1. **Referential Game description**

The game includes multiple turns.

In each turn, two symmetrically-designed agents randomly drawn from the same linguistic community play a referential game. The two agents are arbitrarily assigned the sender role and the receiver. An image represented by a z-dimensional feature vector sampled from a 2D geometric shapes dataset is presented to the sender. The sender, after processing the target images, send a message of maximum length L to the receiver. The receiver, based on this message, has to guess the target image presented to the sender from a lined-up confounders. The sender’s strategy is to map the states of the world, or 2D geometric figures in this case, to signals, or a sequence of tokens in this setting. On the other hand, the receiver’s strategy is to map the signals to the states of the world. The communication loss function, or the inverse of both the sender’s and the receiver’s shared utility, is defined as following:

where *Lc* denotes the loss function; *q*(*t*) is the similarity between the receiver’s guess and the target image; *q*(*dk*​) is the how similar it is between the receiver’s guess and the kth distractor image. Specifically, the similarity,  , is defined by the dot product between the receiver’s guess vector, , and the feature representation vector of an image, . After the loss function is computed in each turn, the sender and the receiver adjust their strategies accordingly.

1. **Dataset**

The dataset contains 224×224-pixel images of 2D geometric figures with different shapes, colors, positions, rotation angles, and sizes labeled with their colors and shapes. They are used as the input of the visual module.

1. **Visual Module**

The visual module is designed to extract features from an image and is pre-trained separately. It processes the 224×224-pixel images from the dataset with 6 layers. The first 5 layers each combine a convolutional layer, a batch normalization, and a ReLu activation. Each layer has 20 filters, a kernel size of 3, and a stride of 2. The sixth layer is a fully connected layer with a ReLu activation, producing a 2048-dimensional feature vector.

1. **Agents’ Architecture**

Agents are capable of both sending the receiving messages and using the same knowledge for both generating and interpreting messages. Specifically, each agent is implemented as a 5-layer LSTM network, and an individual agent uses the same, shared LSTM network for both the sender and the receiver roles.

**4.1 The Sender’s Architecture**

The feature of an image extracted by a visual module, after a linear transformation, is used as the initial hidden state of the first layer, and the starting token, <S>, is the initial input token. After that, a token is sampled from the output distribution using Gumbel-Softmax technique (temperature=1.2, learning rate=0.0001). The following four layers each inherit the cell state and the hidden state of the previous one, and take the output sequence of the previous layer as the input sequence. Also, the sampled token from the output distribution of the previous layer is passed into the layer as the input token. The tokens generated are stacked and thus form a sequence of token, or the output message.

**4.2 The Receiver’s Architecture**

The receiver role uses the same, share-weighted 5-layer LSTM network as the sender role of the same agent. The receiver’s input is the message generated by the sender. As in the sender’s architecture, the layers also each inherit the cell state and the hidden state of the previous one, and take the output sequence of the previous layer as the input sequence. The difference is that the receiver’s LSTM layers’ input token is the corresponding token in the received message, since the LSTM network is now used for decoding the message, not generating it. A dense layer processes the final hidden layer of the LSTM and produces a guessing vector, which also have 2048 dimensions and serves as the ouput of the receiver.