

498 **A Dataset**

499 We generate 128,000 images as agents’ observations using python’s matplotlib library Hunter [2007]
500 for training, and 12,800 images for validation and testing. Each observation is a 32 by 320 image that
501 displays two shapes of different colors and shapes. As shown in Figure 6, we modify the image for
502 the color-only and shape-only agents according to the given instruction. For example, in the image,
503 the instruction is ‘Find a green circle’. In this case, we unified the shape to ‘circle’ for the color-only
504 agent. Similarly, we unified the color to ‘green’ for the shape-only agent. Additionally, we diversity
505 the data by endowing different sizes, orientations, locations, and hues of the objects.

506 For testing the generalizability beyond the training experiences, we give additional degrees of freedom
507 to the objects, so that the objects shown are not seen during training. The detailed specifications can
508 be reference from the code submitted in the supplementary material.



(a) Original training observation



(b) Training observation for shape-only agent



(c) Decoded training observation for shape-only agent



(d) Training observation for color-only agent



(e) Decoded training observation for color-only agent



(f) Original test observation



(g) Test observation for shape-only agent



(h) Decoded test observation for shape-only agent



(i) Test observation for color-only agent



(j) Decoded test observation for color-only agent

Figure 6: Example observations.

509 **B Group communication training algorithm**

510 Algorithm 1 details the group communication training algorithm for emergent shared multi-agent communication.

Algorithm 1: Emergent group communication

Input: Encoded vectors z from the image observations o

```

1      Instruction vectors  $w$ 
2      Batch size  $B$ 
3      Number of agents in the group  $N$ 
4      Training iteration  $I$ 
5      Communication links  $\mathcal{E}$ 

6 for  $iter\ i = 1, \dots, I$  do
7   | sample color-only agent  $c$  and shape-only agent  $s$  from  $\mathcal{E}$ ;
8   | sample the listener agent from  $\{c, s\}$ ;
9   | sample the training batch of size  $B$ ;
10  | get  $B$  messages  $m^c$  and  $m^s$  from the speaker networks;
11  | get  $B$  decision actions  $d$  from the messages and the listener network;
12  | calculate the loss  $\mathcal{L}$ ;
13  | update the parameters of two speaker networks and one listener network;
14 end
```

Output: N Trained speaker and listener networks

511

512 C Model architecture

513 Here we delineate the details on model architecture for the emergent group communication.

514 C.1 Variational Autoencoder

515 Variational autoencoder [Kingma and Welling, 2014] is used to encode the observations. The batch
516 size is 512, and the total number of training epochs is set to 1,000. ReLU [Nair and Hinton, 2010]
517 and LeakyReLU (0.2) [Maas et al., 2013] are used as the activation functions for the encoder and
518 decoder, respectively. Input is flattened 30,720-dimensional vector (32 by 320 by 3). Both encoder
519 and decoder have one hidden layer with the dimension size being 1,024. The latent variable z is a
520 20-dimensional vector. Finally, Adam optimizer [Kingma and Ba, 2014] is used with the learning
521 rate being 10^{-4} to minimize the binary entropy error.

522 C.2 Speaker and listener network

523 The speaker network takes the concatenation of the encoded observation image (20-dimensional)
524 and the instruction (11-dimensional) as an input. The network has two hidden layers, each with size
525 256. The output (communication message) is a 10-dimensional vector. Throughout the hidden layers,
526 ReLU is used as the activation function. In the final layer, no additional activation function is used.

527 The listener network takes the 10-dimensional aggregated communication messages from the color-
528 only and shape-only agents as an input. The network has one hidden layer with size 64. The output
529 (decision action) is a 10-dimensional vector, each feature assigned to the different positions of the
530 predicted target object. The hidden layer uses ReLU as the activation function, and softmax is used
531 in the final layer. For training, we set the batch size to 256. We used Adam optimizer and the binary
532 cross entropy loss function. The learning rate is set to 10^{-4} .

533 C.3 Early-stopping

534 For group communication, our analytical results can be affected by the number of training epochs.
535 For example, the reason why the message agreement is higher for $N = 8$ than for $N = 32$ might not
536 be because of its intrinsic group communication nature but just because the communication links
537 for $N = 8$ have gone through more training epochs. To prevent this from happening and to prevent
538 overfitting, we adopt the early stopping criteria for group communication settings. Specifically,
539 early stopping is enabled when the current best accuracy on the validation set has happened before
540 $N * \textit{patience}$ epochs. Throughout the experiments, we set the patience value to 50. Figure 7 shows
541 the training procedures for all-to-all communication with varying N .

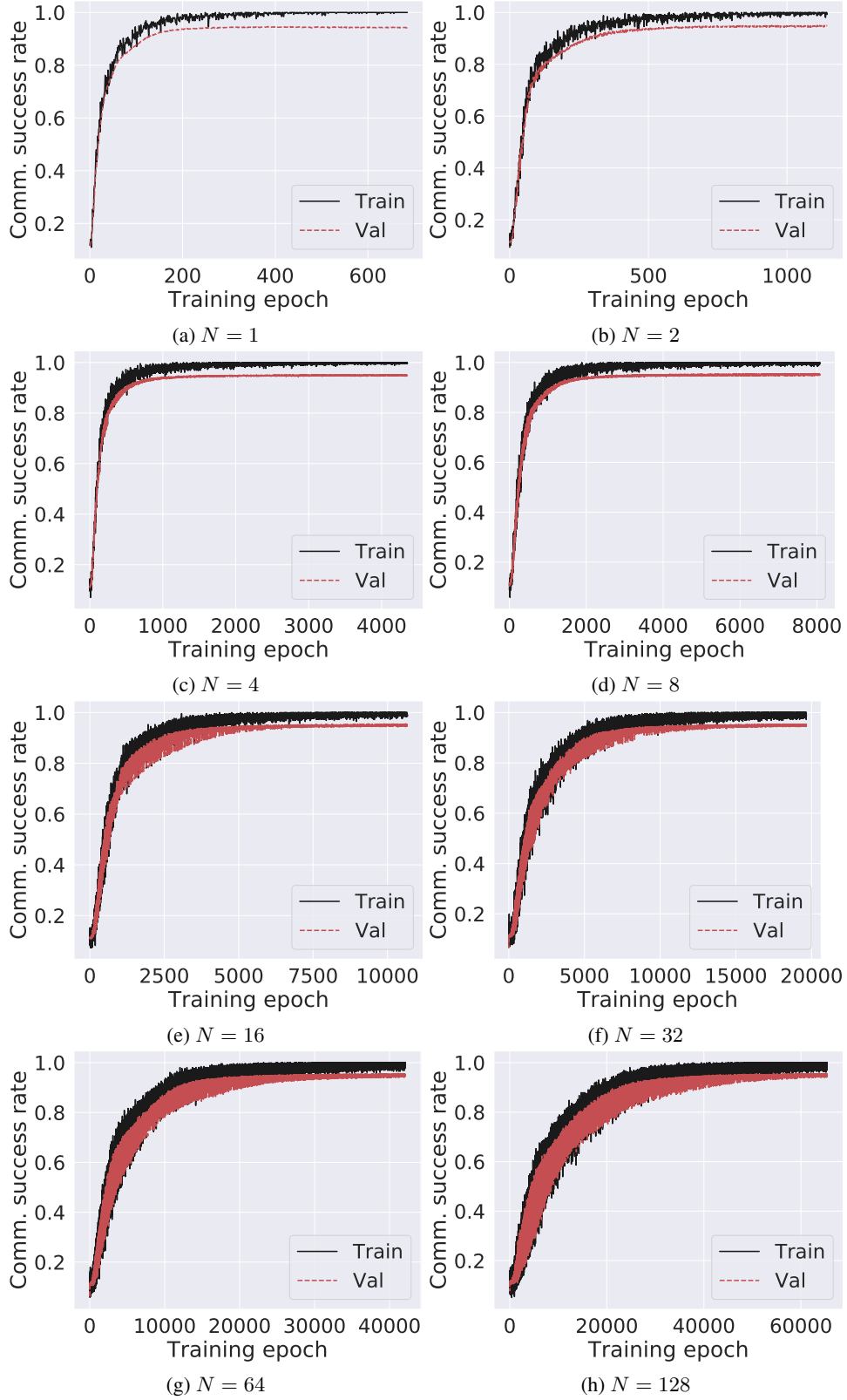


Figure 7: Training of all-to-all communication with varying N . Early stopping is enabled for fair comparison and to prevent over-fitting.

542 D Preliminary results

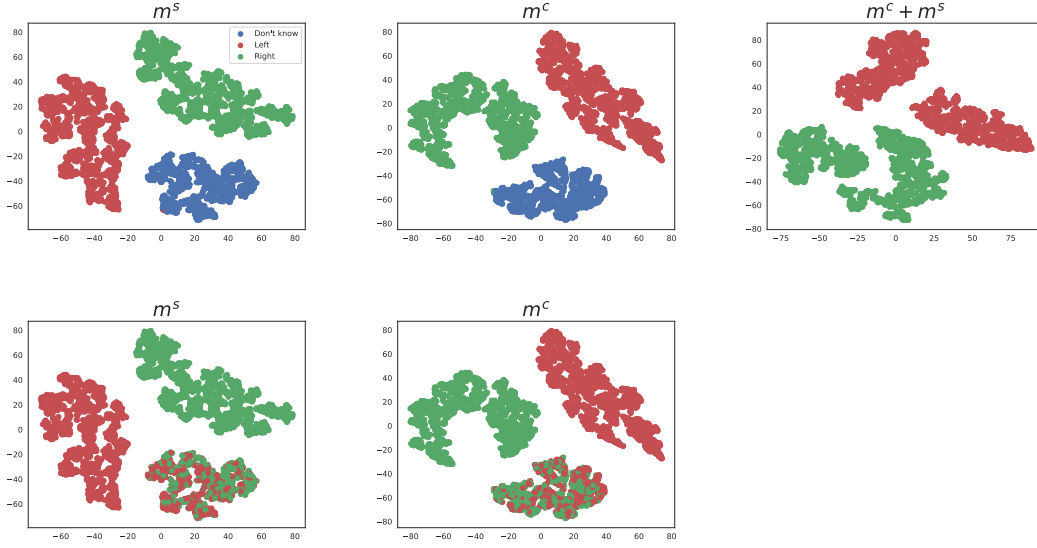


Figure 8: Communication when $N = 1$.

543 We show that agents can successfully communicate with interpretable messages when $N = 1$. In the
 544 preliminary study, for the sake of visualization and interpretation, we set the environment extremely
 545 simple, with objects having three shapes and colors only, and each image contains two objects only.
 546 Therefore, there can be only two decision actions: left, right. Figure 8 shows the communication
 547 between a color-only agent c and a shape-only agent s (i.e., $N = 1$). In the figure, each dot represents
 548 a single message. We colored the dots according to different observations and instructions. When the
 549 instructed object is on the left side of the observation, we colored the dots in red. When the instructed
 550 object is on the right side of the observation, we colored the dots in blue. In the bottom panels, there
 551 are clusters with mixed-colored dots. This is because of the perceptual limitations of the agents; for
 552 example, the instruction is 'find a red circle' and the left object is a red circle and the right object is a
 553 blue circle; in this case, a shape-only agent cannot differentiate the given two objects (bottom panel,
 554 left), and thus, the message can be interpreted as 'don't know' (top panel, left).

555 However, when the messages of the shape-only and the color-only are aggregated (top panel, right),
 556 we can always decipher the message (i.e., the messages are separable) and output a correct decision
 557 action, either left or right.

558 To summarize, when agents communicate, the messages encode information about where the target
 559 object described in the instruction is in the given observation. On the other hand, when we color
 560 the messages by e.g., the shapes or the colors of the objects, we cannot observe meaningful clusters.
 561 These messages are aggregated to compensate for each agent's perceptual limitations, and yield
 562 correct decision action d .

563 E Communication link optimization

564 To evaluate the communication success rate for a single graph structure, in order to reduce the training
565 time, we set the training epoch to 50. Since the communication success rate of a certain graph
566 fluctuates according to different initializations, we used the average value over 4 different trials
567 so as to stabilize the optimization process. Figure 9 shows the training of group communication
568 optimization. We observe that the optimization value stabilizes at around the 1000th iteration. With
569 the optimized link structure, we re-trained the communication graph with higher number of training
570 epochs and report the resulting number accordingly.

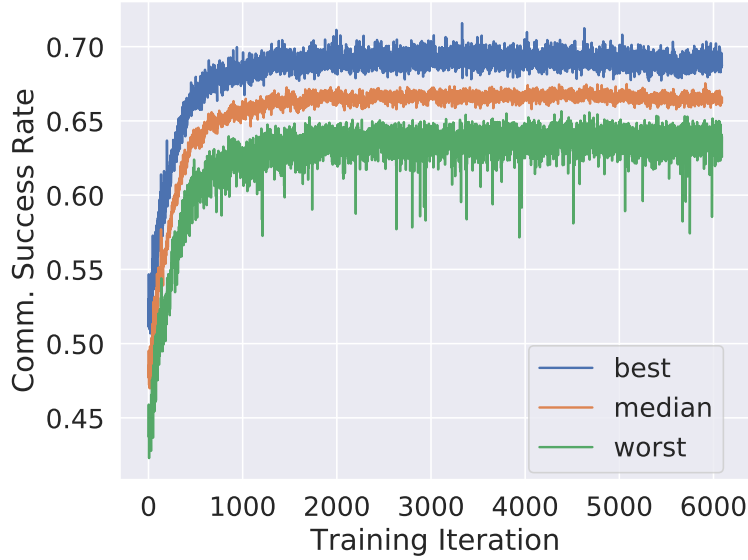


Figure 9: CMA-ES training for communication graph link structure optimization.

571 **F Reproducibility - Computing infrastructure**

572 For all group communication, we used single GPU (GeForce RTX 2080 Ti 10GB). For CMA-ES, we
573 used multi-CPU with the number of cores being 128, since can be straightforwardly parallelizable,
574 and optimizing with 128 CPUs was faster than optimizing with 8 GPUs.

575 All codes are based on Python’s PyTorch module [Paszke et al., 2019].

G Further discussions

In this work, we addressed the task of agents communicating in a shared, agreed language, using the emergent protocol. We acknowledge that it is extremely challenging to predict the future impact of our work in different levels and aspects, especially considering the nascent literature on emergent communication, and thereby limit our focus to:

1) The impact of emergent communication in relation to the human-in-the-loop training: In our research, our primary focus was on studying whether a universal language can be achieved in an agent system comprising of a large number of agents. Orthogonal to our research, there are some advance on emergent communication of artificial agents developing communication protocols under the human guidance [Lowe et al., 2020]. We believe that involving humans in the development of artificial agents’ communication system would be an indispensable direction for understanding the agent behavior, and our endeavor of focusing on the massive setting will *conditionally* bring positive impact on promoting human-AI interactions when this direction of development is taken into consideration.

2) The impact on/from foundational research: We discussed the potential linkage between the emergent communication with a growing number of agents in the system and overparameterization in neural networks. Future development in both the emergent communication field, as well as in the foundational research regarding generalization might result in bridging the two seemingly separate branches of machine learning.