Swarms on a Mission

Jan M. Rabaey[1,7], Amandeep Singh[1], Ali Moin[1], Mieke De Ketelaere[1], Jo De Boeck[1], Tom Ashby[1], Pascal Costanza[1], Piet Demeester[3,1], Bart Dhoedt[3,1], Ellie D'Hondt[1], Auke Ijspeert[6], Steven Latré[4,1], Sam Leroux[3,1], Myunghee Na[1], Peter Simoens[3,1], Werner Van Leekwijck[4,1], Marian Verhelst[2,1], Tim Verbelen[3,1], DD Verkest[1], Roel Wuyts[1,2]

Imec[1], KU Leuven[2], UGent[3], UAntwerpen[4], VU Brussels[5], EPFL[6], UC Berkeley[7]

ABSTRACT

Distributing Artificial Intelligence (AI) over multiple nodes or devices makes perfect sense when *information and/or the resulting action is spatially distributed*. In addition to reducing latency in decision making, it leads to solutions that are more robust and reliable in the presence of failures or interference, and provides better security and privacy. *How to distribute the intelligence over the devices that may be heterogenous in their capabilities, and how to efficiently and effectively make nodes collaborate, coordinate and join forces are questions that are under intense research and debate as we speak*. The challenge becomes even more profound when these nodes are mobile, the network changes dynamically, privacy of data and/or models, and explainability become important. This White Paper describes the state-of-the-art from a functional and algorithmic point of view, and provides preliminary insights into where and how technology, circuits, architectural and algorithmic innovations may be necessary in light of tight energy and form factor constraints. It concludes with a couple of recommendations for future actions.

INTRODUCTION: WHAT AND WHY

There are many reasons why intelligent processing is migrating from the Cloud to Edge and Extreme Edge devices. Perhaps security and privacy reasons may be a strong motivation for processing information locally, and sharing only essential outcomes over the network. The energy cost of communicating raw data over wireless links may be another one. Moreover, latency, fast response to changing conditions, and robustness in the presence of failure or interference make it even more desirable to have a significant part of the processing performed locally. Think, for instance, about intelligent wearable devices such as lightweight AR/VR glasses, "babel-fish"-like earpods, or intelligent medical implants such as brain-machine interfaces that observe, diagnose and stimulate. Finally, the combined computational power of multiple devices working in parallel may lead to faster learning and understanding. All these ideas can be clustered under the header of "distributed AI".

Yet there are other motivations that go beyond the pure distribution of machine leaning functions over multiple devices. This is the case whenever information and the resulting action/reaction is spatially distributed or/and when devices have different capabilities and functionalities. The most compelling scenario is when these devices establish a level of autonomy, in which they not only observe locally, but also learn, reason, decide and act in close form to solve problems. They may do this in concert and in coordination with other autonomous but collaborating agents. Consider for instance autonomous mobile entities such as drones and robots in any form or shape as the (extreme) edge devices. Most of the time we think about these devices as individual entities, performing one or more designated tasks – such as, for instance, a Roomba robot clearing a floor or mowing a lawn. However, for many tasks it may be more desirable to have *a group/cluster/flock of devices collaborating*. Those could be identical devices, such as a swarm[1] of drones, or they could be heterogenous, each with their own capabilities, such as drones collaborating with ground-based robots. The collaboration can be as simple as building a joint understanding of the world around it. However, we can imagine scenarios in which the individual modules combine forces and together form an ensemble of higher complexity, such as assembling furniture [ACK 20].

Some simple examples of recent activities and projects can help to illustrate the concept. Already in 2012, the Kumar group at the University of Pennsylvania demonstrated a group of drones doing joint maneuvering indoors (Fig 1a) [KUM 13-19]. The processing and coordination of this realization was highly centralized, however, with all the sensing done in the surrounding environment (using a VIKONG localization system), and only commands sent to the drones wirelessly. In 2018, the same group demonstrated a similar experiment outdoors, this time with a fully decentralized realization (Fig 1b) [KUM 13-19]. One possible industrial application of such an approach could be to have heavy machinery such as agriculture, decontamination or mining equipment, surrounded by a swarm of drones to help in terrain mapping and guidance (Fig 1.c). More down to earth, mobile components acting together could help create a far more conducive environments for assisted living, as is envisioned in the Roombots project in the Ijspeert group at EPFL (Fig 1.d) [ACK 20, HAU 20].

Intelligent swarms can be accomplished in many different ways, depending on the amount of communication between the nodes, how tasks are distributed and where decisions are made. On one extreme, one has the fully centralized approach in which the edge nodes share all their observations with a centralized processor that performs the decision making and action planning, and translates them in control commands for the remote entities. On the other extreme, we have a fully decentralized solution where nodes are fully autonomous and collaborate through observation and limited communication, similarly to social insects such as ants or bees. In between, we can imagine scenarios in which entities share what they learned and what they know (either by inter-node communication or through a centralized agent).

To explore the trade-off's, challenges and opportunities in realizing truly distributed AI that fit within the form factors and energy budgets of Extreme Edge devices, we envision a "*swarms on a mission*" scenario.[2] In this, a group of small devices gets assigned a task(s) to be executed. This could be the mapping of a large agriculture field and deciding where to harvest, the discovery of human survivors after a major disaster, the guidance of a convoy through unknown terrain, or to help support the elderly in living-at-home conditions. It could also be a collection of wireless devices, deciding what spectrum to use under interference conditions so that the capacity is optimized (see, for instance, the DARPA Spectrum Challenge [SPE 19]). In the extreme case, it may be a group of digestibles or submersibles nanobots searching and addressing an anomaly in the

[1] To clarify: when we use the terms "swarm" in this paper, we allude to a group of dynamically interacting devices. This is only marginally related to the often-used term of "swarm intelligence", which refers to the collective behavior of decentralized self-organized systems [SWA 20] (such as ants), in which each individual device is very simple and limited.

[2] Notice that the form factor and the energy budget of an "extreme edge" devices can vary over orders of magnitude, so does the computational capability. An autonomous car can obviously carry a much larger load then a flying bot. However, as nature shows us, the need for computational complexity scales with system complexity (humans versus flies). In this paper, we assume that energy of computation will be a constraining factor, independent of the size or complexity of the device.

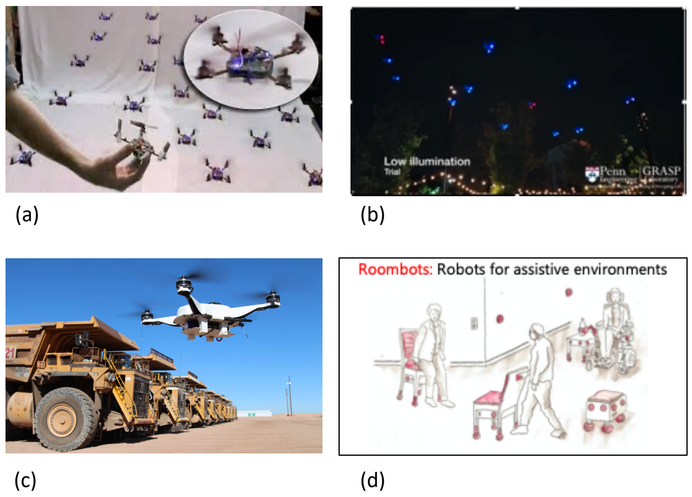

Fig.1: Examples of collaborative swarms. (a) drone swarms controlled by centralized agent [REF]; (b) full decentralized drone swarms; (c) drones assisting heavy machinery; and (d) robot swarms in assistive environments.

human body. The task at hand needs to be divided over the nodes, information gained needs to be shared and actions decided upon. There may or may not be a centralized node helping to orchestrate the operations or provide background information (such as pre-trained learning), and there may be humans involved in the whole process. Some nodes may want to limit what is shared so as not to divulge proprietary internal information on how they accomplish their tasks, while some other nodes may be hostile and try to disrupt by providing fake information. *One prominent task for research to address is how to create solutions/platform/components that can be rapidly deployed across these widely different applications*. Phrased differently, how can we make reusable components that can be parameterized/configured to the different use cases with wide variance in environmental settings and optimization goals.

While the swarms-on-a-mission scenario builds on many concepts developed in the AI community, it brings a number of novel aspects to the table, which makes it an interesting opportunity for innovative implementation platforms. More precisely: individual nodes must realize a variety of different functions including observation, learning, recognition, reasoning and decisioning; intelligence is not centralized but distributed; and communication plays a central role in the whole system concept. What, when and how to communicate information will play a decisive role in determining the responsiveness, robustness and safety of the resulting swarms. And again, how to efficiently implement and deploy these nodes starting from configurable and reusable components, hence avoiding the trap of having to develop a customized solution for each individual use case.

In this paper, we first present the state-of-the-art relevant to autonomous and distributed AI from an algorithmic point-of-view. We then identify the design metrics and design constraints that must be considered if one wants to arrive at practical realizations of these swarm devices and systems. Combining the two helps us to identify opportunities from an implementation perspective (algorithm, architecture, circuit, device), and prescribe some opportunities for high-impact research on that front. Note that smart sensors and actuators play an equally important role in the realization of swarm nodes, but their realization is out of the scope of this paper.

THE STATE OF THE ART IN AUTONOMOUS AND DISTRIBUTED AI

This section aims to provide a brief review of the current state of the art in areas of distributed artificial intelligence and autonomous control of multi-agent systems, two areas that are critical in enabling our vision of achieving true swarm intelligence. It is worth mentioning that intelligent swarms and distributed AI have been the topic of research for quite a while, but activity and interest have picked up substantially over the past few years. A broad spectrum of academic research groups has been contributing to the progress in the field[3]. Recently, the C-BRIC multi-university center [CBR 18] was funded by the SRC/DARPA JUMP consortium to explore a new generation of autonomous intelligent systems, bringing together some of the leading academic researchers in the field.

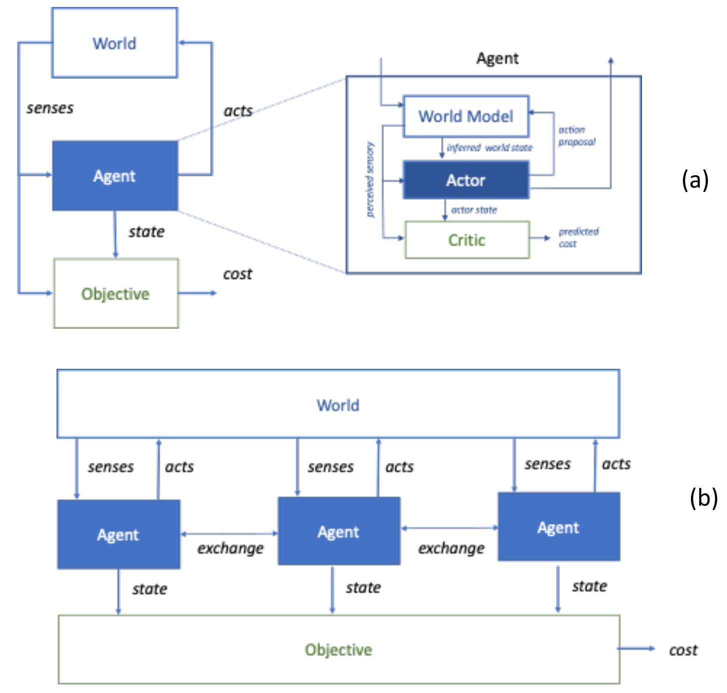

Fig. 2: Anatomy of an autonomous agent. (a) Agent to perform planning, reasoning and decision making based on learned/inferred view of the world [LEC19]; (b) collaborating agents share information about their world view.

Before diving into the details of the most prevalent techniques in use today, it is worth penciling the big picture first. State-of-the-art machine-learning techniques have demonstrated huge success on various perceptual tasks. Going beyond and to realize true swarm systems requires the conception of agents with an ability to reason, plan and make decisions. The anatomy of a prototypical swarm agent is shown in Figure 2a. In its simplest form, it can be decomposed in two functions: (a) An inner Agent network that essentially presents a generative model of the world, which the agent learns through its interactions with the environment in a self-supervised setting.[4] Possible approaches to this include variational auto-encoders and generative adversarial networks (GAN); (b) An outer reinforcement learning function that uses the generative model of the world to generate and imagine possible future trajectories, which are then used for planning and decision making. The Critic evaluates the trajectories and informs the Actor what action(s) to select to, for instance, maximize a scalar reward signal. This also reflects how the brain might work according to the active inference framework [FRI 16]. In this case, the Critic minimizes the expected free energy or future prediction error of the generative model. In contrast to maximizing a scalar reward, minimizing the expected free energy also comprises minimizing ambiguity, minimizing complexity and maximizing epistemic

[3] For the interest of the reader, we enumerate a list of some of the leading research labs in swarms and distributed robotics (without any claim of completeness): Vijay Kumar and Mark Yim, Alcherio Martinoli, University of Pennsylvania; Radhika Nagpal, Harvard; Daniela Rus, MIT; Gaurav Sukhatme, USC; Alcherio Martinoli and Auke Ijspeert, EPFL; IDLab, University of Antwerp; and others.

[4] Note that for some systems the world model may be known or may be pre-trained (e.g. the map of the operational space is known in detail). This would eliminate the learning step (or at least part of it).

value, which are indispensable in real-world scenarios [e.g. CAT 20, CAT 20a]." Observe that this diagram is purely functional, and that it does not imply where these functions actually reside. By necessity, sensors and actuators are distributed in the environment. All other functions could be performed locally on the device next to the sensors, on a mobile device, or centralized on a remote server, or partitioned over all of them.

A multi-agent version of this architecture is shown in Fig 2b. In this case, agents work together to create the world model, either by direct interaction or through a centralized server. This differs from the single agent case in that each agent may have only a partial worldview and that information exchange is an essential part of the picture. *How*, *how often* and *how much* are hence critical questions to be addressed. From an actor perspective, each agent may pursue its own individual task, or multiple agents can collaborate to accomplish a joint task.

The remainder of this section is hence structured along the same lines. We first describe approaches for distributed learning, essential in the creation of the world model. This is followed by an overview of the state-of-the-art in distributed intelligent control, addressing the actor part of the agent(s). Observe that we do not cover (distributed) inference, as this domain is the most mature and covered extensively in the literature [e.g. SZE 20]. In addition, energy-efficient realizations are steadily becoming available. This limits the room for innovation[5]. The same holds for the area of distributed decision making (consensus building, leadership election, …) that is relevant to the intelligent control part of the agents (see, for instance, the seminal work of Nancy Lynch at MIT [LYN 20] on those topics).

Distributed Learning

In a group of swarm agents, each agent is characterized by limited observability, i.e. each agent has at best a partial view of its environment at any time, limited compute and memory capacity and an uncertainty in communication with other agents. *How to enable a comprehensive vision on the environment surrounding the agents by collating data and enabling collaboration amongst these agents is an open problem.* The sub-sections below aim to provide a brief overview of some recent advancements and different approaches in distributed learning (See Table 1 for a global overview), and how these could be applied to swarm intelligence. A more detailed overview of distributed learning methods and architectures can be found in a recently published survey [VER 20].

	Federated	**Decentralized**	**Split**
Location of learning	Central and local (concurrent)	local	Split between central and local
Communication	star	peer-to-peer	star

Table 1: Classification of distributed learning approaches.

A. Federated Learning

[5] This statement does not imply that no further progress or advances can be expected in the domain of distributed inference. Available methods and hardware for edge devices are still very limited and focus almost uniquely on image classification. There is no question that the swarm domain will be open new opportunities and insights in this domain as well.

Federated learning (FL) [BRE 17] is a machine learning approach in which multiple agents/clients (e.g. edge devices, mobile phones) collaboratively train a single global model (typically one or another form of a DNN) under the orchestration of a central server while keeping the training data decentralized.[6] Essentially, each client calculates a local update to that model based on its own data, and the central server then aggregates the updates from all the clients to update the overall model. The updated model is then sent back to each client for future updates. FL tries to leverage the fact that stochastic gradient descent (SGD) forms the backbone of most of the state-of-the-art deep learning algorithms, and scaling SGD to a massively parallel setting has been quite successful for large-scale training in datacenter (e.g. the Google DistBelief [DEA 12]). Consequently, a key question is *"Can we parallelize training over a large number of agents, or edge devices each constrained with limited computation, energy and communication capacity?"* However, unlike parallelizing SGD in large-scale data center settings, Federated Learning across separate devices brings a unique set of challenges:

- Computation – Communication Tradeoff: Unlike in a datacenter setting, the lack of an efficient, low power, reliable & dedicated communication channel amongst different agents and the central server makes FL more complex. How to manage the energy tradeoff between on-device computation and communication amongst agents and central server while minimizing loss in algorithmic accuracy is one of the most important challenges in FL. Recent works such as [KON 16] [SUR 17] try to reduce the communication requirements by compressing, sparsifying, and quantizing the gradient updates while minimizing the loss in training accuracy. In [CAL 18], authors further try to reduce communication cost by compressing the model update being sent from central server to client nodes. However, it is still unclear if we have found an optimal distribution between communication, on-device compute and training/test accuracy. Hence, more work is warranted in this direction, especially in a swarm setting. Another approach allows local nodes to judge whether their updates are significant, and only send model changes if this passes a threshold.

- Non-IID Data Distribution: Since each agent gathers its own dataset, it is likely that the data gathered by different agents is not independent & identically distributed (IID). For example, different agents may observe covariate shifts, i.e. have different feature distributions, or distribution of the label that each client views may be different. Presence of locally non-IID data makes convergence of algorithms difficult and often slower and increases the error in optimization, although some progress has been reported recently on the front [ZHA 18].

The initial results in FL have been encouraging, and the first commercial use cases have emerged, such as in Google's GBoard Mobile Keyboard [AND 18, SUN 19], and Apple's QuickTime Keyboard and vocal classifier [APP 19]. Yet, valid questions arise as to how and if FL could be applied towards building swarm intelligence. In particular, the only demonstrated use cases are for supervised learning tasks in a privacy preserving setting (where individual training data is not shared in order to build the global model). Extending FL to a myriad of other intelligent tasks such as for generative modeling, reinforcement learning and unsupervised learning is largely unexplored. Furthermore, most of the algorithmic success in FL has been restricted to applying vanilla stochastic gradient descent (SGD), and there is no consensus yet around how to incorporate various other first order-

[6] One of the main motivators behind this approach is for security or privacy considerations. Yet it may offer advantages in terms of communication efficiency and consistency.

optimization techniques such as momentum methods and variance reduction techniques in a FL setting constrained with limited communication bandwidth [JIA19]. Note also that the limitation of learning a single model limits its potential applicability to fairly homogeneous swarms with nodes that essentially all use the same model.

B. Amalgamated learning

(Privacy-preserving) Amalgamated Learning (PPAML) [CEU 19] is a distributed machine learning approach where each agent/client learns and uses its own prediction model on its own data. The accuracy of each of the local models is improved by sending information derived from the node to a broker (which can be centralized or distributed) that aggregates this information and sends back information to improve the models of the individual clients. PPAML was developed to be applicable in contexts where neither the raw data nor the complete local models can be shared between partners due to privacy or commercial confidentiality reasons – for example, for the IP protection of the models. This has a side advantage that the amount of inter-node communication is reduced as well. Because different agents can have different models, PPAML could be applicable in heterogenous swarms where agents' behavior can differ more than what a single model can generalize. On the other hand, it can still be considered a centralized technique since part of the learning phase is a centralized machine learning step.

C. Peer-to-peer fully decentralized learning

The need for a central server to orchestrate the collection and aggregation of the model updates limits the potential of federated learning or amalgamated learning in several mobile and collaborative learning scenarios. Further, while federated learning adds partial redundancy and makes the system robust to failure of one or more of the clients, it makes the solution prone to a single point of failure which is undesirable. Also, the communication between client and server becomes a serious bottleneck as the number of clients increase.

Fully decentralized learning replaces the communication between the agents and a centralized parameter server with peer-to-peer communication between individual clients. Such fully decentralized learning algorithms for machine learning have attracted a lot of interest recently [DUC 12] [COL 16] [LIA 17] [LIA 18]. In essence, most of these algorithms try to implement a fully decentralized version of stochastic gradient descent (SGD) by way of a two-step algorithm: In the first step each client calculates its own local update, i.e. a gradient update based on its own local data, and in the second step all clients try to reach a consensus or an average model by means of averaging their own model parameters with those of theirs neighbors.

While initial results look promising, several algorithmic and practical challenges need to be overcome before such algorithms become usable in practical machine learning systems. From an algorithmic perspective there are several questions around the convergence of these algorithms for dynamically changing network topologies. Since clients may be temporarily unavailable or may drop out and rejoin, developing a better understanding of convergence of these algorithms under dynamically varying graph topologies is essential. Further, since the agents in swarm or edge devices are usually constrained in power, the energy consumption needs to be balanced carefully between on-device computation and communication amongst clients. Approaches such as merging gradient compression, quantization and sparsification, as proposed in [KOL 19], [KOL 20], may help to address some of these concerns. In addition, whereas each client may take several local updates before communicating its update to the parameter server in Federated Learning, convergence for fully decentralized peer-to-peer learning algorithms with locally non IID datasets and multiple local updates still needs further understanding [JIA 19]. From a more practical perspective, several questions remain such as which agent decides what is the underlying model that should be trained, which algorithm hyperparameters to use, and how hyperparameter tuning gets executed.

Finally, similar to Federated Learning, almost all the recent work so far focuses on supervised learning tasks. How to translate these algorithms to other machine learning tasks such as sequence models, reinforcement learning and unsupervised learning remain open questions. Again, the learning of a single model limits its potential applicability to fairly homogeneous swarms with nodes that essentially all use that same model.

D. Split Learning

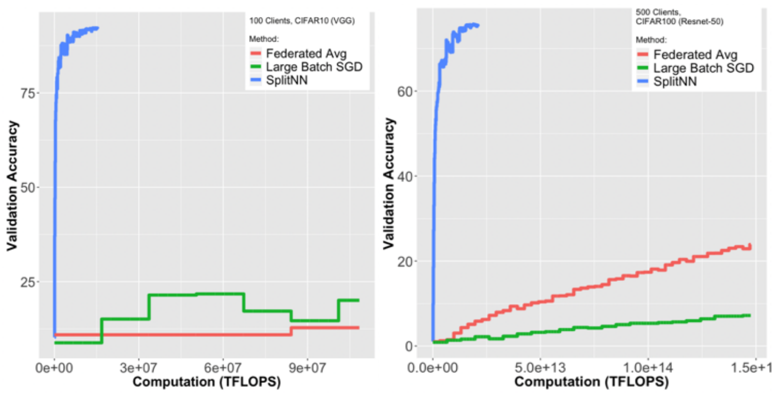

Figure 3a: Computational requirements (clients only) in Split Learning compared to SGD and Federated Learning for 100 and 500 clients, respectively (adapted from [VEP18]).

Method	100 Clients	500 Clients
Large Batch SGD	13 GB	14 GB
Federated Learning	3 GB	2.4 GB
SplitNN	6GB	1.2 GB

Figure 3b: Communication bandwidth required per client with a large number of clients. For setups with a smaller number of clients, federated learning requires a lower bandwidth. Large batch SGD methods popular in data centers use a heavy bandwidth in both settings. (adapted from [VEP 18]).

Split learning [GUP 18] is another distributed learning approach, in which we split the training of the model between the clients and a (edge, hub, central) server. In its simplest configuration the client locally executes the first few layers of the model/network up-to the cut layer. The output of the cut layer is transferred to the central server for further processing.[7] During backpropagation the gradients from the last layer to the cut-layer are calculated at the central server, and only the final gradients from the cut layer are transferred back to the client, which then completes backpropagation for the first few layers. The above concept has been extended to a multi-client / single server setting with promising results [VEP18]. A major advantage of split learning over federated learning is that it significantly reduces both the client-side processing requirements and the amount of data that needs to be transferred between clients and the server. This is because unlike federated learning where all the weights/gradients of the network need to be exchanged between the client and the server, here only the weights/gradients of the cut-layer need to be exchanged. Figure 3a [VEP 18] shows a comparison between per client computation needed to achieve different degrees of validation accuracy when training VGGNET on CIFAR10 dataset over 100 clients, and RESNET-50 on CIFAR100

[7] This model resembles to a degree the human sensory system (olfactory, vision) in which the first layers of the network are located next to the sensors, while the higher layers of abstraction are placed in the cerebellar cortex. The partitioning is done such that the communication bandwidth is minimized.

dataset over 500 clients using federated averaging, large batch SGD [GOY 17] and split learning. The communication requirements per node are presented in Figure 3b.

In a related approach, one can imagine a distributed DNN architecture that places bandwidth- and task-aware encoding at the edge device, while the core neural networks are located at the hub [CHI 18]. The encoders would be optimized for machine perception rather than human perception (such as the MPEG-4 encoder for video). This is a form of a hybrid network in which different NN models are combined.

While split learning between edge devices and servers is a promising direction for a number of applications, similar to federated learning it makes the system prone to a single point of failure. It is also heavily dependent on a reliable communication channel between the central server and clients. This makes it less attractive for large scale swarm networks operating in settings with limited communication reliability and dynamic reconfiguration.

A difference with federated learning is that split learning does not necessarily learn a single global model: the models can potentially structurally differ in the layers on the devices. One can imagine a setup in which the first layers differ as long - as the cut layer is the same. This setup embraces the situation where data gathered by different agents is non-IID. Local models for each agent are able to adapt to these variations over agents. Extensive research is needed to see how much differentiation in device models can be afforded in such a setup.

E. Transfer Learning

Another related learning technique that is important in swarm settings is transfer learning (TL). Transfer learning refers to learning methodology wherein a network pre-trained on a similar task is used as a starting point for the learning of a new network. It allows for improving the performance of a network on a new task through the transfer of knowledge from a related task that has already been learned. Transfer learning is particularly interesting for swarms in which the lifetimes of the agents may not all be the same. If one agent gets replaced, expires or added, the newly introduced nodes do not want to re-learn everything from scratch. Conventionally, transfer learning approaches have been quite successful in many computer vision and natural language processing tasks, such as [YOS 14] [RAZ 14]. It still needs to be established how well TL techniques extend to reinforcement learning and multi-agent settings. It definitely presents an interesting direction for further research.

In summary, a broad range of distributed learning approaches can be considered. What can work in practice is all a function of the memory, communication and compute limitations of the distributed devices. For instance, the availability of storage will determine the size of the model that can be learned. Latency is another concern. Finally, minimizing communications will be essential as wireless bandwidth is always at premium, and its availability may vary substantially: hence, “send only information that is needed, and send it as slowly as possible”. This will be discussed in more detail later.

Autonomous Control of Multi-Agent Systems

In addition to developing distributed and collaborative intelligence algorithms that enable us to gather insights from data, efficient algorithms for planning, control, coordination and communication between a swarm of agents are essential to enable them to accomplish meaningful tasks. Multi-agent planning and coordination is a long-standing problem, and has been well studied by researchers. Yet, only over the past one or two decades have we seen a convergence between traditional control systems theory and machine learning.

Most deployed solutions employ a centralized approach to the problem. In such a setting, the controller gets access to full information about the state of the environment, gathers data from all the sensors in all the agents (locally and globally), and then solves the planning, control and coordination problem as a giant optimization problem [SCH 18]. The outcomes are then conveyed to the agents. Unfortunately, this does not scale well with increasing numbers of agents. This rising computational complexity of the optimization problem may necessitate offloading the task to a central server with large computational resources, which further increases the communication bandwidth requirements and latency. Additionally, it is often hard to establish a reliable, low latency communication network between all the agents and the central node. Decentralized controllers can be scaled more easily, reduce the communication bottleneck, and are more robust to failure of one or more agents. Unfortunately, real-life performance of today’s decentralized controllers lags substantially behind that of their centralized counterparts.

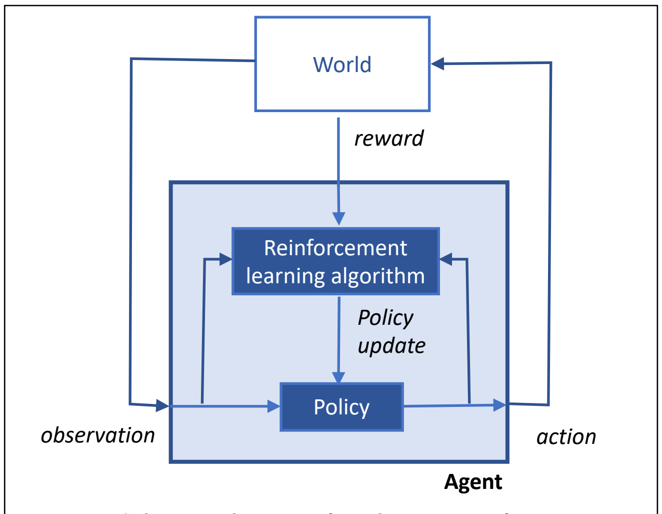

Figure 4. Schematic diagram of single-agent reinforcement learning framework.

The success of deep learning for various perceptual tasks in computer vision and natural language processing has inspired researchers to explore addressing sequential decision making and control problems as a machine learning approach as well. Among the many options to do so, *reinforcement learning* (RL) has emerged as the leading candidate. In an RL framework, the agent observes the environment, infers its state, and decides on an action based on a learned policy (Figure 4) that attempts to maximize the long-term reward. In its basic form, the control problem is modelled as a *Markov decision problem* (MDP)[8]. A policy presents a distribution of actions to undertake given that the agent is in a particular state, chosen so that it can maximize its long-term reward. While the basic concept of RL is quite old [SUT 98], combining RL together with deep learning models has had remarkable success in many different areas, such as in playing Atari Games [MNI 13], in using policy gradient techniques for playing games [SCH 15], in applying policy gradient and Q-learning techniques to real-world robotic problems [LEV 15],[KAL 18], and in combining supervised learning,

[8] Reinforcement Learning (RL) solves Markov Decision Process (MDP) when the MDP is not known, in other words you don’t know the states you can visit and you don’t know the transition function from each state.

policy gradients, value functions and Monte-Carlo tree search to beat the human GO champion [SIL 16].[9]

Given these successes, it was only logical for the research community to consider the problem of decentralized control of multiple agents as a *multi-agent reinforcement learning* problem. A collaborative multi-agent task can then be formulated as a decentralized-partially observed Markov Decision Problem (DEC-POMDP). Unlike an MDP, a DEC-POMDP is characterized by partial observability. The overall goal of the collaborative team of agents is to learn a joint policy that maximizes long term expected rewards. Deep reinforcement learning for DEC-POMDP can be partitioned along two classes, depending upon where and how the training is done, (a) Centralized training (CTDE) [RAS 18] [SUH 18], and (b) Decentralized (DTDE) [OMI 17] [AMA 19]. In both cases, the execution is decentralized.

A. Centralized Training with Decentralized Execution (CTDE)

The CTDE paradigm usually relies on use of simulation engines for their initial training. For example, a robot can learn path planning and object localisation in a *simulated environment*. The policies learned along the way are then transferred to real life devices using transfer learning techniques such as [SAD16, YEV19, STE19]. The key advantage of using CTDE is that during the simulated learning exercises all the virtual agents have access to all the global state information, and that parameters and gradients of the policy can be shared amongst all of them. This significantly simplifies the training task. However, the execution of the policy is decentralized, with each agent taking an action based on its own observed action-observation history. In practice, this is achieved by choosing the overall joint policy function for the agents such that it is easily factorizable into parts. For example, [RAS 18] proposed to use the joint policy function as the sum of all individual policy functions, and [SUH 18] improved upon the idea by forcing the joint action-value function to be any monotonic function of individual action-value functions. One disadvantage of the CTDE approach is that the simulation complexity puts an upper bound on how realistic the scenarios can be and on the scaling of the number of nodes involved. One could imagine a centralized learning scenario in which all learning is done on-line and all observations are transmitted to the central server. However, bandwidth requirements and reliability may make this approach impractical for constrained devices.

B. Decentralized Training with Decentralized Execution (DTDE)

Unlike CTDE, DTDE attempts to solve the DEC-POMDP without allowing the agents to share global state even during training. A major advantage of these techniques is that they are not limited to the "training by simulation" paradigm only, and that they can be coupled with on-line training techniques. This makes them very attractive for most environment which makes learning difficult. In [OMI 17] authors tried to overcome non-stationarity and partial observability by adding hysteresis and recurrent layers to deep-Q networks. In [AMA 19] authors also introduced hierarchical policies using macro-actions to further improve scalability of such techniques.

[9] There are other approaches to smart distributed control beyond reinforcement learning. Given the current success of RL and the limited scope of the paper, we choose to focus our attention on RL and its distributed formulations.

However, even after significant efforts current state of the art techniques in DTDE significantly lag in performance as compared to CTDE counterparts on most benchmark tasks.

While multi-agent reinforcement learning provides a promising learning-based approach to solving challenges in decentralized planning, control and coordination, a number of known challenges in RL need to still be addressed before we will see multi-agent RL for autonomous control in real life day-to-day applications. Some of these are: overcoming data inefficiency of RL by self-supervised learning; finding novel ways of imitation learning; inverse reinforcement learning to improve rewards engineering; and extending RL to hierarchical settings to extend scalability.

Other Relevant Approaches

In addition to above technologies in distributed AI, we anticipate graph neural networks and memory augmented neural networks might also play a major role in developing true swarm intelligence. Since the data gathered by a collaborative team of agents is inherently irregular, recent advances in graph neural networks [WU 19] which aim to apply deep learning to irregular data structures such as graphs could play an important role in swarm settings. Further, recent success in imagination augmented planning networks [WEB 17] and memory augmented neural networks [SEB 17] [GRA 16] [WAY 18] for solving tasks with partial observability make them prime candidates for use in swarm settings. Unfortunately, given the huge complexity and large energy consumption, these networks have yet to be brought to swarm agents with limited power and compute capacity.

Swarms and Distributed Robotics

The above-mentioned approaches have been applied to some extent to the swarms and distributed robotics application domains. This is definitely a hot area of research. Earlier, we have identified a number of active research efforts that are active in this domain (See footnote 3). While a number of these projects are based on intelligent control, other take inspiration from the biological world (such as swarms of insects). Given the focus of this paper on machine learning approaches, we will leave it to the interested reader to browse the provided references.

RELEVANT METRICS & CONSTRAINTS

As mentioned, there are a number of reasons why one may choose to bring the intelligence from the cloud to the extreme edge, such as operational conditions and availability of wireless connectivity, enhanced security and privacy, reduced latency, and improved robustness. An overview of the relevant metrics is presented in Table 2. It is crucial to consider the specific requirements and constraints of the specific application (i.e. the "mission) before making any decision on where to locate or how to partition the AI. For instance, energy cost may not be a concern when the edge-device is wall-powered, or when the compute power is minor compared to the power consumed by the mechanical components. Security may be of essence in autonomously operating devices (such as drones or bots), where malicious intrusions may lead to physical harm. In medical devices, safety and/or privacy may be the dominant decider. Another scenario where privacy-preservation is the priority constraint, while performance/energy savings are more relaxed, is the domain of *distributed edge computing*. These networks are rather static in nature, but communication is expensive because of privacy preservation rather than latency (while this not a scenario that one easily labels as a swarm, many of the same principles apply.

	Physical footprint	**Real time Closed-loop**	**Data transfer**	**Robustness**	**Security**	**Privacy**	**Scalability**
Measure	Ultra-low energy Cost	Low latency Local vs distributed Adaptivity /Accuracy	Overall bandwidth	Fault-tolerant Redundant Self-repair Safe fallback	Authentication Distributed trust	Locality of data and models	Complexity Convergence
Metric	Lifetime (sec) Volume (mm^3) Weight (g)	Response time (sec)	Bits/sec	Time-to-Failure (sec)	Vulnerability Overhead cost ($, J)	Cost ($,J)	Number of nodes (#)

Table 2: Relevant metrics for Extreme Edge devices, complement or superseding traditional Power-Performance-Cost (PPC) considerations.

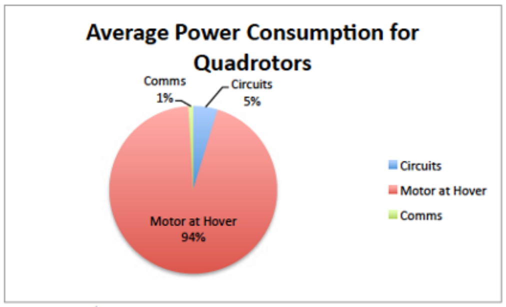

Figure 5: Power distribution in quadrotors.

As an example, consider the power budget for a swarm of drones. Simultaneous localization and mapping (SLAM) [SCH 18] and autonomous flight for search and rescue [SCR 15] are among the tasks that can benefit from autonomous and distributed AI. Figure 5 [MUL 14] shows that a significant portion of the power consumption in drones is spent on motors and mechanical parts. In fact, it is estimated that 200W/kg is needed just to hover the platform [MOH 18]. A typical drone such as DJI Flamewheel 450 used in [MOH 18] with all its sensing, computation and battery weighs 2 to 3 kg, which translates to more than 400W power just for hovering. An embedded GPU such as NVIDIA Jetson Nano burns 10W in total at its maximum, and a radio transceiver dissipates even less. Therefore, the power consumption of the mechanical portion dominates any savings available from optimizing the AI algorithms/hardware or reducing the communication costs. However, this balance does shift over time as the size and weight of drones is shrinking. For instance, the mini-drone platform used in [HUL 15] has a total power budget of only 35W. In addition, more sophisticated sensing modalities and increasing intelligence will most lead to a larger fraction of the overall power budget to be assigned to electronic processing.[10] This is apparent in the more recent the Skydio-2 drone [SKY 20]. It weighs only 0.75 kg and has a flight time of 23 minutes, while performing some advanced tracking functions using an NVIDIA Tegra TX2 SOC (7.5W), and video capture using an additional GPU, CPU and signal processors. It sports 45 MegaPixels of visual sensing from six 200-degree color cameras, as well as other video and motion sensors. With all of this, the electronics

[10] In a sense, this is analogous to the evolution in petrol-powered automobiles over the last 100 years, where electrical functions have taken an ever-increasing fraction of the overall power budget at the expense of the mechanical functions. This is both due to more efficient engines as well the introduction of electrical functions and "smartness".

(including a WiFi wireless) link take a much larger fraction of the power budget. This trend has definitely not been lost to drone manufacturers, who are increasingly developing their own custom SOCs to improve energy efficiency (and consequently mission or flight time).

To get a perspective on where this may lead, let us explore a hypothetical scenario, in which we add distributed learning and autonomous decision making to the drone. To make it more concrete, assume a task similar to learning a ResNet DNN. From [BAR 20, Table 2.1], we derive that this would take 3 MJ of energy on a 1 TOPS/s/W GPU. On a 10W GPU (Jetson Nano), it will take 167 hours to complete this learning. Even if we do it collectively on 100 drones, it is still 1.7 hours per drone – which translates in many learning sessions. To do it in shorter time you would use a more powerful processor (a 100 W GPU might do it in 10 minutes). The point is that innovative solutions (algorithm, architecture, circuit) will be crucial in making autonomous drone and robot swarms a reality.

The trade-off will be even more pronounced in other platforms such as ground-bound robots or swarms of micro- or nanobots, depending on overall energy supply, operational circumstances and expected duration of the mission. For instance, the processor in a coin-size ground-bound swarm robot [TAR 16] consumes a comparable amount of energy compared to motors and camera-based sensors (actually closer to 80%).

A NOTE ON BENCHMARKING AND TEST BEDS:
Standardized environments and benchmark tasks have played a significant role in moving machine learning in general and single agent RL more specifically from toy problems to real life examples. However, lack of such standardized environments for multi-agent cooperative control tasks has been hampering progress in this domain. Fortunately, two benchmark tasks, "Starcraft Multi-Agent Challenge (SMAC)" [SAM 19], and "Multi-Agent Mujoco" [WIT 20], have been recently proposed. SMAC is based on the popular real-time strategy game StarCraft II. It focuses on micromanagement challenges in which each unit is controlled by an independent agent that must act based on local observations. It may be noted that, while SMAC can be a good benchmark for many control problems in multi-agent robotics, SMAC restricts the action space to discrete actions only. Multi-Agent Mujuco [WIT 20] on the other hand allows for continuous actions as well. It must be noted that neither SMAC nor Multi-agent Mujuco allow for the presence of a dedicated communication channel between agents. Presence of local communication channels, albeit with finite probability of failure, can significantly aid in solving multi-agent control tasks, and hence should be considered in future benchmark tasks.

SWARMS ON A MISSION: FROM CONCEPTS TO REALITY

It is fair to state that major progress has been made over the past few years in bringing energy-efficient *inference* to Edge and Extreme Edge devices. Solutions include innovation at the algorithm (network architecture, pruning, numerical accuracy), architecture (logic/memory interaction, matrix-vector multiply accelerators) and circuit/device level (in-memory and in-sensor computing). As a result, facial, voice and object recognition functionality has started to appear in IoT and wearable devices, albeit often in a simplified and limited format. Examples are the speaker recognition in the Amazon Echo, face recognition on the iPhone, and the person recognition on the Google Nest.

Building effective swarms of autonomous edge devices requires all of this, and then some more. The necessary functionality goes substantially beyond the recognition and classification of well-known objects and signals in the world around it, and includes the interpretation of novel conditions and environments, the decision of what actions to take based on these observations, and the learning of the efficacy of these actions (that is, the *reward*). Building these autonomous edge agents requires algorithms that allow agents to build world models of its environment in a self-supervised manner by intelligently interacting with the environment, and then using the world models to learn from past memories, predict, plan, and make decisions. The distributed learning and control algorithms described in previous sections definitely provide us with a window on what these functions may look like. Yet, one conclusion is abundantly clear: **realizing these functions on severe {power, size, bandwidth}-constrained devices will require design improvements of orders of magnitude with respect to the current state-of-the-art in machine learning hardware, while meeting the robustness, security and privacy constraints of the application domains**. In addition, solutions must be able to cope with rapidly varying dynamic conditions caused by changes in the environment, circumstance or varying network topologies. Beyond the realization concerns, it is also worth highlighting the crucial role to be played by simulation and verification environments. Evaluating the robustness and the trustworthiness of the learned models and policies may requires computational power many times that of the learning and inference tasks. While of uttermost importance, the task of developing powerful simulation and verification environments is beyond the scope of this paper.

From the previous sections, we also learned that fully-centralized solutions, where the brunt of the computations are performed on a server or servers - where ample power is available - is not a true option from an applications perspective, due to bandwidth, reliability, security and privacy considerations. The actual solution we may converge to in the end may very well be a hybrid centralized/distributed one: expensive functions that are not latency constrained and only need to be done on an occasional base (such as interpreting a new object), or learning of models across many devices, are performed on the server, while anything that needs quick response time, requires high reliability, or is intrinsically related to a specific location is performed on the distributed device, or on the Edge. However, from a high-impact research perspective, it pays to focus on a “push-the-boundaries” challenge: what would it take and what are the paths to realization of *a* **fully-distributed autonomous and collaborative swarm intelligence with life-long on-device learning**.

NOTE: This is an impact-driven decision. This should by no means preclude a parallel agenda and roadmap, exploring the hybrid Cloud-Edge-Extreme Edge Scenario [see, for instance, NTT VLSI 2020].

To that effect, we evaluate the algorithms and approaches introduced earlier and identify unique challenges and opportunities for efficient realization (as compared to the state-of-the-art).

Unique challenges:

- *On-device learning* with extremely limited computational and energy resources.

This includes building local world models, simulation and verification of policies in those models, gathering experience, and evaluating the effectiveness of local actions.

- *Lack of supervision, and reliance on exploration and self-supervision* inherently implies that the algorithm shall need many learning interactions with the environment to reach optimal performance.

Collaboratively learning the world model in a decentralized manner across multiple edge devices might help reduce the computational load for each device ("shared education"). This of course raises the question of the cost and overhead of *inter-device communications*.

- *Fault-tolerance and resiliency against adversarial attack*s.

The system should be robust against malfunctioning agents as (ideally) also against adversarial agents.

- Operating under the *constraint of limited memory.*

As stated earlier, the limited availability of memory at the swarm nodes impacts the size of the models that can be stored and operated on. The impact is however is even more profound from an architectural perspective. In a traditional deep learning setting, such as in cloud-based learning of large networks for perception, the presence of large batch sizes during learning allows us to exploit extensive parallelism to improve performance and reduce the power consumption, especially for the pervasive matrix multiplications. When memory is constrained, so is the sized of the available data that can be operated on. This by necessity leads to a different memory hierarchy and data flow.

- Operating under *limited communication bandwidth*.

Given the decentralized nature of the swarm model, wireless communication and its limitations has a major impact on the solution space. The amount of information that can be shared over a period of time is limited. In addition, noise and interference can impact the reliability and robustness of the information shared. And finally, varying channel conditions and network topology requires dynamic adaptation.

- Operating under *dynamically-varying conditions*.

Not only the wireless network conditions change dynamically, but so can the environment, the number of agents and their location, and the workloads and expectations. Some of these changes may occur quite rapidly, or can be quite devastating (for instance, a denial of service attack can shut down most network operation). Hence, swarm AI has to be able to effectively deal with change through adaptation and reconfiguration.

- Dealing with *heterogeneity.*

Swarms can consist of drastically different devices with different capabilities and roles and different "views" of the world. Consider, for example, smart construction sites or smart agriculture. The learning and operating functions such a swarm will be much more difficult since transferring (parts of) models or aggregating model parameters between devices becomes far less effective or impossible when the devices differ too much.

- Programmability and adaptivity.

How to program a swarm consisting of many distributed devices? Individually programming devices that need to intimately collaborate is difficult, error prone, hard to test and deploy, maintain, and upgrade. Rather, swarms should be programmed and deployed as a whole. There are some strong similarities with regard to requirements solved by modern PGAS-style HPC programming languages (Chapel, Fortress, X11, Shark, UPC, CoArray Fortran), which view an HPC system conceptually as a single compute system rather than a collection of independent units (cf. MPI), but then distribute the computation over several nodes. This not only allows for a higher level of abstraction when designing the system, but also provides a number of unique guarantees with regard to data locality and communication bounds, among other things. By backing a PGAS-style programming language with a resilient work-stealing scheduler (Cilk Cobra, Go, TBB), it should then be possible to harden a computational swarm against inadvertent dropouts of single drones or other failures.

A number of possible approaches to address these challenges can be considered.

Unique Opportunities:

- Develop novel training schemes that enable **sparse, low-resolution and compressed training** of deep generative and reinforcement learning models to reduce memory and power consumption; Derive **compute architectures exploiting sparsity, low-resolution operations, and unique dataflow and memory hierarchy** that emerges in reinforcement leaning and generative models to reduce memory and power consumption during training,

The size of neural networks (with some networks now fielding tens of millions to few billon parameters) makes it difficult to store and evaluate these models on edge devices with limited memory. Hence the need for aggressive reduction of the memory need. Initially focused on *inference* for perceptual tasks such as image classification, techniques such pruning, sparsification, weight and activation quantization, and model compression have shown major success [HAN 16, ZHU 17, VER 18]. These techniques give an existential proof that there exists a sparse and low-resolution neural network that can achieve similar performance to a dense network for many tasks. Yet, translating these approaches to the learning phase has proven challenging. Fortunately, recent research has made some major progress, including a focus on learning sparse or compressed models from scratch, as well as direct training of quantized parameters [MOBv1, MOBv2]. One research avenue is to explore how these approaches (sparse, low resolution, compressed) translate to RL and generative modeling.

One major challenge is how sparsity can be combined with compute efficiency. For example, training strategies that directly work with low resolution weights and activations allow us to leverage novel analog in-memory compute architectures, which are limited in resolution due to the underlying physics of the devices. However, these “systolic” compute engines are not optimized for sparse matrix multiplications and are unable to efficiently use sparsity. On the other hand, large-scale sparsity holds a lot of promise for reducing memory and power consumption. Hence, how to design **efficient hardware for sparse operations** is an interesting venue for research. This includes structured and block-level sparsity [e.g. MAO 17]. Co-designing memory optimized software and hardware for sparse and block sparse RL and generative models could help us significantly in reducing power consumption. Also, how to incorporate sparsity in next generation analog in-memory computing architectures is an area of research with large potential.

The closed-loop nature of RLs adds another twist to this story. It involves gathering experience, updating world models, imagining trajectories for planning, and selecting actions. This makes the dataflow and data-memory interaction different from conventional deep neural networks – In fact, instead of operating of large batches of data, it uses only small local experience buffer composed of most recent states, actions and rewards which could reduce the raw memory needed, but further increases the memory-bandwidth requirements. This may require revisiting memory structure and hierarchy and how to optimally address and access memory for RL and generative modeling tasks. For instance, very-wide scratch memories and register files could be sufficient to hold all the necessary data for processing, and could help to perform in-situ data transformations.

- Enable efficient techniques for **distributed AI across multiple edge devices** to reduce each agent's individual memory and power requirement; Develop supporting **efficient wireless communication and networking approaches** to support the distributed algorithms at a minimum overhead cost.

In the Swarm setting, each distributed agent interacts with the world and learns based on its own observations using a small batch size, and then aggregates the model with its neighboring agents using various forms of consensus and gossip algorithms. Aspects of the world model can be shared or learned together by a group of autonomous agents. In addition to the world model, each autonomous agent can also learn a specific task or collaborate with other agents for shared tasks. Inspiration for this operational model can be found in OpenAI's RoboSchool framework for RL [ROB 17], a model that expands very well into distributed RL.

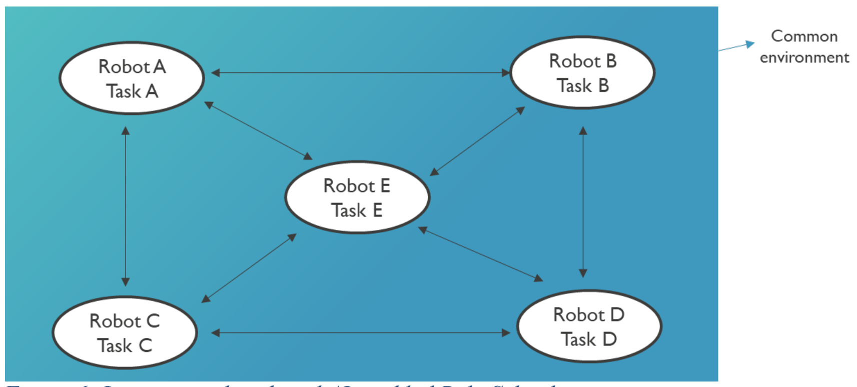

Figure 6: Low-power distributed AI-enabled RoboSchool.

The differentiating factor in this model is *the role of communication*. The cost and latency of communicating information over a wireless network should not be ignored. Privacy-preservation can also add important constraints on communication. Hence what, when and how information is communicated is of essence with as guiding mantra: "Send only the information that is needed and send it as slowly as possible".[11] Current distributed agents rely on standard wireless protocols. While proven, this may not be the most effective way of sharing observed or learned information with neighboring nodes. For example, wireless communication has the intrinsic advantage that broadcasting is for free. Hence information or knowledge can easily be shared with all who care to hear – supporting consensus and gossip algorithms.

[11] This may seem counter-intuitive. It is however a direct result of Shannon's Theorem. If you can slow down the required communication rate, you can reduce the energy/bit of information. This is a concept that biological systems (such as the brain) have perfected [STE 15]– but that is another story …

On the other hand, broadcasting restricts the exchange of information to robots that are within the communication range. As an alternative to proximity-based networks, other logical topologies, such as containing scale-free aspects or network motifs, may result in a faster spread of information across the swarm and yield a better task performance [TUR 20, RAU 20]. In such a model, agents transmit information to targeted robots on a longer distance. How a system consisting of mobile agents could self-organize its interaction network topology is an active topic of study [XIA 20]. Such self-organization would also have to account for the underlying hardware capabilities. For instance, an agent taking the role of a central hub will need to spend a lot more energy on network communication."

Finally, various means of temporal or spatial compression can be used. Using innovative data encoding and modulation schemes (such as high-dimensional data representations [KAN 09]), the communication itself can be used to perform computation ("in-network computation") [LIN 19]. This is a domain for research with wide open opportunities that has been minimally explored so far. Finally, it may also become worthwhile to look into adaptive protocols, in which the information shared between agents varies over time, depending on the devices and the task at hand (only sending what is needed). This can be done similarly to how bytecodes in networked virtual machine can be adapted depending on the application scenario in order to optimize communication (but dynamically so) [AND 04]. Essential to progress however are simulation and modeling environments for ML that include accurate models of wireless communication and networking.

- Explore decentralized distributed agents collaborating performing a set of tasks **under dynamically varying conditions caused by changes in the environment, circumstances or network topology**. Develop efficient algorithms/architectures (supporting dynamic pruning, resolution and sparsity) that can deal with dynamics.

The three most-differentiating factors of the "swarms on a mission" scenario are: (1) heavily constrained edge devices; (2) distributed; and (3) dynamic. Most of the research (as reported in this paper) has been focused on the former two issues. Yet, the dynamic nature of swarms is one of the most challenging problems. Wireless networks are dynamic by nature. Channel conditions vary over time, interference can cause links to be corrupted, and network topology can change dynamically. While wireless communication technology has evolved to mitigate most of these effects, the overhead can be large. In the end, the communication channel is still a stochastic process with a potential for large variability (a link can be completely dropped, for instance). In addition, the environment the swarms operate in changes dynamically, external circumstances can change the setting, nodes may come and disappear, etc. Hence, developing algorithms, models and architectures that can effectively deal with these dynamics are of essence. This may include architectures that can adjust accuracy or computational complexity dynamically, reconfigure to adjust to changing workloads, make fast context switches, learn by equivalence, etc. In the end though, one design principle will prove to be essential: whatever happens, the distributed system must prove to be robust and safe. While some of these concerns can be addressed through traditional wireless communication and networking techniques, others may require innovative solutions such as learning-based networks. The latter has received a lot of attention lately as is reflected in recent government programs and calls [NSF 19].

- Note that these are just some possible paths for progress, building and extending primarily on the current state-of-the-art. We may come to the realization that "same as before" will not do, and that we need radically different models and implementation platforms. Hence, other pathways may and will arise over the next couple of years, some based on novel algorithms and computational models, others driven by technology innovation.

For instance, the availability of large amounts of high density ultra-low energy nonvolatile storage, may allow for as much-as-possible pre-learning to occur. Replicating all the relevant information/ models on the distributed edge nodes would allow for fast adaptation to changing conditions (context-switching), and may help to reduce computation and communication.

At the algorithm and architecture level, neuro- and biology-inspired approaches building on fundamentally different compute and encoding principles may be well positioned to address the identified challenges of autonomous intelligent systems of the "swarm on a mission". This opportunity includes exploration of spiking neural networks [MAA 97], encoding information in the timing or rates of spikes, inherently use of event-based computation, and allowing new learning rules such as e-prop [BEL 20], STDP and its variants [SJO 10]. Also, different forms of population encoding using large sparse distributed representations (implemented either with spiking neurons [ZAM 19], in architectures inspired by cortical microcircuits [HAW17]), or using simple binary vectors like hyperdimensional computing approaches [KAN 09, KAR 20]), could bring unique benefits to future autonomous intelligent systems.
Of course, here too, a crucial part of the investigation is to engineer novel hardware components (both for the individual compute element and the tunable interconnections between them) and develop efficient learning and inference algorithms with those components.

RECOMMENDATIONS

In this paper, we have enumerated a set of options that can lead to true swarm intelligence to arise over the next decades. Based on these observations, we give the following recommendations for progress:

- Identify forward-looking **heterogenous multi-agent scenarios** (with partners). Translate into simple case studies to drive algorithm exploration in simulation environments (RoboSchool [ROB17] can serve an example). FOCUS: Establish playground and prototyping testbeds.
- Establish benchmarking methodologies, and **define quantitative comparison metrics** (expanding on Table 2). Should build on scenarios. FOCUS: Quantifiable metrics beyond PPC.
- Explore **decentralized distributed agents collaborating performing a set of tasks** under dynamically varying conditions caused by changes in the environment, circumstances, network topology, or swarm composition. Develop efficient algorithms/architectures (supporting dynamic pruning, resolution and sparsity) that can deal with dynamics. FOCUS: Dynamic trade-off and adaptation.
- Develop and deploy **a shared simulation and verification environment** for distributed autonomous swarms. This would serve as a playground for algorithmic development, and as a source for the collection of data statistics (computational complexity, performance, efficiency, robustness and reliability. FOCUS: metrics, prototyping.
- Explore algorithms and schemes that enable **sparse, low-resolution and compressed learning (and inference)** of deep *generative and reinforcement learning models*. FOCUS: HW-aware algorithm selection for distributed learning.
- Build novel **compute architectures exploiting sparsity, low-resolution operations, and unique dataflow and memory hierarchy** that emerges in *reinforcement leaning and generative models* to reduce memory and power consumption. FOCUS: In- or near memory computing exploiting sparsity.
- Enable efficient techniques for **distributed operation across multiple edge devices** to reduce each agent's individual memory, power and communication requirements. FOCUS: Balance between computation, storage and communication.

REFERENCES

- [ACK 20] E. Ackerman, Roombot Swarm Creates On-Demand Mobile Furniture, IEEE Spectrum, April 2020.
- [AMA 19] Amato, Konidaris, How and Kaelbling - JAIR 19.
- [AND 04] Jakob R. Andersen, Lars Bak, Steffen Grarup, Kasper V. Lund, "Design, Implementation, and Evaluation of the Resilient Smalltalk Embedded Platform," Proceeding ESUG Conference 2004.
- [AND 18] Andrew Hard, Kanishka Rao, Rajiv Mathews, Francoise Beaufays, Sean Augenstein, Hubert Eichner, Chloe Kiddon, and Daniel Ramage, "Federated learning for mobile keyboard prediction", arXiv preprint 1811.03604,2018.
- [APP 19] Apple. Private Federated Learning (NeurIPS 2019 Expo Talk Abstract).
- [BAR 20] L. Barosso, The datacenter as a computer, Morgan & Claypool Publishers, 2020.
- [BEL 20] Bellec, G., Scherr, F., Subramoney, A. *et al.* A solution to the learning dilemma for recurrent networks of spiking neurons. *Nat Commun* **11,** 3625 (2020). https://doi.org/10.1038/s41467-020-17236-y.
- [BRE 17] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas, "Communication-efficient learning of deep networks from decentralized data", In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, pages 1273–1282, 2017.
- [CAL 18] Sebastian Caldas, Jakub Konecny, H Brendan McMahan, and Ameet Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements", arXiv preprint arXiv:1812.07210, 2018.
- [CAT 20] Çatal, Ozan & Verbelen, Tim & Nauta, Johannes & De Boom, Cedric & Dhoedt, Bart. (2020). Learning Perception and Planning With Deep Active Inference, in ICASSP 2020, doi: 10.1109/ICASSP40776.2020.9054364.
- [CAT 20a] Çatal, Ozan & Wautier, Samuel & De Boom, Cedric & Verbelen, Tim & Dhoedt, Bart. (2020). Deep Active Inference for Autonomous Robot Navigation, in BAICS workshop at ICLR 2020.
- [CBR 18] C-BRIC, Center for Brain-Inspired Computing, https://engineering.purdue.edu/C-BRIC.
- [CEU 19] H Ceulemans, R Wuyts, W Verachtert, J Simm, A Arany, Y Moreau, C Herzeel. Secure Broker-Mediated Data Analysis and Prediction. US Patent App. 15/722,742, 2019.
- [CHI 18] Chinchali, Sandeep & Cidon, Eyal & Pergament, Evgenya & Chu, Tianshu & Katti, Sachin. (2018). Neural Networks Meet Physical Networks: Distributed Inference Between Edge Devices and the Cloud. 50-56. 10.1145/3286062.3286070.
- [COL 16] Colin, I., Bellet, A., Salmon, J., and Cl´emen¸con, S. (2016). Gossip dual averaging for decentralized optimization of pairwise functions. In ICML
- [DEA 12] Jeffrey Dean et.al. "Large scale distributed deep networks", In Advances in Neural Information Processing Systems (NIPS), pages 1223-1231, 2012.
- [DUC 12] Duchi, J. C., Agarwal, A., and Wainwright, M. J. (2012). Dual Averaging for Distributed Optimization: Convergence Analysis and Network Scaling. IEEE Transactions on Automatic Control, 57(3):592–606.
- [GOY 17] Goyal, Priya & Dollár, Piotr & Girshick, Ross & Noordhuis, Pieter & Wesolowski, Lukasz & Kyrola, Aapo & Tulloch, Andrew & Jia, Yangqing & He, Kaiming. (2017). Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour.
- [FRI 16] Friston, Karl & FitzGerald, Thomas & Rigoli, Francesco & Schwartenbeck, Philipp & Pezzulo, Giovanni. (2016). Active Inference: A Process Theory. Neural computation. 29. 1-49. 10.1162/NECO_a_00912.
- [GRA16] Graves, A., Wayne, G., Reynolds, M. et al. Hybrid computing using a neural network with dynamic external memory. Nature 538, 471–476 (2016). https://doi.org/10.1038/nature20101
- [GUP18] Gupta, Otkrist and Raskar, Ramesh, Distributed learning of deep neural network over multiple agents, Journal of Network and Computer Applications, Vol.116, pp.1–8, 2018.

- [HAN 16] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In International Conference on Learning Representations, 2016a.
- [HAU 20] Hauser, S., M. Mutlu, P. -A. Léziart, H. Khodr, A. Bernardino, and A. J. Ijspeert. “Roombots Extended: Challenges in the nACKext Generation of Self-Reconfigurable Modular Robots and Their Application in Adaptive and Assistive Furniture.” Robotics and Autonomous Systems 127 (May 1, 2020): 103467. https://doi.org/10.1016/j.robot.2020.103467.
- [HAW 17] Hawkins J, Ahmad S and Cui Y (2017) A Theory of How Columns in the Neocortex Enable Learning the Structure of the World. Front. Neural Circuits 11:81. doi: 10.3389/fncir.2017.00081.
- [HOR 14] M. Horowitz, Computing's energy problem (and what we can do about it), in: ISSCC, 2014
- [JIA 19] Jianyu Wang, Vinayak Tantia, Nicolas Ballas, and Michael Rabbat, “SlowMo: Improving communication efficient distributed SGD with slow momentum”, arXiv preprint arXiv:1910.00643, 2019.
- [JIA 19] Jianyu Wang and Gauri Joshi. Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms. preprint, August 2018.
- [KAL 18] D. Kalashnikov et al. “QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation”. (2018).
- [KAN 09] P. Kanerva,
- [KAR 20] Karunaratne, G., Le Gallo, M., Cherubini, G. *et al.* In-memory hyperdimensional computing. *Nat Electron* **3,** 327–337 (2020). https://doi.org/10.1038/s41928-020-0410-3
- [KOL 19] Koloskova, Anastasia & Stich, Sebastian & Jaggi, Martin, “Decentralized Stochastic Optimization and Gossip Algorithms with Compressed Communication”, ICML 2019
- [KOL 20] Koloskova, Anastasia & Lin, Tao & Stich, Sebastian & Jaggi, Martin, “Decentralized Deep Learning with Arbitrary Communication Compression”, ICLR 2020
- [KON 16] Jakub Konecny, H Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon, “Federated learning: Strategies for improving communication efficiency”, arXiv preprint arXiv:1610.05492, 2016.
- [KUM 13-19] Grasp Laborartory – Vijay Kumar Lab, University of Pennsylvania, https://youtu.be/YQIMGV5vtd4.
- [LEC 19] Y.LeCunn, Deep Learning Hardware: Past, Present, and Future in ISSCC 2019.
- [LEV 15] S. Levine*, C. Finn*, T. Darrell, P. Abbeel. “End-to-end training of deep visuomotor policies”. (2015)
- [LIA 17] Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., and Liu, J. (2017). Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent. In NIPS.
- [LIA 18] Lian, X., Zhang, W., Zhang, C., and Liu, J. (2018). Asynchronous Decentralized Parallel Stochastic Gradient Descent. In ICML.
- [LYN 20] Nancy Lynch Research Projects, https://www.csail.mit.edu/person/nancy-lynch.
- [MAA 97] W. Maass. Networks of spiking neurons: the third generation of neural network models. Neural Networks, 10:1659-1671, 1997.
- [MAO 17] H. Mao et al, “Exploring the Regularity of Sparse Structure in Convolutional Neural Networks,” https://arxiv.org/pdf/1705.08922.pdf.
- [MNI 13] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, et al. “Playing Atari with Deep Reinforcement Learning”. (2013).
- [MOBv1] MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H, arXiv:1704.04861, 2017.
- [MOBv2] MobileNetV2: Inverted Residuals and Linear Bottlenecks, Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC. arXiv preprint. arXiv:1801.04381, 2018.

- [NSF 19] NSF/Intel Partnership on Machine Learning for Wireless Networking Systems, NSF 19-951, https://www.nsf.gov/pubs/2019/nsf19591/nsf19591.htm.
- [OMI 17] Omidshafiei, Pazis, Amato, How and Vian - ICML 17.
- [RAS 18] Rashid et.al, "QMIX : Monotonic value function factorization for deep multi-agent RL", ICML 2018.
- [RAU 20] Adaptive Foraging in Dynamic Environments Using Scale-Free Interaction Networks. I Rausch, P Simoens, Y Khaluf - Frontiers in Robotics and AI, 2020.
- [RAZ 14] Razavian, Ali Sharif, H. Azizpour, J. Sullivan and S. Carlsson. "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition." 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops (2014): 512-519.
- [ROB 17] Roboschool, openai.org, https://openai.com/blog/roboschool/.
- [SAD 16] Fereshteh Sadeghi and Sergey Levine. "CAD2RL: Real single-image flight without a single real image." arXiv:1611.04201 (2016).
- [SAM 19] Samvelyan, Mikayel & Rashid, Tabish & Witt, Christian & Farquhar, Gregory & Nardelli, Nantas & Rudner, Tim & Hung, Chia-Man & Torr, Philip & Foerster, Jakob & Whiteson, Shimon. (2019). The StarCraft Multi-Agent Challenge, NIPS 2019.
- [SCH 15] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. "Trust Region Policy Optimization". (2015).
- [SCH 18] Schmuck, Patrik & Chli, Margarita. (2018). CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams. Journal of Field Robotics. 36. 10.1002/rob.21854.
- [SEB 17] Sébastien Racanière, Théophane Weber, David P. Reichert, Lars Buesing, Arthur Guez, Danilo Rezende, Adria Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, Razvan Pascanu, Peter Battaglia, Demis Hassabis, David Silver, and Daan Wierstra. 2017. Imagination-augmented agents for deep reinforcement learning. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17).
- [SIL16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, et al. "Mastering the game of Go with deep neural networks and tree search". Nature (2016).
- [SJO 10] Jesper Sjöström and Wulfram Gerstner, Scholarpedia, 5(2):1362, (2010).
- [SKY 20] Skydio 2, https://www.skydio.com.
- [SPE 19] Darpa Spectrum Challenge, https://www.darpa.mil/news-events/2019-10-24.
- [STE 15] P. Sterling, S. Luaghlin, Principles of Neural Design, MIT Press, 2015.
- [STE 19] Stephen James et al. "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks" CVPR 2019.
- [SUH 18] Suhenag et.al, "Value-Decomposition network (VDN) for cooperative multi-agent learning based on team reward", IAAMS 2018 (Autonomous agents and Multiagent Systems)
- [SUN 19] Sundar Pichai. Google's Sundar Pichai: Privacy Should Not Be a Luxury Good. New York Times, May 7, 2019.
- [SUR 17] Ananda Theertha Suresh, Felix X. Yu, Sanjiv Kumar, and H Brendan McMahan, "Distributed mean estimation with limited communication", In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 3329–3337. JMLR. org, 2017.
- [SUT 98] Sutton, R. S. and Barto, A. G. (1998). Reinforcement Learning: An Introduction. Bradford Books, MIT Press, Cambridge, MA, 2002 edition.
- [SWA 20] Swarm Intelligence, Wikipedia, https://en.wikipedia.org/wiki/Swarm_intelligence.
- [SZE 20] Vivienne Sze , Yu-Hsin Chen, Tien-Ju Yang, Joel S. Emer, Efficient Processing of Deep Neural Networks, Morgan Claypool Publishers, June 2020.
- [TAR 16] S. Taranovich. *Efficient Powering of a Robot Swarm | EDN*. Accessed: Aug. 2016. [Online]. Available: https://www.edn.com/design/power-management/4442493/Efficient-powering-of-a-robot-swarm.

- [TUR 20] A Turgut, I Boz, I Okay, E Ferrante, C Huepe, Interaction network effects on position- and velocity-based models of collective motion, Journal of the Royal Society interface, 2020.
- [VEP 18] Vepakomma, Praneeth & Gupta, Otkrist & Swedish, Tristan & Raskar, Ramesh. (2018). Split learning for health: Distributed deep learning without sharing raw patient data, NIPS 2018.
- [VER 20] Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, Jan S. Rellermeyer, *A Survey on Distributed Machine Learning*, ACM Computing Surveys, Vol. 53, No. 2, March 2020.
- [WAY 18] Wayne, G., Hung, C.-C., Amos, D., Mirza, M., Ahuja, A., Grabska-Barwinska, A., Rae, J., Mirowski, P., Leibo, J. Z., Santoro, A., et al. Unsupervised predictive memory in a goal-directed agent. arXiv preprint arXiv:1803.10760, 2018.
- [WEB 17] Weber, Theophane & Racanière, Sébastien & Reichert, David & Buesing, Lars & Guez, Arthur & Jimenez Rezende, Danilo & Badia, Adria & Vinyals, Oriol & Heess, Nicolas & Li, Yujia & Pascanu, Razvan & Battaglia, Peter & Silver, David & Wierstra, Daan. (2017). Imagination-Augmented Agents for Deep Reinforcement Learning.
- [WIT 20] Witt, Christian & Peng, Bei & Kamienny, Pierre-Alexandre & Torr, Philip & Böhmer, Wendelin & Whiteson, Shimon. (2020). Deep Multi-Agent Reinforcement Learning for Decentralized Continuous Cooperative Control.
- [WU 19] Wu, Zonghan & Pan, Shirui & Chen, Fengwen & Long, Guodong & Zhang, Chengqi & Yu, Philip. (2019). A Comprehensive Survey on Graph Neural Networks.
- [Xiao20] H Xiao, C Chen, D Yu, Two-level structure swarm formation system with self-organized topology network, Neurocomputing, vol 384, 2020.
- [YEV 19] Yevgen Chebotar et al. “Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience.” Arxiv: 1810.05687 (2019).
- [YOS 14] Yosinski, Jason, Clune, Jeff, Bengio, Yoshua and Lipson, Hod. "How transferable are features in deep neural networks?." Advances in Neural Information Processing Systems 27 (2014): 3320--3328.
- [ZAM 19] Zambrano D, Nusselder R, Scholte HS and Bohté SM (2019) Sparse Computation in Adaptive Spiking Neural Networks. Front. Neurosci. 12:987. doi: 10.3389/fnins.2018.00987.
- [ZHA 18] Y. Zhao at al., Federated Learning with Non-IID Data, arXiv:1806.00582 [cs.LG], June 2018.
- [ZHU 17] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. arXiv preprint arXiv:1710.01878, 2017.