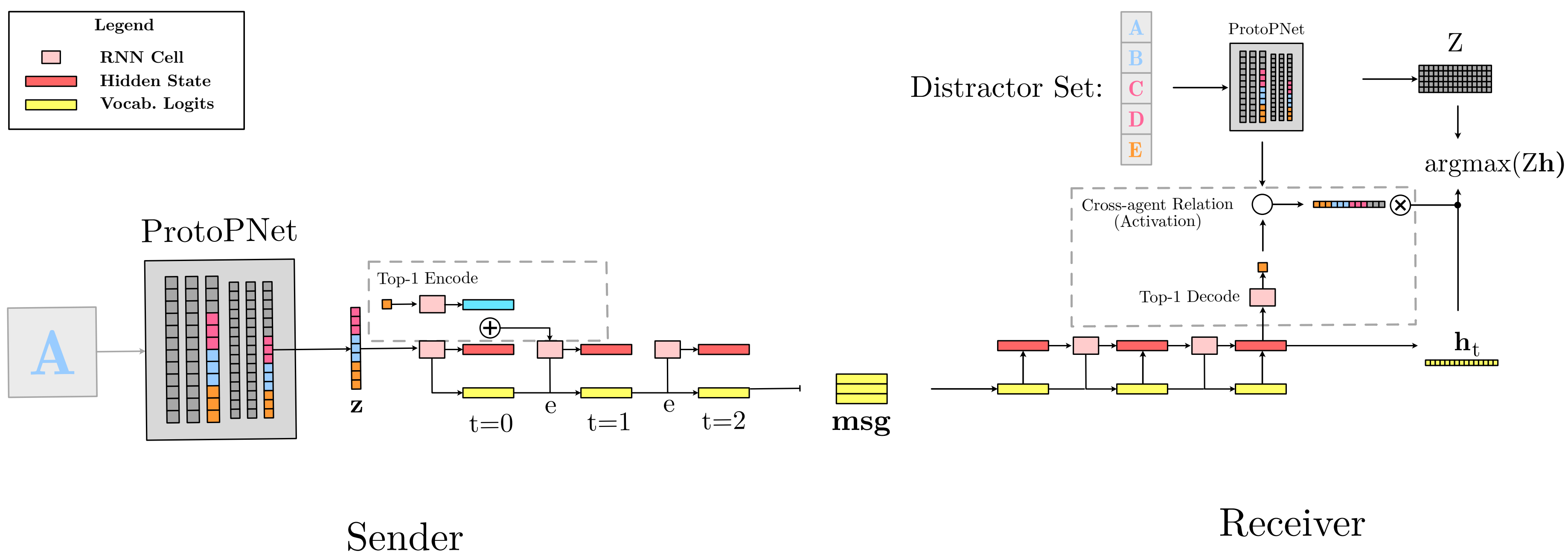
Receiver



Distractor Set:
A
B
C
D
E
ProtoPNet
Z
argmax(Zh) t
2. Measure distance, apply
1. Top-1 Reconstruction
Cross-agent Relation
(Activation)
Top-1 Decode
h t
msg

Receiver

Architecture



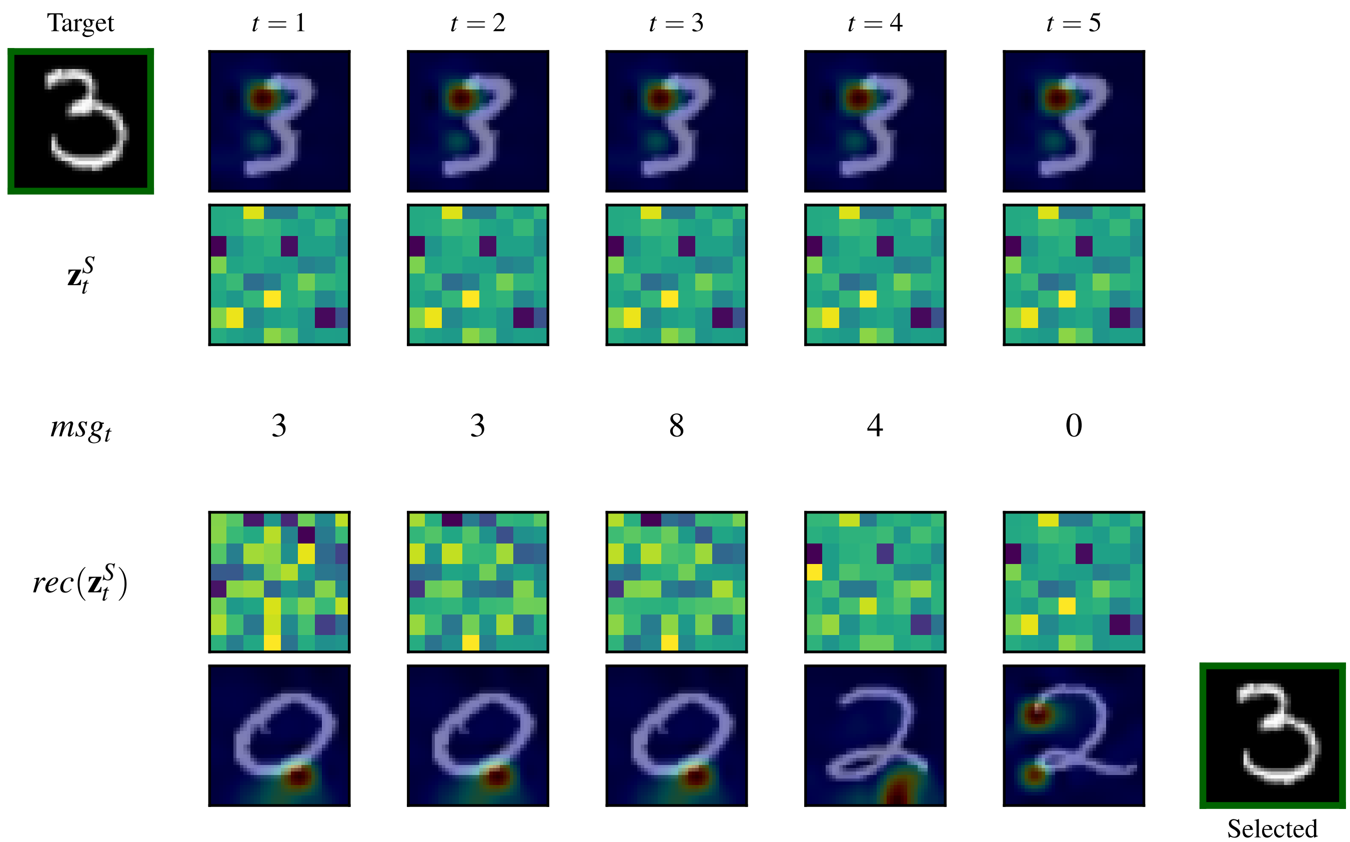
Setup

Challenges:

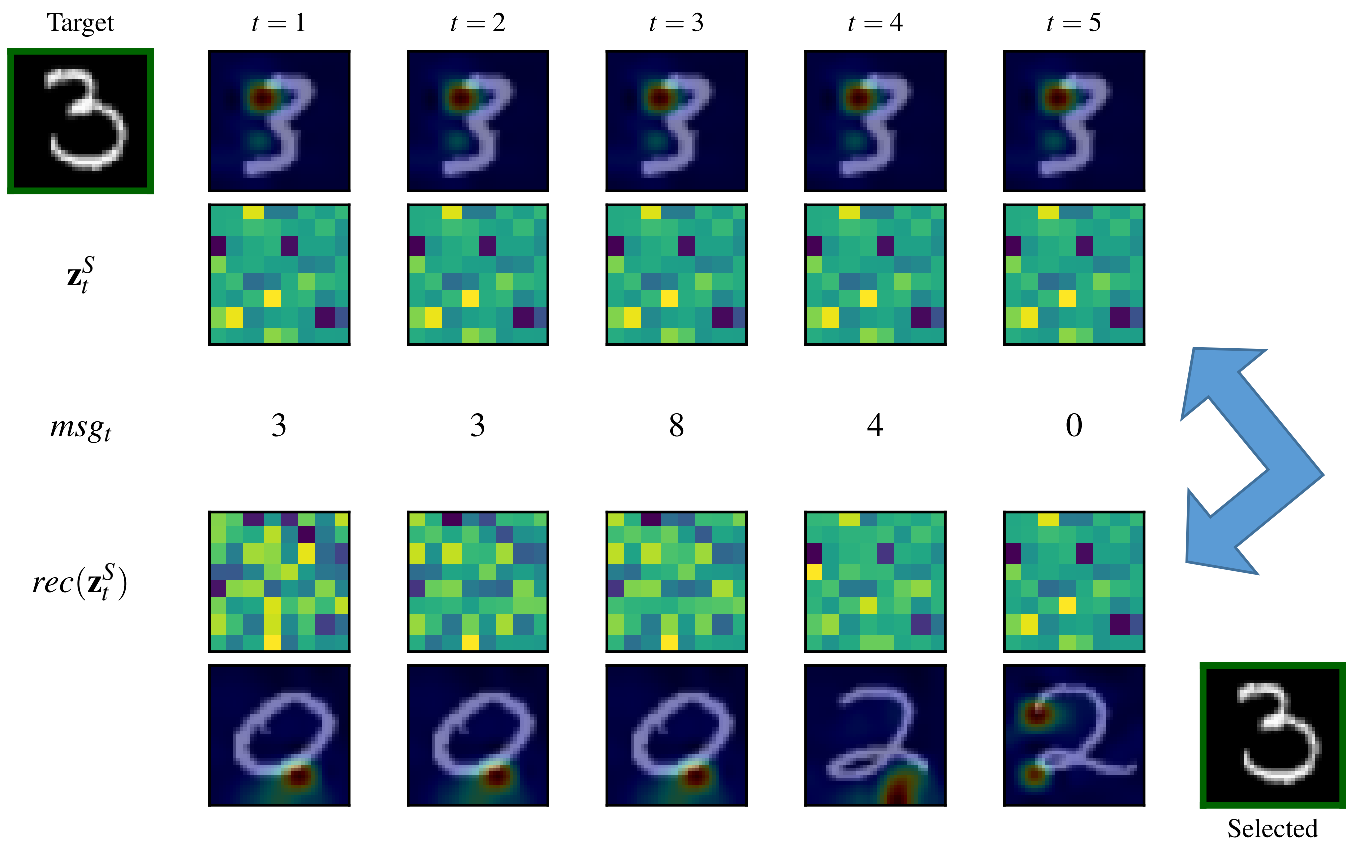
- Although DR have structural consistency, those used in ProtoPNet follow an arbitrary distribution.
- Sender and Receiver are different models, so their DR may not concentrate in the same regions of latent space, hurting comparison
- Practically speaking, architecture is very sensitive to hyper-parameters -> grid search for 200 GPU hours on ACT3 cluster

This talk: Preliminary study on MNIST

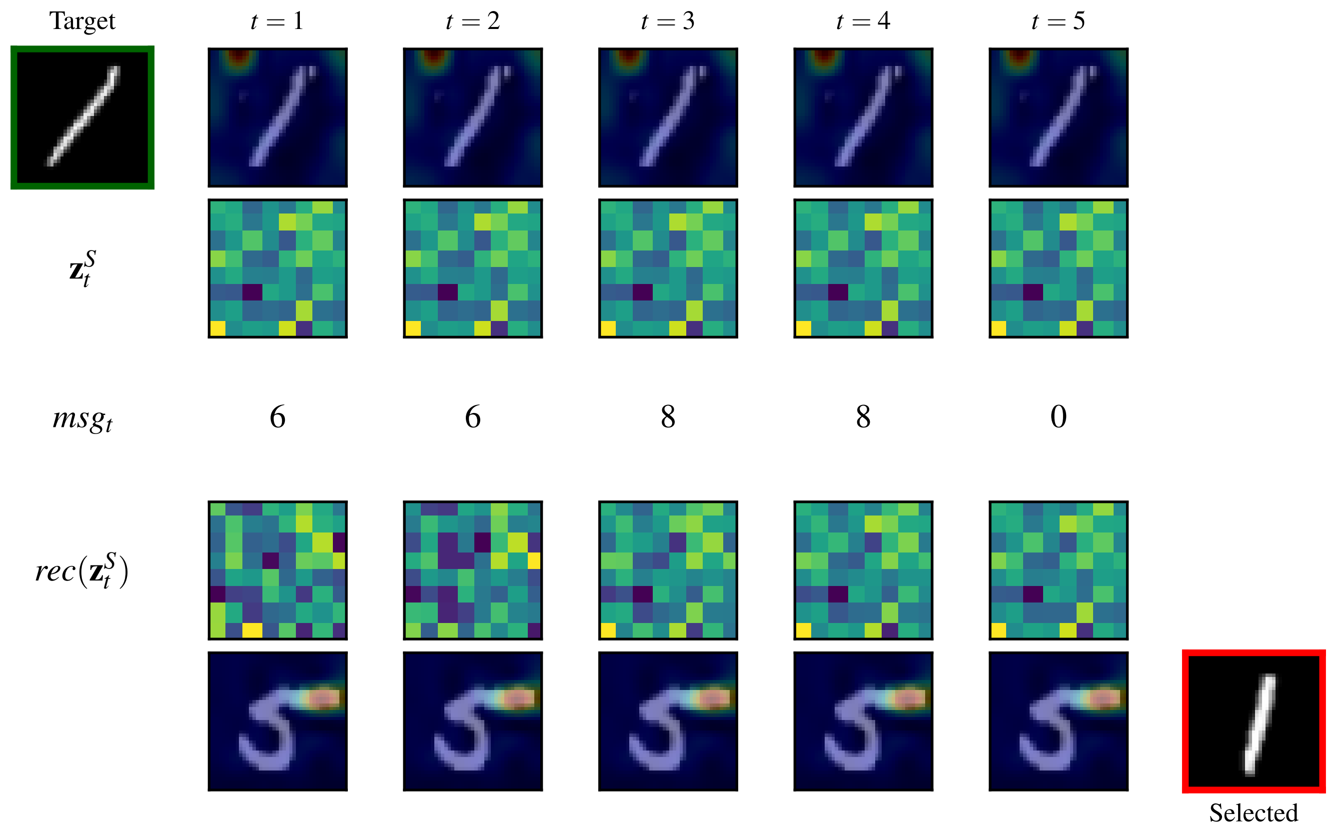
Qualitative Results



Qualitative Results

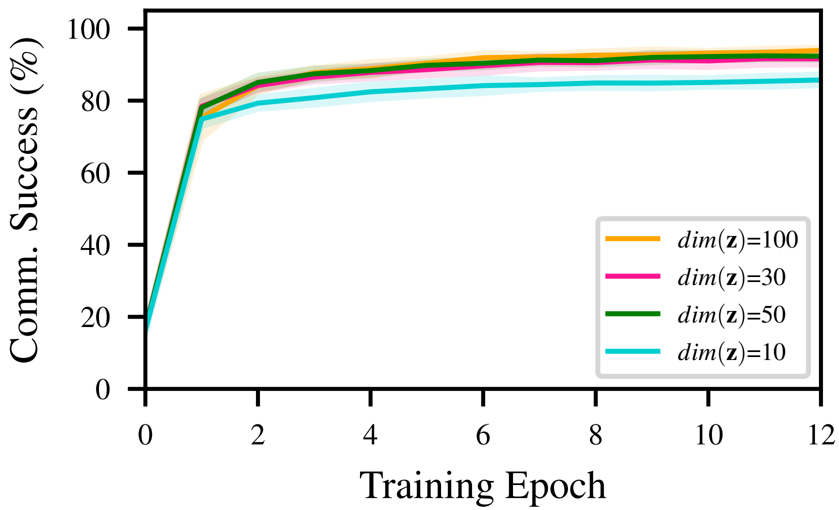


Qualitative Results

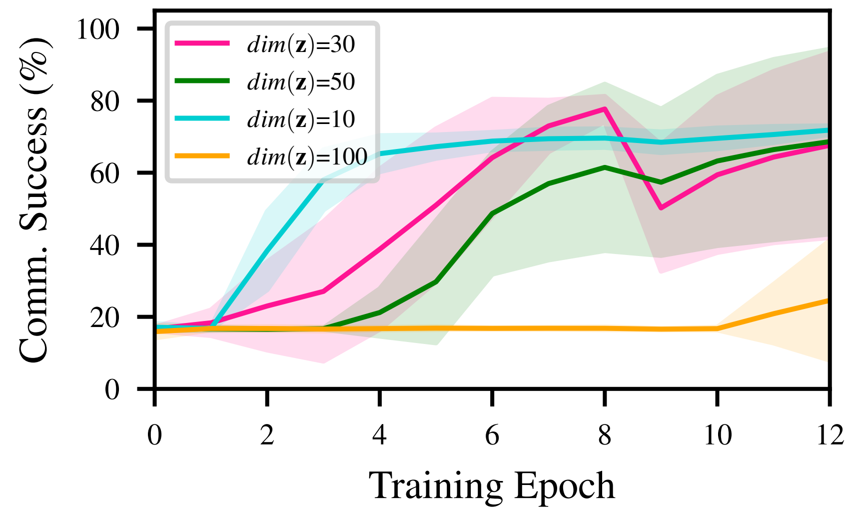


Performance

Baseline



Ours



Only train Top-1 decoder

Apply activation

Lingering Questions

How to ensure similar latent space concentration between sender and receiver?

Current scheme assumes (vector) latent space, what about other knowledge priors like graphs?

Future work:

- Embedding top-k structures as n-gram
- Submission to ACL RR
- Leveraging different agent logic (e.g., Dan Guralnik's UMA models) for structural comparison

Future Work

- Senders solve ~~two~~ three joint tasks:
    1. Learn to embed their top-1 activate structure in the message
    2. Learn to describe the target objects
    3. Update knowledge structure based on embedding difficulty
- Receivers solve ~~two~~ three joint tasks:
    1. Learn to reconstruct the sender's top-1 structure from the message
    2. Learn to signal the correct target object
    3. Update knowledge structure based on perceived utility of sender structure

Thank you

w.garcia@ufl.edu

UF UNIVERSITY of FLORIDA

Duke UNIVERSITY

TEXAS The University of Texas at Austin

UC SANTA CRUZ