ENTROPY MINIMIZATION IN EMERGENT LANGUAGES

Eugene Kharitonov [†], **Rahma Chaabouni**[†*], **Diane Bouchacourt**[†], **Marco Baroni**[†¶]
{kharitonov,rchaabouni,dianeb,mbaroni}@fb.com
[†] Facebook AI, [*] LSCP, [¶] ICREA

ABSTRACT

Converging evidence suggests that human languages evolve to optimize complexity-efficiency tradeoffs. We report here that, similarly, when deep networks are trained to jointly solve a task while communicating through a discrete channel, the emergent communication code naturally adapts to minimize its entropy, that is, the mutual information between the communicating agents' inputs and the messages is minimized, within the range afforded by the need for successful communication. Further, the entropy minimization trend is amplified as we increase channel discreteness, suggesting that the latter property might play an important role in enforcing efficiency pressures on communication systems.

1 INTRODUCTION

Multiple studies indicate that efficiency pressures are at work in language and other biological communication systems (Ferrer i Cancho et al., 2013; Gibson et al., 2019). One particular aspect of communicative efficiency, that has been robustly observed across many semantic domains, is a tendency to minimize lexicon entropy, to the extent allowed by the counteracting need for accuracy (Zaslavsky et al., 2018; 2019). For example, while most languages distinguish grandmothers from grandfathers, very few have separate words for mother- and father-side grandmothers, as the latter distinction would make communication only slightly more accurate at the cost of an increase in lexicon complexity (Kemp & Regier, 2012). We show here that *the protocol evolved by deep neural network agents that jointly solve a task by communicating through a discrete channel is subject to the same complexity minimization pressure.*

We establish our results in the context of signaling games (Lewis, 1969), as used in the recent deep agent language emergence literature (Lazaridou et al., 2016; Havrylov & Titov, 2017; Li & Bowling, 2019). There are two neural network agents, Sender and Receiver, provided with individual inputs at the beginning of each episode. Sender sends a message to Receiver, and Receiver has to perform an action based on its own input and the received message. Importantly, there is no direct supervision on the message protocol. We consider agents that are deterministic functions of their inputs (test-time).

As an example, consider a task of communicating a n-bit number, sampled uniformly at random from $0...2^n - 1$. The full number is shown to Sender, and its k ($0 \leq k \leq n$) bits are also revealed to Receiver. Receiver has to output the full number, based on the message from Sender and its own input. Would the Sender transmit the entire number through its message? In this case, the protocol would be "complex," encoding n bits. Alternatively, Sender could only encode the bits that Receiver does not know, and let Receiver fill in the rest by itself. This emergent protocol would be "simple," encoding less information about the number. We find experimentally that, once the agents are successfully trained to jointly solve the task, *the emergent protocol achieves the minimal entropy of the messages* or, equivalently in our setup, *the mutual information between Sender's input and messages*. In other words, the agents consistently approximate the simplest successful protocol (in the current example, the one transmitting $\approx n - k$ bits).

We can connect the entropies of Sender and Receiver inputs $\boldsymbol{i}_s$ and $\boldsymbol{i}_r$, messages $\boldsymbol{m}$, Receiver's output (the chosen action) $\boldsymbol{o}$, and ground-truth outputs $\boldsymbol{l}$ by standard inequalities[1] (Cover & Thomas, 2012). Denoting Sender's computation as a function $S : S(\boldsymbol{i}_s) = \boldsymbol{m}$, and R as Receiver's function ($R(\boldsymbol{m}, \boldsymbol{i}_r) = \boldsymbol{o}$), we obtain:

$$H(\boldsymbol{i}_s) \geq H(S(\boldsymbol{i}_s)) = H(\boldsymbol{m}) \geq H(\boldsymbol{m}|\boldsymbol{i}_r) \geq H(R(\boldsymbol{m}, \boldsymbol{i}_r)|\boldsymbol{i}_r) = H(\boldsymbol{o}|\boldsymbol{i}_r) \approx H(\boldsymbol{l}|\boldsymbol{i}_r), \quad (1)$$

An extended version of this study can be found in (Kharitonov et al., 2019a).

[1] We also use the fact that for a discrete r.v. x and a (deterministic) function g it holds that $H(x) \geq H(g(x))$.

where the last relation stems from the fact that after a successful training $\boldsymbol{o} \approx \boldsymbol{l}$. Our empirical measurements indicate that the entropy of the messages $\boldsymbol{m}$ in the emergent protocol tends to approach the lower bound: $H(\boldsymbol{m}) \to H(\boldsymbol{l}|\boldsymbol{i}_r)$, even if the upper-bound $H(\boldsymbol{i}_s)$ is far. Moreover, we observe that when the amount of information that Receiver needs is reduced, without changing other parameters, the emergent protocol becomes simpler (lower entropy).

2 METHODOLOGY

To study the informational complexity of the emergent communication protocol as a function of Receiver's information need, we devise signalling games that allow us to easily control the amount of information Receiver needs to perform its task. We can achieve this in two ways: either by controlling the amount of side information Receiver has, or by changing the required complexity of Receiver's outputs. We study these two possibilities in two games: Guess Number and Image Classification.

In Guess Number, the agents are trained to recover a vector with uniform Bernoulli-distributed components. In the second game, Image Classification, uses more naturalistic data, as the agents are jointly trained to classify pairs of hand-written MNIST digits (LeCun et al., 1998b).

Guess Number We draw an 8-bit integer z, $0 \leq z \leq 255$ uniformly at random. All bits are revealed to Sender as a 8-dimensional binary vector $\boldsymbol{i}_s$. The last k bits are revealed to Receiver ($0 \leq k \leq 8$) as its input $\boldsymbol{i}_r$. Sender outputs a single-symbol message $\boldsymbol{m}$ to Receiver. In turn, Receiver outputs a vector $\boldsymbol{o}$ that recovers all the bits of z and should be equal to $\boldsymbol{i}_s$.

Image Classification The agents are jointly trained to classify 28x56 images of two MNIST digits, stacked side-by-side (more details in Supplementary). Unlike Guess Number, Receiver has no side input. Instead, we control the informational complexity of Receiver's task by controlling the size of its output space, i.e., the number of labels we assign to the images. To do so, we group all two-digit sequences 00..99 into $N_l \in \{2, 4, 10, 20, 25, 50, 100\}$ equally-sized classes.

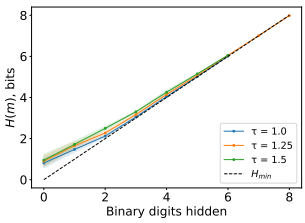
We report hyperparameter grids and architecutres of the agents in Supplementary. In the following experiments, we fix vocabulary to 1024 symbols (we observed the reported effect in experiments with other vocabulary sizes, multi-symbol messages, and larger architectures, which we omit due to space constraints). When training with REINFORCE (see below), we use the 0/1 loss. Otherwise, we use binary cross-entropy (Guess Number) and negative log-likelihood (Image Classification).

Training with discrete channel Training to communicate with discrete messages is non-trivial, as we cannot back-propagate through the messages. To highlight the generality of the observed effect, we use two approaches, most popular in the language emergence work: Gumbel-Softmax relaxation (Maddison et al., 2016; Jang et al., 2016) and REINFORCE (Williams, 1992). We also explore the Stochastic Computation Graph (SCG) optimization approach (Schulman et al., 2015), where Receiver is trained via conventional backpropagation and Sender is trained with REINFORCE. We plug the obtained gradient estimates into the Adam optimizer (Kingma & Ba, 2014). Due to space constraints, for further details on these methods we refer to a description of the EGG framework (Kharitonov et al., 2019b) which we used to implement the experiments.

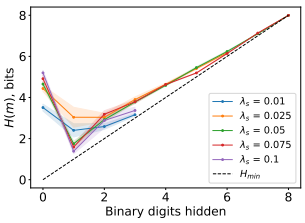
Entropy regularization When training with REINFORCE and SCG, we use a standard trick to aid exploration by adding an entropy regularization term which explicitly *maximizes* Sender's output entropy (Williams & Peng, 1991). The trade-off between the communication loss and the entropy regularization is controlled by a coefficient λ_s. Clearly, this regularization is at odds with the entropy *minimization* effect we observe. In our experiments, we found that high values of λ_s prevent communication success; on the other hand, small non-zero λ_s is crucial for successful training. In Section 3 we investigate the effect of λ_s on entropy minimization.

Gumbel-Softmax temperature Another important hyperparameter is the Gumbel-Softmax relaxation temperature, τ. As τ tends to 0, the samples from the Gumbel-Softmax disrtibution get closer to one-hot samples; as $\tau \to +\infty$, the samples become uniform in their components. During training, we use these relaxed samples as messages from Sender, making the entire setup differentiable.

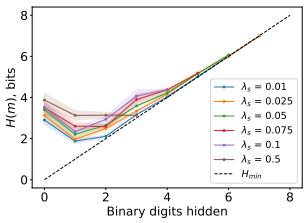
Experimental protocol In Guess Number, we use all 256 possible inputs for training, early stopping and analysis. In Image Classification, we train on random image pairs from the MNIST training



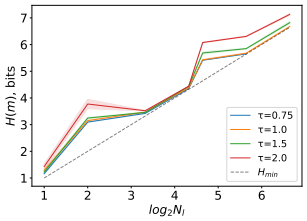
(a) Guess Number, Gumbel-Softmax relaxation.



(b) Guess Number, Stochastic Computation Graph.



(c) Guess Number, REINFORCE.



(d) Image Classification, Gumbel Softmax.

Figure 1: Entropy of the messages $\boldsymbol{m}$. Top: Guess Number. Bottom: Image Classification. Shaded regions mark standard error of the mean.

data, and use image pairs from the MNIST held-out set for validation. We select the runs that achieved high performance (training accuracy above 0.99 for Guess Number and validation accuracy above 0.98 for Image Classification), thus studying agent behavior *provided they succeeded at the game*.

At test time, we select the Sender's message symbol greedily, hence the messages are discrete and Sender represents a (deterministic) function S of its input $\boldsymbol{i}_s$, $\boldsymbol{m} = S(\boldsymbol{i})$. Calculating the entropy $H(\boldsymbol{m})$ of the distribution of discrete messages $\boldsymbol{m}$ is straightforward. In Guess Number, we enumerate all 256 possible values of z as inputs, save the messages from Sender and calculate entropy $H(\boldsymbol{m})$. For Image Classification, we sample image pairs from the MNIST hold-out set.

The upper bound on $H(\boldsymbol{m})$ is as follow: $H_{max} = 8$ bits (bounded by $H(\boldsymbol{i}_s)$) in Guess Number, and $H_{max} = 10$ bits (bounded by vocabulary size) in Image Classification. Its lower bound is equal to $H_{min} = H(\boldsymbol{l}|\boldsymbol{i}_r) = 8 - k$ bits for Guess number. In Image Classification, communication can only succeed if $H(\boldsymbol{m})$ is not less than $\log_2 N_l$, where N_l is the number of equally-sized classes.

3 Experiments

Guess Number In Figures 1a-1c, the horizontal axes span the number of bits of z that Receiver lacks, $8-k$. The vertical axis reports the information content of the protocol, measured by messages entropy $H(\boldsymbol{m})$. Each integer on the horizontal axis corresponds to a game configuration, and for each such configuration we aggregate multiple (successful) runs with different hyperparameters and random seeds. H_{min} indicates the minimal amount of bits Sender has to send in a particular configuration for the task to be solvable. The upper bound (not shown) is $H_{max} = 8$ bits.

Consider first the configurations where Receiver's input is insufficient to answer correctly (at least one binary digit hidden, $k \leq 7$). From Figures 1a-1c, we observe that the transmitted information is strictly monotonically increasing with the number of binary digits hidden from Receiver. Thus, even if Sender sees the very same input in all configurations, a more nuanced protocol is only developed when it is necessary. Moreover, the entropy $H(\boldsymbol{m})$ stays close to the lower bound. This entropy minimization property holds for all the considered training approaches across all configurations.

Consider next the configuration where Receiver is getting the whole integer z as its input ($k = 8$, the leftmost configuration in Figure 1, corresponding to 0 on x axis). One would expect that the protocol would approach zero entropy in this case (as no information needs to be transmitted). However, the measurements indicate the opposite. It turns out that this information is entirely ignored by Receiver.

To demonstrate this, we shuffled the messages from Sender in the batch, destroying any information about the inputs they might carry. The overall performance was not affected by this manipulation, confirming the hypothesis that Receiver ignores messages. We conclude that in this case there is no apparent entropy minimization pressure on Sender simply because there is no communication.

We further consider the effect of various hyperparameters. In Figure 1a, we split the results obtained with Gumbel-Softmax by relaxation temperature. As discussed in Section 2, lower temperatures more closely approximate discrete communication, hence providing a convenient control of the level of discreteness imposed during training (recall that at test time we select the symbol greedily). The figure shows that lower temperatures consistently lead to lower $H(\boldsymbol{m})$ values. This implies that, as we increase the "level of discreteness" at training, we get stronger entropy minimization pressures.

In Figures 1b & 1c, we report $H(\boldsymbol{m})$ when training with Stochastic Graph Optimization and REINFORCE across degrees of entropy regularization. We report curves corresponding to λ_s values which converged in more than three configurations. With REINFORCE, we see a weak tendency for a higher λ_s to trigger higher entropy in the protocol (only violated at $\lambda_s = 0.5$). However, message entropy stays generally close to the lower bound even in presence of strong exploration, which favors higher entropy in Sender's output distribution.

Image Classification As the models are more complex, we only had consistent success when training with Gumbel-Softmax. In Figure 1d we report all successful runs, aggregated by temperature. The information encoded by the protocol grows as Receiver's output requires more information. However, in all configurations, the transmitted information stays well below the 10-bit upper bound and tends to be close to H_{min}. A natural interpretation is that Sender prefers to take charge of image classification and directly pass information about the output label, rather than sending along a presumably more information-heavy description of the input. Again, we see that lower temperatures consistently lead to stronger entropy minimization pressures.

Summarizing, *when communicating through a discrete channel, there is consistent pressure for the emergent protocol to encode as little information as necessary*. This holds across games, training methods and hyperparameters. When training with Gumbel-Softmax, temperature controls the strength of this pressure, confirming the relation between entropy minimization and discreteness.

4 Discussion

Entropy minimization is pervasive in human language, where it constitutes a specific facet of the more general pressure towards communication efficiency. We found that the same property consistently characterizes the protocol emerging in simulations where two neural networks learn to solve a task jointly through a discrete communication code.

In a comparative perspective, our results suggest that entropy minimization is a general property of discrete communication systems, independent of specific biological constraints humans are subject to. In particular, our analysis tentatively establishes a link between this property and the inherent difficulty of encoding information in discrete form.

Our results also have implications for the efforts to evolve agents interacting with each other and with humans through a discrete channel. Due to the entropy minimization, we should not expect the agents to develop a richer protocol than the simplest one that will ensure accurate communication. For example, Bouchacourt & Baroni (2018) found that agents trained to discriminate pairs of natural images depicting instances of about 500 high-level categories, developed a lexicon that does not denote such categories, but low-level properties of the image themselves. This makes sense from an entropy-minimization perspective, as talking about the 500 high-level categories demands $\log_2 500$ bits of information, whereas many low-level strategies (e.g., discriminating average pixel intensity in the images) will only require transmitting a few bits. To have agents developing rich linguistic protocols, we must face them with varied challenges that truly demand them.

In the future, we would like to study more continuous domains, such as color maps, where perfect accuracy is not easily attainable, nor desirable. Will the networks find an accuracy/complexity trade-off similar to those attested in human languages?

REFERENCES

Diane Bouchacourt and Marco Baroni. How agents see things: On visual representations in an emergent language game. In *EMNLP*, 2018.

Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012.

Ramon Ferrer i Cancho, Antoni Hernández-Fernández, David Lusseau, Govindasamy Agoramoorthy, Minna Hsu, and Stuart Semple. Compression as a universal principle of animal behavior. *Cognitive Science*, 37(8):1565–1578, 2013.

Edward Gibson, Richard Futrell Steven Piantadosi, Isabelle Dautriche, Kyle Mahowald, Leon Bergen, and Roger Levy. How efficiency shapes human language. *Trends in Cognitive Science*, 2019. In press.

Serhii Havrylov and Ivan Titov. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. In *NIPS*, 2017.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-Softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Charles Kemp and Terry Regier. Kinship categories across languages reflect general communicative principles. *Science*, 336(6084):1049–1054, 2012.

Eugene Kharitonov, Rahma Chaabouni, Diane Bouchacourt, and Marco Baroni. Entropy minimization in emergent languages. *arXiv preprint arXiv:1905.13687*, 2019a.

Eugene Kharitonov, Rahma Chaabouni, Diane Bouchacourt, and Marco Baroni. Egg: a toolkit for research on emergence of language in games. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pp. 55–60, 2019b.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*, 2016.

Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *NIPS*, 1990.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998a.

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998b.

David Lewis. Convention harvard university press. *Cambridge, MA*, 1969.

Fushan Li and Michael Bowling. Ease-of-teaching and language structure from emergent communication. In *Advances in Neural Information Processing Systems*, pp. 15825–15835, 2019.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. In *NIPS*, 2015.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

Noga Zaslavsky, Charles Kemp, Terry Regier, and Naftali Tishby. Efficient compression in color naming and its evolution. *Proceedings of the National Academy of Sciences*, 115(31):7937–7942, 2018.

Noga Zaslavsky, Terry Regier, Naftali Tishby, and Charles Kemp. Semantic categories of artifacts and animals reflect efficient coding. In *Proceedings of CogSci*, pp. 1254–1260, Montreal, Canada, 2019.

A AGENT ARCHITECTURE DETAILS

Guess Number Sender has a linear layer that maps the input vector $\boldsymbol{i}_s$ to a hidden representation of size 10, followed by a leaky ReLU activation. Next is a linear layer followed by a softmax over the vocabulary. Receiver linearly maps both its input $\boldsymbol{i}_r$ and the message to 10-dimensional vectors, concatenates them, applies a fully connected layer with output size 20, followed by a leaky ReLU. Finally, another linear layer and a sigmoid nonlinearity are applied. When training with REINFORCE and the Stochastic Computation graph approach (see Section 2), we increase the hidden layer sizes threefold, as this leads to more robust convergence.

Image Classification In Sender, input images are embedded with a LeNet-1 instance (LeCun et al., 1990) into 400-dimensional vectors. These embedded vectors are passed to a fully connected layer, followed by a softmax selecting a vocabulary symbol. Receiver embeds the received messages into 400-dimensional vectors, passed to a fully connected layer with a softmax activation returning the class probabilities.

B TWO-DIGIT MNIST DATASET

As discussed in Section 2, to ensure high output informational complexity in the Image Classification task, we use a two-digit variant of the MNIST dataset (LeCun et al., 1998a). We construct it as follows. When iterating over the original MNIST dataset, we take a batch b and (a) select the first $|b|/2$ and last $|b|/2$ images, refer to them as b_1 and b_2, respectively; (b) create a new batch where the ith image from b_1 is placed to the left of the ith image from b_2 and then *vice versa*. As a result, we obtain a new stream of images, where each MNIST digit is seen twice, on the left and on the right side. Note that not all possible pairwise combinations of the original images are generated (there are 60000^2 of those in the training set alone) and the exact combinations change across epochs. As labels, we use the depicted two-digit number modulo N_l, where N_l is the required number of classes. All pixels are scaled into [0, 1]. We use the same process to generate training and test sets, based on the training and test images of the original MNIST dataset, respectively.

C HYPERPARAMETERS

In our experiments, we used the following hyperparameter grids.

Guess Number (Gumbel-Softmax) Vocab. size: [256, 1024, 4096]; temperature, τ: [0.5, 0.75, 1.0, 1.25, 1.5]; learning rate: [0.001, 0.0001]; max. number of epochs: 250; random seeds: [0, 1, 2, 3]; batch size: 8; early stopping thr.: 0.99; bits shown to Receiver: [0, 1, 2, 3, 4, 5, 6, 7, 8].

Guess Number (REINFORCE) Vocab. size: [256, 1024, 4096]; Sender entropy regularization coef., λ_s: [0.01, 0.05, 0.025, 0.1, 0.5, 1.0]; Receiver entropy regularization coef., λ_r: [0.01, 0.1, 0.5, 1.0]; learning rate: [0.0001, 0.001, 0.01]; max. number of epochs: 1000; random seeds: [0, 1, 2, 3]; batch size: 2048; early stopping thr.: 0.99; bits shown to Receiver: [0, 1, 2, 3, 4, 5, 6, 7, 8].

Guess Number (Stochastic Computation Graph approach): Vocab. size: [256, 1024, 4096]; Sender entropy regularization coef., λ_s: [0.01, 0.025, 0.05, 0.075, 0.1, 0.25]; learning rate: [0.0001, 0.001]; max. number of epochs: 1000; random seeds: [0, 1, 2, 3]; batch size: 2048; early stopping thr.: 0.99; bits shown to Receiver: [0, 1, 2, 3, 4, 5, 6, 7, 8].

Image Classification experiments Vocab. size: [512, 1024, 2048]; temperature, τ: [0.5, 0.75, 1.0, 1.5, 2.0]; learning rate: [0.01, 0.001, 0.0001], max. number of epochs: 100; random seeds: [0, 1, 2]; batch size: 32; early stopping thr.: 0.98; number of classes: [2, 4, 10, 20, 25, 50, 100].

D EVOLUTION OF MESSAGE ENTROPY DURING TRAINING

In this Section, we aim to gain additional insight into development of the communication protocol by measuring its entropy during training. We concentrate on Guess Number and use the same experimental runs summarized in Figure 1 of the main text.

For each game configuration (that is, number of bits hidden from Receiver), we randomly select one successful run and plot the evolution of Sender message entropy and accuracy over training epochs.[2] We also plot entropy and accuracy curves for a randomly selected failed run, to verify to what extent entropy development depends on task success.

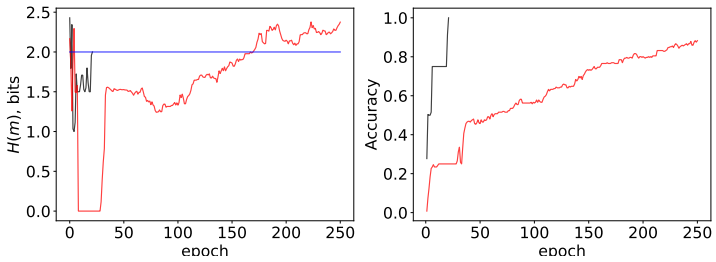
We report results for runs where training was performed with Gumbel-Softmax relaxation and with the Stochastic Graph Computation approach in Figures 2 and 3, respectively. The reported entropy and accuracy values are calculated in evaluation mode, where Sender's output is selected greedily, without sampling. A higher entropy of such deterministic Sender indicates that the latter can encode more information about inputs in its messages.

From these results, we firstly observe that the initial entropy of Sender's messages (before training) can be both higher than required for communication success (Figures 2a and 3a) and lower (the rest). When it starts higher than needed, it generally falls closer to the minimum level required for the solution. When the initial value is low, it increases during training. The failed runs can have message entropy above (Figures 2a, 2b & 3a) and below (e.g. Figures 2c, 2d & 3d) successful runs, suggesting that there is no systematic relation between degree of entropy and task success.

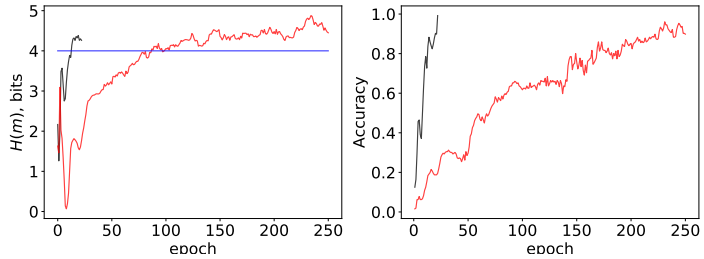
The fact that the entropy can be reduced with no decrease in accuracy or even with accuracy growth (e.g. Figure 2a, red line, epochs 5..30) indicates that the tendency to discover new messages (increasing entropy) is counter-balanced by the complexity of mutual coordination with Receiver when entropy is larger. In our interpretation, it is this interplay that serves as a source of the natural bottleneck.

Finally, while in some runs the entropy is effectively increased w.r.t. its initialization level, the resulting protocol's entropy is at, or slightly above the lower bound of what the task allows. In this sense, we argue that the reported effect can be correctly denoted as a "minimization" result.

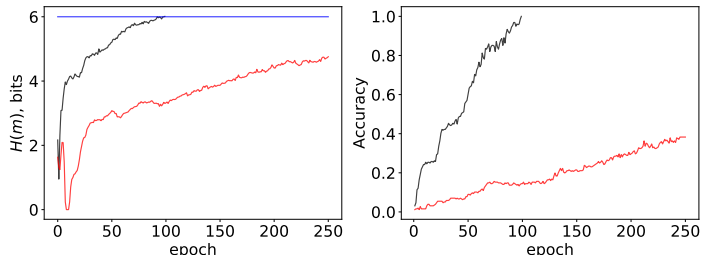
[2]We exclude the configuration in which Receiver sees the entire input, as it is a degenerate case of non-communication, as discussed in Section 4 of the main text.



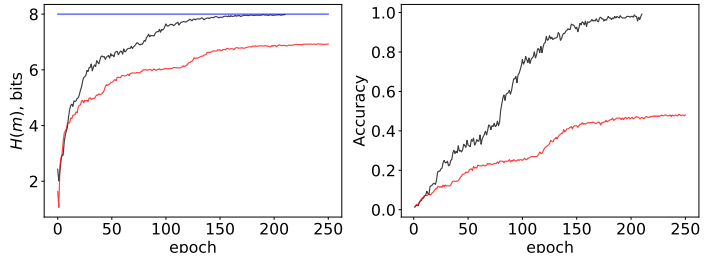
(a) Binary digits hidden: 2



(b) Binary digits hidden: 4

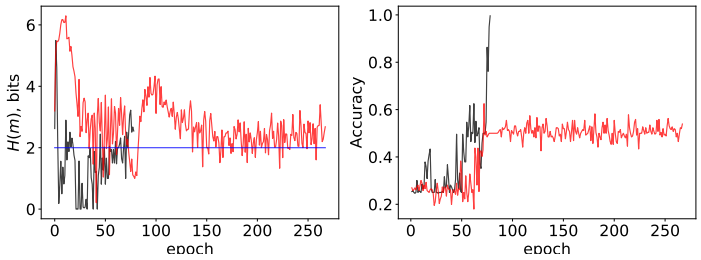


(c) Binary digits hidden: 6

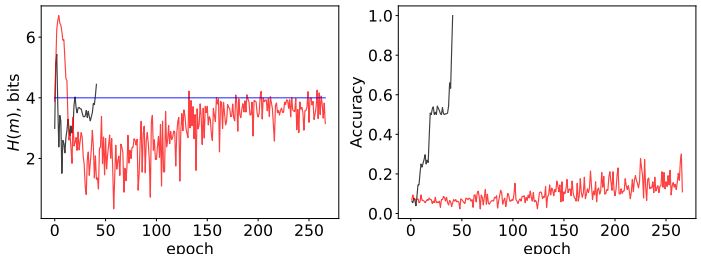


(d) Binary digits hidden: 8

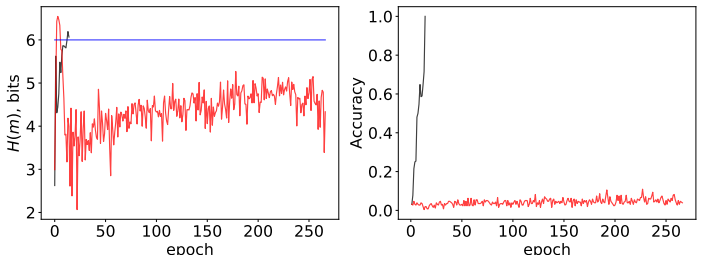
Figure 2: Evolution of $H(m)$ over training epochs. Gumbel Softmax-based optimization, Guess Number. For each game configuration, specified by the number of bits Receiver lacks, we sample one successful (black line) and one failed (red line) training trajectory. The blue line marks H_{min}, minimal entropy for a successful solution.



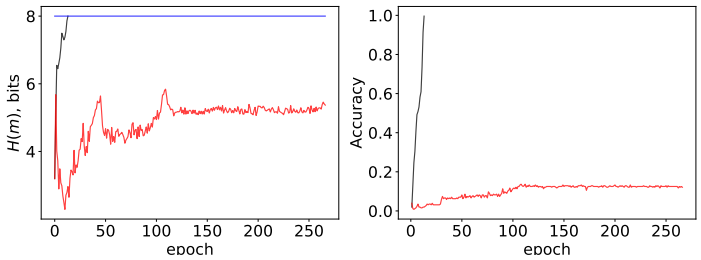
(a) Binary digits hidden: 2



(b) Binary digits hidden: 4



(c) Binary digits hidden: 6



(d) Binary digits hidden: 8

Figure 3: Evolution of $H(m)$ over training epochs. Stochastic Computation Graph-based optimization, Guess Number. For each game configuration, specified by the number of bits Receiver lacks, we sample one successful (black line) and one failed (red line) training trajectory. The blue line marks H_{min}, minimal entropy for a successful solution.