Multi-Agent Image Classification via Reinforcement Learning

Hossein K. Mousavi[1], Mohammadreza Nazari[1], Martin Takáč[1], and Nader Motee[1]

[1]Lehigh University

ISE Technical Report 19T-007

Multi-Agent Image Classification via Reinforcement Learning

Hossein K. Mousavi, Mohammadreza Nazari, Martin Takáč, and Nader Motee*

***Abstract*— We investigate a classification problem using multiple mobile agents that are capable of collecting (partial) pose-dependent observations of an unknown environment. The objective is to classify an image (e.g, map of a large area) over a finite time horizon. We propose a network architecture on how agents should form a local belief, take local actions, extract relevant features and specification from their raw partial observations. Agents are allowed to exchange information with their neighboring agents and run a decentralized consensus protocol to update their own beliefs. It is shown how reinforcement learning techniques can be utilized to achieve decentralized implementation of the classification problem. Our experimental results on MNIST handwritten digit dataset demonstrates the effectiveness of our proposed framework.**

I. Introduction

With the rise of the Internet of Things (IoT), the demand for the cooperation of autonomous agents is ever increasing. The interconnected robots will be major players in future, accomplishing many duties in industrial automation [1], military support [2], and health-care [3]. In many of these applications, a major issue is that every single agent has a limited sensing capability, and therefore, may not have sufficient information for accomplishing a complex task. One way to mitigate this shortcoming is to let the considered task to be solved collectively by multiple agents. This means that the agents need not only to learn through individual interaction with their environment but also they can *learn* from each others' experiences through a communication.

In several deep learning applications, the problem suffers from high-dimensionality of feature space, which may make the learning process inefficient or even infeasible in practice. One may list facial element recognition, genome disorder identification, or fault detection as instances of problems facing these issues. In these examples, the main challenge is that a large portion of the input data might be irrelevant to the task. One possibility is to pre-process the data, which lets us filter the irrelevant pieces of information. However, this might be challenging or result in loss of data accuracy. In this paper, we propose an approach may provide an alternative mechanism for complexity reduction: we translate the initially intractable supervised learning task into a multi-agent reinforcement learning setting, where the space of observations per agent has a comparatively lower dimension.

*The first and second author have equal contribution to this work.

H.K.M. and N.M. are with the department of Mechanical Engineering and Mechanics, Lehigh University, Bethlehem, PA 18015, USA {mousavi, motee}@lehigh.edu.

M.N. and M.T. are with the department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA 18015, USA mon314@lehigh.edu, takac.mt@gmail.com,

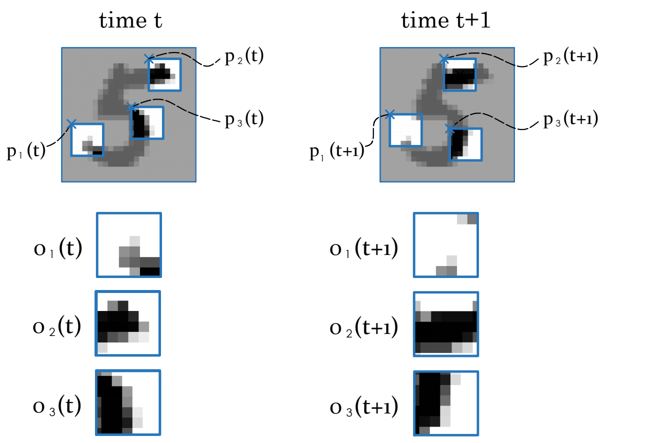


Fig. 1: An example showing how the spatial variables dictate the observations of agents in the environment. The observations by 3 *agents at two consecutive time instants have been magnified. This highlights the need for a communication mechanism and temporal memories for a reliable classification.*

At the same time, we may still provide competitive levels of performance.

In this work, we study the multi-agent image classification problem within an unknown environment. The setup consists of an environment, in which multiple homogeneous agents, each with a partial observation of the environment, are collaborating with each other to do a classification task. To explore the environment efficiently, the agents need to learn how to optimally traverse the environment. In addition to new observations that the agents can receive from the environment through re-positioning, they are also capable of communication with each other in order to update their beliefs. This is motivated by the idea that a more experienced teammate could provide or explain hints, which can be used to improve the overall performance. We are interested in maximizing a long-term collaborative objective such that the agents may effectively coordinate and cooperate to correctly classify the environment. Our goal in this work is to co-design the decentralized data-processing, communication, and decision-making policies on the agents, while that the agents are capable of establishing autonomous connections to other agents according to an information structure. This would allow implicit or explicit knowledge sharing among the agents, which will facilitate aggregating the received information for an efficient perception (see Fig. 1 for an example).

For solving this problem, we formulate it within a multi-agent reinforcement learning (MARL) framework and propose a policy gradient, which can effectively optimize the agents' behavior. In contrast to the vanilla policy gradient (PG) settings, our framework introduces a differentiable reward rather than a reward that is independent of the network parameters. The proposed mechanism enables the PG algorithm to not only maximize the probability of generating desirable outcomes but also to explicitly increase the rewards. The mathematical derivation of the latter argument is investigated in Section IV, which generalizes the policy gradient approach for the case of differentiable rewards.

We have demonstrated a competitive performance of the proposed framework in the MNIST classification task. We show that we can correctly classify 88% of the testing dataset by using two agents, each only with 2 $\times 2$ pixels observations. We observe that with larger observations or longer communications, the prediction quality can further increase up to 97.75%.

II. Problem Statement

Suppose that N identical agents move in a static unknown environment and collect partial observation about that environment. Let the agents start from a pre-determined spatial configuration. At each time step, each agent is capable of collecting a partial observation from the environment, perform some local data processing, and communicate the result with neighboring agents. The agents are allowed to communicate over a directed graph, where neighbors of an agent are those agents whose messages can be received by that agent. We assume that each agent knows its own pose with respect to the environment and can take certain actions, move, and update its pose at each time step.

The collective objective of these agents is to classify the instance of the environment from a finite number of possibilities $\{1, 2, \ldots, M\}$ over a finite time horizon.

In order to decentralize the process throughout the execution, the actions by each agent should be decided solely based on the local information available to them. As a result, agents need to learn how to communicate, extract relevant features and specifications from partial observations, navigate in the environment, and reliably solve the classification problem.

In Section IV, we propose a tractable architecture for the network of multiple agents and discuss details of different modules of an agent in order to achieve decentralized communication, data-processing, decision-making, and prediction. In Section VI, we demonstrate all required steps for learning the design parameters using reinforcement learning techniques. This allows the agents to reliably fulfill their mission through communication and cooperation.

III. Connections to the Literature

In most of the MARL literature, it is typical to assume that the agents are non-identical since their respective policies might be very distinct [4], [5]. Even though this assumption might be non-avoidable in some applications, assuming the existence of a common shared policy for all agents can be helpful, especially when their policies can be distinguished via some visible characteristics e.g. their location. Among the literature, [6] has considered a similar setup for the homogeneity of the agents. The assumption of homogeneity of the agents provides us with the advantage that the agents can learn from each others' experience. A similar idea is followed in [7], where an agent interacts with multiple instances of an environment, allowing her to learn from a concurrent stream of experiences. Even though they study a single-agent scenario, the homogeneity assumption allows approaching this multi-agent problem as a single-learner problem.

In a number of works on multi-agent reinforcement learning, the focus is on cases where the data is spread over a number of agents, and the goal is to conduct the *training* in a distributed or decentralized manner. For instance, a promising framework for this purpose is the federated learning, where the goal is to conduct the stochastic gradient descent by combining the partially computed gradients over different agents [8]–[10]. Contrary to this line of research, we consider settings in which the agents need to communicate throughout the *execution* as well.

In a more related paper, the authors of [5] consider a value function approximation approach for decentralized learning with interconnected agents, and bring operational guarantees in the case of linear value functions. A similar multi-agent learning problem is addressed in [11], where deep neural networks are used as function approximators. A generalization of this problem within an actor-critic scenario appears in [12]. Contrary to these works, we address the case where each data point is observed by a number of agents that are collaborating to fulfill a task (e.g. see Fig. 1), while the next set of observations are affected by (locally) decided actions of the agents. Moreover, in our settings, the reward for reinforcement learning becomes differentiable. This necessitates revisiting the derivation of the policy gradients.

Another related line of research concerns the end-to-end design of distributed or decentralized control architectures, where the goal is to learn optimal control laws in a setting with multiple dynamic agents [13]–[15].

A. Notations and Preliminaries

The set of real numbers, nonnegative real numbers, and nonnegative integer numbers are denoted by $\mathbb{R}$, $\mathbb{R}_+$ and $\mathbb{Z}_+$, respectively. The sets are denoted by script letters; e.g. $\mathcal{A}$. The cardinality of set $\mathcal{A}$ is denoted by $|\mathcal{A}|$. We use bold letters to denote maps; e.g. $\mathbf{g}(x)$. The unknown parameter of a map is denoted by θ_i, which appears as an index; e.g. $\mathbf{f}_{\theta_1}$. The map $\mathbf{SoftMax}(x) : \mathbb{R}^M \to \mathbb{R}_+^M$ is standard softmax map, whose element k is given by $\exp(x_k)/\sum_{i=1}^M \exp(x_i)$ A directed graph $\mathcal{G}$ is characterized by a set of nodes (or vertices) $\mathcal{V} := \{1, 2, \ldots, N\}$ and set of directed edges (or arcs) denoted as $\mathcal{E} \subset \{(i,j) : \; i,j \in \mathcal{V}, \; i \neq j\}$. We say that node $j \in \mathcal{V}$ is an in-neighbor of node $i \in \mathcal{V}$ if $(j,i) \in \mathcal{E}$, and denote the set of all of its in-neighbors by $\mathcal{N}_i := \{j \in \mathcal{V} : \; (j,i) \in \mathcal{E}\}$.

IV. Architecture of the Multi-Agent Network

In the following, we discuss details of a modular design methodology to solve the image classification problem using multiple autonomous agents.

A. Temporal Evolution of Agents' Beliefs

In order to enable learning long-term dependencies during the classification task, we equipped each agent by a dynamic module using the notion of Long Short-Term Memory (LSTM) cell [16]. The role of this module is to encapsulate the aggregate belief of an agent throughout the task. Following the widely accepted terminology [17], let us denote the hidden state and the cell state of a LSTM module on agent $i \in \mathcal{V}$ at time $t \geq 0$ by $h_i(t) \in \mathbb{R}^n$ and $c_i(t) \in \mathbb{R}^n$, respectively. Each agent updates its own belief upon communicating with its neighbors and forming an information input $u_i(t) \in \mathbb{R}^{3n}$ that contains three components: features of local observations, the average of the decoded messages received from its neighbors and information about its location. The time evolution of the LSTM module is governed by

$$\begin{bmatrix} h_i(t+1) \\ c_i(t+1) \end{bmatrix} = \mathbf{f}_{\theta_1}\left(\begin{bmatrix} h_i(t) \\ c_i(t) \end{bmatrix}, u_i(t)\right), \tag{1}$$

where nonlinear map $\mathbf{f}_{\theta_1} : \mathbb{R}^{2n} \times \mathbb{R}^{3n} \to \mathbb{R}^{2n}$ is parametrized by a trainable vector $\theta_1 \in \mathbb{R}^{n_f}$. In the following subsections, we discuss each component of the information input to the LSTM module.

B. Agent Motion and Stochastic Action Policy

Let us represent the spatial state (or pose) of agent $i \in \mathcal{V}$ by $p_i(t) \in \mathbb{R}^d$ and the finite set of all possible actions, which agents can take, by $\mathcal{A}$. Each agent moves in the spatial domain according to dynamics

$$p_i(t+1) = \mathbf{g}\big(p_i(t), a_i(t+1)\big), \tag{2}$$

where $\mathbf{g} : \mathbb{R}^d \times \mathcal{A} \to \mathbb{R}^d$ is a known transition map and action $a_i(t+1)$ is sampled from set $\mathcal{A}$ according to a probability mass function $\pi : \mathcal{A} \to \mathbb{R}$ that is computed as follows. We use a state-dependent stochastic action policy by updating the action probabilities according to

$$\pi(a \,|\, O(t)) = \boldsymbol{\pi}_{\theta_3}(a, \hat{h}_i(t+1)), \tag{3}$$

where a is an action in $\mathcal{A}$, $O(t)$ is the history of all observations by all agents at time, $\hat{h}_i(t+1)$ is the hidden state of an auxiliary LSTM unit whose dynamics is governed by

$$\begin{bmatrix} \hat{h}_i(t+1) \\ \hat{c}_i(t+1) \end{bmatrix} = \mathbf{f}_{\theta_2}\left(\begin{bmatrix} \hat{h}_i(t) \\ \hat{c}_i(t) \end{bmatrix}, u_i(t)\right). \tag{4}$$

This LSTM unit is fed with exactly the same information input $u_i(t)$ as the belief LSTM module (1).

We should emphasize that the action policy (3) is non-Markovian because it takes into account the history rather than immediate observations. The corresponding trainable parameter is $\theta_3 \in \mathbb{R}^{n_\pi}$. In this paper, we consider one fully connected layer with a ReLU activation and one fully connected linear layer for the output.

Example 1: For a flying robot that can translate and rotate in a 3D space, a natural choice for the spatial state $p_i(t)$ is a vector in $\mathbb{R}^6$ that is created by stacking three position components of the robot and three Euler angles describing its orientation (relative to the environment).

C. Inter-Agent Communication Architecture

The agents are allowed to communicate over a directed graph $\mathcal{G}$ with the node set $\mathcal{V}$ and link set $\mathcal{E}$. For distinct agents $i, j \in \mathcal{V}$, $(i, j) \in \mathcal{E}$ implies that agent j receives a message from agent i. Each agent generates a message[1] using its belief hidden state according to

$$m_i(t) = \mathbf{m}_{\theta_4}(h_i(t)), \tag{5}$$

where map $\mathbf{m}_{\theta_4} : \mathbb{R}^n \to \mathbb{R}^{n_m}$ is parameterized by a trainable vector $\theta_4 \in \mathbb{R}^{n_e}$. A sequence of two layers for is considered for this map: a fully-connected layer with ReLU activation followed by a fully connected linear layer as output.

D. Observation Model and Feature Extraction

Suppose that agent i at time t collects (partial) observation $o_i(t) \in \mathbb{R}^{f \times f}$. It is assumed that agents' observations can be completely characterized by its pose $p_i(t)$. Thus,

$$o_i(t) = \mathbf{o}(I, p_i(t)), \tag{6}$$

where $I \in \mathbb{R}^{n_I \times n_I}$ is the entire image. This identity can be interpreted as the *repeatability* property of the observations: two agents with different past history will observe the same image provided that they both have identical poses at the observation time. The relevant features (or specifications) of an observation can be extracted by a parametrized map

$$b_i(t) = \mathbf{b}_{\theta_5}(o_i(t)), \tag{7}$$

where $\theta_5 \in \mathbb{R}^{n_c}$ is a trainable vector. The nonlinear map $\mathbf{b}_{\theta_5} : \mathbb{R}^{f \times f} \to \mathbb{R}^n$ results from the following three layers: two convolutional neural network (CNN) layers followed by a fully connected layer.

Example 2: In Fig. 1, we illustrate an example in which the spatial state is simply the location of the agent (relative to the image frame) and observation map (6) crops a subset of the image based on its position. Moreover, the map describing the motion of the robots, according to (2), has resulted in the horizontal and vertical translation of the agents across the image.

Example 3: Let us consider the settings of Example 1, where a camera is mounted on the robot. Then, map $\mathbf{o}(.)$ in (6) for this case is the map based on the field of the view of the robot, which in turn, is completely characterized by the position and orientation of the robot with respect to the environment.

E. Structure of Information Inputs

In the previous subsections, we explained the details of belief dynamics, agent motion and actuation, observation processing, communication, and decision-making modules on each agent. The same information input is fed to both

[1]It is assumed that when agent i broadcasts its message $m_i(t) \in \mathbb{R}^{n_m}$ at time t, all neighboring agents receive identical copies of that message.

LSTM modules in (1) and (4). We design the information input as a vector in $\mathbb{R}^{3n}$ with components

$$u_i(t) = \left[b_i(t)^T \quad \bar{d}_i(t)^T \quad \lambda_i(t)^T\right]^T. \tag{8}$$

All these three components can be calculated using locally accessible data. In Subsection IV-D, it was shown that $b_i(t)$ contains important information about features of the (partial) observation.

After communicating with neighbors, each agent decodes the received messages using a parameterized map $\mathbf{d}_{\theta_6} : \mathbb{R}^{n_m} \to \mathbb{R}^n$ to get

$$d_i(t) = \mathbf{d}_{\theta_6}(m_i(t)), \tag{9}$$

where θ_6 is trainable vector. We consider a fully connected layer with a ReLU activation for this map. Then, each agent takes the average of the received messages from its neighbors to find

$$\bar{d}_i(t) = \frac{1}{\Delta_i} \sum_{(j,i)\in\mathcal{E}} d_j(t), \tag{10}$$

in which Δ_i is the in-degree of node i in graph $\mathcal{G}$. This, $\bar{d}_i(t)$ is the aggregate message received by agent i at time t.

It is useful for agents to tag their beliefs and information by their spatial state. This can be done by the following map

$$\lambda_i(t) = \boldsymbol{\lambda}_{\theta_7}(p_i(t)) \tag{11}$$

where $\boldsymbol{\lambda}_{\theta_7} : \mathbb{R}^d \to \mathbb{R}^n$ is a parametrized map with a trainable vector θ_7.

In the final step, we close the loop by applying information input (8) to (1) and (4).

V. Decentralized Prediction and Classification

Recall that the image should be classified from M categories, while we have T rounds of observation and communication. To do this, first the raw prediction vector by agent i is evaluated using the final cell state and a map $\mathbf{q}_{\theta_8} : \mathbb{R}^n \to \mathbb{R}^M$ as

$$q_i = \mathbf{q}_{\theta_8}(c_i(T)). \tag{12}$$

We use a fully-connected linear layer with ReLU activation followed by a fully connected linear layer in place of this map. Then, we run a distributed average consensus algorithm over a strongly connected directed graph. Upon this averaging, on each agent, we will have a shared prediction vector $\bar{q} \in \mathbb{R}^M$, which is given by

$$\bar{q} = \frac{1}{N} \sum_{i=1}^{N} q_i. \tag{13}$$

It has been shown that if the communication graph is strongly connected, this task can be conducted in a completely decentralized manner [18]. Finally, each agent evaluates the system-wide prediction category using

$$q_c = \operatorname*{argmax}_{j\in\{1,\dots,M\}} \ \textbf{SoftMax}(\bar{q}). \tag{14}$$

One should note that in the current approach, we do not force the agents to reach a consensus on the predicted category, but rather we aim at combining the beliefs due to sequences of partial observations to produce a single prediction.

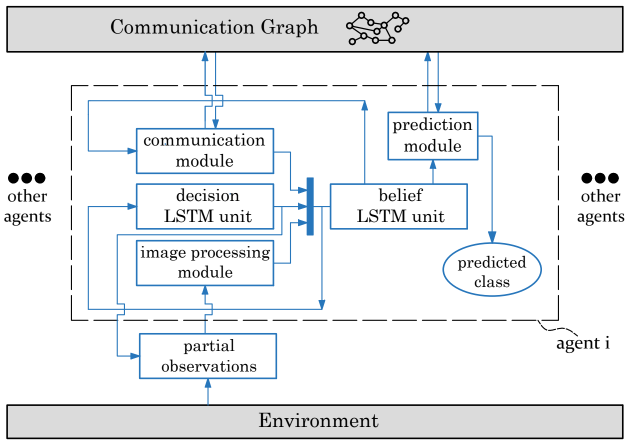


Fig. 2: The diagram illustrating the essence of this framework. The partial observations are based on the position of the agent, which in turn depend on the decision taken by the agent. The observation is fed into the image processing module. The input to the LSTM unit depends on this processed image in addition to the position of the agent and the messages from the neighbors. The prediction module finds the average of the embedded output signals from LSTM and then uses a map to come up with a prediction.

In Fig. 2 we have illustrated the connectivity architecture of the framework that has been described throughout the section. These settings and steps can be summarized to build Algorithm 1, whose output for an image is the prediction category $q_c \in \{1, \dots, M\}$ (shared by all agents).

VI. Reinforcement Learning

We derive a generalization of the vanilla policy gradient algorithm, which utilizes the intrinsic differentibilty of the rewards for simultaneous training of both prediction and motion planning parameters. First, let us denote all trajectories with positive probability of occurrence by $\mathcal{T}$. Suppose that in a sample execution $\tau \in \mathcal{T}$, image I corresponds to category $j \in \{1, \dots, M\}$ (i.e., its actual category is j). Then, we define the reward corresponding to the outcome of this sample trajectory

$$r_\tau := -\mathbf{f}_l(\bar{q}_\tau - e_j), \tag{15}$$

where $\mathbf{f}_l$ is a differentiable nonnegative loss function (e.g. mean squared error (MSE)), $\bar{q}_\tau$ is the prediction at the end of this sample trajectory, and $e_j \in \mathbb{R}^M$ is the unit coordinate vector in direction j. Let us stack our parameters as as single design parameter according to

$$\Theta := \left[\theta_1^T, \theta_2^T, \dots, \theta_8^T\right]^T. \tag{16}$$

Based on the goal of this problem, we define our objective function as

$$J(\Theta) = \mathbb{E}\{r_\tau\} = \sum_{\tau\in\mathcal{T}} p_\tau r_\tau, \tag{17}$$

Algorithm 1 Multi-Agent Classification (Execution)

```
input: Input image $I \in \mathbb{R}^{n_I \times n_I}$
       initial spatial states $p_1(0), \ldots, p_N(0)$
output: prediction category $q_c$

initialize:
  for $i \in \mathcal{V}$ do
      initialize the states $h_i(t) \leftarrow 0$, $c_i(t) \leftarrow 0$
      for $j \in \mathcal{N}_i$ do
          initialize the messages $m_j(0) \leftarrow 0$
      end for
  end for
  for $t = 0$ to $T - 1$ do        ▷ communication & observation
      for $i \in \mathcal{V}$ do
          conduct the observation $o_i(t) \leftarrow \mathbf{o}(I, p_i(t))$
          map the observation $b_i(t) \leftarrow \mathbf{b}_{\theta_5}(o_i(t))$
          for $j \in \mathcal{N}_i$ do
              decode message $d_j(t) \leftarrow \mathbf{d}_{\theta_6}(m_j(t))$.
          end for
          find average message $\bar{d}_i(t) \leftarrow \frac{1}{\Delta_i} \sum_{(j,i) \in \mathcal{E}} d_j(t)$
          map the spatial state $\lambda_i(t) \leftarrow \boldsymbol{\lambda}_{\theta_7}(p_i(t))$
          form input $u_i(t) \leftarrow \left[b_i(t)^T \quad \bar{d}_i(t)^T \quad \lambda_i(t)^T\right]^T$.
          run the belief LSTM unit (1)
          evaluate message $m_i(t+1) \leftarrow \mathbf{m}_{\theta_4}(h_i(t+1))$
          run the action LSTM unit (4)
          update policy distribution $\boldsymbol{\pi}_{\theta_3}(.\,|\hat{h}_i(t+1))$
          samples action $a_i(t+1)$ based on $\pi$
          update spatial state $p_i(t+1) \leftarrow \mathbf{g}(p_i(t), a_i(t+1))$
      end for
  end for
  for $i \in \mathcal{V}$ do                ▷ local raw predictions
      find raw prediction vector $q_i \leftarrow \mathbf{q}_{\theta_8}(c_i(T))$
  end for
  conduct the distributed average consensus $\bar{q} \leftarrow \frac{1}{N} \sum_{i=1}^{N} q_i$
  find the prediction category
      $q_c \leftarrow \mathbf{SoftMax}\big(\operatorname{argmax}_{i \in \{1,\ldots,M\}} \ \bar{q}\big)$
```

and we need to solve the optimization problem

$$\underset{\Theta}{\text{maximize}} \quad J(\Theta). \tag{18}$$

The gradient of J with respect to Θ can be written as

$$\nabla J = \sum_{\tau \in \mathcal{T}} r_\tau \nabla p_\tau + p_\tau \nabla r_\tau. \tag{19}$$

Using the well-known gradient derivation technique similar to that of the REINFORCE algorithm [19], we can write

$$\begin{aligned} \nabla J &= \sum_{\tau \in \mathcal{T}} p_\tau \nabla (\log p_\tau) r_\tau + p_\tau \nabla r_\tau \\ &= \mathbb{E}\{\nabla(\log p_\tau) r_\tau + \nabla r_\tau\}. \end{aligned} \tag{20}$$

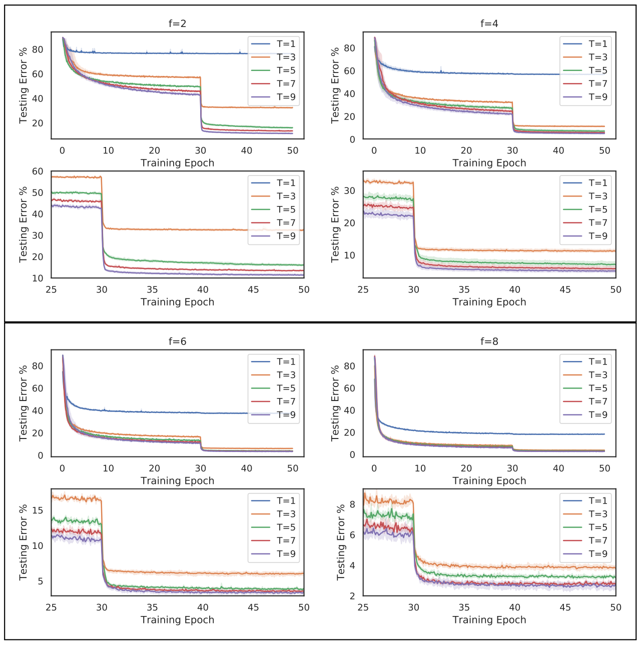


Fig. 3: The testing accuracy for different frame sizes f and time horizons T versus the number of training epochs.

Let us execute Algorithm 1 for N_r independent experiments. Then, for each sample $k = 1, \ldots, N_r$, we use $p^{(k)}$ to denote the probability that this particular trajectory is selected. Now, inspired by (20), we define the proxy sampler for J to be

$$\hat{J} := \frac{1}{N_r} \Big(\sum_{k=1}^{N_r} \log p^{(k)} r_d^{(k)} + r^{(k)} \Big), \tag{21}$$

where quantity $r_d^{(k)}$ has a value equal to $r^{(k)}$, but has been detached from the gradients. Then, we inspect that

$$\mathbb{E}\left\{\nabla \hat{J}\right\} = \nabla J, \tag{22}$$

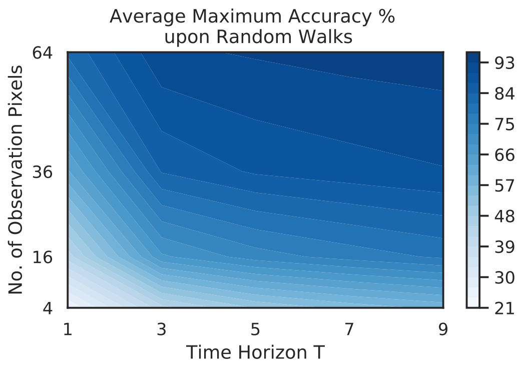
i.e., $\nabla \hat{J}$ is an unbiased estimator of $\nabla(\log p_\tau) r_\tau + \nabla r_\tau$ that appears in (20). Therefore, it is justified to follow the approximation for the gradient given by

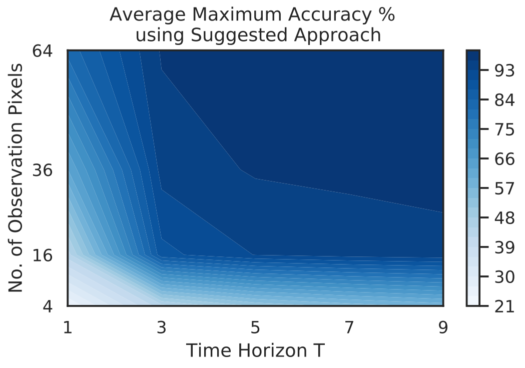
$$\nabla J \approx \nabla \hat{J}. \tag{23}$$

Note that the first term in summation (21) is identical to the quantity that is derived in the policy gradient method with a reward that is independent of the parameters (i.e., identical to REINFORCE algorithm). The second term accounts for the fact that the reward in our settings directly depends on parameter Θ: for two networks, if the agents receive exactly the same sequences of observations and take exactly the same actions, still the reward explicitly depends on the parameters of the network.

VII. Numerical Experiments

We use the MNIST database of handwritten digits [20] to study the proposed multi-agent learning algorithm. The data set consists of 60,000 training images and 10,000 testing images, where each image has 28×28 pixels. We suppose that each agent is able to observe a portion of the image (i.e.,





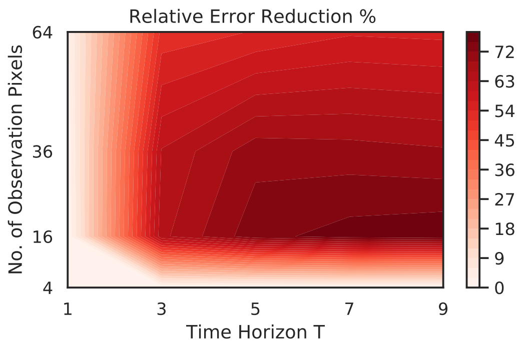


Fig. 4: Average maximum testing accuracy versus frame size f and time horizon T after 30 epochs with random walks (top figure). In the middle one, we trained extra 20 training epochs for the motion planning policy. The last figure illustrates the error reduction as a result of the design of coordination policy instead of random walks.

partial observation of the environment) that has $f \times f$ pixels. The spatial variable p_i is the pixel coordinate of the top left corner of the observation window. The possible movements by each agent can be characterized by

$$\mathcal{A} = \{\text{up}, \text{down}, \text{left}, \text{right}\}. \tag{24}$$

By each movement, the agent is translated across the desired direction by f_m pixels. If the sampled action is infeasible, then the agent will remain at its location at that time instant. In Fig. 1, as an example, we have illustrated an image from these data and the observations that three agents observe during the horizon. In all experiments of this section, we choose a batch-size of 64 images for the purpose of training. We also choose the size of the states of LSTM unit to be $n = 64$, the number of neurons of all fully connected layers

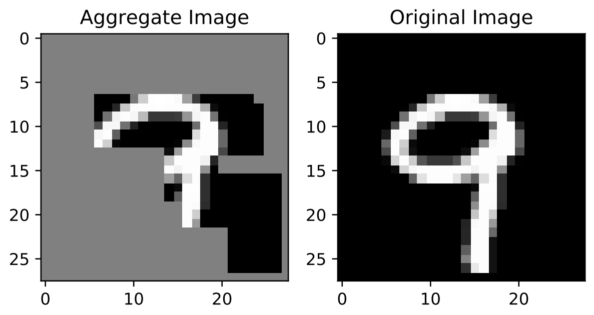


Fig. 5: A sample of masked images created by putting together the observations by 3 different agents for a time horizon of $T = 3$ with $f = 8$. This image is the input to the centralized image classifier as an alternative classification approach (method (i)). The (random) uncovered parts have been reached due to random walks.

to be 64, and the dimension of the broadcasted message to be $n_m = 12$. We have implemented this approach using the machine-learning toolbox PyTorch [21], and will release our codes upon publication.

Testing Accuracy Results: We consider $N = 2$ agents that are communicating over the only option for a strongly connected graph; i.e., graph with arc set $\mathcal{E} = \{(1, 2), (2, 1)\}$. We choose $N_r = 30$ and conduct a parametric study by varying the observation frame size f and time horizon T according to $f \in \{2, 4, \ldots, 8\}$ and $T \in \{1, 3, \ldots, 9\}$, respectively. For each pair of f and T, we conduct the training for 50 epochs. However, we break-down the training procedure into two parts: first, we consider random walks for 30 epochs. Then, we fix all of the parameters except for the decision-making module and train the model for another 20 epochs. The intuition behind this method of training is that we initially train the perception and communication scheme while agents are exploring the environment. Once these modules are sufficiently trained, we let the agents learn how to traverse the image. The numerical experiments suggest that this training method generally results in smaller testing errors, compared to the case in which we simultaneously optimize the parameters of all modules. One possible justification of this observation is that the two-stage training method is less prone to getting stuck in local non-stationary solutions due to higher exploration in the first phase. In Fig. 3, we demonstrate the progress of the testing error versus the number of training epochs. Also, in Fig. 4, we show the average maximum testing accuracy for each pair of frame size f and time horizon T in the case of random walks (i.e, at the end of 30 epochs) and also with the designed law for the movements of the agents (i.e., after additional 20 epochs of training for motion planning). The results suggest that that having a policy that governs the movements of the agents may significantly decrease the testing error; e.g. Fig. 5 implies that for $f = 4$ (i.e., 16 observation pixels) and $T = 7$ communication and observation steps, the testing error has decreased by more than 70%.

Alternative Classification Schemes: We consider two alternative methods to classify the images based on similar

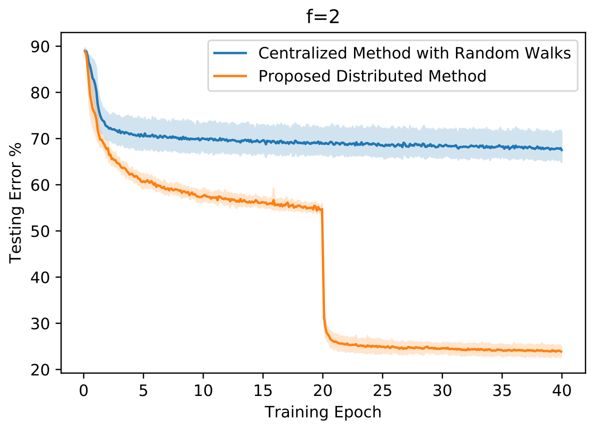


Fig. 6: Comparison of the testing error when using a centralized method with the suggested approach. The first 20 *epochs of training corresponds to the case for random walks, while in the next* 20 *epochs we optimize the movements of the agents.*

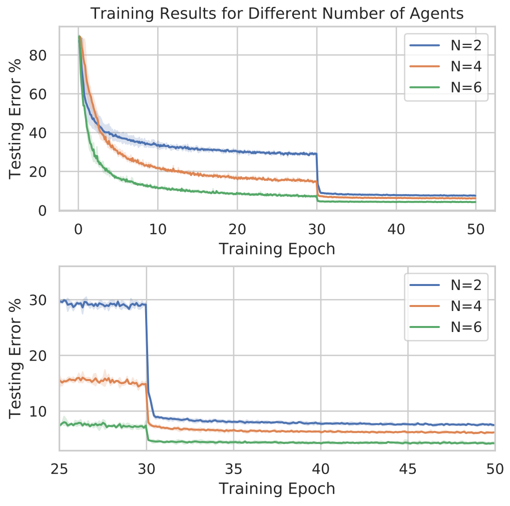


Fig. 8: The training results for different number of agents.

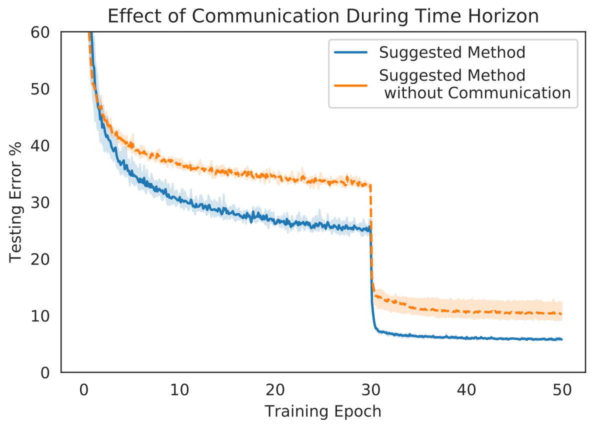


Fig. 7: The result of training when with and without communications.

observations, which will be compared against our approach. In both cases, each agent independently conducts a *random walk*. *(i) Centralized Classification with Random Walks:* An alternative is to collect all the images from all agents based on random movements that have equal probability in each of four directions (i.e. $1/4$). Then, we feed the resulting unmasked image to a single CNN which is embedded into a prediction vector $q \in \mathbb{R}^M$ (similar to $\bar{q}$ in (13)). In Fig. 5, we demonstrate a typical input of this simple centralized classification, which has the same dimensions as the images in MNIST (i.e., $f = 28$). *(ii) Distributed Classification with Random Walks:* We also consider a variant of Algorithm 1 in which instead of learning the optimal distributions for the policy, we suppose that all possible movements in (24) have an equal probability of $1/4$. We set the parameters to $f = 2$, $N = 2$ agents, $N_r = 10$ samples, time horizon $T = 4$, and a complete communication graph. We conduct the training with random walks for 20 epochs, which is followed by training of the decision-making motion planning module for another 20 epochs. We train the centralized classification for 40 epochs as well. In Fig. 6, we illustrate the results, which show that the quality of prediction using the proposed method is superior relative to the method with random walks as well as the centralized method.

Effect of Communication: We compare the result of training for an alternative structure in which the agents do not communicate during the horizon (although, they finally do so conduct the prediction). We set the parameters to be $N = 2$, $f = 4$, $N_r = 40$, and $T = 6$. As shown in Fig. 7, it turns out that smaller testing errors compared to the case that the agents do not communicate.

Effect of Number of Agents: We consider the set of parameters $f = 4$, $N_r = 25$, and $T = 4$ and conduct the training for a different number of agents communicating over a complete directed graph. In Fig. 8, we illustrate the result of training for 30 epochs with random walks followed by learning the moving policies for 20 epochs. As expected, we observe that increasing the number of agents significantly reduces the testing error.

Visualization of Communicated Messages: We explore the patterns in the messages that are broadcasted by the agents using the learned communication medium. The goal is to visualize how the agents express the shared memory within an episode for solving the task. After training, we simulated 500 sample trajectories with $T = 9$ and recorded the messages of all agents at every $t = 0, \dots, 8$. Then, we used the dimensionality reduction technique called t-SNE [22] to produce meaningful visualizations of the messages. In Fig. 9, we illustrate two t-SNE plots for the messages at $t = 0$ and $t = 6$, which correspond to the messages before any communication and after a few rounds of communication and observation, respectively. In these figures, every message, that is initially in $\mathbb{R}^{12}$, is reduced to a vector in $\mathbb{R}^2$ and is illustrated with a color corresponding to the true label of the image. The result of clustering at $t = 0$ implies that initially, there is no meaningful pattern in the distribution of the labels. However, as the agents move across the image and communicate, they construct an internal belief about the true category. The second figure shows a t-SNE plot after 6

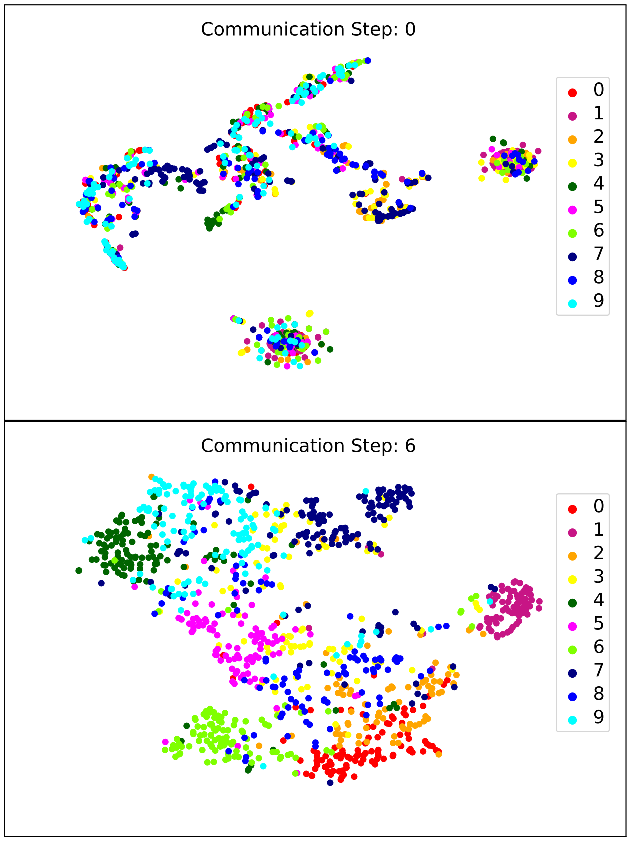


Fig. 9: Visualization of the learned communications strategies and how they share their beliefs with each other using t-SNE plots.

time-steps, which suggests that agents' beliefs are reflected in the communicated messages. In fact, we observe that the digits are now clearly clustered in their own groups; i.e., they have learned to broadcast their beliefs about the true labels to their neighboring agents.

Video: We have created a short minute video describing our framework as well as the result of our experiments, which is accessible online through the following link:

`https://youtu.be/j67sy8RK_A4`

VIII. Concluding Remarks

We introduce and analyze a multi-agent image classification framework using the vanilla policy gradient as the core reinforcement learning technique. The underlying ideas that are discussed in this paper are applicable to the other value-based, policy-based, or even while using novel variance reduction techniques [23]. The problem studied in this paper has a discrete action space within a typically short time horizon. Depending on the problem structure (e.g. in aerial robotic applications), one may prefer a continuous action space. Then, it is straightforward to generalize our methodology to deal with these policies within this framework; for instance, using Gaussian policies [24].

Our extensive simulations suggest that the current models are temporally robust: if we train the model for a time horizon $T = T_1$ and execute the model for $T = T_2 > T_1$, the prediction quality using the second model will remain at a meaningfully high level. Also, changing the number of agents would not result in dramatic performance degradation. For example, a model trained with 3 agents will still produce acceptable outcomes for the problems with 2 and 4 agents. Due to space limitations, we have not included the related numerical experiments.

In this paper, our numerical experiments have been limited to 2-D image classification. However, the proposed framework can be applied, with minor adjustments, to more realistic scenarios. For instance, as explained in Example 3, the current framework allows the classification of 3-D objects using a sequence of intelligently chosen 2-D observations conducted by moving agents. The optimal (stochastic) movement of the agents around the object could be potentially related to the *next best view* problem [25]. Therefore, we expect that the sample efficiency of our methodology can be potentially enhanced by incorporating the developed optimal movement theories (e.g. see [26]). It is also an interesting line of research to study the cases where the graph structure dynamically or randomly evolves over time.

References

[1] M. A. K. Bahrin, M. F. Othman, N. H. N. Azli, and M. F. Talib, "Industry 4.0: A review on industrial automation and robotic," *Jurnal Teknologi*, vol. 78, no. 6-13, 2016.

[2] L. Yushi, J. Fei, and Y. Hui, "Study on application modes of military internet of things (miot)," in *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, vol. 3. IEEE, 2012, pp. 630–634.

[3] C. Bhatt, N. Dey, and A. S. Ashour, *Internet of things and big data technologies for next generation healthcare*. Springer, 2017, vol. 23.

[4] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems*, 2017, pp. 6379–6390.

[5] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, "Fully decentralized multi-agent reinforcement learning with networked agents," *arXiv preprint arXiv:1802.08757*, 2018.

[6] A. Khan, C. Zhang, V. Kumar, and A. Ribeiro, "Collaborative multi-agent reinforcement learning in homogeneous swarms," 2018.

[7] D. Silver, L. Newnham, D. Barker, S. Weller, and J. McFall, "Concurrent reinforcement learning from customer interactions," in *International Conference on Machine Learning*, 2013, pp. 924–932.

[8] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

[9] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 1175–1191.

[10] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[11] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 2137–2145.

[12] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[13] T. Nguyen and S. Mukhopadhyay, "Selectively decentralized q-learning," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2017, pp. 328–333.

[14] M. Fazel, R. Ge, S. M. Kakade, and M. Mesbahi, "Global convergence of policy gradient methods for linearized control problems," 2018.

[15] S. Alemzadeh and M. Mesbahi, “Distributed q-learning for dynamically decoupled systems,” *arXiv preprint arXiv:1809.08745*, 2018.

[16] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[18] A. D. Domínguez-García and C. N. Hadjicostis, “Distributed strategies for average consensus in directed graphs,” in *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE, 2011, pp. 2124–2129.

[19] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[21] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” in *NIPS-W*, 2017.

[22] L. v. d. Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[23] R. Johnson and T. Zhang, “Accelerating stochastic gradient descent using predictive variance reduction,” in *Advances in neural information processing systems*, 2013, pp. 315–323.

[24] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, “Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search,” in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 528–535.

[25] C. Connolly, “The determination of next best views,” in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 1985, pp. 432–435.

[26] C. Esteves, C. Allen-Blanchette, A. Makadia, and K. Daniilidis, “Learning so (3) equivariant representations with spherical cnns,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 52–68.