AN ABSTRACT OF THE THESIS OF

Victor Agostinelli for the degree of Master of Science in Electrical and Computer Engineering presented on April 14, 2023.

Title: A Case for Application-Based Linear Transformers: Applied Towards Text-to-Spectrogram

Abstract approved: _____

Lizhong Chen

Various natural language processing (NLP) tasks necessitate deep models that are fast, efficient, and small based on their ultimate application at the edge or elsewhere. While significant investigation has furthered the efficiency and reduced the size of these models, reducing their downstream latency without significant trade-offs remains a difficult task. This thesis proposes the modular utilization of state-of-the-art attention linearization techniques that results in fully linearized run-time with respect to the size of the sample at inference for autoregressive decoding tasks for neural machine translation (NMT), text-to-spectrogram (TTS) tasks, and other applications while minimizing the associated costs to downstream model performance.

A Case for Application-Based Linear Transformers: Applied Towards Text-to-Spectrogram

by

Victor Agostinelli

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented April 14, 2023
Commencement June 2023

Master of Science thesis of Victor Agostinelli presented on April 14, 2023.

APPROVED:

Major Professor, representing Electrical and Computer Engineering

---

Head of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Victor Agostinelli, Author

ACKNOWLEDGEMENTS

I would like to acknowledge Lizhong Chen's continued guidance and contributions, both intellectual and professional. He has served as a constant guide as I transitioned into new and exciting research, providing intuition where necessary and orienting me towards novel opportunities. I would also like to thank Drew Penney, Matthew Raffel, and Asif Kazi Fuad for enabling very early versions of this work and serving as resources for various NLP and AI related questions and ideas.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

LIST OF TABLES (Continued)

LIST OF TABLES (Continued)

LIST OF TABLES (Continued)

LIST OF ALGORITHMS

LIST OF APPENDIX FIGURES

LIST OF APPENDIX FIGURES (Continued)

LIST OF APPENDIX FIGURES (Continued)

LIST OF APPENDIX FIGURES (Continued)

Chapter 1: Introduction

1.1 Transformers and Their Impact

The transformer class of models has provided natural language processing (NLP) researchers and industry leaders with new ways to take advantage of relationships within sequential data, primarily through the delegation of tasks between encoder and decoder stacks along with an, admittedly not new, attention mechanism and significant parallelization potential versus competitive RNNs. Their effectiveness was first demonstrated by Vaswani et. al [57] and later expanded upon by a number of others for many tasks outside of the scope of the original paper, including various computer vision [15, 39], audio processing [18, 58], and biological science applications (i.e. protein structure modeling) [34].

In regards to NLP applications, the attention mechanisms within the original transformer architecture were effective for full sentence neural machine translation (NMT), but by virtue of their complete nature across the entire length of the sample and the softmax normalization applied to them, these attention calculations are costly. Speedy and efficient alternatives are desirable, but often come with significant penalties as far as accuracy is concerned. For some alternatives, this can be overcome by a specification of environment conditions and assumptions about the conditions of the attention calculation, but these solutions, such as kernel-

based [56], hashing-based [24], or sparsity-based approaches [3, 9, 65], generally lack widespread applicability by their very nature.

1.2 Attention Linearization and Autoregressive Applications

The state-of-the-art within the aforementioned space of maximimization of transformer efficiency and speed is cosFormer [45], which proposes an attention scheme that replaces softmax functionality with a ReLU based activiation, provides additional normalization that performs a similar function to the softmax normalization, and provides a linear-time, decomposable, and cosine-based re-weighting mechanism such that the attention calculation can be reordered into linear run-time. They applied their scheme to a number of sequence to sequence tasks and achieved competitive downstream results.

Unfortunately, for autoregressive tasks, their approach and implementation are insufficient. cosFormer makes critical assumptions about the state of their inference environment that renders their scheme difficult to apply without some adaptations. Moreover, they fail to take advantage of a critical data reuse opportunity introduced by Katharopoulos et. al [21] for autoregressive applications that actually renders an initial application of their scheme quadratic with respect to the number of samples. Finally, their general approach to the problem of linearizing attention mechanisms is likely faulty, as they seek a one-size-fits-all linearization scheme.

In this thesis, it is demonstrated that with aggressive data reuse mechanisms, the above run-time can be reduced to fully linear run-time with respect to the

length of each sample in autoregressive tasks, which include the very popular simultaneous translation task as well as autoregressive speech synthesis (S^2), specifically the text to spectrogram (TTS) sub-problem. Moreover, this thesis highlights under what practical circumstances latency reductions are likely achievable and provide inference results for commonly used datasets for easy comparison to contemporary schemes.

Chapter 2: Literature Review

2.1 Fundamentals of Transformers and Attention

The core architecture of Vaswani et. al's [57] original transformer has been studied exhaustively, but this thesis will review it briefly and focus on the attention-based elements. At a high level, a transformer is composed of an encoder stack that encodes the input into a hidden embedding space and a decoder stack that takes that encoded embedding space alongside outputs at previous time steps (most relevant for autoregressive applications) and decodes it into the target data, most commonly in the form of output probabilities for some visible embedding space.

Each encoder layer within the encoder stack is primarily composed of a self-attention block with a residual connection and a normalization block for the attention output matrix alongside a fully connected feed-forward network with a similar residual connection and normalization block. Decoder layers are very similar in structure, but host, interposed between the self-attention and feed-forward blocks, an encoder to decoder cross-attention block that assists in generating relationships between the hidden embedding output of the encoder stack and the previous outputs of the decoder alongside the same residual connection and normalization block exhibited in encoder and decoder self-attention.

Every attention block takes in a query matrix Q in $\mathbb{R}^{N_1 \times d_k}$, a key matrix K

in $\mathbb{R}^{N_2 \times d_k}$, and a value matrix V in $\mathbb{R}^{N_2 \times d_v}$. Classically, a softmax operator is employed on the product of the query and key to further distance tokens that are less relevant to one another and normalize the output into what can be intuited as probabilities. These attention blocks, for both encoder and decoder layers, are multi-headed for some heads H, as opposed to scaled dot product attention that works across the entire embedding space all at once, and the original embedding space is reformed by concatenating the outputs of each head together. Each head projects the query, key, and value matrices into this new sub-space via weight matrices W_h^q in $\mathbb{R}^{d_k \times d_{kh}}$, W_h^k in $\mathbb{R}^{d_k \times d_{kh}}$, and W_h^v in $\mathbb{R}^{d_v \times d_{vh}}$ for some head h in H. As a quick note, the scaling factor d_k or d_{kh} is a heuristic applied primarily to ensure that the emphasized relationships produced by the softmax operator are not so large as to introduce performance degradation. An output projection layer is applied at the end of the concatenation of various attention head outputs to ensure that the attention mechanism output maps back to the dimensionality of the model as a whole as opposed to d_v and that projection layer is characterized by W_O in $\mathbb{R}^{Hd_v \times d_{model}}$ General formulations for the attention calculations of a classical transformer are supplied below. A depiction of classical transformers is provided in Figure 2.1.

$$A_{sdpa} = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

$$a_h = softmax(\frac{QW_h^q K^T W_h^k}{\sqrt{d_{kh}}})W_h^v V \tag{2.2}$$

$$A_{mha} = concat(a_1, a_2, \ldots, a_H)W_O \tag{2.3}$$

2.2 Early Efficient Transformers

Almost since its inception, various groups have worked to better the efficiency of transformers. Obvious solutions were quickly applied, mostly in the form of reduced precision [36], gradient-checkpointing [8], and similar methods [52]. Some useful heuristics exist that involve pruning, most notably attention head pruning to varying degrees [40, 59], to reduce the computational complexity of these models, but these heuristics are only really applicable to models that are significantly over-parametrized, in which case reducing the number of attention heads or the size of the embedding space of the model during training would probably be an adequate solution that would also significantly reduce the cost of training. Sparsity analyses (especially in hardware [9, 49, 55]) function similarly, avoiding stretches of computation according to observed sparsity patterns or gating computational units upon encountering null values. While sparsity is a significant issue for many downstream tasks that over-parameterized transformers are often applied to, the applicability of these methodologies is necessarily task specific.

For models that are not necessarily over-parameterized, alternatives become desirable that focus on more generalizable computation reduction. A number of options emerged during the early development of transformer architecture, including Liu et. al's Memory-Compressed Attention mechanism [30] and Parmer et. al's

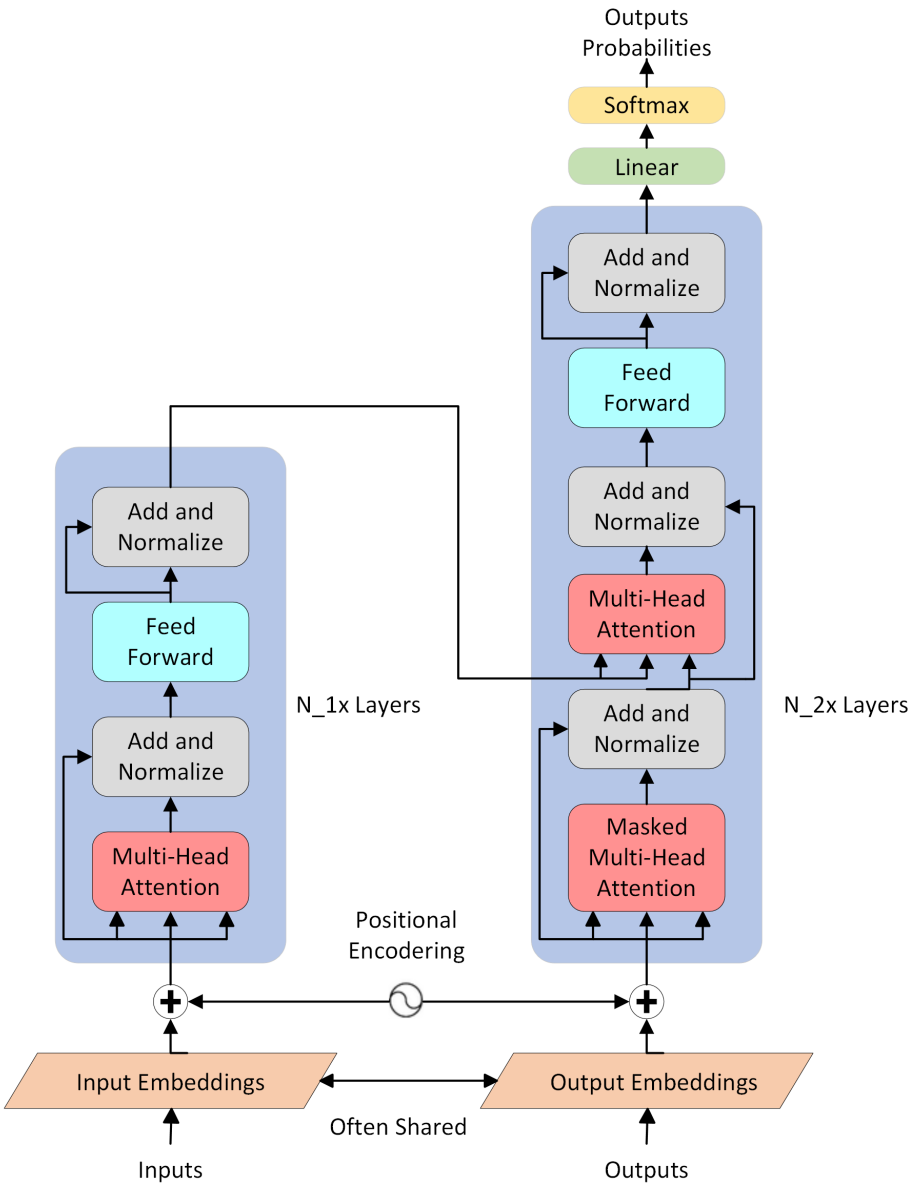

Figure 2.1: Depiction of classical transformer architecture as constructed by Vaswani et. al [57]. Input and output embeddings are commonly shared, as shown, to reduce model size and it is typical to use similar, if not identical, positional encoding schemes to produce recurrence. While not shown here, it has become somewhat popular to rearrange the order of the normalization blocks [64], placing them before the attention and feed-forward blocks to slightly speed up and stabilize training with little to no cost to end performance.

Image Transformer [39], both of which engaged with or were inspired by convolutional elements in varying fashions to reduce their computational complexity to manageable levels for long sequence lengths, often using sliding window mechanics that heavily resemble convolution. These, alongside some others [13, 27, 63] that also largely focused on these sliding window-like methodologies to reduce computation to manageable levels, introduced novel ideas concerning efficient computation that would be built upon by later work. In a very similar vein, Sukhbaatar et. al [54] proposed a learnable span for every query position, adding some adaptability to the more rigid previous solutions.

Child et. al [9] introduced one of the most popular and foundational efficient transformers for embedding spaces that are particularly sparse and/or patterned consistently, and was followed up by a number of similar schemes including Longformer [3], Big Bird [65], and Extended Transformer Construction [1]. Ultimately, while they still explored ideas relevant to CNNs and employed mechanisms somewhat resembling convolution at their core, their methodology elaborated upon the general concepts introduced by earlier works and focused on factorizing the general form of an attention mechanism via strided or fixed implementations, where strided attention resembles the aforementioned efficient attention schemes and fixed factorized attention allowed for some limited attention for previous positions in autoregressive applications via specified stride and a high-level parameter specifying the span of the blocks that can be observed with this limited look-back mechanism, more closely approximating softmax behavior for applications that do not exhibit extreme locality. Generally, such schemes that depend on fine-tuned pat-

tern recognition and extreme reliance on locality to function can struggle when dealing with applications that require long-range dependencies, and while fixed factorized attention is efficient and can roughly approximate softmax behavior, more generalizable alternatives are desirable.

Linformer [61] operates similarly, observing that self-attention resulting matrices are often low-rank, and, as such, can be approximated via linear projections or similar methods being applied to the key and value matrices to reduce their size preemptively from $n \times d_k$ and $n \times d_v$ to some $n \times k$ space, resulting in a significant run-time reduces when k is much smaller than the size of the sample. These learned approximations proved themselves very accurate when k remained somewhat large, but tended to decay quickly as k shrank, rendering it most useful when operating with extremely large samples so as to avoid a loss of expressivity.

In contrast, Kitaev et. al [24] propose Reformer, a more generalizable form of efficient transformer with an attention mechanism that focuses on location-sensitive hashing (LSH) to reduce computation to a manageable level. They employ a variety of techniques to enable their LSH solution, and achieve competitive results with Vaswani et. al [57] on English to German neural machine translation (NMT) on the WMT 2014 dataset. Their hashing scheme ultimately provides a reasonable performance baseline that later efficient transformer implementations would consistently work to beat and a number of other schemes build directly upon their work [14, 47].

2.3 Linear Transformers

Katharopoulos et. al [21] introduce the first of what this thesis considers state-of-the-art implementations for efficient attention calculations, allowing for a true reduction to $O(n)$ complexity with respect to the number of tokens in a given sample by reordering the attention mechanism demonstrated by Equation 2.1 such that the key and value matrices can be multiplied together, first, before applying the query to that product. This is not normally possible due to the softmax operator removing any associability for the query and key matrices. In response, Katharapoulos et. al choose to remove it and attempt to roughly model its behavior with a non-linear activation function that they denote as some similarity function S that is distributable, meaning that $S(Q, K^T) = S_q(Q)S_k(K^T)$. This reordering is demonstrated in a row-wise manner for the output matrix of the attention mechanism via Equations 2.4, 2.5, and 2.6 where A_i is equivalent to one output row i in N_1 for classical softmax attention and $\tilde{A}_i$ is equivalent to one output row i in N_1 for reordered and linear attention. The effects of this reordering are further showcased in Figure 2.2.

$$A_i = \sum_j \frac{exp(Q_i K_j^T)}{\sum_j exp(Q_i K_j^T)} V_j \tag{2.4}$$

$$\tilde{A}_i = \sum_j \frac{S(Q_i K_j^T)}{\sum_j S(Q_i K_j^T)} V_j = \sum_j \frac{S_q(Q_i) S_k(K_j^T)}{\sum_j S_q(Q_i) S_k(K_j^T)} V_j \tag{2.5}$$

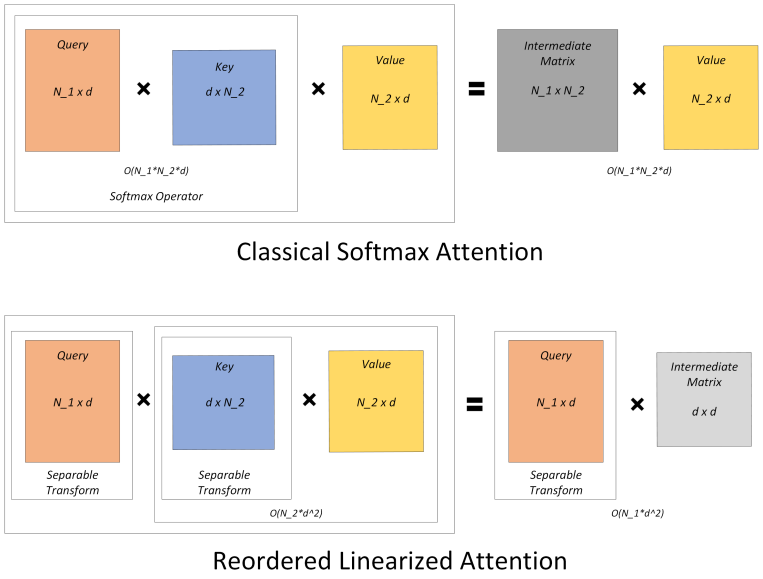

Figure 2.2: Illustration of attention calculation reordering and linearization for one attention head. This is useful when both N_1 and N_2 are significantly larger than d, which occurs for many applications or for models with many attention heads with smaller samples. When this condition is met, run-time is linearized from $O(N_1 N_2)$ to $O(N_1) + O(N_2)$ for an arbitrary decoding time-step, assuming that neither size is significantly greater than the other.

$$\tilde{A}_i = \sum_j \frac{S_q(Q_i)(S_k(K_j^T)V_j)}{S_q(Q_i)\sum_j S_k(K_j^T)} \tag{2.6}$$

Critically, Katharapolous et. al noted that their implementation could somewhat approach softmax attention accuracy for some tasks, beat out the performance of Reformer [24], and, during inference for bi-directional and autoregressive tasks, they could achieve practical run-times that were orders of magnitude faster than classical attention for very long sequences. This is accomplished for autoregressive tasks, in large part, due to a data-reuse opportunity present in the

intermediate matrices of the attention calculation, which is defined very generally by Algorithm 2 in the Appendix. By changing the intermediate matrix in the numerator of Equation 2.5 from $S_q(Q)S_k(K^T)$ in $\mathbb{R}^{N_1 \times N_2}$ to $S_k(K^T)V$ in $\mathbb{R}^{d_k \times d_v}$, any reliance on the query is removed for some decoding time-step m in time-steps 1 to N_1 (as a note, classically the query sample length is set to 1 during autoregressive inference, as no previous query tokens can attend to future information, in theory this could be applied for an arbitrary length query where that requirement is not so strict but the output is still incremental). Given that, one could store and update the $S_k(K^T)V$ intermediate matrix for every time-step where new information is added, resulting in a run-time reduction from $O(mN_2)$ for some arbitrary decoding time-step m in time-steps 1 to N_1 (during self-attention N_2 would also be equal to m) to $O(m)$. For typical autoregressive applications, this would result in constant run-time at every decoding step as opposed to a run-time with dependence on N_2, which corresponds, of course, to linear run-time with respect to the number of tokens across the entire autoregressive prediction. Note that the dimensionality of attention heads where this occurs are omitted in this run-time notation, as it is typical for N_1 and N_2 to be much larger than some general embedding dimension d (this is technically application-specific and model-specific). This data-reuse opportunity and its benefits are further underscored in Figure 2.3.

While Katharapolous et. al's work was remarkable for its time and sparked the development of a number of linearized attention mechanisms or their application to new tasks [10, 29, 45], it proved only foundational, as some fundamental problems existed with their approach. While their formulation of a hypothetical linear

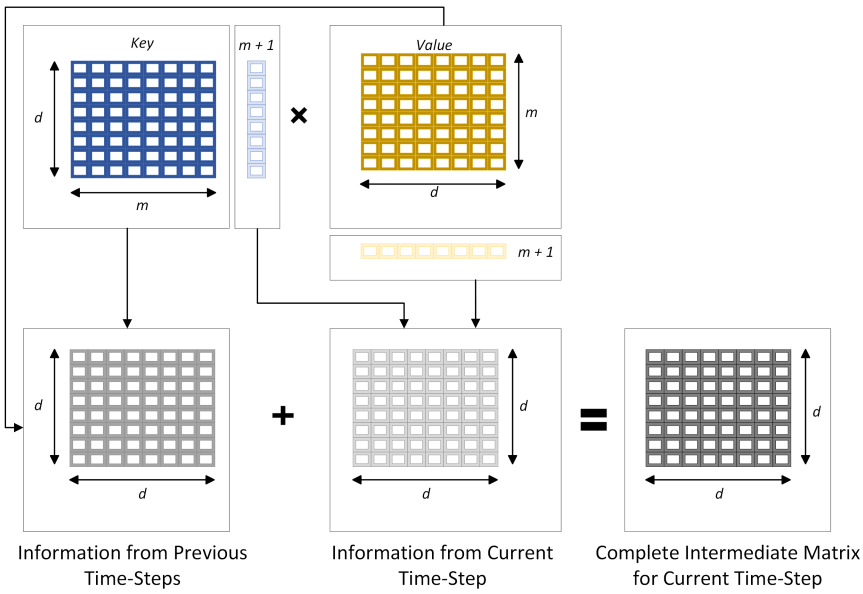

Figure 2.3: Illustration of one of the data-reuse opportunities present under reordered and linearized attention for the $K^T V$ intermediate matrix for one attention head. Past time-steps are represented from decoding time-step 1 to m, with the current decoding time-step being represented as $m + 1$.

transformer via reordering and the replacement of softmax with some general, and distributable, similarity function is still commonly used, their selected similarity function seemed somewhat arbitrary. $S(M) = elu(M) + 1$ was settled on as a similarity function, which does not approximate even the basics of softmax behavior particularly well.

In response, Choromanski et. al [10] proposed Performer, an extremely efficient transformer that makes use of the reordering mechanism proposed by Katharapolous et. al [21] in addition to several techniques to better approximate softmax-based attention without any particular prior assumptions about sparsity or low-rank representations that some other schemes rely on [9, 61]. Instead of using an exponential linear unit for their similarity function, they choose to use positive and orthogonal random feature maps as a kernel function to approximate softmax attention in a lower dimensional space, although they note that their methodology can be used to further speed up any other similarity function (ReLU, elu, trigonometry based similarity functions, etc.). Performer remains a somewhat popular paradigm and has inspired a number of capable, but similar, schemes [2, 41].

2.4 cosFormer and Modern Linearized Attention

Similar to its prominent predecessors, the Linear Transformer [21] and Performer [10], cosFormer, proposed by Zhen et. al [45], takes advantage of the reordering opportunities present in linearized attention when the softmax operator is replaced with an approximate similarity function and propose an additional step that pre-

serves the ability of an attention mechanism to attend across the entire sample, unlike sparsity [3, 9, 65] or CNN-inspired [30, 39] mechanisms. In addition to replacing softmax with an alternative similarity function (in their case, they choose ReLU as it maintains the non-negativity of softmax), cosFormer employs a re-weighting mechanism to better approximate softmax behavior, noting that under many circumstances and for many NLP tasks, significant locality bias is observed [12, 25] that should render tight approximation of the softmax operator possible in linear time.

In seeking out a re-weighting function that is suitable for their needs, Zhen et. al focus on producing a non-linear operator that can be applied to the query and key matrices in linear time so as to better approximate softmax non-linearity. They note that softmax, classically, emphasizes relationships in the QK^T matrix (i.e. small scores are significantly reduced, larger scores are emphasized heavily) and, as such, they land upon the re-weighting mechanism described in Equation 2.7 where N is the length of the query matrix and M is the length of the key and value matrices, utilizing a cosine-based and location sensitive function to modulate the aforementioned scores such that tokens that are close to one another are encouraged to attend strongly whereas tokens that are far away are discouraged from doing so. A critical property of this re-weighting mechanism is that it is also decomposable, meaning that it can be broken down such that all elements relating to the row of the query can be applied element-wise and all elements relating to the column of the transpose key matrix can be applied element-wise. This is demonstrated via Equation 2.8 and 2.9, where Ptolemy's Theorem is ap-

plied to decompose the proposed cosine-based re-weighting mechanism that is then distributed to the linearly transformed query and key matrices.

$$S(Q_i, K_j^T) = S_q(Q_i)S_k(K_j^T)cos(\frac{\pi}{2}(\frac{i}{N} - \frac{j}{M})) \tag{2.7}$$

$$cos(\frac{\pi}{2}(\frac{i}{N} - \frac{j}{M}) = cos(\frac{i\pi}{2N})cos(\frac{j\pi}{2M}) + sin(\frac{i\pi}{2N})sin(\frac{j\pi}{2M}) \tag{2.8}$$

$$\begin{aligned} S(Q_i, K_j^T) &= (S_q(Q_i)S_k(K_j^T)(cos(\frac{i\pi}{2N})cos(\frac{j\pi}{2M}) + sin(\frac{i\pi}{2N})sin(\frac{j\pi}{2M})) \\ &= S_q(Q_i)cos(\frac{i\pi}{2N})S_k(K_j^T)cos(\frac{j\pi}{2M}) + S_q(Q_i)sin(\frac{i\pi}{2N})S_k(K_j^T)sin(\frac{j\pi}{2M}) \end{aligned} \tag{2.9}$$

It can be noted that re-weighting the query and key matrices in this way is, functionally, the application of positional embedding within the attention mechanism itself, which others have attempted before [31, 53]. However, these alternate re-weighting schemes/embedding methods do not necessarily make the same guarantees concerning softmax approximation that cosFormer ensures, so this thesis does not consider them as viable alternatives for re-weighting during attention linearization.

cosFormer achieves state-of-the-art scores for a number of tasks, including limited autoregressive language modeling on WikiText-103, bi-directional language modeling via being inserted into RoBERTa during pre-training, downstream fine-tuning for various text classification tasks on RoBERTa, and competitive results

on the Long-Range Arena benchmark, which is especially superb given the locality-based assumptions baked into their reweighting mechanism that should, naturally, discourage long-range dependencies. Given the above, this thesis considers cosFormer to be the state-of-the-art in relation to generally applicable attention linearization solutions.

Some problems do exist for cosFormer, however, when it is applied to autoregressive tasks. First and foremost, Zhen et. al do not provide a methodology to deal with the mismatch between casual training and the downstream autoregressive environment as far as target length availability is concerned, and simply stepping the target length at each decoding time-step (i.e. setting it equal to the current sequence length) can prove to be problematic and introduces significant exposure bias downstream. This can be considered for some arbitrary decoding time-step where a given autoregressive model with cosFormer may believe that a target sequence should be shorter than the ground-truth because N in Equation 2.7 classically corresponds to the entire sequence length, but during inference corresponds to the current decoding time-step m from 1 to N. Moreover, while locality bias is often present for attention mechanisms, it is not always relevant depending on the application. Indeed, certain NLP tasks demand consistent long-range dependencies and it can lead to significant performance degradation, in spite of what the Long-Range Arena scores for cosFormer suggests of its ability to deal with long-range dependencies. Finally, cosFormer was never applied towards and tested on encoder to decoder cross-attention, despite its general applicability, and has, as such, not been empirically shown to be a viable candidate for full attention

linearization of all attention blocks in a given model.

Chapter 3: Modular Linearized Attention

3.1 Impetus for Linearization Modularity on S2T NMT and SimulST

Previous works largely explored the necessity of linearized solutions for some tasks that are not feasible with typical softmax attention, but this thesis focuses on the benefits of applying linearization schemes to some tasks that, while possible for classical attention mechanisms, could experience notable run-time benefits from being linearized if costs to prediction quality are mitigated. Speech-to-text (S2T) NMT and simultaneous speech translation (SimulST) were explored early in this thesis as possible linearization applications. While these are not the main NLP tasks explored in this thesis for attention linearization, they are used to underscore the need for more flexible linearized attention implementations. A number of assumptions are made about the intermediate softmax attention matrix that cosFormer [45] attempts to approximate that are not always true on a task by task basis. Chief amongst these assumptions is that the attention matrix typically exhibits significant locality bias. While this can be observed anecdotally by examining the intermediate attention matrices of individual samples, it tends to be more useful to examine the overall evaluation results of a trained model to get a sense of how a linearization scheme performs as a whole (although this thesis does provide some TTS QK^T attention graphs later on for anecdotal examination).

The chosen language pair for this brief experiment was English to German (en-de), mostly because of the common, long-range reordering behavior that exists within this language pair, which is to say that oftentimes German words that would correspond to a strong attention score with some English word are located in a disparate place when compared to that English word. This notion bucks the assumption of cosFormer that a word roughly halfway through some English sentence should exhibit some locality bias towards a word about halfway through a corresponding German translation.

Both translation tasks were trained on identical model architectures based on ESPnet-ST [19], with a slightly different cross-attention module for simultaneous translation that functions based on a wait-k and fixed-predecision paradigm [32]. To avoid cascading multiple models [4], all models were pretrained for automatic speech recognition (ASR) and their encoders were used in initialization when training for S2T NMT from English to German, making them end-to-end S2T NMT pipelines. Unless otherwise specified and in contrast to typical configurations for ESPnet-ST, an increase of 4 to 8 attention heads were used, primarily to explore just how beneficial linearized attention could be for decreased per-head embedding sizes. All models were optimized via Adam [23] with a typical optimization configuration: β_1 and β_2 were set to 0.9 and 0.98 respectively, the learning rate was set to 6e-4, and the learning rate scheduler used an inverse square-root to decay the learning rate. The models were trained with dynamic batching and warmed up for 8000 updates, starting with a learning rate of 1e-4, and trained for around 14000 updates, corresponding to about 30 epochs, with gradients clipped to 10.0

on the MuST-C English to German dataset [7]. All models were trained on four NVIDIA Tesla V100 GPUs and were evaluated on an Intel Xeon Gold 6130 CPU. All models were evaluated via detokenized BLEU-4 through sacreBleu [43] and via SimulEval [33], with the S2T NMT task being executed on a wait-k of 100 to ensure non-simultaneous autoregressive behavior.

For speech-to-text NMT, a number of configurations were tried with varying degrees of success, but fully linearized versions struggled to the point of producing essentially useless translation results. Quantitative results for these initial experiments can be found in Table 3.1, and results for an alternate set of hyperparameters can be found in Table 3.2. It must be noted that cosFormer has no special capability to deal with unknown target lengths during autoregressive inference, so, to demonstrate a simple solution, this thesis analyzes the training split for MuST-C and generate a single ratio to ball-park the target length, multiplying the source length by an α of 0.6. This is significantly larger than the average target to source length ratio, but it was initially considered critical to overestimate in an attempt to preserve the non-negativity of the query and key matrices (this is no longer guaranteed, but is typically true under these circumstances). cosFormer was not applied vigorously towards SimulST as preliminary findings were not encouraging and initial probes did not yield viable target length prediction schemes.

Between the various results below, a few key things can be observed for these particular applications that provide an impetus for later examinations. While there is a severe accuracy penalty for linearizing either encoder self-attention or encoder to decoder cross-attention, the observed penalty for decoder self-attention is sig-

Attention Linearization Scheme	BLEU	ppl(dev)
Softmax Attention (non-Linear)	10.47	9.36
Full cosFormer	2.37	20.08
Full Simple ReLU	1.95	18.33
cosFormer Decoder Self-Attn, Softmax Elsew.	–	9.92
Simple Decoder Self-Attn, Softmax Elsew.	11.07	9.74

Table 3.1: Results from S2T NMT for MuST-C en-de for various linearization schemes with softmax as a baseline. BLEU scores (higher is better) are generated at inference on the test set, are detokenized, and are generated via sacreBLEU. Perplexity (lower is better) is generated during training on the validation set. Missing entries were not explored due to resource/time constraints.

nificantly reduced for both cosFormer and simple ReLU, with their results being fairly close together (it was observed that models could converge to solutions during training that might result in significant variation at inference, i.e. two models trained on different seeds would often differ between 0.1 to 0.5 BLEU). While it was expected that encoder to decoder cross-attention would be difficult to approximate, especially for cosFormer given the aforementioned reordering behavior that bucks cosFormer's locality assumptions, there was a noticeable increase in quality from simple ReLU attention to cosFormer where it might be expected that the opposite would be true. These somewhat surprising results demonstrate two primary ideas that this thesis respects moving forward: firstly, it is insufficient to simply estimate the characteristics of an attention block and find a seemingly suitable linearization scheme as the QK^T intermediate matrix may not always align with expectations, and secondly, attention linearization schemes must be both application-specific and tested on various attention blocks for a given application to gain an accurate understanding of how well softmax behavior is approximated.

Attention Linearization Scheme	BLEU	ppl(dev)
Full cosFormer	2.73	22.12
cosFormer Self-Attn, Simple Cross-Attn	1.89	23.1
cosFormer Self-Attn, Softmax Cross-Attn	7.71	13.35

Table 3.2: Results from S2T NMT for MuST-C en-de for various linearization schemes with slightly different hyperparameters. These models were trained with an embedding dimension d_{model} of 224 and 4 attention heads in an attempt to explore run-time differences for various linearization schemes.

Attention Linearization Scheme	BLEU	ppl(dev)
Softmax Attention (non-Linear)	9.25	10.15
Simple Self-Attn, Softmax Cross-Attn	7.35	12.81
Simple Decoder Self-Attn, Softmax Elsew.	–	9.71

Table 3.3: Results from SimulST for MuST-C en-de for various linearization schemes. Models were trained on a wait-k of 5 and a fixed-predecision ratio of 9 and evaluated on a wait-k of 3 and a fixed-predecision ratio of 5.

3.2 Applying Modular Linearization with cosFormer Towards TTS

Given the particular foibles of the previous task, another application was sought out that might better fit cosFormer for autoregressive environments. Specifically, an application that exhibited significant locality bias and operated on longer sequences would benefit most from cosFormer as far as accuracy and run-time were concerned. In line with the aforementioned requirements, English text-to-spectrogram (TTS, any references to TTS are defined as text-to-spectrogram as opposed to text-to-speech, which implies an embedded vocoder is present) was chosen to demonstrate the capability of attention linearization on autoregressive tasks. Moreover, another attempt at linearizing attention for autoregressive TTS was not found during this paper's literature review, rendering this research novel (implementations for non-

autoregressive TTS do exist, although they are limited [66]). This thesis notes that the paper that introduced cosFormer [45] made no attempt at applying it towards cross-attention for their chosen applications, although it can be generalized for cross-attention, and applied it in a very limited capacity to autoregressive applications. As such, this thesis considers this work novel with respect to those elements as well.

As the previous section demonstrated, attempting to estimate the best linearization scheme based on assumed characteristics is a poor method of applying linearization to attention mechanisms. It is vastly preferable to examine training and inference results for various linearization configurations and view a number of randomly sampled QK^T intermediate matrices to supplement those results. Given that, this thesis proposes a particular design attitude and methodology meant to validate the functionality of a particular linearization scheme (in this case, cosFormer) for use on a given attention block by considering a simple ReLU implementation as a performance baseline, one where the re-weighting function is essentially an identity function that functionally makes no assumptions about approximate softmax behavior beyond ensuring non-negativity of the query and key matrices.

This thesis firmly urges designers to avoid seeking a "one size fits all" attention linearization solution. As the previous section demonstrated and as the results later in this thesis will demonstrate, the applicability of a particular linearization scheme is necessarily application-specific and attention block-specific (one should note that it is possibly layer-specific, or even attention head-specific, as well, but

this level of granularity is beyond the scope of this thesis and would require an excessive amount of training to enable, rendering it too costly to be practical). Given that, this thesis proposes a modular linearization solution where, for some set of possible linearization schemes that function similarly to the solutions proposed by Katharapolous et. al [21], cosFormer [45], and others [10, 66], one tests the applicability of those solutions for encoder self-attention, decoder self-attention, and encoder to decoder cross-attention, comparing them to one another and baseline softmax performance while considering a simple ReLU implementation as a required performance floor for that attention linearization solution's viability.

3.3 Challenges Associated with cosFormer Linearization

This thesis notes some small challenges associated with cosFormer linearization, some of which are associated with the released code-base for this linearization scheme. First and foremost, the released code-base does not natively support batching when batches are not guaranteed to be of the same source and target length, as it uses the largest sequence length in the batch and inherently does not support any kind of key matrix padding. Secondly, as mentioned previously, there is no true support for cross-attention in the code-base or appropriate expression generalizations for cross-attention in the paper introducing cosformer [45]. Finally, the released code-base does not address some of the memory constraints observed for linearized training without some specialized CUDA implementation that ensures limited memory consumption, and the paper introducing cosFormer does not

address problems related to this either. These memory constraints can result in linearized training being so inefficient as to be impractical, requiring a reordering from a $Q(K^TV)$ sequence of matrix multiplications to a $(QK^T)V$ sequence.

Chapter 4: Implementation and Experiments

4.1 Experimental Setup for Linearized Autoregressive TTS

For the purpose of demonstrating the effectiveness of the proposed modular linearized attention, this thesis formalizes the following experiments on Fairseq [38], a language and sequence-modeling toolkit built in PyTorch that is widely used in both research initiatives and industry. This thesis specifically makes use of the Fairseq S^2 extension [60], which enables the development of solutions for text-to-spectrogram, text-to-speech, and speech-synthesis (S^2) tasks and provides a number of popular architectures as development frameworks, most notably amongst those being Tacotron2 [50], TransformerTTS [28], and FastSpeech2 [46]. While FastSpeech2 is considered state-of-the-art for non-autoregressive TTS, it is not usable for autoregressive TTS and is missing some classical transformer elements. Therefore, this thesis chooses to adapt the TransformerTTS architecture, changing the hyperparameters of the model and adding a length predictor to assist cosFormer in downstream environments. This adaptation is showcased in Figure 4.1.

To meet the challenges related to using cosFormer downstream as far as target lengths are concerned, this thesis proposes a few simple solutions that emphasize various aspects of the guarantees and objectives of cosFormer during train-

ing. First, this thesis proposes a simple statistical analysis alongside a set large ratio between the source length to the predicted target length that, while generally ball-parking the reference target length, consistently over-estimates it so as to somewhat ensure the non-negativity of the query and key matrices. This is only possible due to the fact that, for later TTS experiments, the input text is pre-processed into phonemes that should result in a relatively consistent number of output frames. This thesis found that a ratio α of 1.5 results in non-negative transforms of the query and key matrices at least 90% of the time while still maintaining somewhat competitive downstream results with other ratios (a comparison of the performances of these various ratios is provided later). It should be noted that all models that made use of this method were trained on the oracle target length as opposed to some predicted target length, and this method is treated as an inference only implementation. It was observed, generally, that attempting to train with this resulted in models that converged to poor solutions instead of models that were simply robust to the estimation error.

Similarly, this thesis proposes a lookup table (LUT) of average mappings from phonemes to audio frames so as to more accurately match the ground truth target length. While this fails to even reasonably ensure non-negativity for the query and key matrices, as it is not uncommon for this method to slightly underestimate the ground truth target length, the more accurate mapping intuitively could result in better evaluation results, as there is less of a mismatch between the training environment and the downstream environment.

Finally, in the spirit of FastSpeech2 [46], this thesis proposes the implementa-

tion of a small, learned length predictor at the output of the encoder stack that will train alongside the model. This length predictor is modeled directly after the length predictor of FastSpeech2, but has differing hyperparameters, and is a two-layer network composed of two convolutional layers in addition to a ReLU activation and a residual connection with a normalization layer. In contrast to the proposed LUT, this method should be more resilient to variance in phoneme alignment, especially as far as silences are concerned (often encoded as "space" phonemes), but has a more consistent and non-negligibile computational cost.

4.2 Model Configuration and Training Hyperparameters

All models trained for the following experiments were based on TransformerTTS with a length predictor augmentation in accordance to the schemes mentioned above. However, some changes to the baseline configuration were made. Typically, TransformerTTS uses an embedding dimension of 512 and 4 attention heads, whereas for the following experiments an embedding dimension of 256 was used with 8 attention heads. The above linearization schemes are primarily meant for environments where latency and resources are a concern, so a reduction in model size is considered appropriate for a demonstration of the capability of these linearization schemes. It should be noted that, qualitatively, the resulting speech when baseline spectrograms were passed through a Griffin-Lim vocoder [17] is understandable but do suffer from some distortion for the baseline model, and this thesis attributes that distortion both to the reduced embedding size as well as to

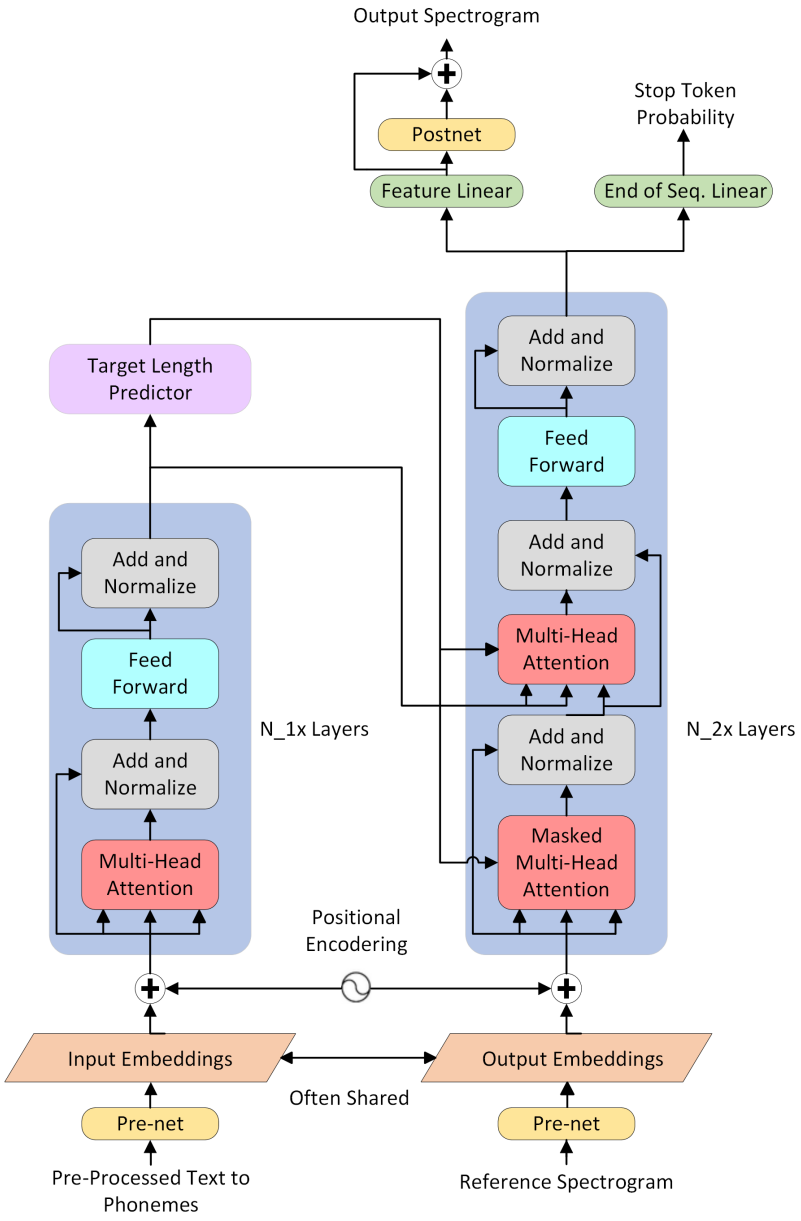

Figure 4.1: Depiction of TransformerTTS-based model used for the following experiments. Input and output embeddings were shared and normalization blocks were reordered to occur before other relevant blocks, despite not being shown here. The length predictor is varied in later experiments, with implementations that are inference only, implementations that are trained, and implementations that must be used during training for model robustness.

not taking advantage of optional pre-processing steps before training [60].

All models were trained with Adam [23] as an optimizer with classical parameters: β_1 and β_2 were set to 0.9 and 0.98 respectively, the learning rate was set to 2e-3, and the learning rate scheduler used an inverse square-root to decay the learning rate. The models were trained with dynamic batching and warmed up for 4000 updates and trained for around 18000 updates with gradients were clipped to 5.0 and layer and attention dropouts of 0.1. All models were trained on four NVIDIA Tesla V100 GPUs and on the single-speaker, English LJSpeech dataset [20] with recommended splits for training, validation, and testing. Around 24 hours of reference audio was extracted from LJSpeech at a sampling frequency of 16 kHz.

4.2.1 End of Sequence Training

As showcased in Figure 4.1, end of sequence prediction is engaged with separately and is a somewhat sensitive part of the training process, with a single positive output resulting in an end of sequence prediction. If the end of sequence linear layer converges to a poor solution, almost unstoppable inference can occur and significant overgeneration on the order of 3x to 4x the number of reference output frames will be observed. To compensate for this issue, a recommended positive weighting of 5.0 in Wang et. al's work [60] was applied to a separately calculated end of sequence loss that serves as an additional training objective.

4.2.2 Target Length Prediction Training

For learned target length prediction, this thesis adopts a very similar approach to FastSpeech2 [46], implementing an identical target length predictor in overall structure but with reduced size (hidden embedding of 128 instead of 256) so as to mitigate the computational cost, in line with previous efforts to properly emulate a low-resource and latency-critical application. A kernel size of 3 was used for the convolutional layers and a steep dropout of 0.5 was applied to train for robustness. Training goals were added via separately calculated loss values for the predicted target length on a phoneme-by-phoneme basis, generated via the Montreal Forced Aligner [35] (directly using the overall sequence length for calculating loss tended to result in volatile training).

It was observed, generally, that training the target length predictor alongside cosFormer when using the oracle target length during training resulted in poorer results than making use of the predicted target lengths during training, but training directly with the predicted target lengths was somewhat volatile. As such, this thesis proposes a brief amount of pre-training on the oracle target length, a brief amount of pre-training with predicted target lengths as a training goal but still using the oracle target length for any cosFormer attention blocks, and the remainder of training being executed using the predicted target length for cosFormer attention blocks. Training results were slightly more stable with this general pre-training framework.

4.2.3 Linearization During Training

While linearization during training can be helpful, it can be difficult to do so without specialized CUDA implementations for casual attention. This is due to the fact that the space complexity of a naive approach for linearized attention during training for decoder self-attention, for example, would require $O(nd^2)$ space complexity, as opposed to the $O(d^2)$ space complexity required during inference. To reuse the $d \times d$ space that is taken by the intermediate K^TV matrix, which must be updated for each row of the query, an algorithm similar to Algorithm 1 would be necessary, and implementing this at a high level tends to introduce significant sequentialism that is not simple for PyTorch to optimize away.

To compensate, a reduction in batch size is necessary, but, in practice, this results in overall slower training. Given that the end results are identical for $(QK^T)V$ and $Q(K^TV)$ orderings, all models were trained in a quadratic rather than linear order. If batch sizes were not a constraint, it would be trivial to implement the above linearization of training in a highly parallelizable manner via PyTorch's Einstein summation notation.

4.3 Evaluation Setup and Metrics

4.3.1 Evaluation Details and Vocoder Use

It is critical to note that directly comparing the synthesized output spectrograms to the reference waveforms would misrepresent the capability of a given model to

Algorithm 1 A simple algorithm for causal linearized attention during training that avoids the $O(nd^2)$ space complexity that would be introduced by a naive implementation.

Input Q in $\mathbb{R}^{N_1 \times d}$, K in $\mathbb{R}^{N_2 \times d}$, V in $\mathbb{R}^{N_2 \times d}$ K_{pr}^{tr} in $\mathbb{R}^{N_2 \times d}$
Output A in $\mathbb{R}^{1 \times d}$
Require Decomposable similarity function defined by S_q and S_k, linear reweighting scheme defined by R_q and R_k

$Q' \leftarrow R_q(S_q(Q))$
$K' \leftarrow R_k(S_k(K))$
$M \leftarrow zeros(N_1, N_2)$
$P \leftarrow zeros(d)$
for i in 1 to N_1 **do**
  $M \leftarrow M + K_i'^T V_i$
  $P \leftarrow P + K_i'^T$
  $A_i \leftarrow \frac{Q_i' M}{P}$
end for

return A

produce understandable speech and fail to account for errors and distortion that might be introduced by fairly classical methods. As such, a Griffin-Lim vocoder [17] was used to process the reference speech into a spectrogram with some minimal distortion so as to avoid the aforementioned issues.

Machines for model evaluation were not necessarily standardized when latency was not being measured, and no difference in evaluation quality was observed for differing machines. For latency-sensitive experiments, a single NVIDIA Tesla V100 GPU was used. A recommended test split of 523 samples from LJSpeech were used for evaluation. Unless otherwise specified, checkpoints of various attention schemes were ranked via their loss, as it encapsulates all of the aforementioned training goals for each model (end of sequence loss, target length prediction, etc.).

4.3.2 Mel Cepstral Distortion and Mel Spectral Distortion

Mel cepstral distortion (MCD) [26] and mel spectral distortion (MSD) have emerged as popular quantitative metrics for speech synthesis and reconstruction [51, 60, 62]. MCD is calculated as demonstrated in Equation 4.1 with dimensionality K of 13 and with the Mel-Frequency Cepstrum Coefficients (MFCCs) $c_{i,k}$ and their synthesized counterparts being calculated via classical methods. For the purposes of this thesis, audio was separated into windows of around 50 ms to avoid time-domain information being collapsed by later Fourier transforms and a hop length of around 12.5 ms generated frames of audio that would be used for MFCC calculation. Those frames were passed through a Hamming window and a 512-point Fast

Fourier Transform (FFT), also known as a Short-Time Fourier Transform (STFT) when composed in this way. Finally, an 80-channel filter bank with a minimum frequency of 20 Hz and a maximum frequency of 8 kHz was applied to generate the MFCCs and the first 13 coefficients were extracted to produce relevant information (the very first one is ignored, as it summarizes the energy of the waveform as a whole).

MSD is provided alongside MCD as they both contain slightly different embedded information. MSD is calculated identically to MCD, but instead on the log-mel spectrum instead of calculated MFCCs. While MFCCs can be intuited as containing primarily phonetic information and not necessarily speaker information, the log-mel spectral features encode both [60]a. As a result, MSD can be considered a more holistic view of the TTS task. Depending on what is desirable, phonetic emphasis versus faithful speech reconstruction, prioritizing one metric over the other may be preferable. This thesis generally defaults to MCD when considering model fitness, but MSD is provided for the sake of completeness.

$$MCD_K = \frac{1}{T}\sum_{t=0}^{T}\sqrt{\sum_{k=1}^{K}(c_{i,k} - c'_{i,k})} \tag{4.1}$$

For samples of differing output frame length between the predicted and reference spectrogram, two primary options exist to align these samples before comparing them: zero padding for the smaller sample [51] and stretching/compression of samples to attempt to align features [5, 62]. This thesis chooses to engage with the latter of these two methods, but also provides MCD/MSD on a per reference frame

basis. This is due to the fact that evaluating purely based on the aforementioned frame alignment tends to suggest, in practice, that models which overgenerate are more capable. This thesis considers frame-aligned MCD/MSD more useful when the difference between the length of the predicted and target frames remains relatively close and defaults to distortion per reference frame when models seem to struggle with end of sequence prediction, which is specified as when a given model predicts a number of output frames larger than two times the number of reference frames.

4.4 Alternative Metrics

Several popular quantitative and qualitative metrics were not employed for this thesis due to its limited scope. Prominent among them are Gross Pitch Error (GPE) [37, 42], Voicing Decision Error (VDE) [37], F0 Frame Error (FFE) [11], and character error rate (CER) [48, 60]. GPE is most useful to determine severe pitching errors for generated waveforms and VDE, which often goes fairly hand in hand with GPE, is relevant when determining whether a voicing error or unvoicing error has been made. Both of these are determined with a number of classical algorithms and heuristics to gather relevant pitch and voicing information [6, 16, 22]. FFE more or less directly combines GPE and VDE into a single metric, measuring both pitching and voicing errors as a proportion of total frames.

On the qualitative side, the most commonly used metric is crowd-sourced mean opinion scores (CMOS), also known as crowdMOS [44], that attempts to judge

the naturalness of speech via the polling of anonymous native speaking listeners. MCD and MSD were considered sufficient for the analysis of quantitative quality of constructed spectrograms with additional context via frame-aligned distortion and distortion per reference frame. This thesis leaves it to later work to examine further insights that can be gathered from employing these other metrics, especially CMOS, which could provide fascinating insights into the overall quality of the produced speech, regardless of the results of observed distortion metrics.

Chapter 5: Results

5.1 Establishing Performance Baselines

Various baseline results were gathered before beginning to explore how combinations of various linearization schemes could further optimize performance. Training results for best checkpoints are provided in Table 5.1 and provided a quick reference for expected quality downstream in later experiments.

It must be noted that cosFormer performs reasonably well during training. While there is significant performance degradation when cosFormer is applied to all attention blocks compared to a model with full softmax attention, cosFormer vastly outperforms a simple ReLU based linearization scheme for every attention block. Of course, it must be noted that training results for cosFormer will, generally, overstate its accuracy, as access to oracle lengths during training ensures excellent results during validation. An example of this can be observed in the end of sequence loss which, for cosFormer implementations in the decoder stack, resulted in loss that nearly converged to zero. As will be observed later on, when cosFormer attention blocks do not have access to the oracle length, they tend to struggle ensuring that an end of sequence is predicted appropriately.

When comparing the two linearization schemes and their various ablations, the most significant gap during training in cosFormer's favor appears to emerge in

the encoder to decoder cross-attention, where cosFormer maintains a very decisive edge in terms of overall loss, although it should be acknowledged that much of this is due to the end of sequence loss having converged to zero. Nonetheless, this aligns with expectations that cosFormer would be particularly suited for encoder to decoder cross-attention for TTS tasks, as the scoring of phonemes, represented in the encoded output of the encoder stack, to their output frames, represented by the hidden embeddings of the decoder stack, should exhibit significant locality bias. cosFormer encoder self-attention also performed well in comparison to simple ReLU encoder self-attention, especially when it comes to end of sequence prediction. A surprising point in simple ReLU's favor is the performance of decoder self-attention for both linearization schemes, where instead of just being competitive a simple ReLU implementation consistently beats out a cosFormer implementation and is even able to engage in end of sequence prediction better than a model with just softmax attention blocks. These results generally suggest that a fully linearized model with cosFormer encoder self-attention, simple ReLU decoder self-attention, and cosformer encoder to decoder cross-attention would perform best downstream with appropriate target length prediction.

Results from the downstream TTS task are presented in Table 5.2 and Table 5.3 for softmax attention, cosFormer encoder self-attention linearization, and simple ReLU linearization alongside its ablations, with cosFormer's results for decoder ablations being presented in later sections due to varying target length prediction schemes. One example result for one of cosFormer's ablations is supplied, however, where no target length prediction was given and, instead, the current decoding

Linearization Scheme	Loss	MSE Loss	EOS Loss
Softmax Attention (non-Linear)	1.021	0.364	0.026
Full cosFormer	1.132	0.440	0.000
cosFormer Enc. Self-Attn, Softmax Elsew.	1.063	0.385	0.029
cosformer Dec. Self-Attn, Softmax Elsew.	1.158	0.455	0.000
cosformer Dec. Cross-Attn, Softmax Elsew.	1.102	0.424	0.000
Full Simple ReLU	1.262	0.496	0.029
Simple ReLU Enc. Self-Attn, Softmax Elsew.	1.097	0.399	0.037
Simple ReLU Dec. Self-Attn, Softmax Elsew.	1.133	0.424	0.025
Simple ReLU Dec. Cross-Attn, Softmax Elsew.	1.162	0.438	0.031

Table 5.1: Training results for various linearization schemes and their ablations, providing a brief overview of their expected performance at inference. All loss values (lower is better) are provided from the best performing checkpoint on the validation split. Note that, generally, cosFormer's loss for decoder attention blocks tends to overestimate its downstream performance given that it can access oracle lengths during training and all of these cosFormer implementations were not trained with the learned target length prediction module.

time-step was used as the supposed sequence length. This is present entirely to validate the importance of predicting the target length and to showcase that the approach provided in Zhen et. al's [45] paper and code-base is insufficient for autoregressive tasks. It should be noted that some downstream volatility was observed for linearization schemes that was, primarily, tied to a given linearized model's ability to properly predict the end of a sequence. As can be clearly seen, the presented linearized models all struggled with significant overgeneration issues that seem to indicate a poor ability to predict the end of a sequence. For some models, such as the provided cosFormer decoder ablation and the ReLU encoder to decoder cross-attention, they essentially achieved unstoppable inference and exceeded nearly six times the reference target length for their synthesized duration.

It is worth noting that it is somewhat odd for the fully linearized solution listed in Table 5.2 and Table 5.3 to perform better downstream compared to a model with just ReLU encoder self-attention or just ReLU encoder to decoder cross-attention, but it should be observed that both of those models struggle significantly with overgeneration, and this can be seen during training in their slightly larger end of sequence loss. This thesis attributes the strange downstream results for full simple ReLU linearization to volatility in the performance of trained checkpoints downstream (i.e. it was observed that checkpoints that achieved essentially identical validation loss could exhibit significant differences in downstream performance, especially for ReLU linearization). This could possibly be compensated for by applying an even larger positive weight to the end of sentence prediction during training [60], but this is left for future work to explore.

Linearization Scheme	Synth. Dur. (frames)	Dist./Ref.	Dist./Align.
Softmax Attention (non-Linear)	285k	5.52	4.69
Full Simple ReLU	414k	10.87	6.31
Simple ReLU Enc. Self-Attn.	819k	14.11	4.45
Simple ReLU Dec. Self-Attn.	323k	8.20	6.10
Simple ReLU Dec. Cross-Attn.	1372k	22.34	4.35
cosFormer Enc. Self-Attn.	313k	7.00	5.20
cosFormer Dec. Cross-Attn.	1989k	29.47	4.01

Table 5.2: MCD values (lower is better) for full softmax attention, simple ReLU linearization, alongside its ablations, and cosFormer implemented in encoder self-attention. Ablations made use of softmax attention for all blocks but the specified linearized block. A single decoder cosFormer ablation is provided without target length prediction to validate the necessity of avoiding using the current decoding time-step in the cosine-based reweighting scheme's denominator. Reference duration is approximately 273k frames across all 523 test samples in LJSpeech.

Linearization Scheme	Synth. Dur. (frames)	Dist./Ref.	Dist./Align.
Softmax Attention (non-Linear)	285k	2.61	2.27
Full Simple ReLU	414k	4.80	2.89
Simple ReLU Enc. Self-Attn.	819k	6.64	2.12
Simple ReLU Dec. Self-Attn.	323k	3.66	2.81
Simple ReLU Dec. Cross-Attn.	1372k	10.44	2.05
cosFormer Enc. Self-Attn.	313k	3.22	2.45
cosFormer Dec. Cross-Attn.	1989k	13.95	1.91

Table 5.3: MSD values (lower is better) for full softmax attention, simple ReLU linearization, alongside its ablations, and cosFormer implemented in encoder self-attention. Ablations made use of softmax attention for all blocks but the specified linearized block. A single cosFormer ablation is provided without target length prediction to validate the necessity of avoiding using the current decoding time-step in the cosine-based reweighting scheme's denominator. Reference duration is approximately 273k frames across all 523 test samples in LJSpeech.

5.2 Ratio-Based Target Length Prediction Iteration

As mentioned previously, one of the proposed target length prediction schemes relies on a simple statistical analysis of the training set target length to source length ratios and a subsequent application of a set ratio α during inference to estimate the reference target length. Here, an α of 1.25 closely approximates the average length that the reference output should have whereas an α of 1.5 produces around a 90% probability, based on the training set, that the query and key matrices will be positive. During experiments early on in this thesis related to NMT and SimulST, a larger ratio was chosen to attempt to mostly ensure the non-negativity of the query and key matrices.

However, as demonstrated in Table 5.4 and Table 5.5, it is not clear that overestimating the target length is beneficial in any way due to the general difficulty

that linearized models have in correctly predicting the end of a sequence. Due to that tendency to overgenerate, this thesis notes the capability of this target length prediction method of dampening the model's eagerness to avoid end of sentence predictions. It should be noted, briefly, the important information encoded in both distortion per reference frame and distortion per aligned frame, as optimizing for either would result in α values that are patently ridiculous compared to the environment that these linearized attention mechanisms were trained in. Even so, an α of 1.125 could be used to better match the reference target length while also maintaining a somewhat balanced performance profile, in spite of the expected poorer approximation of softmax behavior by consistently, slightly underestimating the target sequence length.

α Length Ratio	Synth. Dur. (frames)	Dist./Ref.	Dist./Align.
0.50	167k	7.42	7.05
0.60	212k	7.73	6.99
0.75	307k	8.49	6.59
1.00	431k	9.70	5.75
1.25	531k	10.74	5.27
1.50	650k	12.00	4.88
1.75	772k	13.30	4.59

Table 5.4: MCD values (lower is better) for differing target length to source length ratios for target length predictions. All models were fundamental TransformerTTS architectures with their encoder to decoder cross-attention mechanism replaced by cosFormer. Reference duration is approximately 273k frames across all 523 test samples in LJSpeech. Extremely low ratios are provided to demonstrate the problems with relying purely on distortion per reference frame and extremely large ratios are provided to demonstrate issues with relying entirely on distortion per alignment frame.

α Length Ratio	Synth. Dur. (frames)	Dist./Ref.	Dist./Align.
0.50	167k	3.28	3.16
0.60	212k	3.42	3.17
0.75	307k	3.78	3.06
1.00	431k	4.42	2.69
1.25	531k	4.96	2.50
1.50	650k	5.62	2.33
1.75	772k	6.30	2.21

Table 5.5: MSD values (lower is better) for differing target length to source length ratios for target length predictions. All models were fundamental TransformerTTS architectures with their encoder to decoder cross-attention mechanism replaced by cosFormer. Reference duration is approximately 273k frames across all 523 test samples in LJSpeech. Extremely low ratios are provided to demonstrate the problems with relying purely on distortion per reference frame and extremely large ratios are provided to demonstrate issues with relying entirely on distortion per alignment frame.

5.3 Lookup Table-Based Target Length Prediction

To construct the aforementioned lookup table, data was collected related to the training set alignments produced by the Montreal Forced Aligner [35]. Averages were determined alongside standard deviations to briefly test whether or not overestimation of the reference target length would yield generally better results (ensuring that 90% of expected encountered target lengths would result in positive queries and keys, similar to previous experiments with NMT and SimulST as tasks). Upon observing that such guarantees did not, typically, result in generally superior output quality, both quantitatively and qualitatively, average phoneme to audio frame mappings were generally used. It should be noted that quiet frames, or "spaces," which correspond to a single phoneme necessarily suffer from very large phoneme

to audio frame mapping variance. While the average mapping for a "space" was around 40 frames of audio, a standard deviation of 20 frames was observed, making predicting the actual length of silences incredibly difficult.

Results are observed for various dampening factors are provided in Table 5.6 and Table 5.7, with these dampening factors serving to suppress or strengthen a linearized model's general tendency to overgenerate. Encouraging overgeneration via a dampening factor of 0.9 is provided purely to test the severity of observed prediction quality degradation. Generally, the use of LUT-based length prediction, while somewhat competitive when dampening was enabled, failed to perform above some simple ratio α, which can also provide a kind of dampening effect.

Dampening Factor	Synth. Dur. (frames)	Dist./Ref.	Dist./Align.
1.1	342k	8.81	6.11
1.0	378k	9.20	5.89
0.9	450k	9.56	5.55

Table 5.6: MCD values (lower is better) for various LUT dampening factors. All models contained cosFormer attention blocks for encoder self-attention and encoder to decoder cross-attention with softmax decoder self-attention. Reference duration is approximately 273k frames across all 523 samples in the test set of LJSpeech.

Dampening Factor	Synth. Dur. (frames)	Dist./Ref.	Dist./Align.
1.1	342k	3.94	2.89
1.0	378k	4.13	2.78
0.9	450k	4.5	2.62

Table 5.7: MSD values (lower is better) for various LUT dampening factors. All models contained cosFormer attention blocks for encoder self-attention and encoder to decoder cross-attention with simple decoder self-attention. Reference duration is approximately 273k frames across all 523 samples in the test set of LJSpeech.

5.4 Learned Target Length Prediction

This thesis briefly explores learned target length prediction as a possible target length prediction scheme, but critically acknowledges that any learned module that does not produce significant improvements on downstream results likely renders this scheme unviable, as there is a non-negligible computational cost associated with this small convolutional neural network embedded into the encoder stack. The results for this experiment can be found in Table 5.8 and Table 5.9 for a few decoder-focused cosFormer ablations. While the generated down-stream results are certainly competitive, they are not notably better than a simple ratio of *alpha* when it is used as a general dampener for target length overgeneration. It should be observed that, when cosFormer was applied to all attention blocks, the synthesized duration was actually closer to the reference length. While this validates learned target length prediction on some level, this thesis notes that even when models with identical training parameters achieved similar training results, their downstream performance could vary significantly, and attributes this observed improvement to that volatility.

Linearization Scheme	Synth. Dur. (frames)	Dist./Ref.	Dist./Align.
cosFormer Dec. Attn. Blocks	328k	9.09	6.68
cosFormer Dec. Self-Attn.	334k	8.98	6.58
cosFormer Dec. Cross-Attn.	367k	9.38	6.35

Table 5.8: MCD values (lower is better) for various cosFormer ablations with learned target length prediction. Ablations made use of softmax for all attention blocks other than the specified linear ones. All models followed the brief pre-training protocol outlined earlier in this thesis. Reference duration is approximately 273k frames across all 523 test samples in the test set of LJSpeech.

Linearization Scheme	Synth. Dur. (frames)	Dist./Ref.	Dist./Align.
cosFormer Dec. Attn. Blocks	328k	4.04	3.13
cosFormer Dec. Self-Attn.	334k	3.97	3.05
cosFormer Dec. Cross-Attn.	367k	4.17	2.93

Table 5.9: MSD values (lower is better) for various cosFormer ablations with learned target length prediction. Ablations made use of softmax for all attention blocks other than the specified linear ones. All models followed the brief pre-training protocol outlined earlier in this thesis. Reference duration is approximately 273k frames across all 523 test samples in the test set of LJSpeech.

5.5 Finalized Model Configuration and Performance Comparisons

5.5.1 Competitive Models and Synthesized Spectrogram Quality

Based on the results from previous sections, a few combinations of linearization schemes were considered reasonable to investigate for this thesis. When examining the training results alone for various linearization schemes, it seems viable to at least consider the following options: cosFormer encoder self-attention and encoder to decoder cross-attention with simple ReLU decoder self-attention, cosFormer encoder self-attention and encoder to decoder cross-attention with softmax decoder self-attention, and cosFormer encoder to decoder cross-attention with softmax self-attention blocks (latency profiles in the Appendix suggest that for this workload, there may not be significant latency benefits for linearizing decoder self-attention and decoding should dominate run-time, so encoder self-attention linearization also may not be necessary). Upon examining the downstream results for various ablations, it can generally be expected that models incorporating cosFormer encoder to decoder cross-attention will perform reasonably well if the target length is some-

what closely predicted. Additionally, cosFormer encoder self-attention performed well, although it should be noted that the encoder self-attention block was rather sensitive to checkpoint selection (i.e. evaluating checkpoints with reasonably close loss values could swing the evaluation significantly, usually due to poor end of sequence prediction).

Of the aforementioned target length prediction options, this thesis notes that a simple ratio α that very roughly estimates the target length performs competitively with the LUT and learned target length prediction alternatives with negligible computational cost, in comparison (LUT run-time complexity is $O(n)$ and target length prediction is roughly $O(kndd_h)$ where k is the kernel size and d_h is the number of filters). As such, a simple α of 1.125 is employed for the following final evaluations, selected at a slightly lower value than the observed average ratio in the training set to dampen the tendency of linearized schemes to overgenerate. It is left to future work to seek out alternative, more accurate target length prediction schemes that produce more noticeable improvements upon downstream results.

Results for inference runs with the aforementioned models are provided in Table 5.10 and Table 5.11. As can be observed below, their performance is surprisingly similar, but given that the synthesized lengths are fairly close to the reference length, it can be generally stated that the model with softmax decoder self-attention likely has a slight edge as far as prediction quality is concerned, as the distortion per aligned frame is slightly lower for both MCD and MSD, indicating general superiority in phonetic reconstruction and other speaker-specific features. A model with only cosFormer encoder to decoder cross-attention is included with

a slightly different training configuration that encourages underestimation of the reference target length that seems to perform the best out of all available options, but given that it does include a slightly different training configuration it is only included for the sake of completeness.

Linearization Scheme	Synth. Dur. (frames)	Dist./Ref.	Dist./Align.
cosFormer Enc. Self-Attn & Dec. Cross-Attn, simple ReLU Elsew.	273k	8.10	6.57
cosFormer Enc. Self-Attn & Dec. Cross-Attn, Softmax Elsew.	293k	8.35	6.43
cosFormer Dec. Cross-Attn, Softmax Elsew. [1]	297k	8.20	6.30

Table 5.10: MCD values (lower is better) for various final linearization schemes, two of which are not fully linearized. Reference duration is approximately 273k frames across all 523 samples in the test set of LJSpeech.

5.5.2 Estimated FLOPs and Observed Latency

This thesis briefly touches upon a calculation of the estimated number of floating point operations (FLOPs) for each attention block for a single attention head during inference in the presented hybrid model. This thesis assumes, initially, that all floating point operations are equal and then follows up that analysis by making the following assumptions about various floating point operations and the relative

Linearization Scheme	Synth. Dur. (frames)	Dist./Ref.	Dist./Align.
cosFormer Enc. Self-Attn & Dec. Cross-Attn, simple ReLU Elsew.	273k	3.59	3.08
cosFormer Enc. Self-Attn & Dec. Cross-Attn, Softmax Elsew.	293k	3.7	3.05
cosFormer Dec. Cross-Attn, Softmax Elsew. [1]	297k	3.64	2.93

Table 5.11: MSD values (lower is better) for various final linearization schemes, two of which are not fully linearized. Reference duration is approximately 273k frames across all 523 samples in the test set of LJSpeech.

time that they consume:

- Addition, subtraction, and multiplication are all equivalent to 1 general FLOP
- Division is equivalent to 2 general FLOPs
- Exponential operations are equivalent to 2 general FLOPs

As a reference, this thesis makes use of the provided run-time complexities in Table 5.12 and Table 5.13, but expands collapsed terms for FLOPs calculation (e.g. the softmax operator application is expanded from m operations to $3m$ operations for exponentiation, summation, and division across a single token's

[1] A somewhat experimental model with slightly differing training parameters is used here that encouraged it to underestimate target sequence lengths. An α of 1.75 was necessary to encourage it to avoid significant underestimation.

attention scores). Estimations for the FLOPs required are provided in Table 5.14, Table 5.15, and Table 5.16.

Attention Mechanism Component	Run-Time Complexity
Q, K, V Linear Projections	$O(nDd)$
(QK^T) Intermediate Matrix Multiplication	$O(n^2d)$
softmax Operator Application	$O(n^2)$
$(QK^T)V$ Matrix Multiplication	$O(n^2d)$
ReLU Operator Application	$O(nd)$
cosine-Based Transform Application	$O(nd)$
(K^TV) Intermediate Matrix Multiplication	$O(nd^2)$
$Q(K^TV)$ Matrix Multiplication	$O(nd^2)$
Linearized Normalization Construction	$O(nd)$
Linearized Normalization Application	$O(nd)$
Output Attention Projection	$O(nDd)$

Table 5.12: Estimated run-time complexities during autoregressive inference of various encoder self-attention mechanisms where D is the entire embedding size for the model and d is the embedding size for a single attention head (d_k is assumed to equal d_v here).

It can be observed that as far as estimated number of FLOPs is concerned, both cosFormer and simple ReLU attention blocks beat out softmax implementations by a notable margin, especially in cross-attention for autoregressive decoding. However, while estimating FLOPs and calculating run-time complexities is a necessary step in estimating the acceleration of attention calculations via linearization, it should be noted that various under-the-hood optimizations on the side of PyTorch may render some of these estimations inaccurate when it comes to practical run-times. Given that, latencies for various viable schemes are provided below and, for completeness, a brief comparison of run-time profiles for various attention schemes in practical environments and engaging in rote calculation are provided

Attention Mechanism Component	Run-Time Complexity
Q, K, V Linear Projections	$O(Dd)$
(QK^T) Intermediate Matrix Multiplication	$O(md)$
softmax Operator Application	$O(m)$
$(QK^T)V$ Matrix Multiplication	$O(md)$
ReLU Operator Application	$O(d)$
cosine-Based Transform Application	$O(d)$
(K^TV) Intermediate Matrix Multiplication	$O(d^2)$
$Q(K^TV)$ Matrix Multiplication	$O(d^2)$
Linearized Normalization Construction	$O(d)$
Linearized Normalization Application	$O(d)$
Output Attention Projection	$O(Dd)$

Table 5.13: Estimated run-time complexities during autoregressive inference of various decoder self-attention mechanisms at some arbitrary decoding time-step m where D is the entire embedding size for the model and d is the embedding size for a single attention head (d_k is assumed to equal d_v here). Decoder cross-attention replaces all m values with the encoded sequence length. Maximal reuse is assumed for relevant linearization elements.

in the Appendix.

Concerning observed latencies, it can be somewhat difficult to account for differences in predicted sequence length between these various schemes. The fairest metric employed in this thesis, beyond estimated FLOPs, is throughput, and throughput results are provided in Table 5.17. As observed in this table, decoder throughput gains of up to 7.2% were achieved with a more balanced approach achieving around 5% decoder throughput gains. While around a 3.9% encoder throughput increase was observed, it should not be emphasized for this application as decoder run-time typically eclipses encoder run-time by a factor of at least 5x to 20x.

Attention Scheme	FLOPs when Equal	FLOPs when Unequal
Softmax Attention	7.86M	7.88M
Simple ReLU Attention	6.98M	6.99M
cosFormer Attention	7.41M	7.41M

Table 5.14: Estimated required FLOPs for encoder self-attention for a single attention head when assuming all FLOPs are equal and when making some assumptions about the relative time consumption of certain operations. For the purposes of this calculation, the encoded sequence length n_e is set to 100 tokens, D is set to a total embedding space of 256, and d is set to an embedding space of 32 for a single attention head, in accordance with the models trained for this thesis.

Attention Scheme	FLOPs when Equal	FLOPs when Unequal
Softmax Attention	12.4M	13.1M
Simple ReLU Attention	10.6M	10.6M
cosFormer Attention	11.4M	11.4M

Table 5.15: Estimated required FLOPs for decoder self-attention for a single attention head when assuming all FLOPs are equal and when making some assumptions about the relative time consumption of certain operations. The represented FLOPs are for a full autoregressive prediction, with a sequence length of n. For the purposes of this calculation, n is set as a sequence length of 150 tokens, D is set to a total embedding space of 256, and d is set to an embedding space of 32 for a single attention head, in accordance with the models trained for this thesis.

Attention Scheme	FLOPs when Equal	FLOPs when Unequal
Softmax Attention	11.6M	12.5M
Simple ReLU Attention	5.4M	5.5M
cosFormer Attention	5.8M	5.8M

Table 5.16: Estimated required FLOPs for decoder cross-attention for a single attention head when assuming all FLOPs are equal and when making some assumptions about the relative time consumption of certain operations. The represented FLOPs are for a full autoregressive prediction, with a sequence length of n and an encoded sequence length of n_e. For the purposes of this calculation, n is set as a sequence length of 150 tokens, n_e is set to a sequence length of 100, D is set to a total embedding space of 256, and d is set to an embedding space of 32 for a single attention head, in accordance with the models trained for this thesis.

Linearization Scheme	Enc. Thrpt. (itr/sec)	Dec. Thrpt. (itr/sec)	FLOPs
Softmax Attention (non-Linear)	1.51	72.81	1.94G
cosFormer eSA & dCA, ReLU dSA	1.57	77.88	1.46G
cosFormer eSA & dCA, Softmax dSA	1.58	76.53	1.60G
cosFormer dCA, Softmax eSA & dSA	1.52	76.56	1.68G

Table 5.17: Efficiency related results from various configurations generating spectrograms on the LJSpeech test set. Encoder and decoder throughput (higher is better) was measured via the number of forward calls and the wall-clock time for those calls. FLOPs (lower is better) were calculated and are an estimation of floating point operations (all operations treated as equal) for a single sample with a source sequence length of 100 and a target sequence length of 150. Small variations in throughput are attributed to slightly different device conditions between efficiency tests.

Chapter 6: Conclusions and Future Work

Classical attention mechanisms for the transformer class of models can be computationally expensive for many NLP tasks. Even for tasks where softmax attention is still viable as far as latency in concerned, it can be desirable for various attention mechanisms to be linearized as long as costs to downstream prediction quality are minimized. This thesis focuses on some of the problems related to applying state-of-the-art linearization techniques to various autoregressive tasks with a focus on TTS.

Specifically, this thesis hones in initially on providing rebuttals to the general effort to find a one-size-fits-all solution for attention linearization by underscoring the importance of choosing a linearization scheme on an application by application basis. NMT and SimulST are employed as example applications that can be difficult to provide linear softmax approximations for, where state-of-the-art solutions like cosFormer [45] largely fail to mitigate prediction accuracy degradation when broadly applied. Additionally, these applications underscore the importance of engaging with multiple linearization schemes and attempting to combine them, as such combinations can yield fascinating insights related to the general downstream behavior of the QK^T intermediate matrix.

Moreover, this thesis seeks to answer some of the challenges specific to autoregressive tasks for cosFormer, specifically related to fixing its initial naive imple-

mentation and providing numerous methods for target length prediction, which are absent in the original implementation for autoregressive tasks. Often, naive implementations of cosFormer fail to predict anything more than end of sequence tokens or often significantly stretch features to the point of rendering the resulting prediction unrecognizable compared to the reference. As demonstrated in the results of this thesis, the improvement from a naive cosFormer-based model and its ablations to one with the solutions provided in this thesis is beyond significant, rendering such improvements mandatory for autoregressive applications.

To better explore cosFormer and how modular linearized attention could be employed to better approximate softmax attention across all attention blocks, TTS was chosen as an application due to its expected strong locality bias across multiple attention blocks. Trained, evaluated, and tested on the single-speaker dataset LJSpeech [20], various linearization ablations based on TransformerTTS [28] were analyzed to determine what combination might best approximate softmax attention for full linearization. As the results in this thesis have shown, for this dataset and for TTS as an application, a cosFormer encoder self-attention block combined with a cosFormer encoder to decoder cross-attention block and a simple ReLU decoder self-attention block perform the best out of a number of ablations with a notable latency reduction compared to full softmax attention. Due to the size of this workload, full linearization is not entirely necessary, and upon replacing the ReLU decoder self-attention block with a softmax decoder self-attention block, a slight quality improvement was observed while still maintaining a significant end-to-end speed-up in comparison to full softmax attention.

A number of tasks remain to be fully explored that this thesis only touches on, most notable amongst them being the creation and selection of a target length prediction scheme with near negligible computational cost that better predicts target sequence lengths than some simple ratio α. As the training results in this thesis demonstrated, cosFormer performs excellently when oracle sequence lengths are available for attention blocks that exhibit significant locality bias, so a more sophisticated and accurate target length predictor should result in vastly improved prediction quality. Additionally, optimization opportunities for the various linearized models within this thesis remains for TTS, as a number of preprocessing steps were not explored that could improve prediction quality. Moreover, the limits of these linearized models for multi-speaker datasets have not been explored by this thesis, and the aforementioned preprocessing steps can assist in cleaning up multi-speaker or noisy datasets which could enable linearized solutions for more complex TTS environments.

Bibliography

[1] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. Etc: Encoding long and structured inputs in transformers, 2020.

[2] Pooya Ashtari, Diana Maria Sima, Lieven De Lathauwer, Dominique Sappey-Marinier, Frederik Maes, and Sabine Van Huffel. Factorizer: A scalable interpretable approach to context modeling for medical image segmentation. *Medical image analysis*, 84, 2022.

[3] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

[4] Luisa Bentivogli, Mauro Cettolo, Marco Gaido, Alina Karakanta, Alberto Martinelli, Matteo Negri, and Marco Turchi. Cascade versus direct speech translation: Do the differences still make a difference? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2873–2887, Online, August 2021. Association for Computational Linguistics.

[5] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS'94, page 359–370. AAAI Press, 1994.

[6] Paul Boersma. Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound. 1993.

[7] Roldano Cattoni, Mattia Antonino Di Gangi, Luisa Bentivogli, Matteo Negri, and Marco Turchi. Must-c: A multilingual corpus for end-to-end speech translation. *Computer Speech & Language*, 66:101155, 2021.

[8] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost, 2016.

[9] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers, 2019.

[10] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2020.

[11] Wei Chu and Abeer Alwan. Reducing f0 frame error of f0 tracking algorithms under noisy conditions with an unvoiced/voiced classification frontend. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3969–3972, 2009.

[12] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does bert look at? an analysis of bert's attention, 2019.

[13] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, 2019.

[14] Giannis Daras, Nikita Kitaev, Augustus Odena, and Alexandros G. Dimakis. Smyrf: Efficient attention using asymmetric clustering, 2020.

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[16] Thomas Drugman and Abeer Alwan. Joint robust voicing detection and pitch estimation based on residual harmonics, 2020.

[17] D. Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984.

[18] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer, 2018.

[19] Hirofumi Inaguma, Shun Kiyono, Kevin Duh, Shigeki Karita, Nelson Enrique Yalta Soplin, Tomoki Hayashi, and Shinji Watanabe. Espnet-st: All-in-one speech translation toolkit, 2020.

[20] Keith Ito and Linda Johnson. The lj speech dataset, 2017.

[21] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention, 2020.

[22] Hideki Kawahara, Jo Estill, and Osamu Fujimura. Aperiodicity extraction and control using mixed mode excitation and group delay manipulation for a high quality speech analysis, modification and synthesis system straight. In *International Workshop on Models and Analysis of Vocal Emissions for Biomedical Applications*, 2001.

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[24] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer, 2020.

[25] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. Revealing the dark secrets of bert, 2019.

[26] Robert F. Kubichek. Mel-cepstral distance measure for objective speech quality assessment. *Proceedings of IEEE Pacific Rim Conference on Communications Computers and Signal Processing*, 1:125–128 vol.1, 1993.

[27] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks, 2018.

[28] Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, Ming Liu, and Ming Zhou. Neural speech synthesis with transformer network, 2018.

[29] Shengqiang Li, Menglong Xu, and Xiao-Lei Zhang. Efficient conformer-based speech recognition with linear attention, 2021.

[30] Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences, 2018.

[31] Antoine Liutkus, Ondřej Cífka, Shih-Lun Wu, Umut Şimşekli, Yi-Hsuan Yang, and Gaël Richard. Relative positional encoding for transformers with linear complexity, 2021.

[32] M. Ma, L. Huang, H. Xiong, R. Zheng, K. Liu, B. Zheng, C. Zhang, Z. He, H. Liu, X. Li, H. Wu, and H. Wang. Stacl: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036, Florence, Italy, 2019. Association for Computational Linguistics (ACL).

[33] Xutai Ma, Mohammad Javad Dousti, Changhan Wang, Jiatao Gu, and Juan Pino. Simuleval: An evaluation toolkit for simultaneous translation, 2020.

[34] Ali Madani, Bryan McCann, Nikhil Naik, Nitish Shirish Keskar, Namrata Anand, Raphael R. Eguchi, Po-Ssu Huang, and Richard Socher. Progen: Language modeling for protein generation, 2020.

[35] Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. Montreal forced aligner: Trainable text-speech alignment using kaldi. In *Interspeech*, 2017.

[36] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training, 2017.

[37] Tomohiro Nakatani, Shigeaki Amano, Toshio Irino, Kentaro Ishizuka, and Tadahisa Kondo. A method for fundamental frequency estimation and voicing decision: Application to infant utterances recorded in real acoustical environments. *Speech Communication*, 50(3):203–214, 2008.

[38] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling, 2019.

[39] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer, 2018.

[40] Archit Parnami, Rahul Singh, and Tarun Joshi. Pruning attention heads of transformer models using a* search: A novel approach to compress big nlp architectures, 2021.

[41] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. Random feature attention, 2021.

[42] Adam Polyak, Lior Wolf, Yossi Adi, and Yaniv Taigman. Unsupervised cross-domain singing voice conversion, 2020.

[43] Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels, October 2018. Association for Computational Linguistics.

[44] Flavio Protasio Ribeiro, Dinei Florencio, Cha Zhang, and Mike Seltzer. Crowdmos: An approach for crowdsourcing mean opinion score studies. In *ICASSP*. IEEE, May 2011.

[45] Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosformer: Rethinking softmax in attention. In *International Conference on Learning Representations*, 2022.

[46] Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech 2: Fast and high-quality end-to-end text to speech, 2020.

[47] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers, 2020.

[48] Ryosuke Sawata, Yosuke Kashiwagi, and Shusuke Takahashi. Improving character error rate is not equal to having clean speech: Speech enhancement for asr systems with black-box acoustic models, 2021.

[49] Guan Shen, Jieru Zhao, Quan Chen, Jingwen Leng, Chao Li, and Minyi Guo. Salo: An efficient spatial accelerator enabling hybrid sparse attention mechanisms for long sequences, 2022.

[50] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ Skerry-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui Wu. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions, 2017.

[51] RJ Skerry-Ryan, Eric Battenberg, Ying Xiao, Yuxuan Wang, Daisy Stanton, Joel Shor, Ron J. Weiss, Rob Clark, and Rif A. Saurous. Towards end-to-end prosody transfer for expressive speech synthesis with tacotron, 2018.

[52] Nimit S. Sohoni, Christopher R. Aberger, Megan Leszczynski, Jian Zhang, and Christopher Ré. Low-memory neural network training: A technical report, 2019.

[53] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2021.

[54] Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. Adaptive attention span in transformers, 2019.

[55] Thierry Tambe, Coleman Hooper, Lillian Pentecost, Tianyu Jia, En-Yu Yang, Marco Donato, Victor Sanh, Paul N. Whatmough, Alexander M. Rush, David Brooks, and Gu-Yeon Wei. Edgebert: Sentence-level energy optimizations for latency-aware multi-task nlp inference, 2020.

[56] Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: A unified understanding of transformer's attention via the lens of kernel, 2019.

[57] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, , and I. Polosukhin. Attention is all you need. In *Proceedings of 31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017.

[58] Prateek Verma and Jonathan Berger. Audio transformers:transformer architectures for large scale audio understanding. adieu convolutions, 2021.

[59] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned, 2019.

[60] Changhan Wang, Wei-Ning Hsu, Yossi Adi, Adam Polyak, Ann Lee, Peng-Jen Chen, Jiatao Gu, and Juan Pino. fairseq s^2: A scalable and integrable speech synthesis toolkit, 2021.

[61] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020.

[62] Ron J. Weiss, RJ Skerry-Ryan, Eric Battenberg, Soroosh Mariooryad, and Diederik P. Kingma. Wave-tacotron: Spectrogram-free end-to-end text-to-speech synthesis, 2020.

[63] Qingyang Wu, Zhenzhong Lan, Kun Qian, Jing Gu, Alborz Geramifard, and Zhou Yu. Memformer: A memory-augmented transformer for sequence modeling, 2020.

[64] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. 2020.

[65] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. 2020.

[66] Haozhe Zhang, Zhihua Huang, Zengqiang Shang, Pengyuan Zhang, and Yonghong Yan. LinearSpeech: Parallel Text-to-Speech with Linear Complexity. In *Proc. Interspeech 2021*, pages 4129–4133, 2021.

APPENDICES

Appendix A: Feature Comparisons for Synthesized Spectrograms

A handful of samples were randomly selected for comparisons between the reference spectrogram and the synthesized spectrograms. In many cases, the linearized model (only cross-attention was linearized for this comparison) was able to somewhat compete with the baseline, full softmax attention model insofar as visible features are concerned. Figure A.1 and Figure A.2 are good examples of an instance where the linearized model seems to have performed somewhat comparably, with both capturing the starting features reasonably well. Spectrogram quality, in general, seemed to degrade for later decoding time-steps.

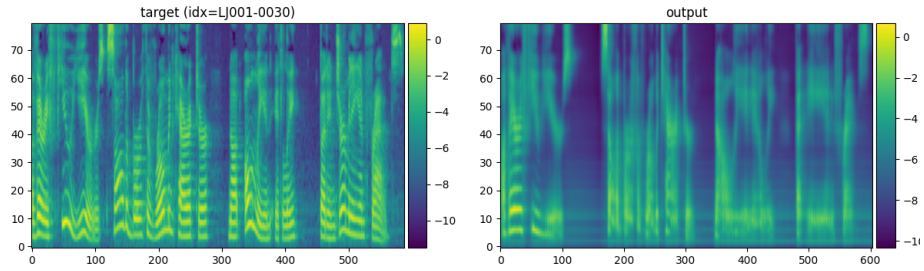

Figure A.1: Spectrogram comparison between the reference spectrogram and the synthesized spectrogram from a baseline model with only softmax attention on sample LJ001-0030. This sample was randomly selected out of the test set. The left axis is the frequency in kHz, the right axis is the decibel value, and the bottom axis is the duration in frames.

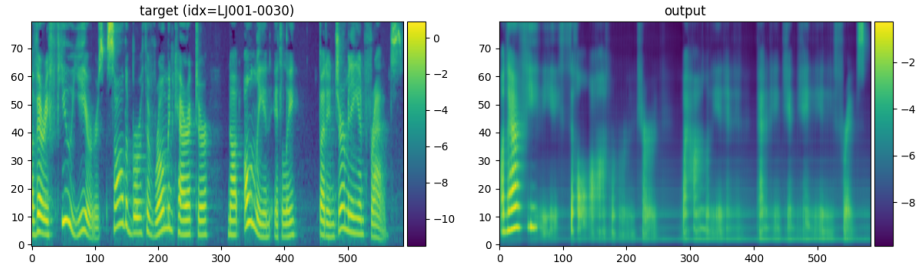

Figure A.2: Spectrogram comparison between the reference spectrogram and the synthesized spectrogram from a model with cosFormer encoder to decoder cross-attention and softmax self-attention blocks on sample LJ001-0030. This sample was randomly selected out of the test set. The left axis is the frequency in kHz, the right axis is the decibel value, and the bottom axis is the duration in frames.

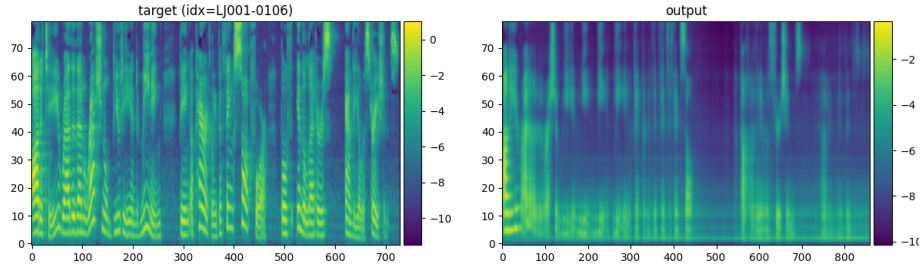

Figure A.3: Spectrogram comparison between the reference spectrogram and the synthesized spectrogram from a baseline model with only softmax attention on sample LJ001-0106. This sample was randomly selected out of the test set. The left axis is the frequency in kHz, the right axis is the decibel value, and the bottom axis is the duration in frames.

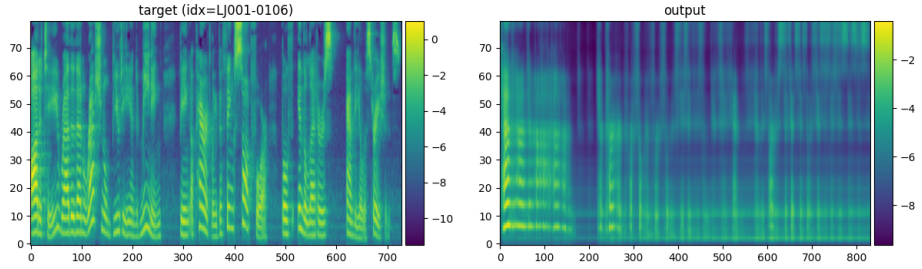

Figure A.4: Spectrogram comparison between the reference spectrogram and the synthesized spectrogram from a model with cosFormer encoder to decoder cross-attention and softmax self-attention blocks on sample LJ001-0106. This sample was randomly selected out of the test set. The left axis is the frequency in kHz, the right axis is the decibel value, and the bottom axis is the duration in frames.

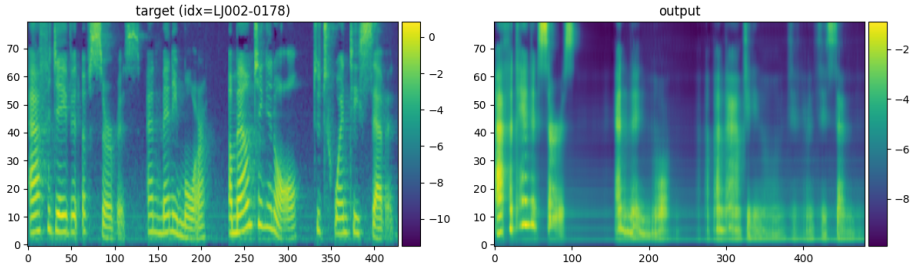

Figure A.5: Spectrogram comparison between the reference spectrogram and the synthesized spectrogram from a baseline model with only softmax attention on sample LJ002-0178. This sample was randomly selected out of the test set. The left axis is the frequency in kHz, the right axis is the decibel value, and the bottom axis is the duration in frames.

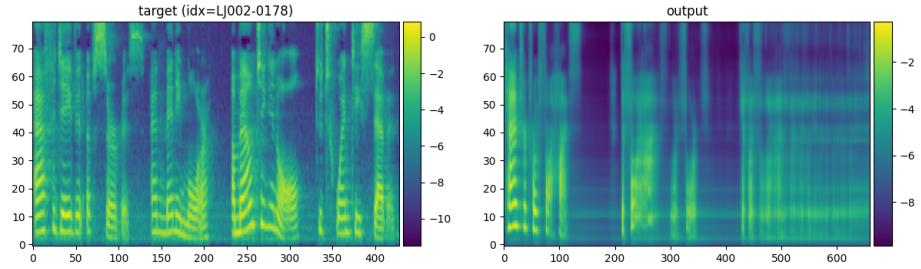

Figure A.6: Spectrogram comparison between the reference spectrogram and the synthesized spectrogram from a model with cosFormer encoder to decoder cross-attention and softmax self-attention blocks on sample LJ002-0178. This sample was randomly selected out of the test set. The left axis is the frequency in kHz, the right axis is the decibel value, and the bottom axis is the duration in frames.

Appendix B: Final Decoder Layer Cross-Attention Comparisons

A handful of samples were randomly selected for comparisons between a baseline softmax attention model and a model with its encoder to decoder cross-attention blocks replaced by a cosFormer implementation for the final layer of the decoder. To generate these scores, features were averaged across attention heads for the intermediate and normalized QK^T matrix (this was generated purely for logging during linearized inference and was not used for prediction). The general shape of this layer's cross-attention does seem to be reasonably approximated by cosFormer cross-attention, but it should be noted that the diagonal structure expected of modules that cosFormer should perform well with is not truly observed here (i.e. a token from the encoder stack's output at some relative position should score strongly with tokens in the same relative position in the decoder stack for cosFormer to perform particularly well). It should be noted that cosFormer seems to be somewhat peakier for its scores, so an initial heatmap comparison looks rather strange but common features are still very much observable. Values that correspond to scores of zero are typically padding symbols.

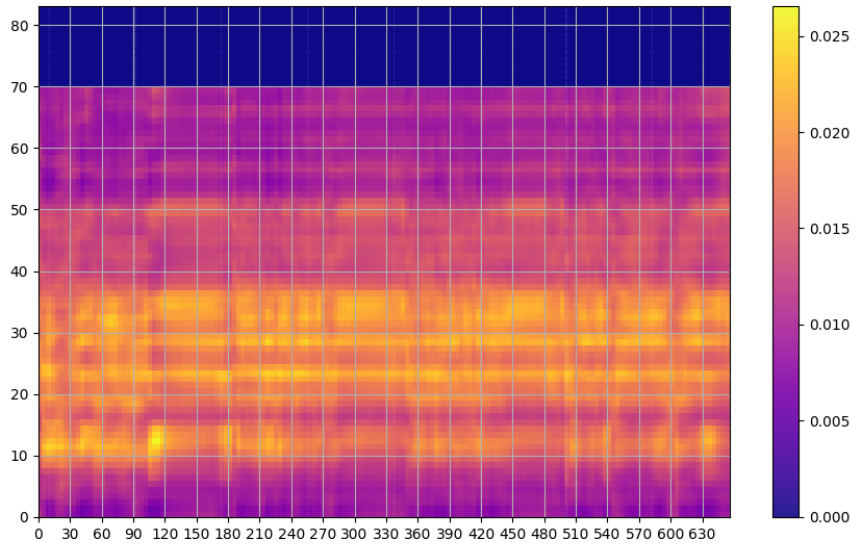

Figure B.1: Softmax QK^T intermediate matrix results for the cross-attention of the final layer of the decoder stack for sample LJ001-0030 in LJSpeech's test set. The left axis represents the source tokens in the form of phonemes and the bottom axis represents the predicted waveform tokens.

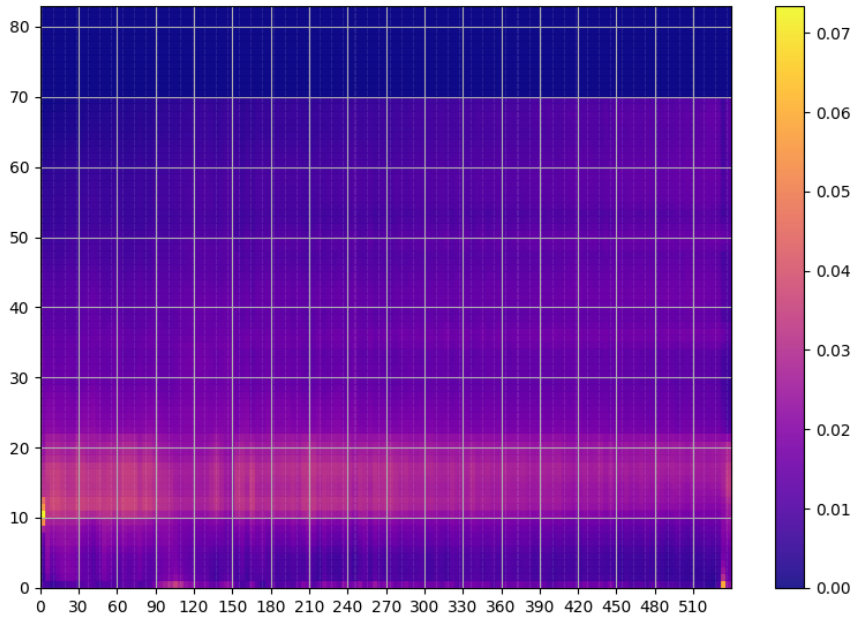

Figure B.2: cosFormer QK^T intermediate matrix results for the cross-attention of the final layer of the decoder stack for sample LJ001-0030 in LJSpeech's test set. The left axis represents the source tokens in the form of phonemes and the bottom axis represents the predicted waveform tokens.

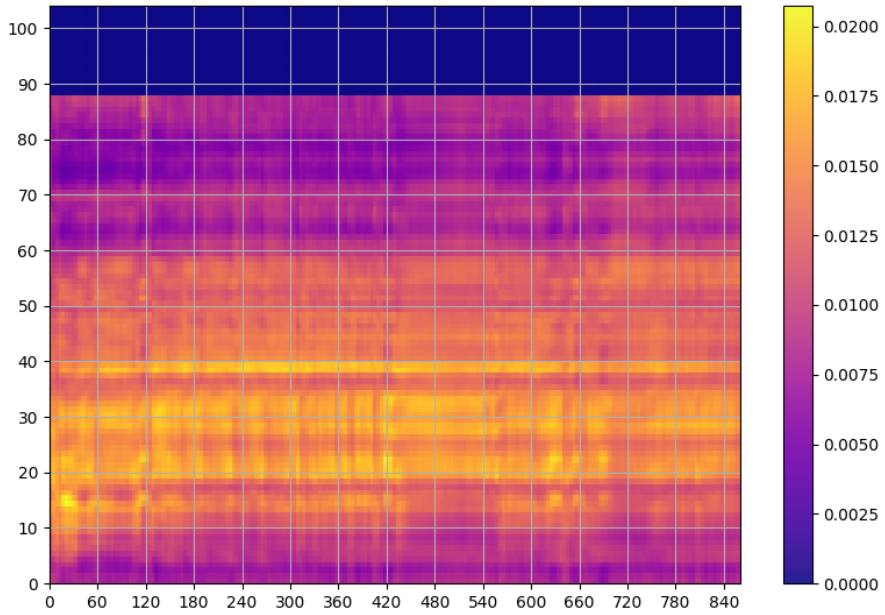

Figure B.3: Softmax QK^T intermediate matrix results for the cross-attention of the final layer of the decoder stack for sample LJ001-0106 in LJSpeech's test set. The left axis represents the source tokens in the form of phonemes and the bottom axis represents the predicted waveform tokens.

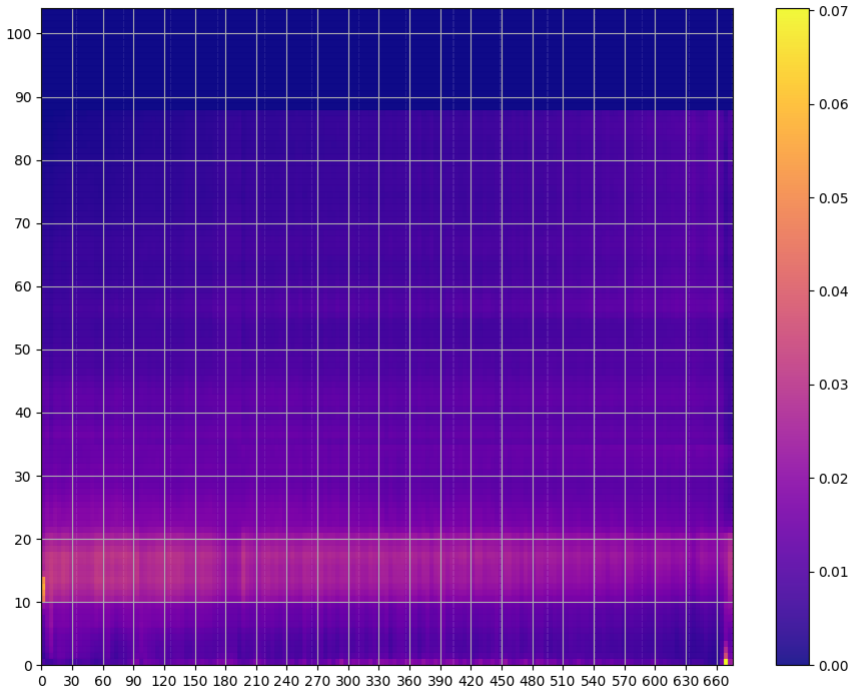

Figure B.4: cosFormer QK^T intermediate matrix results for the cross-attention of the final layer of the decoder stack for sample LJ001-0106 in LJSpeech's test set. The left axis represents the source tokens in the form of phonemes and the bottom axis represents the predicted waveform tokens.

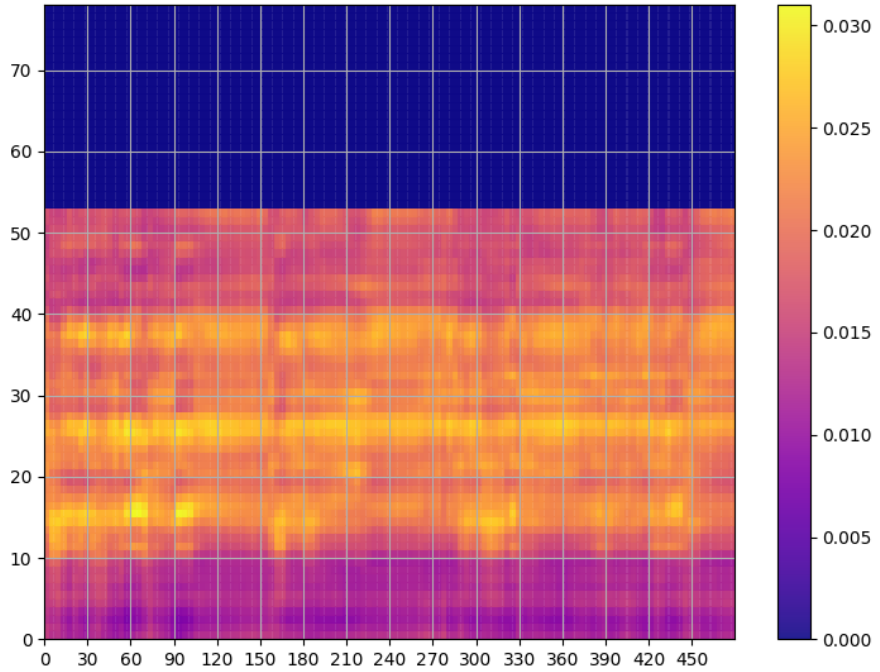

Figure B.5: Softmax QK^T intermediate matrix results for the cross-attention of the final layer of the decoder stack for sample LJ002-0178 in LJSpeech's test set. The left axis represents the source tokens in the form of phonemes and the bottom axis represents the predicted waveform tokens.

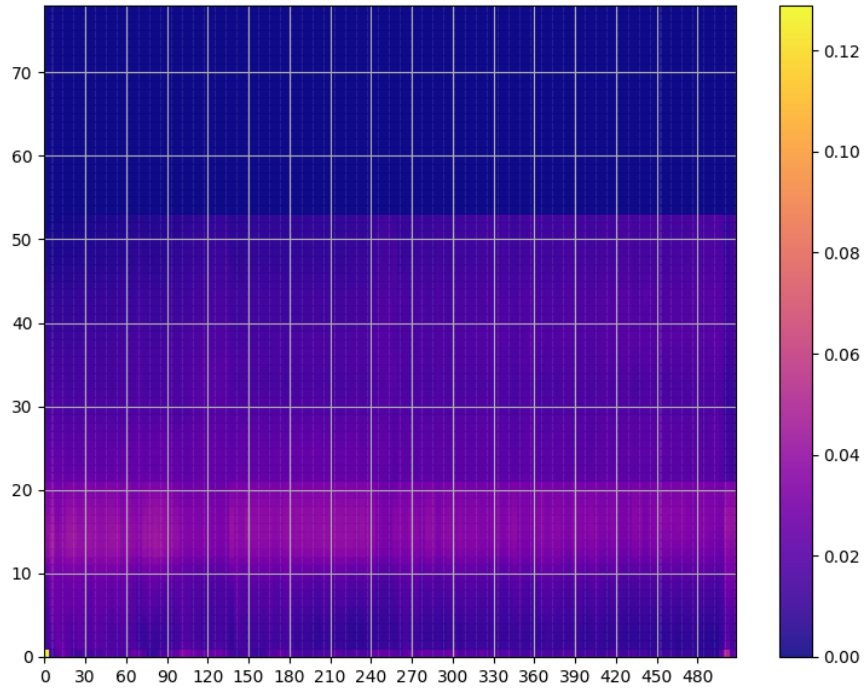

Figure B.6: cosFormer QK^T intermediate matrix results for the cross-attention of the final layer of the decoder stack for sample LJ002-0178 in LJSpeech's test set. The left axis represents the source tokens in the form of phonemes and the bottom axis represents the predicted waveform tokens.

Appendix C: Relevant Algorithms

The following algorithm is somewhat relevant to this thesis, but was not considered critical to include in its main body. Algorithm 2 provides a reference for the data-reuse implementation inspired by Katharapalous et. al's work [21], resulting in a run-time complexity that is linear with respect to the number of samples during inference. This particular algorithm is extremely generalizable and portions of it are cut away for the sake of efficiency given a particular linearization scheme.

Algorithm 2 Generalized fully linearized attention for autoregressive decoding at inference for some arbitrary decoding time-step $m+1$ for a single attention head. Note that keeping track of the transformed keys is optional, but can be useful under some specific circumstances.

Input Q in $\mathbb{R}^{1\times d}$, K in $\mathbb{R}^{N_2\times d}$, V in $\mathbb{R}^{N_2\times d}$, M_{pr} in $\mathbb{R}^{d\times d}$, S_{pr} in $\mathbb{R}^{1\times d}$, K^{tr}_{pr} in $\mathbb{R}^{N_2\times d}$
Output A in $\mathbb{R}^{1\times d}$
Require Flag f for K and V updates, chunk size k for K and V updates, decomposable linear similarity function defined by S_q and S_k, linear re-weighting scheme defined by R_q and R_k

$Q' \leftarrow S_q(Q)$
$Q^{tr} \leftarrow R_q(Q')$
if f is True **then**
  $K'_{up} \leftarrow S_k(K[-k:])$
  $K^{tr}_{up} \leftarrow R_k(K'_{up})$
  $K^{tr} \leftarrow concat(K^{tr}_{pr}, K^{tr}_{up})$
else
  $K^{tr} \leftarrow K^{tr}_{pr}$
end if

if f is True **then**
  $S_{up} \leftarrow \sum K^{trT}_{up}$
  $S \leftarrow S_{pr} + S_{up}$
else
  $S \leftarrow S_{pr}$
end if

if f is True **then**
  $M_{up} \leftarrow K^{trT}_{up} \times V[-k:]$
  $M \leftarrow M_{pr} + M_{up}$
else
  $M \leftarrow M_{pr}$
end if

$D \leftarrow Q^{tr} \times M$
$N \leftarrow Q^{tr} \times S$
$A' \leftarrow \frac{D}{N}$
$A \leftarrow out(A')$

return A

Appendix D: Synthetic Dataset for Brief Latency Comparisons

In addition to the run-time analyses for practical workloads in this thesis, below is a brief analysis of the effect of embedding dimension size and sequence length on the latency of softmax attention and various linearization schemes for decoding attention blocks and a single attention head. The data used for this analysis is entirely synthetic, generated for rote calculation to demonstrate run-time profiles with no priors. CPU and GPU-based analyses are provided below, with the CPU in question being an Intel Xeon Platinum 8168 and the GPU being a single NVIDIA Tesla V100. Batching is disabled for CPU runs and enabled for some GPU runs to showcase differences in run-time profiles for practical workloads and environments.

Generally, performances on CPUs resulted in observed latency advantages for simple ReLU attention implementations and essentially no latency advantages for cosFormer. On the side of GPUs, when batching was disabled, similar behavior was observed. Only with batching enabled (e.g. something like TTS for speeches where the entire speech is available immediately is a possible example of when batching might be enabled), does cosFormer begin to demonstrate latency advantages. This is attributed to some overhead in PyTorch's under-the-hood optimizations that can render very lightweight operations cumbersome if they are not engaging in large scale, and highly parallelizable, floating point operations.

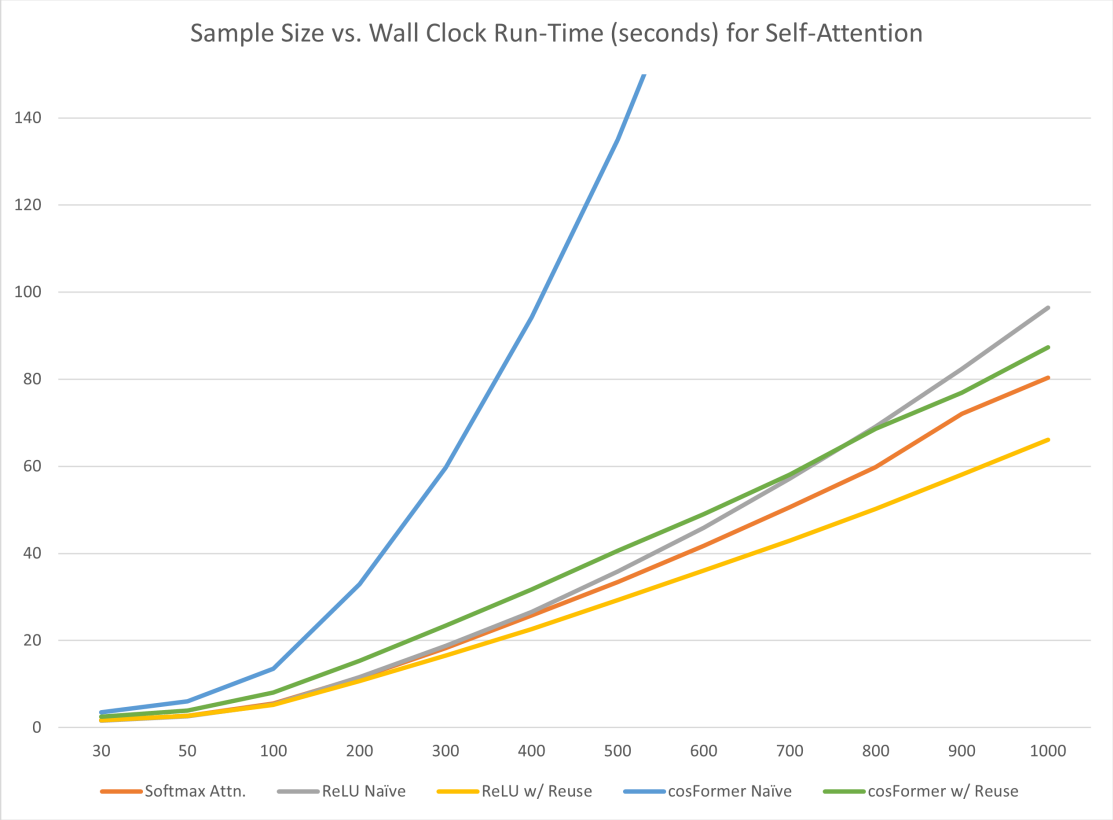

Figure D.1: Comparisons of run-time profiles for decoder self-attention for varying sample lengths on a CPU. 250 samples are present in this synthetic dataset with batching disabled. Naive linearization techniques make no attempt at reusing the K^TV intermediate matrix while reuse implies storing older information for later use. Notably, cosFormer does not beat out softmax implementations for any of the provided sample lengths.

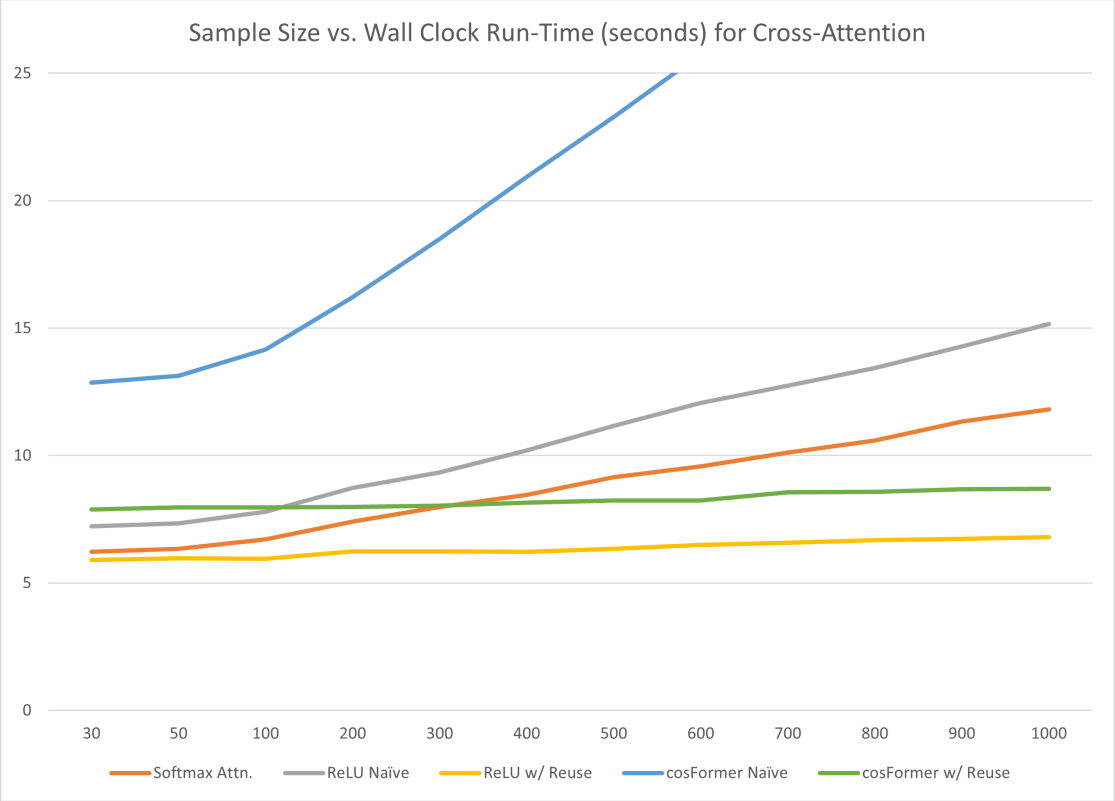

Figure D.2: Comparisons of run-time profiles for decoder cross-attention for varying sample lengths on a CPU. 250 samples are present in this synthetic dataset with batching disabled. For cross-attention, the sample length variations vary the source length (key and value sizes) while the target length is set to 150 tokens. Naive linearization techniques make no attempt at reusing the K^TV intermediate matrix while reuse implies storing older information for later use. Notably, cosFormer does not beat out softmax implementations at practical sample lengths for TTS.

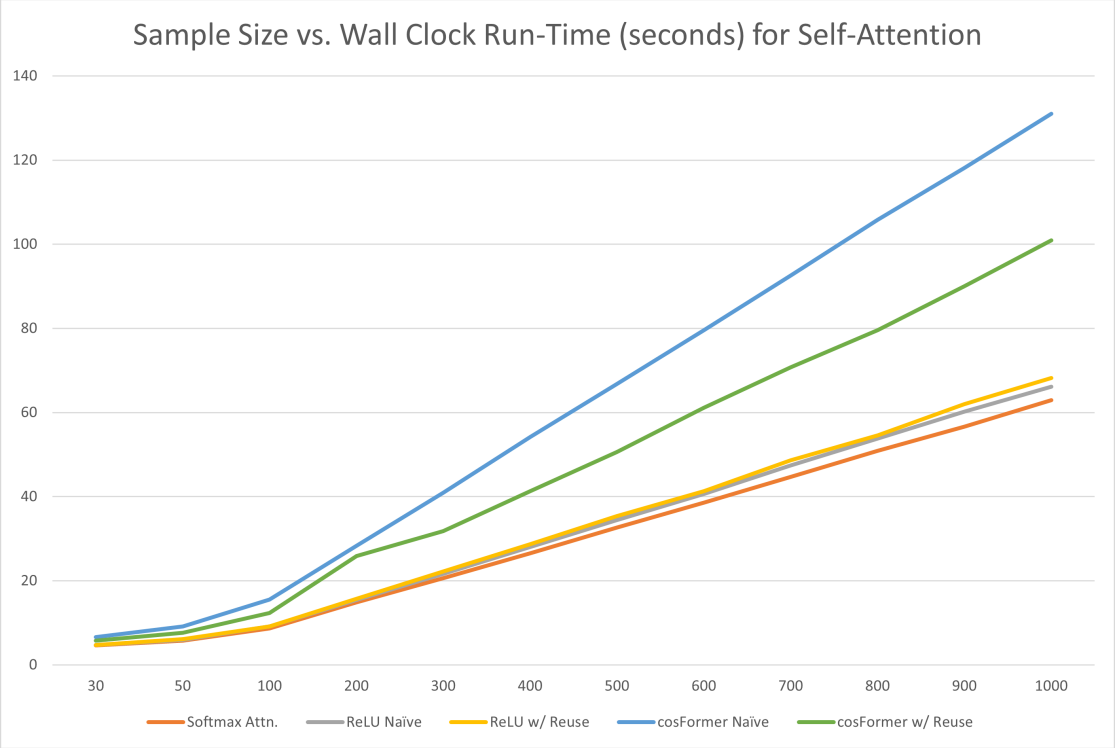

Figure D.3: Comparisons of run-time profiles for decoder self-attention for varying sample lengths on a GPU. 250 samples are present in this synthetic dataset with batching disabled. Naive linearization techniques make no attempt at reusing the $K^T V$ intermediate matrix while reuse implies storing older information for later use. Notably, cosFormer does not beat out softmax implementations for any of the provided sample lengths.

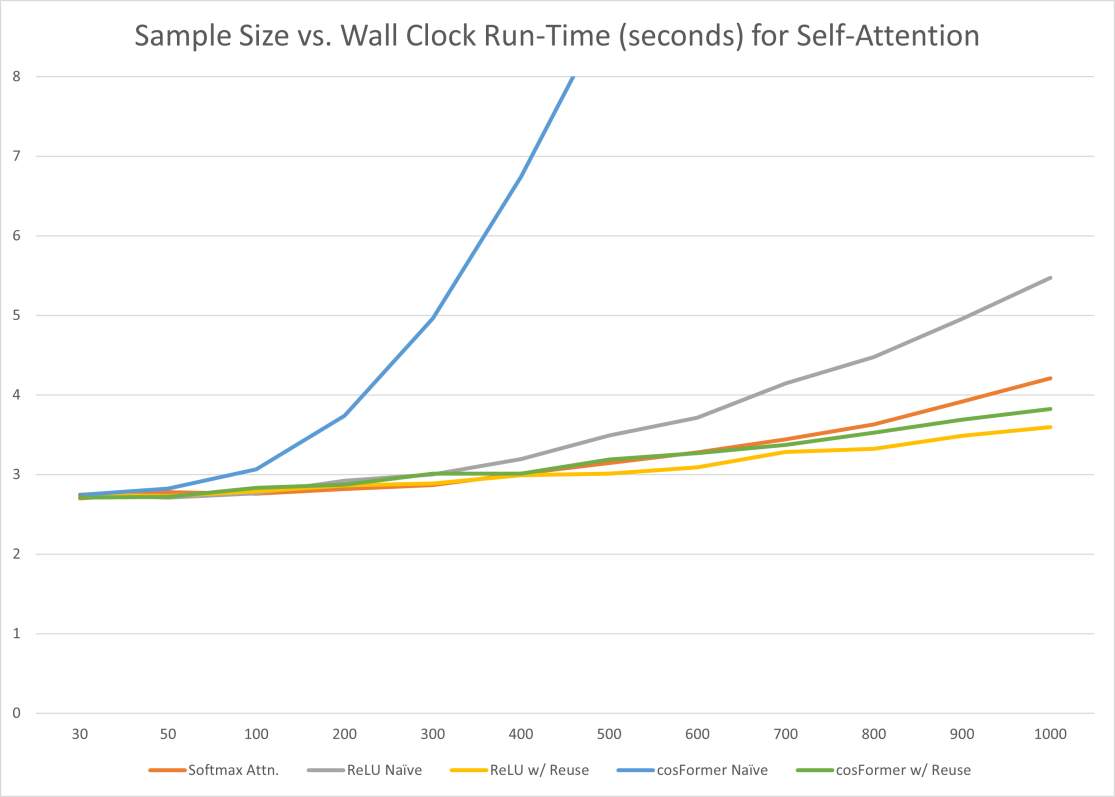

Figure D.4: Comparisons of run-time profiles for decoder self-attention for varying sample lengths on a GPU. 250 samples are present in this synthetic dataset with batches of 125. Naive linearization techniques make no attempt at reusing the $K^T V$ intermediate matrix while reuse implies storing older information for later use. Notably, cosFormer does not beat out softmax implementations at practical sample lengths for TTS.

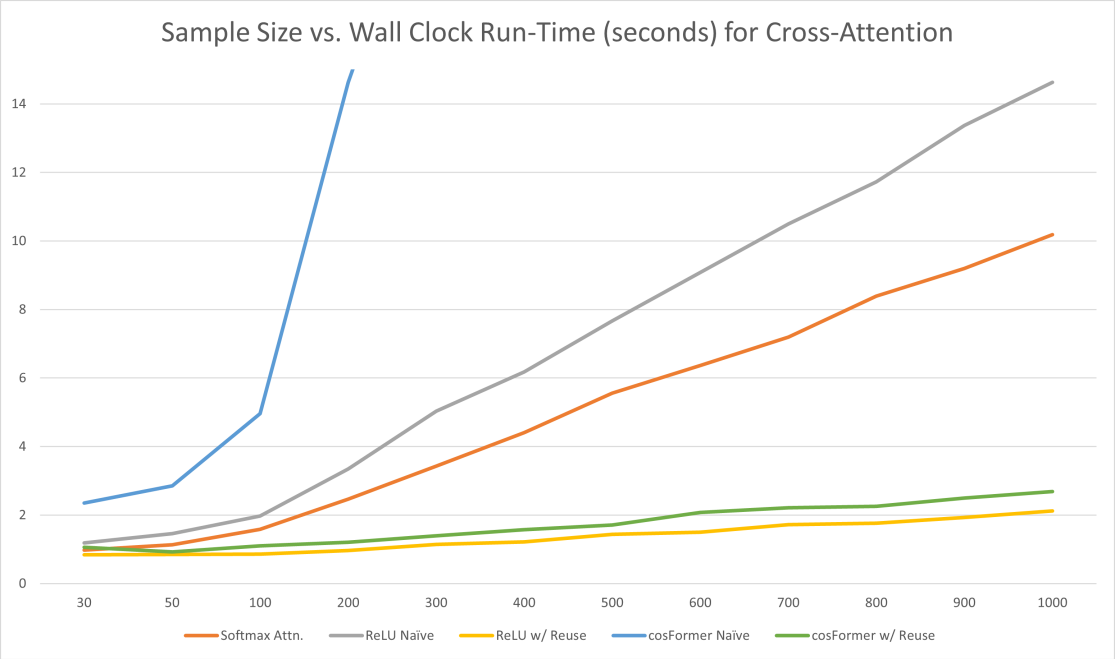

Figure D.5: Comparisons of run-time profiles for decoder cross-attention for varying sample lengths on a GPU. 250 samples are present in this synthetic dataset with batches of 125. For cross-attention, the sample length variations vary the source length (key and value sizes) while the target length is set to 150 tokens. Naive linearization techniques make no attempt at reusing the K^TV intermediate matrix while reuse implies storing older information for later use. Notably, cosFormer easily beats out softmax implementations at all relevant sample lengths.