Delta Keyword Transformer: Bringing Transformers to the Edge through Dynamically Pruned Multi-Head Self-Attention

Zuzana Jelčicová
Technical University of Denmark; and Demant A/S
Kongens Lyngby; and Smørum, Denmark
zuje@dtu.dk

Marian Verhelst
MICAS, KU Leuven
Leuven, Belgium
marian.verhelst@kuleuven.be

ABSTRACT

Multi-head self-attention forms the core of Transformer networks. However, their quadratically growing complexity with respect to the input sequence length impedes their deployment on resource-constrained edge devices. We address this challenge by proposing a dynamic pruning method, which exploits the temporal stability of data across tokens to reduce inference cost. The threshold-based method only retains significant differences between the subsequent tokens, effectively reducing the number of multiply-accumulates, as well as the internal tensor data sizes. The approach is evaluated on the Google Speech Commands Dataset for keyword spotting, and the performance is compared against the baseline Keyword Transformer. Our experiments show that we can reduce $\sim 80\%$ of operations while maintaining the original 98.4% accuracy. Moreover, a reduction of $\sim 87 - 94\%$ operations can be achieved when only degrading the accuracy by 1-4%, speeding up the MHSA inference by a factor of $\sim 7.5 - 16$.

KEYWORDS

Transformers, delta computations, pruning, compression, keyword spotting, edge devices

1 INTRODUCTION

The Transformer architecture [32] is an emerging type of neural networks that has already proven to be successful in many different areas such as natural language processing [3, 6, 21, 26], computer vision [7, 25, 31, 36], and speech recognition [4, 5, 11, 19]. Its success lies in the multi-head self-attention (MHSA), which is a collection of attention mechanisms executed in parallel. Although Transformers achieve state-of-the-art results, deployment to resource-constrained devices is challenging due to their large size and computational complexity that grows quadratically with respect to the sequence length. Hence, self-attention, despite being extremely efficient and powerful, can easily become a bottleneck in these models. A widely used compression technique to reduce the size and computations of

DNNs is pruning, that has been extensively researched throughout the years [1, 8, 13, 14]. An increasing number of works focusing on MHSA pruning recently emerge. These mainly aim for reducing the number of attention heads in each Transformer layer [22, 23, 33], and token pruning [10, 15, 16, 34]. Eliminating attention heads completely to speed up the processing might significantly impact accuracy. Therefore, token (a vector in the sequence) pruning represents a more suitable approach, where attention heads are preserved and only unnecessary tokens within the individual heads are removed. However, most of the methods above i) require demanding training procedures that hinder utilizing a single method across various models and applications without unnecessary overhead, and ii) focus on coarse-grained pruning.

In this work, we further push pruning to finer granularity, where individual features within tokens are discarded at runtime using a threshold in the MHSA pipeline. The reduction is based on the comparison of similarities between corresponding features of subsequent tokens, where only the above-threshold delta differences are stored and used for performing the multiplications (MACs). This technique significantly reduces computational complexity during inference and offers intermediate data compression opportunities. Our method does not require any training and can, therefore, be used directly in the existing pre-trained Transformer models. Moreover, no special and expensive hardware has to be developed as only comparisons are used in the algorithm. The evaluation is done on a pretrained Keyword Transformer model (KWT) [2] using the Google Speech Commands Dataset (GSCD) [35] with the focus on the accuracy-complexity trade-off. The results show that the number of computations can be reduced by $4.2x$ without losing any accuracy, and $7.5x$ while sacrificing 1% of the baseline accuracy. Furthermore, the processing of the original MHSA block can be sped up by a factor of ~ 16 while still achieving high accuracy of $\sim 95\%$. Therefore, this work represents the next step to enable efficient inference of Transformers in low-power edge devices with the tinyML constraints.

2 RELATED WORK

Different approaches have been used to reduce the computational complexity of the MHSA, such as cross-layer parameter sharing [17], trimming individual weights [9] or removing encoders by distillation [20, 27–30]. Recent research [18, 22, 23, 33] demonstrates that some attention heads can be eliminated without degrading the performance significantly. However, in order to obtain substantial computational savings and thus inference time gains, a considerable portion of heads would have to be discarded, inevitably leading to noticeable accuracy drops.

Other works focus on token pruning instead of removing redundant

parameters. In [10], redundant word-vectors are eliminated, outperforming previous distillation [27, 28] and head-pruning methods [23]. However, it requires training of a separate model for each efficiency constraint. This issue is resolved in [15] by adopting one-shot training that can be used for various inference scenarios, but the training process is complicated and involves multiple steps. Cascade pruning on both the tokens and heads is applied in [34], i.e., once a token and/or head is pruned, it is removed in all following layers. Nonetheless, this approach requires sorting of tokens and heads depending on their importance dynamically to select the top-k candidates, which needs specialized hardware. Similar to our work, recently published [16] also adopts a threshold-based pruning approach, which removes unimportant tokens as the input passes through the Transformer layers. However, this method requires a three-step training procedure to obtain a per-layer learned threshold, which again prevents to easily deploy the technique across a wide range of pre-trained networks. Most of the previous methods, moreover, only focus on optimizing Transformers for the natural language processing task.

The idea of threshold-based pruning using delta values for performing computations has already been explored for other types of DNNs, such as recurrent [24] and convolutional [12] neural networks. However, incorporating a delta threshold in these networks results in significant memory overhead, as it requires storing intermediate states and activations. This issue is eliminated in our Delta Transformer, where almost no additional resources are required.

3 THE KEYWORD TRANSFORMER

The typical Transformer encoder [32] adopted in KWT consists of a stack of several identical Transformer blocks. Each Transformer block comprises of Multi-Head Self-Attention (MHSA), Multi-Layer Perceptron (MLP), layer normalizations, and residual connections as illustrated in Figure 1. The key component in Transformers is the MHSA containing several attention mechanisms (heads) that can attend to different parts of the inputs in parallel. We base our explanation on the KWT, proposed in [2]. This model takes as an input the MFCC spectrogram of T non-overlapping patches $X_t \in R^F$, with $t = 1, ..., T$ and $f = 1, ..., F$ corresponding to time windows and frequencies, respectively. This input is first mapped to a higher dimension d using a linear projection matrix $W_0 \in R^{F \times d}$ along the frequency dimension, resulting in T tokens of dimension d. These are then concatenated with a learnable class embedding token $X_{CE} \in R^{1 \times d}$ representing a global feature for the spectrogram. Subsequently, a learnable positional embedding $X_{PE} \in R^{(T+1) \times d}$ is added to form a final input to the Transformer encoder:

$$X = [X_{CE}; X_{MFCC}W_0] + X_{PE} \tag{1}$$

The Transformer encoder multiplies the input X with the projection matrices W_Q, W_K, $W_V \in R^{d \times d}$, producing Query (Q), Key (K), and Value (V) input embedding matrices:

$$Q = XW_Q; \quad K = XW_K; \quad V = XW_V \tag{2}$$

The matrices are then divided into k attention heads to perform the self-attention computations in parallel, where each of the heads $i = 1, 2, .., k$ is given by:

$$head_i = Attention(Q_i, K_i, V_i) = softmax\left(\frac{Q_i (K_i)^T}{\sqrt{d_h}}\right) V_i \tag{3}$$
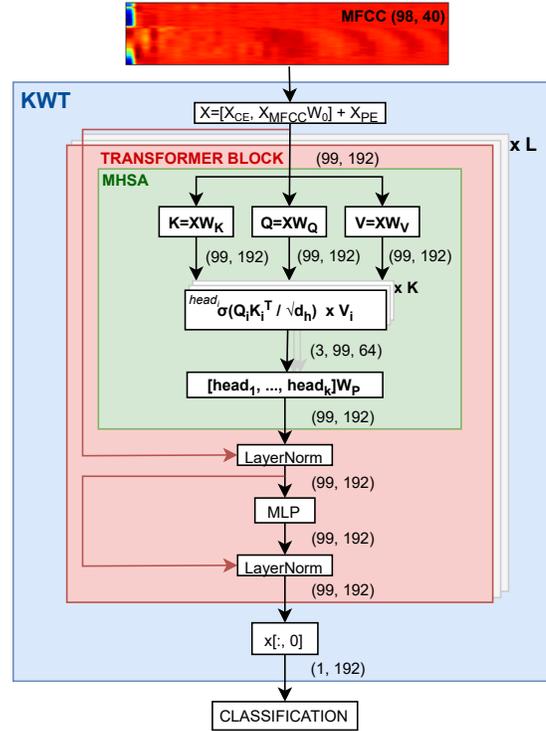

Figure 1: A high-level overview of the KWT model along with its dimensions. Red lines denote the residual connections.

The MHSA is defined as a concatenation of the attention heads, weighted by a projection matrix $W_P \in R^{d \times d}$, where $d_h = d/k$:

$$MHSA(Q, K, V) = [head_1, head_2, ..., head_k] W_P \tag{4}$$

The MHSA output is then added to the input X with a residual connection and passed though the first layer normalization and the MLP block, followed by another addition of a residual input and second normalization:

$$X_{l1} = LN(X + MHSA); \quad X_{l2} = LN(X_{l1} + MLP(X_{l1})) \tag{5}$$

This structure is repeated L times, denoting layers, to create an architecture of stacked Transformer layers.

In the KWT model, the MLP block is a two-layer feed-forward neural network using a GELU activation function after the first layer. The class embedding vector is extracted from the output of the last Transformer block to perform classification.

Three KWT models are proposed in the original work: KWT-1 (607k parameters, 97.72% ± 0.01 accuracy), KWT-2 (2,394k parameters, 98.21% ± 0.06 accuracy), and KWT-3 (5,361k parameters, 98.54% ± 0.17 accuracy). We selected KWT-3 for our experiments, as it poses the biggest challenge as well as potential for compressing and reducing the computational complexity. The KWT-3 configuration is listed in Table 1.

Table 1: Configuration of the KWT-3 architecture

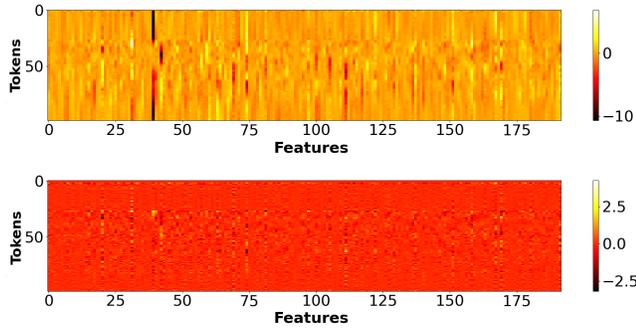| Model	dim d	dim d_{mlp}	heads k	layers L	#params
KWT-3	192	768	3	12	5,361k

Figure 2: Input data to the 7th Transformer layer at the top along with its delta version at the bottom for keyword right.
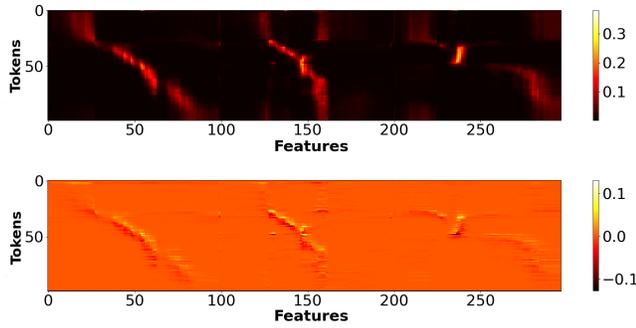

Figure 3: Softmax output of the 7th Transformer layer at the top along with its delta version at the bottom for the keyword right. **The figure illustrates three attention heads.**

4   KWT MODEL ANALYSIS

The attention mechanism involves MACs of two matrices, resulting in $O(n^2)$ time and space complexity. However, as all tokens attend to each other, a certain level of redundancy is expected to be found in the system due to diffusion of information. Therefore, we analyze the KWT model on the GSCD to observe the degree of change across the tokens as they pass though the MHSA. We feed multiple different keywords through the 12-layer KWT and inspect the MHSA inputs as well as intermediate results within the block. While considerable correlation across the tokens is expected for the initial input and intermediate results in the first layer, it is noteworthy to observe such behavior also in the MHSA of deeper layers, which is in line with cosine similarity measurements on word-vectors performed in [10]. Correlation is illustrated in Figure 2 showing the input X (top) together with the difference between subsequent rows of this tensor (bottom), for the 7th layer of a keyword *right*. Figure 3 repeats the same analysis for the softmax output of layer 7. It is clear that there is a significant amount of correlation between consecutive tokens, which opens up opportunities for data compression and/or computational data reuse. For example, $\sim 84\%$ of the differences between corresponding features of subsequent tokens in X are smaller than 1% of the dynamic range of X (7th layer). Such a tendency was observed for all voice-containing input sequences. Moreover, when analyzing intermediate tensors from
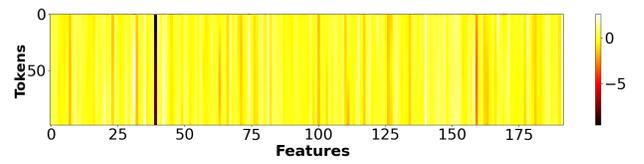

Figure 4: Input data to the 7th Transformer layer for _silence_.

inputs of the *_silence_* class, even larger data redundancy can be observed (Figure 4). It is clear that fully computing every single token would be a waste of computational and memory resources. All these observations demonstrate that the amount of a significant change across the tokens constitutes only a small portion of the whole. Hence, introducing a threshold for recomputing could drastically decrease the computational load and inference time. Furthermore, exploiting sparsity across the tokens can also offer data compression. Therefore, we propose a delta algorithm that utilizes a threshold to discard insignificant values, further described in Section 5.

5   DELTA ALGORITHM

The objective of the delta algorithm is to transform a dense matrix-vector multiplication into a highly-sparse matrix-vector multiplication to reduce computational complexity and enable data compression, where only non-zero deltas are stored and used for computations.

The input X always starts with the class embedding vector, followed by the first input vector. These two vectors (rows of the tensors) will always be left untouched throughout the complete MHSA pipeline. Every subsequent token after these will be represented by its delta value. This delta change $X(t)$ is calculated as the difference between the current input $X(t)$ and reference vector $\hat{X}(t-1)$. Only delta differences larger than a threshold θ are retained and used to update the reference vector $\hat{X}(t)$:

$$\Delta X(t) = \begin{cases} X(t) - \hat{X}(t-1) & \text{if } |X(t) - \hat{X}(t-1)| > \theta \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

$$\hat{X}(t) = \begin{cases} X(t) & \text{if } |X(t) - \hat{X}(t-1)| > \theta \\ \hat{X}(t-1) & \text{otherwise} \end{cases} \quad (7)$$

Where the $\hat{X}$ vector is initialized to 0s and updated once the first token arrives. Figure 5 visualizes this encoding over three tokens with $\theta = 1.0$. The top row represents the first input vector that is left untouched (no delta algorithm applied). The orange and green colors in $\hat{X}$ show which values from the current input X are propagated for the next token. White X positions denote values of which magnitude equals to/is below θ and thus are skipped.

 We apply the delta encoding of data at six different places in the MHSA: layer input X, matrices K and Q, scaled $QK^{)}$, softmax output, and the attention head output. While the computations of delta values are the same everywhere, the subsequent operations with these deltas differ depending on whether i) a delta-encoded matrix is multiplied with a regular matrix, ii) two delta-encoded matrices are multiplied together, or iii) a non-linear *softmax* function is applied. These three versions are described in the next subsections.

Figure 5: Delta algorithm example across three tokens with threshold θ = 1.0. The top row corresponds to the first input vector that is always left untouched (no threshold).

Figure 6: Baseline delta algorithm.

5.1 Delta-regular matrix multiplication

Thanks to the delta representation, only non-zero ΔX are stored and used for multiplications as visualized in Figure 6. A weight matrix is denoted as W, and indices for Δxw in the result matrix R are excluded for clarity. The output $R(t)$ of the tensor operation can hence be computed by accumulating the result of the previous reference token $R(t-1)$ with the multiplication results of the weights with the delta values only. The updated $R(t)$ will then be the new baseline for the upcoming token:

$$R'(t) = R(t), + R'(t-1) \tag{8}$$

With $R(0)$ initialized to 0. These delta multiplications are used in $XW_Q, XW_K, XW_V, softmaxV$ and $[head_1, head_2, head_3]W_O$.

5.2 Delta-delta matrix multiplication

As a result of the delta encoding, both Q and K will be expressed in their delta versions, and the multiplications will thus be slightly modified. This is described below and illustrated in Figure 7 in a general form, with matrices A and B representing Q and K^T, respectively. The multiplication of the first A row with the first B column is done as usually without using deltas:

$$r_{00} = a_{00}b_{00} + a_{01}b_{10} + a_{02}b_{20} \tag{9}$$

Then, the multiplication of the first A row and second B column exploits the delta approach in horizontal direction, where the $a_{00}b_{00} + a_{01}b_{10} + a_{02}b_{20}$ expression can be replaced with r_{00} from eq. 9 (marked with red):

$$r_{01} = a_{00}(b_{00} + \Delta b_{01}) + a_{01}(b_{10} + \Delta b_{11}) + a_{02}(b_{20} + \Delta b_{21}) \tag{10}$$
$$= a_{00}b_{00} + a_{00}\Delta b_{01} + a_{01}b_{10} + a_{01}\Delta b_{11} + a_{02}b_{20} + a_{02}\Delta b_{21}$$
$$= r_{00} + a_{00}\Delta b_{01} + a_{01}\Delta b_{11} + a_{02}\Delta b_{21}$$

Figure 7: Delta algorithm for QK^T represented with matrices A and B.

Similarly, calculating results in the vertical direction for the rows of A and first column of B is given by:

$$r_{10} = b_{00}(a_{00} + \Delta a_{10}) + b_{10}(a_{01} + \Delta a_{11}) + b_{20}(a_{02} + \Delta a_{12}) \tag{11}$$
$$= b_{00}a_{00} + b_{00}\Delta a_{10} + b_{10}a_{01} + b_{10}\Delta a_{11} + b_{20}a_{02} + b_{20}\Delta a_{12}$$
$$= r_{00} + \Delta a_{10}b_{00} + \Delta a_{11}b_{10} + \Delta a_{12}b_{20}$$

An approach for multiplications for all the other positions is demonstrated on the second A row and second B column:

$$r_{11} = (a_{00} + \Delta a_{10})(b_{00} + \Delta b_{01}) + (a_{01} + \Delta a_{11})(b_{10} + \Delta b_{11}) \tag{12}$$
$$+ (a_{02} + \Delta a_{12})(b_{20} + \Delta b_{21})$$
$$= a_{00}b_{00} + a_{00}\Delta b_{01} + \Delta a_{10}b_{00} + \Delta a_{10}\Delta b_{01}$$
$$+ a_{01}b_{10} + a_{01}\Delta b_{11} + \Delta a_{11}b_{10} + \Delta a_{11}\Delta b_{11}$$
$$+ a_{02}b_{20} + a_{02}\Delta b_{21} + \Delta a_{12}b_{20} + \Delta a_{12}\Delta b_{21}$$
$$= r_{01} + r_{10} - r_{00} + \Delta a_{10}\Delta b_{01} + \Delta a_{11}\Delta b_{11} + \Delta a_{12}\Delta b_{21}$$

Where different colors mark each of the three multiplications. Simplifying parenthesis shows that the expressions not involving any deltas can be substituted with r_{00}. Next, the terms with Δb are replaced with r_{01}, while those containing Δa with r_{10}. Since r_{00}, r_{01}, and r_{10} have already been computed in previous timesteps, we only need to do the (sparse) delta multiplications themselves and subtract the r_{00} result as it is present in both r_{01} and r_{10}. These steps are then applied to all the other slots as shown in Figure 7.

5.3 Delta for softmax

Delta algorithm cannot be directly applied for softmax as this function introduces a non-linearity to the system:

$$softmax(A)_i = \frac{exp(A_i)}{\sum_j exp(A_j)} \tag{13}$$

We will have to introduce a scaling factor to correct the softmax computations. As done earlier, we will again start by performing unaltered processing of the initial row $r_0 = [r_{00} \; r_{01} \; r_{02}]$ (class embedding excluded for clarity) with a regular softmax function:

$$softmax(A)_0 = \frac{[exp(A_{00}) \; exp(A_{01}) \; exp(A_{02})]}{[exp(A_{00}) \; exp(A_{01}) \; exp(A_{02})]} \tag{14}$$

The next row of the scaled input QK^T is already expressed with deltas:

$$A_1 = [\; \Delta A_{10} \; \Delta A_{11} \; \Delta A_{12}] \tag{15}$$

The r_1 nominator NOM_{A_1} for softmax is thus given by:

$$NOM_{A_1} = [\exp(A_{10})\ \exp(A_{11})\ \exp(A_{12})] \tag{16}$$

While the denominator $DENOM_{A_1}$ as:

$$DENOM_{A_1} = \frac{[\exp(A_{00} + A_{10})\ \exp(A_{01} + A_{11})\ \exp(A_{02} + A_{12})]}{[\exp(A_{00})\ \exp(A_{01})\ \exp(A_{02})]} \tag{17}$$

Finally, a scaling factor for each of the values to correct the softmax result is:

$$\sigma_{A_1} = softmax(A)_1 \cdot \frac{NOM_{A_1}}{DENOM_{A_1}} \tag{18}$$

5.4 Computational savings

To assess the potential computational savings for the Delta KWT, we differentiate between the two main sublayers: i) MHSA, and ii) MLP. The MLP block consists of two fully connected layers with weight matrices of dimensions (192,768) and (768,192), respectively. Without any delta modification, ~39% of the multiplication of the original KWT can be found in the MHSA and ~61% in the MLP. Although MLP is the prevailing module in this specific scenario, its complexity does not grow quadratically with the input sequence length. Moreover, there are many well-established compression techniques available, some of them presented in Section 2. Hence, pruning of the MLP is out of the scope of our work, and it is only stated for completeness. The MHSA multiplication operations can be further split into XW_q, XW_k, XW_v (~59.63%), QK^T (~10.25%), $softmax(QK^T)V$ (~10.25%), and final projection with attention heads $[head_1, head_2, head_3]W_o$ (~19.88%). The KWT model offers an optimization in the last layer. As shown in Figure 1, only the class embedding token is used for the final prediction, making the rest of the tokens within the sequence unused. This dependency can be tracked up to QK^T. The MAC savings in last layer are thus worth 59.64%, always making the total savings at least 4.97% for the whole KWT without losing any accuracy.

Maximum possible computational savings, i.e., cases when only the class embedding and first vector are computed since all deltas are 0, are stated below for each of the MHSA parts. For simplicity, all the terms use matrices A and B, and row and col for dimensions. Savings for XW_q, XW_k, and XW_v for each of the first 11 layers are:

$$\gamma_{0-10} = 1 - \frac{(row_A - 1)\ col_A\ col_B}{row_A\ col_A\ col_B} =\sim 97.98\% \tag{19}$$

Where $A = (99, 192)$ and $B = (192, 192)$. Computations for XW_k in the last layer are expressed as:

$$\gamma_{11} = 1 - \frac{(row_A - 1)\ col_A\ col_B + row_A\ col_A\ col_B}{row_A\ col_A\ col_B} =\sim 98.32\% \tag{20}$$

Savings for QK^T:

$$\gamma_{0-10} = 1 - \frac{(row_A - 1)\ col_A col_B\ head}{row_A\ col_A\ col_B\ head} =\sim 99.96\% \tag{21}$$

$$\gamma_{11} = 1 - \frac{(row_A - 1)\ col_A\ head}{row_A\ col_A\ col_B\ head} =\sim 99.98\% \tag{22}$$

Where $A = (99, 64)$ and $B = (64, 99)$. Savings for $softmax(QK^T)V$:

$$\gamma_{0-10} = 1 - \frac{(row_A - 1)\ col_A col_B)\ head}{row_A\ col_A\ col_B\ head} =\sim 97.98\% \tag{23}$$

$$\gamma_{11} = 1 - \frac{(row_A - 1)\ col_A)\ head}{row_A\ col_A\ col_B\ head} =\sim 98.99\% \tag{24}$$

Where $A = (99, 99)$ and $B = (99, 64)$. Finally, the projection with attention heads:

$$\gamma_{0-10} = 1 - \frac{(row_A - 1)\ col_A\ col_B}{row_A\ col_A\ col_B} =\sim 97.98\% \tag{25}$$

$$\gamma_{11} = 1 - \frac{row_A\ col_A}{row_A\ col_A\ col_B} =\sim 98.99\% \tag{26}$$

Where $A = (99, 192)$ and $B = (192, 192)$.

Of course, the savings estimated above only hold for the extreme case, which means that either a) all tokens are perfectly correlated, or b) very large thresholds are used, resulting in significant accuracy degradation. Section 7 will therefore analyze the complete accuracy-complexity trade-off for real data sequences.

5.5 Resources

The proposed delta approach neither requires expensive hardware nor comes with a large memory overhead. Only a single token has to be stored as a reference whenever the delta method is used. The softmax delta version additionally needs to keep the sum of exp from one timestep to another. In terms of computations, an additional division is needed when calculating scaling factors, along with multiplications with scaling factors for features within a token. The downside of our method is compute and data irregularity due to the algorithm's *unstructured* pruning. However, there are many techniques proposed in literature such as [37] on how to handle this challenge.

6 EXPERIMENTAL SETUP

The GSCD v2 [35] is used to evaluate our method as well as the original KWT performance. The dataset contains 105,000 1-second audio snippets of 35 different words sampled at 16 kHz. The model classifies 4,800 keywords from a test set into one of the 12 categories: "up", "down", "left", "right", "yes", "no", "on", "off", "go", and "stop", "_silence_" and "_unknown_".

To assess the impact of the thresholds for the different parts of the MHSA on accuracy and model complexity, we executed threshold sweeps on a subset of 100 keywords (6-12 words from each category). While the thresholds might be different for each delta encoding within the MHSA block, they are the same across every Transformer layer. This means that MHSA in the first layer uses the same thresholds as MHSAs in other layers. From these sweeps, the thresholds leading to a Pareto-optimal accuracy-computations trade-off are used in a full run with all 4,800 keywords. We focused on those configurations that yielded at least 94% accuracy. Since the thresholds are first determined on a subset of the complete dataset, it was expected to obtain variations in the results when performing the test on the full dataset. Additional finetuning, i.e., threshold adjusting, was done and the results are presented and discussed in Section 7.

7 RESULTS AND DISCUSSION

The Pareto-optimal results evaluated on all 4,800 audio files are shown in Figure 8, where the delta configurations are provided in the legend. The x- and the left y-axis show a percentage of executed MACs averaged across the layers and achieved accuracy, respectively. The second y-axis represents a speedup factor derived from the amount of MACs. The blue circle corresponds to the original

(a) Delta inputs

(b) Delta softmax outputs (three attention heads)

Figure 9: Deltas for a) inputs and b) softmax outputs for the 7th Transformer layer of the keyword right. Black color marks 0s.

Figure 10: Deltas for inputs to the 7th Transformer layer for _silence_. Black color marks 0s.

of the input data X and the softmax output of the 7th layer of a keyword *right* (same instance as used in Table 2). One special case are the instances from the _silence_ class, that have the amount of discarded computations very close to the theoretical maximum defined in Section 5.4. Figure 10 shows the _silence_ input, for which only a small fraction of the deltas are non-zero, resulting in $97 - 99.9\%$ of skipped operations.

A potential future improvement involves applying deltas on the input embedding matrix V. Although these cannot be exploited in multiplications with the softmax output due to the direction of computations (softmax output compensates for it), it would still contribute to V's data compression. Future work also explores the most optimal thresholds for each of the layers individually. This might further optimize the point where the accuracy starts dropping since a varying number of MACs is executed within each of the 12 layers.

8 CONCLUSION

This paper introduced a dynamic threshold-based pruning technique that drastically reduces MAC operations during inference. It was demonstrated on a keyword spotting task on the GSCD, where $\sim 80\%$ of operations in the MHSA can be discarded without degrading the accuracy. If the accuracy requirements can be slightly relaxed, a speedup factor of $\sim 5 - 16x$ is achieved. Our method thus helps to considerably decrease the computational complexity and enable significant data compression. The proposed technique can be exploited to enable an ultra-low power wake-up word detection front-end, that triggers a more powerful detector once a keyword is recognized. More generally, this work represents a stepping stone towards enabling the execution of Transformers on low-power devices.
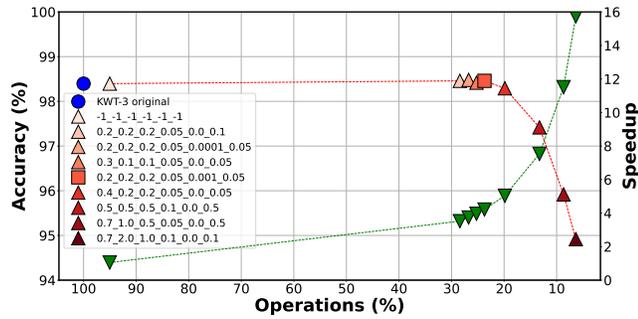

Figure 8: The results of running the original and the delta version of the KWT model. X-axis represents MACs, while the left and right y-axis correspond to the accuracy and speedup, respectively. Each of the red-shaded triangles (and a square) in the legend is annotated with thresholds used during the experiment in order: θ_X, θ_Q, θ_K, θ_{QK^T}, $\theta_{so\ max}$, and $\theta_{head_{1\cdots k}}$

Table 2: Percentage of executed MACs averaged across the layers for one instance of each keyword category. The configuration is: $\theta_X = 0.2$, $\theta_Q = 0.2$, $\theta_K = 0.2$, $\theta_{QK^T} = 0.05$, $\theta_{so\ max} = 0.001$, and $\theta_{head_{1\cdots k}} = 0.05$.

Keyword	$XW_{Q\cdot K\cdot V}$	QK^T	$so\ maxV$	$head_{1\cdots k}W_P$	Total
silence	3.02	0.08	2.4	2.02	2.46
unknown	35.97	7.68	26.95	16.93	28.36
yes	31.82	6.26	24.06	16.31	25.32
no	36.98	8.93	25.42	14.27	28.41
up	33.48	6.28	23.64	13.32	25.68
down	29.88	5.08	22.38	14.24	23.46
left	38.73	10.09	26.95	16.97	30.26
right	33.52	6.68	25.65	16.55	26.59
on	31.13	5.64	21.21	13.02	23.9
off	39.5	10.17	26.68	15.11	30.33
stop	32.39	6.15	23.36	13.72	25.06
go	33.37	6.57	23.26	14.65	25.87

KWT-3 model that achieves $\sim 98.4\%$ accuracy with 100% MACs. The red and green triangles represent our delta KWT-3 model with regard to accuracy and speedup, respectively. The inference time gains for the MHSA range from $\sim 1.05x$ to $\sim 16x$, and there is no accuracy degradation down to $\sim 23.7\%$ MACs (4.2x speedup). Moreover, some of the configurations even slightly outperform the original KWT-3 (98.46%, 98.48%, and 98.42%). Decreasing the accuracy by only 0.1% results in further speedup of $5x$. Moreover, if the accuracy requirements can be relaxed by 1-4%, the MHSA inference becomes faster by $7.5 - 15.7x$, which translates to 86.73-93.65% of skipped MACs. Table 2 shows the % of executed MHSA operations for one instance of each keyword category, averaged across the layers. The configuration (0.2_0.2_0.2_0.05_0.001_0.05) used to obtain the results is represented with a square in Figure 8. Although the MAC percentage naturally fluctuates for keywords within the same group, the objective of the table is to provide a general overview of how much operations are approximately performed in each of the parts. We can observe that $\sim 60 - 70\%$ of $XW_{Q\cdot K\cdot V}$, $90 - 95\%$ of QK^T, $73 - 79\%$ of $softmaxV$, and $83 - 87\%$ of $head_{1\cdots k}W_P$ are discarded, which sums up to $70 - 77\%$ of skipped operations for the entire model. To visualize the savings, Figure 9 shows the delta values
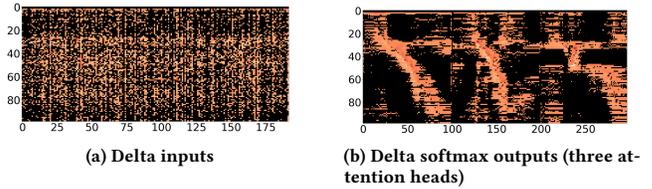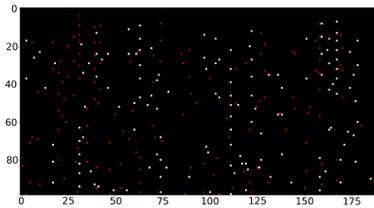
REFERENCES

[1] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. 2017. Structured Pruning of Deep Convolutional Neural Networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 13, 3 (2017), 1–18. https://doi.org/10.1145/3005348

[2] Axel Berg, Mark O'Connor, and Miguel Tairum Cruz. 2021. Keyword Transformer: A Self-Attention Model for Keyword Spotting. In *Proc. Interspeech 2021*. 4249–4253. https://doi.org/10.21437/Interspeech.2021-1286

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems, NeurIPS 2020*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901. https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[4] Feng-Ju Chang, Martin Radfar, Athanasios Mouchtaris, Brian King, and Siegfried Kunzmann. 2021. End-to-End Multi-Channel Transformer for Speech Recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021*. IEEE, 5884–5888. https://doi.org/10.1109/ICASSP39728.2021.9414123

[5] Xie Chen, Yu Wu, Zhenghao Wang, Shujie Liu, and Jinyu Li. 2021. Developing Real-Time Streaming Transformer Transducer for Speech Recognition on Large-Scale Dataset. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021*. IEEE, 5904–5908. https://doi.org/10.1109/ICASSP39728.2021.9413535

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, 4171–4186. https://doi.org/10.18653/v1/n19-1423

[7] 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. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *9th International Conference on Learning Representations, ICLR 2021, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=YicbFdNTTy

[8] Jonathan Frankle and Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=rJl-b3RcF7

[9] Mitchell Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing BERT: Studying the Effects of Weight Pruning on Transfer Learning. In *Proceedings of the 5th Workshop on Representation Learning for NLP*. Association for Computational Linguistics, Online, 143–155. https://doi.org/10.18653/v1/2020.repl4nlp-1.18

[10] Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan T. Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. PoWER-BERT: Accelerating BERT Inference via Progressive Word-vector Elimination. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 3690–3699. http://proceedings.mlr.press/v119/goyal20a.html

[11] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. 2020. Conformer: Convolution-augmented Transformer for Speech Recognition. In *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, Helen Meng, Bo Xu, and Thomas Fang Zheng (Eds.). ISCA, 5036–5040. https://doi.org/10.21437/Interspeech.2020-3015

[12] Amirhossein Habibian, Davide Abati, Taco S. Cohen, and Babak Ehteshami Bejnordi. 2021. Skip-Convolutions for Efficient Video Processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 2695–2704. https://openaccess.thecvf.com/content/CVPR2021/html/Habibian_Skip-Convolutions_for_Efficient_Video_Processing_CVPR_2021_paper.html

[13] Song Han, Huizi Mao, and William J. Dally. 2016. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1510.00149

[14] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both Weights and Connections for Efficient Neural Network. In *Advances in Neural Information Processing Systems, NeurIPS 2015*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc., 1135–1143. https://proceedings.neurips.cc/paper/2015/file/ae0eb3eed39d2bcef4622b2499a05fe6-Paper.pdf

[15] Gyuwan Kim and Kyunghyun Cho. 2021. Length-Adaptive Transformer: Train Once with Length Drop, Use Anytime with Search. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), August 1-6, 2021*, Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (Eds.). Association for Computational Linguistics, 6501–6511. https://doi.org/10.18653/v1/2021.acl-long.508

[16] Sehoon Kim, Sheng Shen, David Thorsley, Amir Gholami, Woosuk Kwon, Joseph Hassoun, and Kurt Keutzer. 2021. Learned Token Pruning for Transformers. arXiv:2107.00910 [cs.CL]

[17] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=H1eA7AEtvS

[18] Jiaoda Li, Ryan Cotterell, and Mrinmaya Sachan. 2021. Differentiable Subset Pruning of Transformer Heads. *Transactions of the Association for Computational Linguistics* 9 (12 2021), 1442–1459. https://doi.org/10.1162/tacl_a_00436

[19] Andy T. Liu, Shang-Wen Li, and Hung-yi Lee. 2021. TERA: Self-Supervised Learning of Transformer Encoder Representation for Speech. *IEEE ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), 2351–2366. https://doi.org/10.1109/TASLP.2021.3095662

[20] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Improving Multi-Task Deep Neural Networks via Knowledge Distillation for Natural Language Understanding. *CoRR* abs/1904.09482 (2019). arXiv:1904.09482 http://arxiv.org/abs/1904.09482

[21] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019). arXiv:1907.11692 http://arxiv.org/abs/1907.11692

[22] J. S. McCarley. 2019. Pruning a BERT-based Question Answering Model. *CoRR* abs/1910.06360 (2019). arXiv:1910.06360 http://arxiv.org/abs/1910.06360

[23] Paul Michel, Omer Levy, and Graham Neubig. 2019. Are Sixteen Heads Really Better than One?. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.), Vol. 32. Curran Associates, Inc., 14014–14024. https://proceedings.neurips.cc/paper/2019/hash/2c601ad9d2ff9bc8b282670cdd54f69f-Abstract.html

[24] Daniel Neil, Junhaeng Lee, Tobi Delbrück, and Shih-Chii Liu. 2017. Delta Networks for Optimized Recurrent Network Computation. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017 (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 2584–2593. http://proceedings.mlr.press/v70/neil17a.html

[25] Daniel Neimark, Omri Bar, Maya Zohar, and Dotan Asselmann. 2021. Video Transformer Network. In *IEEE/CVF International Conference on Computer Vision Workshops, ICCVW 2021, Montreal, BC, Canada, October 11-17, 2021*. IEEE, 3156–3165. https://doi.org/10.1109/ICCVW54120.2021.00355

[26] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI blog* (2019).

[27] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR* abs/1910.01108 (2019). arXiv:1910.01108 http://arxiv.org/abs/1910.01108

[28] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient Knowledge Distillation for BERT Model Compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, 4322–4331. https://doi.org/10.18653/v1/D19-1441

[29] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (Eds.). Association for Computational Linguistics, 2158–2170. https://doi.org/10.18653/v1/2020.acl-main.195

[30] Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling Task-Specific Knowledge from BERT into Simple Neural Networks. *CoRR* abs/1903.12136 (2019). arXiv:1903.12136 http://arxiv.org/abs/1903.12136

[31] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event (Proceedings of*

Machine Learning Research, Vol. 139), Marina Meila and Tong Zhang (Eds.). PMLR, 10347–10357. http://proceedings.mlr.press/v139/touvron21a.html

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, NeurIPS 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[33] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, Anna Korhonen, David R. Traum, and Lluís Màrquez (Eds.). Association for Computational Linguistics, 5797–5808. https://doi.org/10.18653/v1/p19-1580

[34] Hanrui Wang, Zhekai Zhang, and Song Han. 2021. SpAtten: Efficient Sparse Attention Architecture with Cascade Token and Head Pruning. In *IEEE International Symposium on High-Performance Computer Architecture, HPCA 2021, Seoul, South Korea, February 27 - March 3, 2021*. IEEE, 97–110. https://doi.org/10.1109/HPCA51647.2021.00018

[35] Pete Warden. 2018. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *CoRR* abs/1804.03209 (2018). arXiv:1804.03209 http://arxiv.org/abs/1804.03209

[36] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis E.H. Tay, Jiashi Feng, and Shuicheng Yan. 2021. Tokens-to-Token ViT: Training Vision Transformers From Scratch on ImageNet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 558–567.

[37] Jie-Fang Zhang, Ching-En Lee, Chester Liu, Yakun Sophia Shao, Stephen W. Keckler, and Zhengya Zhang. 2019. SNAP: A 1.67 - 21.55TOPS/W Sparse Neural Acceleration Processor for Unstructured Sparse Deep Neural Network Inference in 16nm CMOS. In *2019 Symposium on VLSI Circuits, Kyoto, Japan, June 9-14, 2019*. IEEE, C306–C307. https://doi.org/10.23919/VLSIC.2019.8778193