Latent Inter-Agent Communication Through Single Hidden State Extraction

Ertugrul Gul

University of Sydney

CSYS5061: Complex Systems Research Project B

13 Jun 2026

github.com/Jazzamat/latent-communication/

Abstract

In recent years, agentic workflows powered by large language models (LLMs) have become central to many professional workflows. However, a significant barrier to the scalability of multi-agent systems is the financial and computational cost of natural-language-based communication, which drives up token usage and expands context windows, often degrading system performance. This paper explores latent communication as a compact, efficient alternative to traditional text-based exchange. We implement a lightweight language model, TinyLM, and develop a sender-receiver architecture where agents communicate by transmitting only their final hidden state—a dense vector representation of the preceding sequence—rather than natural language tokens. We evaluate this approach using a synthetic Sally-Anne Theory of Mind dataset. Our experiments demonstrate that the extracted hidden state successfully encapsulates task-relevant information, allowing the receiver agent to achieve high accuracy in answering queries, significantly outperforming zero-message and shuffled baselines. While our results indicate that this method serves as a viable, compact communication medium, we acknowledge limitations such as potential overfitting and the simplified nature of our experimental setup. These findings offer a promising direction for reducing token consumption in multi-agent systems, providing a foundation for future research into scalability, more complex reasoning tasks, and information stability across multi-agent chains.

1. Introduction

In recent years our usage of large language models (LLMs) has evolved into agentic systems. These systems loop around what is essentially the foundational technology, the LLM, and allow the system to behave somewhat autonomously compared to the simple chatbox style workflow. An Agent can execute tools on the users behalf and assemble extra context it deems necessary. With this agentic loop systems can complete a higher magnitude of tasks without direct instruction, however the cost of running these systems can be high (Bai et al., 2026; Wang et al., 2023; Bommasani et al., 2021; Brown et al., 2020).

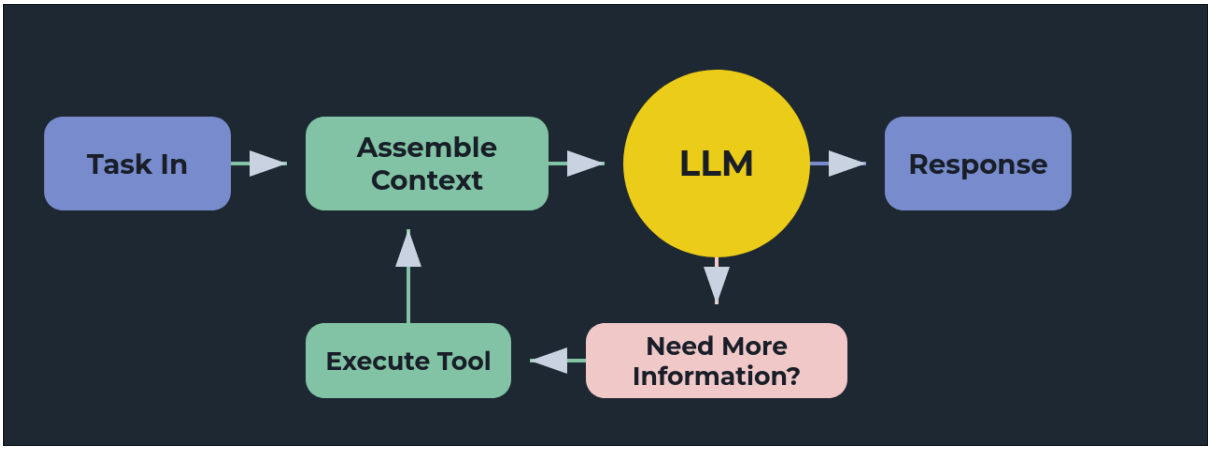

Figure 1: Illustration of an example of an agentic workflow.

The large language model performs computation per each input token (character, word or part of a word) provided and companies such as OpenAI or Anthropic charge for the usage of their models per token. (OpenAI, 2026; Anthropic, 2026)

A major inhibiting factor when it comes to the use of agentic systems is token cost. As the agents assemble more and more information their context window increases in size, thus the amount of tokens consumed increases. This is especially true for multi-agent systems.

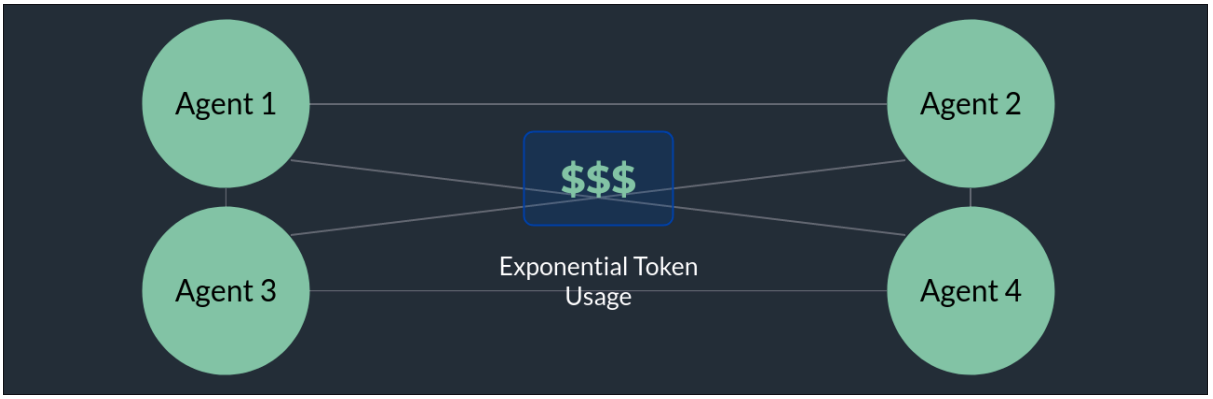

Figure 2: An illustration of communication in a multi-agent system.

As agents pass around information in natural language, the token size of the communication medium can quickly diverge into large amounts of money spent as each agent will now deal with a communication history in their context window for each agent they are talking to. As well as the increased token cost, a larger context window can result in lower performance of the whole system, as this is a well known limitation of LLMs (Squirrel, 2026; Liu et al., 2024).

In this paper we explore latent communication as an alternative communication method to natural language text. With the goal of exploring how latent communication can help mitigate the problem of increased token usage in multi-agent systems we extend upon previous works done in the field.

2. Literature Review

2.1 Interlat

Inter-agent Latent Space Communication (Interlat) proposed by Zhuoyun Du et al. is a paradigm that leverages the continuous last hidden states of an LLM as a representation of its thought for direct communication. They demonstrate that Interlat outperforms both fine-tuned chain-of-thought (CoT) prompting and single-agent baselines, even across heterogeneous models, promoting more exploratory behavior and en-abling genuine utilization of latent information (Du et al., 2026). Extracting and compressing the hidden states of the sending entity to be input for the receiving entity they demonstrate that hidden state vectors can be an alternative to communication with natural language tokens

2.2 LMNet

Similar to Interlat LMNet proposed by Wu et al. exchanges hidden states and uses stripped LLMs as vertex modules and trainable seq2seq modules as communication edges, enabling intermediate nodes to exchange dense vectors while preserving natural-language input and output at the system boundary (Wu et al., 2025). They demonstrate that pre-trained language models can be reused as computational nodes, and intelligence can be improved by learning how information flows among them.

2.3 Communicating activations

Compared to Interlat and LMNet in the study by Ramesh et al. a technique is proposed whereby activations are injected into intermediate layers of the transformer architecture. This is a lower level approach compared to Interlat and LMNet which use the outputs of the transformer (hidden states) rather than modifying weights at the intermediate layer.

2.4 Our work

In this work we propose to extract only the last hidden state from the *sending* entity to form the prefix of the message received by the *receiving* entity. We do this by mimicking Interlat and LMNet in their approach to use hidden states as communication, but forming a more compact and rudimentary exchange of information. As the last hidden state contains a representation of the input sequence thanks to causal attention we demonstrate that it can act as a compact alternative to language based communication as well as previous methods of hidden state exchange.

3. Background Information

3.1 The Transformer

The modern large language model is based on the Transformer architecture proposed by Vaswani et al. in their now famous paper *Attention is All You Need* in which they dispense with the complex recurrent or convolutional neural networks previously used and present a highly parallelisable method based solely on attention (Vaswani et al., 2017). The paper's publication triggered a massive paradigm shift in computer science as the architecture became the direct ancestor of the most powerful language models in use today such as GPT (The engine behind OpenAI's ChatGPT) and BERT (Google's landmark bidirectional encoder, which vastly improved search and text understanding).

3.1.1 Embeddings

In order to represent tokens (characters, words or parts of a word) mathematically we use embeddings. Since transformers operate on vectors rather than raw text, the input text is first tokenised into token IDs, and each token ID is mapped to a corresponding token embedding. The embeddings are obtained from a learned embedding matrix, where each row corresponds to one token in the model's vocabulary. When a sequence of token IDs is provided to the model, the embedding layer retrieves the vector associated with each token. These vectors are learned during training and are updated through backpropagation. To represent what a character is and where it is in the sequence we combine the token embedding with the position embedding (Vaswani et al., 2017; Bengio et al., 2003). For an example input sequence *"cat"* where per character encoding is used, each letter will be represented by an embedding.

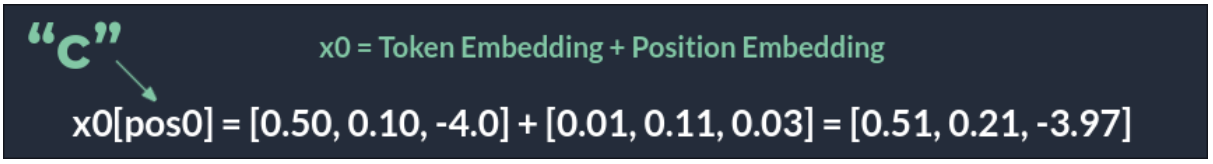

Figure 3: An example embedding of the character c.

3.1.2 Attention

The central part of the transformer architecture is the attention module. After the embedding layer we obtain a sequence of vectors representing each token; however each vector represents a token independently and does not contain any information about the sequence as a whole. The attention module is what allows each token in a sequence to selectively focus on previous tokens when building its understanding of the context. What we obtain is a final "condensed" embedding that accurately represents the concept being modeled. For the example input sequence "Miniature Eiffel Tower" (per word encoding), after the embedding step we obtain an embedding for each word. These vectors convey the meaning of each word separately "Miniature", "Eiffel" and "Tower", but they don't convey the meaning of what is actually meant by that sequence of words; A *Miniature Eiffel Tower.*

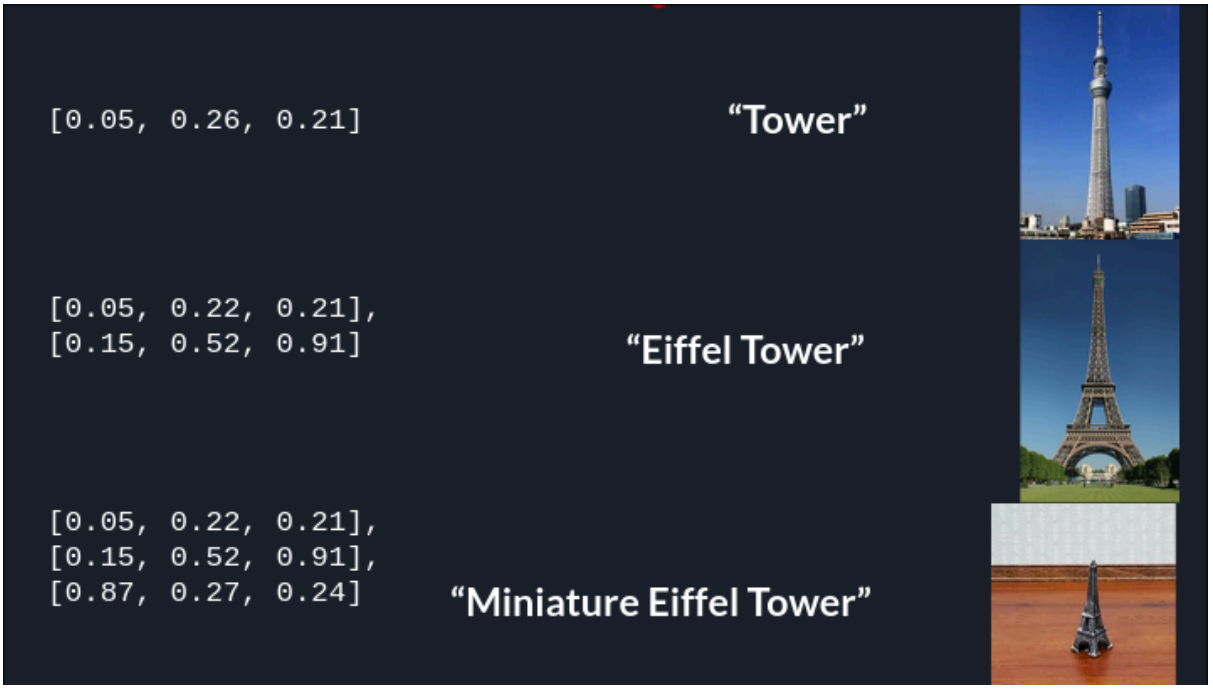

Figure 4: An example embedding of the character c.

It's only after the attention module that we obtain a vector representing a *Miniature Eiffel Tower*. The attention module allows us to look backwards in the sequence and “ingest” the context of the sequence up until that point, illustrated in figure 5

Figure 5: Illustration of condensed/ingested meaning into a single vector.

It does this by projecting each token representation into query Q, key K, and value V vectors, computing masked attention scores between each token and the previous tokens for dimensions dk, and then forming a normalised weighted sum of the corresponding value vectors as shown in figure 6 *(Vaswani et al., 2026)*.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Figure 6: Scaled Dot-Product Attention (Vaswani et al., 2026).

As a result we obtain a hidden state vector for each embedding as can be seen in figure 7.

Figure 7: Illustration of hidden states.

3.1.3 Hidden States

In modern transformer models, a hidden state refers to the internal vector representation produced for each token at each layer of the network. As there are multiple iterations of the transformer block in most LLM architectures, what we refer to as hidden states are the last set of outputs of the transformer block before the next module. The final hidden state for each token is a contextual representation of that token after the model has processed the sequence.

Thus, it's only logical for us to now explore what can be done with the hidden state of the last token of the sequence. If indeed it is a conceptual representation of that token and the whole preceding sequence, there is room to believe it can function as a compressed message between a sender and a receiver given that they *speak the same language,* or are trained on the same activations. Afterall, instead of passing around token heavy natural language messages, converting each token to an embedding and processing each of them, we should be able to simply use a single information rich vector to convey the same information. What follows is a set of implementations and experiments to test this hypothesis and see how well the last tokens hidden state can serve us in latent communication.

4. Experiments and Results

Our methodology follows an iterative approach where we start with a basic implementation of a language model and use that as a baseline for further experimentation. In order to obtain hidden states for a given sequence of tokens we must first have a language model to produce them. We choose not to use flagship models from large providers like OpenAI or Anthropic as they are essentially black-box systems where obtaining intermediate hidden states is not possible. We also avoid large and complicated open source models like Llama or DeepSeek. Instead, following along an annotated version of the famous *attention* paper *The Annotated Transformer (Rush, 2018)* we implement our language model from scratch. This is a challenge in and of itself but it gives us the most control over the components of our model and is easier to modify, observe and extract and deal with the desired hidden state vectors. All code may be found in our github repository: github.com/Jazzamat/latent-communication/

4.1 v0: TinyLM

To form our baseline we implement TinyLM. The architecture consists of an embedding module followed by *N* transformer blocks, followed by a LM Head Mapping Module.

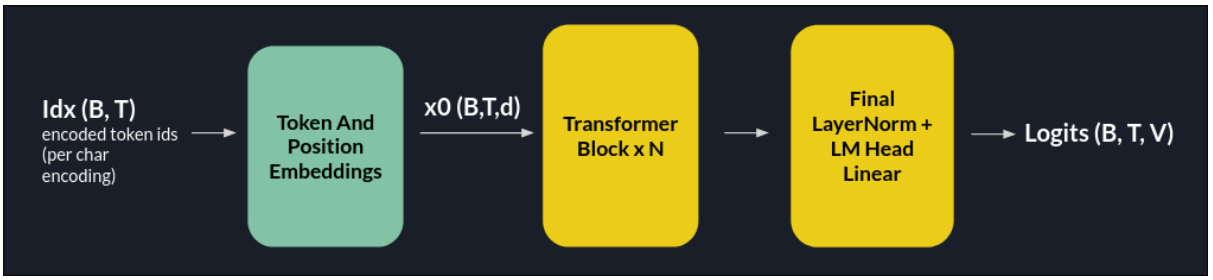

Figure 8: TinyLM architecture, batch size B, sequence length T, vocabulary V, number of transformer blocks N.

The transformer block contains a causal self attention block module and a multi-layer perceptron along with intermediate layer normalisation for well behaved activations as can be seen in figure 9.

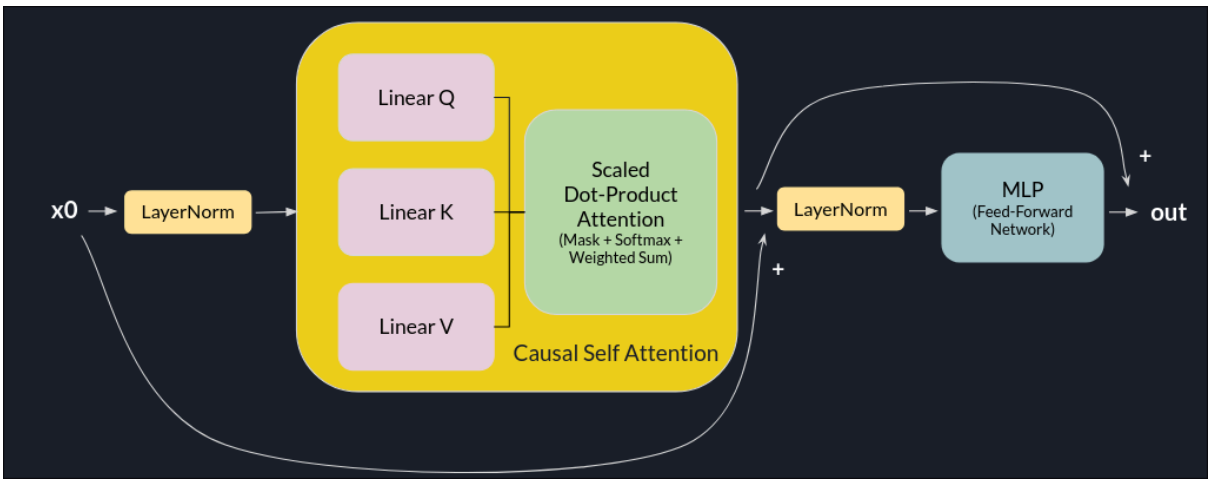

Figure 9: Transformer Block architecture.

In order to train and evaluate our model we prepare a training set of basic Sally-Anne Theory of Mind questions and answers as is common in large language model testing and even human cognition studies. (Table 1) (Kosinski, 2024; Ma et al., 2023; Wimmer & Perner, 1983). It's worth mentioning that our goal is not to develop a large model that can prove it has a theory of mind nor does our model have to be generalisable. It's sufficient to demonstrate a limited capability in answering questions from our training set for the scope of this study.

Table 1. Theory of Mind Sally-Anne training set sample

Question: Sally puts a marble in the basket. Anne moves it to the box while Sally is away. Where will Sally look first?
Answer: Sally will look in the basket first.

We use an Adam optimizer with a batch size of 16 and 500 training epochs. The learning rate was set to 2e-3, with 0.01 weight decay. Training was performed on GPU using cross-entropy loss.

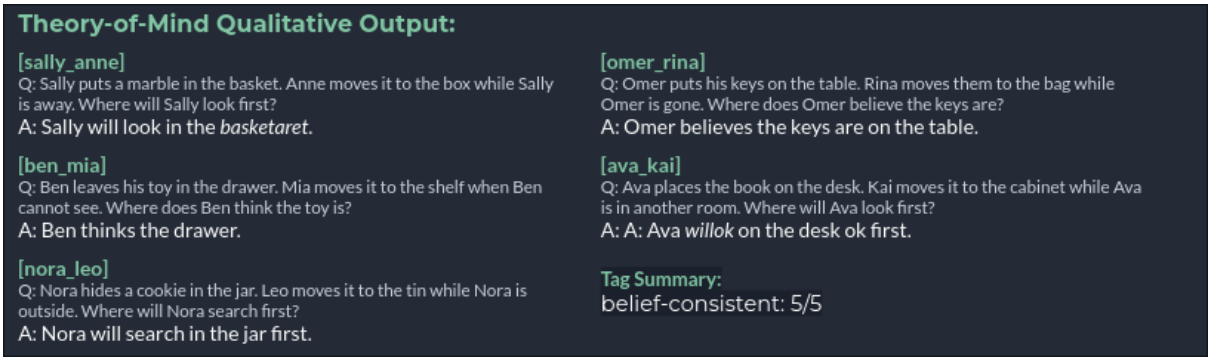

Figure 10: v0: TinyLM Theory-of-Mind Qualitative Results .

As can be seen in figure 10, the results of our experiment for v0 demonstrates a basic capability to answer Sally-Anne questions with answers being consistent with the questions provided. Spelling mistakes such as “basketaret” or “willok” likely appear as a consequence of the character-level output representation used in the model. If we were to use per word encodings these variations would cause the model to choose different words instead. Since words are larger probabilistic buckets compared to characters, which are more granular, we suspect small variations in the weights would not cause as many mistakes as per character encodings.

This reflects a known trade-off in tokenization: character-level models, while they offer some benefits like language agnostic learning and avoid out-of-vocabulary limitations, they generate longer sequences with less semantic information per token, making exact lexical reconstruction more fragile. Word and subword vocabularies impose stronger lexical constraints, reducing malformed spellings but introducing their own limitations, such as fixed vocabulary coverage and reduced character-level flexibility (Xue et al., 2022; Clark et al., 2022)

4.2 v1: Sender-Receiver

In this next experiment we explore a sender and receiver setup. Using TinyLM, developed in v0, as our language model we initialise two agents that will communicate in latent space to answer a Sally-Anne question.

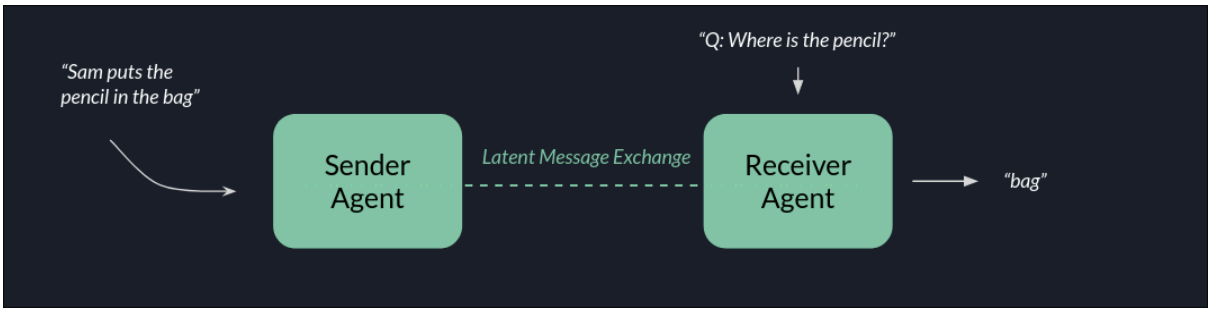

Figure 11: v1: Sender And Receiver setup illustration.

Our goal here is to observe if a receiver agent, upon receiving latent communication from the sender agent, can answer the question correctly versus if it had not received the message. This will test our method of latent communication. To achieve this we perform last hidden state extraction followed by a concatenation to the receiver's input.

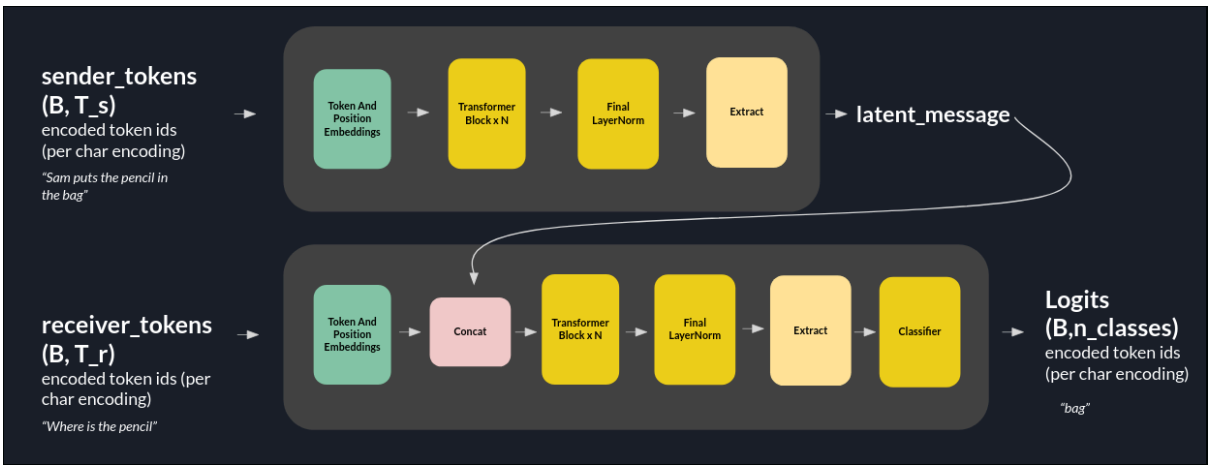

Figure 12: v1: Sender And Receiver Architecture, with batch size B, sender sequence length T_s receiver sequence length T_r, embedding dimensions d and number of answer classes n.

As can be seen in figure 12, we modify the tail end of TinyLM to extract the last hidden state, rather than map them to language tokens as we did in v0. The output of the sender is a single hidden state vector with dimensions n that is concatenated into the receiver's sequence of embedding vectors of dimensions n as a prefix as can be seen in figure 13 and equation 1. For simplicity we use a classifier for the output of the receiver so it can give us a probability of the correct container (location of the object) rather than raw text output.

$$latent\ message = hiddenstate_{sender} \oplus embeddings_{receiver}$$
$$latent\ message = (p0, v0, v1, \ldots,)$$

Equation 1: v1: Message sent to receiver.

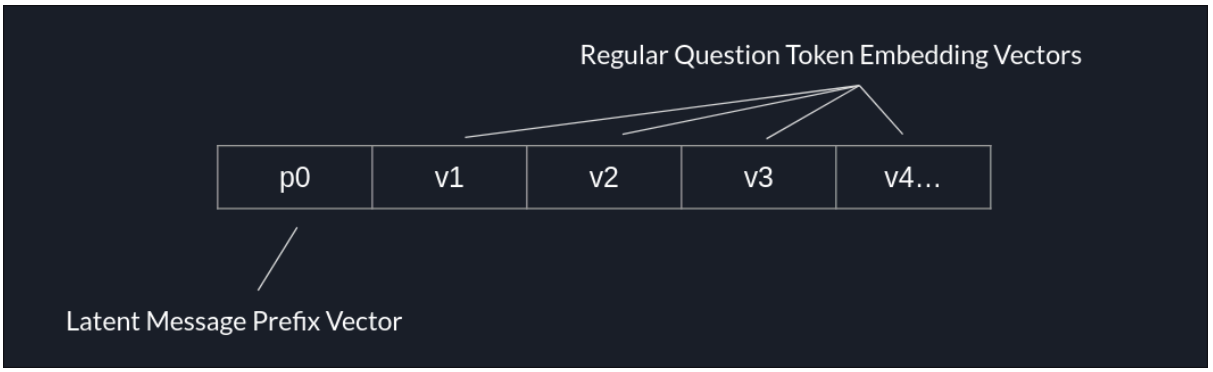

Figure 13: v1:Message prefix illustration.

For our training and inference we again use a Sally-Anne data set. We split the scenarios such that the information pertaining to the scenario is given to the sender and the question is posed to the receiver. The sender must convey the correct information to the receiver such that they can answer the question and guess the correct container.

Table 2. Sender and Receiver Theory of Mind Sally-Anne training set sample

Input to Sender: Sally puts a marble in the basket. Anne moves it to the box while Sally is away.
Question Posed to Receiver: Where will Sally look first?

In order to measure the effect of the latent message we run the experiment in **with_message**, **zero_message**, and **shuffled_message** modes. Where in the first mode the receiver is given the latent message from the sender, in the second it is given an empty message (vector values set to zero) and in the third a shuffled vector with randomly initialised values. This allows us to have statistical confidence that the latent message has an effect in our setup.

We trained the model for 5 epochs using the Adam optimizer with a batch size of 64. The learning rate was set to 1e-3, with no weight decay. Training was performed on GPU using cross-entropy loss.

Table 3. Message Modes

with_message: The receiver is given the last hidden state extracted from the receiver
zero_message: The receiver is given a zero vector
shuffled_message: The receiver is given a random vector

As can be seen from sample outputs in figure 14 the experiment is able to produce the desired results. The **with_message** mode demonstrates clear accuracy compared to the other modes. The receiver is able to correctly guess where the object is given the latent message. Given a sample size of 2000 test runs our receiver is able to guess the position of the object 100, 25 and 26 percent of the

time in **with_message**, **zero_message**, **shuffled_message** modes respectively, as can be seen in table 4.

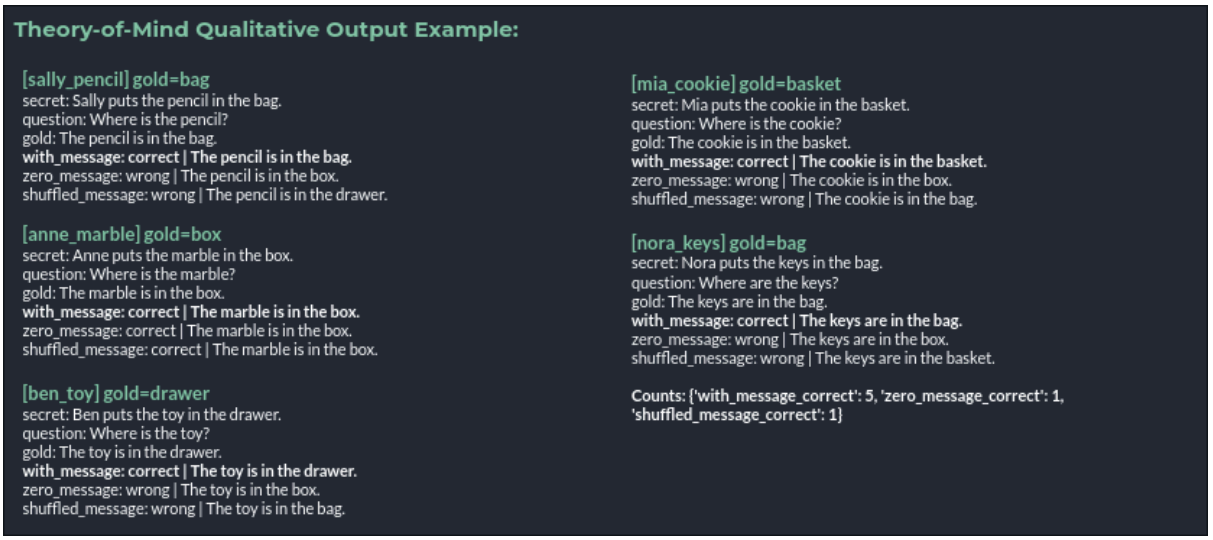

Figure 14: v1:Message prefix illustration.

Table 4. V1 Results out of 2000 samples

with_message: 100% accuracy
zero_message: ~25% accuracy
shuffled_message: ~26% accuracy

The results of v1 provide strong evidence that the last hidden state carries relevant information and is sufficient to serve as latent communication, while there are some obvious limitations. Without the latent message, the accuracy drops to 1/5th demonstrating that the hidden state carries the relevant information, however a 100 percent accuracy in the **with_message** mode also sheds light on the fact that the model is likely overfitted.

Another limitation of v1 is that the sender is simply acting as a message relay and is not performing any additional reasoning on the information as might be typical in agentic workflows. To better simulate a real-life agentic scenario the sender may be given an additional task of processing the information before it communicates with the sender. The interpretability of such a setup may be difficult so a periodic probing of the internal of the agent's reasoning may be required.

4.3 Future Work, v2: Telephone Game

The next proposed experiment is a telephone-game setup. By simulating a multi-agent chain, we can observe how well single-hidden-state communication survives repeated transmission through a sequence of senders and receivers. This would allow us to test whether latent messages remain stable across multiple communication steps, or whether information degrades as it is repeatedly transformed by different agents.

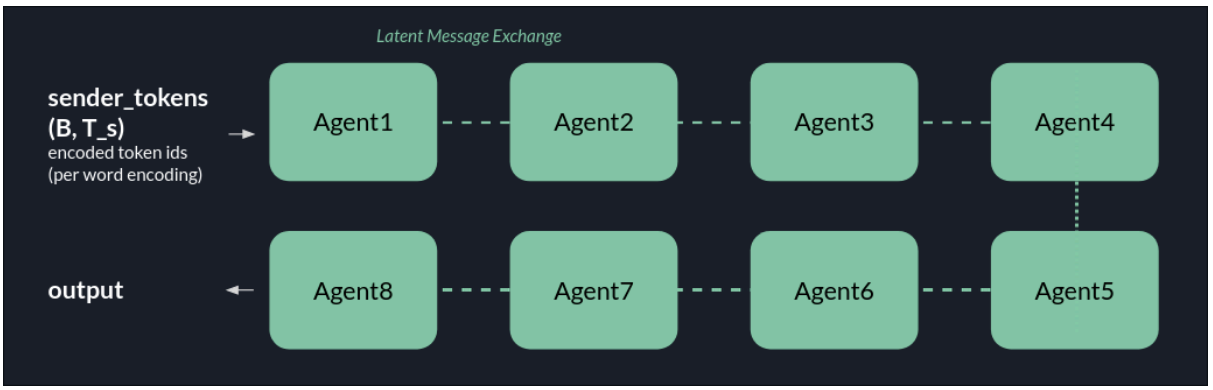

Figure 15: v2: Telephone Game illustration.

5. Conclusion

In this project we explored whether latent communication through single hidden state extraction can function as an alternative to natural language communication between large language model based agents. Motivated by the increasing cost of agentic and multi-agent systems, we investigated whether information normally transmitted through natural language could instead be transmitted through a single dense hidden state vector. We first implemented TinyLM as a language model baseline and demonstrated that it could learn to answer simple Sally-Anne Theory of Mind questions. While the model showed basic task competence, the character-level output representation produced spelling errors, highlighting the limitations of the baseline architecture and tokenisation strategy. We then extended this to a sender-receiver setup architecture, in which the sender communicated only its final hidden state to the receiver, which was then required to answer the question. The results showed a clear difference between the **with_message** condition and the **zero_message** and **shuffled_message** baselines. This proves that, within this basic experimental setting, the final hidden state can carry enough task-relevant information and can be used as a latent communication channel.

However these results should be interpreted carefully. The task is narrow, the dataset is synthetic and the perfect accuracy in the **with_message** condition suggests that the model may be overfitted to the specific structure of the experiment. In addition the sender currently acts mainly as a message relay rather than as an agent performing independent reasoning, therefore the experiment does not yet fully capture the complexity of real agentic workflows.

Despite these limitations, the results support our hypothesis: a single hidden state vector can serve as a compact communication medium between trained agents. This suggests a promising direction for reducing token-intensive natural language exchange in multi-agent systems. Future work should test whether this approach remains effective across longer chains, more complex reasoning tasks and larger datasets.

References

- Screwy Squirrel, Cat, T., & tom-and-jerry-lab. (2026, April). *Communication overhead in multi-agent LLM systems grows quadratically with agent count* [Preprint]. clawRxiv. https://clawrxiv.io/abs/2604.00736
- Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2024). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics, 12*, 157–173. https://doi.org/10.1162/tacl_a_00638
- Du, Z., Wang, R., Bai, H., Cao, Z., Zhu, X., Cheng, Y., Zheng, B., Chen, W., & Ying, H. (2026). *Enabling agents to communicate entirely in latent space*. arXiv. https://doi.org/10.48550/arXiv.2511.09149
- Wu, S., Wang, Y., & Yao, Q. (2025). *Dense communication between language models*. arXiv. https://doi.org/10.48550/arXiv.2505.12741
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention is all you need*. arXiv. https://doi.org/10.48550/arXiv.1706.03762
- Rush, A. M. (2018). The annotated transformer. Harvard NLP. https://nlp.seas.harvard.edu/annotated-transformer/
- Xue, L., Barua, A., Constant, N., Al-Rfou, R., Narang, S., Kale, M., Roberts, A., & Raffel, C. (2022). *ByT5: Towards a token-free future with pre-trained byte-to-byte models*. arXiv. https://doi.org/10.48550/arXiv.2105.13626
- Clark, J. H., Garrette, D., Turc, I., & Wieting, J. (2022). CANINE: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics, 10*, 73–91.
- Bai, L., Huang, Z., Wang, X., Sun, J., Mihalcea, R., Brynjolfsson, E., Pentland, A., & Pei, J. (2026). *How do AI agents spend your money? Analyzing and predicting token consumption in agentic coding tasks*. arXiv.
- Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., Zhao, W. X., Wei, Z., & Wen, J.-R. (2023). *A survey on large language model based autonomous agents*. arXiv. https://doi.org/10.48550/arXiv.2308.11432
- Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N. S., Chen, A., Creel, K. A., Davis, J. Q., Demszky, D., ... Liang, P. (2021). On

the opportunities and risks of foundation models. arXiv. https://doi.org/10.48550/arXiv.2108.07258

- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems, 33, 1877–1901.

- Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research, 3*, 1137–1155.

- Kosinski, M. (2024). Evaluating large language models in theory of mind tasks. *Proceedings of the National Academy of Sciences, 121*(45), Article e2405460121. https://doi.org/10.1073/pnas.2405460121

- Ma, X., Gao, L., & Xu, Q. (2023). ToMChallenges: A principle-guided dataset and diverse evaluation tasks for exploring theory of mind. arXiv. https://doi.org/10.48550/arXiv.2305.15068

- Wimmer, H., & Perner, J. (1983). Beliefs about beliefs: Representation and constraining function of wrong beliefs in young children's understanding of deception. *Cognition, 13*(1), 103–128. https://doi.org/10.1016/0010-0277(83)90004-5

- Anthropic. (2026). *Pricing*. Anthropic Docs. https://docs.anthropic.com/en/docs/about-claude/pricing

- OpenAI. (2026). *Pricing*. OpenAI API. https://platform.openai.com/docs/pricing

Acknowledgement of use of Artificial Intelligence

AI tools were used in the completion of this work in the following ways. Any content generated underwent meticulous manual review and subsequently received final approval from the author.

OpenAI (GPT-5.5)

- Brainstorming
- Researching background information
- Literature scans
- Proofreading

Opencode (GPT-5.2)

- Boilerplate code completion
- Unit test completion
- Training data generation
- Shape verification & prototyping

Google Slides Gemini

- Slide formatting
- Image generation