Learning Connectivity-Maximizing Network Configurations

Daniel Mox, *Graduate Student Member, IEEE*, Vijay Kumar, *Fellow, IEEE*, and Alejandro Ribeiro, *Member, IEEE*

***Abstract*—In this letter we propose a data-driven approach to optimizing the algebraic connectivity of a team of robots. While a considerable amount of research has been devoted to this problem, we lack a method that scales in a manner suitable for online applications for more than a handful of agents. To that end, we propose a supervised learning approach with a convolutional neural network (CNN) that learns to place communication agents from an expert that uses an optimization-based strategy. We demonstrate the performance of our CNN on canonical line and ring topologies, 105k randomly generated test cases, and larger teams not seen during training. We also show how our system can be applied to dynamic robot teams through a Unity-based simulation. After training, our system produces connected configurations over an order of magnitude faster than the optimization-based scheme for teams of 10-20 agents.**

***Index Terms*—Networked Robots, Multi-Robot Systems, Deep Learning Methods**

I. Introduction

ENSURING a team of robots remains in contact with one another while accomplishing an objective has been the focus of a considerable amount of research in the robotics community. Indeed, most multi-robot systems operate on the underlying assumption that robots can freely exchange information in order to coordinate their actions. One common method for ensuring this condition is met considers the network graph induced by the spatial configuration of the robots coupled with an underlying communication model. This problem formulation emits a graph theoretic approach focused on maximizing or preserving the connectivity of the state dependent communication graph [1]–[13] which can be solved in both a centralized [1]–[5] and decentralized manner [1], [6]–[13].

Communication graph connectivity is inherently a heuristic for network performance. This choice is based on the intuition that the more connected a network is the easier information flows through it. However, in practice, network performance depends on the underlying routing protocol and the extent to which maximizing graph connectivity directly translates to improved performance is unclear. More recent work has sought to bridge this gap by integrating aspects of wireless systems into the planning problem formulation. In [14] the authors propose a mobile router solution that optimizes end-to-end bit error rate between a pair of nodes. Other work goes even further by combining the node positioning and packet routing problems [15]–[18]. While these methods more accurately model the underlying wireless network they result in computationally expensive, coupled optimization problems that can be challenging to solve, even approximately.

In this work, we consider the problem of *mobile wireless infrastructure on demand* introduced in [15] wherein mobile relay nodes must be strategically positioned to facilitate communication between task-oriented robots whose actions may take them out of direct transmission range with one another. While many of the aforementioned approaches offer solutions to this problem, we lack a method that scales as the number of agents in the network grows. Indeed, centralized approaches to maximizing algebraic connectivity require solving costly optimization problems that become prohibitively slow for large teams. Decentralized variants rely on distributed optimization which is iterative in nature and requires significant time to converge. Likewise, the more sophisticated methods face similar scaling challenges just to find feasible solutions.

To address this problem, we propose a data-driven approach to maximizing connectivity that demonstrates attractive scaling characteristics compared with existing methods. In particular, we show how supervised learning coupled with convolutional neural networks (CNNs) can be used to learn how to provide mobile wireless infrastructure on demand, ensuring a multi-robot team forms a connected network. CNNs have been used in many robotics tasks such as perception, control, and planning [19]–[23]. Similarly, images have emerged as a natural way to encode the spatial configuration of agents relative to each other and their environment [23]–[25]. A key insight motivating the use of learning in our case is that while training can be a long process, inference is inexpensive. In addition, the existing connectivity maximization solutions that aren't suitable for real-time applications can readily be

Manuscript received September 9, 2021; accepted January 3, 2022. Date of publication January 27, 2022; date of current version March 22, 2022. This letter was recommended for publication by Associate Editor Guillaume Adrien Sartoretti and Editor M. Ani Hsieh upon evaluation of the reviewers comments. The work of Daniel Mox was supported by the National Science Foundation Graduate Research Fellowship under Grant DGE-1845298. This work was supported in part by ARL under Grant DCIST CRA W911NF-17-2-0181, in part by NSF under Grant CNS-1521617, in part by ARO under Grant W911NF-13-1-0350, in part by ONR under Grants N00014-20-1-2822 and N00014-20-S-B001, in part by Qualcomm Research, and in part by the NVIDIA Corporation through the donation of the DGX Station used for this research. (Corresponding author: Daniel Mox.)

Daniel Mox and Vijay Kumar are with the General Robotics, Automation, Sensing and Perception (GRASP) Laboratory at the University of Pennsylvania, Philadelphia, PA 19104 USA (email: mox@seas.upenn.edu, kumar@seas.upenn.edu)

Alejandro Ribeiro is with the Electrical and Systems Engineering Department at the University of Pennsylvania, Philadelphia, PA 19104 USA (email: aribeiro@seas.upenn.edu)

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2022.3146524, provided by the authors.

Digital Object Identifier 10.1109/LRA.2022.3146524

used to generate expert training samples offline for supervised learning.

In this letter we present a novel approach to mobile infrastructure on demand utilizing supervised learning with CNNs. We show that our approach achieves comparable performance to the expert it learns from in a fraction of the time, generalizes to larger teams not seen during training, and operates effectively in dynamic teams of robots. The rest of this letter is organized as follows: Section II provides an overview of the problem and our approach, covering how we model communication between agents in II-A, how we generate training samples in II-B, and what CNN architecture we use in II-C, Section III contains results and analysis, and finally in Section IV we provide concluding remarks.

II. Methodology

Consider the scenario where a team of mobile robots seeks to accomplish a task requiring communication. Instead of creating and sustaining a wireless network in addition to completing their objective, these robots, referred to as task agents, assume the availability of communication infrastructure which is provided by a different set of robots, referred to as communication or relay agents. This team of relay agents positions itself in the environment and facilitates communication so that the task agents can go about their objective without considering the impact their actions have on their ability to exchange vital information. Our focus is on how the communication agents should be positioned.

More formally, given a set of N task agents with configuration denoted $\boldsymbol{x}_T \in \mathbb{R}^{N\times 2}$, we seek to position a set of M communication agents $\boldsymbol{x}_C \in \mathbb{R}^{M\times 2}$ so that the entire group $\boldsymbol{x} = [\boldsymbol{x}_T; \boldsymbol{x}_C]$ maximizes the algebraic connectivity of the underlying communication graph. While more sophisticated placement algorithms exist [15]–[17], they neither provide locally optimal solutions nor are they computationally suitable even for offline data generation. We target algebraic connectivity maximization since locally optimal configurations can be readily found using an optimization approach [5]. Unfortunately, such optimization methods come at a high computational cost as the size of the team grows and are unsuitable for online applications (see Fig. 9). On the other hand, inference on a trained neural network is inexpensive. Thus, as a solution to this scaling problem, we propose a supervised learning approach wherein a CNN learns to position mobile communication nodes according to solutions obtained from an optimization scheme similar to [5].

A. Communication Model

In order to reason about communication in a group of robots we must be able to predict the ability of pairs of agents to exchange information. In this work we use a function of the distance between two agents borrowed from probabilistic channel approaches that balances accuracy with model complexity by capturing the dominant fading characteristics of wireless channels [26]. Concretely, agents i and j located at $x_i, x_j \in \mathbb{R}^2$, respectively, can communicate according to the following wireless channel normalized rate function:

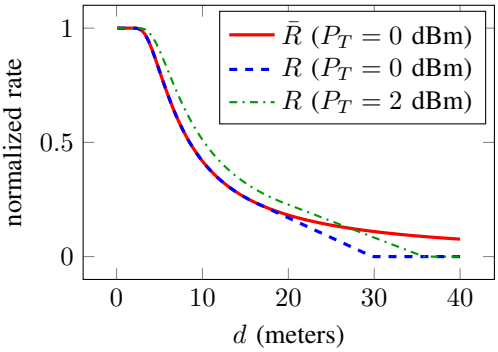


Fig. 1: The non-vanishing wireless channel model in Eq. (1) and the piecewise smooth model of Eq. (3) with $n = 2.52$, $P_{N_0} = -70$ dBm, and $K = 5.01 \times 10^{-6}$ (-53 dB).

$$\bar{R}(d) = \text{erf}\left(\sqrt{\frac{P_R}{P_{N_0}} d^{-n}}\right) \tag{1}$$

where P_R is the received signal power computed from:

$$P_R = P_T K d^{-n}. \tag{2}$$

In the above equations, P_T is the transmit power (mW), $d = ||x_i - x_j||$, K is a constant specific to the communication hardware used, P_{N_0} is the ambient noise at the receiver (mW), and n is the signal decay exponent [26]. Note that while Eqs. (1), (2) assume absolute power (mW), it is often convenient to refer to various quantities in relative units of decibel-milliwatts (dBm). It is also useful to express Eq. (1) in terms of the robot positions as $\bar{R}(x_i, x_j) = \bar{R}(||x_i - x_j||)$. Note that Eq. (1) approaches but does not reach zero for increasing d, implying that two agents can communicate at arbitrarily large distances, albeit at very low rates (see Fig. 1). This behavior does not align with reality as a wireless channel will certainly cease to function at large distances as the signal becomes indistinguishable from noise. Thus, we wrap Eq. (1) in a piecewise smooth function that eventually drives the channel rate to zero:

$$R(d) = \begin{cases} \bar{R}(d) & d \le d_t \\ \frac{\partial \bar{R}}{\partial d}\Big|_{d_t} (d - d_t) + R(d_t) & d_t < d \le d_c \\ 0 & d_c < d \end{cases} \tag{3}$$

where the transition distance d_t is found from a chosen, fixed cutoff rate and d_c is the cutoff distance at the zero rate crossing of the linear function. A plot of Eq. (3) can be seen in Fig. 1. In this work we use a default set of parameters for Eq. (3) (see Fig. 1) with a fixed cutoff rate but allow the transmit power to be varied from its default value of 0 dBm when necessary, which effectively extends the max range of the channel, d_c.

B. Dataset Generation

Our CNN learns to position communication agents from an expert. In this case, the expert is an optimization approach to

maximizing the connectivity of a communication graph based on [5]:

$$\begin{aligned}
\underset{\boldsymbol{x}_C^{k+1}}{\text{maximize}} \quad & \gamma \\
\text{subject to} \quad & P^T L P \succeq I\gamma && \text{(4a)} \\
& L = \text{diag}(A\mathbf{1}) - A \\
& [A]_{ij} = R(x_i^k, x_j^k) + \nabla_{x_i^{k+1}} R^T (x_i^{k+1} - x_i^k) \\
& \qquad + \nabla_{x_j^{k+1}} R^T (x_j^{k+1} - x_j^k) && \text{(4b)} \\
& |\boldsymbol{x}_C^{k+1} - \boldsymbol{x}_C^k|_1 \leq \Delta && \text{(4c)}
\end{aligned}$$

where L and A are the state dependent graph Laplacian and adjacency matrix, respectively, and P is an orthogonal basis spanning $\mathbf{1}^\perp$. The tunable parameter Δ along with the constraint in Eq. (4c) ensures that the optimization variables $\boldsymbol{x}_C^{k+1}$ remain in a region around the current configuration $\boldsymbol{x}_C^k$ where the linearized channel model in Eq. (4b) is valid. Eq. (4a) forces the algebraic connectivity to be greater than or equal to γ. Note that the optimization variables are the communication agent positions $\boldsymbol{x}_C$ as they are the only agents that can be controlled. Problem (4) adjusts $\boldsymbol{x}_C$ in order to maximize the algebraic connectivity of the team, γ, and can be solved in an iterative fashion using an available SDP solver, converging to a local optimum given an initial feasible configuration. For more detail see [5].

We use the optimization expert in Eq. (4) to generate training samples consisting of pairs of images that capture the given task team and target communication team configurations. In order to ensure that spatial relationships between agents are consistent throughout the dataset, the metric distance each image pixel represents is fixed at 1.25 meters per pixel. Task team configurations $\boldsymbol{x}_T$ are sampled from a uniform distribution (Fig. 2a) and marked in the image using a Gaussian kernel to avoid issues with sparsity (Fig. 2b). Next, a feasible initial network team configuration must be found to seed the optimization. To do this, a minimum spanning tree (MST) of the task team configuration is computed and communication agents are used to break up graph edges longer than d_c (Fig. 2c). The corresponding communication graph can be readily found from the augmented MST by applying Eq. (3) to each pair of agents and keeping edges where the communication rate is greater than zero. The resulting graph is guaranteed to be connected since no edge in the augmented MST is greater than d_c.

With this feasible initial solution, Problem (4) can be iteratively solved to adjust the network node locations into a locally optimal configuration (Fig. 2d). The resulting communication team configuration is marked in a separate image using the same Gaussian kernel (Fig. 2e). Figs. 2b and 2e constitute the input and target output images of the CNN, respectively. This entire data generation sequence is illustrated for one sample in Fig. 2. The richness of the training dataset is critical to the learned model's performance. In problems where training samples are scarce it is common to perform extensive data augmentation to increase the size of the dataset and ensure the samples capture sufficient angular diversity to account for the rotational sensitivity of CNNs. In our case, training samples are not only abundant but also randomly generated in a way that the dataset naturally encodes sufficient angular diversity for the CNN to learn the rotational symmetries of our problem. In light of this, we do not perform any explicit data augmentation.

C. Learning Architecture

When it comes to learning connectivity maximizing configurations, choosing an appropriate model architecture is of paramount importance. Our choice of a CNN architecture might seem surprising. Why not use a learning model that accepts the position of task agents and produces communication agent positions? In fact, a CNN with an image as an input is a more appropriate representation because it leverages the symmetries of the connectivity problem. We can think of the communication team as filling in the gaps between the members of the task team, a goal that depends on the relative positions of agents and is invariant to their absolute positions. CNNs possess this very property making them a suitable choice for this problem. Furthermore, images are a natural way to represent spatial information *without incurring penalties with scaling up to large numbers of agents*. Provided the image is sized to cover an adequate metric area, configurations with many agents can be readily represented alongside those with few. Once in image form, all inputs to the CNN are processed equally meaning there is no performance difference between a team consisting of 4 agents or 20. This is not the case with the optimization in Eq. 4, which becomes prohibitively slow for large teams (see Fig. 9).

Considering the input and output of our network is an image, we employ an autoencoder (AE) like architecture comprised entirely of convolutional layers which we refer to as ConvAE. While typical AEs use fully connected layers or a probability distribution at the information bottleneck between the encoder and decoder, we found that using convolutional layers instead resulted in better generalization performance, especially as the size of the task team increased. As a side effect, ConvAE can operate on arbitrarily sized input images provided they pass cleanly through the network (for our model compatible image resolutions are given by $(N+4)\cdot 2^6$ for $N \in \mathbb{Z} \geq 0$). In other words, we never run out of image space to represent teams with many agents spread over large areas. Interestingly, for the smallest image resolution that the CNN can process, the convolutions at the bottleneck are effectively fully connected neurons. The encoder transforms the 256x256 input image into a volume with dimension 4x4xF, where F is the number of filters. Immediately after, the first convolutional transpose layer of the decoder takes this 4x4xF volume and processes it with a 4x4 kernel. Notice how the entire transformed input volume fits cleanly into the 4x4 convolutional transpose filter so that every extracted feature factors into each filters output, exactly like a fully connected layer. The utility of autoencoders lies in the information bottleneck, which forces the network to learn a lower dimensional representation of the input data in order to faithfully replicate the output. In our case, the information bottleneck at the deepest convolutional layer causes the network to glean salient geometric features

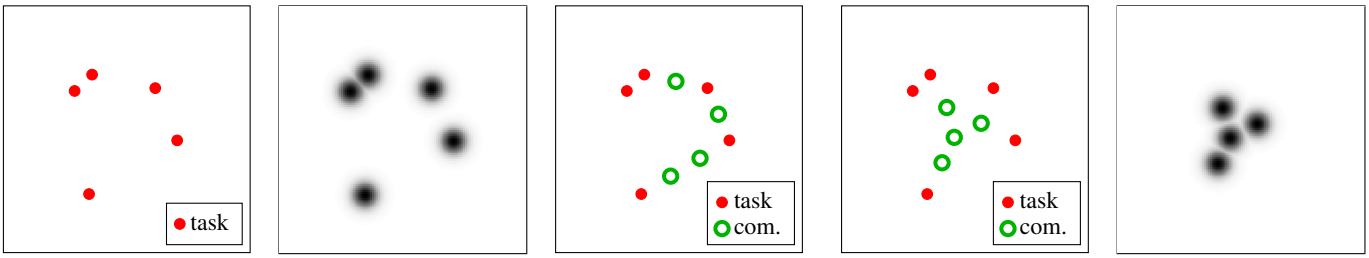


Fig. 2: (a) A randomly generated task team configuration; (b) The corresponding task team image; (c) The augmented MST team configuration derived from (a); (d) The locally optimal team configuration found by solving (4); (e) The network team image corresponding to (d). Each image has been cropped from 256x256 to 128x128 pixels which represents a square area with a side length of 160 meters.

from the input configuration image that inform the placement of network agents in the output image. The details of our architecture are shown in Table I.

TABLE I: CNN architecture: network proceeds sequentially from top (input) to bottom (output). Each Conv2D / ConvTranspose2D layer is padded, uses a stride of 2, and contains 128 input and output channels.

model	layer type	count	kernel	activation
encoder	Conv2D	2	8x8	LeakyReLU
encoder	Conv2D	4	4x4	LeakyReLU
decoder	ConvTranspose2D	4	4x4	ReLU
decoder	ConvTranspose2D	2	8x8	ReLU

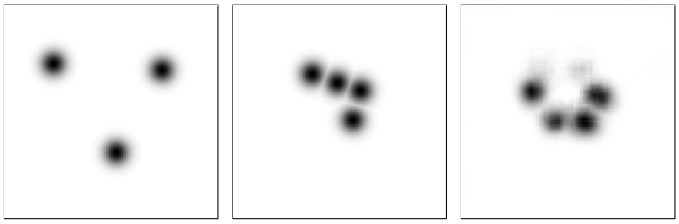


Fig. 3: results for one test sample showing (a) the input image to the CNN; (b) the expected output; and (c) the CNN output. Each image has been cropped from 256x256 to 128x128 for clarity.

III. Results

Our ConvAE model was trained for 14 epochs on a dataset comprised of 595k images generated using the process described in Section II-B, with task teams comprised of 2-6 agents. Adam was used as the optimizer with a learning rate of 10^{-4}, a batch size of 4, and mean-squared-error as the loss function [1]. The output of the CNN for one test sample can be seen in Fig. 3. It is immediately apparent that the CNN output in Fig. 3c differs from that of the optimization in Fig. 3b; what may not be so obvious is that the CNN yields a valid configuration that connects the task agents shown in Fig. 3a. This outcome is not unexpected. The optimization produces *locally optimal* solutions and for a given task team configuration there may be many distinct local optimums. Clearly the performance of the CNN must be judged by its ability to produce connected configurations and not the relative closeness of the output image to the reference image in a mean-squared-error sense.

In order to compute algebraic connectivity we must be able to extract relay node positions in $\mathbb{R}^2$ from the output images of the CNN. While the peaks in Fig. 3c are distinct and easy to pick out, the CNN images are not always so clear (see Fig. 5c). To overcome this issue we take a coverage perspective, interpreting the output image as a distribution the communication agents must assume. First, we determine the number of agents to deploy by utilizing an adaptive thresholding scheme to pick out peaks in the intensity image. Then, we employ Lloyd's algorithm for the specified number of agents to find a configuration that achieves locally optimal coverage of the intensity distribution [27]. The CNN may output redundant agents, especially when the task team configuration is symmetric (see Figs. 5d, 5f). We prune these extra agents by extracting a minimal connected sub-graph. Finally, the algebraic connectivity can readily be found using the known positions of the task agents used to generate the input image.

Algebraic connectivity alone does not tell the whole story. Since our channel model in Eq. (3) is truncated, there is a hard cutoff at d_c beyond which it is assumed no communication is possible. Any configuration relying on an edge greater than d_c has an algebraic connectivity of zero (i.e. is disconnected). However, for performance evaluation we want to know how close a configuration is to being connected (i.e. distinguish between one configuration relying on an edge one centimeter beyond d_c and another tied together with an edge many meters greater than d_c). Thus, as a more informative criterion, we consider the transmit power P_T required to achieve a connected configuration. For the optimization this value is guaranteed to be 0 dBm. For the CNN we extract the configuration as described above and check connectivity with 0 dBm, increasing it if necessary until connectivity is established. In this way, the performance of our system is

[1] code and multimedia available at: www.danmox.com/projects/convae.html

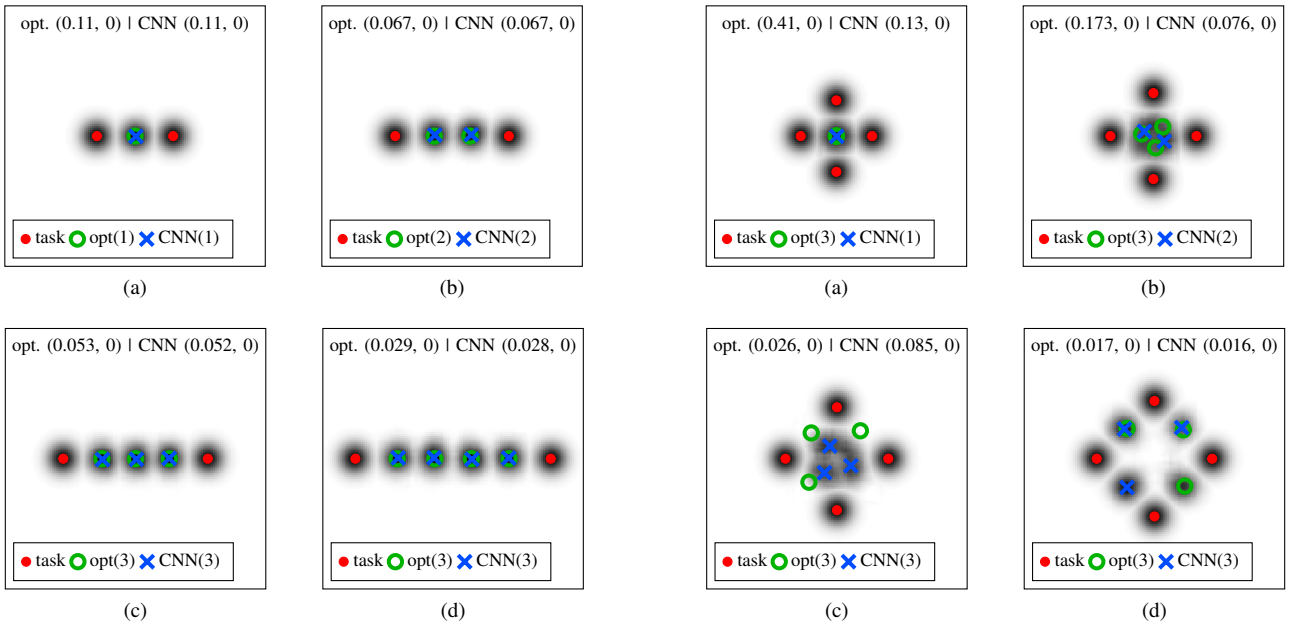


Fig. 4: Line scenario results progressing from (a) to (d). Dimensions are in meters. Relay agents are shown as x's for the CNN and o's for the optimization (refered to as opt.), with the number deployed by each method shown in the legend, and the background image combines the input and output of the CNN. The title format is: method (algebraic connectivity, transmit power in dBm). The images have been cropped from 256x256 to 128x128 for clarity, covering a 160x160m area.

Fig. 5: Circle scenario results progressing from (a) to (f). Dimensions are in meters. Relay agents are shown as x's for the CNN and o's for the optimization (refered to as opt.), with the number deployed by each method shown in the legend, and the background image combines the input and output of the CNN. The title format is: method (algebraic connectivity, transmit power in dBm). The images have been cropped from 256x256 to 128x128 for clarity, covering a 160x160m area.

quantified by the amount of power required to connect the network as compared to the optimization.

The following Sections III-A and III-B provide qualitative results of our system on canonical line and ring topologies. Afterwards, we present ConvAE's performance over an extensive test dataset in Section III-C, show its ability to generalize to larger teams not seen during training in Section III-D, detail how our approach scales in Section III-E, and finally demonstrate how our system can be deployed in real, dynamic teams of robots in Section III-F.

A. Line Test

Perhaps the simplest test of the CNN's ability to produce connected configurations is a line test. Two agents starting relatively close together progressively move away from one another, necessitating the formation of a chain of communication relay nodes in between. Snapshots of the CNN and optimization results can seen in Fig. 4.

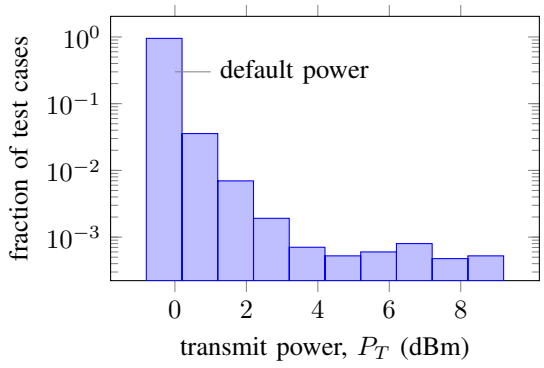
Across the snapshots, the configurations and corresponding connectivity values produced by the CNN and the optimization are very similar. This is not surprising as the CNN was trained on 170k images of 2 agent task teams of varying density and orientation, providing ample opportunity for the model to learn line topologies. In all cases, the CNN was able to produce connected configurations with the same number of agents as the optimization without needing to vary transmit power from its default value of 0 dBm.

B. Circle Test

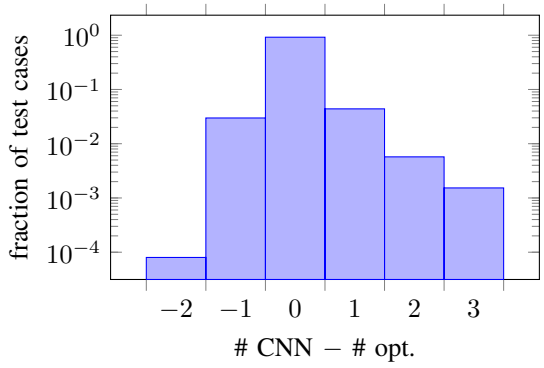
A harder test involves a group of task agents distributed on the perimeter of an expanding circle. The job of the optimization and CNN is to effectively deploy communication agents so that the task agents remain connected. Results for this circle test can be seen in Fig. 5.

Our CNN-based approach also performs well in the circle test when compared with the optimization. There are a few interesting results to note. First, because there is no explicit labeling in the images, the CNN only learns to paint one blob in cases where agent positions overlap. This can be seen in Fig. 5a, 5e where the optimization placed multiple overlapping agents while the CNN used fewer to connect the network.

Another significant takeaway from the circle test is that our CNN can outperform the optimization that it learned from, as seen in Fig. 5c. Since algebraic connectivity decreases when links between agents are severed, the optimization will



(a)



(b)

Fig. 6: Histograms of a) transmit power used by the CNN to achieve connectivity and b) the difference between the number of agents deployed by the CNN and optimization.

only ever maintain or increase the edges in the network and is thus confined to a local optimum that depends on the topology of the initial configuration. However, there are likely many different locally optimal configurations represented in the training data as randomly sampled task agent teams that are similar in shape may have different solutions from the optimization (see Figs. 5c and 5d where it yielded considerably different local optimums despite slight changes in the input configuration). Additionally, subsets of a given task agent configuration are likely to appear similar to other training samples. As a result, the CNN may output a configuration with a different topology than the optimization (see also Fig. 5c).

C. Dataset Statistics

The line and circle tests provide a sense of how the CNN functions in two canonical scenarios. In this section we provide a more rigorous evaluation by computing the performance of ConvAE over a test dataset of 105k images generated in the same way as the training dataset with task teams of 2-6 agents. The results can be seen in Fig. 6.

As mention in the beginning of Section III, the CNN increases transmit power only when the configuration is not connected at the default value of 0 dBm. As can be seen in Fig. 6a, the CNN produces connected configurations at 0 dBm (indicated by the leftmost bar) a vast majority of the time and in most other cases requires very little additional power. On average our CNN required 0.05 dBm of transmit power with a variance of $+0.438$ dBm to produce connected configurations.

Fig. 6b shows a histogram comparing the number of agents deployed by the CNN and optimization. In cases where the CNN required more transmit power it was often due to deploying fewer agents. Thus to make this comparison fair, we restrict ourselves to test samples where the CNN used the default transmit power (100k out of the 105k cases) and show the number of agents deployed by the optimization subtracted from the number used by the CNN. Negative values indicate instances where the CNN used fewer agents to connect the network and positive values those where the CNN deployed more. In most cases the number of agents deployed by each method is within ± 1 of each other showing that our approach achieves connectivity with roughly the same number of communication nodes as the optimization: 0.031 ± 0.344 more agents on average. In some cases, the CNN deploys more agents than required, typically for configurations near a topology change due to ambiguities in the training data. As a result, there is a slight asymmetry in the distribution in Fig. 6b; however, we note that the y-axis is a log scale and these instances represent a very small fraction of the 105k test cases.

D. Generalization

Our CNN-based approach performs well on test samples similar to ones seen during training. However, in practice it can be cumbersome to generate new data and retrain the network for every team size the CNN might encounter. Furthermore, even generating meaningful training data offline with the optimization becomes increasingly difficult as the number of agents increases. Given the symmetries of the problem, we expect our CNN to have learned something about connecting teams of 2-6 agents that would apply to larger teams.

Indeed, we find this to be the case. We applied our ConvAE network trained exclusively on 2-6 task agent teams to a test dataset with much larger teams of 8 and 12 task agents with 3,000 and 1,500 samples, respectively. Fig. 7 shows that our model generalizes well to these out-of-distribution test cases. Representative results are show in Fig. 8. We note that this is not a trivial result. To achieve good performance with our method on large teams one need only train ConvAE on a comprehensive dataset of smaller configurations and fine tune on a much smaller set of larger team samples.

E. Scalability

A main motivation for our work is scalability. While Eq. (4) offers an elegant solution to the connectivity problem it becomes prohibitively slow as the size of the team grows and thus is unsuited for real-time deployment. On the other hand, our approach built on CNN inference is exceptionally scalable. Fig. 9 shows a comparison of computation time between the optimization and CNN. For the CNN we measure the time it takes to convert a task team configuration to a kernelized image (like Fig. 2b), perform inference, and extract the resulting communication team configuration. For the optimization we measure the time it takes to either converge to a local optimum

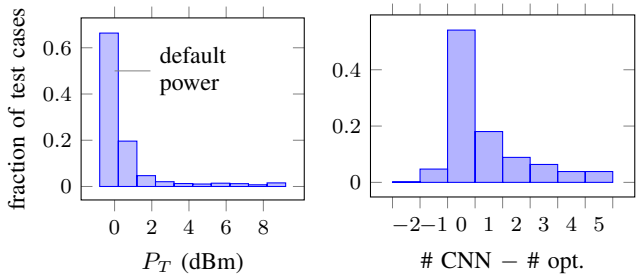


Fig. 7: Histograms of a) transmit power used by the CNN to achieve connectivity and b) the difference between the number of agents deployed by the CNN and optimization.

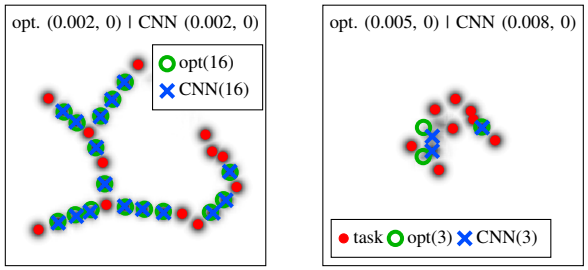


Fig. 8: Generalization examples for 8 and 12 agent teams. See Figs. 4, 5 for complete description. Images are 256x256 covering an area 320 meters on a side.

or reach 20 iterations. Note that it often takes more than 20 iterations for the optimization to converge; however, to account for often brittle convergence criteria we impose an aggressive upper limit on the maximum number of optimization iterations for the purpose of comparison. The advantage of our learning approach is clear: while the optimization quickly climbs to 10s of seconds the CNN increases from 520ms to only 800ms, with the increase in time coming entirely from unoptimized post processing steps. For a team of 20 agents the CNN is nearly 40 times faster!

F. Dynamic Scenario

While we have shown the performance of our system across a variety of static test cases, our goal remains to provide *mobile wireless infrastructure on demand* to teams of task agents collaborating to accomplish an objective. To that end, we have also deployed our system in a dynamic scenario in a Unity-based robotics simulator. To adapt our system for online use, we wrap our CNN in a high level controller that takes in the current state of the task agents, marks their positions in an image with a gaussian kernel, and passes it to the CNN for inference. From the image produced by the CNN, the high level controller extracts the target network team configuration using the procedure outlined in Section III and sends waypoint commands to the communication agents accordingly.

Our approach is inherently centralized and relies on aggregating state information about each agent in the team at a

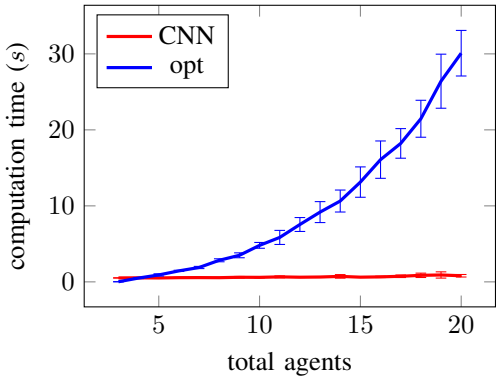


Fig. 9: A comparison of the computation time required to find relay node positions using our CNN approach compared with the optimization-based approach in Eq. (4) (opt).

planning node and disseminating control commands back to the communication agents. This network overhead amounts to transmitting a few floats at each planning step, which need not be run at a high rate. Thus, our method could operate using the network that it provides as a backbone or utilizing a low-bandwidth, long-range control channel. Note also that once deployed our CNN-based planner requires little computation and could easily be run on-board one of the communication agents, alleviating the need for a fixed ground station.

In the dynamic scenario, five task agents patrol a large urban area covering an space approximately 500 meters on a side. These patrolling agents require communication in order to secure their perimeter and thus relay agents are deployed to form a connected network. For this test, the transmit power of the agents was increased to 21 dBm ($d_c \approx 200$ meters) in order to scale their communication range relative to the size of the space. While the CNN was trained on configurations with an underlying transmit power of 0 dBm, it can be used directly in this scenario without retraining simply by appropriately scaling the input and output configurations of the CNN. A snapshot of the paths taken by the task and communication agents in this scenario can be seen in Fig. 10. During the patrol our CNN-based controller produced solutions to the connectivity problem at a rate of 2 Hz, keeping the patrolling task agents connected the entire time. Contrast that with the optimization approach that takes on the order of ~7 seconds to come up with a single connected configuration (see Fig. 9).

One might wonder about the continuity of the target communication team configurations produced by the CNN between iterations. While each image is processed independently, the rate at which the control loop is closed means that the CNN processes relatively small changes in the input image at each time step. Since CNNs are robust to perturbations the target communication team configurations do not change dramatically between iterations, as can be seen in Fig. 10. We do note that over time significant changes in the task agent configuration can result in the network topology changing. However, this challenge is faced by every connectivity based method and can be mitigated in practice by planning ahead.

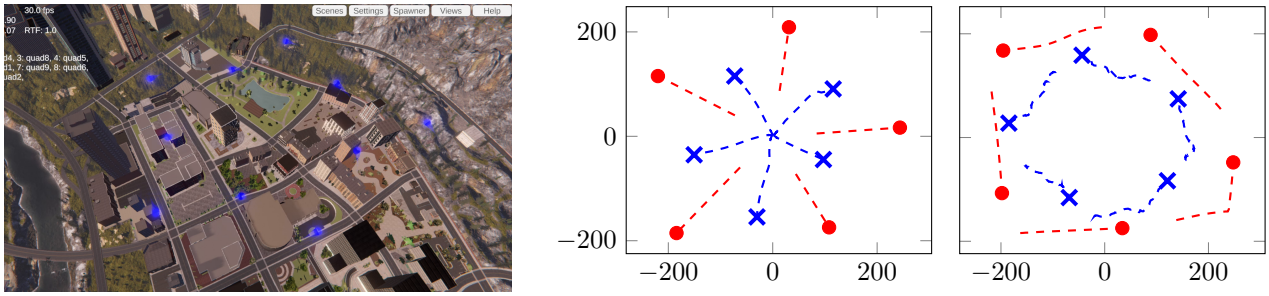


Fig. 10: A screenshot of the Unity-based robotics simulator showing the location of robots as blue blobs (left) and two snapshots of the dynamic patrol showing the positions of the task agents as red dots and the communication agents following our CNN-based planner as blue x's at the start (center) and middle (right) of the patrol. Dimensions are in meters.

IV. Conclusion

In this letter we have proposed a data-driven approach to maximizing algebraic connectivity by employing a convolutional autoencoder that learns how to provide *mobile wireless infrastructure on demand* in robot reams requiring communication. While optimization-based methods become slow as the number of agents increases, our CNN-based method scales exceptionally well, running over an order of magnitude faster for large teams. Our system achieves all this while consuming nearly the same amount of transmit power and using an almost equal number of communication agents on average. A natural question is if there is a better network architecture for this specific application that can yield better performance, especially in dynamic scenarios. This is an avenue for future research.

References

[1] Michael M Zavlanos, Magnus B Egerstedt, and George J Pappas. Graph-theoretic connectivity control of mobile robot networks. *Proceedings of the IEEE*, 99(9):1525–1540, 2011.

[2] Michael M Zavlanos and George J Pappas. Controlling connectivity of dynamic graphs. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 6388–6393. IEEE, 2005.

[3] Michael M Zavlanos and George J Pappas. Potential fields for maintaining connectivity of mobile networks. *IEEE Transactions on robotics*, 23(4):812–816, 2007.

[4] Ethan Stump, Ali Jadbabaie, and Vijay Kumar. Connectivity management in mobile robot teams. In *2008 IEEE international conference on robotics and automation*, pages 1525–1530. IEEE, 2008.

[5] Yoonsoo Kim and Mehran Mesbahi. On maximizing the second smallest eigenvalue of a state-dependent graph laplacian. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 99–103. IEEE, 2005.

[6] Maria Carmela De Gennaro and Ali Jadbabaie. Decentralized control of connectivity for multi-agent systems. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 3628–3633. IEEE, 2006.

[7] Meng Ji and Magnus Egerstedt. Distributed coordination control of multiagent systems while preserving connectedness. *IEEE Transactions on Robotics*, 23(4):693–703, 2007.

[8] Michael M Zavlanos and George J Pappas. Distributed connectivity control of mobile networks. *IEEE Transactions on Robotics*, 24(6):1416–1428, 2008.

[9] Giuseppe Notarstefano, Ketan Savla, Francesco Bullo, and Ali Jadbabaie. Maintaining limited-range connectivity among second-order agents. In *2006 American control conference*, pages 6–pp. IEEE, 2006.

[10] Demetri P Spanos and Richard M Murray. Robust connectivity of networked vehicles. In *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*, volume 3, pages 2893–2898. IEEE, 2004.

[11] Peng Yang, Randy A Freeman, Geoffrey J Gordon, Kevin M Lynch, Siddhartha S Srinivasa, and Rahul Sukthankar. Decentralized estimation and control of graph connectivity for mobile sensor networks. *Automatica*, 46(2):390–396, 2010.

[12] Meng Ji and Magnus Egerstedt. Distributed formation control while preserving connectedness. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 5962–5967. IEEE, 2006.

[13] Dimos V Dimarogonas and Karl H Johansson. Decentralized connectivity maintenance in mobile networks with bounded inputs. In *2008 IEEE International Conference on Robotics and Automation*, pages 1507–1512. IEEE, 2008.

[14] Yuan Yan and Yasamin Mostofi. Robotic router formation in realistic communication environments. *IEEE Transactions on Robotics*, 28(4):810–827, 2012.

[15] Daniel Mox, Miguel Calvo-Fullana, Mikhail Gerasimenko, Jonathan Fink, Vijay Kumar, and Alejandro Ribeiro. Mobile wireless network infrastructure on demand. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7726–7732. IEEE, 2020.

[16] James Stephan, Jonathan Fink, Vijay Kumar, and Alejandro Ribeiro. Concurrent control of mobility and communication in multirobot systems. *IEEE Transactions on Robotics*, 33(5):1248–1254, 2017.

[17] Jonathan Fink, Alejandro Ribeiro, and Vijay Kumar. Robust control of mobility and communications in autonomous robot teams. *IEEE Access*, 1:290–309, 2013.

[18] Michael M Zavlanos, Alejandro Ribeiro, and George J Pappas. Network integrity in mobile robotic networks. *IEEE Transactions on Automatic Control*, 58(1):3–18, 2012.

[19] Somil Bansal, Varun Tolani, Saurabh Gupta, Jitendra Malik, and Claire Tomlin. Combining optimal control and learning for visual navigation in novel environments. In *Conference on Robot Learning*, pages 420–429. PMLR, 2020.

[20] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[21] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[22] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.

[23] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2154–2162, 2016.

[24] Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, TK Satish Kumar, Sven Koenig, and Howie Choset. Primal: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters*, 4(3):2378–2385, 2019.

[25] Kamal K Ndousse, Douglas Eck, Sergey Levine, and Natasha Jaques. Emergent social learning via multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 7991–8004. PMLR, 2021.

[26] Jonathan Fink. *Communication for teams of networked robots*. PhD thesis, University of Pennsylvania, 2011.

[27] Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on robotics and Automation*, 20(2):243–255, 2004.