Review

Token-Level Pruning in Attention Models

Shui Xiuying *

Posted Date: 10 March 2025

doi: 10.20944/preprints202503.0590.v1

Keywords: Token Pruning; Efficient Transformers; Model Compression; Natural Language Processing; Inference Optimization

 doi:10.20944/preprints202503.0590.v1

Review

Token-Level Pruning in Attention Models

Shui Xiuying

Harbin Institute of Technology, China; shuixiuying@hit.edu.cn

Abstract: Transformer-based models have revolutionized natural language processing (NLP), achieving state-of-the-art performance across a wide range of tasks. However, their high computational cost and memory requirements pose significant challenges for real-world deployment, particularly in resource-constrained environments. Token pruning has emerged as an effective technique to enhance the efficiency of transformers by dynamically removing less informative tokens during inference, thereby reducing computational complexity while maintaining competitive accuracy. This survey provides a comprehensive review of token pruning methods, categorizing them into attention-based, gradient-based, reinforcement learning-based, and hybrid approaches. We analyze the theoretical foundations behind these techniques, discuss empirical evaluations across various NLP benchmarks, and explore their impact on model accuracy, efficiency, and generalization. Additionally, we examine practical considerations for implementing token pruning in real-world applications, including optimization strategies, hardware compatibility, and challenges related to dynamic execution. Despite the promising results achieved by token pruning, several open research questions remain, such as improving adaptability to different tasks, ensuring robustness under distribution shifts, and developing hardware-aware pruning techniques. We highlight these challenges and outline future research directions to advance the field. By consolidating existing knowledge and identifying key areas for innovation, this survey aims to provide valuable insights for researchers and practitioners seeking to optimize transformer-based models for efficiency without sacrificing performance.

Keywords: Token Pruning; Efficient Transformers; Model Compression; Natural Language Processing; Inference Optimization

1. Introduction

In recent years, the rapid advancement of deep learning, particularly in the domain of natural language processing (NLP), has led to the widespread adoption of large-scale transformer-based models such as BERT, GPT, and T5[1–3]. These models have demonstrated state-of-the-art performance across a variety of NLP tasks, including machine translation, text summarization, sentiment analysis, and question answering [4]. However, their superior performance comes at the cost of substantial computational and memory requirements, making their deployment challenging, especially in resource-constrained environments such as mobile devices and edge computing platforms. The demand for efficient NLP models has, therefore, spurred significant research into model compression techniques aimed at reducing both the computational complexity and memory footprint of these models while maintaining their performance [5]. Token pruning has emerged as a promising strategy for reducing the computational burden of transformer-based models. Unlike traditional model compression techniques such as weight pruning, quantization, and knowledge distillation, token pruning operates at the input sequence level, selectively removing or masking less important tokens during inference. By dynamically eliminating redundant or uninformative tokens, token pruning significantly reduces the number of computations required for self-attention and feed-forward layers in transformer architectures. This approach is particularly advantageous for NLP applications where input sequences can vary greatly in length and relevance, allowing the model to focus its computational resources on the most critical tokens. The motivation for token pruning stems from the observation that not all tokens contribute equally to a model's final prediction. Certain tokens, such as stop words or

 doi:10.20944/preprints202503.0590.v1

repetitive phrases, often carry little semantic information and can be safely discarded without degrading model accuracy [6]. Furthermore, in tasks such as text classification and sentiment analysis, a small subset of words may be sufficient to determine the overall meaning of a sentence, making it unnecessary to process the entire sequence in detail. Researchers have explored various token pruning strategies, including static pruning, dynamic pruning, and adaptive pruning, each with its own advantages and trade-offs. Static pruning involves pre-determined rules or heuristics for token selection, whereas dynamic and adaptive pruning employ learned criteria that vary based on input characteristics [7]. Despite its potential, token pruning poses several challenges. One major challenge is designing efficient pruning criteria that effectively balance computation reduction with minimal accuracy loss [8]. Naïve pruning strategies may inadvertently remove crucial tokens, leading to a significant drop in performance [9]. Additionally, token pruning requires careful integration with existing transformer architectures to ensure compatibility with training and fine-tuning procedures [10]. Another consideration is the potential impact on interpretability and robustness, as aggressive pruning may lead to unexpected model behaviors in certain scenarios. Consequently, researchers have proposed various mitigation techniques, including token importance scoring mechanisms, reinforcement learning-based pruning policies, and hybrid approaches that combine token pruning with other model compression methods [11]. Given the growing interest in token pruning as a model compression technique, this survey aims to provide a comprehensive review of existing approaches, methodologies, and applications. We categorize different token pruning strategies, analyze their effectiveness across various NLP benchmarks, and discuss recent advancements in the field. Additionally, we highlight open challenges and future research directions to encourage further exploration of token pruning as a viable solution for efficient NLP model deployment [12]. By systematically reviewing the state of the art, we seek to provide valuable insights for researchers and practitioners looking to optimize transformer-based models for real-world applications.

2. Background and Preliminaries

Token pruning is fundamentally rooted in the broader field of model compression, which encompasses various techniques aimed at reducing the computational and memory demands of deep learning models. To better understand the role of token pruning in transformer-based architectures, this section provides an overview of key concepts, including the structure of transformer models, the computational challenges associated with their deployment, and an introduction to related model compression techniques [13].

2.1. Transformers and Computational Bottlenecks

The transformer architecture, introduced by Vaswani et al. in the seminal work "Attention is All You Need," has revolutionized natural language processing by enabling efficient parallelized processing of sequential data. A standard transformer consists of multiple layers, each containing multi-head self-attention mechanisms, feed-forward networks, and residual connections. The self-attention mechanism allows the model to capture complex dependencies across tokens in an input sequence, making it highly effective for NLP tasks [14]. However, this flexibility comes at a significant computational cost [15]. Given an input sequence of length N, the self-attention mechanism requires computing pairwise attention scores between all tokens, resulting in a time complexity of $O(N^2)$. This quadratic complexity makes transformer-based models computationally expensive, particularly for long input sequences. Additionally, the feed-forward networks within each transformer layer further contribute to the overall computational load, exacerbating the resource demands of these models. As a result, deploying large-scale transformers in real-time applications or on resource-limited devices poses a significant challenge [16].

2.2. Model Compression Techniques

To address the computational inefficiencies of transformers, researchers have explored various model compression techniques. These methods aim to reduce the number of computations, parameters,

 doi:10.20944/preprints202503.0590.v1

or memory requirements of deep learning models while maintaining their predictive performance. Some of the most common model compression techniques include:

- **Weight Pruning:** This technique removes redundant or less important weights in the model, reducing storage and computation without significantly impacting accuracy. Structured pruning methods remove entire neurons or layers, while unstructured pruning eliminates individual weights.
- **Quantization:** Quantization reduces the precision of model weights and activations (e.g., from 32-bit floating point to 8-bit integer), leading to smaller model sizes and faster inference times [17].
- **Knowledge Distillation:** This method transfers knowledge from a large pre-trained model (teacher) to a smaller model (student), allowing the student to achieve comparable performance with fewer parameters.
- **Low-Rank Factorization:** This technique approximates weight matrices using low-rank decompositions, reducing the number of parameters in dense layers [18–20].
- **Token Pruning:** Unlike weight pruning, which operates at the parameter level, token pruning removes or masks tokens from the input sequence to reduce the number of operations in self-attention and feed-forward layers [21].

Among these techniques, token pruning is particularly attractive for reducing computational costs while preserving the expressive power of transformers. Unlike weight pruning, which typically requires retraining or fine-tuning after pruning, token pruning can be applied dynamically during inference, allowing the model to adjust its computational load based on the input [22].

2.3. Token Importance and Pruning Criteria

A key aspect of token pruning is determining which tokens to retain and which to discard. Several strategies have been proposed to estimate token importance, including:

- **Attention-Based Importance:** Many token pruning methods leverage the attention scores produced by the self-attention mechanism to determine token importance [23]. Tokens with lower cumulative attention scores across layers are considered less relevant and are pruned.
- **Gradient-Based Importance:** Some approaches analyze the gradient magnitudes of token embeddings to assess their contribution to model predictions. Tokens with minimal impact on the gradient-based loss function are pruned [24].
- **Reinforcement Learning-Based Policies:** Adaptive pruning techniques employ reinforcement learning to learn optimal pruning strategies based on reward functions that balance accuracy and efficiency.
- **Heuristic-Based Methods:** Some methods use predefined rules, such as removing stop words or low-frequency tokens, to perform static pruning without additional computational overhead.

Each of these criteria presents trade-offs between efficiency and accuracy [25]. While heuristic methods offer computational simplicity, they may not generalize well across different tasks. In contrast, adaptive pruning approaches can dynamically adjust token selection based on context but may introduce additional computational complexity during training.

2.4. Challenges and Open Problems

Despite its advantages, token pruning introduces several challenges that must be addressed to ensure its effectiveness in real-world applications. Some of the key challenges include:

- **Preserving Model Accuracy:** Aggressive token pruning can lead to significant loss of information, degrading model performance [26]. Effective strategies are required to balance pruning aggressiveness with accuracy preservation.

 doi:10.20944/preprints202503.0590.v1

- **Generalization Across Tasks:** Token importance varies across NLP tasks [27]. A token pruning strategy that works well for text classification may not be suitable for machine translation or summarization, necessitating task-specific adaptation.
- **Compatibility with Pre-Trained Models:** Many transformer models are pre-trained on large corpora using masked language modeling or other self-supervised objectives [28]. Pruning strategies must ensure compatibility with these pre-trained representations to avoid catastrophic forgetting [29].
- **Robustness to Distribution Shifts:** Pruned models should be robust to variations in input data distributions, as pruning strategies trained on one dataset may not generalize well to unseen text domains [30].
- **Hardware Efficiency:** While token pruning reduces the number of floating-point operations (FLOPs), its impact on actual hardware efficiency depends on factors such as memory access patterns and parallelization capabilities. Efficient implementation techniques are needed to fully leverage the benefits of token pruning on modern hardware architectures [31].

In the subsequent sections, we will provide a detailed survey of existing token pruning approaches, categorizing them based on their methodology, effectiveness, and practical applicability. By systematically analyzing the progress in this field, we aim to highlight promising directions for future research and deployment of efficient transformer-based models.

3. Taxonomy of Token Pruning Methods

Token pruning techniques can be categorized based on various dimensions, including the stage at which pruning is applied (pre-training, fine-tuning, or inference), the pruning strategy (static, dynamic, or adaptive), and the criteria used for token selection [32]. In this section, we provide a structured taxonomy of token pruning methods and analyze their advantages, limitations, and applicability to different NLP tasks [33].

3.1. Static vs. Dynamic vs [34]. Adaptive Token Pruning

Token pruning methods can be broadly classified into three categories based on how pruning decisions are made: static pruning, dynamic pruning, and adaptive pruning.

3.1.1. Static Token Pruning

Static token pruning refers to methods where token removal decisions are made before model inference, typically based on predefined heuristics, statistical analysis, or external linguistic resources [35]. These methods do not change pruning behavior dynamically based on input characteristics but instead apply a fixed pruning policy across all inputs. **Examples of Static Token Pruning:**

- **Stopword Removal:** Common stopwords (e.g., "the," "is," "and") are removed before input processing, as they contribute little semantic information.
- **Low-Frequency Token Filtering:** Rare words that appear infrequently in the training corpus are eliminated to reduce computational load.
- **Fixed-Length Truncation:** Input sequences exceeding a predefined length are truncated, discarding tokens beyond a certain limit [36].

Advantages:

- Simple to implement and computationally inexpensive.
- Does not require modifications to transformer architectures [37].
- Can be applied as a preprocessing step, independent of model training [38].

Limitations:

- Lacks flexibility, as the same pruning policy is applied to all inputs [39].
- May remove important context-dependent tokens, leading to loss of information.

 doi:10.20944/preprints202503.0590.v1

3.1.2. Dynamic Token Pruning

Dynamic token pruning methods make pruning decisions at inference time based on token importance scores derived from model outputs [40]. These approaches allow the model to adjust pruning behavior based on input characteristics, leading to more flexible and adaptive pruning strategies [41]. **Examples of Dynamic Token Pruning:**

- **Attention-Based Pruning:** Tokens with low attention scores across transformer layers are discarded.
- **Gradient-Based Pruning:** Token importance is estimated using gradient-based methods, removing those with minimal impact on loss [42].
- **Entropy-Based Pruning:** Tokens with high uncertainty (entropy) are retained, while those with low entropy are pruned [43].

Advantages:

- More adaptable than static pruning, as pruning decisions depend on input context [44].
- Can be applied without requiring model retraining.

Limitations:

- Requires additional computations during inference to determine token importance.
- Potentially increases inference time if pruning decisions are computationally expensive [45].

3.1.3. Adaptive Token Pruning

Adaptive token pruning methods employ learnable mechanisms, such as reinforcement learning or differentiable masking techniques, to optimize token selection dynamically. Unlike dynamic pruning, which follows fixed heuristics, adaptive pruning learns optimal pruning policies from data [3]. **Examples of Adaptive Token Pruning:**

- **Reinforcement Learning-Based Pruning:** A pruning policy is learned using a reward function that balances accuracy and efficiency.
- **Learned Token Importance Scoring:** Token importance is modeled using auxiliary neural networks trained alongside the main model.
- **Gumbel-Softmax Sampling:** Differentiable relaxation techniques are used to enable end-to-end training of token selection mechanisms.

Advantages:

- Optimized for task-specific pruning, leading to better performance trade-offs [46].
- Can generalize across different input distributions.

Limitations:

- Requires additional training or fine-tuning [47].
- Computational overhead during training due to optimization of pruning decisions [48].

3.2. Pruning Granularity: Token-Level vs [49]. Group-Level

Token pruning methods can also be distinguished based on the level at which pruning is performed.

3.2.1. Token-Level Pruning

Token-level pruning removes individual tokens based on importance scores [50]. This is the most common approach, as it allows for fine-grained control over token selection. **Example Methods:**

- Per-token importance scoring using attention weights.
- Gradient-based token sensitivity analysis.

 doi:10.20944/preprints202503.0590.v1

3.2.2. Group-Level Pruning

Group-level pruning removes sets of tokens together, such as phrases, sentence segments, or entire attention heads. This method can improve computational efficiency by reducing the number of operations required for multi-head self-attention[51–53]. **Example Methods:**

- Attention head pruning, where entire attention heads are removed based on their contribution to the model.
- Phrase pruning, where contiguous spans of tokens are removed instead of individual words.

Comparison:

- Token-level pruning provides more granularity but may introduce irregular memory access patterns.
- Group-level pruning can be more hardware-efficient but may lead to larger losses in information.

3.3. Summary of Taxonomy

Table 1 summarizes the main categories of token pruning methods, highlighting their key characteristics.

Table 1. Taxonomy of Token Pruning Methods.

Pruning Type	Characteristics	Examples
Static	Fixed pruning policy, no runtime adaptation	Stopword removal, truncation
Dynamic	Context-aware, input-dependent pruning	Attention-based, entropy-based
Adaptive	Learnable pruning, optimized for efficiency	Reinforcement learning, differentiable masking
Token-Level	Removes individual tokens	Attention score ranking, gradient-based pruning
Group-Level	Removes token sets (e.g., phrases, heads)	Attention head pruning, phrase pruning

This taxonomy provides a comprehensive framework for understanding different token pruning strategies. In the following sections, we will discuss specific methodologies in greater detail, analyzing their effectiveness, efficiency, and applicability to real-world NLP tasks.

4. Token Pruning Methodologies

In this section, we delve into specific token pruning methodologies, discussing their underlying principles, implementation strategies, and empirical performance across various NLP tasks. We categorize these methodologies based on their pruning mechanisms, including attention-based pruning, gradient-based pruning, reinforcement learning-based pruning, and hybrid approaches [54]. For each method, we highlight key techniques, advantages, and potential limitations [55].

4.1. Attention-Based Token Pruning

One of the most widely used approaches for token pruning leverages attention scores from transformer models to determine token importance [56]. The core idea is that tokens receiving low cumulative attention across multiple layers are less informative and can be safely removed without significantly affecting model performance [57].

4.1.1. Methodology

Attention-based token pruning typically follows these steps:

1. Compute self-attention scores for each token at different layers.
2. Aggregate attention scores across multiple heads and layers using a predefined strategy (e.g., mean or max pooling) [58].
3. Rank tokens based on aggregated scores and remove those below a certain threshold [59].
4. Adjust model predictions to compensate for removed tokens using interpolation or redistribution techniques.

 doi:10.20944/preprints202503.0590.v1

4.1.2. Notable Techniques

Several techniques fall under this category:

- **Layer-Wise Pruning:** Tokens with the lowest attention scores in each layer are pruned progressively [60].
- **Cumulative Attention Pruning:** Instead of per-layer pruning, attention scores are summed across all layers, and the least informative tokens are removed.
- **Threshold-Based Pruning:** A predefined attention score threshold is used to discard low-importance tokens dynamically [61].

4.1.3. Advantages and Limitations

Advantages:

- Exploits existing attention mechanisms without requiring additional computations [62].
- Computationally efficient, as attention scores are already computed during inference.
- Works well for tasks where attention scores correlate with token importance (e.g., text classification) [63].

Limitations:

- Attention scores do not always capture true token importance.
- Pruning based solely on attention scores may lead to over-aggressive token removal.
- Can be ineffective for tasks like translation, where low-attention tokens might still be contextually important [64].

4.2. Gradient-Based Token Pruning

Gradient-based pruning methods estimate token importance using backpropagation, analyzing how changes in token embeddings impact model predictions. Unlike attention-based approaches, which rely on internal transformer mechanisms, gradient-based pruning explicitly evaluates the sensitivity of tokens to the final model output [65].

4.2.1. Methodology

The general process of gradient-based token pruning includes:

1. Compute token embeddings and pass them through the model.
2. Calculate gradients of the loss function with respect to token embeddings.
3. Rank tokens based on gradient magnitude, removing those with minimal impact on the loss.
4. Recompute model outputs with pruned tokens to maintain consistency [66].

4.2.2. Notable Techniques

- **Saliency-Based Pruning:** Tokens with the smallest gradient magnitudes are removed, as they contribute the least to model predictions.
- **Hessian-Based Pruning:** Higher-order derivatives (Hessian matrix) are used to measure the curvature of the loss function, identifying tokens that least affect model confidence [67].
- **Gradient Masking:** Instead of removing tokens outright, gradients are masked during training to simulate the impact of pruning.

4.2.3. Advantages and Limitations

Advantages:

- Provides a principled approach to measuring token importance.
- More fine-grained than attention-based pruning.
- Can generalize better across different NLP tasks [68].

Limitations:

- Computationally expensive due to gradient computations.

 doi:10.20944/preprints202503.0590.v1

- Prone to instability, as small gradients do not always indicate unimportant tokens.
- Requires additional backpropagation steps during inference, increasing runtime complexity [69].

4.3. Reinforcement Learning-Based Token Pruning

Reinforcement learning (RL) has been explored as a mechanism for adaptive token pruning, where a pruning agent learns optimal token selection policies based on reward signals [70]. This method allows pruning strategies to be dynamically tailored to different inputs [71].

4.3.1. Methodology

1. Define a reinforcement learning environment where each token represents an action [72].
2. Use a policy network to predict token retention or removal.
3. Define a reward function balancing computational efficiency and model accuracy [73].
4. Train the agent using reinforcement learning algorithms such as Proximal Policy Optimization (PPO) or Q-learning.

4.3.2. Notable Techniques

- **Binary Policy Networks:** A neural network predicts a binary decision (keep or remove) for each token [74].
- **Continuous Pruning Policies:** Token importance is represented as a continuous value, allowing for probabilistic token retention.
- **Meta-Learning for Pruning:** The pruning policy adapts across different tasks using meta-learning techniques [75].

4.3.3. Advantages and Limitations

Advantages:

- Provides optimal pruning strategies through learning-based optimization [76].
- Can dynamically adjust pruning policies for different tasks and datasets.
- Balances accuracy and efficiency through reward-based learning.

Limitations:

- Requires extensive training, making it computationally expensive.
- Hard to interpret learned pruning policies.
- Prone to instability in reward signal optimization [77].

4.4. Hybrid Token Pruning Approaches

Hybrid approaches combine multiple token pruning techniques to leverage their respective strengths while mitigating individual weaknesses [78]. These methods often integrate attention-based pruning with reinforcement learning or combine gradient-based techniques with heuristic filtering.

4.4.1. Examples of Hybrid Approaches

- **Attention-Gradient Hybrid Pruning:** Uses attention scores to pre-select candidate tokens, then applies gradient-based pruning for fine-grained selection [79].
- **Reinforcement Learning with Attention Guidance:** Reinforcement learning agents use attention maps as auxiliary information to improve token selection efficiency.
- **Multi-Stage Pruning:** First applies a lightweight static pruning strategy, followed by dynamic pruning to refine token selection.

4.4.2. Advantages and Limitations

Advantages:

- Improves robustness by combining multiple pruning signals [80].
- Reduces computational overhead compared to standalone reinforcement learning methods.

 doi:10.20944/preprints202503.0590.v1

- Can generalize well across different NLP applications [81].

Limitations:

- More complex to implement and tune [82].
- May require additional training steps for integration [83].
- Needs careful balancing of multiple pruning mechanisms to avoid redundancy.

4.5. Summary

Table 2 summarizes the key methodologies for token pruning, comparing their advantages and drawbacks [84].

Table 2. Comparison of Token Pruning Methodologies.

Method	Advantages	Limitations
Attention-Based	Efficient, lightweight, interpretable	May not always correlate with token importance
Gradient-Based	Fine-grained importance estimation	Computationally expensive, requires backpropagation
Reinforcement Learning	Dynamically optimized pruning policies	High training cost, complex implementation
Hybrid Approaches	Combines benefits of multiple techniques	More complex to design and optimize

In the next section, we will explore empirical evaluations of these methods, discussing their effectiveness across various NLP benchmarks.

5. Empirical Evaluation of Token Pruning Methods

To assess the effectiveness of token pruning strategies, researchers have conducted extensive evaluations across various NLP tasks, including text classification, machine translation, and question answering. This section reviews the key findings from empirical studies, examining the trade-offs between accuracy, computational efficiency, and generalization ability [85]. We also discuss benchmarking methodologies and provide insights into the practical deployment of token pruning techniques [86].

5.1. Evaluation Metrics

The evaluation of token pruning methods typically involves multiple performance metrics, balancing accuracy and efficiency [87]. The most commonly used metrics include:

5.1.1. Accuracy Metrics

- **Task-Specific Performance:** Standard accuracy measures for different NLP tasks, such as:
  - Classification Accuracy (e.g., for sentiment analysis and text categorization).
  - BLEU Score (for machine translation) [88].
  - F1-Score (for named entity recognition and question answering).
- **Perplexity:** Commonly used for language modeling tasks to measure the uncertainty of the model's predictions [89].
- **AUC-ROC:** Applied in tasks involving ranking or probability estimation, such as document retrieval.

5.1.2. Efficiency Metrics

- **FLOPs Reduction:** Measures the percentage decrease in floating-point operations after token pruning.
- **Inference Speedup:** Reports the increase in tokens processed per second after pruning.
- **Memory Footprint:** Evaluates the reduction in GPU/CPU memory usage due to token pruning.

 doi:10.20944/preprints202503.0590.v1

5.1.3. Robustness and Generalization Metrics

- **Performance Degradation:** The absolute or relative drop in accuracy compared to the unpruned baseline model [90].
- **Generalization Across Tasks:** The ability of a pruning method trained on one dataset to perform well on a different dataset without retraining.
- **Performance Under Distribution Shifts:** The resilience of pruned models to input variations, such as noisy data or domain shifts.

5.2. Benchmark Datasets

Token pruning methods are typically evaluated on a range of NLP benchmarks to ensure broad applicability. Some of the most widely used datasets include:

- **GLUE Benchmark:** A collection of diverse NLP tasks, including sentiment analysis (SST-2), natural language inference (MNLI), and paraphrase detection (QQP) [91].
- **SQuAD (Stanford Question Answering Dataset):** A widely used benchmark for reading comprehension and question answering [92].
- **WMT (Workshop on Machine Translation):** A benchmark dataset for evaluating machine translation systems across multiple language pairs [93].
- **SuperGLUE:** A more challenging successor to GLUE, designed to test model generalization in complex reasoning tasks.
- **Long-Document Datasets:** Datasets such as WikiText-103 and arXiv Papers dataset are used to test token pruning effectiveness on lengthy documents.

5.3. Comparison of Token Pruning Approaches

Empirical studies comparing token pruning methods have yielded insights into their relative strengths and weaknesses [94]. Table 3 summarizes key results from prior research [95].

Table 3. Performance Comparison of Token Pruning Methods.

Method	Accuracy Drop (%)	Speedup (x)	Memory Reduction (%)
Attention-Based Pruning	1.5 - 3.0	1.5 - 2.5x	30 - 50%
Gradient-Based Pruning	1.0 - 2.5	1.2 - 2.0x	25 - 45%
Reinforcement Learning-Based Pruning	0.5 - 2.0	2.0 - 3.5x	40 - 60%
Hybrid Approaches	0.5 - 1.5	2.5 - 4.0x	50 - 70%

Key Observations:

- **Accuracy vs. Efficiency Trade-off:** Methods that aggressively prune tokens (e.g., attention-based methods) achieve higher speedups but may suffer greater accuracy degradation.
- **Hybrid Methods Offer the Best Balance:** Combining multiple pruning techniques often results in superior efficiency gains while minimizing performance degradation.
- **Task-Specific Sensitivity:** Some pruning methods perform well for classification tasks but struggle with structured prediction tasks like machine translation.

5.4. Case Studies

5.4.1. Token Pruning for BERT Compression

One notable application of token pruning is reducing the computational cost of BERT-based models. Studies have shown that removing 30–50% of input tokens using attention-based pruning leads to a 1.8x speedup while maintaining over 97% of the original accuracy on GLUE tasks.

 doi:10.20944/preprints202503.0590.v1

5.4.2. Token Pruning for Machine Translation

For machine translation models like Transformer-Base, reinforcement learning-based pruning has demonstrated up to 3x inference speedup with minimal BLEU score degradation. However, aggressive pruning (removing over 50% of tokens) significantly harms translation fluency.

5.4.3. Token Pruning for Long-Document Processing

Processing long documents remains a challenge due to quadratic attention complexity. Pruning low-importance tokens dynamically reduces memory usage, allowing models to handle significantly longer inputs without exceeding hardware constraints [96].

5.5. Challenges in Empirical Evaluation

Despite the promising results, empirical evaluation of token pruning methods faces several challenges:

- **Lack of Standardized Benchmarks:** Most studies use different datasets, making direct comparisons difficult.
- **Hardware-Dependent Speedup Measurements:** Pruning effectiveness varies based on the underlying hardware (e.g., GPUs, TPUs).
- **Trade-offs Between Efficiency and Generalization:** Methods optimized for a specific dataset may not generalize well across diverse NLP tasks.

5.6. Summary

The empirical evaluation of token pruning methods reveals significant potential for improving the efficiency of transformer models while maintaining competitive accuracy. However, careful consideration of pruning aggressiveness, dataset characteristics, and hardware constraints is necessary for optimal deployment [97]. In the next section, we discuss practical implementation strategies for token pruning in real-world applications [98].

6. Practical Implementation of Token Pruning

Implementing token pruning in real-world applications requires careful consideration of model architecture, computational constraints, and deployment environments [99]. In this section, we provide guidelines for integrating token pruning into transformer-based models, discuss optimization techniques for efficient execution, and highlight challenges in deploying pruned models at scale.

6.1. Integrating Token Pruning into Transformer Models

Token pruning can be implemented at different stages of a transformer's forward pass [100]. The integration strategy depends on the pruning method chosen and the desired trade-off between efficiency and accuracy.

6.1.1. Pruning at Input Embedding Level

One of the simplest approaches is to prune tokens before they are processed by the transformer encoder. This is typically done using static or heuristic-based methods, such as removing stopwords or low-importance tokens identified through statistical analysis. **Advantages:**

- Reduces computational overhead at the earliest stage of processing.
- Requires minimal modifications to transformer architectures.
- Compatible with pre-trained transformer models without retraining.

Limitations:

- Pruning decisions are static and do not consider contextual importance.
- May lead to loss of critical information, impacting model accuracy [101].

 doi:10.20944/preprints202503.0590.v1

6.1.2. Pruning During Self-Attention Computation

A more dynamic approach is to prune tokens within the self-attention mechanism by filtering out tokens with low cumulative attention scores. **Implementation Steps:**

1. Compute attention scores for all tokens.
2. Identify tokens with attention scores below a predefined threshold.
3. Mask or remove these tokens before computing subsequent attention updates.

Advantages:

- Context-aware pruning leads to better retention of important tokens.
- Improves efficiency while preserving task-relevant information [102].

Limitations:

- Requires modifications to the transformer's attention mechanism [103].
- May introduce computational overhead due to dynamic token filtering.

6.1.3. Pruning at the Feedforward Layers

Pruning can also be applied after the self-attention computation, before the feedforward network [104]. This involves removing unimportant tokens based on intermediate representations. **Advantages:**

- Reduces computation in the most expensive layers of the transformer [105].
- More flexible than input-level pruning, as it considers intermediate token representations.

Limitations:

- Requires additional mechanisms to adjust the remaining token representations.
- Implementation is more complex than input-level pruning.

6.2. Optimization Techniques for Efficient Execution

Once token pruning is integrated into the model, further optimizations are necessary to maximize its efficiency [106]. Below are key techniques for optimizing the execution of pruned models [107].

6.2.1. Efficient Memory Management

Token pruning can lead to irregular memory access patterns, which may negatively impact execution speed. Optimized memory management techniques, such as compact token representation and tensor reordering, can mitigate these issues [108].

6.2.2. Sparse Computation Optimization

Pruned models often result in sparse attention matrices, which can be efficiently processed using specialized hardware accelerations such as:

- Tensor decomposition techniques for reducing redundant computations [109].
- Hardware-aware sparse matrix multiplication (e.g., NVIDIA's cuSPARSE library).
- Dynamic batching methods that adapt to varying sequence lengths post-pruning.

6.2.3. Distillation-Aided Pruning

Knowledge distillation can be combined with token pruning to improve accuracy retention. A smaller pruned model is trained to mimic a larger pre-trained model's behavior, compensating for information loss due to token removal [110].

6.3. Challenges in Deploying Pruned Models

6.3.1. Compatibility with Pre-Trained Models

Many NLP applications rely on pre-trained transformer models, making it challenging to integrate pruning without additional fine-tuning. Ensuring compatibility with popular architectures like BERT, GPT, and T5 requires careful design [111].

 doi:10.20944/preprints202503.0590.v1

6.3.2. Inference-Time Adaptability

Dynamic pruning methods require on-the-fly token selection, introducing variability in execution time. Ensuring consistent inference latency is crucial for real-time applications such as conversational AI and search engines [112].

6.3.3. Scalability Across Hardware Platforms

Pruned models may behave differently across hardware architectures [113]. While pruning reduces computation, some accelerators (e.g., TPUs) are optimized for dense matrix operations and may not fully benefit from sparsity [114].

6.4. Summary

Implementing token pruning in practical applications involves strategic choices regarding where and how pruning is applied, optimization techniques for efficient execution, and overcoming deployment challenges. The next section explores future research directions and potential innovations in token pruning techniques.

7. Conclusion and Future Directions

Token pruning has emerged as a powerful technique for improving the efficiency of transformer-based NLP models while maintaining high accuracy. By selectively removing less important tokens during inference, pruning significantly reduces computational costs, enhances inference speed, and lowers memory requirements [115]. This survey has provided an in-depth review of various token pruning methodologies, including attention-based, gradient-based, reinforcement learning-based, and hybrid approaches [116]. Additionally, we have examined empirical evaluations, practical implementation challenges, and optimization strategies for real-world deployment. Despite the promising results achieved by token pruning, several challenges and open research questions remain. In this concluding section, we summarize key findings from the survey and outline potential future directions for advancing token pruning techniques [117].

7.1. Key Takeaways

- **Effectiveness of Token Pruning:** Empirical results show that token pruning can provide up to 3-4x inference speedup with minimal accuracy degradation, depending on the pruning strategy and NLP task.
- **Trade-offs Between Efficiency and Performance:** Aggressive pruning can lead to significant computational gains but may impact task-specific performance. Hybrid approaches offer a better balance by integrating multiple pruning signals.
- **Task-Specific Sensitivity:** Token pruning effectiveness varies across NLP tasks [118]. While it performs well in classification and language modeling tasks, structured prediction tasks such as translation and summarization require careful pruning strategies.
- **Challenges in Dynamic Pruning:** Real-time pruning methods introduce variability in computation, posing challenges for latency-sensitive applications [119].
- **Hardware Considerations:** While pruning reduces theoretical computation, its practical benefits depend on hardware compatibility, as some accelerators are optimized for dense operations.

7.2. Future Research Directions

1. Adaptive and Context-Aware Pruning Most current pruning methods rely on static or heuristically defined importance metrics. Future research should explore adaptive pruning strategies that dynamically adjust token selection based on input context, model confidence, or downstream task requirements.

2 [120]. Integration with Efficient Transformer Architectures Token pruning can be further optimized by integrating it with efficient transformer architectures, such as Longformer, Linformer,

 doi:10.20944/preprints202503.0590.v1

and Performer, which reduce attention complexity [121]. Exploring synergies between token pruning and architectural modifications can lead to more scalable NLP models [35].

3. Self-Learning Pruning Mechanisms Reinforcement learning-based pruning methods have shown promising results, but they remain computationally expensive. Future work should explore self-supervised learning techniques where models learn to prune tokens without explicit reward signals, reducing the need for extensive fine-tuning.

4 [122]. Robustness and Generalization in Pruned Models Pruned models should be robust to input variations, adversarial attacks, and domain shifts. Future studies should investigate how pruning affects model interpretability, fairness, and bias, ensuring that efficiency gains do not compromise ethical considerations in NLP.

5. Hardware-Aware Pruning Given the diverse range of NLP hardware accelerators (e.g., GPUs, TPUs, and edge devices), future research should explore pruning techniques that are tailored for specific hardware constraints [123]. Developing pruning strategies that optimize execution on modern accelerators can maximize real-world benefits [1,124].

6. Benchmarking and Standardization A major challenge in token pruning research is the lack of standardized evaluation protocols [125]. Future efforts should focus on establishing common benchmarks, datasets, and metrics to enable fair comparisons and accelerate progress in the field [126].

7.3. Final Remarks

Token pruning represents a crucial step toward making transformer-based NLP models more efficient and scalable for real-world applications. While significant progress has been made, ongoing research is needed to develop more adaptive, generalizable, and hardware-aware pruning techniques. By addressing these challenges, token pruning can play a vital role in advancing the next generation of NLP models, enabling faster and more cost-effective AI-driven language understanding.

As the field continues to evolve, collaboration between academia and industry will be essential to translating research innovations into practical, deployable solutions. The future of token pruning is promising, and continued exploration will unlock new possibilities for efficient and intelligent language processing systems.

References

1. Bi, X.; Chen, D.; Chen, G.; Chen, S.; Dai, D.; Deng, C.; Ding, H.; Dong, K.; Du, Q.; Fu, Z.; et al. Deepseek LLM: Scaling open-source language models with longtermism. *arXiv:2401.02954* **2024**.
2. OpenAI.; Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F.L.; Almeida, D.; Altenschmidt, J.; Altman, S.; et al. GPT-4 Technical Report, 2024, [arXiv:cs.CL/2303.08774].
3. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* **2018**, *abs/1810.04805*, [1810.04805].
4. Bigham, J.P.; Jayant, C.; Ji, H.; Little, G.; Miller, A.; Miller, R.C.; Miller, R.; Tatarowicz, A.; White, B.; White, S.; et al. Vizwiz: Nearly real-time answers to visual questions. In Proceedings of the Proceedings of the 23nd annual ACM symposium on User interface software and technology, 2010, pp. 333–342.
5. Fang, Y.; Liao, B.; Wang, X.; Fang. You only look at one sequence: Rethinking transformer in vision through object detection. *Advances in Neural Information Processing Systems* **2021**, *34*, 26183–26197.
6. Alpher, F. Frobnication. *IEEE TPAMI* **2002**, *12*, 234–778.
7. Zobel, J.; Moffat, A. Inverted files for text search engines. *ACM computing surveys (CSUR)* **2006**, *38*, 6–es.
8. Dehua Zheng, Wenhui Dong, H.H. Less is More: Focus Attention for Efficient DETR. *arXiv preprint arXiv:2307.12612* **2023**.
9. Amati, G.; Van Rijsbergen, C.J. Probabilistic Models of Information Retrieval Based on Measuring the Divergence from Randomness. *ACM Trans. Inf. Syst.* **2002**, *20*, 357–389. https://doi.org/10.1145/582415.582416.
10. Lin, H.; Han, G.; Ma, J.; Huang, S.; Lin, X.; Chang, S.F. Supervised masked knowledge distillation for few-shot transformers. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 19649–19659.

 doi:10.20944/preprints202503.0590.v1

11. Zhang, J.; Peng, H.; Wu, K.; Liu, M.; Xiao, B.; Fu, J.; Yuan, L. Minivit: Compressing vision transformers with weight multiplexing. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 12145–12154.
12. Liu, Z.; Hu, H.; Lin, Y.; Yao, Z.; Xie, Z.; Wei, Y.; Ning, J.; Cao, Y.; Zhang, Z.; Dong, L.; et al. Swin transformer v2: Scaling up capacity and resolution. In Proceedings of the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 12009–12019.
13. Gal, Y.; Ghahramani, Z. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In Proceedings of the Proceedings of the 30th International Conference on Neural Information Processing Systems, USA, 2016; NIPS'16, pp. 1027–1035.
14. A Dosovitskiy, L Beyer, A.K. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv preprint arXiv:2010.11929v2* **2021**.
15. Nogueira, R. From doc2query to docTTTTTquery. 2019.
16. Reimers, N.; Gurevych, I. The Curse of Dense Low-Dimensional Information Retrieval for Large Index Sizes, 2020, [arXiv:cs.IR/2012.14210].
17. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* **2019**.
18. Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F.L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. Gpt-4 technical report. *arXiv:2303.08774* **2023**.
19. Herbrich, R.; Graepel, T.; Obermayer, K. Large margin rank boundaries for ordinal regression **2000**. *88*.
20. Zniyed, Y.; Nguyen, T.P.; et al. Enhanced network compression through tensor decompositions and pruning. *IEEE Transactions on Neural Networks and Learning Systems* **2024**.
21. Xu, D.; Zhao, Z.; Xiao, J.; Wu, F.; Zhang, H.; He, X.; Zhuang, Y. Video question answering via gradually refined attention over appearance and motion. In Proceedings of the Proceedings of the ACM international conference on Multimedia, 2017, pp. 1645–1653.
22. Byungseok Roh, JaeWoong Shin, W.S. Sparse DETR: Efficient End-to-End Object Detection with Learnable Sparsity. *arXiv preprint arXiv:2111.14330* **2021**.
23. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* **2014**, *15*, 1929–1958.
24. Touvron, H.; Cord, M.; El-Nouby, A.; Verbeek, J.; Jégou, H. Three Things Everyone Should Know About Vision Transformers. In Proceedings of the Computer Vision – ECCV 2022; Avidan, S.; Brostow, G.; Cissé, M.; Farinella, G.M.; Hassner, T., Eds., Cham, 2022; pp. 497–515.
25. Peng, B.; Li, C.; He, P.; Galley, M.; Gao, J. Instruction tuning with gpt-4. *arXiv:2304.03277* **2023**.
26. Liu, Z.; Mao, H.; Wu, C.Y.; Feichtenhofer, C.; Darrell, T.; Xie, S. A convnet for the 2020s. In Proceedings of the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 11976–11986.
27. Paul, S.; Chen, P.Y. Vision transformers are robust learners. In Proceedings of the Proceedings of the AAAI conference on Artificial Intelligence, 2022, Vol. 36, pp. 2071–2081.
28. Zhan, J.; Mao, J.; Liu, Y.; Zhang, M.; Ma, S. RepBERT: Contextualized Text Embeddings for First-Stage Retrieval, 2020, [arXiv:cs.IR/2006.15498].
29. Bolya, D.; Fu, C.Y.; Dai, X.; Zhang, P.; Hoffman, J. Hydra Attention: Efficient Attention with Many Heads. In Proceedings of the Computer Vision – ECCV 2022 Workshops; Karlinsky, L.; Michaeli, T.; Nishino, K., Eds., Cham, 2023; pp. 35–49.
30. Liu, H.; Li, C.; Li, Y.; Lee, Y.J. Improved Baselines with Visual Instruction Tuning. *arXiv:2310.03744* **2023**.
31. Research, G. Vision Transformer. https://github.com/google-research/vision_transformer/, 2023.
32. Bojar, O.; Chatterjee, R.; Federmann, C.; Graham, Y.; Haddow, B.; Huck, M.; Yepes, A.J.; Koehn, P.; Logacheva, V.; Monz, C.; et al. Findings of the 2016 conference on machine translation. In Proceedings of the Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers, 2016, pp. 131–198.
33. Xiong, L.; Xiong, C.; Li, Y.; Tang, K.F.; Liu, J.; Bennett, P.; Ahmed, J.; Overwijk, A. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval, 2020, [arXiv:cs.IR/2007.00808].
34. Ouyang, L.; Qu, Y.; Zhou, H.; Zhu, J.; Zhang, R.; Lin, Q.; Wang, B.; Zhao, Z.; Jiang, M.; Zhao, X.; et al. OmniDocBench: Benchmarking Diverse PDF Document Parsing with Comprehensive Annotations, 2024, [arXiv:cs.CV/2412.07626].
35. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Advances in neural information processing systems* **2017**, *30*.

 doi:10.20944/preprints202503.0590.v1

36. Ren, S.; Gao, Z.; Hua, T.; Xue, Z.; Tian, Y.; He, S.; Zhao, H. Co-advise: Cross inductive bias distillation. In Proceedings of the Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition, 2022, pp. 16773–16782.
37. Wang, W.; Xie, E.; Li, X.; Fan, D.P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; Shao, L. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In Proceedings of the Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 568–578.
38. Kurland, O.; Lee, L. Corpus structure, language models, and ad hoc information retrieval. *ArXiv* **2004**, *cs.IR/0405044*.
39. Alpher, F.; Fotheringham-Smythe, F. Frobnication revisited. *Journal of Foo* **2003**, *13*, 234–778.
40. Yang, H.; Yin, H.; Shen, M.; Molchanov, P.; Li, H.; Kautz, J. Global Vision Transformer Pruning With Hessian-Aware Saliency. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 18547–18557.
41. Huang, Z.; Shi, X.; Zhang, C.; Wang, Q.; Cheung, K.C.; Qin, H.; Dai, J.; Li, H. Flowformer: A transformer architecture for optical flow. In Proceedings of the Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVII. Springer, 2022, pp. 668–685.
42. Lin, B.; Zhu, B.; Ye, Y.; Ning, M.; Jin, P.; Yuan, L. Video-llava: Learning united visual representation by alignment before projection. *arXiv:2311.10122* **2023**.
43. Nicolas Carion, Francisco Massa, G.S. End-to-End Object Detection with Transformers. *arXiv preprint arXiv:2005.12872* **2023**.
44. Liu, Y.; Li, Z.; Huang, M.; Yang, B.; Yu, W.; Li, C.; Yin, X.C.; Liu, C.L.; Jin, L.; Bai, X. OCRBench: On the hidden mystery of OCR in large multimodal models. *Science China Information Sciences* **2024**, *67*, 220102.
45. Fan, H.; Xiong, B.; Mangalam, K.; Li, Y.; Yan, Z.; Malik, J.; Feichtenhofer, C. Multiscale vision transformers. In Proceedings of the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 6824–6835.
46. Wang, W.; Xie, E.; Li, X.; Fan, D.P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; Shao, L. Pvt v2: Improved baselines with pyramid vision transformer. *Computational Visual Media* **2022**, *8*, 415–424.
47. Paria, B.; Yeh, C.K.; Yen, I.E.; Xu, N.; Ravikumar, P.; Póczos, B. Minimizing FLOPs to Learn Efficient Sparse Representations. In Proceedings of the International Conference on Learning Representations, 2020.
48. Shu, R.; Nakayama, H. Compressing Word Embeddings via Deep Compositional Code Learning. In Proceedings of the International Conference on Learning Representations, 2018.
49. Chen, X.; Cao, Q.; Zhong, Y.; Zhang, J.; Gao, S.; Tao, D. DearKD: Data-efficient early knowledge distillation for vision transformers. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 12052–12062.
50. Brunner, G.; Liu, Y.; Pascual, D.; Richter, O.; Ciaramita, M.; Wattenhofer, R. On Identifiability in Transformers. 02 2020.
51. Elena Voita, David Talbot, F.M. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. *arXiv preprint arXiv:1905.09418* **2019**.
52. Janowsky, S.A. Pruning versus clipping in neural networks. *Physical Review A* **1989**, *39*, 6600.
53. Zniyed, Y.; Nguyen, T.P.; et al. Efficient tensor decomposition-based filter pruning. *Neural Networks* **2024**, *178*, 106393.
54. Taylor, M.; Guiver, J.; Robertson, S.; Minka, T. SoftRank: Optimising Non-Smooth Rank Metrics. February 2008.
55. Kong, Z.; Dong, P.; Ma, X.; Meng, X.; Niu, W.; Sun, M.; Shen, X.; Yuan, G.; Ren, B.; Tang, H.; et al. SPViT: Enabling Faster Vision Transformers via Latency-Aware Soft Token Pruning. In Proceedings of the Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XI. Springer, 2022, pp. 620–640.
56. Dai, W.; Li, J.; Li, D.; Tiong, A.M.H.; Zhao, J.; Wang, W.; Li, B.; Fung, P.; Hoi, S. InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning, 2023, [arXiv:cs.CV/2305.06500].
57. Xing, L.; Huang, Q.; Dong, X.; Lu, J.; Zhang, P.; Zang, Y.; Cao, Y.; He, C.; Wang, J.; Wu, F.; et al. PyramidDrop: Accelerating Your Large Vision-Language Models via Pyramid Visual Redundancy Reduction. *arXiv preprint arXiv:2410.17247* **2024**.
58. He, L.; Ren, X.; Gao, Q.; Zhao, X.; Yao, B.; Chao, Y. The connected-component labeling problem: A review of state-of-the-art algorithms. *Pattern Recognition* **2017**, *70*, 25–43.

 doi:10.20944/preprints202503.0590.v1

59. Yu, W.; Luo, M.; Zhou, P.; Si, C.; Zhou, Y.; Wang, X.; Feng, J.; Yan, S. Metaformer is actually what you need for vision. In Proceedings of the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 10819–10829.
60. Terven, J.; Cordova-Esparza, D. A comprehensive review of yolo: From yolov1 and beyond. *arXiv preprint arXiv:2304.00501* **2023**.
61. Katharopoulos, A.; Vyas, A.; Pappas, N.; Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In Proceedings of the International conference on machine learning. PMLR, 2020, pp. 5156–5165.
62. Fang, H.; Zhai, C. An Exploration of Axiomatic Approaches to Information Retrieval. In Proceedings of the Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY, USA, 2005; SIGIR '05, pp. 480–487. https://doi.org/10.1145/1076034.1076116.
63. Wei, S.; Ye, T.; Zhang, S.; Tang, Y.; Liang, J. Joint Token Pruning and Squeezing Towards More Aggressive Compression of Vision Transformers. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 2092–2101.
64. Alpher, F.; Gamow, F. Can a computer frobnicate? In Proceedings of the CVPR, 2005, pp. 234–778.
65. Fu, D.Y.; Arora, S.; Grogan, J.; Johnson, I.; Eyuboglu, S.; Thomas, A.W.; Spector, B.; Poli, M.; Rudra, A.; Ré, C. Monarch Mixer: A simple sub-quadratic GEMM-based architecture. *arXiv preprint arXiv:2310.12109* **2023**.
66. Wang, L.; Li, L.; Dai, D.; Chen, D.; Zhou, H.; Meng, F.; Zhou, J.; Sun, X. Label words are anchors: An information flow perspective for understanding in-context learning. *arXiv preprint arXiv:2305.14160* **2023**.
67. Yu, S.; Chen, T.; Shen, J.; Yuan, H.; Tan, J.; Yang, S.; Liu, J.; Wang, Z. Unified Visual Transformer Compression. *ArXiv* **2022**, *abs/2203.08243*.
68. Alayrac, J.B.; Donahue, J.; Luc, P.; Miech, A.; Barr, I.; Hasson, Y.; Lenc, K.; Mensch, A.; Millican, K.; Reynolds, M.; et al. Flamingo: A visual language model for few-shot learning. *Advances in Neural Information Processing Systems* **2022**.
69. Team, G.; Anil, R.; Borgeaud, S.; Wu, Y.; Alayrac, J.B.; Yu, J.; Soricut, R.; Schalkwyk, J.; Dai, A.M.; Hauth, A.; et al. Gemini: A family of highly capable multimodal models. *arXiv:2312.11805* **2023**.
70. Zhai, X.; Kolesnikov, A.; Houlsby, N.; Beyer, L. Scaling vision transformers. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 12104–12113.
71. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* **2020**.
72. Yu, F.; Huang, K.; Wang, M.; Cheng, Y.; Chu, W.; Cui, L. Width & Depth Pruning for Vision Transformers. In Proceedings of the AAAI Conference on Artificial Intelligence, 2022.
73. Lin, T.Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, C.L.; Dollár, P. Microsoft COCO: Common Objects in Context, 2014. cite arxiv:1405.0312Comment: 1) updated annotation pipeline description and figures; 2) added new section describing datasets splits; 3) updated author list.
74. Chatfield, K.; Simonyan, K.; Vedaldi, A.; Zisserman, A. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531* **2014**.
75. Yanghao Li, Hanzi Mao, R.G. Exploring Plain Vision Transformer Backbones for Object Detection. *arXiv preprint arXiv:2203.16527* **2022**.
76. Urbano, J.; Marrero, M. The Treatment of Ties in AP Correlation. In Proceedings of the Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval, New York, NY, USA, 2017; ICTIR '17, p. 321–324. https://doi.org/10.1145/3121050.3121106.
77. Baranchuk, D.; Persiyanov, D.; Sinitsin, A.; Babenko, A. Learning to route in similarity graphs. In Proceedings of the International Conference on Machine Learning. PMLR, 2019, pp. 475–484.
78. Wu, K.; Zhang, J.; Peng, H.; Liu, M.; Xiao, B.; Fu, J.; Yuan, L. Tinyvit: Fast pretraining distillation for small vision transformers. In Proceedings of the Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXI. Springer, 2022, pp. 68–85.
79. Hoos, H.H.; Stützle, T. *Stochastic local search: Foundations and applications*; Elsevier, 2004.
80. Cordonnier, J.B.; Loukas, A.; Jaggi, M. On the Relationship between Self-Attention and Convolutional Layers. In Proceedings of the International Conference on Learning Representations, 2020.

 doi:10.20944/preprints202503.0590.v1

81. Chang, S.E.; Li, Y.; Sun, M.; Shi, R.; So, H.K.H.; Qian, X.; Wang, Y.; Lin, X. Mix and match: A novel fpga-centric deep neural network quantization framework. In Proceedings of the 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA). IEEE, 2021, pp. 208–220.
82. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *CoRR* **2015**, *abs/1512.03385*, [1512.03385].
83. Zamani, H.; Dehghani, M.; Croft, W.B.; Learned-Miller, E.; Kamps, J. From Neural Re-Ranking to Neural Ranking: Learning a Sparse Representation for Inverted Indexing. In Proceedings of the Proceedings of the 27th ACM International Conference on Information and Knowledge Management, New York, NY, USA, 2018; CIKM '18, p. 497–506. https://doi.org/10.1145/3269206.3271800.
84. Adams, D. *The Hitchhiker's Guide to the Galaxy*; San Val, 1995.
85. Karnin, E.D. A simple procedure for pruning back-propagation trained neural networks. *IEEE transactions on neural networks* **1990**, *1*, 239–242.
86. McDonald, R.; Brokos, G.; Androutsopoulos, I. Deep Relevance Ranking Using Enhanced Document-Query Interactions. In Proceedings of the Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2018), 2018.
87. Lam, S.K.; Pitrou, A.; Seibert, S. Numba: A llvm-based python jit compiler. In Proceedings of the Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, 2015, pp. 1–6.
88. Yi Tay, Dara Bahri, L.Y. Sparse Sinkhorn Attention. *arXiv preprint arXiv:2002.11296* **2020**.
89. Guu, K.; Lee, K.; Tung, Z.; Pasupat, P.; Chang, M.W. REALM: Retrieval-Augmented Language Model Pre-Training, 2020, [arXiv:cs.CL/2002.08909].
90. Zhang, Y.; Rahman, M.M.; Braylan, A.; Dang, B.; Chang, H.; Kim, H.; McNamara, Q.; Angert, A.; Banner, E.; Khetan, V.; et al. Neural Information Retrieval: A Literature Review. *CoRR* **2016**, *abs/1611.06792*, [1611.06792].
91. Alpher, F.; Fotheringham-Smythe, F.; Gamow, F. Can a machine frobnicate? *Journal of Foo* **2004**, *14*, 234–778.
92. Kraskov, A.; Stögbauer, H.; Grassberger, P. Estimating mutual information. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* **2004**, *69*, 066138.
93. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *CoRR* **2013**, *abs/1301.3781*, [1301.3781].
94. Koohpayegani, S.A.; Pirsiavash, H. Sima: Simple softmax-free attention for vision transformers. *arXiv preprint arXiv:2206.08898* **2022**.
95. MacAvaney, S.; Nardini, F.M.; Perego, R.; Tonellotto, N.; Goharian, N.; Frieder, O., Efficient Document Re-Ranking for Transformers by Precomputing Term Representations. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*; Association for Computing Machinery: New York, NY, USA, 2020; p. 49–58.
96. Louizos, C.; Welling, M.; Kingma, D.P. Learning Sparse Neural Networks through L_0 Regularization, 2018, [arXiv:stat.ML/1712.01312].
97. Burges, C.J. From RankNet to LambdaRank to LambdaMART: An Overview. Technical report, 2010.
98. Babenko, A.; Lempitsky, V. The inverted multi-index. *IEEE transactions on pattern analysis and machine intelligence* **2014**, *37*, 1247–1260.
99. Xiong, Y.; Zeng, Z.; Chakraborty, R.; Tan, M.; Fung, G.; Li, Y.; Singh, V. Nyströmformer: A nyström-based algorithm for approximating self-attention. In Proceedings of the Proceedings of the AAAI Conference on Artificial Intelligence, 2021, Vol. 35, pp. 14138–14148.
100. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* **2018**.
101. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* **2020**, *21*, 1–67.
102. Andrew Howard, Mark Sandler, G.C. Searching for MobileNetV3. *arXiv preprint arXiv:1905.02244* **2019**.
103. Gong, C.; Wang, D.; Li, M.; Chandra, V.; Liu, Q. Vision transformers with patch diversification. *arXiv preprint arXiv:2104.12753* **2021**.
104. Jégou, H.; Douze, M.; Schmid, C. Product Quantization for Nearest Neighbor Search. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *33*, 117–128.
105. Craswell, N.; Mitra, B.; Yilmaz, E.; Campos, D.; Voorhees, E.M. Overview of the trec 2019 deep learning track. *arXiv preprint arXiv:2003.07820* **2020**.
106. Li, Y.; Hu, J.; Wen, Y.; Evangelidis, G.; Salahi, K.; Wang, Y.; Tulyakov, S.; Ren, J. Rethinking Vision Transformers for MobileNet Size and Speed. *arXiv preprint arXiv:2212.08059* **2022**.

 doi:10.20944/preprints202503.0590.v1

107. Li, Y.; Wu, C.Y.; Fan, H.; Mangalam, K.; Xiong, B.; Malik, J.; Feichtenhofer, C. Mvitv2: Improved multiscale vision transformers for classification and detection. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 4804–4814.
108. Tonellotto, N.; Macdonald, C. Query Embedding Pruning for Dense Retrieval. *CoRR* **2021**, *abs/2108.10341*, [2108.10341].
109. Nogueira, R.; Jiang, Z.; Lin, J. Document Ranking with a Pretrained Sequence-to-Sequence Model, 2020, [arXiv:cs.IR/2003.06713].
110. Ouyang, L.; Qu, Y.; Zhou, H.; Zhu, J.; Zhang, R.; Lin, Q.; Wang, B.; Zhao, Z.; Jiang, M.; Zhao, X.; et al. OmniDocBench: Benchmarking Diverse PDF Document Parsing with Comprehensive Annotations, 2024, [arXiv:cs.CV/2412.07626].
111. Voorhees, E.M.; Harman, D.K. *TREC: Experiment and Evaluation in Information Retrieval (Digital Libraries and Electronic Publishing)*; The MIT Press, 2005.
112. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. Language models are unsupervised multitask learners. *OpenAI blog* **2019**.
113. Liu, H.; Yan, W.; Zaharia, M.; Abbeel, P. World Model on Million-Length Video And Language With RingAttention, 2024, [arXiv:cs.LG/2402.08268].
114. Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; Zhang, C. Learning efficient convolutional networks through network slimming. In Proceedings of the Proceedings of the IEEE international conference on computer vision, 2017, pp. 2736–2744.
115. Jang, Y.; Song, Y.; Yu, Y.; Kim, Y.; Kim, G. Tgif-qa: Toward spatio-temporal reasoning in visual question answering. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2758–2766.
116. Li, H. *Learning to Rank for Information Retrieval and Natural Language Processing*; Morgan & Claypool Publishers, 2011.
117. Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; Jégou, H. Training data-efficient image transformers & distillation through attention. In Proceedings of the International conference on machine learning. PMLR, 2021, pp. 10347–10357.
118. Clinchant, S.; Gaussier, E. Information-based Models for Ad Hoc IR. In Proceedings of the Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY, USA, 2010; SIGIR '10, pp. 234–241. https://doi.org/10.1145/1835449.1835490.
119. Cai, H.; Li, J.; Hu, M.; Gan, C.; Han, S. EfficientViT: Lightweight Multi-Scale Attention for High-Resolution Dense Prediction. In Proceedings of the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 17302–17313.
120. Câmara, A.; Hauff, C. Diagnosing BERT with Retrieval Heuristics. In Proceedings of the Advances in Information Retrieval - 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14-17, 2020, Proceedings, Part I; Jose, J.M.; Yilmaz, E.; Magalhães, J.; Castells, P.; Ferro, N.; Silva, M.J.; Martins, F., Eds. Springer, 2020, Vol. 12035, *Lecture Notes in Computer Science*, pp. 605–618. https://doi.org/10.1007/978-3-030-45439-5_40.
121. Lit, Z.; Sun, M.; Lu, A.; Ma, H.; Yuan, G.; Xie, Y.; Tang, H.; Li, Y.; Leeser, M.; Wang, Z.; et al. Auto-ViT-Acc: An FPGA-aware automatic acceleration framework for vision transformer with mixed-scheme quantization. In Proceedings of the 2022 32nd International Conference on Field-Programmable Logic and Applications (FPL). IEEE, 2022, pp. 109–116.
122. Zhang, L.; Xu, D.; Arnab, A.; Torr, P.H. Dynamic graph message passing networks. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 3726–3735.
123. Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.; Savarese, S. Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression, 2019, [arXiv:cs.CV/1902.09630].
124. Chen, T.; Li, L.; Sun, Y. Differentiable product quantization for end-to-end embedding compression. In Proceedings of the International Conference on Machine Learning. PMLR, 2020, pp. 1617–1626.
125. Wang, X.; Zhang, H.; Huang, W.; Scott, M.R. Cross-Batch Memory for Embedding Learning. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 6388–6397.
126. Nogueira, R.; Cho, K. Passage Re-ranking with BERT, 2019, [arXiv:cs.IR/1901.04085].

 doi:10.20944/preprints202503.0590.v1

disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.