Entropy Minimization In Emergent Languages: Supplementary

Eugene Kharitonov [1] **Rahma Chaabouni** [1 2] **Diane Bouchacourt** [1] **Marco Baroni** [1 3]

1. How much does Receiver rely on messages in Guess Number?

We supplement the experiments of Section 3 of the main text by studying the degree to which Receiver relies on messages in Guess Number. In particular, we show that when Receiver has the full input ($\boldsymbol{i}_s = \boldsymbol{i}_r$), it ignores the messages.

We measure the degree to which Receiver relies on the messages from Sender by constructing a setup where we break communication, but still let Receiver rely on its own input. More precisely, we first enumerate all test inputs for Sender $\boldsymbol{i}_s$ and Receiver $\boldsymbol{i}_r$. We obtain messages that correspond to Sender's inputs, and shuffle them. Next, we feed the shuffled messages alongside Receiver's own (unshuffled) inputs and compute accuracy, as a measure of Receiver's dependence on the messages. This procedure preserves the marginal distribution of Sender's messages, but destroys all the information Sender transmits.

Without messages, Receiver, given k input bits, can only reach an accuracy of 2^{8-k}. In Figure 1, we report results aggregated by training method. Receiver is extremely close to the accuracy's higher bound in all configurations. Moreover, when Receiver gets the entire input, the drop in accuracy after shuffling is tiny, proving that Receiver's reliance on the message is minimal in that setting.

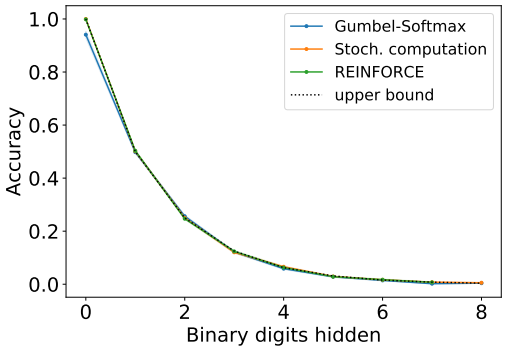


Figure 1. Guess Number: Receiver's dependence on messages, measured as performance drop under message intervention.

2. Influence of architecture choices

2.1. Does vocabulary size affect the results?

We repeat the same experiments as in Section 3 of the main text while varying vocabulary size. Note that, to make Guess Number solvable across each configuration, the vocabulary has to contain at least 256 symbols. Similarly, for Image Classification, vocabulary size must be of at least 100. We tried vocabulary sizes of 256, 1024, 4096 for Guess Number, and 512, 1024, 2048 for Image Classification. The results are reported in Figures 2 (Guess Number) and 3 (Image Classification). We observe that there is little qualitative variation over vocabulary size, hence the conclusions we had in Section 3 of the main paper are robust to variations of this parameter.

2.2. Does Receiver's capacity affect the results?

One potential confounding variable is the capacity of Receiver. Indeed, if Receiver is very simple, then, for the task to be solved, Sender would have to calculate the answer itself and feed it to Receiver. To investigate this, we repeat the Image Classification experiment from Section 4.1 of the main paper while controlling the power of Receiver's architecture: we put two additional fully-connected 400x400 hidden layers between the input embedding and the output layer, while in Section 4, Receiver had a single hidden layer.

In Figure 4, we compare the results obtained with these two variations of Receiver. The reported entropy minimization effect holds: even in presence of additional layers, the entropy of messages $H(\boldsymbol{m})$ is far from the upper-bound $H_{max} = 10$ bits and closely follows the lower bound, $H_{min} = \log_2 N_l$. Thus, again, a more nuanced protocol only appears when it is needed. Finally, we see that results for both architectures are close, although in three out of seven task setups (the number of classes N_l is 2, 10, and 20) a deeper model results in a slightly higher entropy of the protocol, on average. Overall, we conclude that Receiver's

[1]Facebook AI Research, Paris, France [2]Cognitive Machine Learning (ENS - EHESS - PSL - CNRS - INRIA) [3]Catalan Institute for Research and Advanced Studies, Barcelona, Spain. Correspondence to: Eugene Kharitonov <kharitonov@fb.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

capacity does not play a major role in the entropy minimization effect and the latter also takes place with a more powerful Receiver.

2.3. What if communication takes place through sequences of symbols?

We also experiment with Guess Number in a setup where the agents communicate via variable-length messages. The general architecture of the agents is same as in Section 3, but we append GRU agents (Cho et al., 2014). Sender GRU is unrolled to generate the message. The message is produced until the GRU outputs a special *eos* token or until the maximal length is reached. In the latter case, *eos* is appended to the message. The produced message is consumed by a Receiver's GRU unit and the hidden state corresponding to *eos* is used by Receiver as input to further processing. When Receiver has additional inputs (in the Guess Number game), these inputs are used as initial hidden state of the GRU cell. We use the Stochastic Computation Graph estimator as described in Section 3.2, as it provided fastest convergence.

We consider the entire variable-length message as the realization of a random variable $\boldsymbol{m}$ when calculating the entropy of the messages, $H(\boldsymbol{m})$. The results are reported in Figure 5, arranged in function of maximal message length and vocabulary size. As before, we aggregate the successful runs according to the entropy regularization coefficient λ_s applied to Sender's output layer.

From Figure 5 we observe that the results are in line with those obtained in the one-symbol scenario. Entropy minimization still holds: a more nuanced (high-entropy) protocol only develops when more digits are hidden from Receiver, which hence requires more information to perform the task. The approximation to the lower bound is however less tight as the overall number of possible messages grows (higher maximum length and/or vocabulary size). There is also a weak tendency for lower λ_s to encourage a tighter bottleneck.

In preliminary experiments, we have similar results when the variable-length communication is performed via Transformer cells (Vaswani et al., 2017) instead of GRUs (not reported here).

3. Two-digit MNIST dataset

As discussed in Section 3, to ensure high output informational complexity in the Image Classification task, we use a two-digit variant of the MNIST dataset (LeCun et al., 1998). 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 this same process to generate training and test sets, based on the training and test images of the original MNIST dataset, respectively.

4. 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].

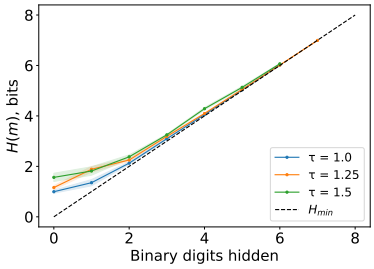
Guess Number (REINFORCE) Vocab. size: [256, 1024, 4096]; Sender entropy regularization coef., λ_s: [0.01, 0.025, 0.05, 0.1, 0.5, 1.0]; Receiver entropy regularization coef., λ_r: [0.01, 0.1, 0.5, 1.0]; learning rate: [0.0001, 0.001, 0.01]; max. number of epochs: 1000; random seeds: [0, 1, 2, 3]; batch 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, 0.05, 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].

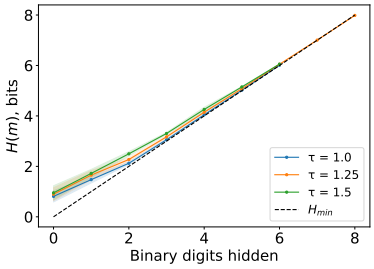
Image Classification experiments Vocab. size: [512, 1024, 2048]; temperature, τ: [0.5, 0.75, 1.0, 1.5, 2.0]; learning rate: [0.001], max. number of epochs: 100; random seeds: [0, 1, 2]; batch size: 32; early stopping thr.: 0.98; number of classes: [2, 4, 10, 20, 25, 50, 100].

Fitting random labels experiments Vocab. size: 1024; temperature, τ: [1.0, 10.0]; learning rate: 0.0001, max. number of epochs: 200; random seeds: [0, 1, 2, 3, 4]; batch size: 32; early stopping thr.: ∞; prob. of label corruption: [0.0, 0.5, 1.0].

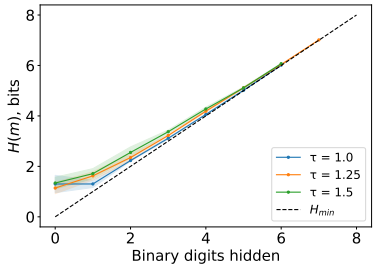
Adversarial attack experiments Vocab. size: 1024; temperature, τ: [0.1, 1.0, 10.0]; learning rate: 0.0001, max. number of epochs: 200; random seeds: [0, 1, 2, 3, 4]; batch size: 32; early stopping thr.: 0.98.



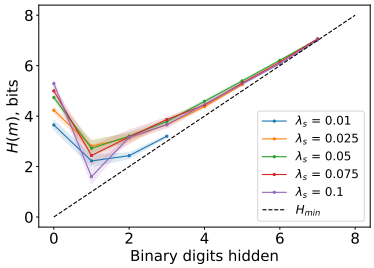
(a) Vocab. size: 256, Gumbel-Softmax



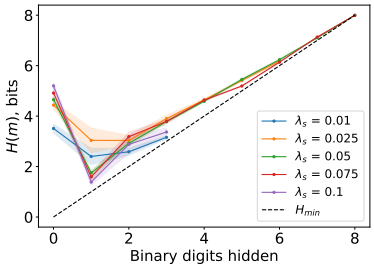
(b) Vocab. size: 1024, Gumbel-Softmax



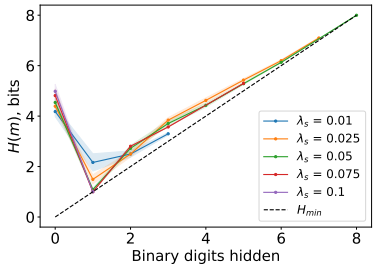
(c) Vocab. size: 4096, Gumbel-Softmax



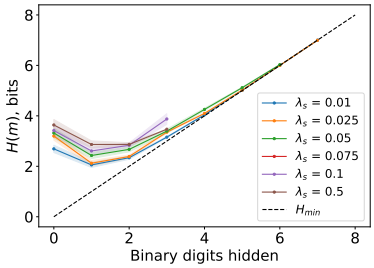
(d) Vocab. size: 256, Stoch. Computation Graph approach



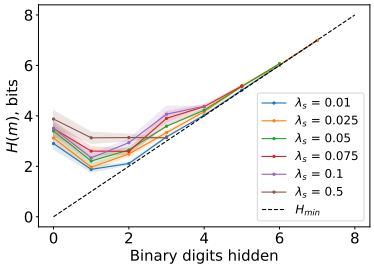
(e) Vocab. size: 1024, Stoch. Computation Graph approach



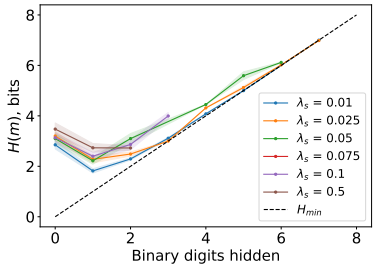
(f) Vocab. size: 4096, Stoch. Computation Graph approach



(g) Vocab. size: 256, REINFORCE

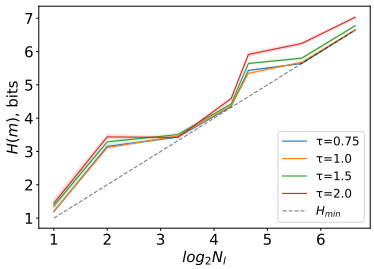


(h) Vocab. size: 1024, REINFORCE

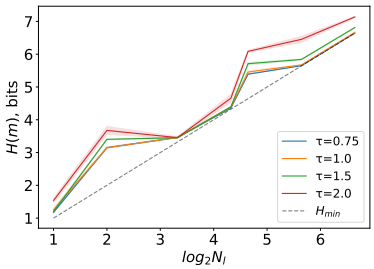


(i) Vocab. size: 4096, REINFORCE

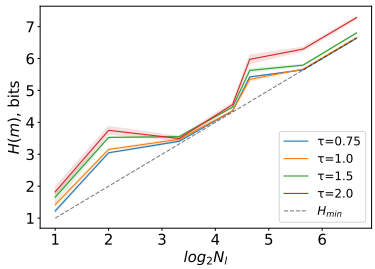
Figure 2. Guess Number: Entropy of the messages $\boldsymbol{m}$, depending on vocabulary size, training method, and relaxation temperature τ (when trained with Gumbel-Softmax) or Sender's entropy regularization coefficient λ_s. Shaded regions mark standard deviation.



(a) Vocab. size: 512



(b) Vocab. size: 1024



(c) Vocab. size: 2048

Figure 3. Image Classification: entropy of the messages $H(\boldsymbol{m})$ across vocabulary sizes. Successful runs are pooled together. Shaded regions mark standard deviation.

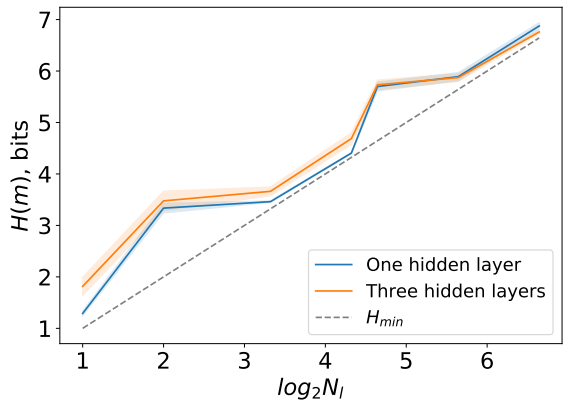


Figure 4. Image Classification: entropy of the messages $H(\boldsymbol{m})$ across Receiver model sizes. Successful runs are pooled together. Shaded regions mark standard deviation.

5. 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.[1] 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 6 and 7, 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 6a and 7a) 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 6a, 6b & 7a) and below (e.g. Figures 6c, 6d & 7d) 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 6a, 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.

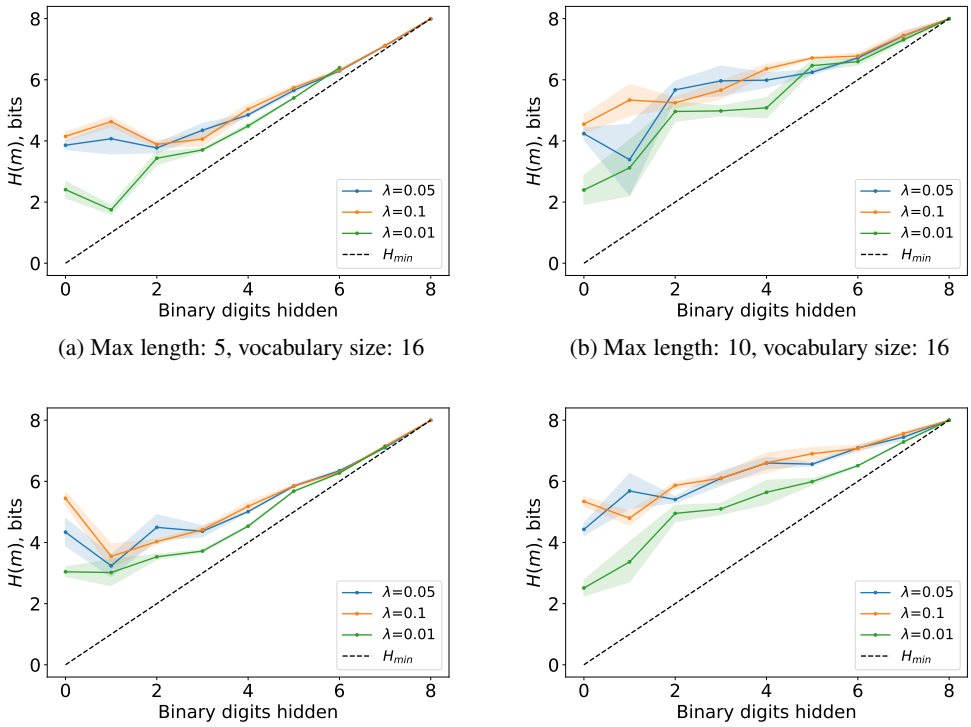
[1] We exclude the configuration in which Receiver sees the entire input, as it is a degenerate case of non-communication, as discussed in Section 4 of the main text.

References

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

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



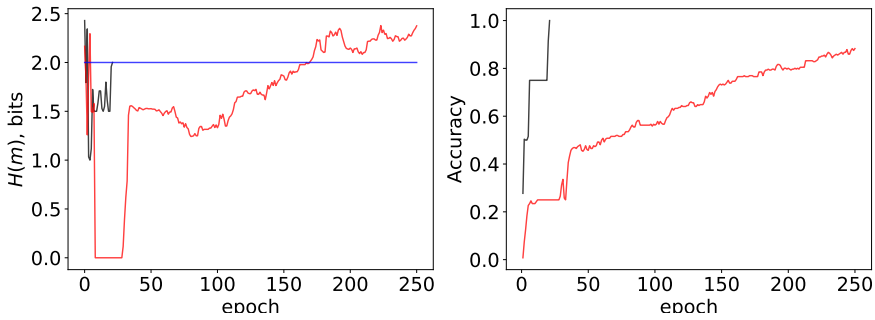
(a) Max length: 5, vocabulary size: 16

(b) Max length: 10, vocabulary size: 16

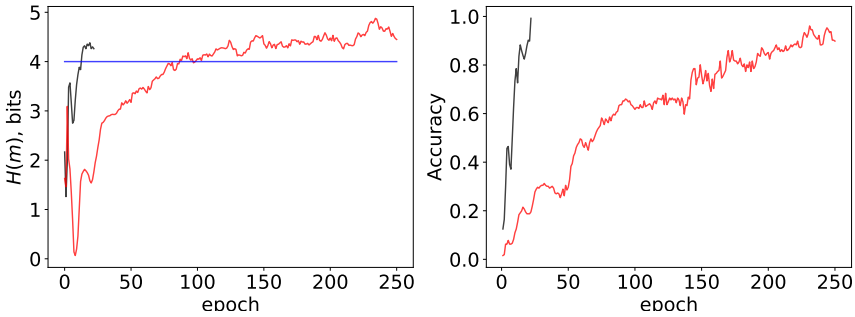
(c) Max length: 5, vocabulary size: 64

(d) Max length: 10, vocabulary size: 64

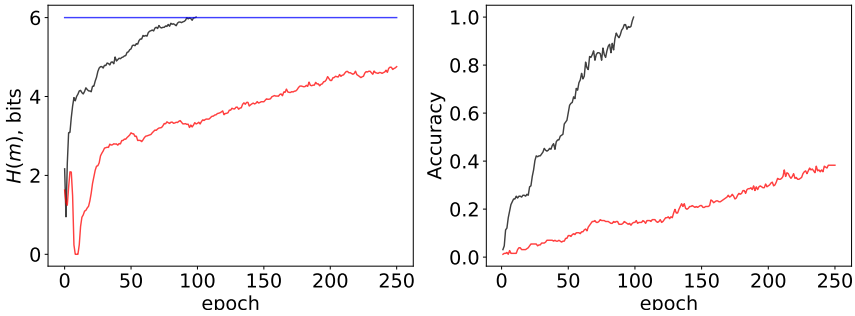
Figure 5. Guess Number: Entropy of the emergent protocol when communication is performed with variable-length messages. Shaded regions mark standard deviation.



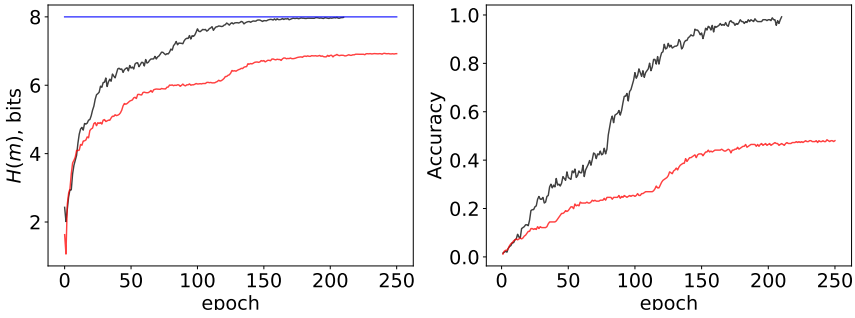
(a) Binary digits hidden: 2



(b) Binary digits hidden: 4

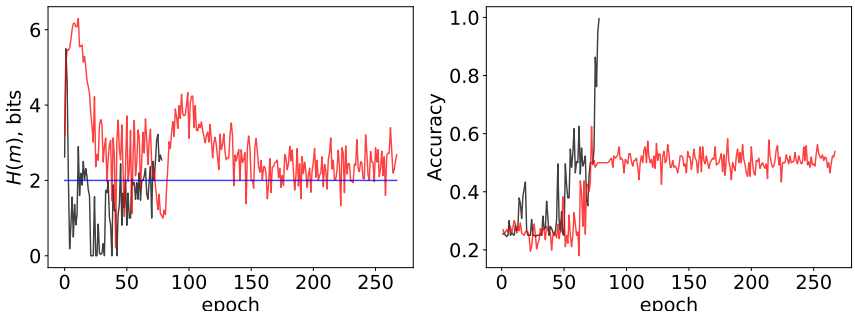


(c) Binary digits hidden: 6

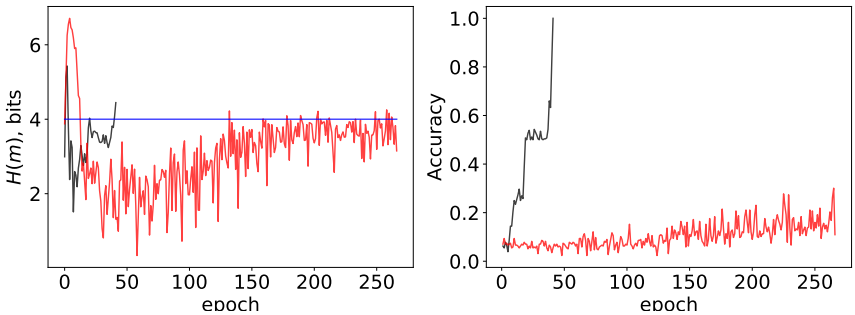


(d) Binary digits hidden: 8

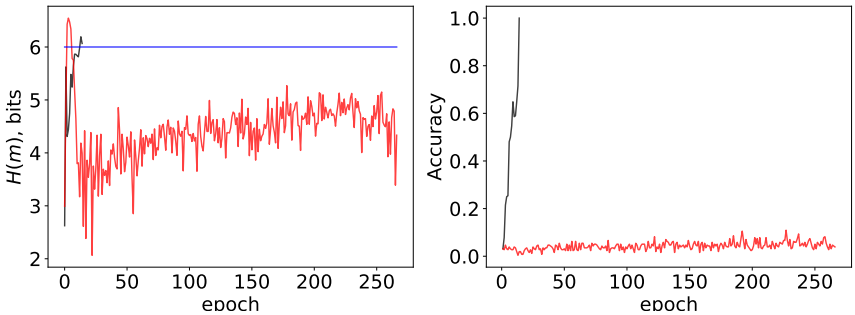
Figure 6. Evolution of $H(m)$ over training epochs. Gumbel Softmax-based optimization, Guess Number. For each game configuration, specified by the number of bits Receiver lacks, we sample one successful (black line) and one failed (red line) training trajectory. The blue line marks H_{min}, minimal entropy for a successful solution.



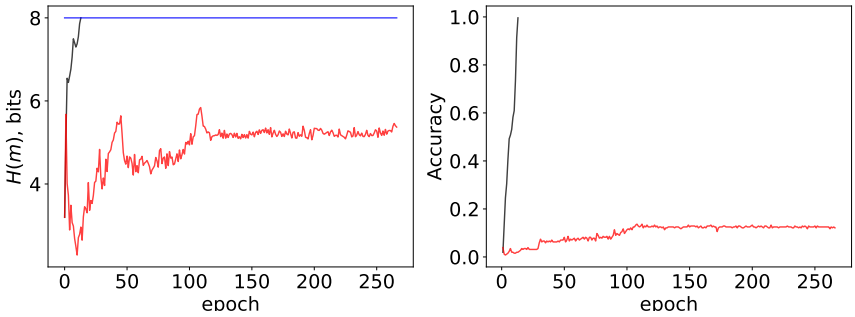
(a) Binary digits hidden: 2



(b) Binary digits hidden: 4



(c) Binary digits hidden: 6



(d) Binary digits hidden: 8

Figure 7. 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.