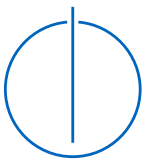
SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Exploring Efficient Approaches for Long-Context NLP

Hugo Henrique Silva Pitorro



SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Exploring Efficient Approaches for Long-Context NLP

Erforschung effizienter Ansätze für NLP mit langem Kontext

Author:	Hugo Henrique Silva Pitorro
Supervisor:	Prof. Dr. Georg Groh
Advisors:	Prof. Dr. André Martins
	Dr. Marcos Treviso
Submission Date:	15th May 2024

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, 15th May 2024 Hugo Henrique Silva Pitorro

Acknowledgments

I really would like to thank my advisors, Marcos and Professor André, for their great help and advice throughout this thesis. In the same spirit, I'd like to thank Professor Georg Groh for allowing me to take advantage of this great opportunity.

A special thanks go to my family and friends who gave their full support during these two years in Munich. Shoutout to Gonçalo, Inês, Diogo, Manuel, Coutinho, and Manuel Coutinho. And finally, to my wonderful Raquel, a sincere thank you for everything thus far and for what's yet to come!

Abstract

Transformers have become the standard architecture in several Natural Language Processing tasks due to their flexibility and strong performance. This is despite their inherent inefficiencies when handling long contexts. In this thesis, we compare Transformers with recently proposed linear recurrent models on the task of Machine Translation. Specifically, we measure the impact of sequence length on translation quality with lexical (BLEU) and neural-based metrics (COMET), using sentence-level and paragraph-level translation datasets. We find that Mamba, a context-dependent linear recurrent model, is a competitive architecture to the Transformer both on sentence-level data and when scaling to paragraph-level translation data. However, we show that its performance in long sequence length samples is directly tied to the training data distribution as it suffers from poor sequence length extrapolation abilities. On the other hand, we find that equipping linear recurrent models with attention leads to a strong combination as the produced hybrid models were able to match and even outperform Transformers in terms of translation quality and robustness to distribution shifts. Overall, this thesis presents promising results and enhances our understanding of the behavior of linear recurrent models, thereby encouraging further research into their potential for natural language applications.

Contents

1. Introduction

Natural Language Processing (NLP) and Machine Learning (ML) have become integral to modern information systems, profoundly influencing how we interact with technology. The adoption of ML has transformed service quality across various domains, significantly enhancing consumer and businesses' quality of life (Maslej et al. 2023). One of the most notable advancements is observed in Machine Translation (MT), where ML has replaced rule-based systems and, consequently, dramatically improved translation quality (Johnson et al. 2017).

Central to these innovations is the Transformer architecture (Vaswani et al. 2017), renowned for its attention mechanism. This mechanism allows different parts of the input to dynamically interact with each other, improving the model's understanding and processing capabilities. Originally designed for Machine Translation (MT), the flexibility of the Transformer has since been adapted to handle other data modalities, with some notable systems including ViT (Dosovitskiy et al. 2021) (images), Whisper (Radford et al. 2022) (audio) and Sora (Brooks et al. 2024) (video). In the field of MT, the impact of Transformers is particularly evident, forming the basis for most WMT General MT shared task submissions (Kocmi et al. 2023).

1.1. Efficiency Challenges and Alternatives

Despite its reported strengths, the Transformer is not without its downsides. Performance has been deeply connected to the scale of the model (Brown et al. 2020; Hoffmann et al. 2022), leading to a substantial increase in the average number of parameters in new model releases. This fact represents a significant downside for mass adoption, especially for resource-constrained environments like embedded systems or mobile computing settings. While there have been various attempts at minimizing this issue (Jiao et al. 2020; Bhandare et al. 2019), the top-performing models are out of reach for the average consumer and are commonly deployed behind a closed API (OpenAI et al. 2024; Deepmind et al. 2024).

Compounding the scalability problem is the Transformer's inefficient handling of long sequences due to the quadratic complexity of the self-attention mechanism. Although this feature allows the model to dynamically focus on relevant parts of the input, it results in high computational costs from the pairwise comparisons needed

for each token. This inefficiency is particularly problematic as many ML applications involve processing long sequences. In response, significant research has been devoted to developing more efficient architectures capable of handling such demands (Liu et al. 2023; Beltagy et al. 2020; Martins et al. 2022; Katharopoulos et al. 2020; Tay et al. 2022), extending across various NLP tasks and modalities (Treviso et al. 2023).

Given the efficiency concerns on Transformers, it is timely to assess the potential of emerging architectures like Mamba (Gu et al. 2023) and RetNet (Sun et al. 2023) as viable alternatives. Mamba, with its use of structured state-space model techniques and RetNet, which revises the traditional attention mechanism by removing its non-linearities, present promising avenues for efficient computational design. However, while these models embody a significant promise in enabling fast, low-latency inference with strong performance, the impact of such architectural innovations on MT is still an open problem, with early experiments suggesting that state-space models lead to accuracy degradation (Vardasbi et al. 2023).

1.2. Machine Translation and Thesis Objectives

Having motivated the project and the related *efficiency* questions, we now state the thesis objectives. Succinctly, this thesis aims to present a thorough evaluation of the performance of efficient models on MT. New model assessments commonly concentrate solely on Language Modeling (LM) performance, often neglecting the distinct challenges presented by downstream text-to-text tasks such as MT or summarization. Specifically, prevalent LM benchmarks, such as MMLU (Hendrycks et al. 2021) or GPQA (Rein et al. 2023), are geared towards assessing general knowledge and reasoning capabilities, which may not directly translate to MT performance. In contrast, MT benefits from direct, objective quality metrics (Rei et al. 2022a; Rei et al. 2022b) that allow for a clear assessment of model capabilities. While there is a notable correlation between LM and MT performance, crucial differences, like whether generation is bounded to a source sentence (as is the case for MT) or open-ended (LM), make the quality relationship not so straightforward. Additionally, most current Large Language Models (LLM) are predominantly trained on English language data, reflecting the language distribution of their training corpora, which can negatively impact MT performance, particularly in languages that are underrepresented in the training data (Singh et al. 2024; Longpre et al. 2023).

In response to these considerations, this thesis aims to train Space State Models (SSM) and linear recurrent models for MT and evaluate them using objective metrics. The goal is to provide a detailed analysis of what linear models can achieve compared to traditional transformer models. By focusing on objective evaluations and the unique

demands of MT, this project seeks to expand our understanding of model capabilities across additional language processing applications. Additionally, we will also explore the significant potential of efficient models in long-document formats. Long-document translation presents unique challenges that are not as prevalent in shorter texts, such as maintaining context and coherence over extended passages. Efficient models are ideally suited for this task, as their ability to handle large sequences of data without a quadratic increase in processing time or cost makes them particularly appealing for translating long documents.

To achieve the aforementioned goals, we detail the remaining structure of this thesis:

- **Chapter 2.** *Background*: provides the theoretical background for the carried out thesis project work while contextualizing it within contemporary MT and model evaluation research.
- **Chapter 3.** *Experiments*: details the experimental setup, particularly regarding datasets, models, and overall training and evaluation framework.
- **Chapter 4.** *Results*: presents this thesis' findings through sentence and paragraph level MT experiments while analyzing model quality in terms of length extrapolation and recall abilities.
- **Chapter 5.** *Conclusion*: gives a retrospection on the overall thesis, how its goals were fulfilled, and provides promising directions for future work.

1.3. Main Contributions

Lastly, as a brief takeaway, we succinctly enumerate our research outcomes. The accompanying project code is available in Github[1].

1. We show that Mamba, a linear recurrent model, is able to produce competitive results, particularly outperforming the Transformer baselines in the WMT14 *de-en* language pair (Section 4.1).
2. We found scaling context from sentence level MT to the paragraph level challenging for small-scale models (77M parameter), revealing extrapolation issues across all models but exacerbated for Mamba (Section 4.2).
3. We perform an extensive analysis on the impact of sequence length on translation quality and measure how these metrics differ when shifting the training distribution towards longer samples, finding that concatenating samples alleviates the aforementioned extrapolation issues (Section 4.3).

[1] `https://github.com/deep-spin/retnet-mt/`

4. We investigate how models perform recall in MT through the recollection of Named Entities (NE) in generated translations, where we find the frequency of NE in the training data to be correlated with the ability for linear models to reproduce them during translations (Section 4.4).

5. Overall, we find that integrating attention and SSMs constitutes a recipe for building strong models in terms of translation quality, robustness to sequence length extrapolation and recalling abilities.

2. Background

This chapter provides the necessary background for a complete understanding of the project work carried out during this thesis. Concretely, in sections 2.1 to 2.3, we detail relevant architectures (Transformers and State Space Models); within Section 2.4, we formalize the MT task, and in Section 2.5, we contextualize this project in the broader MT training and evaluation research space.

2.1. Transformers

Transformers, introduced by Vaswani et al. 2017, have become ubiquitous in the natural language processing domain. They have demonstrated exceptional performance in tasks such as language modeling, translation, text summarization, among others. One of the differentiating factors enabling their success lies in the self-attention mechanism, which allows the model to meticulously take into account the contribution of different words in an input sequence when generating an output.

2.1.1. Attention Mechanism

The attention mechanism in Transformers operates by mapping a query and a set of key-value pairs to an output. This output is a weighted sum of the values, where the weights are computed based on the query's compatibility with the corresponding key. The attention function, in matrix notation, can be mathematically represented as follows:

$$\begin{aligned} &\mathbf{Q} = \mathbf{X}\mathbf{W}_{\mathbf{Q}}, \qquad \mathbf{K} = \mathbf{X}\mathbf{W}_{\mathbf{K}}, \qquad \mathbf{V} = \mathbf{X}\mathbf{W}_{\mathbf{V}}, \\ &\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{V}' = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}. \end{aligned} \tag{2.1}$$

Here, $\mathbf{Q} \in \mathbb{R}^{m \times d}$ and $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times d}$ denote the queries, keys, and values, respectively, and d is the dimensionality of the keys. Additionally, $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are obtained through three separate linear projections of the input, $\mathbf{W}_{\mathbf{Q}}, \mathbf{W}_{\mathbf{K}}, \mathbf{W}_{\mathbf{V}} \in \mathbb{R}^{f \times d}$, where f denotes the feature representation of each token. Moreover, the input $\mathbf{X}$ is of the form $\mathbb{R}^{n \times f}$,

the operation $\mathbf{QK}^\top$ performs dot product attention, and the division by d_k serves as a scaling factor.

To further note, in (Vaswani et al. 2017), the authors include multiple attention heads, entitled Multi-Head Attention (MHA), with the premise of each head being able to model different interaction patterns between tokens. The resulting states are concatenated before being fed into an output projection to form the attention mechanism output. Using the notation from before, along with H denoting the total number of attention heads and l the current layer, we can describe this in mathematical form:

$$\text{MHA}_l = \text{concat}\left(\mathbf{V}'_{l,1}, \ldots, \mathbf{V}'_{l,H}\right) \mathbf{W_O}. \tag{2.2}$$

2.1.2. Transformer Architecture

At its core, the Transformer architecture consists of two main components: an encoder and a decoder, each comprising multiple layers that are essentially identical in structure but function differently. A schematic for the architecture is depicted in Figure 2.1.

Each layer in the encoder part of the Transformer includes two primary subcomponents: the MHA mechanism and a position-wise fully connected feed-forward network. The outputs of the MHA are processed independently by the feed-forward network, which enhances the resulting representation with non-linear transformations.

The decoder in the Transformer mirrors the encoder's architecture but introduces an additional Cross-Attention component between the MHA and the feed-forward network. Notably, we denote the decoder MHA as Self-Attention since it focuses only on the output sequence while preventing interactions between current and future tokens. Cross-attention is crucial as it directs attention to the appropriate places in the input sequence by interacting with the encoder's outputs, which is particularly vital for tasks like machine translation, where the relevance of input features can vary widely across different parts of the output.

2.2. Efficient Transformers

While Transformers have revolutionized many areas of ML with their outstanding performance, especially in NLP, they are not without limitations, particularly when it comes to handling long sequences. The intrinsic complexity of the traditional Transformer architecture poses significant challenges in terms of computational time and memory usage.

The core of the computational complexity in Transformers lies in the attention mechanism, as described in Equation 2.1, where we calculate the dot product between every pair of tokens, with associated $\mathcal{O}\left(n^2\right)$ time complexity, where n is the sequence

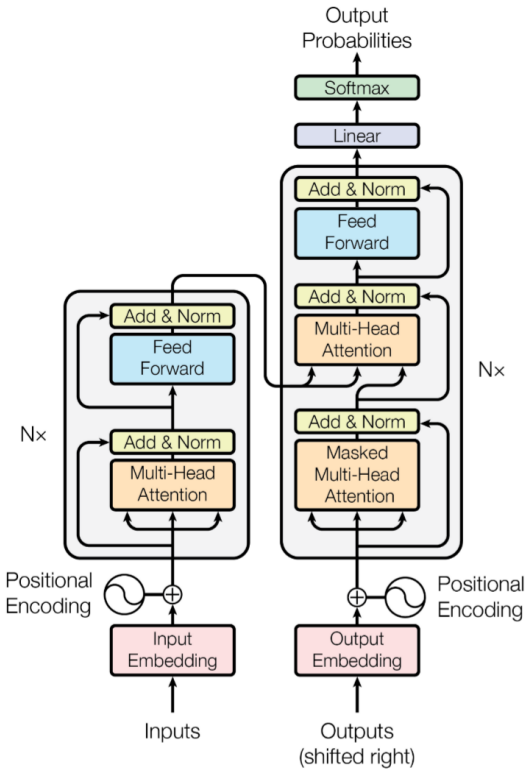


Figure 2.1.: Diagram of the Transformer architecture, taken from (Vaswani et al. 2017).

length. This results from the necessity to compute the interaction between each query (**Q**) and key (**K**), where $\mathbf{Q},\mathbf{K} \in \mathbb{R}^{n \times d}$, with d being the feature or embedding dimension. These interactions materialize in the $\mathbf{QK}^\top \in \mathbb{R}^{n \times n}$ matrix, leading to the aforementioned quadratic complexity.

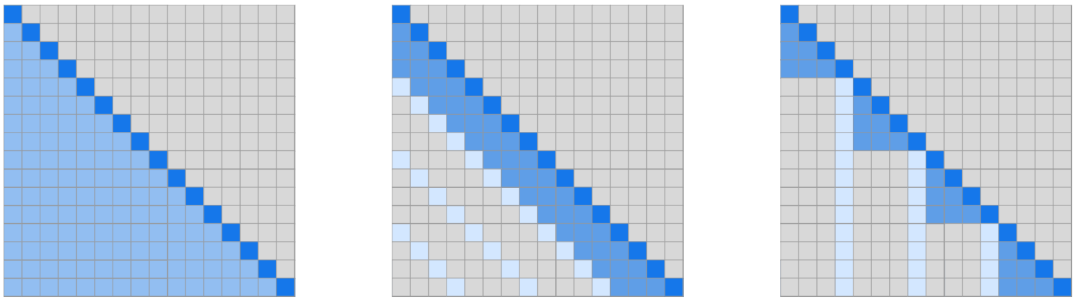
Nowadays, the Transformer has been subject to many optimizations, drastically reducing the memory footprint and increasing overall throughput. We refer specifically to Flash Attention (Dao et al. 2022; Dao 2023) and the more recent, Ring Attention (Liu et al. 2023), which scaled context to the millions magnitude by employing a distributed attention algorithm through several Tensor Processing Units. Nonetheless, these do not solve the inherent complexity challenges we have discussed. To this regard, during our project work, we intend to evaluate naturally efficient (sequence-wise subquadratic) architectures that are either adaptations of the traditional Transformer model or entirely novel architectures (SSMs), with further details within this section.

2.2.1. Sparse Attention

There have been several attempts to address this complexity issue. One of the most popular approaches is local or windowed attention, where each token only attends to a fixed-size window of neighboring tokens, effectively reducing the complexity from $\mathcal{O}\left(n^2\right)$ to $\mathcal{O}\left(nw\right)$, a more manageable formulation based on the window size, w. To expand on this idea, we must comment on the amount of overlap between adjacent windows, a design choice that effectively trades between language modeling performance and computational efficiency.

In the first variant, which we call Local Attention, there is no overlap, implying that token interactions can only be captured inside the w window. This simplifies the attention computation but is not as performant overall (Child et al. 2019). Here, the computational complexity for the entire sequence becomes $\mathcal{O}\left(nw\right)$, as each of the n tokens computes attention with w tokens once, optimizing the process especially when $w \ll n$.

On the other hand, Windowed (or Sliding Window) Attention features overlapping windows that shift one token at a time across the sequence. Each window overlaps with its predecessor by $w - 1$ tokens but maintains a computational complexity of $\mathcal{O}\left(nw\right)$. This overlapping nature allows for more continuous updates of context for each token, potentially improving the modeling of sequential data with finer attention resolution.



(a) Transformer (b) Sparse Transformer (strided) (c) Sparse Transformer (fixed)

Figure 2.2.: Local Attention from Child et al. 2019. In subfigure *a*, every token can attend to previous tokens. In subfigure *b*, each token can attend to up to w tokens before it, representing a fixed window size. In subfigure *c*, the number of attended tokens depends on the current time step i, specifically $w_i = i \mod w$, where w_i is the window size for step i.

These sparse attention techniques are particularly useful in scenarios where pro-

cessing long sequences is imperative, but computational resources are limited. They allow for significant reductions in time complexity compared to traditional attention mechanisms, though they may sacrifice some ability to capture long-range dependencies within the data. This trade-off is critical for tasks where local context suffices but may impact performance in tasks requiring comprehensive sequence understanding (Beltagy et al. 2020; Child et al. 2019).

2.2.2. Linear Transformers

While Windowed Attention boasts efficiency gains for large sequences by limiting the considered context to a local subset, it does not inherently alter the attention mechanism, as previously described in Equation 2.1. In fact, this solution seems quite restrictive in terms of modeling capacity: models that capture long-range interactions tend to outperform local ones (Beltagy et al. 2020).

Katharopoulos et al. 2020 depart from this line of thought and present a hybrid formulation between Transformers and Recurrent Neural Networks (RNN), pioneering the concept we now call linear Transformers that has since become its own research space. The authors' main contribution is detailing the connection between attention and a recurrent process, which we will now cover in detail.

First, we start by considering the equation for the attention mechanism as we presented in Equation 2.1, and note the role of the softmax as a similarity function between queries ($\mathbf{Q}$) and keys ($\mathbf{K}$), which we denote as $\text{sim}(\cdot)$. Taking this into consideration, we can rewrite the output of attention, $\mathbf{V}'$, by indexing it into an arbitrary i-th row:

$$\mathbf{V}'_i = \frac{\sum_{j=1}^{N} \text{sim}(\mathbf{Q}_i, \mathbf{K}_j)\mathbf{V}_j}{\sum_{j=1}^{N} \text{sim}(\mathbf{Q}_i, \mathbf{K}_j)}. \tag{2.3}$$

In this case, the similarity function, $\text{sim}(q, k)$ is defined as $\exp(\frac{q^\top k}{\sqrt{d}})$, recovering the softmax formulation. More generally, the only requirement we impose on a potential similarity function is for it to be non-negative (Tsai et al. 2019); thus, if we are to model $\text{sim}(\cdot)$ as a kernel, the valid design space would still be vast, the space of non-negative kernels $k(x, y) : \mathbb{R}^{2\times F} \to \mathbb{R}_+$ with the corresponding feature map as $\phi(\cdot)$.

As a side note, although not as a hard restriction, Zhang et al. 2024 comment on additional desirable properties for a similarity function that are intrinsic to the softmax: the monotonicity and low overall entropy (or "spikiness" as the authors name it) of the resulting attention weights. These factors are central to the performance of the overall attention operation and distinguish the softmax from other similarity maps (Choromanski et al. 2022; Qin et al. 2022).

Nonetheless, continuing with the introduced kernel notation, $k(x, y) = \phi(x)^\top \phi(y)$, we can rewrite Equation 2.3 as:

$$\mathbf{V}'_i = \frac{\sum_{j=1}^{N} \phi(\mathbf{Q}_i)^\top \phi(\mathbf{K}_j)\mathbf{V}_j}{\sum_{j=1}^{N} \phi(\mathbf{Q}_i)^\top \phi(\mathbf{K}_j)} = \frac{\phi(\mathbf{Q}_i)^\top \sum_{j=1}^{N} \phi(\mathbf{K}_j)\mathbf{V}_j^\top}{\phi(\mathbf{Q}_i)^\top \sum_{j=1}^{N} \phi(\mathbf{K}_j)}. \tag{2.4}$$

Moreover, considering the self-attention layer in decoder-only models, an additional causal mask prevents tokens from attending to their successors, which we have yet to model. This causal mask can be incorporated into the linear attention formulation by limiting the summation index to the current i position:

$$\begin{aligned} \mathbf{S}_i &= \sum_{j=1}^{i} \phi(\mathbf{K}_j)\mathbf{V}_j^\top, \qquad \mathbf{Z}_i = \sum_{j=1}^{i} \phi(\mathbf{K}_j), \\ \mathbf{V}'_i &= \frac{\phi(\mathbf{Q}_i)^\top \sum_{j=1}^{i} \phi(\mathbf{K}_j)\mathbf{V}_j^\top}{\phi(\mathbf{Q}_i)^\top \sum_{j=1}^{i} \phi(\mathbf{K}_j)} = \frac{\phi(\mathbf{Q}_i)^\top \mathbf{S}_i}{\phi(\mathbf{Q}_i)^\top \mathbf{Z}_i}. \end{aligned} \tag{2.5}$$

Finally, after the above steps and the definition of $\mathbf{S}_0$ and $\mathbf{Z}_0$ as per Equation 2.5, by unrolling the summation, we can formulate attention as a recurrent process. More concretely, for a batch of samples b, we have:

$$\begin{aligned} \mathbf{S}_0 &= 0, \qquad \mathbf{Z}_0 = 0, \\ \mathbf{S}_i &= \mathbf{S}_{i-1} + \phi(\mathbf{K}_i)(\mathbf{V}_i)^\top, \\ \mathbf{Z}_i &= \mathbf{Z}_{i-1} + \phi(\mathbf{K}_i), \\ \mathbf{V}'_i &= \frac{\phi(\mathbf{Q}_i)^\top \mathbf{S}_i}{\phi(\mathbf{Q}_I)^\top \mathbf{Z}_i}, \end{aligned} \tag{2.6}$$

where $\mathbf{Q}_i = \mathbf{X}_i\mathbf{W}_\mathbf{Q}, \mathbf{K}_i = \mathbf{X}_i\mathbf{W}_\mathbf{K}, \mathbf{V}_i = \mathbf{X}_i\mathbf{W}_\mathbf{V} \in \mathbb{R}^{b \times d}$.

In conclusion, while the traditional Transformer has $\mathcal{O}\left(n^2\right)$ time complexity with respect to sequence length, this new variant is linear during training, as we can compute $\sum_{j=1}^{N} \phi(\mathbf{K}_j)\mathbf{V}_j^\top$ and $\sum_{j=1}^{N} \phi(\mathbf{K}_j)$ from Equation 2.4 once per sequence and reuse the results for every query. Moreover, due to the recurrent formulation, the linear Transformer enjoys $\mathcal{O}\left(1\right)$ inference complexity (in the decoder case), making it ideal for real-time use cases.

2.2.3. Retentive Networks

Building on the insights from (Katharopoulos et al. 2020), Retentive Networks (abbreviated as RetNet, Sun et al. 2023) remove the softmax component from attention, allowing for a recurrent computation path. Retention behaves similarly to attention in the sense

that it captures interactions between **Q** and **K** entries. On the other hand, it is able to compress attention mass into neighboring token pairs via a fixed decay. In Figure 2.3, we can distinguish the alternative computation paths for the Retention component, where, in the rightmost figure, we note the γ decaying factor applied to the hidden state at each time step. Effectively modeling an exponential naive "forget" behavior directly without additional gating parameters.

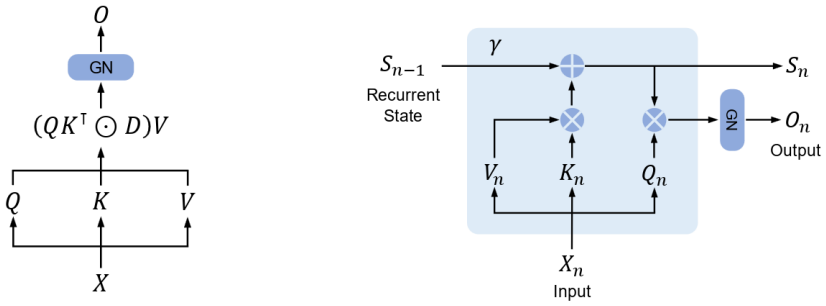


Figure 2.3.: RetNet formulations from Sun et al. 2023.

Formally, the alternative computation paths can be described using the same tensors as attention (Equation 2.1), with the addition of the decay mask and the XPos positional embeddings (Sun et al. 2022). To be concrete, the decay is modeled together with the customary causal mask in the $\mathbf{D} \in \mathbb{R}^{N \times N}$ matrix, which is elementwise multiplied by the $\mathbf{QK}^\top$ token interaction matrix as denoted through $\odot$. The positional embeddings, on the other hand, are modeled through Θ and $\overline{\Theta}$, which act as a rotation over the keys and queries, in a similar fashion to rotary embeddings (Su et al. 2023). Specifically, we can define Retention ($\mathbf{X}$) as:

$$\begin{aligned} &\mathbf{Q} = (\mathbf{X}\mathbf{W}_\mathbf{Q}) \odot \Theta, \qquad \mathbf{K} = (\mathbf{X}\mathbf{W}_\mathbf{K}) \odot \overline{\Theta}, \qquad \mathbf{V} = \mathbf{X}\mathbf{W}_\mathbf{V}, \\ &\Theta_n = e^{in\theta}, \qquad \mathbf{D}_{nk} = \begin{cases} \gamma^{n-k} & \text{if } n \geq k \\ 0 & \text{if } n < k \end{cases}, \\ &\text{Retention } (\mathbf{X}) = (\mathbf{QK}^\top \odot \mathbf{D})\mathbf{V}. \end{aligned} \tag{2.7}$$

Moreover, we can model the recurrent computation path by indexing (i) into the time dimension of the various tensors and accumulating the hidden state throughout the sequence. To note, the **D** decay mask is now decomposed into the constant γ, which gets compounded at each time step. The mathematical description follows:

$$\begin{aligned} \mathbf{S}_0 &= 0, \\ \mathbf{S}_i &= \gamma \mathbf{S}_{i-1} + \mathbf{K}_i^\top \mathbf{V}_i, \\ \text{Retention}(\mathbf{X}_i) &= \mathbf{Q}_i \mathbf{S}_i. \end{aligned} \tag{2.8}$$

Further detailing the model architecture, similar to Transformers and attention, Retention includes a multi-head extension of the retention mechanism entitled Multi-Scale Retention (MSR). Each head has its own decaying factor, allowing the model to capture up to h differently ranged interactions. As for the remaining overall architecture, we see components present in the modern Transformer, including the SwiGLU (Shazeer 2020) as the feed-forward component and RMSNorm (Zhang et al. 2019) as the normalization layer. In this work, we replace the self-attention component in decoder-only Transformers with the MSR component.

Finally, to conclude this section on RetNet, it is worth noting that the model allows for an in-between computation path entitled "Chunkwise Retention". This mode behaves similarly to the recurrent view but with chunks of the sequence being computed at each recurrence step. We refrain from further detailed descriptions as we didn't make use of this mode in our experiments.

2.3. Space State Models

Deep SSMs (Gu et al. 2020; Gu et al. 2022a; Gu et al. 2022b), represent a novel class of deep learning architectures that integrate principles from RNN, convolutional neural networks, and classical state space models from control theory (Basar 2001). Like the previous linear transformers, SSMs allow for both recurrent and parallel views for efficient inference and training.

The central premise for SSMs is modeling context through a continuous hidden state $h(t)$ that is transformed through a transition matrix $\mathbf{A}$ and updated with a linear input signal projection. This notion constitutes the update rule for a time step $t + \epsilon$ at the continuous level as:

$$\begin{aligned} h'(t) &= \mathbf{A}h(t) + \mathbf{B}x(t), \\ y(t) &= \mathbf{C}h(t). \end{aligned} \tag{2.9}$$

Since we are working with discrete inputs like text tokens, this signal must be discretized before being applied in the SSM model. This step can be achieved through the zero-order hold method or the bilinear method (Tustin 1947), with the first being

detailed as follows:

$$\begin{aligned}\overline{\mathbf{A}} &= \exp(\Delta\mathbf{A}),\\ \overline{\mathbf{B}} &= (\Delta\mathbf{A})^{-1}(\exp(\Delta\mathbf{A}) - \mathbf{I}) \cdot \Delta\mathbf{A},\\ \overline{\mathbf{C}} &= \mathbf{C}.\end{aligned} \tag{2.10}$$

Lastly, we note the inclusion of an additional Δ parameter, which models the continuous "time" between contiguous tokens. The SSM formulation used in the S4 family of models is described in discrete form:

$$\begin{aligned}h_t &= \overline{\mathbf{A}}h_{t-1} + \overline{\mathbf{B}}x_t,\\ y_t &= \overline{\mathbf{C}}h_t.\end{aligned} \tag{2.11}$$

While this recurrent formulation is ideal for inference settings, with associated $\mathcal{O}(1)$ complexity (with respect to sequence length), it needs to be further parallelized for training. In this regard, we can unroll the recurrence and note that Equation 2.11 can be reformulated as a convolution. However, this is only possible because SSMs do not depend on the current time step for computations, which is known as the Linear Time Invariance (LTI) property. The unrolled recurrence and convolutional view can be described as follows:

$$\begin{aligned}y_k &= \overline{\mathbf{C}}\overline{\mathbf{A}}^k\overline{\mathbf{B}}x_0 + \overline{\mathbf{C}}\overline{\mathbf{A}}^{k-1}\overline{\mathbf{B}}x_1 + \cdots + \overline{\mathbf{C}}\overline{\mathbf{A}}\overline{\mathbf{B}}x_{k-1} + \overline{\mathbf{C}}\overline{\mathbf{B}}x_k,\\ Y &= \overline{\mathbf{K}} * x,\\ \overline{\mathbf{K}} \in \mathbb{R}^L &= (\overline{\mathbf{C}}\overline{\mathbf{B}}, \overline{\mathbf{C}}\overline{\mathbf{A}}\overline{\mathbf{B}}, \ldots, \overline{\mathbf{C}}\overline{\mathbf{A}}^{L-1}\overline{\mathbf{B}}).\end{aligned} \tag{2.12}$$

Furthermore, if the convolution kernel $\overline{\mathbf{K}}$ is known, the whole convolution can be computed efficiently through the Fast Fourier Transform with $\mathcal{O}(n \log n)$ complexity; however, since $\overline{\mathbf{A}}$ is not diagonal, this is not trivial. A numerically stable and efficient algorithm for this problem is the main contribution in (Gu et al. 2022a), and we refer to it for further details.

To conclude, we further comment on the initialization of the $\mathbf{A}$ matrix, which requires a special structure (hence the structured state spaces nomenclature) in order for learning to be effective and converge smoothly, as noted in HiPPO, S4D and LRU (Gu et al. 2020; Gu et al. 2022b; Orvieto et al. 2023). Concretely, while the original HiPPO framework required a specific structure and complex initialization, the latest formulation in S4D is able to simplify this procedure towards a diagonal matrix, allowing for simpler and faster kernel computation. Additionally, while the latter initialization is still complex, for text and discrete modalities in general, it is sufficient to retain the real part, enabling the use of 32-bit and lower precision data types instead of the 64-bit required for complex values.

2.3.1. Mamba

Mamba (Gu et al. 2023) presents a successor to the S4-class family of models that solves one of its key issues. Until now, SSM models were all LTI, meaning that the SSM process was not data-aware and thus could not reset or overwrite the hidden state based on particular inputs. This behavior is a known limitation, dating back to the introduction of LSTMs (Hochreiter et al. 1997) that superseded RNNs.

Akin to the gating mechanisms in LSTMs, Mamba makes the SSM parameters $(\mathbf{B}, \mathbf{C}, \Delta)$ linear time-variant, *i.e.*, there are additional linear projections dependent on the current input at time step t. To illustrate the differences between the S4 and Mamba SSM layers, we decompose each model's forward pass in Algorithm 1 and Algorithm 2, respectively. Marked in red are the changes in the computation process; by taking into account the current time step, the $\mathbf{B}, \mathbf{C}$, and Δ tensors gain additional dimensions: batch size B and sequence length L.

Algorithm 1 SSM (S4)

Input: $x : (B, L, D)$
Output: $y : (B, L, D)$
$\mathbf{A} : (D, N) \leftarrow$ Parameter
▷ Structured $N \times N$ matrix
$\mathbf{B} : (D, N) \leftarrow$ Parameter
$\mathbf{C} : (D, N) \leftarrow$ Parameter
$\Delta : (D) \leftarrow \tau(\text{Parameter})$
$\overline{\mathbf{A}}, \overline{\mathbf{B}} : (D, N) \leftarrow \text{discretize}(\Delta, \mathbf{A}, \mathbf{B})$
$y \leftarrow \text{SSM}(\overline{\mathbf{A}}, \overline{\mathbf{B}}, \mathbf{C})(x)$
▷ Time-invariant: recurrence or convolution
return y

Algorithm 2 SSM + Selection (S6)

Input: $x : (B, L, D)$
Output: $y : (B, L, D)$
$\mathbf{A} : (D, N) \leftarrow$ Parameter
▷ Structured $N \times N$ matrix
$\mathbf{B} : (B, L, N) \leftarrow S_B(x)$
$\mathbf{C} : (B, L, N) \leftarrow S_C(x)$
$\Delta : (B, L, D) \leftarrow \tau(\text{Parameter} + S_A(x))$
$\overline{\mathbf{A}}, \overline{\mathbf{B}} : (B, L, D, N) \leftarrow \text{discretize}(\Delta, \mathbf{A}, \mathbf{B})$
$y \leftarrow \text{SSM}(\overline{\mathbf{A}}, \overline{\mathbf{B}}, \mathbf{C})(x)$
▷ Time-varying: recurrence (scan) only
return y

While this change turns the models much more expressive, it breaks the convolution view that enabled fast training on the previous models. To circumvent this issue, Mamba resorts to a hardware-aware selective scan algorithm inspired by the recently popularized parallel associative scan from (Blelloch 1990; Martin et al. 2018; Smith et al. 2023) and from the FlashAttention line of work (Dao et al. 2022; Dao 2023). This algorithm allows for efficient training by computing the various sequence elements in parallel and fusing the results in a tree-like structure, therefore maintaining the $\mathcal{O}(n \log n)$ time complexity from S4, albeit through a different algorithm. We refer to Blelloch 1990; Martin et al. 2018; Smith et al. 2023 for further details on the associative scan procedure.

Notably, Figure 2.4 illustrates the deliberate use of different Graphics Processing Unit (GPU) memory hierarchies to optimize the overall Mamba computation. In

addition, we comment on data-dependent **B**, **C** and Δ parameters resulting from linear projections of the input signal. On the other hand, the **A** matrix is not exactly input dependent but is influenced by the Δ discretization factor as detailed in Equation 2.10. The authors justify the need for keeping all of these components selective through ablations in (Gu et al. 2023).

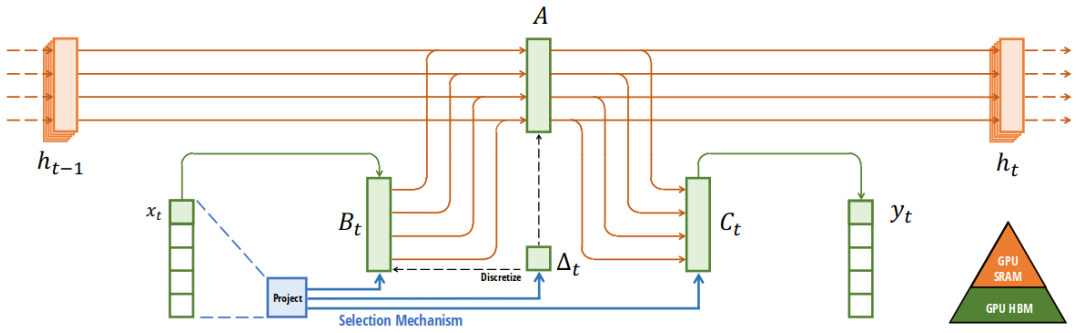


Figure 2.4.: Mamba recurrence step from $t-1$ to t (Gu et al. 2023). As denoted in the pyramid, the different colors represent different levels of GPU memory hierarchy: GPU SRAM is smaller but has a higher bandwidth than GPU HBM.

Architecture. After detailing the base component for the Mamba architecture, the inner SSM, we now turn to its overall architecture, which draws inspiration from several other models (Fu et al. 2023; Shazeer 2020; Hua et al. 2022). Traditionally, transformers are composed of stacked layers, with the two main components being the self-attention mechanism and the feed-forward component. The latter has recently become standardized as a SwiGLU layer (Shazeer 2020) as seen in Figure 2.5 as *Gated MLP*.

Lastly, observe that the SSM component is a standalone sequence transformation that can incorporated quite flexibly into modern neural networks. In turn, the most common overall SSM block is the H3 block, which includes a convolution before the SSM component (*H3* in Figure 2.5). In an effort to produce a simple architecture combining the two, SwiGLU and H3, the Mamba block is based on the Gated MLP but replaces the residual branch with the Conv + Activation + SSM branch. Figure 2.5 is illustrative of the overall block. Stacking this block produces the overall Mamba architecture, along with the inclusion of residual branches connecting each model layer and normalization components placed before each block, in this case, LayerNorm (Ba et al. 2016).

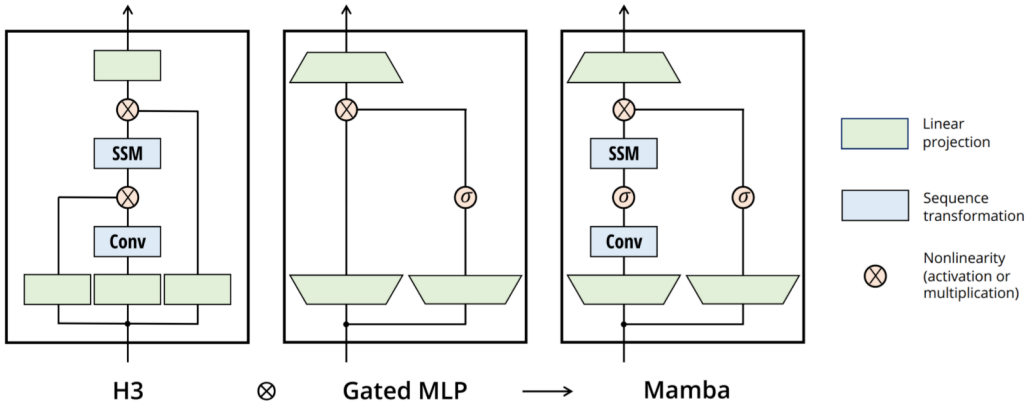


Figure 2.5.: Mamba block from Gu et al. 2023.

2.4. Machine Translation

Lastly, we present an overview of the MT task, detailing how it is usually formulated and the corresponding optimization objectives.

2.4.1. Task Definition

MT models are fundamentally designed to maximize the probability of generating a correct target sequence y given a source sequence x. This is mathematically represented as the conditional probability $P(y|x)$ which can be decomposed as:

$$P(y|x) = \prod_{t=1}^{T} P(y_t|x, y_{<t}), \tag{2.13}$$

where y_t is the t-th word in the target sequence and $y_{<t}$ represents all previous words. This model is well-aligned with the standard Encoder-Decoder Transformer decomposition, where the encoder portion is tasked with generating a strong embedding representation for the source sentence while the decoder generates the target sequence, conditioned on the source through the Cross-Attention module. Particularly, we can denote the encoder as enc_θ, which processes the input and produces the corresponding hidden state $h = \text{enc}_\theta(x)$. Whereas the decoder (dec_ϕ) is responsible for computing the conditional probability of every subsequent word in the target sequence, $P(y_t|x, y_{<t}) = \text{dec}_\phi(y_t|h, y_{<t})$. Moreover, we can depict the negative log-likelihood of this probability, as typically used as a loss function during training:

$$-\log P(y|x) = \sum_{t=1}^{T} -\log P(y_t|x, y_{<t}). \tag{2.14}$$

Nonetheless, the advances in decoder-only language models, with models displaying strong language understanding and multilinguality (Brown et al. 2020; Chowdhery et al. 2022) led to the question of whether this encoder-decoder split is necessary (Gao et al. 2022; Wang et al. 2021). In fact, we can still ground the generation in the source context by prefixing it to the target sentence. This formulation allows us to pose MT as a LM problem by integrating the encoding of the input and the generation of the output into a single probabilistic model, simplifying the overall problem:

$$P(x, y) = \prod_{s=1}^{S} P(x_s|x_{<s}) \prod_{t=1}^{T} P(y_t|x, y_{<t}). \tag{2.15}$$

The corresponding loss function for these models combines the auto-encoding loss ($\mathcal{L}^{AE}$) for the input with the translation loss ($\mathcal{L}^{MT}$) for the output:

$$-\log P(x, y) = \mathcal{L}^{AE} + \mathcal{L}^{MT} = -\sum_{s=1}^{S} \log P(x_s|x_{<s}) - \sum_{t=1}^{T} \log P(y_t|x, y_{<t}). \tag{2.16}$$

Additionally, with Transformer-like models that enable bidirectional encoding (time-wise), we can alternatively model translation in a PrefixLM setting (Raffel et al. 2023), simplifying the $\mathcal{L}^{AE}$ term to $-\log P(x)$. However, this term is usually ignored during training (Xu et al. 2023; Vardasbi et al. 2023).

As a final consideration, we also comment on the potential of using instruction-tuned models to facilitate translation through zero or few-shot prompting (Wei et al. 2022; Chowdhery et al. 2022). This approach capitalizes on the extensive pre-training these models have undergone, which is designed to equip them with a broad comprehension of language and a flexible adaptation to new tasks via in-context learning. This formulation does not deviate substantially from the probabilistic model in Equation 2.15, additionally including terms for the instruction and provided examples.

2.5. Related Work

Having explored the required background underlying this thesis project, we now detail the key ideas and related work we build upon. Succinctly, these include training machine learning models to perform MT, evaluating the models' performance in long-context scenarios, and an overall analysis of generation and model quality for MT.

2.5.1. Machine Translation Training

After the discussions in Section 2.4, we refer to some of the training choices that typically distinguish MT work. Initially, we distinguish two approaches: the traditional training from the scratch method, which we employ in this project, and the more recent fine-tuning of pre-trained LLMs. The latter has proven highly successful, given the results reported in (Xu et al. 2023) and (Alves et al. 2024). Both works have a two-stage training process, the first of which is to train the model further on multilingual data to expand its language awareness, followed by either parallel or instruction-based high-quality data to bridge its internal language representations. Particularly, Tower (Alves et al. 2024), which leverages LLaMa 2 7B and 13B (Touvron et al. 2023b), achieves competitive MT results with the much larger GPT-4 (OpenAI et al. 2024).

In this project, we opt for the training-from-scratch approach, which nevertheless offers several advantages. First, it enables a direct comparison with existing literature (Vardasbi et al. 2023). Secondly, it allows us to rigorously ablate architectural modifications without the interference of pre-training conditions. Finally, it allows us to include a more extensive range of models in our testing, thanks to the significantly smaller model size. In the training-from-scratch approach, the work by Vardasbi et al. 2023 is particularly relevant, as it focuses on efficient SSM models, which are of primary interest to us. Their models, trained on parallel WMT data, demonstrate the performance differences between standard Transformers and efficient SSMs, such as the S4 model (Gu et al. 2022a), the predecessor of Mamba (Gu et al. 2023). Moreover, their hybridization approaches exhibit strong performance, motivating our own hybrid experiments, which are also informed by other recent studies that combine the efficiency of linearized models with the modeling capacity of attention mechanisms. Notable examples include Griffin (De et al. 2024), Based (Arora et al. 2024), and Jamba (Lieber et al. 2024).

2.5.2. Long Context Model Evaluation

Through the use of paragraph-level translation data, we leverage the model's ability to understand long-context scenarios. This angle has been studied extensively, notably in Long Range Arena (Tay et al. 2020), a popular long-range sequence modeling benchmark that does not include a proper generative language task, distancing it from the real-world evaluation we intend to conduct.

Most existing work on measuring context awareness in language models relies on synthetic benchmarks. For instance, the Needle-in-a-Haystack task (Kamradt 2023) and passkey retrieval (Mohtashami et al. 2023) prompt models to recover information hidden within a context window filled with unnecessary tokens. However, our models

are not designed for instruction following, making these benchmarks unsuitable for evaluation. In contrast, MT offers direct quality assessment metrics, enabling us to measure model performance across different sequence lengths and examine how it changes when trained on varying sequence length distributions. To modify the training length distribution, we adopt a strategy similar to (Kondo et al. 2021), concatenating random sentences to alleviate potential issues with extrapolation in different sequence length scenarios.

2.5.3. Understanding Model Capabilities

The final aspect of this project involves understanding each model's strengths and weaknesses based on the computed metrics and generated outputs. To guide our analysis, we draw on recent studies of interpretability and analysis for efficient (linear) models in contrast with traditional Transformers. Specifically, recent research has shed light on the limitations of efficient models, inspiring our methodology and analysis. For instance, Akyürek et al. 2024 construct a synthetic language through automata, and measure how well models are able to learn and replicate it through in-context learning. Since our models lack the flexibility of pretrained or instruction-tuned models, similar experiments in MT are unfeasible. Nevertheless, the work in (Arora et al. 2023) has led us to investigate the recall capabilities of our models in a more applicable scenario. Their MQAR benchmark measures a model's ability to perform associative recall in a toy task that requires training. We employ this idea but test it using the recall of Named Entities, which appear both in the source and target and require recollection to produce a successful translation.

3. Experiments

With the premise of evaluating the MT performance of SSM models, we design an ample set of experiments covering both sentence-level translation and expanding context to paragraphs. Given the efficiency vector of this thesis, paragraph analysis is crucial since subquadratic models display their strengths over longer sequences of text. Meanwhile, Transformer models are already quite efficient for the shorter sentence scenario by leveraging efficient GPU matrix multiplication code.

At the sentence level, we aim to get a preliminary analysis of the workings of the efficient models and how they stack up against their conventional counterparts. Expanding this to the paragraph level, the goal shifts to understanding not only how they perform but also how well they are able to make use of context. This is especially relevant when working with RNN-style models that, unlike attention, cannot directly access the previous tokens. Their performance in these tasks is directly tied to how well they are able to compress sequence information into their hidden state.

Furthermore, following recent literature, we experiment with hybrid architectures where an attention component is mixed in with the otherwise subquadratic models. We believe this to be a promising line of work following the reported shortcomings of linear models (Arora et al. 2023; Jelassi et al. 2024) in tasks where attention excels.

3.1. Datasets

We focus on two high and low-resource language pairs, German-English (*de-en*) in WMT14 (Bojar et al. 2014) and Romanian-English (*ro-en*) in WMT16 (Bojar et al. 2016), respectively. We perform our set of experiments on both translation directions. These sentence-level datasets were chosen given their popularity in current literature, making our results comparable to those in (Vardasbi et al. 2023). Moreover, we further use IWSLT17 (Cettolo et al. 2017) and the *de-en* language pair for architectural validation purposes. Given the smaller number of samples, this dataset is adequate for running small-scale experiments designed to validate design choices and perform hyperparameter tuning to a better degree, akin to what was done in (Peters et al. 2021; Correia et al. 2019). Focusing on the paragraph level, we turn to WMT23 (Kocmi et al. 2023) and focus on the *de-en* language pair in both directions. The choice is natural, given

the abundance of data when compared with other language pairs and, importantly it being the only non-sentence-breaked test set.

WMT23 *de-en*, in its total capacity, equates to around 300M samples, meaning thoroughly evaluating this dataset would be out of the scope of the compute allocated to this project. Moreover, the quality improvement of training small-scale models against such a large dataset is dubious. To solve this challenge, we sample 6M high-quality samples obtained from scoring the dataset using COMET-QE 22[1] (Rei et al. 2022b), a strong neural quality estimation metric. More precisely, we sort the scored samples according to translation quality and select the top 6M; we refer to this dataset as *WMT23-6M*.

While this language pair is paragraph level in the test set, the training data are mostly single sentences. To compound this effect, we found that our models had poor length extrapolation performance, resulting in low overall performance (more details in Chapter 4) in a standard training procedure. To circumvent this issue, we further constructed "paragraph" level datasets by concatenating both 5 and 10 random samples from the previous 6M dataset. This means that the training sentence length distribution would now be closer to that of the test set. As for validation, we use Ted Talks data (Qi et al. 2018), which is also paragraph-level. We refer to these concatenation datasets as *WMT23-CAT-5* and *WMT23-CAT-10*. Readers can find complete dataset statistics in Table 3.1.

Dataset	Size	Mean	Std. dev	Max. len.
IWSLT17 (*de-en*)	200k	45.2	29.5	335
WMT16 (*ro-en*)	610k	58.9	31.1	927
WMT14 (*de-en*)	4.5M	62.1	45.6	17107
WMT23-6M (*de-en*)	6M	58.4	32.9	753
WMT23-CAT-5 (*de-en*)	2M	171.3	134.9	1965
WMT23-CAT-10 (*de-en*)	1M	312.4	282.3	2982
Ted Talks Validation (*de-en*)	995	268.5	189.6	1370
WMT23 Test (*de-en*)	543	135.1	147.7	1174
WMT23 Test (*en-de*)	557	185.2	188.2	1234

Table 3.1.: Dataset statistics for both sentence and paragraph-level datasets.

[1] `https://huggingface.co/Unbabel/wmt22-cometkiwi-da`

3.2. Metrics and Evaluation

Evaluating machine translations is a well-established research branch with a variety of effective methods. For this project, we utilize automated metrics for assessment purposes at both the sentence and paragraph levels. Evaluation is traditionally conducted using SacreBLEU (Post 2018). Moreover, besides lexical analysis with BLEU, we incorporate neural metrics such as COMET, which is recommended in recent literature due to its strong correlation with human judgments (Freitag et al. 2022).

Throughout our results and analysis sections, we report SacreBLEU with the default signature (`|nrefs:1|case:mixed|eff:no|tok:13a|smooth:exp|`) and COMET 22 scores[2]. This choice arises from the need to compare with existing literature and because, when combined, these two metrics provide good MT quality estimation, both in terms of lexical and semantic integrity with the neural approach. Moreover, despite the relative scale difference between the two reported metrics, the delta between them is still comparable, as seen in (Kocmi et al. 2024).

In conclusion, we discuss the generation and evaluation aspects of the conducted project work: during training, we compute BLEU scores based on the validation set, keeping the two best-performing checkpoints and selecting the best overall for testing where we compute both BLEU and COMET; for generation, we use greedy decoding in for validation and Beam Search with a beam size of 5 for testing.

3.3. Models

With the purpose of establishing a proper ranking of MT performance, we select an ample set of models for evaluation, including Transformers, SSMs, and hybrid models, which leverage the attention layer to complement the SSM architecture.

3.3.1. Transformers

We select 2 variants of the Transformer model as a baseline: one base Encoder-Decoder formulation and a modern decoder-only version, which we will detail below.

Transformer Encoder-Decoder (Transformer Enc-Dec). Base model from the original paper (Vaswani et al. 2017), model size is 77M parameters. Includes the usage of sinusoidal positional embeddings and standard ReLU activations.

[2] `https://huggingface.co/Unbabel/wmt22-comet-da`

Transformer++. Decoder-only Transformer formulation including advancements such as Rotary positional embeddings (Su et al. 2023) and the usage of SwiGLU as the feed-forward component (Shazeer 2020). Both of these have become standard in contemporary large models. Thus, we select the LLaMa architecture (Touvron et al. 2023a) as a representative of the strong Transformer formulation, reducing the embedding dimension to roughly match the parameter count of the base Transformer (79M). Notably, LLaMa 2 (Touvron et al. 2023b) further introduces Grouped Query Attention (Ainslie et al. 2023), but we refrain from using it to achieve parity with the model denoted as *Transformer++* in (Gu et al. 2023).

Additionally, we consider whether the Transformer benefits from decomposing its layers into an Encoder-Decoder setting. Vardasbi et al. 2023 report that the optimal parameter allocation for MT is exactly 50-50% between encoder-decoder. We validate this claim by evaluating on IWLST17 (Table 3.2), our validation dataset, where we find that the decomposed variation outperforms its uniform counterpart.

	de-en		en-de	
Model	BLEU	COMET	BLEU	COMET
Transformer Encoder-Decoder	30.56	77.28	25.55	73.05
Transformer Decoder	28.91	75.96	23.20	69.86

Table 3.2.: Comparison between Encoder-Decoder and Decoder-only Transformer on IWLST17.

3.3.2. State Space Models

We select two representative SSM models based on two factors, whether the recurrent hidden state updates are based on the current token or invariant throughout the sequence, which we had previously denoted as the LTI property, and the claimed language modeling performance. These models are RetNet (Sun et al. 2023) and Mamba (Gu et al. 2023), two LTI and Linear Time Variance (LTV) models, respectively.

We hypothesize that LTI is the main differentiating factor between these types of models, as controlling the degree of information to keep from the hidden state and to absorb from the new token is a significant modeling advantage. While RetNet prioritizes interactions between close tokens, given the decay factor in its positional encoding, Mamba is able to learn interaction patterns through backpropagation of the **A** transition matrix.

Other good choices here would be H3 (Fu et al. 2023) and GLA (Yang et al. 2024), which are input-independent (LTI) and input-dependent (LTV), respectively. H3 was the first release that noted the combined strength of SSMs and attention, retaining two layers of self-attention. We refrain from including it due to Mamba's reported higher LM performance. On the other hand, GLA can be seen as a successor to the RetNet architecture by including an additional gating mechanism on the hidden state, making it behave similarly to Mamba. This decision was influenced by the timing of GLA's release, which coincided with the ongoing project work.

RetNet. This model is tested with a total of 77M parameters to match the previous Transformer models. To achieve this, the following hyperparameters are used: `|d_model:512|d_values: 1024|ffn_dim: 1024|heads: 4|layers: 12|`. In particular, we set `ffn_dim = d_values` and `heads: 4` as the proportions and time scale used by the authors, implying that we can capture interactions between 4 different time scales. While we utilized the official model implementation for this project, we had to additionally contemplate how to disregard padding tokens, an issue in our MT setting but not in the original LM work. Further details on the padding implementation can be found in Appendix A.

Mamba. Likewise, we use the official implementation, but with a branch that includes left padding [3]. Additionally, we add a dropout module to the inner Mamba-SSM class as seen in Figure 2.5. The resulting model is 77M parameters in size when built with the following hyperparameters: `|d_model: 610|layers: 24|`. Mamba is a homogeneous architecture with no feed-forward component therefore, the layer count is doubled compared to the other decoder-only models.

3.3.3. Hybrids

Building on the reported shortcomings of linear models (Akyürek et al. 2024; Arora et al. 2023; Jelassi et al. 2024), we additionally design hybrid models in order to complement SSMs with attention. The ability to recall previously seen tokens is one of the most significant advantages of the attention mechanism and, as reported, is a fundamental issue in SSMs. As such, we add three additional models to our MT evaluation test suite, **Mamba-MHA**, **Mamba-Local** and **Mamba Encoder-Decoder** (Mamba Enc-Dec), which will be specified next.

Incorporating attention to linear models has been done previously (Poli et al. 2023; Fu et al. 2023; Arora et al. 2024; De et al. 2024) with various formulations, and it has

[3] `https://github.com/state-spaces/mamba/pull/70`

been found to increase performance by a substantial margin. Now, although the hybrid architecture design space is quite vast, we settle on these three variations to limit the total number of experiments and because they are sufficiently illustrative of the performance differences between the base architectures and the ones with additional attention modules.

Mamba-MHA. The simplest hybrid formulation includes replacing some of the Mamba layers with attention modules. Some natural questions then arise: where to place them and how many attention modules do we need? We ablate this in Table 3.3 and provide some interesting takeaways. As discussed, based on our analysis of the COMET scores, it seems 2 attention layers are enough to greatly boost the model's performance, agreeing with the reports in H3 (Fu et al. 2023). On the other hand, their placement seems to have little effect, as the performance delta is small. In the end, we select version *(2 layers - 11,23)* for the rest of the experiments in this thesis, given its consistently higher COMET scores.

	de-en		*en-de*	
Model (Mamba-MHA)	BLEU	COMET	BLEU	COMET
0 layers - Mamba	28.10	76.63	22.90	71.75
12 layers - Interleaved	30.81	77.98	24.40	72.48
2 layers - 1,11	30.52	78.10	**24.99**	73.76
2 layers - 11,23	**30.92**	**78.30**	24.71	**73.94**

Table 3.3.: Mamba-MHA BLEU and COMET scores per number and location of Attention modules. *Interleaved* refers to alternating Mamba and Attention layers, *11,23* refers to placing attention in layers $N/2$ and N and *1,11* to layers 2 and $N/2$. We include scores for the homogeneous Mamba model. **Bold** represents top results.

Mamba-Local. In light of the resulting model's inference time complexity, we consider local attention variants such as Windowed Attention (Beltagy et al. 2020; Child et al. 2019), utilized in prominent models like Mistral (Jiang et al. 2023) and recent hybrid models, Based (Arora et al. 2024) and Griffin (De et al. 2024). While aiming to achieve robust performance, the introduction of full attention in Mamba disrupts its inference constant runtime (with respect to sequence length), which is a significant advantage of Mamba over traditional Transformers. On the other hand, Windowed Attention

preserves this property by only attending to the w preceding tokens. Although these models exhibit reduced performance compared to full-dense attention, their overall efficiency deserves careful analysis.

When determining the appropriate window size for this MT context, we observe a range of practices. While language models like Mistral use a large window size of 4,096 tokens, effectively simulating full attention, Based utilizes a more moderate 64-token window. This smaller window size is deemed sufficient for capturing near-token interactions while allowing the linear components of the model to handle long-range dependencies. Furthermore, the choice of window size also impacts hardware efficiency. Based reports that window sizes smaller than 64 do not fully leverage the capabilities of modern GPUs, which are optimized with Tensor Cores designed to enhance parallel computation.

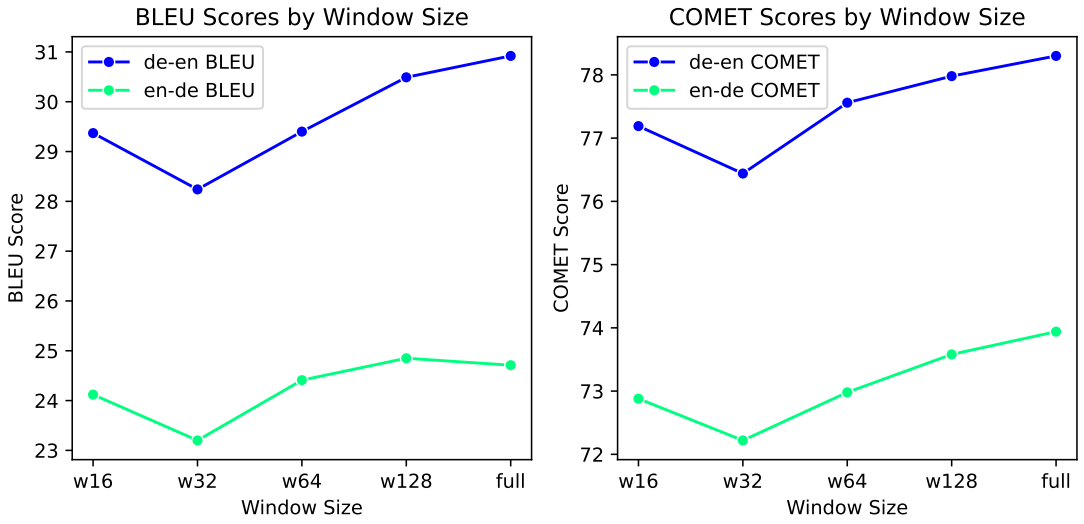


Figure 3.1.: Model scores per window size. Different window sizes are denoted as $w\{16, 32, 64, 128\}$ while *full* refers to the full self-attention from Mamba-MHA.

We ablate this phenomenon in Figure 3.1 and note the model linearly increases performance as the window size grows. We further remark on the anomaly at window size 32, where performance consistently decreases across both translation directions and metrics, but do not explore this further.

Furthermore, the choice of keeping only two attention layers might not hold for the local attention case, so we revisit the interleaving (windowed) attention layers experiment, as can be seen in Table 3.4 where we additionally display all BLEU and

COMET scores from Figure 3.1. All in all, we note that 128 window size mostly recovers full attention and that the configuration with 2 layers of 64 window size attention is a good trade-off between model performance and efficiency, thereby using it in future experiments.

	de-en		*en-de*	
Model (Mamba-Local)	BLEU	COMET	BLEU	COMET
11,23 - w16	29.37	77.19	24.12	72.88
11,23 - w32	28.24	76.44	23.20	72.22
11,23 - w64	29.40	77.56	24.41	72.98
11,23 - w128	30.49	77.98	24.85	73.58
Interleaved - w64	28.85	76.76	23.61	72.10
11,23 - Full	30.81	78.30	24.40	73.94

Table 3.4.: Mamba-Local BLEU and COMET scores per window size. Different window sizes are denoted as $w\{16, 32, 64, 128\}$ while *Full* refers to the full self-attention from Mamba-MHA. *Interleaved* refers to alternating Mamba and Local Attention layers, *11,23* refers to placing attention in layers $N/2$ and N.

Mamba Enc-Dec. Lastly, we emulate the *S4-S4A* experiment in (Vardasbi et al. 2023), replacing the encoder and self-attention portions of the Transformer Encoder-Decoder with Mamba blocks. Figure 3.2 serves as an illustration for this architecture. By separating the encoding and decoding tasks to different Mamba layer stacks and incorporating the Cross-Attention module to propagate information between them, we should make the model more robust to the aforementioned recall issues.

3.4. Training

All models have been trained on a single Nvidia RTX A6000[4] with 48GB VRAM. Training batch size has been kept constant at 64, using gradient accumulation to accommodate different sequence sizes and memory requirements, which is especially relevant when working with paragraph-level MT. Moreover, we use Pytorch Lightning[5] as the

[4] `https://www.nvidia.com/en-gb/design-visualization/rtx-a6000/`
[5] `https://lightning.ai/docs/pytorch/stable/`

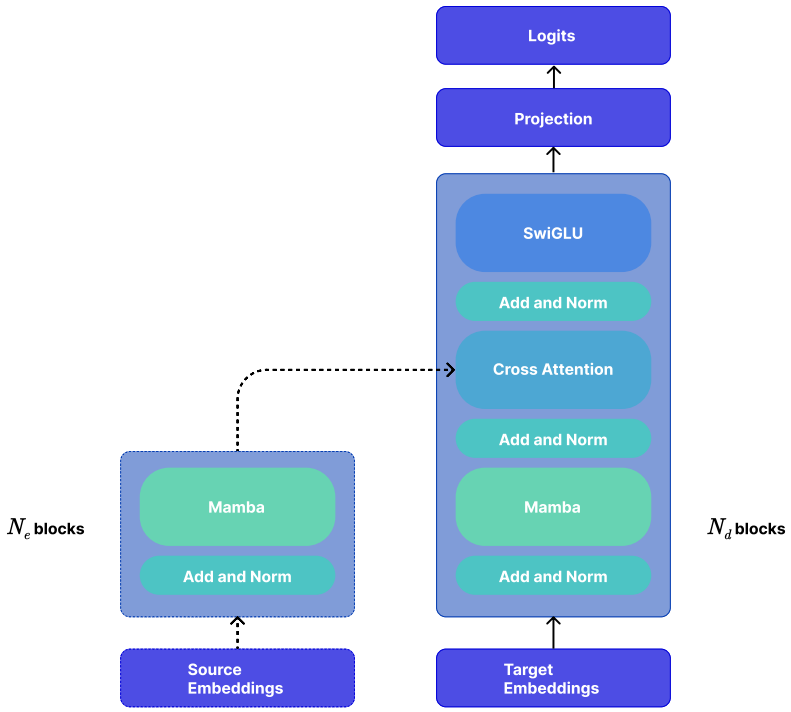


Figure 3.2.: Mamba Encoder-Decoder illustrative architecture. The encoder stack is composed of N_e blocks and the decoder from N_d blocks.

training framework and Weights and Biases[6] for monitoring experiments. Regarding tokenization, we leverage the HuggingFace *tokenizers* library[7] and construct a separate BPE tokenizer (Sennrich et al. 2016) per dataset. The total vocabulary size is 32000 tokens.

Given the extensive suite of runs to be conducted, we tried to limit the hyperparameter space so that the tuning procedure did not occupy the majority of the project's duration. Even still, we have to account for different learning rates between models and different Dropout (Srivastava et al. 2014) percentages between datasets. In contrast, we keep some essential hyperparameters and training choices constant between runs: our Learning Rate (LR) scheduler is the **Inverse Square Root** (Vaswani et al. 2017) scheduler popularized by the original Transformer publication; we use 4000 **warmup steps**; we use low **Weight Decay** (Goodfellow et al. 2016) (0.001) given initial experimental results; all of the models are trained with `bfloat16` **precision**; we modify **Dropout** values based on the dataset in question, 0.3 for IWSLT17 *de-en* and WMT16 *ro-en*, and 0.1 for WMT14

[6] `https://wandb.ai/site`
[7] `https://github.com/huggingface/tokenizers`

de-en and the different variations of WMT23 *de-en*. We include the full hyperparameters set based on each model in Table 3.5.

Model	Size	LR	L	H	D	FFN	Misc.
Transformer models							
Transformer Enc-Dec	77M	$4e-4$	$6-6$	8	512	2048	
Transformer++	79M	$4e-4$	12	8	496	1984	
SSMs							
RetNet	77M	$1e-3$	12	4	512	1024	
Mamba	77M	$1e-3$	24	$\times$	610	$\times$	
Hybrid models							
Mamba-MHA	78M	$7e-4$	24	8	624	$\times$	
Mamba-Local	78M	$7e-4$	24	8	624	$\times$	`window_size: 64`
Mamba Enc-Dec	82M	$7e-4$	$8-6$	8	512	2048	

Table 3.5.: Detailing the full set of hyperparameters for the different models. Encoder-Decoder models have their number of layers separated by each module. LR represents the Learning Rate; L represents the number of layers; H is the number of Attention Heads; D is the model dimension; Miscellaneous (Misc.) is for extra, model-specific hyperparameters.

As a last note, during training for decoder-only models, we optimize the joint probability based on the objective detailed in Equation 2.16. We format our data by combining source and target samples with a specific separator token, formatted as `"{source}<SEP>{target}"`, and focus exclusively on minimizing the target cross-entropy loss. Due to the recurrent characteristics of our Mamba and RetNet models, we do not use the PrefixLM objective. Additionally, our Encoder-Decoder models are trained according to the objective outlined in Equation 2.14.

4. Results

Following the detailed account of models and datasets in the previous chapter, we proceed to present the results and analysis stemming from the performed experiments. We start by evaluating our models in a sentence-level setting in Section 4.1 and move to the paragraph-level in Section 4.2. Additionally, in Section 4.3, we perform a deeper analysis of the resulting translations, particularly with respect to understanding the impact of sequence length on quality, as measured through our automated metrics. Lastly, in Section 4.4, we leverage the reproduction of NEs from source to target sentences to evaluate the models' recall performance.

4.1. Sentence-level Machine Translation

As a first step, we evaluate our standard and hybrid models on WMT16 *ro-en* and WMT14 *de-en* datasets in both translation directions. Note that the Transformer Encoder-Decoder is a standard model, and therefore, one of the goals of this thesis is to measure how recent models are compared to this Transformer baseline. Next, we provide a discussion on the key findings after reporting the resulting BLEU and COMET scores from our experiments in Table 4.1.

RetNet falls short of the remaining models. As expected, RetNet's performance is relatively subpar but performs sufficiently well on some language pairs. We argue its strong locality bias works well for language modeling but is not ideal in an MT setting where the source sentence can be arbitrarily long. More concretely, for this model and others that share the decoder-only formulation, we follow the `f"{source}<SEP>{target}"` MT autoregressive task setting. While this works well for Transformers that can perfectly attend to the source sentence, linear models need to properly compress the source information into their recurrent state to perform this task well enough. Now, considering RetNet's architecture and the exponential decay (γ) applied at each time step, we note how the source signal's strength tends to diminish as more tokens get decoded, making the task inherently harder.

	WMT16				WMT14			
	ro→en		en→ro		de→en		en→de	
Model	BLEU	COMET	BLEU	COMET	BLEU	COMET	BLEU	COMET
Standard models								
Transf. Enc-Dec	**29.17**	<u>74.77</u>	<u>21.96</u>	**78.62**	27.40	78.60	22.33	77.06
Transformer++	26.40	72.57	21.73	72.72	26.92	79.00	22.82	77.94
RetNet	26.35	72.35	19.85	75.99	23.36	74.69	19.57	71.73
Mamba	27.01	73.80	21.40	77.88	**27.49**	80.21	22.38	77.84
Hybrid models								
Mamba-MHA	<u>28.54</u>	**75.14**	21.69	78.34	<u>27.43</u>	**80.57**	**23.23**	**79.89**
Mamba-Local	25.85	73.85	20.59	76.84	27.23	80.10	<u>23.16</u>	<u>79.53</u>
Mamba Enc-Dec	28.51	74.39	**22.68**	77.88	27.24	79.99	21.62	78.75

Table 4.1.: MT BLEU and COMET scores for various models and language pairs. **Bold** represents top results; <u>underline</u> represents second-best.

Mamba is close to Transformers when trained from scratch. Building on the previous thoughts, Mamba's selection mechanism allows for finer control over the information retained in the hidden state. While the model is still fundamentally limited by the finite space in the hidden dimension, it is able to compress information more effectively. Overall, Mamba performs competitively with Transformer models, having close or higher COMET scores, especially when it sees more data, as in the WMT14 dataset.

Here, an intriguing case can be made for the advantages of both larger and pre-trained Mamba models. We have previously discussed (Subsection 2.3.1) how the data-dependent A matrix that is updated through backpropagation within the model, unlike the fixed γ decay observed in RetNet. Additionally, there is a significant disparity in performance between the WMT16 and WMT14 datasets, which contain 600k and 4.5 million samples, respectively. This discrepancy raises an important research question: do Mamba models narrow the performance gap with Transformers when used with larger datasets or when initialized from a language model checkpoint that possesses pre-formed language representations, enabling it to more effectively manage information? We leave exploring this idea as future work.

Integrating Attention and SSMs is helpful. Even if the performance of Mamba in the tested datasets was very positive, we further include attention layers in its overall architecture, finding the two layers to be complementary. Specifically, Mamba-MHA,

which resorts to only two attention layers, is able to outperform both Transformer variants in most datasets and language pairs. On the other hand, as we had mentioned before, using full attention breaks Mamba's asymptotic efficiency gains, which is largely undesirable. Mamba-Local retains constant inference complexity but is not as strong. Ideally, the Mamba layers would be able to compress sequence information and allow the local attention window to be as expressive as its integral counterpart, but this seems to not be the case, as we will observe in Section 4.3. Finally, Mamba Encoder-Decoder's performance is competitive, falling just short of Mamba-MHA but higher than the local variant. However, this model comes with the extra complexity of adhering to an encoder-decoder architecture.

4.2. Paragraph-level Machine Translation

The next set of experiments is designed to test how models handle larger contexts and how they extrapolate between different sequence length settings between training and testing. This idea stemmed from observing the provided WMT23 dataset training data, which is mostly sentence-level. In contrast, the test data is made of multiple sentences composing a paragraph, characterizing a data distribution mismatch. This has already been mentioned and can be seen in Table 3.1.

In order to assess this line of thought, we constructed two additional datasets besides the quality-filtered WMT23-6M; they are still composed of the same data but result from concatenating either 5 or 10 (WMT23-CAT-5 and WMT23-CAT-10) random sentences to form longer samples. Reviewing the average sequence length from these datasets, we conclude that WMT23-CAT-5 is closer to the test distribution than its 10-sentence sibling. However, the WMT23-CAT-10 dataset is particularly useful to assess whether models can still produce as high-quality translations as their short-version counterparts.

Finally, to reduce our overall experiment set, we settled on a subset of models based on the performance we saw at the sentence level. In this case, we remove RetNet and Mamba-Local from our model selection. The resulting BLEU and COMET scores are shown in Table 4.2.

BLEU scores have high variance. An initial observation reveals a notable variance in BLEU scores compared to sentence-level evaluations. This can be attributed to the increased number of tokens analyzed per sample, which naturally leads to a greater likelihood of word mismatches in longer sentences or paragraphs. Consequently, we observe significant fluctuations in BLEU scores, with differences reaching nearly 10 points between models that previously differed by only 1 or 2 points at the sentence level. In contrast, COMET scores display far greater stability across longer text sequences,

		de-en		en-de	
Dataset	Model	BLEU	COMET	BLEU	COMET
WMT23-6M	Transformer Enc-Dec	**25.25**	72.34	22.03	65.23
	Transformer++	21.61	70.70	20.12	64.80
	Mamba	16.73	69.93	15.69	63.17
	Mamba-MHA	23.91	**72.68**	**23.19**	**66.95**
	Mamba Enc-Dec	22.77	70.61	21.87	65.27
WMT23-CAT-5	Transformer Enc-Dec	**30.36**	**74.50**	29.28	69.81
	Transformer++	28.74	73.52	27.59	68.93
	Mamba	26.57	73.25	23.48	67.31
	Mamba-MHA	29.49	74.15	23.47	68.62
	Mamba Enc-Dec	30.02	73.95	**29.57**	**70.88**
WMT23-CAT-10	Transformer Enc-Dec	29.17	69.29	28.58	**69.76**
	Transformer++	29.16	72.74	27.90	68.45
	Mamba	25.44	72.24	24.52	67.61
	Mamba-MHA	27.76	74.47	25.89	69.71
	Mamba Enc-Dec	**31.61**	**75.58**	**29.95**	69.73

Table 4.2.: Paragraph-level MT BLEU and COMET scores for different WMT23-based datasets. **Bold** represents top results; underline represents second-best.

which underscores the advantage of relying on COMET for our analysis.

Mamba extrapolates poorly. Observing WMT23-6M results in isolation, it becomes evident that Mamba significantly underperforms compared to other models, as we see a gap of 2 to 3 COMET points and a notable difference in BLEU scores. Nevertheless, taking the other datasets into consideration, we note that all models have difficulties in this setting as translation quality increases substantially by training with longer samples. Notably, we see an increase of five COMET points in Mamba Enc-Dec when scaling from 1 to 10 concatenated samples.

Hybrid models are competitive. Again, it seems there is merit to the inclusion of Attention within SSMs. Based on the reported results, both models show good extrapolation ability, on par or superior to Transformers, with Mamba Enc-Dec outperforming them in terms of the overall translation quality. Moreover, in contrast to the Transformer Encoder-Decoder, Transformer++, and Mamba, the hybrid models show essentially

no apparent performance degradation when increasing the number of concatenated samples from 5 to 10.

Transformer Encoder-Decoder is still strong. Our experiments continue to support the efficacy of the base Transformer Encoder-Decoder model, particularly for smaller, specialized models developed specifically for a given task. Even when considering the difficulties associated with longer context and the absolute sinusoidal positional embeddings, it was able to outperform the Transformer++ model. This indicates that the encoder-decoder decomposition is a significant factor in translation quality.

4.3. Measuring Length Generalization

Based on the previous thoughts, we can see that the relative performance between models varies considerably when dealing with different sequence lengths. In this section, we intend to measure exactly how sequence length impacts translation quality and observe if there are meaningful trends we can distinguish between the different models.

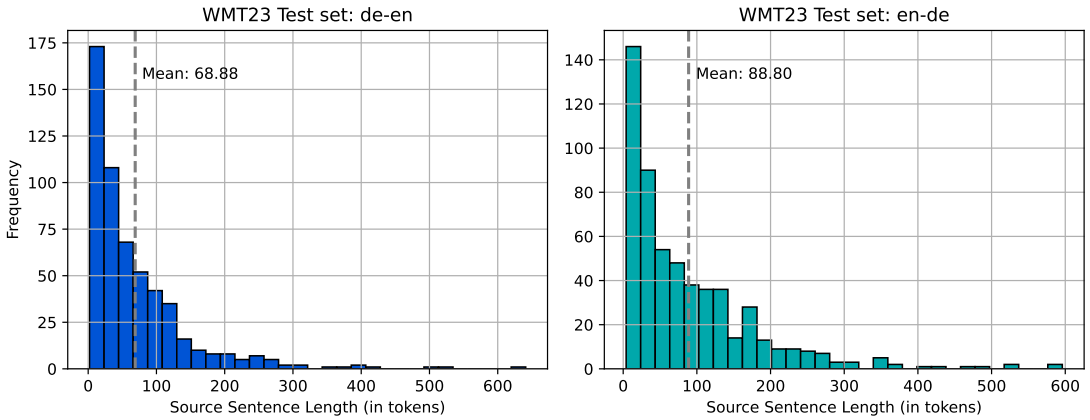


Figure 4.1.: Histogram of the source sequence lengths in WMT23 *de-en* and *en-de* test sets.

In Table 4.2, despite intuitively believing that the increase in sequence length from WMT-CAT-5 to WMT23-CAT-10 would be helpful for performance, as the model gets exposed to more sentences, there is no general corresponding increase in model quality. We plot the test distribution in Figure 4.1 and note that the data is still skewed towards

the lower spectrum despite the paragraph-level premise. Could training over a larger set of sequences degrade performance on shorter samples?

To test this hypothesis, we produce plots displaying both BLEU and COMET scores individually, per model, but distinguish between what dataset was used for training. These plots, which can be seen in Figure 4.2, intend to show how robust each model is to distribution shifts between training and test data. Next, we provide some analysis and takeaways.

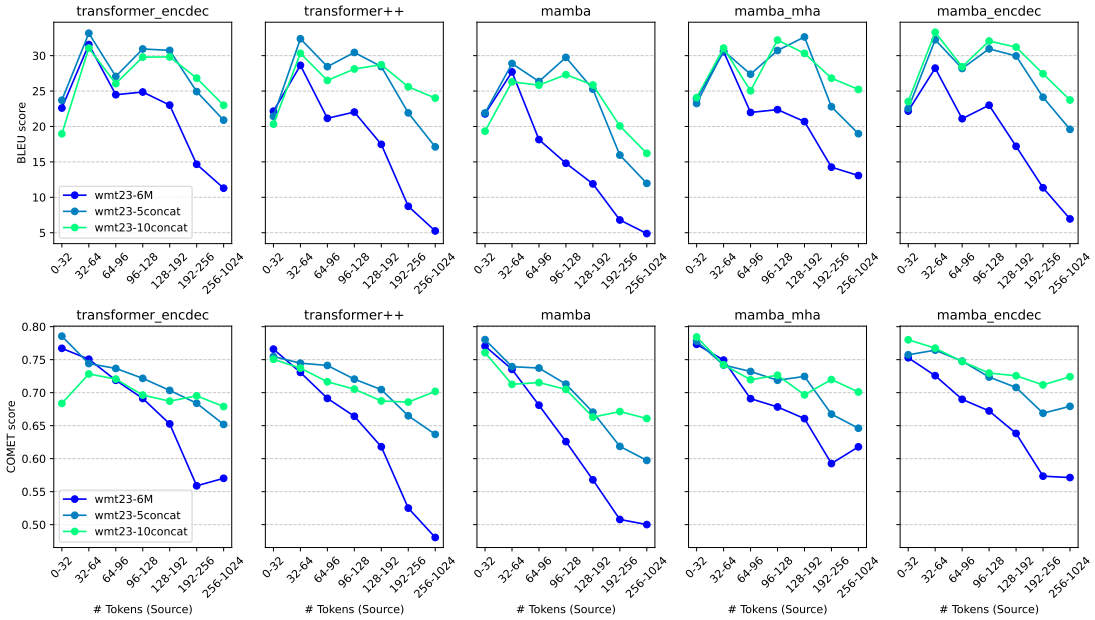


Figure 4.2.: BLEU and COMET scores across different datasets and sequence lengths. # *Tokens (Source)* denotes the number of tokens in the source sentence. WMT23 *de-en*.

Sentence length mismatch leads to poor extrapolation. We note that performance degrades with longer sequences when considering the WMT23-6M trend across all models. Interestingly, while this is true for Mamba, which can be expected given its finite hidden state, this task also reveals to be challenging for the Transformer++ model despite its theoretically good extrapolation abilities due to its relative positional embeddings.

Hybrid models robust to distribution shifts. The hybrid models display a more consistent performance between sequence lengths across different datasets, even at

the sentence level for WMT23-6M. This suggests that hybrid models might be more effective in long-context MT. In addition, we also comment on how, in Mamba-MHA, the two concatenation datasets are mostly indistinguishable up until very large sequence lengths. Meanwhile, Mamba Enc-Dec is the only model where the CAT-10 training is consistently better across sentence lengths.

WMT23-CAT-10 helps for extremely large samples. The data is coherent in showing that training on the WMT23-CAT-10 dataset provides an advantage when dealing with extremely long samples (in the 256-1024 bin). However, as a counterpart, this also makes Transformer-based models and Mamba less performant in smaller-lengthed samples. In contrast, hybrid variants are able to maintain or improve performance on smaller sentences.

Overall, we note that performing long-context translations is a particularly hard challenge for models of this scale. While we find hybrid models to maintain mostly consistent performance across different sequence lengths, we remark that our BLEU and COMET scores are below the standards set by the current generation translation systems (Alves et al. 2024), which usually leverage large models pre-trained on a vast amount of textual data. However, when comparing our models to those of similar scale from the WMT General MT shared task submissions (Kocmi et al. 2023), such as the AIRC model (Rikters et al. 2023), our results show comparable quality. Notably, there is only a minor difference—approximately 3 COMET points—between our best-performing models and this comparably trained model. This suggests that our approach to sample quality filtering and concatenation is effective, even with a significantly smaller amount of training data.

4.4. Measuring Recall Performance

This section aims to assess how well our models perform in recall tasks: the ability of a model to "recover" pieces of information from the input context. While the evaluation of associative recall is common in recent LM research (Fu et al. 2023; Arora et al. 2023; Jelassi et al. 2024), applying similar methodologies in MT presents unique challenges. In MT, it is rare for translations to exhibit repeated token patterns due to the inherent linguistic variations between languages. More often, translations involve words that are not only spelled differently but may also be entirely distinct lexically, particularly when comparing languages of different origins.

Upon manually inspecting some of the generated outputs for Mamba and RetNet, we visually notice a trend where NEs fail to be generated whenever sequences are longer,

possibly indicating information contained within the model's hidden state gets diluted in this setting. Additionally, we found that common NEs, which appear frequently in training data, are reproduced more often than low-frequency ones. We could consider frequent names like "USA" or "Warner Bros." to have a good signal representation in each model's hidden state. Would this be the case for an arbitrary name?

In MT, ensuring the accuracy of translated NEs is crucial for maintaining the integrity and usefulness of the translated text. This evaluation is particularly important for entities that must be directly transferred from the source language to the target language without modification, like a person or an organization's name. Furthermore, an interesting parallel exists with the bigram work conducted in (Arora et al. 2023). While the authors claim most of the performance difference in LM between linear models and Transformers stems from the ability to recall tokens that mostly appear together (bigrams), there is no analog situation for MT. It's uncommon for sentence-level translations to contain a token pattern multiple times. However, if we consider the decoder-only MT paradigm we are using, the source sentence is a prefix to the target, meaning that words that remain untranslated should behave similarly to the bigram case. This is precisely the case for NEs; the model has to be able to recall them in order to produce a good translation.

To systematically evaluate this phenomenon, we designed an experiment where translations are deemed accurate if a NE is correctly recalled and inaccurate otherwise. Concretely, we are only focusing on samples that contain the same NE verbatim in both the source and reference sentences. While this method simplifies the overall complexity of MT quality assessment, it provides a clear metric for our specific focus on NE recall. To complement our analysis, we track the frequency of these individual NEs across the training dataset, allowing us to correlate the average recall accuracy with how much exposure the model has to these specific token patterns.

Recall per NE frequency. This relationship is illustrated in Figure 4.3 with the IWSLT17 dataset, where NE recall accuracy is plotted against how often these particular NEs appear in training samples. Analysis of these plots reveals a pronounced correlation: high-frequency NEs tend to be recalled more effectively. This trend holds not only for linear models but also, surprisingly, for Transformer-based ones. Furthermore, there is a notable gap between recall rates for unseen NEs between RetNet and Mamba, suggesting that the selection process with the LTV property is important for recall ability and, therefore, translation quality. On the other hand, these linear models are clearly outclassed by their hybrid counterparts, which inclusively outperform the Transformer models.

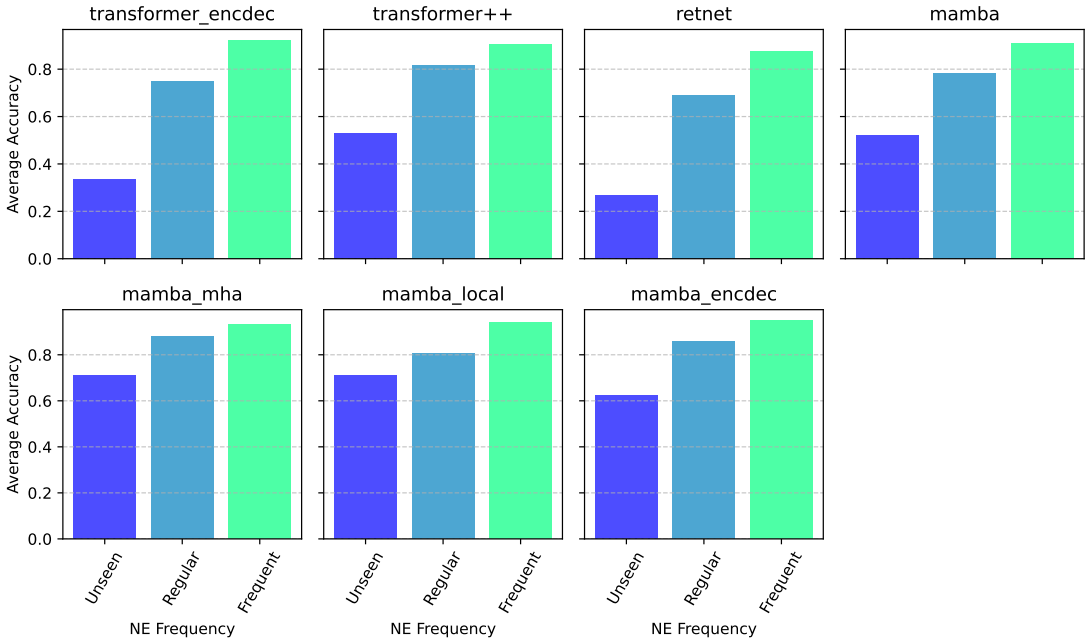


Figure 4.3.: Named Entity Recall Accuracy plotted against their frequency in the training set. *Unseen* entities do not appear in the training data, *Regular* and *Frequent* appear between $[1, 16)$ times and $[16, 1028)$ times, respectively. IWSLT17 *de-en*

.

Recall per sentence length. In Figure 4.4, we additionally assess how recall is associated with sequence length, measured by the number of tokens in each source sample. Preliminary analysis provides some takeaways: linear models are more accurate with smaller sentences – this is to be expected given their finite hidden state dimension – while the transformer-based ones tend to be more robust to this phenomenon. Considering the hybrid models as a whole, we observe good recall performance across sequence lengths. However, Mamba-Local is only performant with sequences smaller than its window size (64), indicating that signal propagation between model layers is not optimal and warrants additional architectural exploration.

Overall, these observations highlight the challenges associated with NE recall in MT and how there is still a gap between the recall abilities of linear models and those with attention, further highlighting the relevance of exploring hybrid architectures in the future.

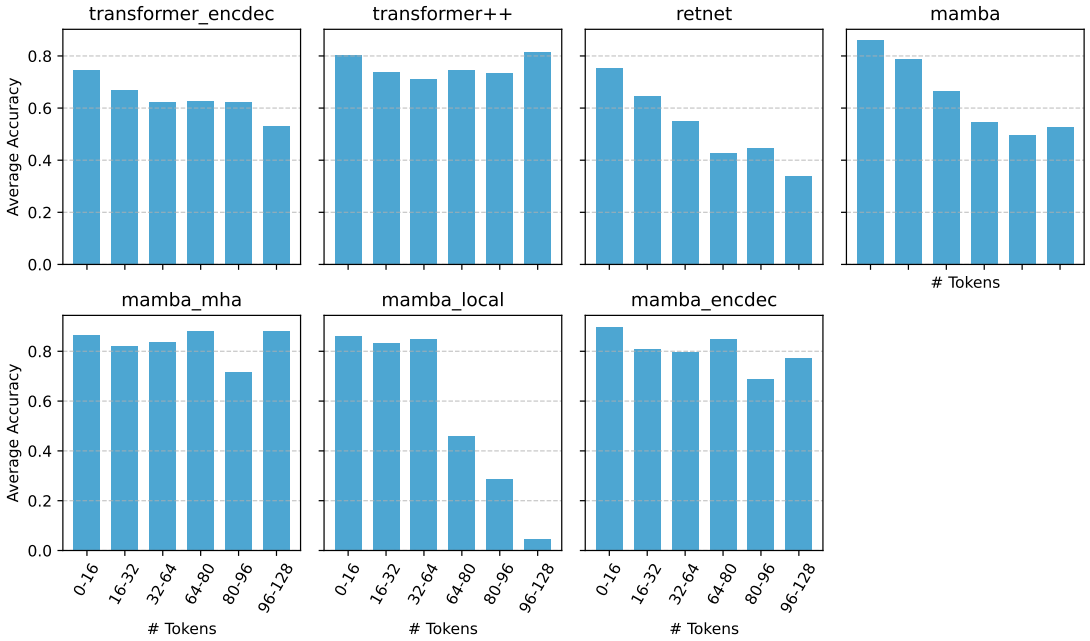


Figure 4.4.: Named Entity Recall Accuracy plotted against tokenized source sentence length. IWSLT17 de-en.

5. Conclusion

Starting with the motivation in Chapter 1, we set out to evaluate recent efficient linear models in Machine Translation tasks, particularly long-context MT. We selected a large set of models, including Transformer baselines, linear recurrent models (RetNet and Mamba), and hybrid models composed of SSMs and Attention, and conducted experiments by training small models from scratch over various parallel MT datasets.

Overall, these experiments revealed Linear Time Variance to be crucial for model performance as the Mamba architecture outclassed RetNet and was competitive with the Transformer Encoder-Decoder baseline. This was revealed to be the case both in sequence (Section 4.1) and paragraph level (Section 4.2) MT, albeit with slightly lower relative performance in larger context settings. Expanding the latter results to dissect the impact of sequence length on translation quality (Section 4.3), we find all models to have poor extrapolation abilities, although this behavior is more pronounced in Mamba. Nonetheless, by constructing alternative datasets with larger samples through concatenation approaches, we are able to alleviate this concern and produce strong (for the trained-from-scratch setting) paragraph-level MT models. On the other hand, performing all of the aforementioned experiments with hybrid models revealed them to be a promising approach for MT, as these outperformed the base Mamba model and mostly matched the standard Transformer. Finally, we leveraged the analysis in Section 4.4 to note the recurrent models' falters in recalling unseen NEs from source to target, indicating potential issues reproducing unknown token patterns within the model's hidden states, which warrants additional experiments.

In conclusion, we show the promise in the efficacy of these linear recurrent and hybrid models in MT tasks, especially when considering their efficiency benefits. Constructing a performant recurrent model that can handle longer sequences without the computational overhead associated with Transformers is paramount to unlocking promising real-world use cases, such as real-time translations or providing translations in resource-constrained environments.

5.1. Future Work

This thesis serves as an initial exploration into building efficient and high-quality MT systems. Based on our findings, several promising directions emerge for further

research, which we detail next.

Scaling Model Size and Pretraining Efforts. A natural progression for improving the performance of linear and recurrent models in MT tasks is scaling model size and extensive pretraining. Larger models have consistently shown improved capabilities in handling complex linguistic patterns and generalization across various NLP tasks (Kaplan et al. 2020). Future efforts could explore distilling these larger, pre-trained models into smaller, efficient counterparts that retain much of the original model's efficacy (Hinton et al. 2015).

Other Recurrent Architectures. Since ML research has been moving at a significant pace since the beginning of the project, new recurrent architectures have emerged as competitive to Mamba and Transformers overall. Particularly, we refer to GLA (Yang et al. 2024), HGRN-2 (Qin et al. 2024) and Griffin (De et al. 2024), which offer promising LM performance, but the same thoughts would apply to other recurrent variants that will certainly be proposed in the near future.

Effective Hybrid Models. The preliminary results from hybrid models incorporating SSMs and attention mechanisms are encouraging. Future work could investigate optimizing the interplay between these components; we noted in Section 4.4 how Mamba-Local is not performant once sequences deplete the window size, suggesting simply stacking Windowed Attention and SSM layers is not optimal. To this end, we refer to the newly proposed Infini-attention mechanism (Munkhdalai et al. 2024) as a promising research direction.

Interpretability of SSMs. Understanding how SSMs process and transform input data into coherent translations is crucial for the further improvement of these models. In this regard, we could leverage earlier interpretability work and adapt it to the SSM case, an example of this would be (Ali et al. 2024). Another aspect could be expanding the signal issues we have discussed when presenting our NE analysis and developing concrete tools to assess how tokens are encoded in the recurrent state.

Longer Datasets. Given the theoretical advantage of our models in managing long-range dependencies, testing these capabilities on datasets with significantly larger sequences is an interesting next step. This could involve creating or adapting existing datasets that offer more extensive and complex multi-paragraph texts, providing a more challenging and realistic environment for assessing model performance.

A. Padding in RetNet

Originally, RetNet was developed for LM applications, which usually involve sequences of fixed length during training. However, in MT, where sequences vary in length, effective management of padding tokens is crucial to prevent the model from processing irrelevant data. This is essential not only for maintaining data integrity but also for ensuring consistent performance metrics across various models during both training and evaluation.

RetNet's architecture, which does not use the typical softmax operation found in standard Transformers, presents unique challenges in handling padding; moreover, the model's dual formulations—recurrent and parallel—require a cohesive approach to effectively ignore padding tokens across both modes of operation. To achieve this, we leverage the decay factor present in both formulations and the recurrent nature of the model and publish our code to Github[1]. Next, we detail how padding is handled concretely in each mode of operation.

A.1. Recurrent Formulation

The first consideration is that due to the recurrent nature of the model, it is simpler to implement left padding. With the model as detailed in Equation 2.8, padding tokens can be ignored by resetting the hidden state to zero up to the first non-pad token. The main difference between this approach and starting directly at the first non-pad token is the handling of the decay factor, which still accumulates values during the padded sequence. This strategy is straightforward and aligns well with the parallel mechanism. The pseudocode for this recurrent formulation can be seen in Figure A.1.

A.2. Parallel Formulation

In contrast, the parallel formulation does not allow for direct manipulation of the hidden states as in the recurrent form. Instead, we utilize the decay mask, which is applied entry-wise to the $\mathbf{QK}^\top$ interactions. Since the decay mask is essentially a lower triangular matrix of dimensions $n \times n$, we can simulate the effect of the softmax masking

[1] `https://github.com/xtwigs/torchscale`

```
def RecurrentRetention(
    q, k, v, # bsz, num_head, len, qkv_dim
    past_kv, # bsz, num_head, qk_dim, v_dim
    decay, # num_head, 1, 1
    padding_mask # bsz, 1
):
    current_kv = decay * past_kv + k.unsqueeze(-1) * v.unsqueeze(-2)
    # zero the padding signal
    current_kv[padding_mask, ...] = 0
    output = torch.sum(q.unsqueeze(-1) * current_kv, dim=-2)
    output = group_norm(output)
    return output, current_kv
```

Figure A.1.: Pseudocode for RetNet's recurrent formulation with padding.

seen in traditional transformers by appropriately masking out columns corresponding to padding tokens. The specifics of this implementation are shown in Figure A.2.

```
def ParallelRetention(
    q, # bsz, num_head, len, qk_dim
    k, # bsz,  num_head, len, qk_dim
    v, # bsz, num_head, len, v_dim
    decay_mask # num_head, len, len
    padding_mask, # bsz, len
):
    retention = q @ k.transpose(-1, -2)
    # zero the padding signal
    decay_mask = decay_mask.masked_fill(padding_mask.unsqueeze(1).
    unsqueeze(-1), 0)
    retention = retention * decay_mask
    output = retention @ v
    output = group_norm(output)
    return output
```

Figure A.2.: Pseudocode for RetNet's parallel formulation with padding.

Abbreviations

GPU Graphics Processing Unit

MT Machine Translation

SSM Space State Models

ML Machine Learning

LM Language Modeling

NLP Natural Language Processing

LTI Linear Time Invariance

LTV Linear Time Variance

MHA Multi-Head Attention

LR Learning Rate

NE Named Entities

LLM Large Language Models

MSR Multi-Scale Retention

RNN Recurrent Neural Networks

List of Figures

List of Tables

Bibliography

Ainslie, J., J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai (2023). *GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints*. arXiv: 2305.13245 [cs.CL].

Akyürek, E., B. Wang, Y. Kim, and J. Andreas (2024). *In-Context Language Learning: Arhitectures and Algorithms*. arXiv: 2401.12973 [cs.CL].

Ali, A., I. Zimerman, and L. Wolf (2024). *The Hidden Attention of Mamba Models*. arXiv: 2403.01590 [cs.LG].

Alves, D. M., J. Pombal, N. M. Guerreiro, P. H. Martins, J. Alves, A. Farajian, B. Peters, R. Rei, P. Fernandes, S. Agrawal, P. Colombo, J. G. C. de Souza, and A. F. T. Martins (2024). *Tower: An Open Multilingual Large Language Model for Translation-Related Tasks*. arXiv: 2402.17733 [cs.CL].

Arora, S., S. Eyuboglu, A. Timalsina, I. Johnson, M. Poli, J. Zou, A. Rudra, and C. Ré (2023). *Zoology: Measuring and Improving Recall in Efficient Language Models*. arXiv: 2312.04927 [cs.CL].

Arora, S., S. Eyuboglu, M. Zhang, A. Timalsina, S. Alberti, D. Zinsley, J. Zou, A. Rudra, and C. Ré (2024). *Simple linear attention language models balance the recall-throughput tradeoff*. arXiv: 2402.18668 [cs.CL].

Ba, J. L., J. R. Kiros, and G. E. Hinton (2016). *Layer Normalization*. arXiv: 1607.06450 [stat.ML].

Basar, T. (2001). "A New Approach to Linear Filtering and Prediction Problems." In: *Control Theory: Twenty-Five Seminal Papers (1932-1981)*, pp. 167–179. DOI: 10.1109/9780470544334.ch9.

Beltagy, I., M. E. Peters, and A. Cohan (2020). *Longformer: The Long-Document Transformer*. arXiv: 2004.05150 [cs.CL].

Bhandare, A., V. Sripathi, D. Karkada, V. Menon, S. Choi, K. Datta, and V. Saletore (2019). *Efficient 8-Bit Quantization of Transformer Neural Machine Language Translation Model*. arXiv: 1906.00532 [cs.LG].

Blelloch, G. E. (Nov. 1990). *Prefix Sums and Their Applications*. Tech. rep. CMU-CS-90-190. School of Computer Science, Carnegie Mellon University.

Bojar, O., C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, L. Specia, and A. Tamchyna (June 2014). "Findings of the 2014 Workshop on Statistical Machine Translation." In: *Proceedings*

of the Ninth Workshop on Statistical Machine Translation. Ed. by O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, C. Monz, M. Post, and L. Specia. Baltimore, Maryland, USA: Association for Computational Linguistics, pp. 12–58. DOI: 10.3115/v1/W14-3302.

Bojar, O., R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. Jimeno Yepes, P. Koehn, V. Logacheva, C. Monz, M. Negri, A. Névéol, M. Neves, M. Popel, M. Post, R. Rubino, C. Scarton, L. Specia, M. Turchi, K. Verspoor, and M. Zampieri (Aug. 2016). "Findings of the 2016 Conference on Machine Translation." In: *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*. Ed. by O. Bojar, C. Buck, R. Chatterjee, C. Federmann, L. Guillou, B. Haddow, M. Huck, A. J. Yepes, A. Névéol, M. Neves, P. Pecina, M. Popel, P. Koehn, C. Monz, M. Negri, M. Post, L. Specia, K. Verspoor, J. Tiedemann, and M. Turchi. Berlin, Germany: Association for Computational Linguistics, pp. 131–198. DOI: 10.18653/v1/W16-2301.

Brooks, T., B. Peebles, C. Holmes, W. DePue, Y. Guo, L. Jing, D. Schnurr, J. Taylor, T. Luhman, E. Luhman, C. Ng, R. Wang, and A. Ramesh (2024). "Video generation models as world simulators." In.

Brown, T. B., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei (2020). *Language Models are Few-Shot Learners*. arXiv: 2005.14165 [cs.CL].

Cettolo, M., M. Federico, L. Bentivogli, J. Niehues, S. Stüker, K. Sudoh, K. Yoshino, and C. Federmann (Dec. 2017). "Overview of the IWSLT 2017 Evaluation Campaign." In: *Proceedings of the 14th International Conference on Spoken Language Translation*. Ed. by S. Sakti and M. Utiyama. Tokyo, Japan: International Workshop on Spoken Language Translation, pp. 2–14.

Child, R., S. Gray, A. Radford, and I. Sutskever (2019). *Generating Long Sequences with Sparse Transformers*. arXiv: 1904.10509 [cs.LG].

Choromanski, K., V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, D. Belanger, L. Colwell, and A. Weller (2022). *Rethinking Attention with Performers*. arXiv: 2009.14794 [cs.LG].

Chowdhery, A., S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira,

R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel (2022). *PaLM: Scaling Language Modeling with Pathways*. arXiv: 2204.02311 [cs.CL].

Correia, G. M., V. Niculae, and A. F. T. Martins (2019). *Adaptively Sparse Transformers*. arXiv: 1909.00015 [cs.CL].

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

Dao, T., D. Y. Fu, S. Ermon, A. Rudra, and C. Ré (2022). "FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness." In: *Advances in Neural Information Processing Systems*.

De, S., S. L. Smith, A. Fernando, A. Botev, G. Cristian-Muraru, A. Gu, R. Haroun, L. Berrada, Y. Chen, S. Srinivasan, G. Desjardins, A. Doucet, D. Budden, Y. W. Teh, R. Pascanu, N. D. Freitas, and C. Gulcehre (2024). *Griffin: Mixing Gated Linear Recurrences with Local Attention for Efficient Language Models*. arXiv: 2402.19427 [cs.LG].

Deepmind et al. (2024). *Gemini: A Family of Highly Capable Multimodal Models*. arXiv: 2312.11805 [cs.CL].

Dosovitskiy, A., L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby (2021). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv: 2010.11929 [cs.CV].

Freitag, M., R. Rei, N. Mathur, C.-k. Lo, C. Stewart, E. Avramidis, T. Kocmi, G. Foster, A. Lavie, and A. F. T. Martins (Dec. 2022). "Results of WMT22 Metrics Shared Task: Stop Using BLEU – Neural Metrics Are Better and More Robust." In: *Proceedings of the Seventh Conference on Machine Translation (WMT)*. Ed. by P. Koehn, L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussà, C. Federmann, M. Fishel, A. Fraser, M. Freitag, Y. Graham, R. Grundkiewicz, P. Guzman, B. Haddow, M. Huck, A. Jimeno Yepes, T. Kocmi, A. Martins, M. Morishita, C. Monz, M. Nagata, T. Nakazawa, M. Negri, A. Névéol, M. Neves, M. Popel, M. Turchi, and M. Zampieri. Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics, pp. 46–68.

Fu, D. Y., T. Dao, K. K. Saab, A. W. Thomas, A. Rudra, and C. Ré (2023). *Hungry Hungry Hippos: Towards Language Modeling with State Space Models*. arXiv: 2212.14052 [cs.LG].

Gao, Y., C. Herold, Z. Yang, and H. Ney (Nov. 2022). "Is Encoder-Decoder Redundant for Neural Machine Translation?" In: *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Ed. by Y. He, H. Ji, S. Li, Y. Liu, and C.-H. Chang. Online only: Association for Computational Linguistics, pp. 562–574.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Gu, A. and T. Dao (2023). *Mamba: Linear-Time Sequence Modeling with Selective State Spaces*. arXiv: 2312.00752 [cs.LG].

Gu, A., T. Dao, S. Ermon, A. Rudra, and C. Re (2020). *HiPPO: Recurrent Memory with Optimal Polynomial Projections*. arXiv: 2008.07669 [cs.LG].

Gu, A., K. Goel, and C. Ré (2022a). *Efficiently Modeling Long Sequences with Structured State Spaces*. arXiv: 2111.00396 [cs.LG].

Gu, A., A. Gupta, K. Goel, and C. Ré (2022b). *On the Parameterization and Initialization of Diagonal State Space Models*. arXiv: 2206.11893 [cs.LG].

Hendrycks, D., C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt (2021). *Measuring Massive Multitask Language Understanding*. arXiv: 2009.03300 [cs.CY].

Hinton, G., O. Vinyals, and J. Dean (2015). *Distilling the Knowledge in a Neural Network*. arXiv: 1503.02531 [stat.ML].

Hochreiter, S. and J. Schmidhuber (Dec. 1997). "Long Short-term Memory." In: *Neural computation* 9, pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

Hoffmann, J., S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre (2022). *Training Compute-Optimal Large Language Models*. arXiv: 2203.15556 [cs.CL].

Hua, W., Z. Dai, H. Liu, and Q. V. Le (2022). *Transformer Quality in Linear Time*. arXiv: 2202.10447 [cs.LG].

Jelassi, S., D. Brandfonbrener, S. M. Kakade, and E. Malach (2024). *Repeat After Me: Transformers are Better than State Space Models at Copying*. arXiv: 2402.01032 [cs.LG].

Jiang, A. Q., A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed (2023). *Mistral 7B*. arXiv: 2310.06825 [cs.CL].

Jiao, X., Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu (Nov. 2020). "TinyBERT: Distilling BERT for Natural Language Understanding." In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Ed. by T. Cohn, Y. He, and Y. Liu. Online: Association for Computational Linguistics, pp. 4163–4174. DOI: 10.18653/v1/2020.findings-emnlp.372.

Johnson, M., M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, M. Hughes, and J. Dean (2017). *Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation*. arXiv: 1611.04558 [cs.CL].

Kamradt, G. (2023). *Needle In A Haystack - pressure testing LLMs*. https://github.com/gkamradt/LLMTest_NeedleInAHaystack/tree/main.

Kaplan, J., S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei (2020). *Scaling Laws for Neural Language Models*. arXiv: 2001.08361 [cs.LG].

Katharopoulos, A., A. Vyas, N. Pappas, and F. Fleuret (2020). *Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention*. arXiv: 2006.16236 [cs.LG].

Kocmi, T., E. Avramidis, R. Bawden, O. Bojar, A. Dvorkovich, C. Federmann, M. Fishel, M. Freitag, T. Gowda, R. Grundkiewicz, B. Haddow, P. Koehn, B. Marie, C. Monz, M. Morishita, K. Murray, M. Nagata, T. Nakazawa, M. Popel, M. Popović, and M. Shmatova (Dec. 2023). "Findings of the 2023 Conference on Machine Translation (WMT23): LLMs Are Here but Not Quite There Yet." In: *Proceedings of the Eighth Conference on Machine Translation*. Singapore: Association for Computational Linguistics, pp. 1–42.

Kocmi, T., V. Zouhar, C. Federmann, and M. Post (2024). *Navigating the Metrics Maze: Reconciling Score Magnitudes and Accuracies*. arXiv: 2401.06760 [cs.CL].

Kondo, S., K. Hotate, M. Kaneko, and M. Komachi (2021). *Sentence Concatenation Approach to Data Augmentation for Neural Machine Translation*. arXiv: 2104.08478 [cs.CL].

Lieber, O., B. Lenz, H. Bata, G. Cohen, J. Osin, I. Dalmedigos, E. Safahi, S. Meirom, Y. Belinkov, S. Shalev-Shwartz, O. Abend, R. Alon, T. Asida, A. Bergman, R. Glozman, M. Gokhman, A. Manevich, N. Ratner, N. Rozen, E. Shwartz, M. Zusman, and Y. Shoham (2024). *Jamba: A Hybrid Transformer-Mamba Language Model*. arXiv: 2403.19887 [cs.CL].

Liu, H., M. Zaharia, and P. Abbeel (2023). *Ring Attention with Blockwise Transformers for Near-Infinite Context*. arXiv: 2310.01889 [cs.CL].

Longpre, S., R. Mahari, A. Chen, N. Obeng-Marnu, D. Sileo, W. Brannon, N. Muennighoff, N. Khazam, J. Kabbara, K. Perisetla, X. Wu, E. Shippole, K. Bollacker, T. Wu, L. Villa, S. Pentland, and S. Hooker (2023). *"The Data Provenance Initiative: A Large Scale Audit of Dataset Licensing & Attribution in AI"*. arXiv: 2310.16787 [cs.CL].

Martin, E. and C. Cundy (2018). *Parallelizing Linear Recurrent Neural Nets Over Sequence Length*. arXiv: 1709.04057 [cs.NE].

Martins, P. H., Z. Marinho, and A. F. T. Martins (2022). *∞-former: Infinite Memory Transformer*. arXiv: 2109.00301 [cs.CL].

Maslej, N., L. Fattorini, E. Brynjolfsson, J. Etchemendy, K. Ligett, T. Lyons, J. Manyika, H. Ngo, J. C. Niebles, V. Parli, Y. Shoham, R. Wald, J. Clark, and R. Perrault (2023). *Artificial Intelligence Index Report 2023*. arXiv: 2310.03715 [cs.AI].

Mohtashami, A. and M. Jaggi (2023). *Landmark Attention: Random-Access Infinite Context Length for Transformers*. arXiv: 2305.16300 [cs.CL].

Munkhdalai, T., M. Faruqui, and S. Gopal (2024). *Leave No Context Behind: Efficient Infinite Context Transformers with Infini-attention*. arXiv: 2404.07143 [cs.CL].

OpenAI et al. (2024). *GPT-4 Technical Report*. arXiv: 2303.08774 [cs.CL].

Orvieto, A., S. L. Smith, A. Gu, A. Fernando, C. Gulcehre, R. Pascanu, and S. De (2023). *Resurrecting Recurrent Neural Networks for Long Sequences*. arXiv: 2303.06349 [cs.LG].

Peters, B. and A. F. T. Martins (2021). *Smoothing and Shrinking the Sparse Seq2Seq Search Space*. arXiv: 2103.10291 [cs.CL].

Poli, M., J. Wang, S. Massaroli, J. Quesnelle, R. Carlow, E. Nguyen, and A. Thomas (Dec. 2023). *StripedHyena: Moving Beyond Transformers with Hybrid Signal Processing Models*. DOI: 10.57967/hf/1595.

Post, M. (Oct. 2018). "A Call for Clarity in Reporting BLEU Scores." In: *Proceedings of the Third Conference on Machine Translation: Research Papers*. Ed. by O. Bojar, R. Chatterjee, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, C. Monz, M. Negri, A. Névéol, M. Neves, M. Post, L. Specia, M. Turchi, and K. Verspoor. Brussels, Belgium: Association for Computational Linguistics, pp. 186–191. DOI: 10.18653/v1/W18-6319.

Qi, Y., D. Sachan, M. Felix, S. Padmanabhan, and G. Neubig (June 2018). "When and Why Are Pre-Trained Word Embeddings Useful for Neural Machine Translation?" In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Ed. by M. Walker, H. Ji, and A. Stent. New Orleans, Louisiana: Association for Computational Linguistics, pp. 529–535. DOI: 10.18653/v1/N18-2084.

Qin, Z., W. Sun, H. Deng, D. Li, Y. Wei, B. Lv, J. Yan, L. Kong, and Y. Zhong (2022). *cosFormer: Rethinking Softmax in Attention*. arXiv: 2202.08791 [cs.CL].

Qin, Z., S. Yang, W. Sun, X. Shen, D. Li, W. Sun, and Y. Zhong (2024). *HGRN2: Gated Linear RNNs with State Expansion*. arXiv: 2404.07904 [cs.CL].

Radford, A., J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever (2022). *Robust Speech Recognition via Large-Scale Weak Supervision*. arXiv: 2212.04356 [eess.AS].

Raffel, C., N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu (2023). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. arXiv: 1910.10683 [cs.LG].

Rei, R., J. G. C. de Souza, D. Alves, C. Zerva, A. C. Farinha, T. Glushkova, A. Lavie, L. Coheur, and A. F. T. Martins (Dec. 2022a). "COMET-22: Unbabel-IST 2022 Submission for the Metrics Shared Task." In: *Proceedings of the Seventh Conference on Machine Translation (WMT)*. Ed. by P. Koehn, L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussà, C. Federmann, M. Fishel, A. Fraser, M. Freitag, Y. Graham, R. Grundkiewicz, P. Guzman, B. Haddow, M. Huck, A. Jimeno Yepes, T. Kocmi, A. Martins, M. Morishita, C. Monz, M. Nagata, T. Nakazawa, M. Negri, A. Névéol, M.

Neves, M. Popel, M. Turchi, and M. Zampieri. Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics, pp. 578–585.

Rei, R., M. Treviso, N. M. Guerreiro, C. Zerva, A. C. Farinha, C. Maroti, J. G. C. de Souza, T. Glushkova, D. Alves, L. Coheur, A. Lavie, and A. F. T. Martins (Dec. 2022b). "CometKiwi: IST-Unbabel 2022 Submission for the Quality Estimation Shared Task." In: *Proceedings of the Seventh Conference on Machine Translation (WMT)*. Ed. by P. Koehn, L. Barrault, O. Bojar, F. Bougares, R. Chatterjee, M. R. Costa-jussà, C. Federmann, M. Fishel, A. Fraser, M. Freitag, Y. Graham, R. Grundkiewicz, P. Guzman, B. Haddow, M. Huck, A. Jimeno Yepes, T. Kocmi, A. Martins, M. Morishita, C. Monz, M. Nagata, T. Nakazawa, M. Negri, A. Névéol, M. Neves, M. Popel, M. Turchi, and M. Zampieri. Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics, pp. 634–645.

Rein, D., B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman (2023). *"GPQA: A Graduate-Level Google-Proof Q&A Benchmark"*. arXiv: 2311.12022 [cs.AI].

Rikters, M. and M. Miwa (Dec. 2023). "AIST AIRC Submissions to the WMT23 Shared Task." In: *Proceedings of the Eighth Conference on Machine Translation*. Ed. by P. Koehn, B. Haddow, T. Kocmi, and C. Monz. Singapore: Association for Computational Linguistics, pp. 155–161. DOI: 10.18653/v1/2023.wmt-1.13.

Sennrich, R., B. Haddow, and A. Birch (Aug. 2016). "Neural Machine Translation of Rare Words with Subword Units." In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by K. Erk and N. A. Smith. Berlin, Germany: Association for Computational Linguistics, pp. 1715–1725. DOI: 10.18653/v1/P16-1162.

Shazeer, N. (2020). *GLU Variants Improve Transformer*. arXiv: 2002.05202 [cs.LG].

Singh, S., F. Vargus, D. Dsouza, B. F. Karlsson, A. Mahendiran, W.-Y. Ko, H. Shandilya, J. Patel, D. Mataciunas, L. OMahony, M. Zhang, R. Hettiarachchi, J. Wilson, M. Machado, L. S. Moura, D. Krzemiński, H. Fadaei, I. Ergün, I. Okoh, A. Alaagib, O. Mudannayake, Z. Alyafeai, V. M. Chien, S. Ruder, S. Guthikonda, E. A. Alghamdi, S. Gehrmann, N. Muennighoff, M. Bartolo, J. Kreutzer, A. Üstün, M. Fadaee, and S. Hooker (2024). *Aya Dataset: An Open-Access Collection for Multilingual Instruction Tuning*. arXiv: 2402.06619 [cs.CL].

Smith, J. T. H., A. Warrington, and S. W. Linderman (2023). *Simplified State Space Layers for Sequence Modeling*. arXiv: 2208.04933 [cs.LG].

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." In: *Journal of Machine Learning Research* 15.56, pp. 1929–1958.

Su, J., Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu (2023). *RoFormer: Enhanced Transformer with Rotary Position Embedding*. arXiv: 2104.09864 [cs.CL].

Sun, Y., L. Dong, S. Huang, S. Ma, Y. Xia, J. Xue, J. Wang, and F. Wei (2023). *Retentive Network: A Successor to Transformer for Large Language Models*. arXiv: 2307.08621 [cs.CL].

Sun, Y., L. Dong, B. Patra, S. Ma, S. Huang, A. Benhaim, V. Chaudhary, X. Song, and F. Wei (2022). *A Length-Extrapolatable Transformer*. arXiv: 2212.10554 [cs.CL].

Tay, Y., M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler (2020). *Long Range Arena: A Benchmark for Efficient Transformers*. arXiv: 2011.04006 [cs.LG].

Tay, Y., M. Dehghani, D. Bahri, and D. Metzler (2022). *Efficient Transformers: A Survey*. arXiv: 2009.06732 [cs.LG].

Touvron, H., T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample (2023a). *LLaMA: Open and Efficient Foundation Language Models*. arXiv: 2302.13971 [cs.CL].

Touvron, H., L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom (2023b). *Llama 2: Open Foundation and Fine-Tuned Chat Models*. arXiv: 2307.09288 [cs.CL].

Treviso, M., J.-U. Lee, T. Ji, B. van Aken, Q. Cao, M. R. Ciosici, M. Hassid, K. Heafield, S. Hooker, C. Raffel, P. H. Martins, A. F. T. Martins, J. Z. Forde, P. Milder, E. Simpson, N. Slonim, J. Dodge, E. Strubell, N. Balasubramanian, L. Derczynski, I. Gurevych, and R. Schwartz (2023). *Efficient Methods for Natural Language Processing: A Survey*. arXiv: 2209.00099 [cs.CL].

Tsai, Y.-H. H., S. Bai, M. Yamada, L.-P. Morency, and R. Salakhutdinov (2019). *Transformer Dissection: A Unified Understanding of Transformer's Attention via the Lens of Kernel*. arXiv: 1908.11775 [cs.LG].

Tustin, A. (May 1947). "A method of analysing the behaviour of linear systems in terms of time series." English. In: *Journal of the Institution of Electrical Engineers - Part IIA: Automatic Regulators and Servo Mechanisms* 94 (1), 130–142(12). ISSN: 2050-5523.

Vardasbi, A., T. P. Pires, R. M. Schmidt, and S. Peitz (2023). *State Spaces Aren't Enough: Machine Translation Needs Attention*. arXiv: 2304.12776 [cs.CL].

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017). "Attention is All you Need." In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc.

Wang, S., Z. Tu, Z. Tan, W. Wang, M. Sun, and Y. Liu (2021). *Language Models are Good Translators*. arXiv: 2106.13627 [cs.CL].

Wei, J., M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le (2022). "Finetuned Language Models are Zero-Shot Learners." In: *International Conference on Learning Representations*.

Xu, H., Y. J. Kim, A. Sharaf, and H. H. Awadalla (2023). *A Paradigm Shift in Machine Translation: Boosting Translation Performance of Large Language Models*. arXiv: 2309.11674 [cs.CL].

Yang, S., B. Wang, Y. Shen, R. Panda, and Y. Kim (2024). *Gated Linear Attention Transformers with Hardware-Efficient Training*. arXiv: 2312.06635 [cs.LG].

Zhang, B. and R. Sennrich (2019). *Root Mean Square Layer Normalization*. arXiv: 1910.07467 [cs.LG].

Zhang, M., K. Bhatia, H. Kumbong, and C. Ré (2024). *The Hedgehog & the Porcupine: Expressive Linear Attentions with Softmax Mimicry*. arXiv: 2402.04347 [cs.LG].