QMamba: Quantum Selective State Space Models for Text Generation

Gerhard Stenzel[a], Michael Kölle, Tobias Rohe, Maximilian Balthasar Mansky, Jonas Nüßlein and Thomas Gabor

LMU Munich, Munich, Germany

fi

Keywords: Quantum Machine Learning, Quantum Generative Models, State Space Models, Variational Quantum Circuits, Quantum Computing, Sequence Modeling.

Abstract: Quantum machine learning offers novel paradigms to address limitations in traditional natural language processing models, such as fixed context lengths and computational inefficiencies. In this work, we propose QMamba, the first quantum adaptation of the Mamba architecture, integrating selective state space models with quantum computation for efficient and scalable text generation. QMamba leverages quantum principles like superposition and entanglement to enable unbounded context sizes and reduced computational complexity. Our contributions include the development of a quantum generative model optimized for hardware constraints, advancements in encoding, embedding, and measurement techniques, and the demonstration of its performance on pattern reproduction and context-challenging tasks like ”Needle in a Haystack.” Experimental results confirm QMamba’s potential to maintain high efficiency and performance across varying sequence lengths, laying the groundwork for future explorations in quantum-enhanced natural language processing.

1 INTRODUCTION

The rapid advancements in large language models (LLMs) have revolutionized natural language processing (NLP). However, their current reliance on transformer architectures comes with inherent limitations, such as fixed context lengths, high computational complexity, and inefficiencies in capturing long-range dependencies. These limitations create a bottleneck, particularly for tasks requiring extensive sequence analysis and efficient processing. State space models (SSMs) have emerged as an alternative architecture, offering linear computational complexity and better scalability for long sequences. However, their inability to focus on contextually relevant input significantly limits their performance on advanced tasks. To address these shortcomings, architectures like Mamba have introduced selective mechanisms, improving context handling while maintaining efficiency.

Quantum machine learning (QML) introduces a novel paradigm capable of addressing these computational bottlenecks. By leveraging quantum principles like superposition and entanglement, QML offers the potential to reduce computational overhead and enhance model performance. Yet, despite this potential, QML in NLP remains underexplored, and existing quantum generative models often suffer from limited context sizes and scalability issues.

This work proposes QMamba, the first quantum adaptation of the Mamba architecture, to bridge these gaps. By integrating the strengths of state space models with quantum computation, QMamba aims to overcome the context and efficiency limitations of existing architectures, enabling scalable and effective text generation. This innovation not only demonstrates the viability of quantum approaches in NLP but also establishes a foundation for future research in quantum generative models.

In section 2, we review related work, covering transformers, state space models (SSMs), the Mamba architecture, and quantum machine learning. Section 3 introduces our novel Quantum Mamba (QMamba) architecture, detailing its encoding and embedding (Section 3.1), measuring (Section 3.2), and circuit architecture (Section 3.3). Section 4 outlines our experimental setup and evaluation tasks. We include a simple pattern reproduction test for compression capabilities and the ”Needle in a Haystack” challenge to assess handling of unbounded context sizes. Finally, Sections 5 and 6 discuss results and propose directions for future research.

This paper makes the following contributions:

[a] https://orcid.org/0009-0009-0280-4911

Stenzel, G., Kölle, M., Rohe, T., Mansky, M. B., Nüßlein, J. and Gabor, T.
QMamba: Quantum Selective State Space Models for Text Generation.
DOI: 10.5220/0013378300003890
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 1*, pages 742-750
ISBN: 978-989-758-737-5; ISSN: 2184-433X
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

- We present the theoretical foundations of quantum state space models, providing a comprehensive framework for understanding their principles and applications.
- We propose efficient methods for token encoding and embedding, leveraging quantum properties to enhance computational efficiency.
- We introduce novel techniques for token decoding and measuring for NLP tasks on hardware-constrained systems, optimizing the performance of quantum generative models.
- We develop a new quantum generative model, QMamba, which integrates quantum computing with state space models, paving the way for more efficient and scalable quantum text generation models.

2 RELATED WORK

2.1 Transformers

The previously rather stale field of models generating human-like language has seen a massive boost in performance with the introduction of the attention mechanism (Bahdanau et al., 2016). This allowed models to focus on the most relevant parts of the input sequence, hence allowing for a much better context understanding. Building on this, transformers using self-attention mechanisms (Vaswani et al., 2017) like BERT (Devlin et al., 2019) and ChatGPT (OpenAI, 2024) have become the state-of-the-art in many NLP tasks (Brown et al., 2020; Touvron et al., 2023) and even some multi-modal tasks (Llama Team, 2024; OpenAI, 2024; Gemini Team, 2024; Qwen Team, 2024). As with most models, the solution quality of transformers roughly scales with their size (Kaplan et al., 2020). The main bottleneck of transformers is their context length, which limits the maximum amount of information that the model can use to generate the next token (Dao et al., 2022). Even though there are numerous ways to improve the efficiency of transformers using better attention mechanisms (Dao et al., 2022; Dao, 2023; Llama Team, 2024; Gemini Team, 2024; Yang et al., 2024; Tay et al., 2020), the search for other model architectures is still ongoing.

2.2 State Space Models

State space models (SSMs) are an end-to-end neural network architecture, with foundations in statistics, but also usable for sequence generation tasks. It consists of two parts: the current input x_t (with t being the current time step) and the current state h_t, and three parameters A, B, C that define the model. State Space Models can be interpreted as continuous functions, with the state being computed as $h_t\prime = Ah_t + Bx_t$ and the output as $x_{t+1} = Ch_t$ (we use a slightly differing notation compared to (Gu et al., 2021; Gu et al., 2022) to keep consistency with the rest of this paper). The continuous formulation is however not usable for training or inference of the model, as it is slow to compute (due to differential equations) and not parallelizable (Gu et al., 2021; Gu et al., 2022).

They can however be discretized using discretizing functions like zero-order hold, expanding the model to four main parameters A, B, C, Δ (equation (1)). This allows for significantly faster inference, as the model can be executed in a recurrent manner, with the state being updated at each time step, while maintaining a $\sim O(L)$ computational complexity (with L being the length of the input sequence). This makes SSMs a very efficient model for long sequences, compared to the $\sim O(L^2)$ complexity of transformers. SSMs can be trained very effectively by precomputing multiple steps of the model and then using a convolution operation to compute the output $x_{t+1} = CA^tBx_0 + CA^{t-1}Bx_1 + \cdots + CABx_{t-1} + CBx_t$, as the parameters A, B, C, Δ are not time-dependent (described as Linear Time Invariance in (Gu et al., 2021)). The convolutional interpretation allows for parallelizable initial training. SSMs however lack performance on advanced tasks, as they are not able to focus on the most relevant parts of the input sequence, like transformers (and other attention-based models) can. This has led to the development of several derivative models, like Mamba (Gu and Dao, 2024). (Gu et al., 2021; Gu et al., 2022; Gu and Dao, 2024)

$$\begin{aligned} \overline{A} &= \exp(\Delta A) \\ \overline{B} &= (\Delta A)^{-1}(\overline{A} - I) \cdot \Delta B \\ h_{t+1} &= \overline{A}h_t + \overline{B}x_t \\ x_{t+1} &= Ch_{t+1} \end{aligned} \tag{1}$$

2.3 Mamba

Mamba (Gu and Dao, 2024) builds upon the SSM architecture, expanding it to a selective state space model (SSSM). Instead of using the same parameters for all input tokens, selective state space models allow the models to influence the B, C and Δ parameters of the model (see figure 1, A does not have to be selective, as it is directly influenced by Δ). This is achieved by creating a projection of the current input token into a high-dimensional space, and then supplying this additional information to the model. Δ, which controls the amount of influence of the previous state on the current state, is now additionally influenced by the input token. This allows the model to focus on the most relevant parts of the input sequence and discard less relevant parts, making it significantly more efficient than the original SSM. As this violates the Linear Time Invariance property of SSMs, the model cannot be parallelized using the convolution method described above. Instead, the paper proposes other hardware efficient methods to train the model, like optimizing which parts of the model should be stored in the GPUs (slow but big) high-bandwidth memory or in the (fast but small) low-bandwidth memory. In combination with the parallel associative scan technique (Harris et al., 2007), this allows for training and inference efficiencies comparable or even better than transformers. (Gu and Dao, 2024)

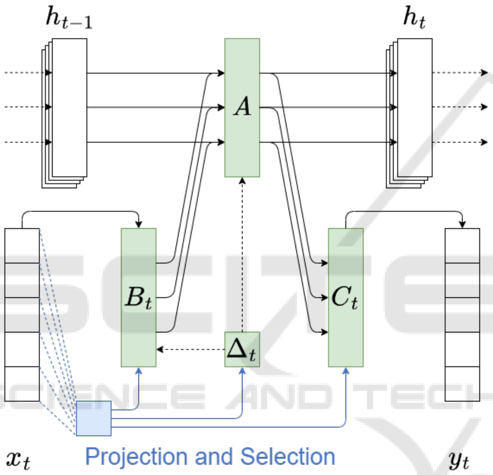


Figure 1: Structured State Space Models have four groups of parameters, marked in green. The selection mechanism is marked in blue. Activations are not shown for clarity. Figure modified from Mamba (Gu and Dao, 2024).

MambaByte builds upon Mamba, but skips the tokenization of the inputs and outputs and instead operates on the byte level. Such an approach is unfeasible for transformer-based models, wasting its precious context size, but is feasible on the Mamba architecture, as the model can use the selection mechanism to discard irrelevant parts efficiently (Wang et al., 2024). RWKV is another novel approach, combining concepts of transformers (using a self-attention mechanism) and Mamba (using a selective state space model), allowing for unbounded context size, in a very efficient manner (Peng et al., 2023; Peng et al., 2024).

2.4 Quantum Machine Learning

Quantum machine learning (QML) harnesses quantum computing to address the increasing computational demands of traditional machine learning algorithms (Biamonte et al., 2017; Gabor et al., 2020; Rohe et al., 2024). Central to QML, variational quantum circuits (VQCs) serve as quantum analogs of classical neural networks, functioning as effective function approximators. VQCs apply parameterized unitary gates to qubits (Barenco et al., 1995), leveraging quantum phenomena such as superposition, entanglement, and interference. These parameters, typically rotation angles, are optimized using standard machine learning techniques. A VQC architecture consists of three key components: input state encoding, a trainable quantum circuit, and measurement output decoding. Notably, VQCs can efficiently process high-dimensional inputs with only $\log_2(N)$ qubits (Lloyd et al., 2013). However, they currently face challenges like substantial qubit overhead and high error rates on real quantum devices in the Noisy Intermediate-Scale Quantum (NISQ) era (Preskill, 2018). Nonetheless, ongoing advancements are expected to strengthen QML's essential role in the future (Preskill, 2018; Gabor et al., 2020; Kölle et al., 2024).

2.5 Quantum Generative Models

For the topic of quantum machine learning based text generation models, the focus has been quantum transformers.

The quantum transformer proposed by (Di Sipio et al., 2022) uses a quantum transformer for sentiment analysis by replacing each of the well-known attention mechanism's parameters with quantum gates, similar to the paper using a quantum transformer for gluon classification (Comajoan Cara et al., 2024). Quixer (Khatri et al., 2024) is a quantum transformer capable of text generation, albeit with a very limiting context size of only 32 tokens on the Penn Tree Bank dataset (Marcus et al., 1993).

To the best of our knowledge, this paper is the first to propose porting the Mamba architecture to quantum machine learning.

3 QUANTUM MAMBA

3.1 Encoding and Embedding

All input tokens are encoded into a unique combination of rotations across q_{in} qubits, utilizing r possible rotation angles. These angles are evenly distributed from 0 (inclusive) to 2π (exclusive), forming the sequence Θ as defined in equation (2). Subsequently, the tokens are embedded using rotational gates such as R_x, R_y, or R_z. This method allows for the encoding of up to $k^{q_{in}}$ different tokens within a circuit comprising q_{in} states. When $k = 2$ and the R_x gate is used for embedding, this corresponds to basis embedding. For higher values of k, this approach facilitates significant input compression. Obvious limitations include the increasing difficulty for the model to distinguish between tokens as the number of possible rotations r grows, therefore a trade-off between the number of qubits q_{in} (and thus execution speed) and the number of distinguishable tokens k must be made. Additionally, when executing on actual quantum hardware in the NISQ era, a high number of angles r is negatively impacted by precision error both during encoding and measurement.

$$\Theta = \left(\frac{2\pi n}{k} \;\middle|\; n \in \{0, 1, 2, \ldots, k-1\} \right) \tag{2}$$

The work by (Comajoan Cara et al., 2024) employs patch angle embedding (thus hindering parallelism), whereas (Khatri et al., 2024) utilizes classical tokenization and classical weights to create input embeddings (which limits the potential benefits of quantum models by requiring classical pre- and post-processing).

3.2 Measuring

The output is processed in a manner analogous to the inputs: the probability p of each qubit being in the state $|1\rangle$ is measured in a predefined basis. Subsequently, the sequence δ (not to be confused with Δ, which is a sub-circuit of Mamba) is computed as defined in equation (3), which enumerates the discrepancies between the predicted and target values. The index corresponding to the smallest error is identified as the predicted output.

The overall confidence of an output token is quantified by summing the discrepancies across all q_{out} qubits. This measurement allows for the introduction of a temperature parameter τ, which facilitates controlled randomness in the output by scaling the confidence scores of the output tokens. Specifically, when $\tau = 0$, the model deterministically selects the most confident token.

$$\delta = \left(\frac{n}{k-1} - p \;\middle|\; n \in \{0, 1, 2, \ldots, k-1\} \right) \tag{3}$$

3.3 Circuit Architecture

The naive quantum implementation of the Mamba architecture, depicted in figure 2, utilizes $e+k+m+s$ qubits. Here, e represents the number of qubits required for embedding the information, m (approximately equal to e) for intermediate processing, k for extracting the most relevant information from the input (also approximately equal to e), and s for storing the entire state of the model. Consequently, this configuration results in a large circuit with $3e+s$ qubits. We define h_0 to be $|0\rangle$. The Δ circuit is executed first. It is parameterized with x_t using re-uploading. This is comparable to the Δ matrices in the non-quantum Mamba (Gu and Dao, 2024), which predicts the impact of the current token x_t on the state. This impact is then used as a weight between the A and B_{x_t} sub-circuits. By using this trade-off, the model can learn to distinguish between relevant and less relevant features. The non-quantum Mamba draws inspiration from the discretization of a continuous signal with an adaptive parameter choosing the step size. For the inherently continuous quantum model, this is achieved by interpreting the A and B gates as controlled by the result of the Δ circuit. Notably, their control basis is complementary, with one activating when the other is not. This, over the course of multiple discrete shots or a full quantum state simulation, allows the model to learn the trade-off between the A and B sub-circuits. The A circuit stretches over multiple qubit groups, spanning the qubits holding the last state $\langle h_t|$, the result of the input group B and the intermediate group m. Just like in the classical Mamba, the A circuit does not contain re-uploading of the input token, as the input tokens are already encoded into B. The C sub-circuit (parameterized with re-uploading just as Δ and B) transforms the output of A and is followed by a measurement. Out of the $e+k+m+s$ qubits, only m get measured, $k+e$ get reset and s holds the state for the next mamba block.

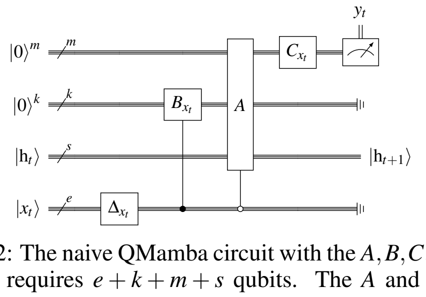


Figure 2: The naive QMamba circuit with the A, B, C, Δ sub-circuits requires $e+k+m+s$ qubits. The A and B sub-circuits are controlled by the result of the Δ sub-circuit.

The circuit efficiency can be improved by reusing the qubits of the C sub-circuit, thus reducing the number of required qubits to $2k+s$ (depicted in figure 3).

Even further improvements can be achieved by executing the Δ operation on a separate circuit, and input its result using an R_x gate. This optimization reduces the qubit requirement to $k+s+1$. A and B are controlled and anti-controlled as before.

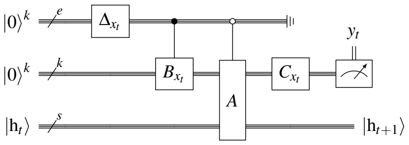


Figure 3: A more efficient QMamba circuit, using only $2k+s$ qubits.

We have implemented our model using the Qandle (Stenzel et al., 2024) framework. The trainable parameters are packed in strongly entangling layers of the sub-circuits as $R_yR_zR_y$ combinations followed by a CNOT gate. The individual model configurations are listed in the appendix (section 6).

4 EXPERIMENTS

We want to evaluate out QMamba model on two main tasks: text generation based on easy to learn patterns and the “Needle in a Haystack” task. For pattern reproduction, the intent is to learn a simple pattern, testing the model’s capability to compress several tokens into as few qubits as possible, using a relatively small context size. The “Needle in a Haystack” task is an advanced challenge to test the “lost in the middle” problem (Liu et al., 2023). It has been shown that Large Language Models boasting with very high potential context sizes can’t utilize all of it, causing the models to ignore parts of context, usually snippets located in the middle of the context (Hsieh et al., 2024). The Mamba architecture has a theoretically unbounded context size due to its selection mechanism, allowing it to keep relevant parts in the context indefinitely (Gu and Dao, 2024). We want to test if the quantum version of Mamba can keep up with this promise.

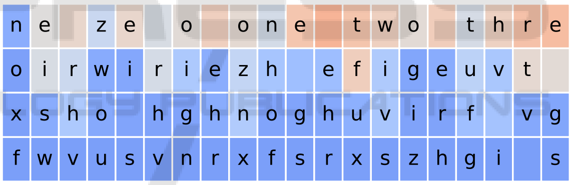
The loss is computed as the mean squared error between the predicted and target values, attempting to minimize the discretization error δ (as defined in equation (3)) for each token. The different tasks have been evaluated on separately trained models.

4.1 Datasets

For the pattern reproduction task, we create a simple synthetic dataset, containing the words ”zero”, ”one”, ..., ”nine”. The words are always presented in the same looping order. As the model operates on the character level, the task can be trivially described to predict the next logical character in the sequence. During training, the model is represented with a random starting character, and the task is to predict the next character in the sequence. Independent of the produced character, the model is always presented with the next character in the sequence parallel. As the dataset is trivial to test, we can predict the error rate of the model per token, e.g., the model should always predict “e” after “z” (from “zero”) and a space after “x” (“six”).

The ”Needle in a Haystack” uses a different setup. We pick n different tokens. Each sequence consist of a fixed number of tokens (the training context size), with one of the tokens being replaced with a globally fixed pointer token. The task for the model is to output the token directly following the pointer token. During testing, sequences with a higher length are generated and primed with an indicator token. All intermediate outputs are discarded, as the only relevant output follows after inputting the entire test string. The model is evaluated on the accuracy of the output token, depending on the test context size and the pointer position, ignoring all other outputs.

4.2 Results



n	e		z	e		o		o	n	e		t	w	o		t	h	r	e
o	i	r	w	i	r	i	e	z	h		e	f	i	g	e	u	v	t	
x	s	h	o		h	g	h	n	o	g	h	u	v	i	r	f		v	g
f	w	v	u	s	v	n	r	x	f	s	r	x	s	z	h	g	i		s

Figure 4: The QMamba model produces high probabilities for the correct tokens in the pattern reproduction task. The most likely token is listed in the first row, less likely tokens listed below. The color indicates the probability of the token. For a $\tau > 0$, non-maximum tokens can be chosen non-deterministically, weighted by their probability.

On the pattern reproduction task, our QMamba models show strong performance. Analyzing the predictions of the model in figure 4, we see the expected behavior: as the beginning of the training strings has been chosen at random, the model is initially unsure, which token to produce, outputting a mixed encoding, containing multiple possible tokens with near equal confidence (we measure the error from the continuous model outputs to the individual discrete token encodings as the confidence in each token). As we have selected $\tau = 0$, we deterministically choose the most likely token, and feed it back into the model for the next iteration. With rising number of iteration and therefore growing context size, the model becomes more confident in outputting the correct tokens.

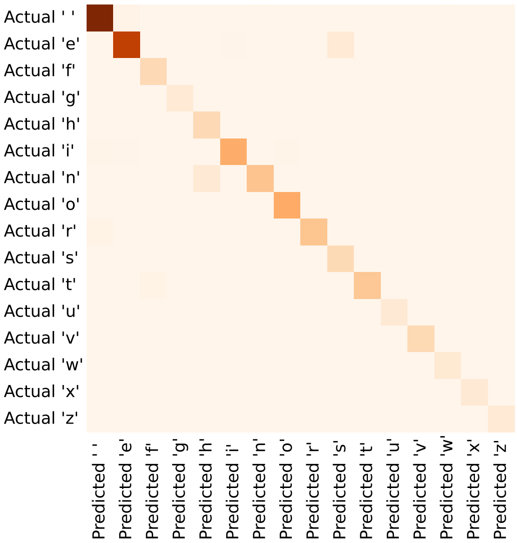


Figure 5: QMamba's confusion matrix for the pattern reproduction task shows very low error rates. Dark colors indicate a higher probability.

The strong quality is underlined by the confusion matrix in figure 5, showing a very strong diagonal distribution for single token continuation. When tasked with predicting only the next token, the correct token is chosen with a very high probability. The few outliers fluctuate from run to run, while maintaining a very acceptable amount.

On the "Needle in a Haystack" task (Hsieh et al., 2024), our QMamba performs very well (evaluation in figure 6). On low document lengths like 10 or 20 tokens (10 was the training context size), the models manage to retrieve close to all target tokens. As the document length increases, the accuracy decreases, but reaches a stable level at around 65% accuracy for a document length of 100 (so ten times the training context length) tokens. The retrieval quality for needles in the last 20% of the document does not deteriorate, thus showing that the model does not suffer from a polluted previous context. The stable (albeit lowered) accuracy for long contexts for needles both at the beginning and the end of the document shows that the model does not suffer from the "lost in the middle" problem, as the difficulty is mostly due to the increased noise from tokens after the needle then by a structural learning problem seen in many classical transformer based models (Liu et al., 2023).

5 DISCUSSION

We have introduced the theoretical foundations of quantum state space models, allowing our quantum model to repeatedly decide the amount of update allowed to the state based on the current token, allowing for theoretically unbounded context sizes. We have experimentally shown the effectiveness of our token encoding, embedding, and measurement strategies and have laid out the path for non-deterministic sampling. The model has shown impressively low error rates, with a very strong diagonal in the confusion matrix. From the very promising results on the ”Needle in the Haystack” task (which the classical Mamba models and the RWKV family have already shown impressive results on despite huge context lengths (Peng et al., 2024)), we conclude that the strengths of selective state space models are not lost in the quantum translation. The models allow efficient training on smaller context sizes, losing only little in performance when running on large context sizes, like 10 times their original size (as seen in figure 6). In summary, our QMamba model manages to reproduce the classical Mamba's impressive performance on current quantum simulators, showing the unused potential of quantum machine learning in natural language processing.

6 FUTURE WORK

Our embedding and measurement techniques for embedding natural language in quantum circuits have shown impressive results. We have, however, noticed a degradation in the results for more than four discrete embedding angles, limiting the model's abilities for more than five angles to a minimum. In a four-qubit setup, this would still allow embedding a very impressive $4^5 = 1024$ different tokens, but a deeper analysis of the drop-off is needed.

As all models operated on synthetic datasets, bigger models trained on real-world datasets like the Penn Tree Bank (Marcus et al., 1993) or the WikiText-103 (Merity et al., 2016) are needed to evaluate the models on real-world tasks. The models should be evaluated on perplexity scores (Jurafsky and Martin, 2008) to compare them to classical models to fully bridge the gap between quantum and classical text generation models. These bigger models should be evaluated with the same time budgets as comparable classical models to allow for a fair comparison in the NISQ (Preskill, 2018) era.

In preparation for the post-NISQ era, the models should be tested for their suitability on running on large-scale, fault-tolerant quantum computers. As QMamba only builds upon basic quantum gates like R_x, R_y, and controlled-not and controlled groups, our models can be easily ported, allowing for even improved performance on real quantum hardware. Im-

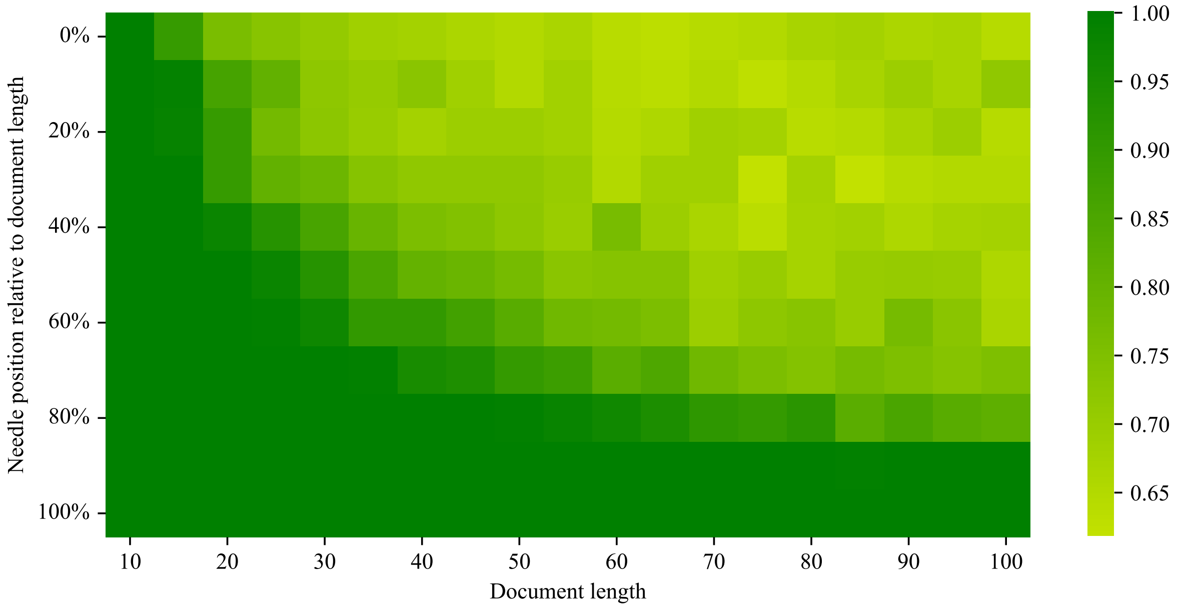


Figure 6: The QMamba model performs well on the "Needle in a Haystack" task. Dark green indicates a perfect score, while dark red (not present in the figure) would indicate a total failure with no correct outputs. The score is averaged from 3 models, with 1000 test sequences each.

plementations using customized encodings in combinations with non-standard circuit layouts could lead to further improvement.

ACKNOWLEDGEMENTS

This paper was partially funded by the German Federal Ministry of Education and Research through the funding program "quantum technologies — from basic research to market" (contract number: 13N16196).

REFERENCES

Bahdanau, D., Cho, K., and Bengio, Y. (2016). Neural Machine Translation by Jointly Learning to Align and Translate.

Barenco, A., Bennett, C. H., Cleve, R., DiVincenzo, D. P., Margolus, N., Shor, P., Sleator, T., Smolin, J. A., and Weinfurter, H. (1995). Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467.

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., and Lloyd, S. (2017). Quantum machine learning. *Nature*, 549(7671):195–202.

Brown, T., 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, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language Models are Few-Shot Learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Comajoan Cara, M., Dahale, G. R., Dong, Z., Forestano, R. T., Gleyzer, S., Justice, D., Kong, K., Magorsch, T., Matchev, K. T., Matcheva, K., and Unlu, E. B. (2024). Quantum Vision Transformers for Quark–Gluon Classification. *Axioms*, 13(5):323.

Dao, T. (2023). FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. (2022). FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Di Sipio, R., Huang, J.-H., Chen, S. Y.-C., Mangini, S., and Worring, M. (2022). The Dawn of Quantum Natural Language Processing. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8612–8616. IEEE.

Gabor, T., Sünkel, L., Ritz, F., Phan, T., Belzner, L., Roch, C., Feld, S., and Linnhoff-Popien, C. (2020). The Holy Grail of Quantum Artificial Intelligence: Major Challenges in Accelerating the Machine Learning Pipeline.

Gemini Team (2024). Gemini: A Family of Highly Capable Multimodal Models.

Gu, A. and Dao, T. (2024). Mamba: Linear-Time Sequence Modeling with Selective State Spaces.

Gu, A., Goel, K., and Ré, C. (2022). Efficiently Modeling Long Sequences with Structured State Spaces.

Gu, A., Johnson, I., Goel, K., Saab, K., Dao, T., Rudra, A., and Ré, C. (2021). Combining Recurrent, Convolutional, and Continuous-time Models with Linear State-Space Layers.

Harris, M., Sengupta, S., and Owens, J. (2007). Chapter 39. Parallel Prefix Sum (Scan) with CUDA — NVIDIA Developer.

Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., Zhang, Y., and Ginsburg, B. (2024). RULER: What's the Real Context Size of Your Long-Context Language Models?

Jurafsky, D. and Martin, J. (2008). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, volume 2.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling Laws for Neural Language Models.

Khatri, N., Matos, G., Coopmans, L., and Clark, S. (2024). Quixer: A Quantum Transformer Model.

Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization.

Kölle, M., Giovagnoli, A., Stein, J., Mansky, M. B., Hager, J., and Linnhoff-Popien, C. (2022). Improving Convergence for Quantum Variational Classifiers using Weight Re-Mapping.

Kölle, M., Stenzel, G., Stein, J., Zielinski, S., Ommer, B., and Linnhoff-Popien, C. (2024). Quantum Denoising Diffusion Models.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. (2023). Lost in the Middle: How Language Models Use Long Contexts.

Llama Team (2024). Llama 3.2: Revolutionizing edge AI and vision with open, customizable models.

Lloyd, S., Mohseni, M., and Rebentrost, P. (2013). Quantum algorithms for supervised and unsupervised machine learning.

Marcus, M., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. (2016). Pointer Sentinel Mixture Models.

OpenAI (2024). GPT-4.

Peng, B., Alcaide, E., Anthony, Q., Albalak, A., Arcadinho, S., Biderman, S., Cao, H., Cheng, X., Chung, M., Grella, M., GV, K. K., He, X., Hou, H., Lin, J., Kazienko, P., Kocon, J., Kong, J., Koptyra, B., Lau, H., Mantri, K. S. I., Mom, F., Saito, A., Song, G., Tang, X., Wang, B., Wind, J. S., Wozniak, S., Zhang, R., Zhang, Z., Zhao, Q., Zhou, P., Zhou, Q., Zhu, J., and Zhu, R.-J. (2023). RWKV: Reinventing RNNs for the Transformer Era.

Peng, B., Goldstein, D., Anthony, Q., Albalak, A., Alcaide, E., Biderman, S., Cheah, E., Du, X., Ferdinan, T., Hou, H., Kazienko, P., GV, K. K., Kocoń, J., Koptyra, B., Krishna, S., au2, R. M. J., Lin, J., Muennighoff, N., Obeid, F., Saito, A., Song, G., Tu, H., Wirawan, C., Woźniak, S., Zhang, R., Zhao, B., Zhao, Q., Zhou, P., Zhu, J., and Zhu, R.-J. (2024). Eagle and Finch: RWKV with Matrix-Valued States and Dynamic Recurrence.

Preskill, J. (2018). Quantum Computing in the NISQ era and beyond.

Qwen Team (2024). Qwen2.5: A Party of Foundation Models.

Rohe, T., Grätz, S., Kölle, M., Zielinski, S., Stein, J., and Linnhoff-Popien, C. (2024). From Problem to Solution: A general Pipeline to Solve Optimisation Problems on Quantum Hardware.

Stenzel, G., Zielinski, S., Kölle, M., Altmann, P., Nüßlein, J., and Gabor, T. (2024). Qandle: Accelerating State Vector Simulation Using Gate-Matrix Caching and Circuit Splitting.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. (2020). Long Range Arena: A Benchmark for Efficient Transformers.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023). LLaMA: Open and Efficient Foundation Language Models.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need.(Nips), 2017. *Advances in Neural Information Processing Systems*.

Wang, J., Gangavarapu, T., Yan, J. N., and Rush, A. M. (2024). MambaByte: Token-free Selective State Space Model.

Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., and others (2024). Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

APPENDIX

Hyperparameters

Hyperparameters have been determined by using a constrained grid search for learning rate, optimizer, and number of parameters in a constrained time frame per run. All models used quantum weight remapping for limiting parameter ranges (Kölle et al., 2022) and the Adam optimizer (Kingma and Ba, 2014). We used a parameter distribution of 2:1:1:1 ratio, spread out over the A, B, C, and Δ sub-circuits (picked after preliminary trials). For the haystack task, we chose a learning rate of $2e^3$ and around 1000 parameters in total.

Encoding and Decoding

The mapping between angle representation and input/output tokens is distinct for each token set. We enumerate both the token set and the possible angle combinations and naively match them up. More advanced approaches could include a statistical analysis of frequent patterns in the training data, matching them up with more similar encodings. This would essentially imply mapping the encoding space to a *q*-dimensional space, with the Hamming distance between two angle embeddings correlating with the frequency of consecutive tokens. This would enable the model to generate specific patterns more easily, improving short-term performance. We did not pursue this idea further, as we worried about a possible lack of output diversity. However, we are confident that investigating this idea further could lead to a significant improvement in the model's performance.