EMERGENT ROBUST COMMUNICATION FOR MULTI-ROUND INTERACTIONS IN NOISY ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

We contribute a novel multi-agent architecture capable of learning a discrete communication protocol without any prior knowledge of the task to solve. We focus on ensuring agents can create a common language during their training to be able to cooperate and solve the task at hand, which is one of the primary goals of the emergent communication field. On top of this, we focus on increasing the task's difficulty by creating a novel referential game, based on the original Lewis Game, that has two new sources of complexity: adding random noise to the message being transmitted and the capability for multiple interactions between the agents before making a final prediction. When evaluating the proposed architecture on the newly developed game, we observe that the emerging communication protocol's generalization aptitude remains equivalent to architectures employed in much simpler and elementary games. Additionally, our method is the only one suitable to produce robust communication protocols that can handle cases with and without noise while maintaining increased generalization performance levels.

1 INTRODUCTION

Emergent communication focuses studying the human language's evolution and conceptualizing artificial languages for human-robot communication. Furthermore, with recent advances in deep learning, emergent communication has received special attention from the machine learning community to create learning experiments with neural agents that learn how to communicate without any prior knowledge. From this research standpoint, most works explore emergent communication through a reference-based game called the *Lewis Game* (LG) (Lewis, 1979). In this game, the *Speaker* describes an object to the *Listener*, which has then to discriminate it given a set of candidates. Usually, the communication protocols learned by neural agents when playing the LG appear to be degenerate and lack some core properties of human languages (Rita et al., 2020; Korbak et al., 2020; Chaabouni et al., 2019). Such an outcome is an effect of applying several constraints and simplifications to the Lewis Game to alleviate task complexity to facilitate its modeling as a machine learning problem. For instance, Choi et al. (2018) simplifies the LG by having the Listener describing only two images, and the images have only two effective degrees of freedom. Ren et al. (2020) developed an iterated learning strategy for the LG, intending to create highly compositional languages by creating an algorithm with several prolonged distinct learning phases. Chaabouni et al. (2022) focused on understanding how a population of agents can interfere with the emergence of a communication protocol. This work also scaled some properties of the LG, such as the dataset size used and the number of candidates. However, the authors assume that the Listener always receives information about the correct candidate during the learning phase.

In this paper, we use the work of Chaabouni et al. (2022) as a starting point and lift some of the constraints imposed by the works discussed in the previous paragraph to design more general games, which we can view as a step forward to emulate human communication. In particular, we propose two novel changes to the original LG. First, we add a noisy communication channel where the Speaker's message can suffer unexpected modifications. Second, we also add a time dependency to the LG, where the Listener can decide to play another round of the game or make a final choice.

With these new extensions, we create a more challenging and complete version of the LG to study how a language can emerge in such conditions and also how the properties of the emerged language fundamentally depend on the environment. As a case point, since the broadcasted messages

have some level of uncertainty (introduced by the noise), we can extrapolate this problem as a memory limitation constraint where it becomes impractical for the Listener to memorize specific patterns (Galke et al., 2022). We note that Ueda & Washio (2021) also introduced a setup with noise in the communication channel, using an adversarial setting, to analyze a specific language property: Zipf's law of abbreviation (Zipf, 1999). Conversely, our work focuses on a different axis, which centers on creating robust communication protocols where the pair of agents can communicate with different levels of noise. Furthermore, the authors also implement a simpler version of LG where one-hot vectors are used as input, contrary to real-world images (our case), making the training and architecture design more straightforward.

Additionally, introducing a time dependency to the LG also makes the Listener's decision process more challenging, diminishing the effect of having a large model capable of memorizing specific message patterns instead of language concepts. Ultimately, we expect this pathway to promote compositionality and generalization. As a reference, Bogin et al. (2018) also developed an environment where communication happens during long-horizon tasks. Contrary to our architecture, the authors constrained the Speaker's learning procedure to explicitly output similar messages in related conditions to be able to create functional emerging languages.

We introduce a novel two-agent architecture to solve the games described above. Our method relies on a new approach where both agents, Speaker and Listener, are modeled as reinforcement learning (RL) agents. For our novel Listener architecture, we add a meta-action to model the Listener's decision to play another round instead of making a final decision. With this new action, we allow the Listener to reflect on the current state of the game and the information obtained to decide whether it is best to gather more information or attempt to predict the correct candidate. Our results show that agents trained in the new game learn robust communication protocols to deal with deterministic or noisy messages without losing their general properties. The same does not happen when the agents train in the original LG, where the emerging language cannot handle any noise level.

To summarize, our contributions are 3-fold. First, we introduce a novel game extending the original LG, where the communication channel can suffer interference, implying that some parts of the message can become concealed. Moreover, we extend the LG to have multiple rounds, where more information can flow between the agents before selecting a candidate. We call the introduced game the Multi-Round Indecisive Lewis Game, or MRILG. Second, we develop a new Listener architecture to play the MRILG. When acting, the new architecture deliberates with current and previous round information, choosing whether to play another round or make a final prediction. Third, we evaluate the generated languages in the original and novel LG games regarding their generalization capabilities and transfer learning competence to new tasks. Our results show that the languages are intrinsically different, where simple modifications to the game, like adding stochasticity, can improve the language being learned, allowing for better generalization.

2 METHODOLOGY

We start this section by describing the original LG. Next, we explain how to extend the LG to create the proposed and more challenging MRILG. The main changes feature a noisy communication channel and a loop to play the game across multiple rounds, significantly increasing the complexity of the environment. Afterward, we detail how the Speaker converts the received input into a message, a sequence of discrete tokens, and how the Listener processes and integrates the message and candidates to make decisions. We impose a RIAL setting (Foerster et al., 2016), where agents are independent and perceive the other as part of the environment. Hence, we describe the learning strategy for both agents independently, focusing on explaining the loss composition and the importance of each loss term to guide training where functional communication protocols can emerge.

2.1 LEWIS GAME (LG)

The Lewis Game (LG) is a discrimination game where one of the agents, the *Speaker*, must describes an object by sending a message to the other agent, the *Listener*. When the game starts, the Speaker receives a target image $\boldsymbol{x}$ retrieved from a fixed dataset $\mathbb{X}$. The Speaker intends to describe the image by encoding a message $\boldsymbol{m}$, a sequence of T discrete tokens, $\boldsymbol{m}\left(\boldsymbol{x};\theta\right) = \left(u_t\left(\boldsymbol{x};\theta\right)\right)_{t=1}^{T}$. We define $\boldsymbol{m} : \mathbb{X} \to \mathbb{W}^T$, where $\mathcal{W}$ is a finite vocabulary set, and $u_t : \mathbb{X} \to \mathbb{W}$ is a symbol of the vocabulary.

Subsequently, the Listener receives the message along with a set of candidate images, $\mathbb{C} \subset \mathbb{X}$, where the goal is to try to identify the image $\boldsymbol{x} \in \mathbb{C}$ that the Speaker received, $\hat{\boldsymbol{x}} = \boldsymbol{x}$. Both agents receive a reward of 1 if the Listener is correct, and -1 otherwise:

$$R(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \begin{cases} 1, & \text{if } \hat{\boldsymbol{x}} = \boldsymbol{x} \\ -1, & \text{otherwise.} \end{cases}$$

When there is no ambiguity, we drop the dependence of for $\boldsymbol{m}(\boldsymbol{x}; \theta)$ on $\boldsymbol{x}$ and θ. Furthermore, the Listener's choice $\hat{\boldsymbol{x}}$ depends on $\boldsymbol{m}$, $\mathbb{C}$, and ϕ, where ϕ contains the Listener's parameters. However, such dependencies will also be omitted for legibility.

2.2 Multi-Round Indecisive Lewis Game (MRILG)

We now introduce and propose a novel extension of the LG, the Multi-Round Indecisive Lewis Game (MRILG). MRILG is composed of several iterative LG rounds. There are at most N rounds to prevent infinite games. At each round, $i \in \{0, \ldots, N-1\}$, and similarly to the LG, the Speaker receives a target image $\boldsymbol{x}$ and describes it to the Listener by sending a message composed of discrete tokens, $\boldsymbol{m}(\boldsymbol{x}; \theta) = (u_t(\boldsymbol{x}; \theta))_{t=1}^{T}$. MRILG has a noisy communication channel, meaning the message can suffer random perturbations when being transmitted. We define a new function that converts each token, with a given probability, into a default unknown token, $\text{noise} : \mathbb{W}^T \rightarrow \mathbb{W}'^T$, where $\mathbb{W}'$ contains the original vocabulary $\mathbb{W}$ plus the unknown token, $\mathbb{W}' = \mathbb{W} \cup \{\texttt{unk}\}$. Accordingly, we define a new function to disrupt the message before giving it to the Listener:

$$\begin{aligned} \text{noise}(\boldsymbol{m}) &= (n_t(u_t(\boldsymbol{x}; \theta)))_{t=1}^{T} \\ \text{s.t. } n_t(u_t(\boldsymbol{x}; \theta)) &= \begin{cases} u_t(\boldsymbol{x}; \theta), & \text{if p} > \lambda \\ \texttt{unk}, & \text{otherwise,} \end{cases} \end{aligned} \tag{1}$$

where p is sampled from a uniform distribution, $\text{p} \sim \mathcal{U}(0, 1)$, and λ is a fixed threshold, indicating the noise present in the communication channel. By definition, the Speaker is agnostic to this process and will never know if the message was modified.

Subsequently, the Listener receives the modified message, $\text{noise}(\boldsymbol{m})$, and the candidate images, $\mathbb{C}$. Due to the recurrent nature of MRILG, the Listener must be able to leverage different types of actions, thus simulating hierarchical reasoning. Namely, the Listener can decide whether it is preferable to play another round, to gather more information, or to try to predict the correct candidate when anticipates a positive tradeoff between the likelihood of success and the expected reward.

When the Listener makes a final prediction $\boldsymbol{x} \in \mathbb{C}$, both agents receive a positive reward of 1 if the Listener is correct in its prediction and -1 otherwise. Alternatively, the agents proceed to a new round when the Listener plays the *I don't know* (*idk*) action, $\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}_{\text{idk}}$. We carefully design the reward associated with the *idk* action to avoid degenerate and exploitable cases. For instance, when the *idk* reward tends to the value of the failure reward, the Listener will always attempt to make a final prediction in the first round to avoid huge penalizations. On the other hand, when the *idk* reward exceeds 0, the Listener gets an incentive to wait for the last round to increase the cumulative reward, even if it knows the correct answer. Additionally, when the gap between the *idk* and failure reward surpasses a certain threshold the Listener will also always play *idk*. In these cases no communication protocol emerges since the Listener completely ignores the Speaker's message.

To tackle all cases above, we define the reward associated with the *idk* action as a negative reward, $\nu \in (-1, 0)$. In Section 3, we explore and compare different values for ν, focusing on how they influence the properties of the emerging language. We define the reward function for the MRILG as:

$$R(\boldsymbol{x}, \hat{\boldsymbol{x}}, i) = \begin{cases} 1, & \text{if } \hat{\boldsymbol{x}} = \boldsymbol{x} \\ \nu, & \text{if } \hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}_{\text{idk}} \text{ and } i < N - 1 \\ -1, & \text{otherwise,} \end{cases}$$

Note that playing the *idk* action in the last round is the same as making a wrong prediction.

2.2.1 Agent Architectures

We now describe the architectures implemented for both agents, the Speaker and the Listener (Figure 1). We design the Speaker agent as an RL, where its network architecture is similar to that

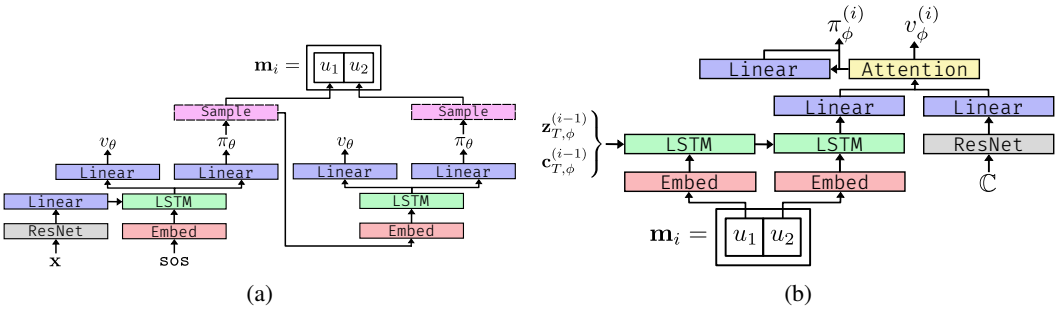


Figure 1: Graphical representation of Speaker, Figure 1a, and Listener, Figure 1b, architectures for the MRILG. In this illustration, the message, $\boldsymbol{m}_i$, contains only two tokens, $T = 2$.

of Chaabouni et al. (2022). In the case of the Listener, we define a novel RL architecture instead of the original Self-Supervised (SS) architecture.

As described in Section 2.1, the Speaker is agnostic to the current game's round i. For this reason, we will describe the Speaker's inference and architecture for one game round. As an overview, the Speaker's objective is to encode a discrete message, $\boldsymbol{m}$, describing an input image, $\boldsymbol{x}$. First, we encode the image using a pre-trained image encoder (Grill et al., 2020), f, to reduce its dimensionality and extract valuable features, $f(\boldsymbol{x})$. Subsequently, a trainable encoder g processes the new sequence of features outputting the initial hidden and cell values, $(\boldsymbol{z}_{0,\theta}, \boldsymbol{c}_{0,\theta}) = g(f(\boldsymbol{x}); \theta)$, used by the recurrent module h, an LSTM. Since f and g are deterministic, $\boldsymbol{z}_{0,\theta}$ and $\boldsymbol{c}_{0,\theta}$ always have the same values when providing the same input image, independently of the game round.

Subsequently, the Speaker will select each token to add to the message iteratively, using h. On this account, we define a complementary embedding module, e, to convert the previous discrete token u_{t-1} into a dense vector $\boldsymbol{d}_{t,\theta} = e(u_{t-1}; \theta)$. Then, the recurrent module, h, consumes the new dense vector and previous internal states to produce the new ones, $(\boldsymbol{z}_{t,\theta}, \boldsymbol{c}_{t,\theta}) = h(\boldsymbol{d}_{t,\theta}, z_{t-1,\theta}, c_{t-1,\theta}; \theta)$. We then pass $\boldsymbol{z}_{t,\theta}$ through two concurrent heads: (i) The actor head yields the probability of choosing each token as the next one, $u_t \sim \pi_S(\cdot|\boldsymbol{z}_{t,\theta})$; (ii) The critic head estimates the expected reward (value function approximation) $V(\boldsymbol{x}) := v(z_{t,\theta}; \theta)$. After the new token is sampled, we feed it back to $e(\cdot\,; \theta)$, and the process repeats itself until we generate T tokens. The first token u_0 is a predefined *start-of-string* token and is not included in the message. Following Chaabouni et al. (2022), we maintain the original vocabulary and message sizes, where $|\mathbb{W}| = 20$, and $T = 10$, which is vastly more extensive than the size of the dataset used $\left(|\mathbb{X}| \approx 10^6\right)$.

The implementation of the Speaker is agnostic to the current game round being played, meaning no information flows between rounds. In contrast, we focus on the listening side by intentionally adding history between rounds and analyzing if the Listener can lead the learning process to more general languages that can effectively play the MRILG.

The Listener architecture has two different modules to process the received message from the Speaker $\boldsymbol{m}$ and the images obtained as candidates $\mathbb{C}$. Additionally, a third module combines the output of both input components and provides it to two heads, actor and critic heads. We now describe each component in detail.

To process the candidate images $\mathbb{C}$, the Listener uses the same pre-trained encoder f combined with the network c to embed the candidates into a lower dimension, $\boldsymbol{l}_{\boldsymbol{x}_j} = c(f(\boldsymbol{x}_j); \phi)$, where $\boldsymbol{x}_j \in \mathbb{C}$.

Concerning the message received each round, $\boldsymbol{m}^{(i)}$, from the communication channel, the Listener uses the recurrent model h (an LSTM) to handle it by processing each token, $u_t^{(i)}$, iteratively. Similarly to the Speaker, there is an embedding layer, $e(\cdot\,; \phi)$, to convert the discrete token into a dense vector before giving it to the LSTM, where we have $(\boldsymbol{z}_{t,\phi}^{(i)}, \boldsymbol{c}_{t,\phi}^{(i)}) = h(e(u_t^{(i)}; \phi), \boldsymbol{z}_{t-1,\phi}^{(i)}, \boldsymbol{c}_{t-1}^{(i)}; \phi)$. The initial internal states of h_ϕ are initialized as $\boldsymbol{z}_{0,\phi}^{(0)} = \mathbf{0}$ and $\boldsymbol{c}_{0,\phi}^{(0)} = \mathbf{0}$ for the first round, and for the consecutive rounds are the final hidden and cell values of the previous round, *i.e.* $\boldsymbol{z}_{0,\phi}^{(i)} = \boldsymbol{z}_{T,\phi}^{(i-1)}$ and $\boldsymbol{c}_{0,\phi}^{(i)} = \boldsymbol{c}_{T,\phi}^{(i-1)}$. After processing all message tokens received in the current round, the fi-

nal hidden state, $\boldsymbol{z}_{T,\phi}^{(i)}$, is passed through a final network g to output the message's hidden value $\boldsymbol{l}_m^{(i)} = g(z_{T,\phi}^{(i)}; \phi)$. Finally, the generated hidden values for the message and all candidates flow through to the head module.

The first operation in the head module executes a straightforward attention mechanism to combine the obtained message features with each candidate's counterpart. The output includes a value per candidate which we concatenate into a single vector $\boldsymbol{s}^{(i)} = \begin{bmatrix} \boldsymbol{l}_m^{(i)} \cdot \boldsymbol{l}_{x_1} & \dots & \boldsymbol{l}_m^{(i)} \cdot \boldsymbol{l}_{x_{|\mathbb{C}|}} \end{bmatrix}^T$, called the candidates' score. For the actor head, $\boldsymbol{s}^{(i)}$ passes through a linear layer $t(\cdot\ ; \phi)$ to compute the score corresponding to the *idk* action, $s_{\text{idk}}^{(i)}$. Therefore, we define the Listener's policy as $\pi_L^{(i)}(\cdot|\boldsymbol{m}^{(i)}, \mathbb{C}) := \pi_L^{(i)}(\cdot|\boldsymbol{s}^{(i)}, s_{\text{idk}}^{(i)})$, which is a valid approximation since $\boldsymbol{s}^{(i)}$ holds information from the current message and candidates, and $s_{\text{idk}}^{(i)}$ has additional knowledge to compute the *idk* action. At the same time, the critic head $v(\cdot\ ; \phi)$ receives the same scores $\boldsymbol{s}^{(i)}$ and uses an MLP to estimate the expected cumulative reward for the current round i, as detailed in Section 2.2.2.

2.2.2 Learning Strategy

As described at the start of Section 2.2, the agents can only transmit information via the communication channel, which has only one direction: from the Speaker to the Listener. Additionally, agents learn how to communicate following the RIAL protocol, where agents are independent and treat others as part of the environment. As such, we have a decentralized training scheme where the agents improve their own parameters solely by maximizing the game's reward.

To perform well and consistently when playing the MRILG, the Speaker must learn how to utilize the vocabulary to distinctively encode each image into a message to obtain the highest expected reward possible. We use Reinforce (Williams, 1992), a policy gradient algorithm, to train the Speaker. Given a target image $\boldsymbol{x}$ and game round i with the corresponding Listener's action $\hat{\boldsymbol{x}}^{(i)}$, we have a loss, $L_A^{(i)}$, to fit the actor's head and another one, $L_C^{(i)}$, for the critic's head. We set $L_A^{(i)}(\theta) = -\sum_{t=1}^{T} \text{sg}(R(\boldsymbol{x}, \hat{\boldsymbol{x}}^{(i)}, i) - v(\boldsymbol{z}_{t,\theta}^{(i)}; \theta)) \cdot \log \pi_S(u_t^{(i)} | \boldsymbol{z}_{t,\theta}^{(i)})$, where $\text{sg}(\cdot)$ is the *stop-gradient* function, in order to optimize the policy. Regarding the critic loss, we set $L_C^{(i)}(\theta) = \sum_{t=1}^{T} (R(\boldsymbol{x}, \hat{\boldsymbol{x}}^{(i)}, i) - v(\boldsymbol{z}_{t,\theta}^{(i)}; \theta))^2$, to approximate the state-value function $V(\boldsymbol{x})$. We also use an additional entropy regularization term, $L_{\mathcal{H}}^{(i)}$, to make sure the language learned by the Speaker will not entirely stagnate by encouraging new combinations of tokens that increase entropy, further incentivizing exploration. Moreover, we define a target policy for the Speaker to minimize an additional KL divergent term, $L_{\text{KL}}^{(i)}$, between the online and target policies, θ and $\bar{\theta}$, respectively. We update $\bar{\theta}$ using an exponential moving average (EMA) over θ, *i.e.* $\bar{\theta} \leftarrow (1-\eta)\theta + \eta\bar{\theta}$ where η is the EMA weight parameter. With $L_{\text{KL}}^{(i)}$, we prevent steep changes in the parameter space, which helps stabilize training (Rawlik et al., 2012; Chane-Sane et al., 2021). We refer to Chaabouni et al. (2022) for a complete analysis on the impact of $L_{\text{KL}}^{(i)}$. Finally, we weigh each loss term and average the resulting sum for each input image in a batch, $\mathbb{X}' \subset \mathbb{X}$, and *effective* game round, $i \in \{1, \dots, I\}$, to obtain the overall Speaker loss: $L(\theta) = \frac{1}{|\mathbb{X}'|} \sum_{\hat{x} \in \mathbb{X}'} \frac{1}{I} \sum_{i=1}^{I} \alpha_{S,A} L_A^{(i)}(\theta) + \alpha_{S,C} L_C^{(i)}(\theta) + \alpha_{S,\mathcal{H}} L_{\mathcal{H}}^{(i)}(\theta) + \alpha_{S,\text{KL}} L_{\text{KL}}^{(i)}(\theta)$, where $\alpha_{S,A}$, $\alpha_{S,C}$, $\alpha_{S,\mathcal{H}}$, $\alpha_{S,\text{KL}}$ are constants, and the total effective game round, $I \in \{0, \dots, N-1\}$, is the number of rounds played by the pair of agents up to (and including) the first round the Listener plays a decisive action, meaning it tries to predict the correct candidate, $\hat{\boldsymbol{x}}^{(I)} \neq \hat{\boldsymbol{x}}_{\text{idk}}$.

We also use Reinforce to train the Listener agent. Since the Listener preserves state between rounds, we compute the expected cumulative reward as follows:

$$G(\boldsymbol{x}, \hat{\boldsymbol{x}}, i, I) = \sum_{j=i}^{I-1} \gamma^{j-i} R(\boldsymbol{x}, \hat{\boldsymbol{x}}_{\text{idk}}, j) + \gamma^{I-i} R(\boldsymbol{x}, \hat{\boldsymbol{x}}, I). \tag{2}$$

We define $L_A^{(i)}(\phi) = -\text{sg}(G_i(\boldsymbol{x}, \hat{\boldsymbol{x}}, i, I) - v(\boldsymbol{s}^{(i)}; \phi)) \cdot \log(\pi_L^{(i)}(\hat{\boldsymbol{x}}' | \boldsymbol{s}^{(i)}, s_{\text{idk}}^{(i)})$, where $\hat{\boldsymbol{x}}'$ is $\hat{\boldsymbol{x}}_{\text{idk}}$ if $i < I$, and $\hat{\boldsymbol{x}} \in \mathbb{C}$ otherwise. Additionally, we set $L_C^{(i)}(\phi) = (G_i(\boldsymbol{x}, \hat{\boldsymbol{x}}, i, I) - v(s^{(i)}; \phi))^2$, to train the critic head. Similarly to the Speaker loss, we add an entropy loss term $L_{\mathcal{H}}^{(i)}(\phi)$ to

encourage exploration. The final Listener loss for a batch of images and each game round is $L(\phi) = \frac{1}{|\mathbb{X}'|}\sum_{\hat{x}\in\mathbb{X}'}\frac{1}{I}\sum_{i=1}^{I}\alpha_{L,A}L_A^{(i)}(\phi) + \alpha_{L,C}L_C^{(i)}(\phi) + \alpha_{L,\mathcal{H}}L_{\mathcal{H}}^{(i)}(\phi)$, where $\alpha_{L,A}$, $\alpha_{L,C}$, and $\alpha_{L,\mathcal{H}}$ are constants. A detailed analysis of the learning strategy, for both agents, can be found in Appendix E.1.

Due to the complexity and non-stationarity of the environment, we found several requirements to be necessary to guide the training of both agents towards regions in the parameter space where viable communication protocols emerge, instead of being degenerate. Namely, we add a pre-train phase to MRILG, where we linearly increase the noise level in the communication channel from 0 to λ. This phase is optional and only helps with data efficiency (we refer to Appendix E.3 for more details). We also observe that the Listener benefits from having both actor and critic heads, where the interplay between the actor and critic loss terms is essential to effectively guide both agents' learning in an early training phase. For instance, having these two loss terms reduces the Listener policy's entropy substantially, meaning the Listener learns that there is only one correct action (one candidate) for each round and game. On the other hand, we show that when the Listener architecture comprises only an actor, the agents never agree on a communication protocol, where the training fails entirely. In Appendix E.4, we present an ablation study that corroborates these findings.

3 Evaluation

We provide an extensive evaluation of MRILG and variants. For completeness, we also consider, as a baseline, the original architecture proposed by Chaabouni et al. (2022) to play the original LG. Our model surpasses this baseline at a slight cost of data efficiency. This trade-off is expected and fully explained in Section 3.2.1. At a glance, this happens because the baseline version can access more information than our implementation, during training. We also present and evaluate several intermediary LG variants where we start at the original LG and incrementally modify it until we arrive at the MRILG. Having a progressive sequence of LG games enables us to assess how each environment modification influences the emergent communication protocol learned by the agents.

We continue this section by introducing all LG variants, giving a broader view of each game, agent architectures, and learning strategy. Next, we evaluate the generality of the emerging language for each game variant when providing new and unseen images.

Due to space constraints, we provide additional results in the appendix, where we evaluate all game variants in several transfer learning tasks, named *ETL*. This supplementary evaluation gives yet another frame of reference to evaluate the generality and robustness of the learned languages. Finally, we also refer to the appendix for further implementation details regarding every game variant (Appendix B), model architectures (Appendix D), and datasets used (Appendix E.2).

3.1 Lewis Game Variants

We briefly report essential aspects of each game variant, and Appendix B presents supplementary information. We consider four variants of the LG, all of which share the same Speaker architecture. The Listener architecture differs in all games. We refer to Appendix C for a comprehensive description of these architectures. Additionally, all variants except for LG (SS) are a contribution of this work. We now describe the LG variants considered:

- LG (SS): The original LG variant of Chaabouni et al. (2022). Here, the Listener is a Self-Supervised agent that tries to find similarities between the received message and the correct candidate through the InfoNCE loss (Oord et al., 2018; Dessi et al., 2021).
- LG (RL): We consider an RL Listener, trained using Reinforce (Williams, 1992). The architecture is similar to that in Section 2.2.1, without computing the *idk* action.
- NLG: In the NLG (which stands for Noisy Lewis Game), we apply an external environmental pressure by adding noise to the message during transmission. The agents' implementation and learning strategy is the same as in the LG (RL).
- MRILG: The most general and the target game introduced in Section 2.2. Note that this is the only game where the *idk* action has an extended impact on the game itself since it allows the agents to play another round.

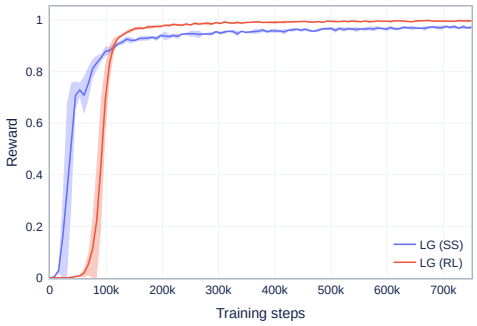


Figure 2: Accuracy during training of LG (SS) and LG (RL) with $|\mathbb{C}| = 1024$ on ImageNet dataset.

3.2 Robust Communication Protocols

This section thoroughly analyzes the performance of the different variants described above. Starting by comparing LG (SS) with LG (RL), we can see an apparent performance boost for the LG when implementing the Listener as an RL agent. Figure 2 shows that the RL version performs better than the SS version. Equivalent results occur in the testing phase (Table 1), where the RL version surpasses the accuracy achieved by the SS counterpart. From Figure 2, we also observe a trade-off between performance and sample efficiency, where the RL version is less sample efficient. We can trace these differences back to the loss function employed by each version. For instance, the SS version employs the InfoNCE loss, which we can see as a Reinforce variant with only a policy to optimize and, particularly, with access to an oracle, which gives information about which action (candidate) is the right one for each received message. As such, the Listener (SS) can efficiently learn how to map messages to the correct candidates. On the other hand, the RL version has no access to such an oracle and needs to interact with the environment to build this knowledge. The decrease in sample efficiency from SS to RL is, therefore, a natural phenomenon. Nonetheless, the RL version introduces a critic loss term whose synergy with the policy loss term helps to improve the final performance compared to the SS version.

One disadvantage of employing, at inference time, the communication protocols learned by playing default LG variants (LG (SS) and LG (RL)) is that they are not robust to deal with message perturbations. Since agents train only with perfect communication, they never experience noisy communication. The performances of SS and RL with noisy communication thus experience an extensive drop as shown in Table 2.

When introducing noise at training time (NLG), we place the agents in a more complex environment where only random fractions of the message are visible. Despite such modifications, they can still adapt to the environment and learn robust communication protocols that handle both types of messages (with and without noise). We notice equivalent accuracy performance for NLG and LG (RL) when testing with perfect communication channels and a significant boost when conducting tests with noisy communication channels (see Tables 1 and 2, respectively). These results indicate that adding noise at training time is an effective solution to learn more robust communication protocols.

3.2.1 Comparing Different Levels of Noise

Examining the results obtained from NLG and MRILG against LG (RL) when evaluating using a noiseless channel, we observe no loss in inference capabilities for any noise level applied during training, as shown in Table 3.

Additionally, since we force the evaluation game to have at most one round, we observe no clear advantage when training agents in the multi-round game against NLG. From Table 4, we observe that the test accuracy decreases as the noise increases. Additionally, when training with $\lambda = 0.25$, the performance loss in test accuracy is almost negligible compared to the results in Table 3. The agents can effectively sustain this noise threshold without losing performance. One possible reason for this behavior is that messages are overly lengthy, where even having just 75% of the tokens is still enough to encode useful information regarding the Speaker's input.

Table 1: Test accuracy with SD for different game variants, using ImageNet dataset and over 10 seeds. During training $|\mathbb{C}| = 1024$. During test λ_{test} is set to 0.

Game	λ	$\lvert\mathbb{C}\rvert$ (test)	
		1024	4096
LG (SS)	0	0.96 (0.02)	0.88 (0.04)
LG (RL)	0	0.99 (0.00)	0.97 (0.00)
NLG	0.5	0.99 (0.00)	0.97 (0.00)

Table 2: Test accuracy with SD for different game variants, using ImageNet dataset and over 10 seeds. During training $|\mathbb{C}| = 1024$. During test λ_{test} is set to 0.5.

Game	λ	$\lvert\mathbb{C}\rvert$ (test)	
		1024	4096
LG (SS)	0	0.05 (0.01)	0.03 (0.01)
LG (RL)	0	0.11 (0.02)	0.06 (0.01)
NLG	0.5	0.82 (0.01)	0.68 (0.02)

Table 3: Test accuracy with SD for different game variants, using ImageNet dataset and over 10 seeds. During test λ_{test} is set to 0.

Game	λ	ν	$\lvert\mathbb{C}\rvert$ (test)	
			1024	4096
NLG	0.25	-	0.99 (0.00)	0.97 (0.00)
NLG	0.5	-	0.99 (0.00)	0.97 (0.00)
NLG	0.75	-	0.98 (0.00)	0.94 (0.01)
MRILG	0.25	-0.2	0.99 (0.00)	0.97 (0.01)
MRILG	0.5	-0.2	0.99 (0.00)	0.97 (0.01)
MRILG	0.75	-0.2	0.98 (0.01)	0.94 (0.04)

Table 4: Test accuracy with SD for different game variants, using ImageNet dataset and over 10 seeds. During test λ_{test} is set to λ.

Game	λ	ν	$\lvert\mathbb{C}\rvert$ (test)	
			1024	4096
NLG	0.25	-	0.97 (0.00)	0.90 (0.01)
NLG	0.5	-	0.82 (0.01)	0.68 (0.02)
NLG	0.75	-	0.36 (0.01)	0.23 (0.01)
MRILG	0.25	-0.2	0.96 (0.01)	0.89 (0.02)
MRILG	0.5	-0.2	0.80 (0.01)	0.65 (0.02)
MRILG	0.75	-0.2	0.37 (0.01)	0.23 (0.01)

3.2.2 Comparing Different Rewards for the IDK Action

Focusing on the *idk* reward's impact, ν, we observe a clear advantage of using a lower value for the *idk* reward, see Tables 5 and 6. We observe similar results when the $\nu = \{-0.5, -0.2\}$. On the other hand, having ν close to 0 decreases test accuracy considerably. In addition, the training is unstable for this *idk* reward level since a considerably high variance appears in all results. As such, we discern that adding more cost to the *idk* action is a clear strategy to ensure the Listener does not exploit this action and only uses it when there is high uncertainty about the correct candidate.

3.2.3 Scaling the Number of Candidates

For every different game, we scale the number of candidates, $|\mathbb{C}|$, from 16 to 1024, using a ratio of 4. Looking at Tables 7 and 8, we can see an evident generalization boost when the number of candidates increases for every game. We posit that increasing the game's difficulty (increasing the number of candidates) helps the agents to generalize. As the candidates' set gets additional images, the input diversity increases, which affects how agents encode and interpret more information to distinguish the correct image from all others.

We note that as $|\mathbb{C}|$ increases, the test performance also increases, but at a smaller scale, e.g., the test gap (when at test time $|\mathbb{C}| = 4096$ candidates), between LG (RL) with $|\mathbb{C}| = 16$ and $|\mathbb{C}| = 64$ is 0.4 and only 0.03 between $|\mathbb{C}| = 256$ and $|\mathbb{C}| = 1024$. The noisy games exhibit the same behavior and have comparable results to LG (RL) despite the increased complexity in the environments.

4 Conclusion & Future Work

In this work, we focus on creating emergent and robust languages that can be safely employed at inference time since they are robust to perturbation in the communication channel. Following this

Table 5: Test accuracy with SD for different game variants, using ImageNet dataset and over 10 seeds. During test λ_{test} is set to 0.

Game	λ	ν	$\lvert\mathbb{C}\rvert$ (test)	
			1024	4096
MRILG	0.5	-0.5	0.99 (0.00)	0.96 (0.00)
MRILG	0.5	-0.2	0.99 (0.00)	0.97 (0.00)
MRILG	0.5	-0.05	0.79 (0.42)	0.72 (0.38)

Table 6: Test accuracy with SD for different game variants, using ImageNet dataset and over 10 seeds. During test λ_{test} is set to 0.5.

Game	λ	ν	$\lvert\mathbb{C}\rvert$ (test)	
			1024	4096
MRILG	0.5	-0.5	0.81 (0.01)	0.66 (0.02)
MRILG	0.5	-0.2	0.80 (0.01)	0.65 (0.02)
MRILG	0.5	-0.05	0.62 (0.33)	0.50 (0.26)

Table 7: Test accuracy with SD for different game variants, using ImageNet dataset and over 10 seeds. During test λ_{test} is set to 0.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	$\lvert\mathbb{C}\rvert$ (test)	
				1024	4096
LG (RL)	0	-	16	0.67 (0.04)	0.39 (0.04)
LG (RL)	0	-	64	0.93 (0.01)	0.79 (0.03)
LG (RL)	0	-	256	0.98 (0.00)	0.94 (0.01)
LG (RL)	0	-	1024	0.99 (0.00)	0.97 (0.00)
NLG	0.5	-	16	0.55 (0.03)	0.27 (0.02)
NLG	0.5	-	64	0.87 (0.01)	0.67 (0.03)
NLG	0.5	-	256	0.98 (0.00)	0.91 (0.01)
NLG	0.5	-	1024	0.99 (0.00)	0.97 (0.00)
MRILG	0.5	-0.5	16	0.56 (0.04)	0.29 (0.04)
MRILG	0.5	-0.5	64	0.90 (0.01)	0.72 (0.03)
MRILG	0.5	-0.5	256	0.98 (0.00)	0.92 (0.01)
MRILG	0.5	-0.5	1024	0.99 (0.00)	0.96 (0.00)

Table 8: Test accuracy with SD for different game variants, using ImageNet dataset and over 10 seeds. During test λ_{test} is set to 0.5.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	$\lvert\mathbb{C}\rvert$ (test)	
				1024	4096
LG (RL)	0	-	16	0.03 (0.01)	0.01 (0.01)
LG (RL)	0	-	64	0.09 (0.07)	0.05 (0.03)
LG (RL)	0	-	256	0.11 (0.06)	0.06 (0.04)
LG (RL)	0	-	1024	0.11 (0.02)	0.06 (0.01)
NLG	0.5	-	16	0.32 (0.02)	0.14 (0.01)
NLG	0.5	-	64	0.57 (0.02)	0.33 (0.02)
NLG	0.5	-	256	0.73 (0.01)	0.54 (0.02)
NLG	0.5	-	1024	0.82 (0.01)	0.68 (0.02)
MRILG	0.5	-0.5	16	0.33 (0.03)	0.14 (0.02)
MRILG	0.5	-0.5	64	0.59 (0.02)	0.35 (0.02)
MRILG	0.5	-0.5	256	0.75 (0.02)	0.55 (0.03)
MRILG	0.5	-0.5	1024	0.81 (0.01)	0.66 (0.02)

motivation, we extend the LG arriving at more complex variations to add robustness to the communication protocol. We add a noisy communication channel during training and enable a more elaborate interaction between agents by allowing the Listener to ask to play another round before making a final decision. These environment modifications and the introduction of a novel Listener architecture permit the emergence of robust languages to noise. These robust communication protocols perform similarly to the original LG using deterministic channels. Additionally, when using noisy channels, the robust communication protocols surpass the performance of the original LG counterparts.

As future work, we could evaluate different types of noise induced in the communication channel, like changing tokens for others of the same vocabulary, or even using an additional agent to control what tokens to hide. One could also employ the agents' architectures in other, more complex, environments with long-horizon rewards, where the Listener intentionally asks for another message from the speaker when evaluating which action to take. Additionally, as we emphasize in the last paragraph of Section 2.2.2, a deeper study is encouraged to understand the conditions necessary for the agents learn how to communicate and agree on a communication protocol.

REFERENCES

Ben Bogin, Mor Geva, and Jonathan Berant. Emergence of communication in an interactive world with consistent speakers. *arXiv preprint arXiv:1809.00549*, 2018.

Rahma Chaabouni, Eugene Kharitonov, Emmanuel Dupoux, and Marco Baroni. Anti-efficient encoding in emergent communication. *Advances in Neural Information Processing Systems*, 32, 2019.

Rahma Chaabouni, Eugene Kharitonov, Diane Bouchacourt, Emmanuel Dupoux, and Marco Baroni. Compositionality and generalization in emergent languages. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4427–4442, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.407. URL https://aclanthology.org/2020.acl-main.407.

Rahma Chaabouni, Florian Strub, Florent Altché, Eugene Tarassov, Corentin Tallec, Elnaz Davoodi, Kory Wallace Mathewson, Olivier Tieleman, Angeliki Lazaridou, and Bilal Piot. Emergent communication at scale. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=AUGBfDIV9rL.

Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. Goal-conditioned reinforcement learning with imagined subgoals. In *International Conference on Machine Learning*, pp. 1430–1440. PMLR, 2021.

Edward Choi, Angeliki Lazaridou, and Nando de Freitas. Compositional obverter communication learning from raw visual input. In *International Conference on Learning Representations*, 2018.

Roberto Dessi, Eugene Kharitonov, and Baroni Marco. Interpretable agent communication from scratch (with a generic visual processor emerging on the side). In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 26937–26949. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/e250c59336b505ed411d455abaa30b4d-Paper.pdf.

Katrina Evtimova, Andrew Drozdov, Douwe Kiela, and Kyunghyun Cho. Emergent communication in a multi-modal, multi-step referential game. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.

Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

Lukas Galke, Yoav Ram, and Limor Raviv. Emergent communication for understanding human language evolution: What's missing? In *Emergent Communication Workshop at ICLR 2022*, 2022.

Laura Graesser, Kyunghyun Cho, and Douwe Kiela. Emergent linguistic phenomena in multi-agent communication games. In *2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*, pp. 3700–3710. Association for Computational Linguistics, 2019.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Shangmin Guo, Yi Ren, Serhii Havrylov, Stella Frank, Ivan Titov, and Kenny Smith. The emergence of compositional languages for numeric concepts through iterated learning in neural agents. *arXiv preprint arXiv:1910.05291*, 2019.

Serhii Havrylov and Ivan Titov. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. *Advances in neural information processing systems*, 30, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

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

Emilio Jorge, Mikael Kågebäck, Fredrik D Johansson, and Emil Gustavsson. Learning to play guess who? and inventing a grounded language as a consequence. *arXiv preprint arXiv:1611.03218*, 2016.

Tomasz Korbak, Julian Zubek, and Joanna Rączaszek-Leonardi. Measuring non-trivial compositionality in emergent communication. *arXiv preprint arXiv:2010.15058*, 2020.

Łukasz Kuciński, Tomasz Korbak, Paweł Kołodziej, and Piotr Miłoś. Catalytic role of noise and necessity of inductive biases in the emergence of compositional communication. *Advances in Neural Information Processing Systems*, 34:23075–23088, 2021.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Hk8N3Sclg.

David Lewis. Scorekeeping in a language game. *Journal of Philosophical Logic*, 8(1):339–359, Jan 1979. ISSN 1573-0433. doi: 10.1007/BF00258436. URL https://doi.org/10.1007/BF00258436.

Fushan Li and Michael Bowling. Ease-of-teaching and language structure from emergent communication. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/b0cf188d74589db9b23d5d277238a929-Paper.pdf.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pp. 1310–1318. Pmlr, 2013.

David Premack and Guy Woodruff. Does the chimpanzee have a theory of mind? *Behavioral and brain sciences*, 1(4):515–526, 1978.

Shuwen Qiu, Sirui Xie, Lifeng Fan, Tao Gao, Jungseock Joo, Song-Chun Zhu, and Yixin Zhu. Emergent graphical conventions in a visual communication game. *Advances in Neural Information Processing Systems*, 35:13119–13131, 2022.

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. *Proceedings of Robotics: Science and Systems VIII*, 2012.

Yi Ren, Shangmin Guo, Matthieu Labeau, Shay B. Cohen, and Simon Kirby. Compositional languages emerge in a neural iterated learning model. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HkePNpVKPB.

Mathieu Rita, Rahma Chaabouni, and Emmanuel Dupoux. "LazImpa": Lazy and impatient neural agents learn to communicate efficiently. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pp. 335–343, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.conll-1.26. URL https://aclanthology.org/2020.conll-1.26.

Mathieu Rita, Florian Strub, Jean-Bastien Grill, Olivier Pietquin, and Emmanuel Dupoux. On the role of population heterogeneity in emergent communication. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=5Qkd7-bZfI.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. *Advances in neural information processing systems*, 29, 2016.

Mycal Tucker, Huao Li, Siddharth Agrawal, Dana Hughes, Katia Sycara, Michael Lewis, and Julie A Shah. Emergent discrete communication in semantic spaces. *Advances in Neural Information Processing Systems*, 34:10574–10586, 2021.

Ryo Ueda and Koki Washio. On the relationship between Zipf's law of abbreviation and interfering noise in emergent languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop*, pp. 60–70, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-srw.6. URL https://aclanthology.org/2021.acl-srw.6.

Nino Vieillard, Tadashi Kozuno, Bruno Scherrer, Olivier Pietquin, Remi Munos, and Matthieu Geist. Leverage the average: an analysis of kl regularization in reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 12163–12174. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/8e2c381d4dd04f1c55093f22c59c3a08-Paper.pdf.

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

George Kingsley Zipf. *The psycho-biology of language: An introduction to dynamic philology*, volume 21. Psychology Press, 1999.

George Kingsley Zipf. *The psycho-biology of language: An introduction to dynamic philology*. Routledge, 2013.

A Extended Related Work

One of the core properties of emergent communication is that agents learn a communication protocol on their own, where coordination is needed to solve a task. The study of emergent communication with neural agents started with continuous communication channels during training. (Sukhbaatar et al., 2016) propose a communication channel that shares continuous vectors where each agent receives a combination of all messages broadcasted by all other agents. Jorge et al. (2016) proposed a recurrent version of the Lewis Game with continuous messages. While such works have good performances, they benefit from having differential communication channels, making it possible for gradients to pass through. Havrylov & Titov (2017); Mordatch & Abbeel (2018); Guo et al. (2019); Chaabouni et al. (2020); Rita et al. (2022) used a formulation of the Lewis Game to study emergent communication, where messages also contain discrete tokens but still allow for gradients to pass through the communication channel. As such, these methods employ a DIAL paradigm (Foerster et al., 2016), allowing for an end-to-end training scheme (across agents) by sampling discrete tokens using straight-through Gumbel-Softmax estimator (Jang et al., 2016). From a natural language perspective, these approaches do not fully align with the properties of human communication, which is discrete and undifferentiable.

Other approaches, similarly to our work, close the gap to human language by using discrete channels to communicate at training and execution times, meaning gradients do not flow through the channel. In this case, we have a RIAL approach where each agent perceives others as part of the environment. In most cases, these works rely on a variation of Reinforce (Williams, 1992) to model the Speaker and Listener, where both agents try to maximize the game's reward. Primarily, Foerster et al. (2016); Lazaridou et al. (2017) developed discrete communication channels composed of only one symbol. The former study designed games where two agents simultaneously have the Speaker and Listener roles to classify images. Nevertheless, the architecture having independent agents performed poorly for the mentioned task, when compared to a DIAL approach. The latter work implemented a simpler version of the Lewis Game where the Listener discriminates between two images. The Speaker also has information about the images that the Listener will receive. Choi et al. (2018) further extended the latter game design creating agents that can handle messages composed of multiple discrete symbols, called the *Obverter* technique. The authors accomplish this by modeling the Speaker to choose the message that maximizes the Speaker's understanding. This assumption roots the theory of mind (Premack & Woodruff, 1978), where the Speaker assumes the Listener's mind and its own are identical. Although this work is an improvement from past works regarding the emulation of human language, there are still severe limitations. For example, the environment considered has only two effective degrees of freedom that must be modeled (shape and color of simple 3D shapes), limiting severily input diversity.

New extensions to the Obverter focus on studying specific properties of human language, like compositionality or pragmatics. Ren et al. (2020) developed an iterated learning strategy for the Lewis Game, trying to create highly compositional languages as a consequence of developing a new language protocol by having several distinct learning phases for each agent. From a linguistics perspective, compositionality is crucial since it encourages the expression of complex concepts through simpler ones. Another approach tries to leverage a population of agents to study its effect on simplifying the emerged language (Graesser et al., 2019). The authors use a simple visual task where agents communicate binary messages through a fixed number of rounds to match an image to a caption. The image depicts a simple shape with a specific color. Since each agent observes only part of the image, they must cooperate to solve the task. Li & Bowling (2019) also considers a variation of the Lewis Game with a discrete communication channel. The main objective is to give another perspective on how to evaluate the structure of the resulting communication protocol, giving experimental evidence that compositional languages are easier to teach to new Listeners. This additional external pressure surfaces when the Speaker interacts with new Listeners during training. The proposed experiments continue in the same line of simplicity since inputs are categorical values with two attributes, messages contain only two tokens, and the number of candidates the Listener discriminates is only five objects.

As a succeeding work, Chaabouni et al. (2022) proposed scaling several dimensions of the Lewis Game to create a setup closer to simulating human communication. Compared to the previous works, this study tries to scale several dimensions of the Lewis Game: the number of candidates received by the Listener, the dataset of images used, and the number of learning agents. Scaling

such dimensions makes the referential game more complex, promoting the generality and validity of the experimental results. Moreover, this study also suggests that compositionality is not a natural emergent factor of generalization of a language as the environment becomes more complex (e.g., using real-world images). This work can be seen as the initial starting point of our proposed work. In particular, we follow this game setup and extensively make more challenging environments where we add a time dependency and a stochastic communication channel. We argue that having a multi-step environment and a noisy communication channel enables learning more general and robust communication protocols.

There are few works in the literature that take into account noise in the communication channel. Tucker et al. (2021); Kuciński et al. (2021) apply a DIAL scheme, where the noise is sampled and added in the continuous space before applying the Gumbel-Softmax trick. As such, the problem is simplified by learning a continuous latent space robust to noise. More closely related to our work is the study presented by Ueda & Washio (2021). In such work, we are in an emergent discrete communication setting where the authors use a noisy communication channel to test the Zipf's law of abbreviation (Zipf, 2013). As such, the goal is to investigate conditions for which common messages become shorter. In this context, the authors propose an adversarial setting to try to deceive the Listener by giving another plausible message (different from the Speaker's message), which, as training progresses, promotes the usage of shorter messages. Our method differs from this work since we assume the communication channel can suffer external perturbations, mimicking the loss of information. In our case, we apply noise by changing the corresponding token to a pre-defined token called the *unknown* token. Therefore, we are interested in evaluating the communication protocols' robustness to handle different noise levels at test time. Additionally, our referential games are more complex since we use datasets with natural images for discrimination instead of categorical inputs.

Another particular extension related to our work is the introduction of multiple message transmissions before the Listener makes a final prediction (Evtimova et al., 2018; Qiu et al., 2022; Bogin et al., 2018). In particular, Evtimova et al. (2018) introduced a new referential game with a bidirectional communication channel. Even though the exchange of information between the Speaker and Listener happens in both directions, the amount of information is reduced since messages are binary vectors and have a fixed length. The Listener's task also adapts to this setting since only one candidate can be predicted in each game round. Compared to our proposed multi-round referential game (MRILG), we only allow information to flow from the Speaker to the Listener. However, the vocabulary we use comprises of discrete tokens, increasing combinatorially the amount of information transmitted. Further complexity also arises by having our Listener agent discriminate all candidates simultaneously in each game round played. Bogin et al. (2018) introduced environments where communication happens during long-horizon tasks based on grid worlds. To compensate for the stochasticity of the environment, especially at the beginning of the training phase, the authors proposed restricting the Speaker's learning to be more consistent in related conditions. By having a pre-trained scoring function for the environment's goal, the Speaker will have the additional objective of composing messages yielding a similar score to the goal score. Learning a new score for each new environment can be time-consuming, and this approach will also limit the Speaker's search space that can be explored when learning the communication protocol. In comparison to our work, we do not limit the exploration capability of the Speaker but force the Speaker to be agnostic to the current round since our main focus is to study how different Listener architectures impact the emergent language. Further, our input distribution is also more complex than in the mentioned work, since we use real-world images as input. Finally, we would like to point out that the work of Qiu et al. (2022) introduces a game with similar dynamics to our MRILG game, where at each round the Listener can decide to play another round or make a final prediction given the set of candidates. Nonetheless, there are still fundamental differences between both approaches. Both studies have different objectives since our games emulate discrete communication channels, and the authors' objective is to communicate drawings by sending continuous strokes. Complementarily, Qiu et al. (2022) simplify the coordination problem by using a DIAL approach. In contrast, our method uses a RIAL approach, where, in this case, the Speaker does not know that the message is being modified, complicating the coordination task.

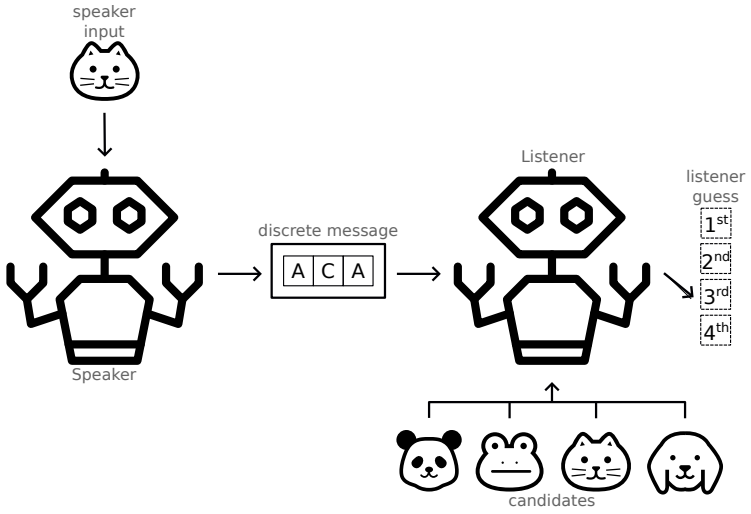


Figure 3: Visual Representation of the Lewis Game (LG).

B LEWIS GAME VARIANTS

For completeness and to have a self-contained study, we now give additional information regarding every LG variant implemented.

B.1 LEWIS GAME (LG)

The original LG is a discrimination game where a Speaker describes an image to the Listener by sending a message composed of discrete tokens. Afterward, the Listener uses the message to deliberate which image of the received set (candidates) was also the Speaker's input. We give a complete description of LG in Section 2.1. Furthermore, the Figure 3 depicts a visual representation of the LG.

B.2 NOISY LEWIS GAME (NLG)

NLG is an extension to the LG, where the communication channel stops being completely reliable, implying that messages can suffer random perturbations, see Figure 4. Similarly to the MRILG (Section 2.2), we define a noise function to convert each message's token into a predefined unknown token given some probability. Specifically, LG is a particular implementation of NLG where the noise level, λ in equation 1, is always 0.

B.3 MULTI-ROUND INDECISIVE LEWIS GAME (MRILG)

The final and most general game we present is the MRILG. This game further generalizes NLG, where agents can play multiple rounds before the Listener makes a final prediction. The Listener architecture attains a new action (*idk*) that allows the agents to play a new round instead of the Listener choosing a candidate, thus terminating the game in the latter case. A full description of MRILG appears in Section 2.2, and Figure 5 illustrates the game visually.

C LISTENER ARCHITECTURES

We now give more information detailing the agent architectures used for each LG variant. We reiterate that the Speaker architecture remains unchanged for all games. The only agent suffering modifications between games is the Listener, to be more precise, the Listener's head module. Hence, the base modules that process the message and candidates also remain unchanged between games. As Section 2.2.1 thoroughly explains, the token and recurrent modules ($e(\cdot\,;\phi)$ and $h(\cdot\,;\phi)$,

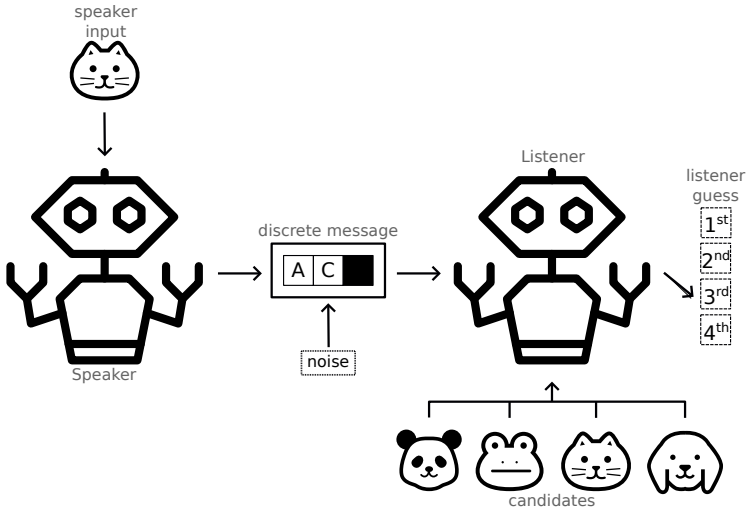


Figure 4: Visual Representation of the Noisy Lewis Game (NLG).

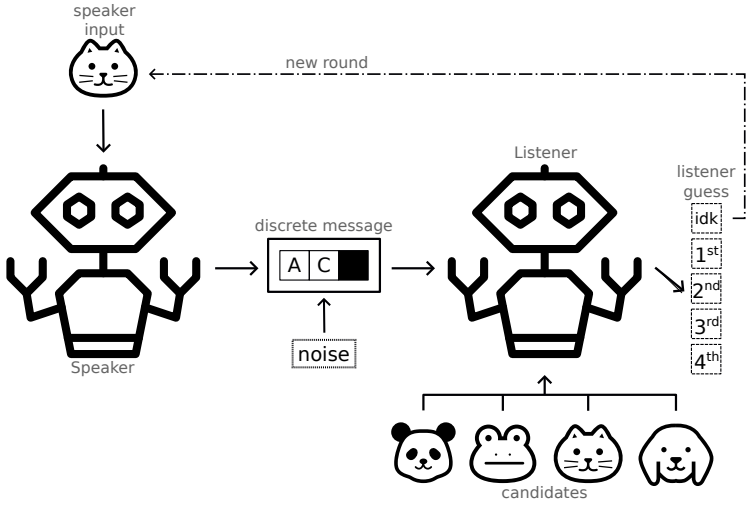


Figure 5: Visual Representation of the Multi-Round Indecisive Lewis Game (MRILG). The `idk` action in the Listener's guess stands for "I don't know".

respectively) convert each message token sequentially into a final hidden state $\boldsymbol{z}_{T,\phi}^{(i)}$. Regarding the candidates, a fixed image encoder, f, converts them into compact representations. Then, two different functions, g and c, further reduce the dimensionality of the message and candidate representations, respectively. Particularly, g and c use a non-linear activation function ($\tanh$) before returning the respective vectors, $\boldsymbol{l}_m^{(i)}$ and $\boldsymbol{l}_{\boldsymbol{x}_j}$. Finally, $\boldsymbol{l}_m^{(i)}$ and $\boldsymbol{l}_{\boldsymbol{x}_j}$ pass through an attention mechanism to output a single vector, $\boldsymbol{s}^{(i)} = \begin{bmatrix} \boldsymbol{l}_m^{(i)} \cdot \boldsymbol{l}_{x_1} & \dots & \boldsymbol{l}_m^{(i)} \cdot \boldsymbol{l}_{x_{|\mathbb{C}|}} \end{bmatrix}^T$, to the Listener head containing fused information from the message and each candidate.

In order to have a frame of reference to benchmark our novel games and agent architectures, we re-implemented the Listener architecture proposed by Chaabouni et al. (2022) (original), which we call LG (SS). In this case, the Listener employs a self-supervised (SS) architecture where the head module sends the attention logits through a softmax to convert them into a distribution. The corresponding learning procedure invokes the InfoNCE loss Oord et al. (2018) to attract the message representation to the representation of the right candidate while, at the same time, repulsing all other candidates. Therefore, the SS architecture can only play in one-round LG variants since it is not straightforward or natural to encode the *idk* action. Lastly, we disclose that our LG (SS) implemen-

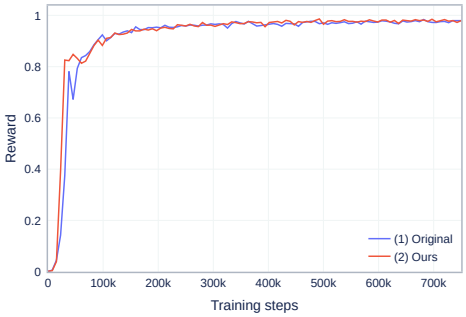


Figure 6: Accuracy during training of two implementations of LG (SS), (1) Original Chaabouni et al. (2022) and (2) Ours (Appendix C). The dataset used was ImageNet and $|\mathbb{C}| = 1024$.

Table 9: Test accuracy with SD for two implementations of LG (SS), (1) Original Chaabouni et al. (2022) and (2) Ours (Appendix C), using ImageNet dataset. During training $|\mathbb{C}| = 1024$.

Game	Implementation	$\|\mathbb{C}\|$ (test)	
		1024	4096
LG (SS)	Original (1)	0.96	0.88
LG (SS)	Ours (2)	0.96	0.88

tation differs slightly from the original implementation, where we place the $\tanh$ activation function before giving the message ($\boldsymbol{l}_m^{(i)}$) and candidate representations ($\boldsymbol{l}_{\boldsymbol{x}_j}$) to the attention mechanism. Figure 6 illustrates the training procedure for both implementations, where we can see similar data efficiency since both curves jump to a mean reward (accuracy) of 0.8 before the $50\,\mathrm{k}$ steps. Additionally, as the training step increases, both implementations converge to similar values (around 0.98). We also introduce test accuracies in Table 9, where both implementations also yield identical results.

Secondly, we implemented two novel Listener head architectures to accommodate the new proposed LG variants (LG (RL), NLG, and MRILG). All architectures differ only in the actor sub-module. As such, the critic sub-module remains unchanged, and its procedure approximates the cumulative discounted reward by supplying the attention logits, $\boldsymbol{s}^{(i)}$, to $v(\boldsymbol{s}^{(i)}\ ;\phi)$. The actor sub-module used to play LG (RL) and NLG uses the softmax function to convert the attention logits, $\boldsymbol{s}^{(i)}$, into a distribution containing the action to pick each candidate. Regarding the Listener head module implemented for the MRILG, there is an extra function, $t(\cdot\ ;\phi)$, to compute the logit corresponding to the *idk* action. See Section 2.2.1 for a fully detailed description of the actor module used for MRILG.

D Architecture Implementation

Succeeding the detailed explanation of the several Listener architectures applied for each LG variant (Appendix C), we present concrete network implementations for both agents in this section.

D.1 Speaker

The network implemented for the Speaker agent will receive as input an image and output a message composed of T discrete tokens, each retrieved from the same fixed vocabulary:

- f: A frozen ResNet-50 (He et al., 2016) model trained with BYOL algorithm (Grill et al., 2020) on ImageNet dataset (Russakovsky et al., 2015). The output size of the resulting features is 2048.
- $g(\cdot\ ;\theta)$: A single linear layer to reduce the features' dimensionality from 2048 to 512 in order to fit in the next layer, an LSTM (Hochreiter & Schmidhuber, 1997). Additionally, we divide the resulting vector into two equal parts, denoting the initial hidden, $\boldsymbol{z}_{0,\theta}$, and cell values ,$\boldsymbol{c}_{0,\theta}$, of the LSTM.
- $e(\cdot\ ;\theta)$: Embedding layer to convert discrete tokens into continuous vectors. The embedding layer receives the discrete token u_t, as an integer and with size $|\mathbb{W}| + 1$, and outputs a feature vector $e(u_t;\theta)$ of size 10.

- $h(\cdot\ ;\theta)$: Recurrent layer, implemented as a single LSTM of size 256. The initial hidden and cell states are the output of $g(\cdot\ ;\theta)$. Additionally, h processes the continuous version of each message token $e(u_{t-1};\theta)$ iteratively. Furthermore, the computed hidden value $\boldsymbol{z}_{t,\theta}$, in each iteration t, will be the input to the value and critics heads, $\pi_S(\cdot|\boldsymbol{z}_{t,\theta})$ and $v(\cdot\ ;\theta)$, respectively.
- $\pi_S(\cdot|\boldsymbol{z}_{t,\theta})$: For a given iteration t, the actor policy head uses a linear layer (output size of $|\mathbb{W}|$) followed by the softmax function to convert the hidden state $\boldsymbol{z}_{t,\theta}$ into a categorical distribution $\text{Cat}(|\mathbb{W}|, \pi_S(\cdot|\boldsymbol{z}_{t,\theta}))$, where each value indicates the probability of choosing each token as the next one to add to the message, u_t.
- $v(\cdot\ ;\theta)$: Critic value head to estimate the expected value, in this case, the reward received for the current round $R(\boldsymbol{x}, \hat{\boldsymbol{x}})$, by implementing a linear layer with an output size of 1. When playing the MRILG, there is a different reward for each round: $R(\boldsymbol{x}, \hat{\boldsymbol{x}}^{(i)}, i)$

D.2 LISTENER

The Listener architecture has two input entries to acquire the Speaker's message and the set of candidate images. This architecture suffers internal modifications to adapt to the specifications of the LG variant to play. Moreover, it is also possible to create different Listener architectures to solve the same game as we show in Appendix C, where we have SS and RL Listener architectures to play the LG. Hence, we first detail the implementation of common modules used in every Listener architecture. These sub-modules are the message and candidates' sub-modules used to extract information from the message and candidates, respectively. Afterward, we detail the sub-module specific to each different Listener, labeled as the ***Listener's head***. As a brief remark, we omit superscripts describing the round index from all variables since all games, except MRILG, have a duration of only one round, and extending to the multi-round case is straightforward.

D.2.1 COMMON SUB-MODULES

We describe every network implementation contained in the common Listener's sub-modules (message and candidates sub-modules):

- f ***(candidates)***: A frozen ResNet-50 trained with BYOL on the ImageNet dataset, where f converts the image into a vector with 2048 features. f is the same network used in the Speaker architecture, see Appendix D.1.
- $c(\cdot\ ;\phi)$ ***(candidates)***: This single linear layer, followed by the $\tanh$ function, receives each candidate's feature vector $\boldsymbol{l}_{\boldsymbol{x}_i} = c(f(\boldsymbol{x}_i);\phi)$ to reduce its dimensionality from 2048 to 256.
- $e(\cdot\ ;\phi)$ ***(message)***: A single embedding layer to convert a discrete message token u_t into a continuous vector of size 10. $e(\cdot\ ;\phi)$ is equal to the Speaker's layers $e(\cdot\ ;\theta)$ but parametrized with a different set of parameters, ϕ.
- $h(\cdot\ ;\phi)$ ***(message)***: An LSTM layer with a size of 512, receiving the continuous featuers of each message token $e(u_t;\phi)$ iteratively. The initial hidden $\boldsymbol{z}_{0,\phi}$ and cell states $\boldsymbol{c}_{0,\phi}$ are both $\mathbf{0}$.
- $g(\cdot\ ;\phi)$ ***(message)***: Linear layer followed by the $\tanh$ function to reduce the dimensionality of the last hidden state value $\boldsymbol{z}_{T,\phi}$ from a vector size 512 to 256, denoted as $\boldsymbol{l}_m$.

D.2.2 LG (SS)

Upon the message and candidate sub-modules outputting the respective hidden values, $\boldsymbol{l}_m$ and $\boldsymbol{l}_{\boldsymbol{x}_j}$, respectively, the SS Listener architecture computes a single value score between $\boldsymbol{l}_m$ and each $\boldsymbol{l}_{x_j}$, using the cosine similarity function (non-parameter function), $\text{cos\_sim} = \boldsymbol{l}_m \cdot \boldsymbol{l}_{\boldsymbol{x}_j} / (\|\boldsymbol{l}_m\| \|\boldsymbol{l}_{\boldsymbol{x}_j}\|)$. Following this step, the similarity outputs pass through a softmax function, yielding a categorical distribution suitable to apply InfoNCE loss (Oord et al., 2018).

D.2.3 LG (RL) & NLG

The Listener's head architecture for both LG (RL) and NLG has the same structure:

- $\boldsymbol{s}$: Non-parametrizable function to combine information coming from the received message and candidate set. The attention mechanism s happens through a dot product between $\boldsymbol{l}_m$, and each $\boldsymbol{l}_{\boldsymbol{x}_j}$, defined as $\boldsymbol{s}_= \begin{bmatrix} \boldsymbol{l}_m \cdot \boldsymbol{l}_{\boldsymbol{x}_1} & \dots & \boldsymbol{l}_m \cdot \boldsymbol{l}_{\boldsymbol{x}_{|\mathbb{C}|}} \end{bmatrix}^T$.
- $\pi_L(\cdot|\boldsymbol{s})$: π_L corresponds to the Listener's head actor head and contains the softmax function to convert the logits $\boldsymbol{s}$ into a categorical distribution Cat$(|\mathbb{C}|, \pi_L(\cdot|\boldsymbol{s}))$. Each distribution value maps the probability of choosing the candidate with the same index $\pi_L(i|\boldsymbol{s}_\phi))$, where $i \in \{1, \dots, |\mathbb{C}|\}$.
- $v(\cdot\,;\phi)$: The Listener's critic head v receives the attention logits $\boldsymbol{s}$ and passes them through a multi-layer perceptron (MLP), outputting a one-dimension value corresponding to a prediction of the reward $R(\boldsymbol{x}, \hat{\boldsymbol{x}})$ for the associated game. The hidden and output sizes of the MLP are $(\max(|\mathbb{C}|/4, 4), \max(|\mathbb{C}|/16, 2), 1)$, and the activation function used in the hidden layers is the ReLU function.

D.2.4 MRILG

For the multi-round game (MRILG), the Listener's head architecture is slightly different. In this case, we also need to model the additional *idk* action to explicitly encode the ability to keep playing the game through succeeding rounds. Thus, we slightly modify the Listener's actor head π_L where we add a linear layer, $t(\cdot\,;\phi)$, to receive $\boldsymbol{s}^{(i)}$ and output a single logit associated with the *idk* action, $s^{(i)}_{\text{idk}}$. As such, the softmax function receives the resulting concatenation of $\boldsymbol{s}^{(i)}$ with $s^{(i)}_{\text{idk}}$ and returns a categorical distribution Cat$(|\mathbb{C}| + 1, \pi_L(\cdot|\boldsymbol{s}^{(i)}, s^{(i)}_{\text{idk}}))$. The new distribution models all possible Listener actions, choosing a candidate or playing the *idk* action. The critic head has the same architecture but now estimates the expected cumulative reward from the current round i until the end of the game, see equation 2.

Additionally, we also slightly adjust $h(\cdot\,;\phi)$ to ensure dependencies between rounds and allow the Listener to have access to information from prior rounds. In this case, the initial hidden and cell values, $\boldsymbol{z}^{(i)}_{0,\phi}$ and $\boldsymbol{c}^{(i)}_{0,\phi}$, where $i > 0$, are not $\mathbf{0}$, but have the final values obtained during the previous round, $\boldsymbol{z}^{(i-1)}_{T,\phi}$ and $\boldsymbol{c}^{(i-1)}_{T,\phi}$, respectively.

D.3 Hyperparameters

In this subsection, we detail the hyperparameters used to instantiate all LG experiments, exposed in Tables 10 to 12.

E Training Details

Following the concise introduction about the learning strategy of both agents (Section 2.2.2), we now add complementary information detailing the loss functions used by both agents. We also propose an ablation study regarding essential architectural and training procedure choices given at the end of Section 2.2.2.

E.1 Learning Strategy

We model both agents as RL agents for the novel LG variants proposed in this study, LG (RL), NLG, and MRILG. We also follow a RIAL procedure (Foerster et al., 2016), where each agent perceives others as part of the environment, meaning no gradients flow between agents allowing us to isolate the loss function of each agent completely. We will now detail, first, the Speaker's loss function and, secondly, the loss function used by the Listener agent.

E.1.1 Speaker

Due to the Speaker's role, its objective will converge on creating messages $\boldsymbol{m} = (u_t)_{t=1}^T$ in such a way as to facilitate the mapping between the message and the right candidate ($\hat{\boldsymbol{x}} = \boldsymbol{x}$) by the Listener. For a given round i, the Speaker, parametrized by θ, generates messages iteratively, where the following message token results from sampling its actor's stochastic policy,

Table 10: Common hyperparameters for all games: LG (SS), LG (RL), NLG, and MRILG. The values of hyperparameters described as a set, $\{\cdot\}$, means we run experiments with each value in the set.

hyperparameter	value
training steps	$750\,\mathrm{k}$
$\lvert\mathbb{C}\rvert$	$\{16, 64, 256, 1024\}$
$\lvert\mathcal{W}\rvert$	20
T	10
$\alpha_{S,A}$	1
$\alpha_{S,C}$	1
$\alpha_{S,\mathcal{H}}$	1×10^{-4}
$\alpha_{S,A}$	0.5
$\alpha_{L,A}$	1
$\alpha_{L,C}$	1×10^{-3}
$\alpha_{L,\mathcal{H}}$	1×10^{-4}
Speaker's optim	adam
Listener's optim	adam
Speaker's optim lr	1×10^{-4}
Listener's optim lr	5×10^{-5}
η	0.99
γ	0.99

Table 11: Hyperparameters exclusive to NLG and MRILG. The values of hyperparameters described as a set, $\{\cdot\}$, means we run experiments with each value in the set.

hyperparameter	value
λ_{init}	0
λ	$\{0.25, 0.5, 0.75\}$
noise init steps	$300\,\mathrm{k}$

$u_t^{(i)} \sim \pi_S(\cdot|\boldsymbol{x}, (u_{t'}^{(i)})_{t'=1}^{t-1})$, conditioned on the target input image $\boldsymbol{x}$ and previously sampled tokens $(u_{t'}^{(i)})_{t'=1}^{t-1}$. As a result, the Speaker's goal attempts to find the best policy to maximize the expected reward:

$$J(\theta) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}(\mathbb{X})}\left[\mathbb{E}_{\pi_S(\cdot|\boldsymbol{x})}\left[R(\boldsymbol{x}, \hat{\boldsymbol{x}}, i)\right]\right],$$

where $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}(\mathbb{X})}$ considers the expectation for $\boldsymbol{x}$ over the dataset $\mathbb{X}$, and $\mathbb{E}_{\boldsymbol{x},\pi_S}$ is over all possible outputs generated by π_S, in this case, all sequences of messages $\boldsymbol{m}$, for each $\boldsymbol{x}$. Also, note that the Speaker is agnostic to the current game round being played, meaning rounds are independent, where π_S will only optimize the reward obtained for the current round, i, $R\left(\boldsymbol{x}, \hat{\boldsymbol{x}}^{(i)}, i\right)$ where $\hat{\boldsymbol{x}}^{(i)} = \hat{\boldsymbol{x}}_{\text{idk}} \vee \hat{\boldsymbol{x}}^{(i)} \in \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{|\mathbb{C}|}\}$. For one-round games, $R(\boldsymbol{x}, \hat{\boldsymbol{x}}, i) = R(\boldsymbol{x}, \hat{\boldsymbol{x}})$.

By assessing a particular instance of an LG variant game, it is possible to define the expected reward, given a target image $\boldsymbol{x}$ and game round i, as the value $V^{\pi_S}(\boldsymbol{x}, \hat{\boldsymbol{x}}, i) = \mathbb{E}_{\boldsymbol{x},\pi_S}\left[R(\boldsymbol{x}, \hat{\boldsymbol{x}}, i)\right]$. Using the policy gradient theorem, we can derive the gradient for the policy π_S as:

$$\partial_\theta V^{\pi_S}\left(\boldsymbol{x}, \hat{\boldsymbol{x}}^{(i)}, i\right) = \mathbb{E}_{\boldsymbol{x},\pi_S}\left[\sum_{t=1}^{T} R\left(\boldsymbol{x}, \hat{\boldsymbol{x}}^{(i)}, i\right) \partial_\theta \log \pi_S\left(u_t^{(i)}|\boldsymbol{x}, \left(u_{t'}^{(i)}\right)_{t'=1}^{t-1}\right)\right]. \tag{3}$$

As we can see in Equation 3, we set the discounted reward as $R(\boldsymbol{x}, \hat{\boldsymbol{x}}, i)$. This is true when the Speaker's discount factor, γ_S, is set to 1, which is our case. Additionally, this is also a valid assumption since messages have a fixed length where only the final Speaker's state (a message with T tokens) has an implication in the game, since it is when the message is sent to the Listener. As such, the Speaker only wants to maximize this final reward without considering delay through time. Moreover, to reduce the overall variance, we also subtract a baseline to $R(\boldsymbol{x}, \hat{\boldsymbol{x}}, i)$. In this case, the

Table 12: Hyperparameters exclusive to MRILG. The values of hyperparameters described as a set, $\{\cdot\}$, means we run experiments with each value in the set.

hyperparameter	value
N	5
ν	$\{-0.5, -0.2, -0.05\}$

policy gradient becomes:

$$\partial_\theta V^{\pi_S}\left(\boldsymbol{x}, \hat{\boldsymbol{x}}^{(i)}, i\right) = \mathbb{E}_{\boldsymbol{x},\pi_S}\left[\sum_{t=1}^{T}\left(R\left(\boldsymbol{x}, \hat{\boldsymbol{x}}^{(i)}, i\right) - V_{t-1}^{\pi_S}\left(\boldsymbol{x}, \hat{\boldsymbol{x}}^{(i)}, i\right)\right)\partial_\theta \log \pi_S\left(u_t^{(i)}|\boldsymbol{x}, \left(u_{t'}^{(i)}\right)_{t'=1}^{t-1}\right)\right], \quad (4)$$

where $V_{t-1}^{\pi_S}(\boldsymbol{x}, \hat{\boldsymbol{x}}^{(i)}, i) = \mathbb{E}_{\boldsymbol{x},\pi_S}[R(\boldsymbol{x}, \hat{\boldsymbol{x}}^{(i)}, i)|(u_{t'}^{(i)})_{t'=1}^{t-1}]$ contains the value currently conditioned on information gathered until timestep $t-1$.

To approximate equation 4, the Speaker's learning strategy will minimize the following two losses. First, the critic's head $v(\boldsymbol{z}_{t,\theta}^{(i)}; \theta)$ will adjust towards matching $V_{t-1}^{\pi_S}\left(\boldsymbol{x}, \hat{\boldsymbol{x}}^{(i)}, i\right)$ by minimizing the critic loss $L_C^{(i)}\left(\boldsymbol{x}, \hat{\boldsymbol{x}}^{(i)}; \theta\right)$:

$$L_C^{(i)}(\theta) = \frac{1}{|\mathbb{X}|}\sum_{x\in\mathbb{X}}\sum_{t=1}^{T}\left(R\left(\boldsymbol{x}, \hat{\boldsymbol{x}}^{(i)}, i\right) - v\left(\boldsymbol{z}_{t,\theta}^{(i)}; \theta\right)\right)^2.$$

Secondly, the actor's policy loss $L_A^{(i)}(\theta)$ targets minimizing the negative of the expected reward:

$$L_A^{(i)}(\theta) = -\frac{1}{|\mathbb{X}|}\sum_{x\in\mathbb{X}}\sum_{t=1}^{T}\text{sg}\left(R\left(\boldsymbol{x}, \hat{\boldsymbol{x}}^{(i)}, i\right) - v\left(\boldsymbol{z}_{t,\theta}^{(i)}; \theta\right)\right)\log\left(\pi_S\left(u_t^{(i)}|\boldsymbol{z}_{t,\theta}^{(i)}\right)\right),$$

where sg is the *stop-gradient* function. Additionally, the Speaker's actor head $\pi_S(\cdot|\boldsymbol{z}_{t,\theta}^{(i)})$ estimates $\pi_S(\cdot|\boldsymbol{x}, (u_{t'}^{(i)})_{t'=1}^{t})$, which is a valid approximation since the conditioned variable, $\boldsymbol{z}_{t,\theta}^{(i)}$, jointly encodes information about the target image $\boldsymbol{x}$ and the tokens already present in the message $(u_{t'}^{(i)})_{t'=1}^{t-1}$ due to its recurrent nature.

Furthermore, as Section 2.2.2 explains, the Speaker considers two additional loss terms to optimize together with equation 4. One such supplementary loss term is the KL divergence, $L_{\text{KL}}^{(i)}(\theta)$, which aims to minimize the relative entropy between the actor's policy π_S and a target version of this policy $\overline{\pi}_S$, computed as an EMA, $\overline{\theta} \leftarrow (1-\eta)\theta + \eta\overline{\theta}$, where η is a constant. As shown in previous studies (Schulman et al., 2017; Vieillard et al., 2020), using an additional KL loss term helps achieve better performance and, especially, stabilizes training which considerably helps in our case. We define $L_{\text{KL}}^{(i)}(\theta)$ as:

$$L_{\text{KL}}^{(i)}(\theta) = \frac{1}{|\mathbb{X}|}\sum_{x\in\mathbb{X}}\sum_{t=1}^{T}\mathbb{E}_{u\sim\pi_S\left(\cdot|\boldsymbol{z}_{t,\theta}^{(i)}\right)}\left[\pi_S\left(u|\boldsymbol{z}_{t,\theta}^{(i)}\right)\log\frac{\pi_S\left(u|\boldsymbol{z}_{t,\theta}^{(i)}\right)}{\overline{\pi}_S\left(u|\boldsymbol{z}_{t,\overline{\theta}}^{(i)}\right)}\right].$$

The other additional loss term is an entropy loss term $L_{\mathcal{H}}(\theta)$, where the objective passes to increase the actor's policy entropy to incentivize the exploration of new actions (tokens to create the message). Given π_S, the entropy loss term minimizes the following negative sampled version of the entropy:

$$L_{\mathcal{H}}^{(i)}(\theta) = \frac{1}{|\mathbb{X}|}\sum_{x\in\mathbb{X}}\sum_{t=1}^{T}\mathbb{E}_{u\sim\pi_S\left(\cdot|\boldsymbol{z}_{t,\theta}^{(i)}\right)}\left[\pi_S\left(u|\boldsymbol{z}_{t,\theta}^{(i)}\right)\log\pi_S\left(u|\boldsymbol{z}_{t,\theta}^{(i)}\right)\right].$$

E.1.2 LISTENER

Looking at the learning strategy for the Listener agent, we first emphasize that, in the case of the MRILG, future rounds depend on past ones. Therefore, it is crucial to model the Listener's reasoning as a sequential decision problem involving all game rounds to ensure information can flow between them, giving the Listener more complete information and helping it decide the best course of action. In this setting, the Listener's actor policy π_L depends on all messages received until the current round, i, $\boldsymbol{M}^{(i)} = (\boldsymbol{m}^{(i')})_{i'=1}^{i}$ and the candidates set $\mathbb{C} \subset \mathbb{X}$. Note also that the Listener intrinsically knows that all its previous actions were the *idk* action since this is the only way for the Listener to play another game round. As a result, the Listener's goal passes to maximize the expected reward given all rounds of the game:

$$\begin{aligned} J(\phi) &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}(\mathbb{X}),\mathbb{C}\sim\mathcal{U}(\mathbb{X}),\boldsymbol{M}^{(N-1)}\sim\pi_L}\left[\mathbb{E}_{\mathbb{C},\boldsymbol{M}^{(N-1)},\pi_L}\left[G_0\left(\boldsymbol{x},\hat{\boldsymbol{x}}\right)\right]\right] \\ &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}(\mathbb{X}),\mathbb{C}\sim\mathcal{U}(\mathbb{X}),\boldsymbol{M}^{(N-1)}\sim\pi_S}\left[\mathbb{E}_{\mathbb{C},\boldsymbol{M}^{(N-1)},\pi_L}\left[\sum_{i=0}^{N-1}\gamma^i R\left(\boldsymbol{x},\hat{\boldsymbol{x}},i\right)\right]\right], \end{aligned} \tag{5}$$

where the expectation $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{U}(\mathbb{X}),\mathbb{C}\sim\mathcal{U}(\mathbb{X}),\boldsymbol{M}^{(N-1)}\sim\pi_S(\cdot|\boldsymbol{x})}$ considers the space containing all possible combinations of the target image $\boldsymbol{x}$ sampled from a discrete uniform distribution over the training dataset $\mathcal{U}(\mathbb{X})$, set of candidates $\mathbb{C}$ also sampled independently from the same distribution $\mathcal{U}(\mathbb{X})$, and sequences of messages $\boldsymbol{M}^{(N-1)}$ each generated by the Speaker's policy π_S. Note, that $\mathbb{C} \sim \mathcal{U}(\mathbb{X})$ is an abuse of notation (used for readability), where in reality, we have $\mathbb{C}'_1, \ldots, \mathbb{C}'_{|\mathbb{C}|-1} \overset{\text{i.i.d.}}{\sim} U(\mathbb{X})$ and $\mathbb{C} = \{\boldsymbol{x}\} \cup \mathbb{C}'$. Similarly, $\boldsymbol{M}^{(N-1)} \sim \pi_S$ is a simplified notation for $\boldsymbol{m}'^{(i)} = \text{noise}(\boldsymbol{m}^{(i)}(\boldsymbol{x};\theta)) = (n_t(u_t^{(i)}))_{t=1}^{T}$, where $\boldsymbol{m}'^{(i)} \in \boldsymbol{M}^{(N-1)}$, $u_t^{(i)} \sim \pi_S(\cdot|\boldsymbol{x}, (u_{t'}^{(i)})_{t'=1}^{t-1})$, and n_t is defined in equation 1. The second expectation $\mathbb{E}_{\mathbb{C},\boldsymbol{M}^{(N-1)},\pi_L}$ of equation 5 is over all possible actions for the listener's policy when fixing the target image $\boldsymbol{x}$, candidates $\mathbb{C}$ and message sequence $\boldsymbol{M}^{(N-1)}$. Moreover, focusing on a particular occurrence of an LG variant, we define $I \in \{0, \ldots, N-1\}$ as the number of effective rounds played, e.g., if the Listener plays the *idk* action in the first two rounds and then makes a final prediction (choosing a candidate), then $I = 2$. By fixing I, $\boldsymbol{M}^{(I)}$, and $\mathbb{C}$, it is possible to define the value $V^{\pi_L}(\boldsymbol{x}, \boldsymbol{M}^{(I)}, \mathbb{C})$ as:

$$V^{\pi_L}\left(\boldsymbol{x},\mathbb{C},\boldsymbol{M}^{(I)}\right) = \mathbb{E}_{\mathbb{C},\boldsymbol{M}^{(I)},\pi_L}\left[\sum_{i=0}^{I}\gamma^i R\left(\boldsymbol{x},\hat{\boldsymbol{x}},i\right)\right].$$

Retaking the policy gradient theorem, we derive:

$$\begin{aligned} &\partial_\theta V^{\pi_L}\left(\boldsymbol{x},\hat{\boldsymbol{x}},\mathbb{C},\boldsymbol{M}^{(I)}\right) = \\ &\quad \mathbb{E}_{\pi_L}\left[\sum_{i=0}^{I-1}\left(G\left(\boldsymbol{x},\hat{\boldsymbol{x}}_{\text{idk}},i,I\right) - V^{\pi_L}\left(\boldsymbol{x},\mathbb{C},\boldsymbol{M}^{(i-1)}\right)\right)\partial_\theta \log\left(\pi_L\left(\hat{\boldsymbol{x}}_{\text{idk}}|\boldsymbol{M}^{(i)},\mathbb{C}\right)\right)\right. \\ &\qquad \left. + \left(G\left(\boldsymbol{x},\hat{\boldsymbol{x}},I,I\right) - V^{\pi_L}\left(\boldsymbol{x},\mathbb{C},\boldsymbol{M}^{(I-1)}\right)\right)\partial_\theta \log\left(\pi_L\left(\hat{\boldsymbol{x}}|\boldsymbol{M}^{(i)},\mathbb{C}\right)\right)\right], \end{aligned}$$

where $V^{\pi_L}\left(\boldsymbol{x},\mathbb{C},\boldsymbol{M}^{(i-1)}\right) = \mathbb{E}_{\pi_L}[R(\boldsymbol{x},\hat{\boldsymbol{x}}_{\text{idk}},i)|\boldsymbol{M}^{(i-1)},\mathbb{C}]$ encodes the value conditioned on all previous information until the previous round $i-1$. The action $\hat{\boldsymbol{x}}_{\text{idk}}$ corresponds to choosing the *idk* action, which happens for every round i, where $i < I$. Additionally, Equation 2 G.

Similarly to the Speaker, the main objective of the Listener's training encompasses minimizing a critic (value) loss $L_C(\phi)$ and a actor's policy loss term $L_A(\phi)$. Regarding the value loss term $L_C(\phi)$, the function takes the form:

$$L_C(\phi) = \frac{1}{I}\sum_{i=0}^{I}\left(G\left(\boldsymbol{x},\hat{\boldsymbol{x}},i,I\right) - v\left(\boldsymbol{s}^{(i)};\phi\right)\right)^2,$$

where the Listener's critic head $v(\boldsymbol{s}^{(i)};\phi)$ fits $V^{\pi_L}\left(\boldsymbol{x},\mathbb{C},\boldsymbol{M}^{(i-1)}\right)$. Additionally, to derive the original policy $\pi_L(\cdot|\boldsymbol{M}^{(i)},\mathbb{C})$, the Listener's actor head $\pi_L(\cdot|\boldsymbol{s}^{(i)}, s_{\text{idk}}^{(i)})$ targets the minimization of

the negative of the expected rewards:

$$L_A(\phi) = -\frac{1}{I}\left[\sum_{i=0}^{I-1} \text{sg}\left(G\left(\boldsymbol{x}, \hat{\boldsymbol{x}}_{\text{idk}}, i, I\right) - v\left(\boldsymbol{s}^{(i)}; \phi\right)\right) \log \pi_L\left(\hat{\boldsymbol{x}}_{\text{idk}} | \boldsymbol{s}^{(i)}, s_{\text{idk}}^{(i)}\right) + \text{sg}\left(G\left(\boldsymbol{x}, \hat{\boldsymbol{x}}, I, I\right) - v\left(\boldsymbol{s}^{(I)}; \phi\right)\right) \log \pi_L\left(\hat{\boldsymbol{x}} | \boldsymbol{s}^{(I)}, s_{\text{idk}}^{(I)}\right)\right],$$

where $\boldsymbol{s}^{(i)}$ encodes information from all previous messages and the candidates set, and $s_{\text{idk}}^{(i)}$ considers information about the *idk* action that also depends on $\boldsymbol{s}^{(i)}$. At last, we also add an entropy loss term $L_{\mathcal{H}}^{(i)}(\phi)$ to the Listener's loss to prevent early stagnation for specific actions:

$$L_{\mathcal{H}}^{(i)}(\phi) = \frac{1}{I}\left[\sum_{i=0}^{I-1} \mathbb{E}_{\hat{\boldsymbol{x}}_{\text{idk}} \sim \pi_L\left(\cdot | \boldsymbol{s}^{(i)}, s_{\text{idk}}^{(i)}\right)}\left[\pi_L\left(\hat{\boldsymbol{x}}_{\text{idk}} | \boldsymbol{s}^{(i)}, s_{\text{idk}}^{(i)}\right) \log \pi_L\left(\hat{\boldsymbol{x}}_{\text{idk}} | \boldsymbol{s}^{(i)}, s_{\text{idk}}^{(i)}\right)\right] + \mathbb{E}_{\hat{\boldsymbol{x}} \sim \pi_L\left(\cdot | \boldsymbol{s}_{\phi}^{(I)}, s_{\text{idk}}^{(I)}\right)}\left[\pi_L\left(\hat{\boldsymbol{x}} | \boldsymbol{s}^{(I)}, s_{\text{idk}}^{(I)}\right) \log \pi_L\left(\hat{\boldsymbol{x}} | \boldsymbol{s}^{(I)}, s_{\text{idk}}^{(I)}\right)\right]\right].$$

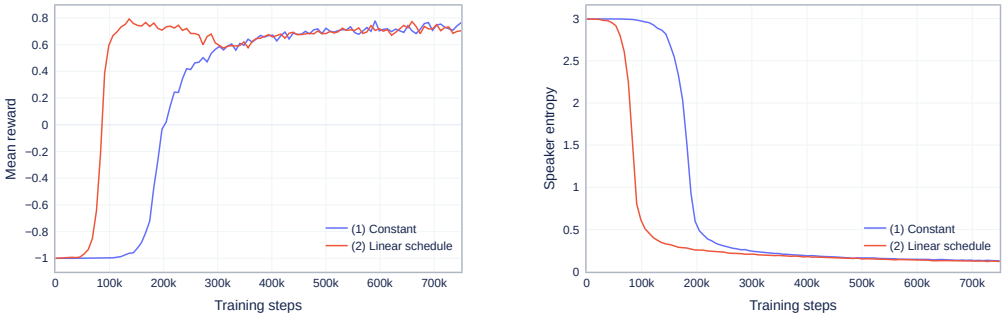
E.2 DATASETS

The datasets used to evaluate the proposed games are the ImageNet (Russakovsky et al., 2015) and CelebA (Liu et al., 2015). We use the datasets provided by Chaabouni et al. (2022), where the authors preprocess the data, as we will explain next. The ImageNet dataset contains mainly RGB images of objects and animals clustered over 1000 labels. The training of the LG and ETL experiments uses 99% of the original training data of ImageNet and the official validation set as the test set. The CelebA dataset contains RGB images of celebrity faces denoting 10177 different identities. This dataset also describes binary attributes for each image, like smiling, hair color, and glasses. Additionally, a new dataset split is performed to ensure there is overlapping for all identities between all sets (train, validation, and test), see Chaabouni et al. (2022).

Finally, images from both datasets are down-sampled using bicubic sampling, forcing the shorted side to have 256 pixels, and then a center crop of 224x224 pixels is applied. The channel axis also suffers normalization using mean and SD obtained from the ImageNet train set (He et al., 2016). Afterward, a ResNet architecture pre-trained with BYOL (Grill et al., 2020) on ImageNet outputs the representation used in all experiments, see Appendix D.1.

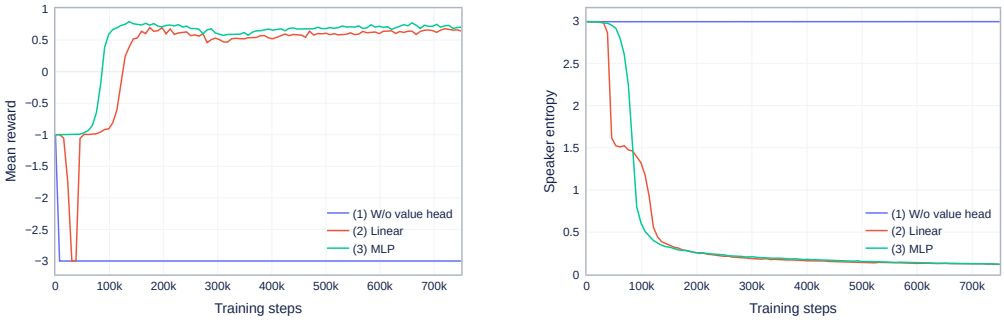
E.3 SCHEDULING NOISE

The approach on how to introduce noise to the NLG and MRILG games during the training phase highly influences the learning of both agents, which ultimately can impact overall performance and sample efficiency. As such, we present the Figure 7 and compare different ways to introduce and schedule the noise threshold during training. From Figure 7a, in the first trial (1), the noise value is fixed at its final value (0.5) during the entire training procedure. For the second one (2), a scheduler linearly scales the noise from 0 to 0.5 during the first 40% of the training steps. Analyzing both experiments, we observe that by gradually scheduling the noise, instead of fixing it at the pretended end value, the mean reward leaves the failing zone considerably earlier, meaning the agents have an easier time starting coordinating on a shared communication protocol. Having the noise increase gradually allows for the pair of agents to first focus on creating a common language and, only after, slowly adjust to the noise in the message. Note that introducing noise makes the environment highly stochastic since the noise introduced is random. Additionally, the Listener is the only one capable of perceiving this modification, which makes the coordination between the pair more challenging. Figure 7b also corroborates our analysis, where in (1), the Speaker's entropy only starts decreasing after $120\,\text{k}$ steps, as opposed to (2), where the same happens just after $50\,\text{k}$ steps. As such, a communication protocol starts to emerge much sooner in (2), where the Speaker begins conveying more stable and regular messages.



(a) (b)

Figure 7: Mean reward (a) and the mean value for the Speaker's policy entropy (b) on the ImageNet dataset of different implementations of MRILG, with $\nu = -0.5$, and $|\mathbb{C}| = 1024$. The MRILG implementations differ on how to schedule the noise λ value during training: (1) λ remains constant at the end value 0.5; (2) λ linearly scales from 0 to 0.5 during the first $300\,\mathrm{k}$ steps, staying at this value afterward.



(a) (b)

Figure 8: Mean reward (a) and the mean value for the Speaker's policy entropy (b) on the ImageNet dataset of different implementations of MRILG, with $\lambda = 0.5$, $\nu = -0.5$, and $|\mathbb{C}| = 1024$. The MRILG implementations differ in the architecture used for the value head $v(\cdot\,;\phi)$: (1) $v(\cdot\,;\phi)$ is not implemented; (2) $v(\cdot\,;\phi)$ is a single linear layer; (3) $v(\cdot\,;\phi)$ is an MLP (w/ 2 hidden layers).

E.4 Compare Different Listener Head Configurations

At the end of the Section 2.2.2, we argued that the Listener must have a critic head complementary to its actor's head. The Listener requires this design choice to have sufficient conditions to learn and understand how to evaluate actions to further adjust its policy on mapping messages to the correct candidates (correct actions) accordingly. Only when the Listener has such capabilities can the pair of agents coordinate on developing a communication protocol efficiently.

We support such a claim in Figure 8, where Figure 8a shows the game's mean reward obtained by the agents during training, and the Speaker's entropy progress during training appears in Figure 8b. From Figure 8a, we observe the agents cannot learn how to communicate when we implement the Listener without a critic head (1), where the reward is the lowest possible until the end of the training phase. In this case, we also see that the Seaker's entropy remains high and constant throughout learning (Figure 8b), indicating that the pair of agents never agree on a communication protocol.

When the Listener architecture includes a linear layer as the critic head (2), there is a considerable performance boost in the final reward obtained by both agents. After an unstable initial training period (first 50k steps), where the Listener tries to exploit the idk action unsuccessfully, the agents gradually learn how to coordinate. Consequently, a common language emerges starting from 100k

steps. Validating this result, we observe that the speaker entropy starts decreasing around the 35k step mark. As such, since early training, the Speaker starts creating more systematic messages, further facilitating the Listerner's job of mapping messages to the positive candidates.

Finally, we arrive at our default implementation by substituting the linear layer with an MLP for the critic head (3). This additional modification improves the sample efficiency compared to the linear version, where the established language emerges succeeding 80k steps (Figure 8a). Furthermore, as the training routine stabilizes (300k step onward), we observe that the MLP version plateaus to a slightly higher mean reward than the linear version. Factoring in the Speaker's entropy, we note a smooth decrease, after the 55k step mark, to the same lower bound as the linear version, around 0.12. Although the entropy for the MLP version leaves a high bound later, it achieves a lower value earlier than the linear version, which is the key to have better sample efficiency.

F EASE & TRANSFER LEARNING

Following Chaabouni et al. (2022), we also utilize secondary tasks to evaluate the generality and applicability of the languages learned in each LG variant. This evaluation procedure is called Ease & Transfer Learning (ETL). The primary purpose of ETL is to evaluate if a preceding communication protocol, which was grounded in a particular task, can be used by new agents to solve new tasks. If this proposition holds, we get solid and empirical evidence about the generality of the emergent language since it can encode high-level descriptive information to solve different types of tasks related to the original environment.

ETL comprises four tasks: *Discrimination*, *Classification*, *Attribute*, and *Reconstruction*. We use the first two tasks to evaluate instances of games played with the ImageNet dataset and all four tasks when using the CelebA dataset. Due to the lack of description and reproducibility in Chaabouni et al. (2022), we thoroughly report the objective of each task accompanied by the training regime and hyper-parameters adopted.

We define new Listener agents to play the new tasks with a previously trained Speaker from one of the original LG games. Additionally, we freeze the Speaker weights, meaning the communication protocol remains fixed throughout training and evaluation of the ETL task. As such, we generate a new discrete dataset that depends on the Speaker (converting each image into a message). On this account, the only learning agent is the Listener. The Listener architecture has a common module used to process the received message that does not depend on the ETL task. This module is equal to the one implemented in the Listener architecture used in the LG, see Section 2.2.1. The difference between architectures occurs in the Listener's head module, as we will describe when introducing each task.

Similar to the tests employed for the LG games, we add a new dimension for the ETL tasks where we perturb the communication channel by adding noise. Accordingly, by varying this new dimension, we can analyze how it affects the generalization capabilities of the communication protocols. We also force all ETL tasks to have only one round for fair comparisons. We will now describe each ETL task in detail, and afterward, the evaluation performed.

F.1 DISCRIMINATION TASK

This task is similar to the original LG task, where the difference is that we add noise to the candidates and the Speaker's input. The noise follows a Gaussian distribution with mean μ and a standard deviation σ, see Table 14. Since we now have a fixed dataset, we regard this task as a supervised problem where we design the Listener to have a similar architecture to the one used in the original LG (SS). As such, we define a distance loss to train the Listener to recognize similarities between messages and the correct candidate and, simultaneously, dissimilarities to opposing ones. To create a more challenging task, we set the size of the candidates' set, $|\mathbb{C}|$, to 4096.

F.1.1 LISTENER'S HEAD ARCHITECTURE

We plug the head defined for the LG (SS) (Appendix D.2.2), where a similarity function computes similarities between the message and each candidate. Then, a softmax function converts similarities into a categorical distribution to train using InfoNCE loss (Oord et al., 2018).

Table 13: Hyperparameters modified (from Table 16) for the Discrimination task.

hyperparameter	value
μ	0
σ	0.5
$\lvert\mathbb{C}\rvert$	4096
batch size	$4096\ (=\lvert\mathbb{C}\rvert)$

Table 14: Hyperparameters modified (from Table 16) for the Classification task.

hyperparameter	value
training steps (for ImageNet dataset only)	$30\,\mathrm{k}$

F.1.2 HYPERPARAMETERS

The hyperparameters adjusted for the Discrimination task appear in Table 13.

F.2 CLASSIFICATION TASK

With the classification task, we aim to evaluate whether the emerging language has any valuable information to identify the category of an image. In this task, the Speaker receives an image and encodes a message to the Listener. Afterward, the Listener tries to determine the class of the image while receiving only the message sent by the Speaker as input. For the ImageNet dataset, we use the label of each image as the target and the identity for the CelebA dataset.

F.2.1 LISTENER'S HEAD ARCHITECTURE

Since the objective of the Classification task is to identify the class of the image received by the Speaker, the Listener's head architecture consists of only one MLP where the final layer has the same size as the number of classes (1000 and 10177, for the ImageNet and CelebA datasets, respectively) and is then used for prediction after applying softmax. The MLP mentioned above also has one hidden layer of size 256 and uses ReLU as the activation function.

F.2.2 HYPERPARAMETERS

Table 14 exposes the hyperparameters modified explicity for the Classification task.

F.3 ATTRIBUTE TASK

This task is very similar to the classification task, where instead of predicting the image class, the Listener tries to predict secondary binary information related to facial attributes, such as gender, hair color, and age.

F.3.1 LISTENER'S HEAD ARCHITECTURE

The Listener architecture is similar to the one described for the classification task, Appendix F.2.1. In this case, we have an independent model to classify each attribute, where the loss adopted is the binary cross-entropy loss.

F.4 RECONSTRUCTION TASK

The final ETL task is the Reconstruction task, where the primary purpose is reconstructing the original image. The Listener's head promotes a convolutional decoder to reconstruct the original input image (Speaker's input) by taking into account only the discrete message sent by the Speaker. See Appendix F.4.1 for a complete description of the Listener's head architecture.

Table 15: Hyperparameters modified (from Table 16) for the Reconstruction task.

hyperparameter	value
G_{max}	500
Listener's optimizer	`adam w/ weight decay reg.`
Listener's optimizer `lr`	3×10^{-4}
Listener's optimizer `b1`	0.9
Listener's optimizer `b2`	0.9
Listener's optimizer `weight decay`	0.01

Table 16: Default hyperparameters used in all ETL tasks. Note that each ETL task can change some parameter values, as specified in Appendices F.1, F.2 and F.4

hyperparameter	value
training steps	$10\,\text{k}$
batch size	128
$\lvert\mathcal{W}\rvert$	20
T	10
Listener's optimizer	`adam`
Listener's optimizer lr	1×10^{-3}

F.4.1 Listener's Head Architecture

The Listener's head architecture encapsulates a decoder to reconstruct the original image. Since the hidden message features $\boldsymbol{z}_{T,\phi}$ are the input of this module and have only one dimension, the first step addresses the redimension of $\boldsymbol{z}_{T,\phi}$ into a 3D shape. As such, a single linear layer of size 2048 upsamples $\boldsymbol{z}_{T,\phi}$, where the resulting dimension is $(4, 4, 128)$ (with layout `(height, width, channel)`) after reshaping, as $128 \times 4 \times 4 = 2048$. Afterward, a decoding procedure aims to upsample the features' first two dimensions. The procedure occurs four times sequentially, and at each step, we upsample the dimensions of the current features by doubling the width and height, using nearest neighbor interpolation. Next, the upsampled features are given to a convolution layer followed by a ReLU to reduce the number of channels as part of the reconstruction process. The number of output channels is $64, 32, 16, 16$ for each convolution, respectively. Finally, a concluding convolution layer, followed by tanh, outputs the decoded image with dimensions $(64, 64, 3)$. With tanh as the final activation, the reconstructed image is the standardized version of the original one after normalization using the ImageNet coefficients (He et al., 2016). Furthermore, the parameters for all convolutions are equal: kernel size of (3×3), padding `same`, stride 1, and without bias. Additionally, we use the MSE loss to train the Listener.

Finally, we employ the same optimizer parameters as the original work (Chaabouni et al., 2022), where gradients are first clipped by their global norm when it is above a maximum threshold G_{max} (Pascanu et al., 2013). Then, `adam` with weight decay regularization (Loshchilov & Hutter, 2017) computes the update rule. For more information on the hyper-parameters used, see Appendix F.4.2.

F.4.2 Hyperparameters

Please refer to Table 15 to consult the modified hyperparameters for the Reconstrunction task.

F.5 Common Hyperparameters

Table 16 describes the hyperparameters commonly used by all ETL tasks.

F.6 Evaluation

In this sub-section, we present all results obtained by evaluating all ETL tasks regarding each LG variant trained. Similarly to the primary evaluation, Section 3, we describe the results incrementally,

Table 17: Test accuracy, with SD, of every ETL task trained using, the ImageNet dataset and over 10 seeds. During the LG training (before ETL), we fixed $|\mathbb{C}|$ to 1024.

Game	$\|\mathbb{C}\|$	ETL tasks	
		Discrimination	Classification
LG (SS)	1024	0.58 (0.05)	0.14 (0.01)
LG (RL)	1024	0.88 (0.01)	0.14 (0.01)

Table 18: Test accuracy, with SD, of every ETL task trained, using the CelebA dataset and over 10 seeds. During the LG training (before ETL), we fixed $|\mathbb{C}|$ to 1024.

Game	$\|\mathbb{C}\|$	ETL tasks			
		Discrimination	Classification	Attribute	Reconstruction
LG (SS)	1024	0.44 (0.05)	0.01 (0.00)	0.86 (0.00)	5814 (74)
LG (RL)	1024	0.62 (0.08)	0.01 (0.00)	0.85 (0.00)	5671 (192)

where we start by comparing both implementations (SS and RL) for the original LG. Afterward, we introduce noise, adding NLG and MRILG to the comparison. Besides this analysis, we also explore the impact of different values for the *idk* reward for the MRILG. Additionally, we compare the influence of the candidate set size in all LG variants. Regarding the metrics adopted, we report accuracy for all tasks except for the Reconstruction task, where we report the obtained final loss.

F.6.1 SS & RL IMPLEMENTATIONS FOR LG

We start the ETL evaluation by showcasing and comparing the performance of both prototypes of the LG (SS and RL) when training without noise. We report the test performance for all applicable ETL tasks for the ImageNet and CelebA datasets in Tables 17 and 18, respectively. As we will see next, our newly proposed variant, LG (RL), reports better results than LG (SS).

Looking at the discrimination task, we continue to observe the same behavior encountered in the LG evaluation, Section 3, where the RL version is superior to SS. From this new perspective, we obtain further evidence that the communication protocol from the RL variant is more general than the SS counterpart. In this case, the messages from the RL version are more appropriate to describe the noisy image, which facilitates the selection process by the new Listener agent. As such, the communication protocol from LG (RL) is more capable of dealing with other noise sources, such as detecting noise in the Speaker's input image.

Regarding the classification task, we see deficient performance for both datasets, with around 0.1 and 0 test accuracy for ImageNet and CelebA, respectively. Not only that, but we also observe the same results for all LG variants testing with or without noise. In Appendix F.6.5, we propose a simple explanation for these results that emerge from the messages' content and structure.

Taking a closer look at the results for the CelebA dataset, we observe similar and considerably high results (around 0.85) for the attribute task, which does not depend on the LG version, SS or RL. We can then assess that there is a specific batch of images where the Listener agent cannot decode any relevant information from the message to accurately classify the attribute.

For the final task, Reconstruction, we showcase the loss function obtained in the test set. We observe that LG (RL) obtains lower loss than LG (SS). To complement this analysis, Figure 9 shows the reconstruction of several images for the LG (SS) and LG (RL) variants. Both versions can reconstruct some particularities of the original image such as sunglasses, hat, hair color, and gender. On the other hand, information about orientation, age, or skin tone are not encoded in the messages. Additionally we can clearly observe different facial layouts and different hair color, reconstructed by LG (SS) and LG (RL), for the same image.

F.6.2 INTRODUCING NOISE

To evaluate the robustness of the proposed LG variants to message perturbations, we add a new train/test regime to ETL where the communication channel disturbs messages sent by the Speaker by adding random noise, similar to NLG and MRILG. To have an insightful analysis, we propose comparing LG (RL), NLG, and MRILG in the case of train/test regimes without and with noise. We disregard LG (SS) since LG (RL) achieved better results, see Appendix F.6.1. Additionally, to have a contained analysis, we fixed the noise level to 0.5, the candidate set size to 1024, and the *idk* reward at -0.2. The exhaustive version of these results appears at Tables 31 to 38.

The results for the ImageNet dataset appear in Tables 19 and 21 for the train/test regime without and with noise, respectively. Regarding the Discrimination task in the deterministic procedure (without noise), the accuracy obtained for the noisy games, NLG and MRILG, are slightly lower than in the LG (RL), around 0.03 and 0.06, respectively. Despite this minor decline, we continue to obtain evidence that the noisy LG variants generate robust communication protocols of similar efficiency to those induced by LG (RL). Another interesting observation is that the accuracy obtained in NLG is slightly higher than MRILG's. In the latter case, the available information suggests that the retained communication protocol by the Speaker had more redundant information in each message which is not valuable for a new one-round task. When introducing noise in the communication channel, we observe a performance increase, around 0.05, for NLG and MRILG against LG (RL). Recall that the discrimination task adds continuous noise to the Speaker's input and candidates. Having noisy inputs creates a different type of uncertainty that the communication protocol is not optimized for. Even having robust agents to message noise practically does not impact the creation of more descriptive messages when a significant part of the noise comes from another source, in this case, combined with the input.

We perform a similar comparison for the CelebA dataset deployed in Tables 20 and 22. Focusing on the Discrimination task and the deterministic regime, LG (RL), NLG, and MRILG have statistically equal accuracy values with mean 0.62, 0.60, and 0.58, respectively. CelebA contains highly identical samples since it only includes images of persons, whereas ImageNet aggregates images from 1000 object categories. In this case, the performance obtained was similar between all variants. When training and testing with noise, we observe a considerable drop in accuracy, around 0.35, for all variants. Similarly to what we discovered for the ImageNet dataset, the noise introduced to the inputs turns the discriminative task more challenging than the original game. Inspecting the results, NLG and MRILG obtain similar accuracy (0.31 and 0.30, respectively) against an accuracy for LG (RL) around 0.26. As such, we get an indication that the NLG and MRILG's protocol are slightly more robust to noise in the communication channel combined with noise in the input image.

Regarding the Attribute task, all values vary between 0.84 and 0.86. As such, the performances for every LG variant and train/test regime are highly identical. We hypothesize that there is a considerable collection of samples where the identification of each attribute is accessible and effortlessly contemplated in the communication protocols. For the other minority of samples, the communication protocols cannot encode information about the attributes. For this reason, we observe a sharp rupture in accuracy for all methods.

Finally, we compare the results obtained in the Reconstruction task. The losses for all LG variants are statistically equal for the train/test regimes without and with noise. No LG variant demonstrated superior performance for this challenging task, which is obviously expected since the reconstructed images originate from discrete tokens (message content), making extracting unique information from each image extremely difficult. Additionally, in Figures 9, 10 and 12, we show some reconstructed images for the compared LG variants (LG (RL), NLG, and MRILG, respectively). The level of reconstruction detail is similar accross variants, where we can see all variants encoding hat, sunglasses, hair color, and ignoring attributes not relevant to the original discrimination task, such as skin tone and face orientation.

F.6.3 VARYING IDK REWARD

From the original evaluation Section 3.2.2, substantial evidence supported that the *idk* reward considerably impacts the agent's overall performance in the MRILG. Thus, we continue this evaluation in the context of transfer learning, ETL, by comparing instances of MRILG with a fixed noise (0.5) and batch size (1024), where we only vary the idk reward. The evaluation description for the ImageNet

Table 19: Test accuracy, with SD, of every ETL task trained, using the ImageNet dataset and over 10 seeds. We do not apply noise during ETL's train and test phase. During the LG training (before ETL), we fixed $|\mathbb{C}|$ to 1024, λ to 0.5, and ν to -0.2.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	ETL tasks	
				Discrimination	Classification
LG (RL)	-	-	1024	0.88 (0.01)	0.14 (0.01)
NLG	0.5	-	1024	0.85 (0.03)	0.14 (0.02)
MRILG	0.5	-0.2	1024	0.86 (0.03)	0.13 (0.01)

Table 20: Test accuracy, with SD, of every ETL task trained, using the CelebA dataset and over 10 seeds. During the LG training (before ETL), we fixed $|\mathbb{C}|$ to 1024. We do not apply noise during ETL's train and test phase. During the LG training (before ETL), we fixed $|\mathbb{C}|$ to 1024, λ to 0.5, and ν to -0.2.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	ETL tasks			
				Discrimination	Classification	Attribute	Reconstruction
LG (RL)	-	-	1024	0.62 (0.08)	0.01 (0.00)	0.85 (0.00)	5671 (192)
NLG	0.5	-	1024	0.60 (0.10)	0.00 (0.00)	0.85 (0.00)	5659 (129)
MRILG	0.5	-0.2	1024	0.58 (0.12)	0.00 (0.00)	0.85 (0.00)	5644 (121)

dataset appears in Tables 23 and 25 for the train/test regime without and with noise, respectively. Similarly, and according to the same respective noise schemes, Tables 24 and 26 present the results for the CelebA dataset. The extensive report divided by the noise level and train/test regime appears in Tables 31 to 38.

Starting with the ImageNet dataset, we focus on the Discrimination task. In Table 23, we place the deterministic train/test regime results and observe that the choice for the *idk* impacts the overall performance moderately. The highest obtained test accuracy for the three reward levels (-0.5, -0.2, and -0.05) is around 0.86 when *idk* is fixed at -0.2, opposing a slight decrease to 0.82 when $\nu = -0.5$ and a large decrease to 0.65 for the remain *idk* level (-0.05). We can describe this conduct as, when having an *idk* reward close to 0 can direct the Listener toward developing a communication protocol containing more redundant information (playing more rounds than necessary), having a higher likelihood of overfitting the training dataset.

When adding noise to the ETL task, we obtain similar results, where the *idk* reward of -0.05 outputs lower results than the other two levels. In this train/test regime, the new Listener agent has more difficulty reasoning and combining the two input noisy sources (message and candidates), decreasing the test accuracy by almost 0.50 (roughly 0.8 to 0.4). Based on the difference in performance for both train/test regimes, we can infer that having access to the complete message generated by the Speaker (all information) is crucial to discriminate noisy inputs.

Addressing now the CelebA dataset, we observe a similar result for the Discrimination task in the deterministic train/test regime, where the *idk* rewards of -0.5 and -0.2) obtains the highest test accuracies of 0.59 and 0.58, against 0.28 got when setting the *idk* reward at -0.05. Thus, we can employ the same reasoning as described above for the ImageNet dataset, and since CelebA is a more challenging dataset (Appendix F.6.2) the differences are even more prominent. Similarly, having an *idk* away from 0 balances the creation of a communication protocol that leverages multi-round information in its development and, at the same time, is not redundant by only continuing playing when new information can be attained. Regarding the Attribute task, the results are the same when setting the *idk* reward to -0.5 or -0.2. For the value -0.05, the accuracy drops slightly to 0.83, which can be, again, a consequence of the presence of more redundant information in the communication protocol (see text above). Moving to the final task, Reconstruction, we point to the

Table 21: Test accuracy, with SD, of every ETL task trained, using the ImageNet dataset and over 10 seeds. We fix the noise during ETL's train and test phase at 0.5. During the LG training (before ETL), we fixed $|\mathbb{C}|$ to 1024, λ to 0.5, and ν to -0.2.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	ETL tasks	
				Discrimination	Classification
LG (RL)	-	-	1024	0.36 (0.02)	0.07 (0.00)
NLG	0.5	-	1024	0.41 (0.02)	0.08 (0.01)
MRILG	0.5	-0.2	1024	0.41 (0.01)	0.07 (0.01)

Table 22: Test accuracy, with SD, of every ETL task trained, using the CelebA dataset and over 10 seeds. We fix the noise during ETL's train and test phase at 0.5. During the LG training (before ETL), we fixed $|\mathbb{C}|$ to 1024, λ to 0.5, and ν to -0.2.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	ETL tasks			
				Discrimination	Classification	Attribute	Reconstruction
LG (RL)	-	-	1024	0.26 (0.03)	0.00 (0.00)	0.85 (0.00)	5871 (155)
NLG	0.5	-	1024	0.31 (0.04)	0.00 (0.00)	0.85 (0.00)	5833 (103)
MRILG	0.5	-0.2	1024	0.30 (0.06)	0.00 (0.00)	0.85 (0.00)	5820 (92)

same reasoning since the obtained losses were similar for the *idk* reward of -0.5 and -0.2, having mean values of 5686 and 5644, respectively. The loss when $\nu = -0.05$ was higher at 6182.

Addressing now the noisy train/test regime, we observe, for each task, the same behavior as the deterministic regime. Focusing on the Discrimination task and similar to the deterministic train/test regime, the test accuracy is higher for the *idk* rewards of -0.5 and -0.2 than the remaining level. The same happens for the Attribute task where the *idk* rewards of -0.5 and -0.2 obtain the same value (0.85) against a slight decrease for the -0.05 level (0.83). As such, we conclude that the same behavior occurs in the noisy regime, indicating that the noise robustness lowers when the *idk* reward is too small. We also observe this in the Reconstruction task where the loss values are lower for the *idk* reward values of -0.5 and -0.2, against the value of -0.05.

F.6.4 Varying Number of Candidates

Finally, we study the last axis of the predominant hyper-parameters, which is the number of candidates. We do a quick analysis since the results comply with the main evaluation (Section 3.2.3), where increasing the number of candidates is essential to improve performance. By increasing the number of candidates, the Listener receives a broader variety of samples, facilitating learning and recalling unique and valuable information to discriminate all candidates. Consequently, a more sophisticated communication protocol is necessary as the number of candidates increases since the discriminative task becomes more difficult. For this evaluation and similar to the previous ones, we fix the noise at 0.5 and the *idk* reward at -0.2. The results are for the following game variants: LG (RL), NLG, and MRILG.

Tables 27 and 29 display the results for the ImageNet dataset employing the deterministic and noisy regimes, respectively. Looking at the Discrimination task, we observe an increase in test accuracy for all tasks as the number of candidates (used in the original RL task) scales up. Another important and related observation is that the gap between accuracies decreases in the noisy train/test regime. For example, in the regular regime (deterministic), the gap between 256 and 1024 candidates for LG (RL) and MRILG variants is about 0.11 and 0.07, respectively. On the other hand, the gaps for the same variants in the noisy regime are 0.08 and 0.05, respectively.

A similar outcome occurs for the results obtained when training with the CelebA dataset, Tables 28 and 30 display the results for the deterministic and noisy regimes, respectively. In the case of the

Table 23: Test accuracy, with SD, of every ETL task trained, using the ImageNet dataset and over 10 seeds. We do not apply noise during ETL's train and test phase. During the LG training (before ETL), we fixed $|\mathbb{C}|$ to 1024 and λ to 0.5.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	ETL tasks	
				Discrimination	Classification
MRILG	0.5	-0.5	1024	0.82 (0.03)	0.12 (0.02)
MRILG	0.5	-0.2	1024	0.86 (0.03)	0.13 (0.01)
MRILG	0.5	-0.05	1024	0.65 (0.34)	0.09 (0.05)

Table 24: Test accuracy, with SD, of every ETL task trained, using the ImageNet dataset and over 10 seeds. We do not apply noise during ETL's train and test phase. During the LG training (before ETL), we fixed $|\mathbb{C}|$ to 1024 and λ to 0.5.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	ETL tasks			
				Discrimination	Classification	Attribute	Reconstruction
MRILG	0.5	-0.5	1024	0.59 (0.09)	0.00 (0.00)	0.85 (0.00)	5686 (121)
MRILG	0.5	-0.2	1024	0.58 (0.12)	0.00 (0.00)	0.85 (0.00)	5644 (121)
MRILG	0.5	-0.05	1024	0.28 (0.32)	0.00 (0.00)	0.83 (0.02)	6182 (466)

Discrimination and Reconstruction tasks applied in both train/test regimes, there is a clear benefit of having a Speaker trained in a more complex RL task (increased number of candidates). As such, we observe that the test accuracy increases for the Discrimination task as the number of candidates increases for all LG variants. Comparetively, the loss obtained at evaluation time for the Reconstruction task also decreases as the number of candidates increases. The performance gap is also more visible in the deterministic regime, where no message noise is involved. As an illustration, consider the Discrimination task where the growth in accuracy for the MRILG variant, when increasing the number of candidates from 256 to 1024, is 0.24 and 0.13, referring to the deterministic and noise regimes, respectively. Comparatively, looking at the Reconstruction task and the same LG variant, the increment in the test loss from 256 to 1024 candidates is 125 and 58 for the deterministic and noisy train/test regimes, respectively. Finally, addressing the Attribute task, we observe no significant improvements in performance when expanding the number of candidates. Similarly to the previous evaluation analyses (Appendix F.6.2), the test accuracy stays fixed at 0.85/0.86.

F.6.5 Particular Analysis of the Classification Task

To conclude the ETL study, we now focus on the Classification task for both datasets (ImageNet and CelebA). In all experiments, the test accuracy is not higher than 0.15 and 0.01 when considering the ImageNet and CelebA datasets, respectively. We argue that this outcome is a consequence of the encoding qualities of the communication protocol. With these results, there is strong evidence that the emerged communication protocol does not encode any particular information about the class of each image since such information is irrelevant to solve the original setting (discriminating between images). Therefore, since the Listener architecture for the Classification task only receives the message sent by the Speaker as input, and it has no helpful information to solve the task.

G Additional Results

Continuing the evaluation present in Section 3, we present all tests performed for each LG variants with different hyperparameters, see Tables 39 to 46.

Table 25: Test accuracy, with SD, of every ETL task trained, using the ImageNet dataset and over 10 seeds. We fix the noise during ETL's train and test phase at 0.5. During the LG training (before ETL), we fixed $|\mathbb{C}|$ to 1024 and λ to 0.5.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	ETL tasks	
				Discrimination	Classification
MRILG	0.5	-0.5	1024	0.40 (0.02)	0.07 (0.01)
MRILG	0.5	-0.2	1024	0.41 (0.01)	0.07 (0.01)
MRILG	0.5	-0.05	1024	0.31 (0.16)	0.05 (0.03)

Table 26: Test accuracy, with SD, of every ETL task trained, using the CelebA dataset and over 10 seeds. We fix the noise during ETL's train and test phase at 0.5. During the LG training (before ETL), we fixed $|\mathbb{C}|$ to 1024 and λ to 0.5.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	ETL tasks			
				Discrimination	Classification	Attribute	Reconstruction
MRILG	0.5	-0.5	1024	0.30 (0.04)	0.00 (0.00)	0.85 (0.00)	5860 (95)
MRILG	0.5	-0.2	1024	0.30 (0.06)	0.00 (0.00)	0.85 (0.00)	5820 (92)
MRILG	0.5	-0.05	1024	0.14 (0.15)	0.00 (0.00)	0.83 (0.02)	6275 (374)

Table 27: Test accuracy, with SD, of every ETL task trained, using the ImageNet dataset and over 10 seeds. We do not apply noise during ETL's train and test phase. During the LG training (before ETL), we fixed λ to 0.5 and ν to -0.2.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	ETL tasks	
				Discrimination	Classification
LG (RL)	-	-	16	0.37 (0.06)	0.07 (0.01)
LG (RL)	-	-	64	0.64 (0.09)	0.07 (0.01)
LG (RL)	-	-	256	0.80 (0.04)	0.13 (0.01)
LG (RL)	-	-	1024	0.88 (0.01)	0.14 (0.01)
NLG	0.5	-	16	0.30 (0.05)	0.07 (0.01)
NLG	0.5	-	64	0.54 (0.05)	0.07 (0.01)
NLG	0.5	-	256	0.77 (0.04)	0.13 (0.01)
NLG	0.5	-	1024	0.85 (0.03)	0.14 (0.02)
MRILG	0.5	-0.2	16	0.25 (0.14)	0.05 (0.03)
MRILG	0.5	-0.2	64	0.51 (0.19)	0.09 (0.03)
MRILG	0.5	-0.2	256	0.76 (0.05)	0.12 (0.02)
MRILG	0.5	-0.2	1024	0.86 (0.03)	0.13 (0.01)

Table 28: Test accuracy, with SD, of every ETL task trained, using the CelebA dataset and over 10 seeds. We do not apply noise during ETL's train and test phase. During the LG training (before ETL), we fixed λ to 0.5 and ν to -0.2.

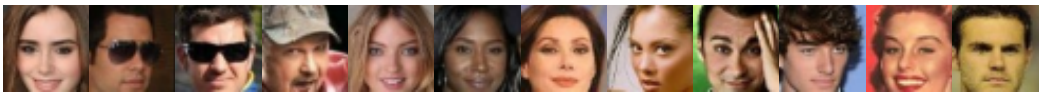
Game	λ	ν	$\lvert\mathbb{C}\rvert$	ETL tasks			
				Discrimination	Classification	Attribute	Reconstruction
LG (RL)	-	-	16	0.17 (0.07)	0.00 (0.00)	0.85 (0.00)	6050 (50)
LG (RL)	-	-	64	0.34 (0.11)	0.00 (0.00)	0.85 (0.00)	6012 (121)
LG (RL)	-	-	256	0.48 (0.09)	0.00 (0.00)	0.86 (0.00)	5750 (1544)
LG (RL)	-	-	1024	0.65 (0.09)	0.01 (0.00)	0.86 (0.00)	5633 (127)
NLG	0.5	-	16	0.16 (0.06)	0.00 (0.00)	0.85 (0.00)	6070.97 (79.74)
NLG	0.5	-	64	0.24 (0.08)	0.00 (0.00)	0.85 (0.00)	5992.45 (163.82)
NLG	0.5	-	256	0.37 (0.10)	0.00 (0.00)	0.85 (0.00)	5814.56 (191.90)
NLG	0.5	-	1024	0.54 (0.01)	0.00 (0.00)	0.85 (0.00)	5686.76 (157.08)
MRILG	0.5	-0.2	16	0.09 (0.05)	0.00 (0.00)	0.85 (0.00)	6179.81 (108.45)
MRILG	0.5	-0.2	64	0.20 (0.09)	0.00 (0.00)	0.85 (0.00)	6057.76 (205.67)
MRILG	0.5	-0.2	256	0.44 (0.12)	0.00 (0.00)	0.85 (0.00)	5696.28 (42.08)
MRILG	0.5	-0.2	1024	0.70 (0.02)	0.01 (0.00)	0.86 (0.00)	5571.15 (64.40)

Table 29: Test accuracy, with SD, of every ETL task trained, using the ImageNet dataset and over 10 seeds. We fix the noise during ETL's train and test phase at 0.5. During the LG training (before ETL), we fixed λ to 0.5 and ν to -0.2.

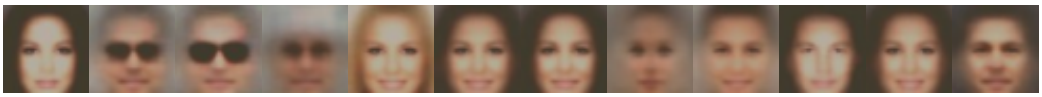
Game	λ	ν	$\lvert\mathbb{C}\rvert$	ETL tasks	
				Discrimination	Classification
LG (RL)	-	-	16	0.11 (0.01)	0.04 (0.00)
LG (RL)	-	-	64	0.22 (0.03)	0.06 (0.01)
LG (RL)	-	-	256	0.31 (0.02)	0.06 (0.00)
LG (RL)	-	-	1024	0.36 (0.02)	0.07 (0.00)
NLG	0.5	-	16	0.13 (0.02)	0.04 (0.00)
NLG	0.5	-	64	0.23 (0.02)	0.04 (0.00)
NLG	0.5	-	256	0.35 (0.02)	0.08 (0.00)
NLG	0.5	-	1024	0.41 (0.02)	0.08 (0.01)
MRILG	0.5	-0.2	16	0.11 (0.06)	0.03 (0.02)
MRILG	0.5	-0.2	64	0.21 (0.08)	0.05 (0.02)
MRILG	0.5	-0.2	256	0.35 (0.03)	0.07 (0.01)
MRILG	0.5	-0.2	1024	0.41 (0.01)	0.07 (0.01)

Table 30: Test accuracy, with SD, of every ETL task trained, using the CelebA dataset and over 10 seeds. We fix the noise during ETL's train and test phase at 0.5. During the LG training (before ETL), we fixed λ to 0.5 and ν to -0.2.

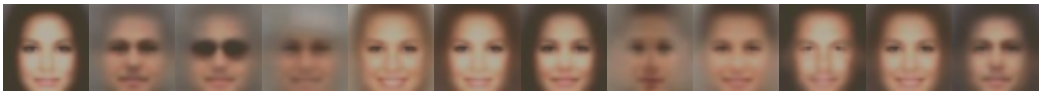
Game	λ	ν	$\lvert\mathbb{C}\rvert$	ETL tasks			
				Discrimination	Classification	Attribute	Reconstruction
LG (RL)	0	-	16	0.06 (0.02)	0.00 (0.00)	0.85 (0.00)	6210 (53)
LG (RL)	0	-	64	0.13 (0.03)	0.00 (0.00)	0.85 (0.00)	6061 (140)
LG (RL)	0	-	256	0.19 (0.03)	0.00 (0.00)	0.85 (0.00)	5896 (117)
LG (RL)	0	-	1024	0.26 (0.03)	0.00 (0.00)	0.85 (0.00)	5871 (155)
NLG	0.5	-	16	0.06 (0.02)	0.00 (0.00)	0.85 (0.00)	6196 (73)
NLG	0.5	-	64	0.11 (0.04)	0.00 (0.00)	0.85 (0.00)	6096 (110)
NLG	0.5	-	256	0.20 (0.04)	0.00 (0.00)	0.85 (0.00)	5968 (160)
NLG	0.5	-	1024	0.31 (0.04)	0.00 (0.00)	0.85 (0.00)	5833 (103)
MRILG	0.5	-0.2	16	0.07 (0.03)	0.00 (0.00)	0.85 (0.00)	6203 (76)
MRILG	0.5	-0.2	64	0.12 (0.05)	0.00 (0.00)	0.85 (0.00)	6116 (234)
MRILG	0.5	-0.2	256	0.25 (0.04)	0.00 (0.00)	0.85 (0.00)	5844 (92)
MRILG	0.5	-0.2	1024	0.30 (0.06)	0.00 (0.00)	0.85 (0.00)	5820 (85)



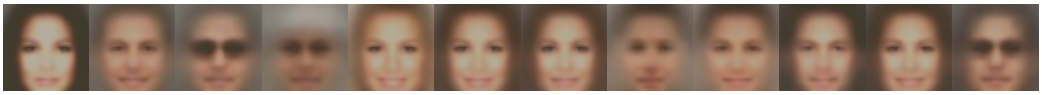
(a) Original CelebA images, randomly selected.



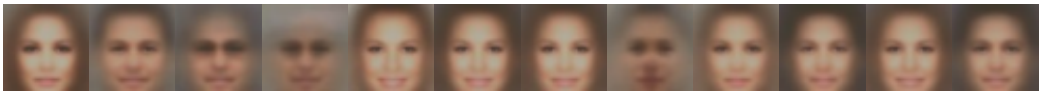
(b) LG (SS), train/test ETL Reconstruction task without noise.



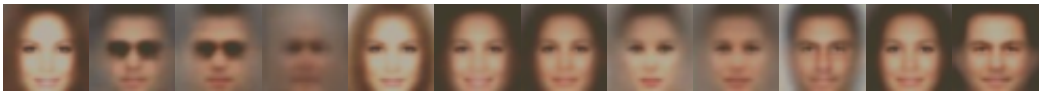
(c) LG (SS), train/test ETL Reconstruction task with a noise level of 0.25.



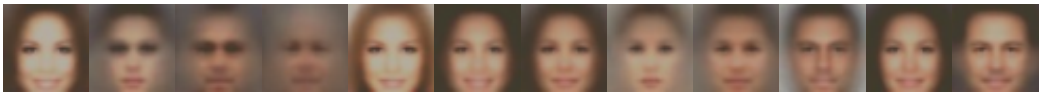
(d) LG (SS), train/test ETL Reconstruction task with a noise level of 0.5.



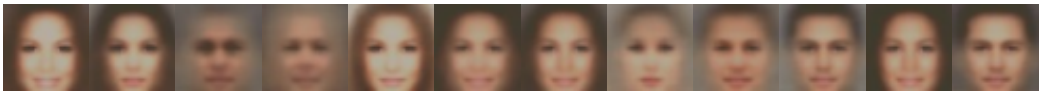
(e) LG (SS), train/test ETL Reconstruction task with a noise level of 0.75.



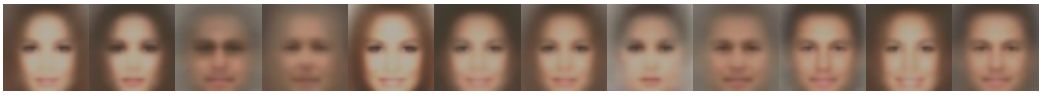
(f) LG (RL), train/test ETL Reconstruction task without noise.



(g) LG (RL), train/test ETL Reconstruction task with a noise level of 0.25.



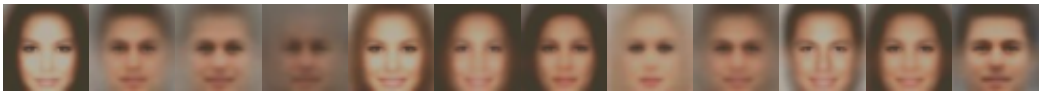
(h) LG (RL), train/test ETL Reconstruction task with a noise level of 0.5.



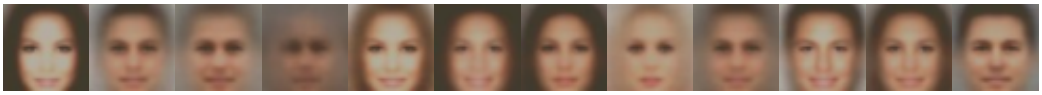
(i) LG (RL), train/test ETL Reconstruction task with a noise level of 0.75.

Figure 9: Random outputs of ETL's Reconstruction task when using a Speaker trained in LG (SS) and LG (RL) games. We report reconstructions when training and testing without (Figures 9b and 9f) and with noise (Figures 9c to 9e and 9g to 9i), in the ETL task. The faces shown in each column are a reconstruction of the original images available in the matching columns of Figure 9a.

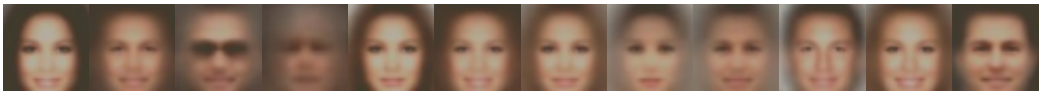
(a) Original CelebA images, randomly selected.



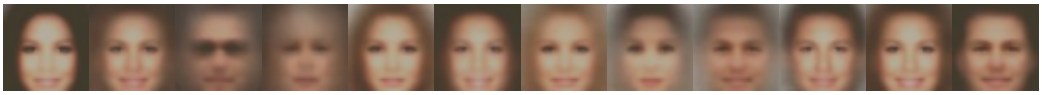
(b) NLG with $\lambda = 0.25$, train/test ETL Reconstruction task without noise.



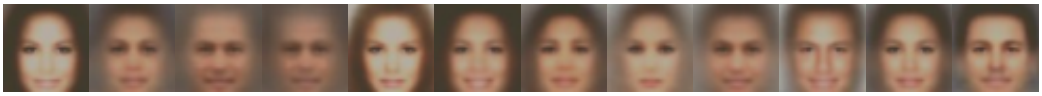
(c) NLG with $\lambda = 0.25$, train/test ETL Reconstruction task with a noise level of 0.25.



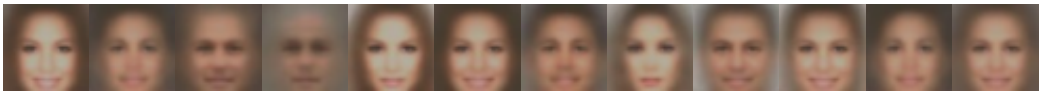
(d) NLG with $\lambda = 0.5$, train/test ETL Reconstruction task without noise.



(e) NLG with $\lambda = 0.5$, train/test ETL Reconstruction task with a noise level of 0.75.

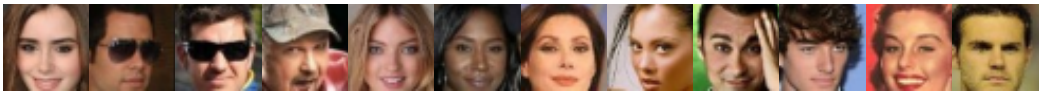


(f) NLG with $\lambda = 0.75$, train/test ETL Reconstruction task without noise.

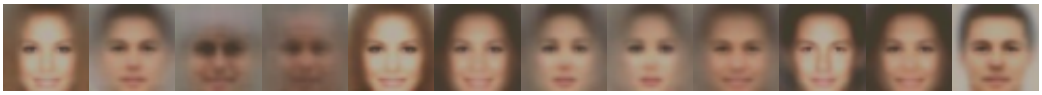


(g) NLG with $\lambda = 0.75$, train/test ETL Reconstruction task with a noise level of 0.25.

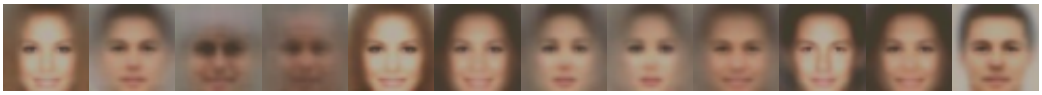
Figure 10: Random outputs of ETL's Reconstruction task when using a Speaker trained in the NLG. We report reconstructions when training and testing without (Figures 10b, 10d and 10f) and with noise (Figures 10c, 10e and 10g), in the ETL task. The faces shown in each column are a reconstruction of the original images available in the matching column of Figure 10a.



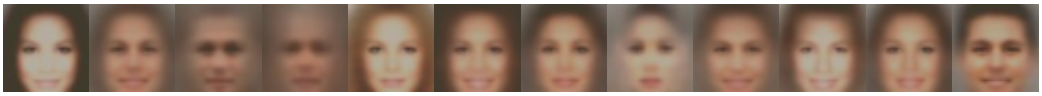
(a) Original CelebA images, randomly selected.



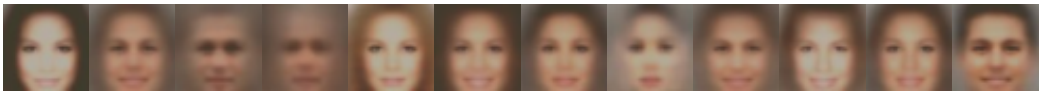
(b) MRILG with $\lambda = 0.25$ and $\nu = -0.5$, train/test ETL Reconstruction task without noise.



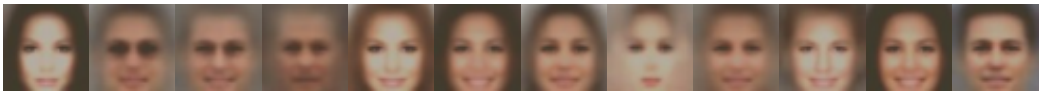
(c) MRILG with $\lambda = 0.25$ and $\nu = -0.5$, train/test ETL Reconstruction task with a noise level of 0.25.



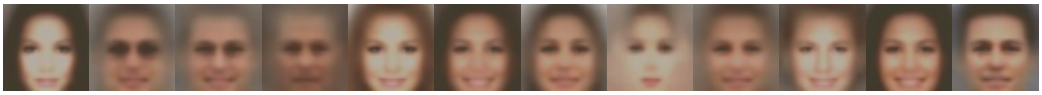
(d) MRILG with $\lambda = 0.25$ and $\nu = -0.2$, train/test ETL Reconstruction task without noise.



(e) MRILG with $\lambda = 0.25$ and $\nu = -0.2$, train/test ETL Reconstruction task with a noise level of 0.25.

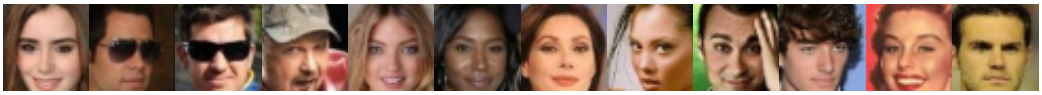


(f) MRILG with $\lambda = 0.25$ and $\nu = -0.05$, train/test ETL Reconstruction task without noise.

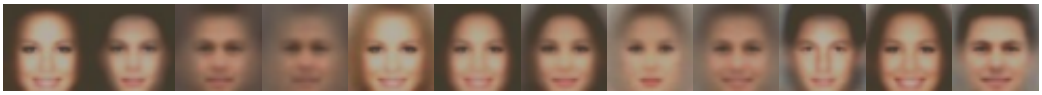


(g) MRILG with $\lambda = 0.25$ and $\nu = -0.05$, train/test ETL Reconstruction task with a noise level of 0.25.

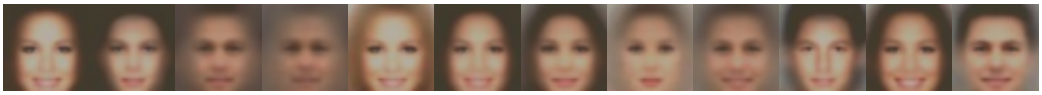
Figure 11: Random outputs of ETL's Reconstruction task when using a Speaker trained in the MRILG, with noise $\lambda = 0.25$. We report reconstructions when training and testing without (Figures 11b, 11d and 11f) and with noise (Figures 11c, 11e and 11g), in the ETL task. The faces shown in each column are a reconstruction of the original images available in the matching column of Figure 11a.



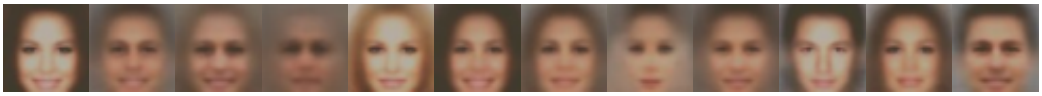
(a) Original CelebA images, randomly selected.



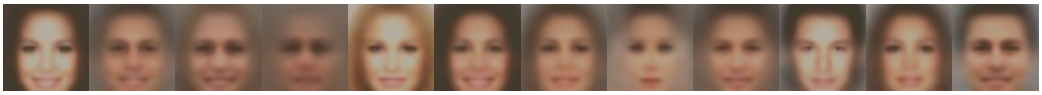
(b) MRILG with $\lambda = 0.5$ and $\nu = -0.5$, train/test ETL Reconstruction task without noise.



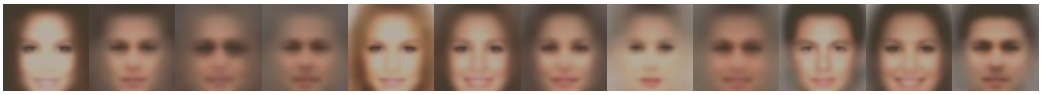
(c) MRILG with $\lambda = 0.5$ and $\nu = -0.5$, train/test ETL Reconstruction task with a noise level of 0.5.



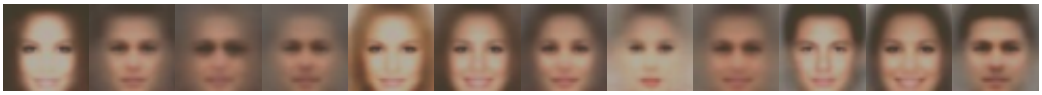
(d) MRILG with $\lambda = 0.5$ and $\nu = -0.2$, train/test ETL Reconstruction task without noise.



(e) MRILG with $\lambda = 0.5$ and $\nu = -0.2$, train/test ETL Reconstruction task with a noise level of 0.5.

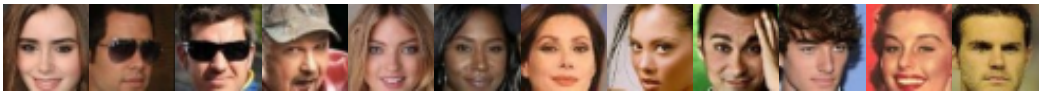


(f) MRILG with $\lambda = 0.5$ and $\nu = -0.05$, train/test ETL Reconstruction task without noise.

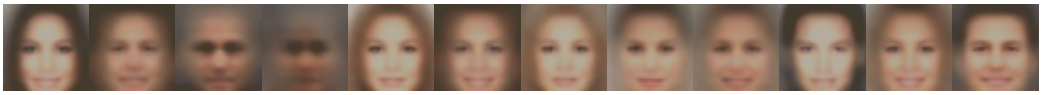


(g) MRILG with $\lambda = 0.5$ and $\nu = -0.05$, train/test ETL Reconstruction task with a noise level of 0.5.

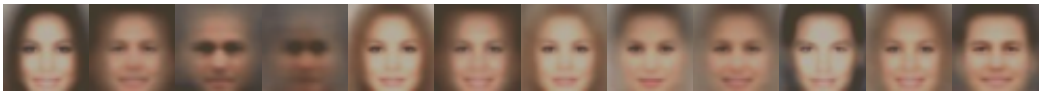
Figure 12: Random outputs of ETL's Reconstruction task when using a Speaker trained in the MRILG, with noise $\lambda = 0.25$. We report reconstructions when training and testing without (Figures 12b, 12d and 12f) and with noise (Figures 12c, 12e and 12g), in the ETL task. The faces shown in each column are a reconstruction of the original images available in the matching column of Figure 12a.



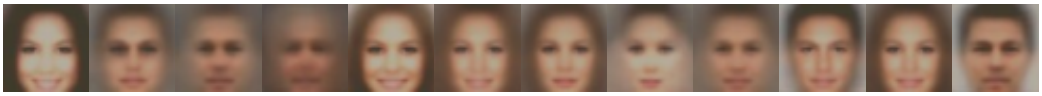
(a) Original CelebA images, randomly selected.



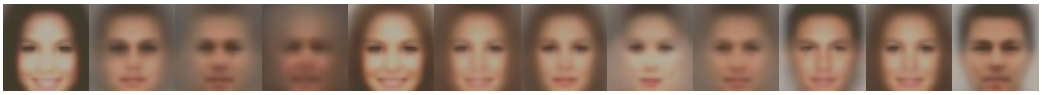
(b) MRILG with $\lambda = 0.75$ and $\nu = -0.5$, train/test ETL Reconstruction task without noise.



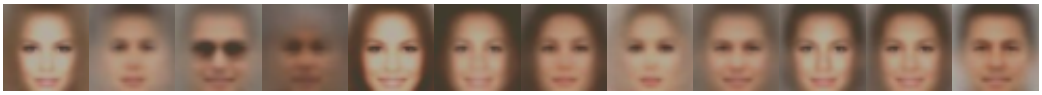
(c) MRILG with $\lambda = 0.75$ and $\nu = -0.5$, train/test ETL Reconstruction task with a noise level of 0.5.



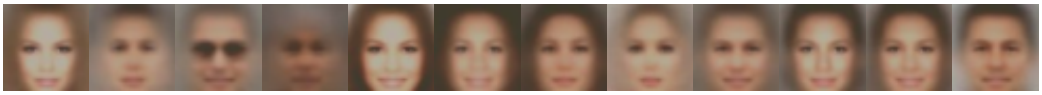
(d) MRILG with $\lambda = 0.75$ and $\nu = -0.2$, train/test ETL Reconstruction task without noise.



(e) MRILG with $\lambda = 0.75$ and $\nu = -0.2$, train/test ETL Reconstruction task with a noise level of 0.75.



(f) MRILG with $\lambda = 0.75$ and $\nu = -0.05$, train/test ETL Reconstruction task without noise.



(g) MRILG with $\lambda = 0.75$ and $\nu = -0.05$, train/test ETL Reconstruction task with a noise level of 0.75.

Figure 13: Random outputs of ETL's Reconstruction task when using a Speaker trained in the MRILG, with noise $\lambda = 0.25$. We report reconstructions when training and testing without (Figures 13b, 13d and 13f) and with noise (Figures 13c, 13e and 13g), in the ETL task. The faces shown in each column are a reconstruction of the original images available in the matching column of Figure 13a.

Table 31: Test accuracy, with SD, of every ETL task trained, using the ImageNet dataset and over 10 seeds. We do not apply noise during ETL's train and test phase.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	Discrimination	Classification
LG (SS)	0	0	16	0.23 (0.02)	0.08 (0.01)
LG (SS)	0	0	64	0.33 (0.06)	0.11 (0.01)
LG (SS)	0	0	256	0.44 (0.05)	0.13 (0.01)
LG (SS)	0	0	1024	0.58 (0.05)	0.14 (0.01)
LG (RL)	0	0	16	0.37 (0.06)	0.07 (0.01)
LG (RL)	0	0	64	0.64 (0.09)	0.11 (0.02)
LG (RL)	0	0	256	0.80 (0.04)	0.13 (0.01)
LG (RL)	0	0	1024	0.88 (0.01)	0.14 (0.01)
NLG	0.25	0	16	0.37 (0.08)	0.08 (0.02)
NLG	0.25	0	64	0.63 (0.03)	0.10 (0.00)
NLG	0.25	0	256	0.81 (0.03)	0.13 (0.01)
NLG	0.25	0	1024	0.86 (0.03)	0.14 (0.02)
NLG	0.5	0	16	0.30 (0.05)	0.07 (0.01)
NLG	0.5	0	64	0.54 (0.05)	0.10 (0.01)
NLG	0.5	0	256	0.77 (0.04)	0.13 (0.01)
NLG	0.5	0	1024	0.85 (0.03)	0.14 (0.02)
NLG	0.75	0	16	0.22 (0.04)	0.06 (0.01)
NLG	0.75	0	64	0.44 (0.04)	0.08 (0.01)
NLG	0.75	0	256	0.66 (0.05)	0.11 (0.01)
NLG	0.75	0	1024	0.80 (0.03)	0.14 (0.02)
MRILG	0.25	-0.5	16	0.32 (0.06)	0.06 (0.01)
MRILG	0.25	-0.5	64	0.64 (0.06)	0.11 (0.01)
MRILG	0.25	-0.5	256	0.78 (0.04)	0.12 (0.01)
MRILG	0.25	-0.5	1024	0.87 (0.04)	0.14 (0.02)
MRILG	0.25	-0.2	16	0.33 (0.12)	0.07 (0.02)
MRILG	0.25	-0.2	64	0.50 (0.27)	0.09 (0.05)
MRILG	0.25	-0.2	256	0.72 (0.25)	0.12 (0.04)

Table 31: Continued from previous page.

MRILG	0.25	−0.2	1024	0.86 (0.03)	0.13 (0.01)
MRILG	0.25	−0.05	16	0.19 (0.15)	0.05 (0.03)
MRILG	0.25	−0.05	64	0.49 (0.21)	0.09 (0.04)
MRILG	0.25	−0.05	256	0.77 (0.05)	0.12 (0.01)
MRILG	0.25	−0.05	1024	0.83 (0.06)	0.12 (0.02)
MRILG	0.5	−0.5	16	0.32 (0.05)	0.07 (0.01)
MRILG	0.5	−0.5	64	0.61 (0.05)	0.11 (0.01)
MRILG	0.5	−0.5	256	0.79 (0.02)	0.13 (0.01)
MRILG	0.5	−0.5	1024	0.82 (0.03)	0.12 (0.02)
MRILG	0.5	−0.2	16	0.25 (0.14)	0.05 (0.03)
MRILG	0.5	−0.2	64	0.51 (0.19)	0.09 (0.03)
MRILG	0.5	−0.2	256	0.76 (0.05)	0.12 (0.02)
MRILG	0.5	−0.2	1024	0.86 (0.03)	0.13 (0.01)
MRILG	0.5	−0.05	16	0.17 (0.13)	0.04 (0.03)
MRILG	0.5	−0.05	64	0.42 (0.23)	0.07 (0.04)
MRILG	0.5	−0.05	256	0.60 (0.32)	0.10 (0.05)
MRILG	0.5	−0.05	1024	0.65 (0.34)	0.09 (0.05)
MRILG	0.75	−0.5	16	0.23 (0.04)	0.06 (0.01)
MRILG	0.75	−0.5	64	0.46 (0.05)	0.09 (0.01)
MRILG	0.75	−0.5	256	0.67 (0.04)	0.12 (0.02)
MRILG	0.75	−0.5	1024	0.76 (0.06)	0.13 (0.01)
MRILG	0.75	−0.2	16	0.24 (0.04)	0.06 (0.01)
MRILG	0.75	−0.2	64	0.41 (0.16)	0.08 (0.03)
MRILG	0.75	−0.2	256	0.54 (0.29)	0.09 (0.05)
MRILG	0.75	−0.2	1024	0.77 (0.05)	0.13 (0.02)
MRILG	0.75	−0.05	16	0.11 (0.12)	0.03 (0.03)
MRILG	0.75	−0.05	64	0.25 (0.22)	0.05 (0.04)
MRILG	0.75	−0.05	256	0.68 (0.04)	0.11 (0.01)
MRILG	0.75	−0.05	1024	0.47 (0.37)	0.08 (0.06)

Table 32: Test accuracy, with SD, of every ETL task trained, using the CelebA dataset and over 10 seeds. We do not apply noise during ETL's train and test phase.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	Discrimination	Classification	Attribute	Reconstruction
LG (SS)	0	0	16	0.18	0.00	0.85	6094
				(0.04)	(0.00)	(0.00)	(84)
LG (SS)	0	0	64	0.31	0.00	0.86	5971
				(0.02)	(0.00)	(0.00)	(78)
LG (SS)	0	0	256	0.39	0.01	0.86	5852
				(0.06)	(0.00)	(0.00)	(122)
LG (SS)	0	0	1024	0.44	0.01	0.86	5814
				(0.05)	(0.00)	(0.00)	(74)
LG (RL)	0	0	16	0.15	0.00	0.85	6110
				(0.06)	(0.00)	(0.00)	(79)
LG (RL)	0	0	64	0.33	0.00	0.85	5930
				(0.08)	(0.00)	(0.00)	(178)
LG (RL)	0	0	256	0.47	0.00	0.86	5710
				(0.11)	(0.00)	(0.00)	(167)
LG (RL)	0	0	1024	0.62	0.01	0.85	5671
				(0.08)	(0.00)	(0.00)	(192)
NLG	0.25	0	16	0.15	0.00	0.85	6123
				(0.07)	(0.00)	(0.00)	(99)
NLG	0.25	0	64	0.27	0.00	0.85	5995
				(0.08)	(0.00)	(0.00)	(148)
NLG	0.25	0	256	0.47	0.00	0.85	5754
				(0.10)	(0.00)	(0.00)	(154)
NLG	0.25	0	1024	0.61	0.01	0.85	5689
				(0.09)	(0.00)	(0.00)	(87)
NLG	0.5	0	16	0.13	0.00	0.85	6142
				(0.04)	(0.00)	(0.00)	(91)
NLG	0.5	0	64	0.22	0.00	0.85	6011
				(0.08)	(0.00)	(0.00)	(135)
NLG	0.5	0	256	0.38	0.00	0.85	5859
				(0.10)	(0.00)	(0.00)	(181)
NLG	0.5	0	1024	0.60	0.00	0.85	5659
				(0.10)	(0.00)	(0.00)	(129)
NLG	0.75	0	16	0.11	0.00	0.85	6164
				(0.06)	(0.00)	(0.00)	(104)
NLG	0.75	0	64	0.24	0.00	0.85	5957
				(0.06)	(0.00)	(0.00)	(101)
NLG	0.75	0	256	0.37	0.00	0.85	5816
				(0.09)	(0.00)	(0.00)	(141)
NLG	0.75	0	1024	0.59	0.00	0.85	5679
				(0.08)	(0.00)	(0.00)	(78)
MRILG	0.25	-0.5	16	0.14	0.00	0.85	6113
				(0.05)	(0.00)	(0.00)	(126)
MRILG	0.25	-0.5	64	0.29	0.00	0.85	5949
				(0.05)	(0.00)	(0.00)	(102)
MRILG	0.25	-0.5	256	0.46	0.00	0.85	5736
				(0.11)	(0.00)	(0.00)	(105)
MRILG	0.25	-0.5	1024	0.65	0.01	0.85	5621
				(0.10)	(0.00)	(0.00)	(109)
MRILG	0.25	-0.2	16	0.14	0.00	0.85	6144
				(0.06)	(0.00)	(0.00)	(69)
MRILG	0.25	-0.2	64	0.25	0.00	0.85	5992
				(0.14)	(0.00)	(0.01)	(283)
MRILG	0.25	-0.2	256	0.41	0.00	0.85	5871
				(0.18)	(0.00)	(0.02)	(328)

Table 32: Continued from previous page.

MRILG	0.25	−0.2	1024	0.60 (0.08)	0.01 (0.00)	0.85 (0.00)	5610 (138)
MRILG	0.25	−0.05	16	0.13 (0.08)	0.00 (0.00)	0.85 (0.01)	6161 (210)
MRILG	0.25	−0.05	64	0.25 (0.13)	0.00 (0.00)	0.85 (0.02)	6057 (256)
MRILG	0.25	−0.05	256	0.41 (0.12)	0.00 (0.00)	0.85 (0.00)	5781 (153)
MRILG	0.25	−0.05	1024	0.60 (0.13)	0.00 (0.00)	0.85 (0.00)	5644 (115)
MRILG	0.5	−0.5	16	0.14 (0.05)	0.00 (0.00)	0.85 (0.00)	6146 (75)
MRILG	0.5	−0.5	64	0.29 (0.08)	0.00 (0.00)	0.85 (0.00)	5909 (125)
MRILG	0.5	−0.5	256	0.43 (0.11)	0.00 (0.00)	0.85 (0.00)	5765 (142)
MRILG	0.5	−0.5	1024	0.59 (0.09)	0.00 (0.00)	0.85 (0.00)	5686 (121)
MRILG	0.5	−0.2	16	0.13 (0.07)	0.00 (0.00)	0.85 (0.00)	6134 (97)
MRILG	0.5	−0.2	64	0.23 (0.11)	0.00 (0.00)	0.85 (0.02)	6037 (272)
MRILG	0.5	−0.2	256	0.49 (0.09)	0.00 (0.00)	0.85 (0.00)	5691 (100)
MRILG	0.5	−0.2	1024	0.58 (0.12)	0.00 (0.00)	0.85 (0.00)	5644 (121)
MRILG	0.5	−0.05	16	0.10 (0.08)	0.00 (0.00)	0.84 (0.02)	6253 (247)
MRILG	0.5	−0.05	64	0.21 (0.11)	0.00 (0.00)	0.85 (0.01)	6059 (261)
MRILG	0.5	−0.05	256	0.35 (0.17)	0.00 (0.00)	0.85 (0.02)	5924 (297)
MRILG	0.5	−0.05	1024	0.28 (0.32)	0.00 (0.00)	0.83 (0.02)	6182 (466)
MRILG	0.75	−0.5	16	0.09 (0.01)	0.00 (0.00)	0.85 (0.00)	6241 (42)
MRILG	0.75	−0.5	64	0.24 (0.07)	0.00 (0.00)	0.85 (0.00)	6002 (117)
MRILG	0.75	−0.5	256	0.36 (0.10)	0.00 (0.00)	0.85 (0.00)	5746 (146)
MRILG	0.75	−0.5	1024	0.57 (0.09)	0.00 (0.00)	0.85 (0.00)	5652 (89)
MRILG	0.75	−0.2	16	0.07 (0.07)	0.00 (0.00)	0.84 (0.02)	6288 (221)
MRILG	0.75	−0.2	64	0.20 (0.07)	0.00 (0.00)	0.85 (0.00)	5943 (175)
MRILG	0.75	−0.2	256	0.37 (0.09)	0.00 (0.00)	0.85 (0.00)	5784 (148)
MRILG	0.75	−0.2	1024	0.51 (0.20)	0.00 (0.00)	0.85 (0.02)	5824 (334)
MRILG	0.75	−0.05	16	0.06 (0.07)	0.00 (0.00)	0.83 (0.02)	6366 (320)
MRILG	0.75	−0.05	64	0.22 (0.12)	0.00 (0.00)	0.84 (0.02)	6072 (328)
MRILG	0.75	−0.05	256	0.31 (0.19)	0.00 (0.00)	0.84 (0.02)	6003 (374)
MRILG	0.75	−0.05	1024	0.30 (0.29)	0.00 (0.00)	0.83 (0.02)	6083 (474)

Table 33: Test accuracy, with SD, of every ETL task trained, using the ImageNet dataset and over 10 seeds. We fix the noise during ETL's train and test phase at 0.25.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	Discrimination	Classification
LG (SS)	0	0	16	0.13 (0.02)	0.06 (0.01)
LG (SS)	0	0	64	0.19 (0.04)	0.08 (0.01)
LG (SS)	0	0	256	0.24 (0.04)	0.09 (0.01)
LG (SS)	0	0	1024	0.36 (0.05)	0.10 (0.01)
LG (RL)	0	0	16	0.23 (0.03)	0.06 (0.01)
LG (RL)	0	0	64	0.43 (0.05)	0.08 (0.01)
LG (RL)	0	0	256	0.59 (0.03)	0.10 (0.01)
LG (RL)	0	0	1024	0.67 (0.02)	0.11 (0.01)
NLG	0.25	0	16	0.26 (0.05)	0.06 (0.01)
NLG	0.25	0	64	0.47 (0.03)	0.09 (0.00)
NLG	0.25	0	256	0.64 (0.03)	0.11 (0.01)
NLG	0.25	0	1024	0.69 (0.03)	0.11 (0.01)
MRILG	0.25	−0.5	16	0.23 (0.04)	0.05 (0.01)
MRILG	0.25	−0.5	64	0.47 (0.04)	0.09 (0.01)
MRILG	0.25	−0.5	256	0.61 (0.04)	0.10 (0.01)
MRILG	0.25	−0.5	1024	0.70 (0.04)	0.11 (0.01)
MRILG	0.25	−0.2	16	0.23 (0.08)	0.06 (0.02)
MRILG	0.25	−0.2	64	0.37 (0.20)	0.07 (0.04)
MRILG	0.25	−0.2	256	0.56 (0.20)	0.09 (0.03)
MRILG	0.25	−0.2	1024	0.69 (0.03)	0.11 (0.01)
MRILG	0.25	−0.05	16	0.13 (0.11)	0.04 (0.02)
MRILG	0.25	−0.05	64	0.36 (0.16)	0.07 (0.03)
MRILG	0.25	−0.05	256	0.60 (0.05)	0.10 (0.01)
MRILG	0.25	−0.05	1024	0.67 (0.05)	0.10 (0.01)

Table 34: Test accuracy, with SD, of every ETL task trained, using the CelebA dataset and over 10 seeds. We fix the noise during ETL's train and test phase at 0.25.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	Discrimination	Classification	Attribute	Reconstruction
LG (SS)	0	0	16	0.11 (0.02)	0.00 (0.00)	0.85 (0.00)	6177 (67)
LG (SS)	0	0	64	0.19 (0.02)	0.00 (0.00)	0.85 (0.00)	6055 (60)
LG (SS)	0	0	256	0.25 (0.05)	0.00 (0.00)	0.85 (0.00)	5955 (116)
LG (SS)	0	0	1024	0.28 (0.05)	0.00 (0.00)	0.85 (0.00)	5932 (67)
LG (RL)	0	0	16	0.11 (0.04)	0.00 (0.00)	0.85 (0.00)	6145 (65)
LG (RL)	0	0	64	0.24 (0.05)	0.00 (0.00)	0.85 (0.00)	5973 (162)
LG (RL)	0	0	256	0.34 (0.07)	0.00 (0.00)	0.85 (0.00)	5769 (144)
LG (RL)	0	0	1024	0.47 (0.06)	0.00 (0.00)	0.85 (0.00)	5734 (178)
NLG	0.25	0	16	0.12 (0.05)	0.00 (0.00)	0.85 (0.00)	6146 (89)
NLG	0.25	0	64	0.21 (0.06)	0.00 (0.00)	0.85 (0.00)	6026 (142)
NLG	0.25	0	256	0.37 (0.08)	0.00 (0.00)	0.85 (0.00)	5793 (141)
NLG	0.25	0	1024	0.49 (0.08)	0.00 (0.00)	0.85 (0.00)	5738 (76)
MRILG	0.25	-0.5	16	0.11 (0.03)	0.00 (0.00)	0.85 (0.00)	6137 (118)
MRILG	0.25	-0.5	64	0.23 (0.04)	0.00 (0.00)	0.85 (0.00)	5974 (98)
MRILG	0.25	-0.5	256	0.36 (0.09)	0.00 (0.00)	0.85 (0.00)	5774 (97)
MRILG	0.25	-0.5	1024	0.51 (0.08)	0.01 (0.00)	0.85 (0.00)	5675 (100)
MRILG	0.25	-0.2	16	0.11 (0.04)	0.00 (0.00)	0.85 (0.00)	6164 (60)
MRILG	0.25	-0.2	64	0.20 (0.10)	0.00 (0.00)	0.85 (0.01)	6023 (277)
MRILG	0.25	-0.2	256	0.32 (0.14)	0.00 (0.00)	0.85 (0.02)	5910 (314)
MRILG	0.25	-0.2	1024	0.48 (0.07)	0.00 (0.00)	0.85 (0.00)	5663 (123)
MRILG	0.25	-0.05	16	0.10 (0.06)	0.00 (0.00)	0.85 (0.01)	6177 (203)
MRILG	0.25	-0.05	64	0.20 (0.10)	0.00 (0.00)	0.85 (0.02)	6085 (245)
MRILG	0.25	-0.05	256	0.33 (0.08)	0.00 (0.00)	0.85 (0.00)	5818 (138)
MRILG	0.25	-0.05	1024	0.49 (0.10)	0.00 (0.00)	0.85 (0.00)	5695 (102)

Table 35: Test accuracy, with SD, of every ETL task trained, using the ImageNet dataset and over 10 seeds. We fix the noise during ETL's train and test phase at 0.5.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	Discrimination	Classification
LG (SS)	0	0	16	0.06 (0.01)	0.04 (0.00)
LG (SS)	0	0	64	0.09 (0.02)	0.05 (0.01)
LG (SS)	0	0	256	0.10 (0.03)	0.05 (0.01)
LG (SS)	0	0	1024	0.16 (0.03)	0.06 (0.01)
LG (RL)	0	0	16	0.11 (0.01)	0.04 (0.00)
LG (RL)	0	0	64	0.22 (0.03)	0.06 (0.01)
LG (RL)	0	0	256	0.31 (0.02)	0.06 (0.00)
LG (RL)	0	0	1024	0.36 (0.02)	0.07 (0.00)
NLG	0.5	0	16	0.13 (0.02)	0.04 (0.00)
NLG	0.5	0	64	0.23 (0.02)	0.06 (0.01)
NLG	0.5	0	256	0.35 (0.02)	0.08 (0.00)
NLG	0.5	0	1024	0.41 (0.02)	0.08 (0.01)
MRILG	0.5	-0.5	16	0.13 (0.02)	0.04 (0.01)
MRILG	0.5	-0.5	64	0.26 (0.02)	0.06 (0.01)
MRILG	0.5	-0.5	256	0.37 (0.02)	0.08 (0.00)
MRILG	0.5	-0.5	1024	0.40 (0.02)	0.07 (0.01)
MRILG	0.5	-0.2	16	0.11 (0.06)	0.03 (0.02)
MRILG	0.5	-0.2	64	0.21 (0.08)	0.05 (0.02)
MRILG	0.5	-0.2	256	0.35 (0.03)	0.07 (0.01)
MRILG	0.5	-0.2	1024	0.41 (0.01)	0.07 (0.01)
MRILG	0.5	-0.05	16	0.07 (0.05)	0.03 (0.02)
MRILG	0.5	-0.05	64	0.17 (0.09)	0.04 (0.02)
MRILG	0.5	-0.05	256	0.28 (0.15)	0.06 (0.03)
MRILG	0.5	-0.05	1024	0.31 (0.16)	0.05 (0.03)

Table 36: Test accuracy, with SD, of every ETL task trained, using the CelebA dataset and over 10 seeds. We fix the noise during ETL's train and test phase at 0.5.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	Discrimination	Classification	Attribute	Reconstruction
LG (SS)	0	0	16	0.05 (0.01)	0.00 (0.00)	0.84 (0.00)	6291 (53)
LG (SS)	0	0	64	0.09 (0.01)	0.00 (0.00)	0.85 (0.00)	6184 (48)
LG (SS)	0	0	256	0.12 (0.03)	0.00 (0.00)	0.85 (0.00)	6107 (104)
LG (SS)	0	0	1024	0.13 (0.03)	0.00 (0.00)	0.84 (0.00)	6102 (73)
LG (RL)	0	0	16	0.06 (0.02)	0.00 (0.00)	0.85 (0.00)	6210 (53)
LG (RL)	0	0	64	0.13 (0.03)	0.00 (0.00)	0.85 (0.00)	6061 (140)
LG (RL)	0	0	256	0.19 (0.03)	0.00 (0.00)	0.85 (0.00)	5896 (117)
LG (RL)	0	0	1024	0.26 (0.03)	0.00 (0.00)	0.85 (0.00)	5871 (155)
NLG	0.5	0	16	0.06 (0.02)	0.00 (0.00)	0.85 (0.00)	6196 (73)
NLG	0.5	0	64	0.11 (0.04)	0.00 (0.00)	0.85 (0.00)	6096 (110)
NLG	0.5	0	256	0.20 (0.04)	0.00 (0.00)	0.85 (0.00)	5968 (160)
NLG	0.5	0	1024	0.31 (0.04)	0.00 (0.00)	0.85 (0.00)	5833 (103)
MRILG	0.5	-0.5	16	0.07 (0.02)	0.00 (0.00)	0.85 (0.00)	6203 (58)
MRILG	0.5	-0.5	64	0.14 (0.03)	0.00 (0.00)	0.85 (0.00)	6009 (109)
MRILG	0.5	-0.5	256	0.22 (0.04)	0.00 (0.00)	0.85 (0.00)	5899 (120)
MRILG	0.5	-0.5	1024	0.30 (0.04)	0.00 (0.00)	0.85 (0.00)	5860 (95)
MRILG	0.5	-0.2	16	0.07 (0.03)	0.00 (0.00)	0.85 (0.00)	6203 (76)
MRILG	0.5	-0.2	64	0.12 (0.05)	0.00 (0.00)	0.85 (0.01)	6116 (234)
MRILG	0.5	-0.2	256	0.25 (0.04)	0.00 (0.00)	0.85 (0.00)	5844 (85)
MRILG	0.5	-0.2	1024	0.30 (0.06)	0.00 (0.00)	0.85 (0.00)	5820 (92)
MRILG	0.5	-0.05	16	0.05 (0.03)	0.00 (0.00)	0.84 (0.02)	6307 (212)
MRILG	0.5	-0.05	64	0.10 (0.05)	0.00 (0.00)	0.84 (0.01)	6140 (221)
MRILG	0.5	-0.05	256	0.18 (0.08)	0.00 (0.00)	0.84 (0.01)	6033 (257)
MRILG	0.5	-0.05	1024	0.14 (0.15)	0.00 (0.00)	0.83 (0.02)	6275 (374)

Table 37: Test accuracy, with SD, of every ETL task trained, using the ImageNet dataset and over 10 seeds. We fix the noise during ETL's train and test phase at 0.75.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	Discrimination	Classification
LG (SS)	0	0	16	0.02 (0.00)	0.02 (0.00)
LG (SS)	0	0	64	0.02 (0.01)	0.02 (0.00)
LG (SS)	0	0	256	0.02 (0.01)	0.02 (0.00)
LG (SS)	0	0	1024	0.04 (0.01)	0.03 (0.00)
LG (RL)	0	0	16	0.03 (0.00)	0.02 (0.00)
LG (RL)	0	0	64	0.06 (0.01)	0.03 (0.00)
LG (RL)	0	0	256	0.08 (0.01)	0.03 (0.00)
LG (RL)	0	0	1024	0.09 (0.01)	0.03 (0.00)
NLG	0.75	0	16	0.04 (0.00)	0.02 (0.00)
NLG	0.75	0	64	0.06 (0.00)	0.03 (0.00)
NLG	0.75	0	256	0.09 (0.01)	0.04 (0.00)
NLG	0.75	0	1024	0.11 (0.00)	0.04 (0.00)
MRILG	0.75	−0.5	16	0.04 (0.00)	0.02 (0.00)
MRILG	0.75	−0.5	64	0.07 (0.01)	0.03 (0.00)
MRILG	0.75	−0.5	256	0.10 (0.00)	0.04 (0.00)
MRILG	0.75	−0.5	1024	0.11 (0.01)	0.04 (0.00)
MRILG	0.75	−0.2	16	0.04 (0.00)	0.02 (0.00)
MRILG	0.75	−0.2	64	0.06 (0.02)	0.03 (0.01)
MRILG	0.75	−0.2	256	0.08 (0.04)	0.03 (0.01)
MRILG	0.75	−0.2	1024	0.11 (0.01)	0.04 (0.00)
MRILG	0.75	−0.05	16	0.02 (0.02)	0.01 (0.01)
MRILG	0.75	−0.05	64	0.04 (0.03)	0.02 (0.01)
MRILG	0.75	−0.05	256	0.10 (0.01)	0.04 (0.00)
MRILG	0.75	−0.05	1024	0.07 (0.05)	0.02 (0.02)

Table 38: Test accuracy, with SD, of every ETL task trained, using the CelebA dataset and over 10 seeds. We fix the noise during ETL's train and test phase at 0.75.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	Discrimination	Classification	Attribute	Reconstruction
LG (SS)	0	0	16	0.01 (0.00)	0.00 (0.00)	0.83 (0.00)	6438 (31)
LG (SS)	0	0	64	0.02 (0.00)	0.00 (0.00)	0.83 (0.00)	6357 (33)
LG (SS)	0	0	256	0.03 (0.01)	0.00 (0.00)	0.83 (0.00)	6311 (68)
LG (SS)	0	0	1024	0.03 (0.01)	0.00 (0.00)	0.83 (0.00)	6316 (59)
LG (RL)	0	0	16	0.02 (0.00)	0.00 (0.00)	0.84 (0.00)	6331 (31)
LG (RL)	0	0	64	0.04 (0.01)	0.00 (0.00)	0.84 (0.00)	6232 (104)
LG (RL)	0	0	256	0.05 (0.01)	0.00 (0.00)	0.84 (0.00)	6122 (75)
LG (RL)	0	0	1024	0.07 (0.01)	0.00 (0.00)	0.84 (0.00)	6108 (110)
NLG	0.75	0	16	0.02 (0.01)	0.00 (0.00)	0.84 (0.00)	6291 (59)
NLG	0.75	0	64	0.04 (0.01)	0.00 (0.00)	0.84 (0.00)	6183 (64)
NLG	0.75	0	256	0.07 (0.01)	0.00 (0.00)	0.84 (0.00)	6128 (88)
NLG	0.75	0	1024	0.10 (0.01)	0.00 (0.00)	0.84 (0.00)	6101 (65)
MRILG	0.75	-0.5	16	0.02 (0.00)	0.00 (0.00)	0.84 (0.00)	6345 (33)
MRILG	0.75	-0.5	64	0.05 (0.01)	0.00 (0.00)	0.84 (0.00)	6210 (75)
MRILG	0.75	-0.5	256	0.07 (0.01)	0.00 (0.00)	0.84 (0.00)	6078 (103)
MRILG	0.75	-0.5	1024	0.09 (0.01)	0.00 (0.00)	0.84 (0.00)	6069 (53)
MRILG	0.75	-0.2	16	0.02 (0.01)	0.00 (0.00)	0.83 (0.02)	6391 (157)
MRILG	0.75	-0.2	64	0.04 (0.01)	0.00 (0.00)	0.84 (0.00)	6158 (122)
MRILG	0.75	-0.2	256	0.07 (0.01)	0.00 (0.00)	0.84 (0.00)	6103 (94)
MRILG	0.75	-0.2	1024	0.08 (0.03)	0.00 (0.00)	0.83 (0.01)	6184 (189)
MRILG	0.75	-0.05	16	0.01 (0.01)	0.00 (0.00)	0.82 (0.02)	6458 (223)
MRILG	0.75	-0.05	64	0.04 (0.02)	0.00 (0.00)	0.83 (0.02)	6291 (212)
MRILG	0.75	-0.05	256	0.06 (0.03)	0.00 (0.00)	0.83 (0.01)	6252 (233)
MRILG	0.75	-0.05	1024	0.05 (0.04)	0.00 (0.00)	0.82 (0.01)	6311 (284)

Table 39: Test accuracy with SD for different game variants, using ImageNet dataset and over 10 seeds. During test λ_{test} is set to 0.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	$\lvert\mathbb{C}\rvert$ (test)	
				1024	4096
LG (SS)	0	0	16	0.61 (0.04)	0.35 (0.04)
LG (SS)	0	0	64	0.76 (0.06)	0.52 (0.08)
LG (SS)	0	0	256	0.88 (0.03)	0.71 (0.06)
LG (SS)	0	0	1024	0.96 (0.02)	0.88 (0.04)
LG (RL)	0	0	16	0.67 (0.04)	0.39 (0.04)
LG (RL)	0	0	64	0.93 (0.01)	0.79 (0.03)
LG (RL)	0	0	256	0.98 (0.00)	0.94 (0.01)
LG (RL)	0	0	1024	0.99 (0.00)	0.97 (0.00)
NLG	0.25	0	16	0.64 (0.05)	0.36 (0.06)
NLG	0.25	0	64	0.92 (0.01)	0.77 (0.01)
NLG	0.25	0	256	0.98 (0.00)	0.94 (0.01)
NLG	0.25	0	1024	0.99 (0.00)	0.97 (0.00)
NLG	0.5	0	16	0.55 (0.03)	0.27 (0.02)
NLG	0.5	0	64	0.87 (0.01)	0.67 (0.03)
NLG	0.5	0	256	0.98 (0.00)	0.91 (0.01)
NLG	0.5	0	1024	0.99 (0.00)	0.97 (0.00)
NLG	0.75	0	16	0.34 (0.04)	0.14 (0.02)
NLG	0.75	0	64	0.75 (0.02)	0.48 (0.03)
NLG	0.75	0	256	0.94 (0.01)	0.81 (0.02)
NLG	0.75	0	1024	0.98 (0.00)	0.94 (0.01)
MRILG	0.25	-0.5	16	0.62 (0.05)	0.34 (0.05)
MRILG	0.25	-0.5	64	0.92 (0.01)	0.77 (0.03)
MRILG	0.25	-0.5	256	0.98 (0.00)	0.94 (0.01)
MRILG	0.25	-0.5	1024	0.99 (0.00)	0.97 (0.00)
MRILG	0.25	-0.2	16	0.58 (0.20)	0.33 (0.12)
MRILG	0.25	-0.2	64	0.74 (0.39)	0.62 (0.33)
MRILG	0.25	-0.2	256	0.88 (0.31)	0.84 (0.30)

Table 39: Continued from previous page.

MRILG	0.25	−0.2	1024	0.99 (0.00)	0.97 (0.00)
MRILG	0.25	−0.05	16	0.38 (0.25)	0.18 (0.14)
MRILG	0.25	−0.05	64	0.81 (0.29)	0.66 (0.24)
MRILG	0.25	−0.05	256	0.98 (0.00)	0.93 (0.01)
MRILG	0.25	−0.05	1024	0.99 (0.00)	0.96 (0.01)
MRILG	0.5	−0.5	16	0.56 (0.04)	0.29 (0.04)
MRILG	0.5	−0.5	64	0.90 (0.01)	0.72 (0.03)
MRILG	0.5	−0.5	256	0.98 (0.00)	0.92 (0.01)
MRILG	0.5	−0.5	1024	0.99 (0.00)	0.96 (0.00)
MRILG	0.5	−0.2	16	0.44 (0.23)	0.22 (0.12)
MRILG	0.5	−0.2	64	0.79 (0.28)	0.61 (0.22)
MRILG	0.5	−0.2	256	0.97 (0.01)	0.90 (0.02)
MRILG	0.5	−0.2	1024	0.99 (0.00)	0.97 (0.00)
MRILG	0.5	−0.05	16	0.31 (0.22)	0.14 (0.10)
MRILG	0.5	−0.05	64	0.69 (0.36)	0.51 (0.27)
MRILG	0.5	−0.05	256	0.78 (0.41)	0.72 (0.38)
MRILG	0.5	−0.05	1024	0.79 (0.42)	0.77 (0.41)
MRILG	0.75	−0.5	16	0.38 (0.05)	0.16 (0.03)
MRILG	0.75	−0.5	64	0.78 (0.02)	0.52 (0.04)
MRILG	0.75	−0.5	256	0.94 (0.01)	0.82 (0.02)
MRILG	0.75	−0.5	1024	0.98 (0.01)	0.93 (0.02)
MRILG	0.75	−0.2	16	0.39 (0.03)	0.16 (0.02)
MRILG	0.75	−0.2	64	0.70 (0.25)	0.47 (0.17)
MRILG	0.75	−0.2	256	0.76 (0.40)	0.67 (0.35)
MRILG	0.75	−0.2	1024	0.98 (0.00)	0.94 (0.01)
MRILG	0.75	−0.05	16	0.19 (0.18)	0.08 (0.08)
MRILG	0.75	−0.05	64	0.46 (0.39)	0.30 (0.26)
MRILG	0.75	−0.05	256	0.95 (0.01)	0.84 (0.02)
MRILG	0.75	−0.05	1024	0.62 (0.46)	0.59 (0.44)

Table 40: Test accuracy with SD for different game variants, using CelebA dataset and over 10 seeds. During test λ_{test} is set to 0.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	$\lvert\mathbb{C}\rvert$ (test)	
				1024	4096
LG (SS)	0	0	16	0.56 (0.07)	0.29 (0.06)
LG (SS)	0	0	64	0.76 (0.03)	0.51 (0.03)
LG (SS)	0	0	256	0.81 (0.03)	0.59 (0.05)
LG (SS)	0	0	1024	0.75 (0.01)	0.53 (0.02)
LG (RL)	0	0	16	0.58 (0.07)	0.28 (0.07)
LG (RL)	0	0	64	0.88 (0.02)	0.65 (0.04)
LG (RL)	0	0	256	0.95 (0.01)	0.82 (0.03)
LG (RL)	0	0	1024	0.97 (0.00)	0.87 (0.01)
NLG	0.25	0	16	0.55 (0.07)	0.26 (0.06)
NLG	0.25	0	64	0.84 (0.04)	0.60 (0.07)
NLG	0.25	0	256	0.95 (0.01)	0.82 (0.03)
NLG	0.25	0	1024	0.97 (0.00)	0.88 (0.01)
NLG	0.5	0	16	0.47 (0.04)	0.20 (0.04)
NLG	0.5	0	64	0.78 (0.05)	0.49 (0.08)
NLG	0.5	0	256	0.93 (0.01)	0.77 (0.03)
NLG	0.5	0	1024	0.96 (0.00)	0.87 (0.01)
NLG	0.75	0	16	0.30 (0.07)	0.11 (0.04)
NLG	0.75	0	64	0.67 (0.06)	0.37 (0.06)
NLG	0.75	0	256	0.87 (0.03)	0.65 (0.05)
NLG	0.75	0	1024	0.95 (0.01)	0.83 (0.02)
MRILG	0.25	-0.5	16	0.55 (0.05)	0.25 (0.05)
MRILG	0.25	-0.5	64	0.86 (0.03)	0.62 (0.06)
MRILG	0.25	-0.5	256	0.94 (0.01)	0.81 (0.03)
MRILG	0.25	-0.5	1024	0.97 (0.00)	0.88 (0.01)
MRILG	0.25	-0.2	16	0.54 (0.05)	0.24 (0.05)
MRILG	0.25	-0.2	64	0.76 (0.27)	0.54 (0.21)
MRILG	0.25	-0.2	256	0.85 (0.30)	0.73 (0.26)

Table 40: Continued from previous page.

MRILG	0.25	−0.2	1024	0.97 (0.00)	0.87 (0.01)
MRILG	0.25	−0.05	16	0.47 (0.19)	0.21 (0.10)
MRILG	0.25	−0.05	64	0.76 (0.27)	0.53 (0.20)
MRILG	0.25	−0.05	256	0.94 (0.01)	0.80 (0.03)
MRILG	0.25	−0.05	1024	0.96 (0.01)	0.87 (0.02)
MRILG	0.5	−0.5	16	0.49 (0.05)	0.21 (0.04)
MRILG	0.5	−0.5	64	0.82 (0.04)	0.56 (0.06)
MRILG	0.5	−0.5	256	0.93 (0.01)	0.78 (0.03)
MRILG	0.5	−0.5	1024	0.96 (0.00)	0.87 (0.01)
MRILG	0.5	−0.2	16	0.46 (0.10)	0.20 (0.07)
MRILG	0.5	−0.2	64	0.72 (0.26)	0.48 (0.18)
MRILG	0.5	−0.2	256	0.94 (0.01)	0.79 (0.03)
MRILG	0.5	−0.2	1024	0.96 (0.01)	0.86 (0.02)
MRILG	0.5	−0.05	16	0.34 (0.22)	0.15 (0.10)
MRILG	0.5	−0.05	64	0.68 (0.24)	0.42 (0.16)
MRILG	0.5	−0.05	256	0.80 (0.30)	0.64 (0.27)
MRILG	0.5	−0.05	1024	0.46 (0.44)	0.40 (0.40)
MRILG	0.75	−0.5	16	0.32 (0.02)	0.11 (0.01)
MRILG	0.75	−0.5	64	0.71 (0.06)	0.41 (0.07)
MRILG	0.75	−0.5	256	0.88 (0.02)	0.66 (0.05)
MRILG	0.75	−0.5	1024	0.94 (0.01)	0.81 (0.03)
MRILG	0.75	−0.2	16	0.23 (0.15)	0.08 (0.06)
MRILG	0.75	−0.2	64	0.65 (0.07)	0.35 (0.07)
MRILG	0.75	−0.2	256	0.88 (0.03)	0.67 (0.05)
MRILG	0.75	−0.2	1024	0.85 (0.30)	0.74 (0.26)
MRILG	0.75	−0.05	16	0.17 (0.19)	0.06 (0.07)
MRILG	0.75	−0.05	64	0.57 (0.30)	0.34 (0.18)
MRILG	0.75	−0.05	256	0.71 (0.38)	0.55 (0.30)
MRILG	0.75	−0.05	1024	0.53 (0.46)	0.45 (0.40)

Table 41: Test accuracy with SD for different game variants, using ImageNet dataset and over 10 seeds. During test λ_{test} is set to 0.25.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	$\lvert\mathbb{C}\rvert$ (test)	
				1024	4096
LG (SS)	0	0	16	0.17 (0.03)	0.08 (0.02)
LG (SS)	0	0	64	0.21 (0.03)	0.11 (0.02)
LG (SS)	0	0	256	0.27 (0.05)	0.17 (0.03)
LG (SS)	0	0	1024	0.30 (0.05)	0.20 (0.04)
LG (RL)	0	0	16	0.20 (0.04)	0.10 (0.02)
LG (RL)	0	0	64	0.38 (0.10)	0.24 (0.06)
LG (RL)	0	0	256	0.46 (0.10)	0.33 (0.09)
LG (RL)	0	0	1024	0.48 (0.06)	0.36 (0.05)
NLG	0.25	0	16	0.52 (0.04)	0.26 (0.04)
NLG	0.25	0	64	0.82 (0.01)	0.60 (0.02)
NLG	0.25	0	256	0.93 (0.01)	0.82 (0.02)
NLG	0.25	0	1024	0.97 (0.00)	0.90 (0.01)
MRILG	0.25	-0.5	16	0.50 (0.04)	0.24 (0.03)
MRILG	0.25	-0.5	64	0.81 (0.02)	0.60 (0.03)
MRILG	0.25	-0.5	256	0.93 (0.01)	0.81 (0.02)
MRILG	0.25	-0.5	1024	0.97 (0.01)	0.90 (0.02)
MRILG	0.25	-0.2	16	0.47 (0.16)	0.24 (0.08)
MRILG	0.25	-0.2	64	0.65 (0.34)	0.48 (0.26)
MRILG	0.25	-0.2	256	0.84 (0.29)	0.73 (0.26)
MRILG	0.25	-0.2	1024	0.96 (0.01)	0.89 (0.02)
MRILG	0.25	-0.05	16	0.30 (0.20)	0.13 (0.10)
MRILG	0.25	-0.05	64	0.71 (0.25)	0.51 (0.19)
MRILG	0.25	-0.05	256	0.92 (0.02)	0.79 (0.03)
MRILG	0.25	-0.05	1024	0.93 (0.02)	0.85 (0.04)

Table 42: Test accuracy with SD for different game variants, using CelebA dataset and over 10 seeds. During test λ_{test} is set to 0.25.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	$\lvert\mathbb{C}\rvert$ (test)	
				1024	4096
LG (SS)	0	0	16	0.14 (0.02)	0.06 (0.01)
LG (SS)	0	0	64	0.21 (0.03)	0.11 (0.01)
LG (SS)	0	0	256	0.21 (0.04)	0.12 (0.02)
LG (SS)	0	0	1024	0.19 (0.03)	0.10 (0.02)
LG (RL)	0	0	16	0.18 (0.06)	0.07 (0.03)
LG (RL)	0	0	64	0.29 (0.05)	0.16 (0.03)
LG (RL)	0	0	256	0.36 (0.06)	0.22 (0.04)
LG (RL)	0	0	1024	0.41 (0.05)	0.27 (0.04)
NLG	0.25	0	16	0.43 (0.06)	0.18 (0.04)
NLG	0.25	0	64	0.71 (0.05)	0.44 (0.06)
NLG	0.25	0	256	0.87 (0.02)	0.67 (0.04)
NLG	0.25	0	1024	0.92 (0.01)	0.77 (0.03)
MRILG	0.25	-0.5	16	0.43 (0.04)	0.18 (0.03)
MRILG	0.25	-0.5	64	0.73 (0.04)	0.46 (0.05)
MRILG	0.25	-0.5	256	0.86 (0.03)	0.65 (0.05)
MRILG	0.25	-0.5	1024	0.92 (0.01)	0.78 (0.03)
MRILG	0.25	-0.2	16	0.42 (0.05)	0.17 (0.03)
MRILG	0.25	-0.2	64	0.64 (0.23)	0.39 (0.15)
MRILG	0.25	-0.2	256	0.77 (0.27)	0.59 (0.21)
MRILG	0.25	-0.2	1024	0.90 (0.02)	0.75 (0.03)
MRILG	0.25	-0.05	16	0.36 (0.15)	0.15 (0.07)
MRILG	0.25	-0.05	64	0.64 (0.23)	0.39 (0.15)
MRILG	0.25	-0.05	256	0.83 (0.03)	0.63 (0.05)
MRILG	0.25	-0.05	1024	0.86 (0.04)	0.72 (0.06)

Table 43: Test accuracy with SD for different game variants, using ImageNet dataset and over 10 seeds. During test λ_{test} is set to 0.5.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	$\lvert\mathbb{C}\rvert$ (test)	
				1024	4096
LG (SS)	0	0	16	0.03 (0.01)	0.01 (0.00)
LG (SS)	0	0	64	0.04 (0.01)	0.02 (0.01)
LG (SS)	0	0	256	0.06 (0.02)	0.03 (0.01)
LG (SS)	0	0	1024	0.05 (0.01)	0.03 (0.01)
LG (RL)	0	0	16	0.03 (0.01)	0.01 (0.01)
LG (RL)	0	0	64	0.09 (0.07)	0.05 (0.03)
LG (RL)	0	0	256	0.11 (0.06)	0.06 (0.04)
LG (RL)	0	0	1024	0.11 (0.02)	0.06 (0.01)
NLG	0.5	0	16	0.32 (0.02)	0.14 (0.01)
NLG	0.5	0	64	0.57 (0.02)	0.33 (0.02)
NLG	0.5	0	256	0.73 (0.01)	0.54 (0.02)
NLG	0.5	0	1024	0.82 (0.01)	0.68 (0.02)
MRILG	0.5	-0.5	16	0.33 (0.03)	0.14 (0.02)
MRILG	0.5	-0.5	64	0.59 (0.02)	0.35 (0.02)
MRILG	0.5	-0.5	256	0.75 (0.02)	0.55 (0.03)
MRILG	0.5	-0.5	1024	0.81 (0.01)	0.66 (0.02)
MRILG	0.5	-0.2	16	0.26 (0.14)	0.11 (0.06)
MRILG	0.5	-0.2	64	0.50 (0.18)	0.29 (0.10)
MRILG	0.5	-0.2	256	0.72 (0.02)	0.51 (0.03)
MRILG	0.5	-0.2	1024	0.80 (0.01)	0.65 (0.02)
MRILG	0.5	-0.05	16	0.16 (0.12)	0.06 (0.05)
MRILG	0.5	-0.05	64	0.40 (0.21)	0.22 (0.12)
MRILG	0.5	-0.05	256	0.56 (0.30)	0.40 (0.21)
MRILG	0.5	-0.05	1024	0.62 (0.33)	0.50 (0.26)

Table 44: Test accuracy with SD for different game variants, using CelebA dataset and over 10 seeds. During test λ_{test} is set to 0.5.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	$\lvert\mathbb{C}\rvert$ (test)	
				1024	4096
LG (SS)	0	0	16	0.03 (0.01)	0.01 (0.00)
LG (SS)	0	0	64	0.04 (0.01)	0.02 (0.01)
LG (SS)	0	0	256	0.04 (0.01)	0.02 (0.01)
LG (SS)	0	0	1024	0.03 (0.01)	0.01 (0.01)
LG (RL)	0	0	16	0.04 (0.02)	0.01 (0.01)
LG (RL)	0	0	64	0.05 (0.02)	0.02 (0.01)
LG (RL)	0	0	256	0.07 (0.03)	0.03 (0.01)
LG (RL)	0	0	1024	0.08 (0.02)	0.04 (0.01)
NLG	0.5	0	16	0.26 (0.03)	0.09 (0.02)
NLG	0.5	0	64	0.46 (0.05)	0.22 (0.04)
NLG	0.5	0	256	0.64 (0.04)	0.40 (0.04)
NLG	0.5	0	1024	0.75 (0.03)	0.55 (0.04)
MRILG	0.5	-0.5	16	0.27 (0.03)	0.10 (0.02)
MRILG	0.5	-0.5	64	0.50 (0.05)	0.25 (0.04)
MRILG	0.5	-0.5	256	0.65 (0.03)	0.40 (0.03)
MRILG	0.5	-0.5	1024	0.74 (0.03)	0.53 (0.05)
MRILG	0.5	-0.2	16	0.26 (0.06)	0.09 (0.03)
MRILG	0.5	-0.2	64	0.44 (0.16)	0.22 (0.09)
MRILG	0.5	-0.2	256	0.66 (0.03)	0.42 (0.04)
MRILG	0.5	-0.2	1024	0.72 (0.03)	0.52 (0.05)
MRILG	0.5	-0.05	16	0.17 (0.11)	0.06 (0.04)
MRILG	0.5	-0.05	64	0.36 (0.13)	0.17 (0.07)
MRILG	0.5	-0.05	256	0.52 (0.21)	0.31 (0.14)
MRILG	0.5	-0.05	1024	0.33 (0.32)	0.23 (0.23)

Table 45: Test accuracy with SD for different game variants, using ImageNet dataset and over 10 seeds. During test λ_{test} is set to 0.75.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	$\lvert\mathbb{C}\rvert$ (test)	
				1024	4096
LG (SS)	0	0	16	0.00 (0.00)	0.00 (0.00)
LG (SS)	0	0	64	0.01 (0.00)	0.00 (0.00)
LG (SS)	0	0	256	0.01 (0.00)	0.00 (0.00)
LG (SS)	0	0	1024	0.01 (0.00)	0.00 (0.00)
LG (RL)	0	0	16	0.00 (0.00)	0.00 (0.00)
LG (RL)	0	0	64	0.01 (0.02)	0.01 (0.01)
LG (RL)	0	0	256	0.01 (0.01)	0.01 (0.01)
LG (RL)	0	0	1024	0.01 (0.00)	0.01 (0.00)
NLG	0.75	0	16	0.10 (0.01)	0.04 (0.00)
NLG	0.75	0	64	0.20 (0.01)	0.09 (0.01)
NLG	0.75	0	256	0.30 (0.01)	0.16 (0.01)
NLG	0.75	0	1024	0.36 (0.01)	0.23 (0.01)
MRILG	0.75	-0.5	16	0.11 (0.01)	0.04 (0.00)
MRILG	0.75	-0.5	64	0.22 (0.01)	0.10 (0.01)
MRILG	0.75	-0.5	256	0.31 (0.01)	0.17 (0.01)
MRILG	0.75	-0.5	1024	0.38 (0.02)	0.23 (0.02)
MRILG	0.75	-0.2	16	0.11 (0.01)	0.04 (0.00)
MRILG	0.75	-0.2	64	0.20 (0.07)	0.09 (0.03)
MRILG	0.75	-0.2	256	0.25 (0.13)	0.13 (0.07)
MRILG	0.75	-0.2	1024	0.37 (0.01)	0.23 (0.01)
MRILG	0.75	-0.05	16	0.05 (0.05)	0.02 (0.02)
MRILG	0.75	-0.05	64	0.12 (0.11)	0.05 (0.05)
MRILG	0.75	-0.05	256	0.31 (0.01)	0.17 (0.01)
MRILG	0.75	-0.05	1024	0.23 (0.17)	0.14 (0.11)

Table 46: Test accuracy with SD for different game variants, using CelebA dataset and over 10 seeds. During test λ_{test} is set to 0.75.

Game	λ	ν	$\lvert\mathbb{C}\rvert$	$\lvert\mathbb{C}\rvert$ (test)	
				1024	4096
LG (SS)	0	0	16	0.00 (0.00)	0.00 (0.00)
LG (SS)	0	0	64	0.00 (0.00)	0.00 (0.00)
LG (SS)	0	0	256	0.00 (0.00)	0.00 (0.00)
LG (SS)	0	0	1024	0.01 (0.00)	0.00 (0.00)
LG (RL)	0	0	16	0.01 (0.00)	0.00 (0.00)
LG (RL)	0	0	64	0.01 (0.00)	0.00 (0.00)
LG (RL)	0	0	256	0.01 (0.00)	0.00 (0.00)
LG (RL)	0	0	1024	0.01 (0.00)	0.00 (0.00)
NLG	0.75	0	16	0.09 (0.02)	0.03 (0.01)
NLG	0.75	0	64	0.17 (0.02)	0.07 (0.01)
NLG	0.75	0	256	0.25 (0.02)	0.12 (0.02)
NLG	0.75	0	1024	0.33 (0.02)	0.19 (0.02)
MRILG	0.75	-0.5	16	0.09 (0.01)	0.03 (0.00)
MRILG	0.75	-0.5	64	0.20 (0.03)	0.08 (0.01)
MRILG	0.75	-0.5	256	0.28 (0.02)	0.13 (0.01)
MRILG	0.75	-0.5	1024	0.33 (0.02)	0.18 (0.02)
MRILG	0.75	-0.2	16	0.07 (0.04)	0.02 (0.01)
MRILG	0.75	-0.2	64	0.18 (0.02)	0.06 (0.01)
MRILG	0.75	-0.2	256	0.27 (0.02)	0.12 (0.02)
MRILG	0.75	-0.2	1024	0.30 (0.11)	0.16 (0.06)
MRILG	0.75	-0.05	16	0.05 (0.05)	0.01 (0.02)
MRILG	0.75	-0.05	64	0.15 (0.08)	0.06 (0.03)
MRILG	0.75	-0.05	256	0.22 (0.12)	0.11 (0.06)
MRILG	0.75	-0.05	1024	0.18 (0.15)	0.10 (0.09)