FACULTY OF ENGINEERING AND ARCHITECTURE

Adaptive Vision Transformers for Efficient Processing of Video Data in Automotive Applications

Wout Verbiest
Student number: 02016863

Supervisors: Prof. dr. ir. Pieter Simoens, Prof. dr. Sam Leroux

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Information Engineering Technology

Academic year 2024-2025

GHENT UNIVERSITY

Acknowledgment

I would like to express my gratitude to Prof. dr. ir. Simoens and Prof. dr. Leroux, my supervisors, for their guidance, constructive feedback, and support throughout the course of this research. Their guidance and advice have played a key role in the success of this master's dissertation.

This work was made possible by the facilities provided by the University of Ghent and Imec, and am thankful for their support throughout my academic journey.

Permission to use and reference this master's dissertation

Explanation regarding the master’s thesis and the oral presentation

This master’s dissertation is part of an exam. Any comments formulated by the assessment committee during the oral presentation of the master's dissertation are not included in this text.

Abstract

Abstract (English): Modern vehicles rely on real-time environmental perception enabled by a combination of sensor technologies, including cameras, lidar, and radar, which generate vast amounts of data. Processing this data to support split-second decision-making poses challenges in accuracy, speed, and energy efficiency, which is especially critical in electric vehicles where computational demand can directly impact range. This thesis investigates efficient adaptive computation techniques in vision transformers for processing video data from vehicles. The research centers on designing and evaluating an adaptive vision transformer architecture for semantic segmentation, which dynamically adjusts computational resources based on environmental complexity by reusing tokens from a token database. Using a real-world dataset, the adaptive architecture is tested across metrics including processing speed, computational load and accuracy. The study's findings contribute to improving the operational efficiency in automotive applications.

Abstract (Dutch): Moderne voertuigen vertrouwen op realtime waarnemingen van de omgeving door een combinatie van sensortechnologieën, zoals camera's, lidar en radar, die enorme hoeveelheden data genereren. Het verwerken van deze data om beslissingen mogelijk te maken in een fractie van een seconde, zorgt voor uitdagingen op het gebied van nauwkeurigheid, snelheid en energie-efficiëntie. Deze kunnen uiterst belangrijk zijn bij batterijgedreven voertuigen waar rekenkracht een directe invloed heeft op het rijbereik. Deze masterproef onderzoekt efficiënte adaptieve technieken in vision transformers voor het verwerken van video's uit voertuigen. Het onderzoek concentreert zich op het ontwerpen en evalueren van een adaptieve vision transformer architectuur voor semantische segmentatie, die dynamisch computationele middelen aanpast aan de complexiteit van de omgeving door tokens te herbruiken uit een token databank. Met behulp van een realistische dataset wordt de adaptieve architectuur getest op onder andere verwerkingssnelheid, computationele belasting en nauwkeurigheid. De bevindingen van het onderzoek dragen bij aan het verbeteren van de operationele efficiëntie in mobiliteitstoepassingen.

Keywords: Semantic Segmentation, Adaptive Vision Transformer, Adaptive Computation, Energy Efficiency, Automotive Applications

Extended Abstract (English)

Verbiest, Wout
Master of Science in Information Engineering Technology
UGent
Ghent, Belgium
verbiest.wout@gmail.com

Prof. dr. ir. Simoens, Pieter
IDLab
Imec
Ghent, Belgium
Pieter.Simoens@UGent.be

Prof. dr. Leroux, Sam
IDLab
Imec
Ghent, Belgium
Sam.Leroux@UGent.be

***Abstract*—This study researches Efficient Adaptive Computation Techniques in Vision Transformers tailored for processing video in automotive applications. The aim is to address critical challenges in the trade-off between accuracy, speed, and energy efficiency, which is particularly important for electric vehicles where computational demand can affect range. The study explores an adaptive architecture for Semantic Segmentation with Vision Transformers that reuses previously computed tokens, significantly reducing computational load while maintaining accuracy.**

***Index Terms*—Semantic Segmentation, Adaptive Vision Transformer, Adaptive Computation, Energy Efficiency, Automotive Applications**

I. Introduction

Modern vehicles increasingly rely on real-time environmental perception supported by a combination of cameras, lidar, and radar. These sensors produce large volumes of data requiring split-second processing to enable critical decision-making. The need for efficient computation is especially urgent in electric vehicles, where resource-intensive operations can directly impact driving range. This work investigates how vision transformers can be optimized to process video data adaptively, balancing computational load and output quality.

II. Methodology

The paper introduces a new architecture for a Vision Transformer encoder [1] that uses a Token Reduction and Reconstruction block that dynamically adjusts computational resources based on environmental complexity. Tokens are pruned through a matching algorithm that matches with a token database and are later replaced with processed counterparts, significantly easing the encoder's computational load. Tokens are reconstructed after the encoder and decoded by the ATM-decode head from SegVit [2].

III. Results

Extensive testing on real-world datasets revealed:

- A 37% reduction in median inference time of the encoder.
- A 49% reduction in median required FLOPS.
- A median pixel-wise loss of 4.62% compared to the baseline model.

These results show the model's ability to achieve computational efficiency without compromising segmentation accuracy, demonstrating its suitability for real-time automotive applications.

IV. Limitations and Future Work

The study's dataset was limited in diversity, lacking scenarios with varied weather conditions, lighting, and vehicle speeds. Ground truth for video sequences was also unavailable, preventing absolute accuracy measurements. Future research should:

- Evaluate the architecture under diverse environmental conditions
- Extend the approach to multi-input configurations, enabling simultaneous processing of multiple cameras

V. Contributions

This research advances understanding of adaptive computation in Vision Transformers and highlights their potential for energy-efficient, real-time applications. The study demonstrates how fixed architectures can be reimagined for specific use cases, providing a pathway for resource-constrained yet effective deployments in modern vehicles.

References

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[2] B. Zhang, Z. Tian, Q. Tang, X. Chu, X. Wei, C. Shen, and Y. Liu, "Segvit: Semantic segmentation with plain vision transformers," *NeurIPS*, 2022.

Extended Abstract (Dutch)

Verbiest, Wout
Master of Science in de industriële wetenschappen: informatica
UGent
Ghent, Belgium
verbiest.wout@gmail.com

Prof. dr. ir. Simoens, Pieter
IDLab
Imec
Ghent, Belgium
Pieter.Simoens@UGent.be

Prof. dr. Leroux, Sam
IDLab
Imec
Ghent, Belgium
Sam.Leroux@UGent.be

***Abstract*—Dit onderzoek richt zich op Efficiënte Adaptieve Computationtechnieken in Vision Transformers voor het verwerken van video in mobiliteitstoepassingen. Het doel is om uitdagingen in de afweging tussen nauwkeurigheid, snelheid en energie-efficiëntie aan te pakken, wat vooral belangrijk is voor elektrische voertuigen waar computationele belasting de range kan beïnvloeden. De masterproef onderzoekt een adaptieve architectuur voor semantische segmentatie met Vision Transformers die eerder berekende tokens hergebruikt, waardoor de computationele belasting aanzienlijk wordt verminderd met een klein verlies in nauwkeurigheid.**

***Index Terms*—Semantic Segmentation, Adaptive Vision Transformer, Adaptive Computation, Energie Efficiënt, Mobiliteitstoepassingen**

I. Introductie

Moderne voertuigen vertrouwen steeds meer op realtime waarneming van de omgeving door een combinatie van camera's, lidar en radar. Deze sensoren produceren grote hoeveelheden data die in een fractie van een seconde verwerkt moeten worden om beslissingen mogelijk te maken. De behoefte aan efficiënte berekeningen is vooral belangerijk in elektrische voertuigen, waar intensieve bewerkingen een directe invloed kunnen hebben op de range. In deze masterproef wordt onderzocht hoe Vision Transformers geoptimaliseerd kunnen worden om video adaptief te verwerken, waarbij de computationele belasting en de uitvoerkwaliteit in balans worden gebracht.

II. Methodiek

Deze paper introduceert een nieuwe architectuur voor een Vision Transformer encoder [1] die gebruikmaakt van een tokenreductie- en reconstructieblok dat de computationele middelen dynamisch aanpast op basis van de complexiteit van de omgeving van het voertuig. Tokens worden verwijderd door middel van een matching-algoritme dat tokens matched met een token-database en worden later vervangen door bewerkte tegenhangers, waardoor de computationele belasting van de encoder aanzienlijk wordt verminderd. Tokens worden na de encoder gereconstrueerd en gedecodeerd door de ATM-decode head van SegVit [2].

III. Resultaten

Testen op de datasets hebben aangetoond dat:

- Een verlaging van 37% in de mediaan van de inference tijd van de encoder.
- Een verlaging van 49% in de median van de nodige FLOPs.
- Een mediaan pixel-wise loss van 4,62% loss in vergelijking met het baseline model.

Deze resultaten tonen aan dat het model in staat is om computationele efficiëntie te bereiken zonder afbreuk te doen aan de nauwkeurigheid, wat de geschiktheid voor real-time automobieltoepassingen aantoont.

IV. Limitaties en Future Work

De dataset van het onderzoek was beperkt in diversiteit, omdat er geen scenario's waren met sterk gevarieerde weersomstandigheden, verlichting en voertuigsnelheden. De ground truth voor videosequenties was ook niet beschikbaar, waardoor absolute nauwkeurigheidsmetingen niet mogelijk waren. Toekomstig onderzoek moet zich focussen op:

- Het evalueren van de architectuur op meer diverse omgevingen.
- De implementatie uitbreiden voor multi-input, om zo meerdere cameras tegelijk te kunnen berekenen.

V. Contributies

Dit onderzoek vergroot de kennis van adaptieve berekeningen in Vision Transformers en benadrukt hun potentieel voor energiezuinige, real-time toepassingen. Het onderzoek laat zien hoe vaste architecturen opnieuw kunnen worden ontworpen voor specifieke gebruikssituaties.

References

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[2] B. Zhang, Z. Tian, Q. Tang, X. Chu, X. Wei, C. Shen, and Y. Liu, "Segvit: Semantic segmentation with plain vision transformers," *NeurIPS*, 2022.

Contents

List of Figures

List of Tables

List of Acronyms

AI Artificial Intelligence.

ATM Attention To Mask.

CNN Convolutional Neural Network.

FLOPS Floating Point Operations per Second.

LiDAR Light Detection and Ranging: Remote sensing technology that uses laser light to measure distances..

NLP Natural Language Processing.

RGB (Red,Blue,Green): A color model used to represent and encode color in digital images..

ToRe Token Reducing.

ViT Vision Transformer.

List of Code Fragments

1

Introduction

1.1 Background

The advancement of automotive technology, from early concepts to today's sophisticated systems, represents one of the greatest technological efforts of modern times. The first significant strides in advanced vehicle technologies were made in the 1980s, with innovative projects such as Carnegie Mellon University's Navlab[11]. Over the years, developments in artificial intelligence, sensor technology, and computing power have transformed the automotive industry, leading to transformative applications across a range of vehicle types, including conventional cars, electric vehicles, and other kinds of vehicles.

The core of any modern automotive system is its ability to perceive and interpret its surroundings in real time. This is made possible by a set of sensors, including cameras, lidar and radar, which capture huge amounts of data. The real-time processing of this sensor data is crucial for making split-second decisions such as detecting obstacles, recognizing traffic signs and predicting the movements of other vehicles and pedestrians. However, the challenging aspect here is that this processing must be both highly accurate and extremely fast to ensure vehicle safety and reliability.

Energy efficiency is a critical concern for advanced automotive applications, particularly electric vehicles. Processing high-resolution sensor data from multiple sources simultaneously is a computationally intensive task. The computational requirements for processing this data can lead to significant energy consumption, having a direct impact on the vehicle's range, thereby reducing the overall efficiency of the vehicle. Optimizing data processing pipelines not only reduces energy consumption but also enhances vehicle performance, contributing to the broader goals of efficiency and sustainability in the automotive sector.

1.2 Problem Statement

Automotive technologies, including self-driving cars, stand at the forefront of a transportation revolution, set to transform our connection with mobility. Beyond more convenience, they promise to fundamentally alter the way

we perceive and utilize transportation infrastructure. At the heart of this technological marvel lies a complex coordination between real-time sensor data processing and decision-making algorithms. However, the enormous complexity of this task demands substantial computational and energy resources, which poses a significant challenge considering the constraints of in-vehicle computation and energy storage.

To ensure the viability of these models, it becomes crucial to develop strategies that optimize the processing pipeline to the vehicle's environment. For example, in a scenario where a vehicle is standing still in a rural environment, it demands fewer computational resources than when navigating a busy urban intersection. By introducing adaptive computation techniques in these deep neural networks, we can dynamically adjust computational resources based on the complexity of incoming data. This implies that the system can selectively allocate resources, customizing them to suit the complexity of the vehicle's environment. This not only enhances computational efficiency but also improves energy efficiency through optimized computations.

1.3 Research Objectives

The primary objective of this thesis is to design, develop and evaluate an adaptive computational technique tailored for the processing of sensor data, such as camera, lidar or radar inputs, in vehicles. The initial phase consists of an in-depth review and analysis of state-of-the-art methodologies applied in sensor data processing. These are the techniques currently utilized within driving systems, as well as techniques borrowed from diverse domains. Subsequently, the study will explore the customization of these existing models and techniques to enable adaptive computation, thereby optimizing performance in response to varying complexities of input data. This adaptability of the model will be achieved through implementing a mechanism capable of dynamically adjusting the architecture. This adjustment may include altering the size or depth of neural networks, as well as fine-tuning input factors such as token pruning or merging, to better align with the complexity of the input data. The subsequent phase involves implementation, testing and benchmarking of the developed adaptive computation model using real-world sensor data sourced from autonomous vehicles. Open-source datasets, such as A2D2 dataset [1] , Waymo Open Dataset [12] and CityScapes dataset [13] will serve as the foundation of our evaluations and validations. The evaluation phase will evaluate the effectiveness of the developed model across the key performance metrics such as accuracy, processing speed and energy consumption. Through this evaluation, the objective is to assess the viability and effectiveness of the adaptive computation techniques within the context of driving

systems.

1.4 Research Question

How can adaptive computation techniques improve the energy efficiency of semantic segmentation algorithms using vision transformers in vehicles and enhance real-time performance while reducing computational load?

1.5 Significance of the Study

Semantic segmentation algorithms are crucial for enhancing the accuracy of environment perception. By enabling vehicles to better understand and classify their surroundings, these algorithms contribute to safer navigation and decision-making.

The efficiency of these algorithms is vital for real-time data processing, which is a fundamental requirement for autonomous vehicles. Fast processing speeds allow for quicker decision-making, which is essential for ensuring safety on the road. For instance, if a vehicle is traveling at 120 km$/$h, and it takes one second for the segmentation model to process the data, the car would have already moved over 33 m from the point where the data was originally captured. This delay could lead to potential risks if the system is not able to react quickly.

Moreover, reducing the computational load of semantic segmentation offers two significant benefits: it either lowers energy consumption, which in the case of electric vehicles, has a direct impact on the vehicle's range, or enables the processing of more data in real time, which enhances the overall safety of the vehicle.

2

Literature Review

2.1 Semantic Segmentation

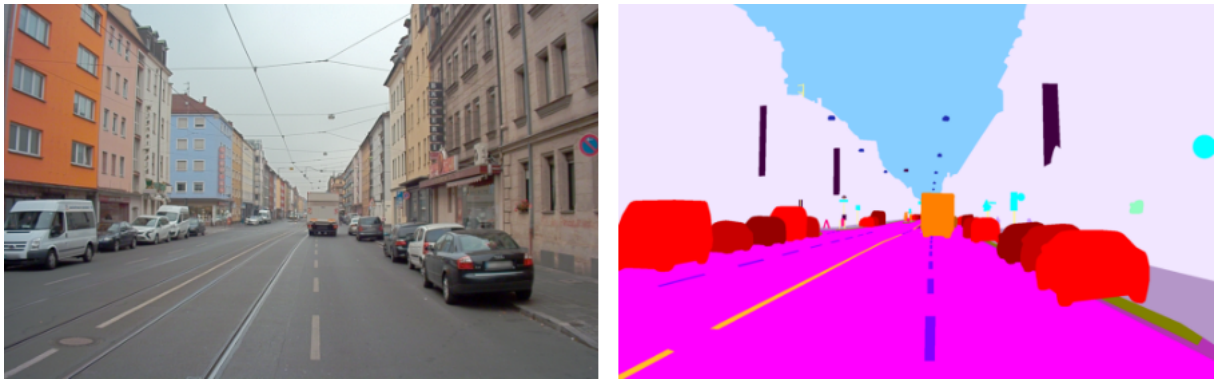

Figure 2.1: Example of Semantic Segmentation from A2D2[1]. (left) RGB image. (right) Color Coded ground truth of the Segmentation

Semantic segmentation is a task in computer vision that involves classifying each pixel of an input image into a specific category. This task extends image classification from the image level to the pixel level, requiring a more detailed analysis. [14] The main benefit to using semantic segmentation in vehicles is scene understanding[15]. Achieving high-accuracy semantic segmentation in a short time enables the vehicle to better comprehend its surrounding environment, allowing it to make appropriate decisions in real time.

Initially, Fully Convolutional Networks [16, 14] were the dominant approach for semantic segmentation. Early advancements focused on expanding the receptive field [17, 18] and improving contextual understanding through methods like dilated convolutions [19]. This allowed the models to capture multi-scale contextual information.

To further enhance the capability of semantic segmentation models, the introduction of attention mechanisms marked a significant milestone. Attention-based methods [20, 21, 22] have replaced the traditional convolutional and pooling operations with attention layers that capture long-range dependencies more effectively.

More recent methods have explored the effectiveness of Transformer-based architectures for semantic segmentation [4, 5, 6, 23]. These architectures leverage the self-attention mechanism to capture global context and dependencies in an image, resulting in improved segmentation performance [24]. While these transformer-based models have demonstrated superior accuracy, they remain computationally intensive.

2.2 Transformers

Self-attention-based architectures, in particular the transformer models [2], have become the standard models in natural language processing, especially in Long Sequence Models [25, 26].

Originally introduced as a sequence-to-sequence model for machine translation [27], Transformers are now deployed in a variety of domains within machine learning. A common approach is to first pre-train these models on a large text corpus, after which they are fine-tuned on a smaller, task-specific dataset [28, 29]. The scalability of Transformers allows them to be trained on extremely large datasets with more than 100 billion parameters [30, 31].

Besides language-related applications such as NLP, Transformers are now being used in other domains [32], such as audio processing, e.g. speech recognition [33, 34], and computer vision for both static images [3, 35] and videos [36, 37].

2.2.1 Architecture

The transformer architecture uses an encoder-decoder structure. The encoder transforms an input sequence of symbol representations $(x_1, ..., x_n)$ into a sequence of continuous representations, denoted as $z = (z_1, ..., z_n)$. The decoder then uses this sequence, z, to produce an output sequence of symbols $(y_1, ..., y_m)$ one element at a time. The model operates in an auto-regressive manner, meaning that each step, It uses the symbols generated in the previous steps as part of the input for generating the next symbol. [2]

The encoder, depicted on the left in Figure 2.2, consists of N identical layers, each containing two sub-layers. The first sub-layer is a multi-head self-attention mechanism and the second is a fully connected feed-forward network. After each sub-layer, a residual connection is applied [38], which is then followed by layer normalization [39].

The decoder, similar to the encoder, consists of N identical layers, as shown on the right in Figure 2.2. Besides the multi-head attention and Feed Forward Neural Network from the encoder, the decoder also includes a 3rd sub-layer, a multi-head attention executes on the output of the encoder. As with the encoder, residual connections are applied, with subsequent layer normalization. There is also a modification to the self-attention sub-layer that ensures that each position in a sequence only considers relevant, i.e. prior, information during processing or

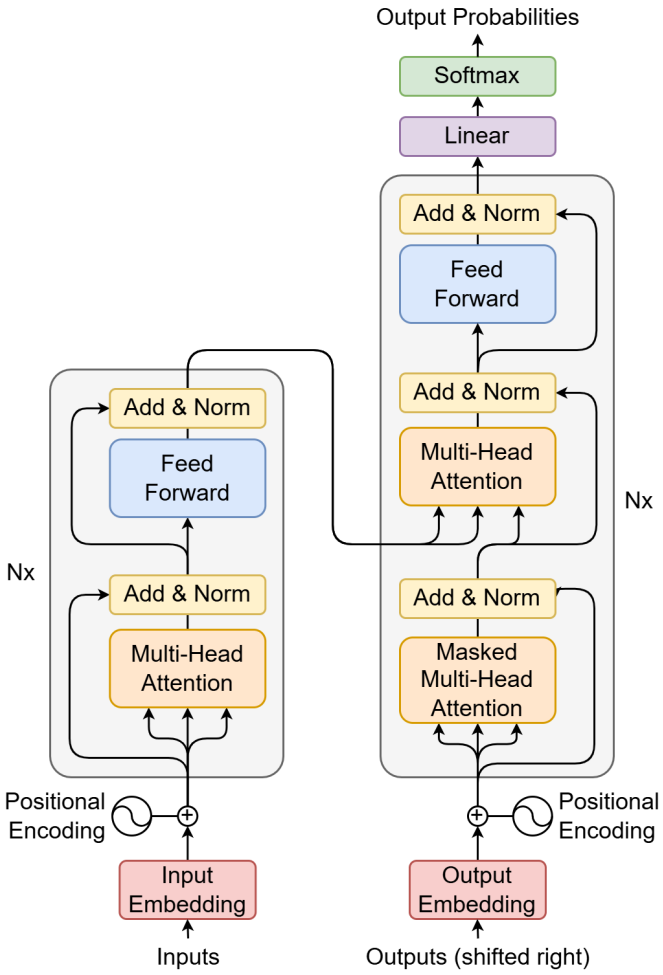

Figure 2.2: Architecture of a vanilla transformer model, adapted from [2]

generating the sequence. [2]

2.2.2 Self-Attention mechanism

An attention function is a mechanism that maps a query to an associated set of key-value pairs to generate an output. In this process, the query, the keys, the values and the final output are all vectors. The output is calculated by making a weighted sum of the values, where the weight assigned to each value is determined by the degree of similarity, or compatibility, between the query and the corresponding key. In essence, the attention function evaluates how well each key matches the query and uses that information to proportionally weigh the values and compose a final output. This process allows the model to emphasize or weaken specific elements in the input, depending on their relevance to the query.

The attention function applied in a transformer is known as Scaled Dot-Product Attention (Figure 2.3). This type

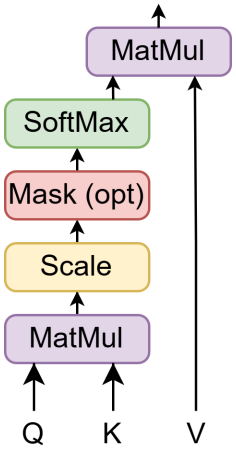

Figure 2.3: Scaled Dot Product Attention, adapted from [2]

of attention is a variation of the traditional dot-product attention [16], where an additional scaling factor of $\frac{1}{\sqrt{d_k}}$ is added to refine the calculation. The input for this function consists of queries and keys, both with dimension of d_k, and values with a dimension of d_v. The process starts by calculating the dot product between the query and all keys. The result is then divided by the root of d_k to scale the values. To obtain the weights of the values, a SoftMax function is applied. This SoftMax function normalizes the weights, allowing them to be used to calculate a weighted sum of the values, which performs the final output. This scaling mechanism is crucial because it prevents the dot products from becoming too large, which could affect the convergence of the model. [2]

In practice, the attention function is calculated on a collection of queries that are collectively hosed in a matrix Q. Similarly, the keys and values are grouped together in matrices K and V respectively. Consequently, the attention calculations can be performed for all queries at the same time, making for more efficient processing. The weights are calculated by applying a SoftMax function to the division between the dot product of Q and K^T and $\sqrt{d_k}$. The output is computed by the dot product of the weight function and the values V. This matrix contains the weighted results for each query, based on the corresponding keys and values. [2]

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

Instead of performing only one attention function with d_{model}-dimensional keys, values and queries, it has proved more advantageous to project the queries, keys and values h times linearly to d_k, d_k and d_v dimensions respectively. Each of these projected versions is then subjected to the attention function in parallel, resulting in d_v-dimensional output values. These output values are then merged and projected again, leading to the final values as shown in Figure 2.4.

Multi-head attention allows the model to simultaneously pay attention to different pieces of information from

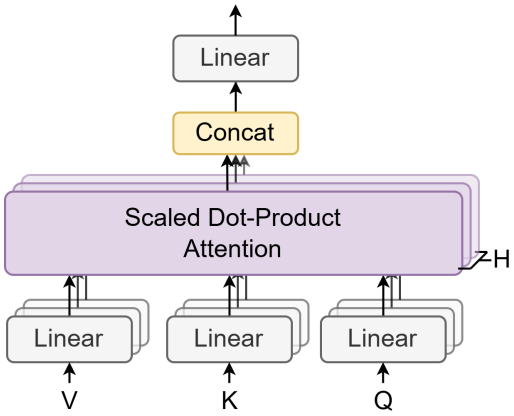

Figure 2.4: Architecture of the multi-head attention architecture, adapted from [2]

different representation spaces and different positions. Whereas a single attention head is limited in its capacity to integrate information, multi-head attention allows the model to combine various perspectives without the averaging of single attention layer suppressing variation.

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \\ \text{where head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned}$$

The projections are determined by parameter matrices, such as $W_i^Q \in R^{d_{model} \times d_k}$ for queries, $W_i^K \in R^{d_{model} \times d_k}$ for keys and $W_i^V \in R^{d_{model} \times d_k}$ for values. There are $h = 8$ parallel attention layers, commonly referred to as heads.

2.2.3 Computational Complexity

Transformers are a computationally expensive models. The complexity of the self-attention mechanism within each layer is $O(n^2 \cdot d)$, where n represents the sequence length and d the representation dimension. [2] This quadratic complexity in terms of sequence length results in a high computational overhead. The inefficiency of the transformer makes it unsuitable for real-time or resource constrained applications without optimizations.

2.3 Vision Transformers

Vision Transformers (ViTs) have recently gained significant attention as a groundbreaking architecture within the field of Computer Vision. Building on the foundational principles of the standard transformer model that has been widely successful in natural language processing. The ViT architecture introduced by Dosovitskiy et al. leverages

this self-attention mechanism to process image data directly without the need for convolutional layers such as in CNN models.

Traditionally, Convolutional Neural Networks (CNNs) (Fukushima, 1980) have been the dominant approach for tasks such as image classification, segmentation, ... Dominating in the industry for many years. However, the introduction of Vision Transformers is shifting this, offering a compelling alternative that challenges the long-standing dominance of CNNs. Whit their ability to model global relationships within images more effectively, Vision Transformers are setting new benchmarks in the field and are poised to redefine how visual data is processed and understood.[3, 40]

The motivation behind ViTs stems from the limitations of CNNs, particularly their reliance on inductive biases like locality and translation invariance, which, while effective, can also restrict the model's ability to capture global dependencies in the data. The Transformer model, known for its ability to model long-range dependencies in sequential data, was adapted for vision tasks by treating images as sequences of patches. Each image is divided into a grid of fixed-size patches, which are then linearly embedded and processed as tokens by the Transformer encoder. This novel approach enables ViTs to capture global context more effectively than CNNs. [3]

2.3.1 Architecture

In the standard transformer model discussed in section 2.2, the input consists of a 1-dimensional sequence of tokens. To adapt this model for 2D image data, a conversion of image $x \in R^{H \times W \times C}$ into a sequence of flattened 2-dimensional patches $x_p \in R^{N \times (P^2 \cdot C)}$ where (H, W) represents the dimensions of the original image and C denotes the number of color channels. (P, P) specifies the size of each patch, usually $P = 16$ and $N = \frac{H \times W}{P^2}$ indicates the total number of patches. This total number N also determines the effective sequence length for the Transformer model. The transformer maintains a consistent latent vector size d throughout its layers. Consequently, the patches are flattened and then mapped to d dimensions using a trainable linear projection. The result is referred to as the patch embeddings. An additional position embedding is added to these patch embeddings to obtain position information. This combination is the input for the Transformer Encoder. [3]

A learnable embedding is added to the beginning of the sequence of patches, denoted as $z_0^0 = x_{\text{class}}$. The state of this embedding at the output of the Transformer encoder, z_L^0, is used as the image representation $y = LN(z_L^0)$, with L representing the amount of layers, in essence the amount of times the Transformer Encoder is executed.[3]

The transformer encoder block in a Vision Transformer follows the design of the original transformer proposed by [2].

Depending on the specific task the model is designed for, an appropriate head is added to the Transformer Encoder. This head is usually a Multi Layer Perceptron (MLP) and can be tailored to different tasks, such as classification, object detection and semantic segmentation.

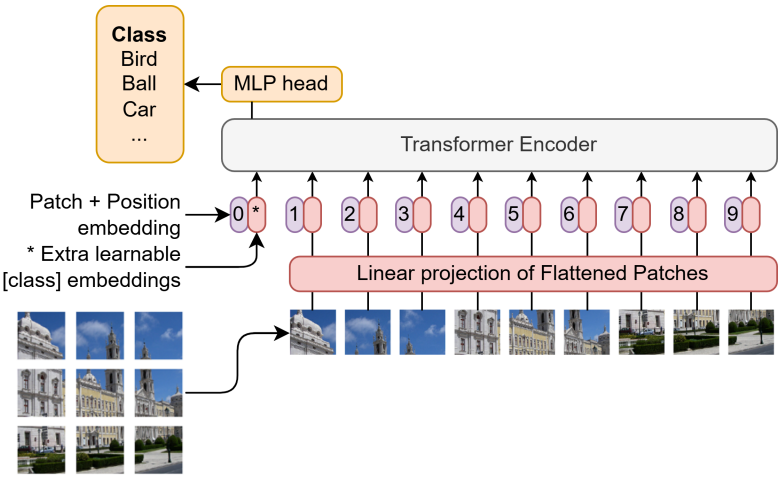

Figure 2.5: Architecture of a Vision Transformer, adapted from [3]

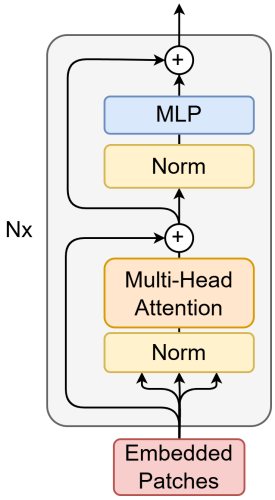

Figure 2.6: Architecture of a Vision Transformer Encoder, adapted from [3]

2.3.2 Scalability

One of the critical advantages of ViTs is their scalability. When processing higher-resolution images, the same patch-size is maintained, leading to an increased effective sequence length. While the Vision Transformer can accommodate sequences of varying lengths, the pre-trained position embeddings might lose their relevance at sizes extremely different from the pre-trained data. This scalability, however, also comes with increased computational requirements, making ViTs more resource-intensive compared to CNNs.

2.4 Semantic Segmentation using Vision Transformers

Several transformer-based architectures have been proposed to address the unique requirements of semantic segmentation. SegFormer [5] introduces a hierarchical transformer with a lightweight decoder, while SETR [41], Segmenter [4] and SegViT [6] refine vision transformers by integrating specialized decoding strategies and hierarchical feature aggregation.

2.4.1 Segmentation Transformer

Segmentation Transformer [41], also known as SETR, uses an encoder-decoder architecture. The encoder is a vision transformer backbone existing of 24 transformer layers, and proposes 3 architectures for the decoder, also referred to as the heads.

The Naïve upsampling decoder head maps the transformer feature maps to the dimensions corresponding to the

number of categories. It consists of a two-layer network with the following structure: A 1x1 convolutional layer followed by synchronized batch normalization with a ReLU activation function, another 1x1 convolutional layer. Following these convolutional operations, the output is bilinearly upsampled to match the full image resolution. This upsampled output is then fed into a classification layer, where pixel-wise classification is performed. [41]

The Progressive upsampling decoder offers an alternative to the traditional one-step upscaling approach, which can often lead to noisy predictions. It is a method that interleaves convolutional layers with upsampling operations. To further minimize potential adversarial effects and maintain stability in the upscaling process, the upsampling factor is limited to a maximum of 2x at each step. This ensures smoother transitions. [41]

The Multi-level feature aggregation draws inspiration from the pyramid network [42]. While it shares the concept of combining features from multiple levels to enhance representation, it fundamentally diverges from the pyramid structure, since this approach maintains a uniform resolution across all layers, in contrast to the varying resolutions used in traditional approach, creating the hierarchical pyramid shape. This design choice allows for a consistent representation of features throughout the network, resulting in more efficient and effective feature extraction without the complexity of managing multiple resolution levels. [41]

2.4.2 Segmenter

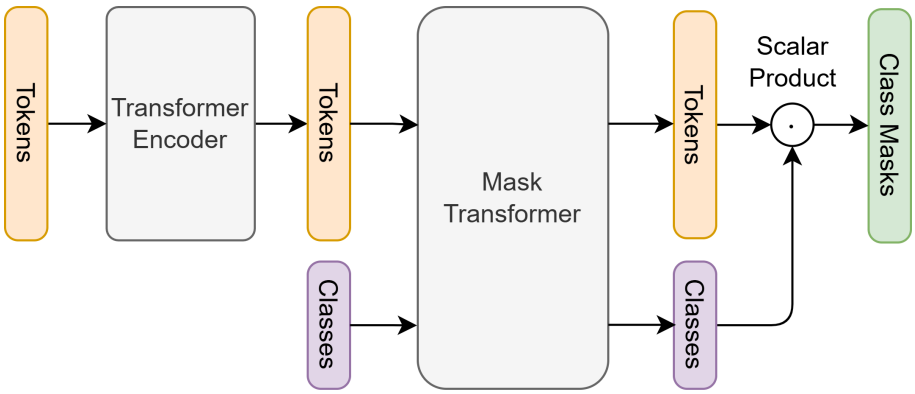

Figure 2.7: Architecture of Segmenter, adapted from [4]

Segmenter [4] is an encoder-decoder architecture that differentiates itself from other semantic segmentation models by employing a transformer in both the encoder and decoder.

The encoder utilises a standard vision transformer backbone. Despite the backbone being pre-trained for image-classification, it was further fine-tuned on a semantic segmentation dataset. [4, 24]

As a decoder, Segmenter introduces a new mask transformer. This allows it to avoid using traditional methods such as convolutional models throughout the process. Convolutional models tend to be inefficient with processing global context of a picture and might therefore give a less ideal result. The transformer-based approach, in

contrast, offers better processing of the global context, although at a cost of higher computational complexity, since the multi-head self attention mechanism has quadratic scaling. [4]

In the decoder, patch encodings generated by the encoder are transformed into a segmentation map. This is done by bilinear interpolation to pixel level, after which SoftMax is applied to obtain the final segmentation map. To achieve this, the mask transformer uses a set of learnable class embeddings specific to each semantic class. [4]

Besides semantic segmentation, the model can also be applied to panoptic segmentation. This is a hybrid form of segmentation that combines both semantic segmentation and instance segmentation. For this, the class-embeddings in the model are replaced by object-embeddings. [24]

2.4.3 SegFormer

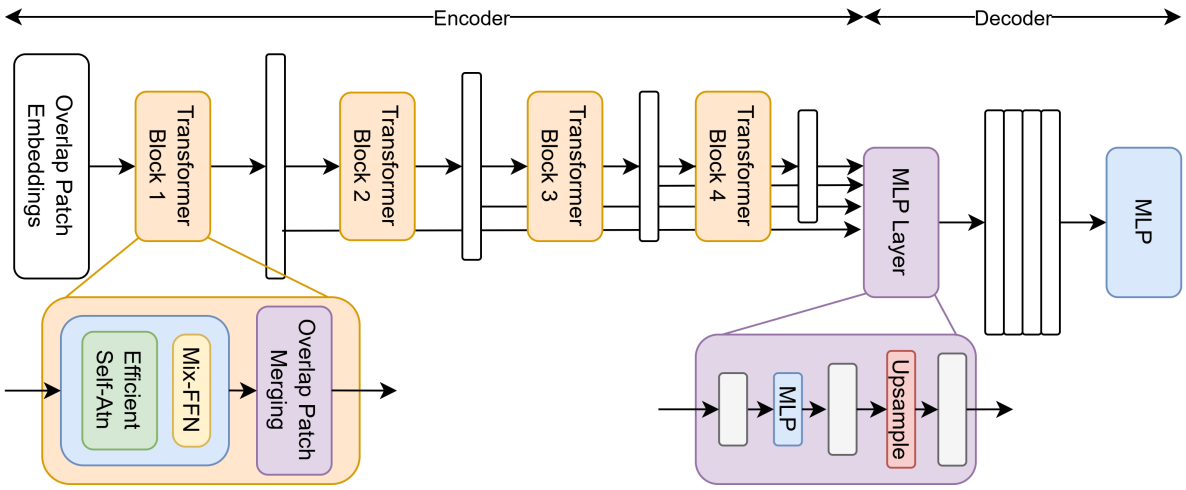

Figure 2.8: Architecture of a SegFormer, adapted from [5]

SegFormer [5] is a transformer-based architecture for Semantic Segmentation that combines a hierarchical transformer encoder with a lightweight Multilayer Perceptron decoder. Unlike the standard ViT [3] that typically uses patches of size 16 ×16, SegFormer uses smaller patches of size 4 ×4 as input. This difference allows for the capturing of more details information from the image.

The hierarchical Transformer processes these smaller patches to generate multi-level feature maps. Unlike traditional ViTs, which produce a single resolution feature map, SegFormer's encoder is designed to output a CNN-like multi-level feature representation. This means it generates a range of features, from high-resolution to low-resolution, enabling the model to capture both fine-grained details and broader contextual information within an image. [5]

These multi-level features are then fed into an all-MLP decoder, which produces the final segmentation mask. The use of an MLP-based decoder, rather than a more complex structure, ensures that the architecture remains

lightweight while delivering high segmentation accuracy. [5]

2.4.4 SegVit

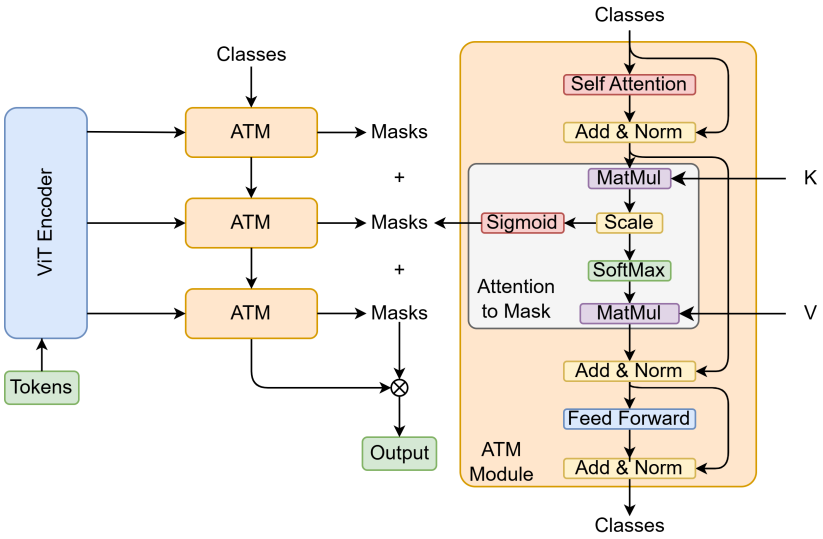

Figure 2.9: General Architecture of SegViT, adapted from [6]

SegVit [6] is an advanced model for semantic segmentation that, similar to Segmenter [4], follows an encoder-decoder structure in which both the encoder and decoder within the model are based on a transformer architecture.

SegVit employs a standard Vision Transformer (ViT) backbone as its encoder. The architecture consists of a multi-head self-attention and feed-forward network, with repeated layer normalization and residual connections. Notably, SegVit does not introduce any additional modules or modifications in the encoder, and the number of tokens remains constant across all layers, which keeps the encoder simple and consistent while processing input data. [6]

SegVit introduces a new approach to semantic segmentation by using a transformer-based decoder enriched with a step called 'Attention to Mask' (ATM). A block within the decoder is known as an ATM module. [6]

The input for an ATM module consists of two important components: class embeddings and the base. Class embeddings are a sequence of N tokens, with each token representing a specific class. They provide an abstract representation of the different object types available for segmentation. The base is made up of keys and values, which are derived from the encoder. [6]

The ATM module begins in a similar way to a traditional transformer, with a self-attention mechanism. After this step, a residual connection is added. This connection ensures that the original information is preserved and that

the model input can be combined with the output of the self-attention. The output is then subjected to layer normalisation to improve the stability and convergence speed of the model. [6]

Self-attention is followed by the core step of the ATM module: the Attention to Mask step. In this step, masks are generated that describe the location and size of each class within the image. This process uses the output of the previous self-attention step and combines it with the base, which consists of keys and values. A mask is created for each class, analysing the degree of similarity between the class embeddings and the base. After generating these masks, a residual join is applied again, followed by layer normalisation. [6]

The final step in the ATM module involves a feed-forward neural network (FFN). After the FFN step, as in the previous steps, a residual connection is added to preserve the original information, and a final layer normalisation is performed. The result of these three steps is a set of re-updated class embeddings ready for further processing or output. [6]

After completing three iterations, the ATM module has three sets of masks, each generated from different layers of the encoder, as well as a final updated set of N class embeddings. The final output of the segmentation process is generated by calculating the dot product of the class probabilities and the generated mask groups. This result provides an accurate and detailed picture of the distribution of classes across the image, which is crucial for applications such as medical image analysis, autonomous vehicles and advanced computer vision tasks. [6]

2.5 Adaptive Architectures in Vision Transformers

The use of adaptive architectures in Vision Transformers is essential to improve performance and efficiency in semantic segmentation tasks. The traditional ViT uses a fixed architecture, resulting in limitations in processing data with varying complexity and characteristics. Adaptive architectures provide flexibility by dynamically adjusting the model's structure based on the characteristics of the input data. As a result, the model can not only handle simpler data more efficiently, but also perform better in more complex tasks. The outcome is a balance between accuracy and computational efficiency, which makes adaptive ViTs particularly suitable for large-scale, real-time applications such as data processing in self-driving cars.

Different techniques have been proposed to improve this computational cost. Some aim to improve the attention mechanism itself [13, 43], while others propose to reduce the input by limiting tokens going into a transformer block by pruning or merging techniques. [44, 9, 8, 7]

2.5.1 Adaptive Vision Transformer

Adaptive Vision Transformer (A-ViT) [7] is an advanced Vision Transformer architecture with a built-in method for token pruning. The architecture is designed to reduce the computational cost of Vision Transformers by automat-

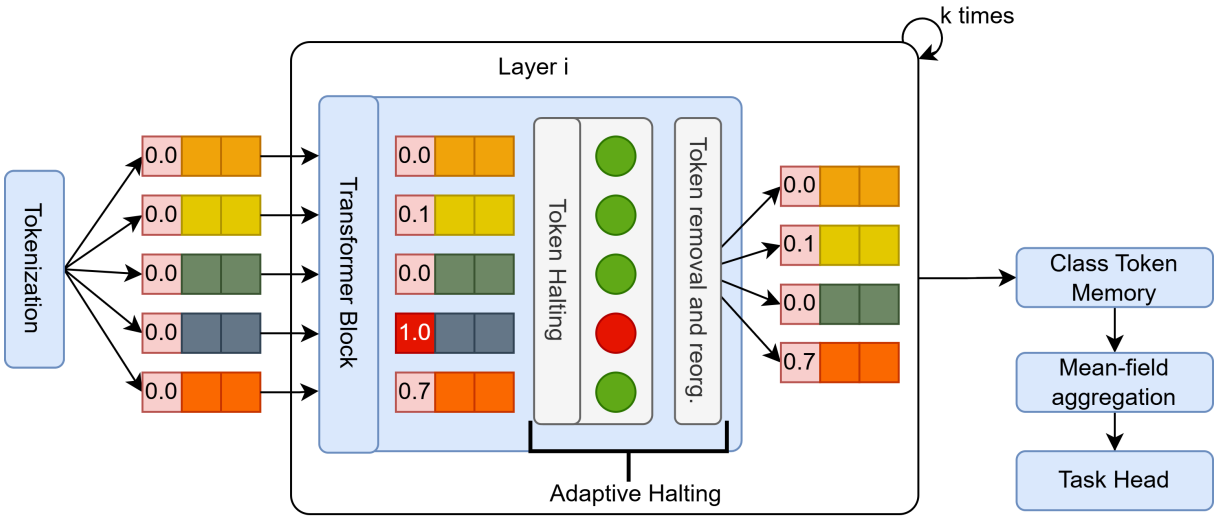

Figure 2.10: The architecture of A-ViT, adapted from [7]

ically scaling down calculations based on the complexity of the input tokens.

Unlike standard Vision Transformers [3], which have a fixed computational cost regardless of input, A-ViT adapts to the difficulty of each individual image. It accomplishes this by disregarding patches of the image with little informative value, such as the background. Simultaneously, it retains patches that contain a lot of information. By applying this approach dynamically, calculations are scaled based on the complexity of the image. Simpler images require fewer patches and therefore less computational load. [7]

This mechanism works through a 'spatially adaptive inference' approach, in which the computation of certain tokens (such as low-information tokens) is stopped at different depths (different layers) in the transformer model. In each layer, a halting probability is calculated. When the cumulative sum of this halting probability reaches 1, processing of the token in question is stopped and it does not proceed through the model. The halting probability is calculated using logistic sigmoid function of the token's $e^t h$ dimension, multiplied by a scale parameter and summed with . This approach ensures that simple tokens do not spend unnecessary time in the transformer, increasing the efficiency of the model. [7]

A-ViT, however, is primarily designed for classification tasks rather than segmentation, particularly semantic segmentation. The pruning of tokens between transformer blocks leads to information loss, resulting in fewer tokens at the output of the transformer encoder than the patches of the input image. While this reduction is acceptable for classification, where the remaining tokens retain sufficient information for accurate categorization, it forms a significant challenge for semantic segmentation, where each pixel must be assigned a category. Due to the pruning of some image patches, it becomes impossible to classify the pixels in those pruned regions, making A-ViT unfit for this use case. [9, 8]

2.5.2 Dynamic Token Pruning in Plain Vision Transformers

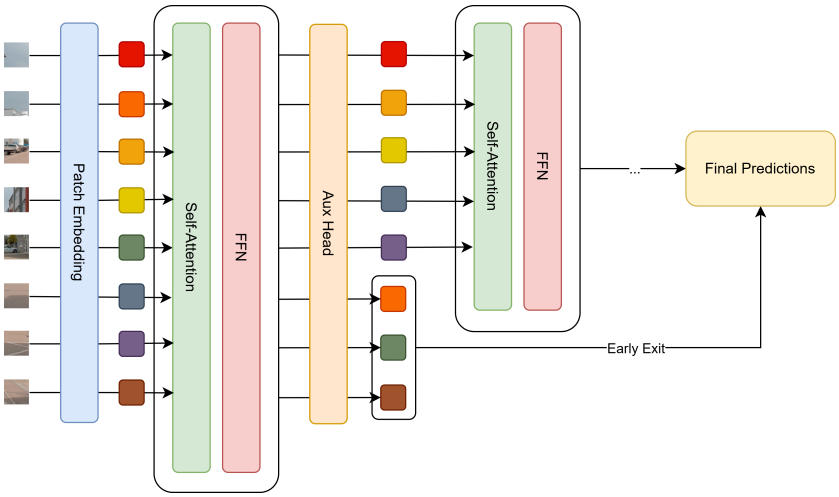

Figure 2.11: Dynamic Token Pruning Architecture. Adapted from [8]

Dynamic Token Pruning (DToP) [8] introduces an approach to improve computational efficiency in semantic segmentation models containing a transformer backbone by halting the processing of easier tokens earlier in the pipeline.

The model consists of M stages in the transformer encoder. After each stage, an auxiliary head evaluates whether the token is simple enough to be halted. If deemed easy, the token is stopped early, and a prediction can be made. If the token is determined to be more complex, it proceeds to the next stage of the network. After all M stages, the remaining tokens are passed through the decoder head.[8]

The auxiliary head of DToP assesses token difficulty using a modified version of the Attention-to-Mask module [6]. This modification involves reducing the depth of the module (i.e., fewer layers) and decoupling multiple cascading modules to function as separate auxiliary segmentation heads. Tokens are halted when their difficulty score reaches a certain threshold. [8]

In an image of a clear sky, many tokens may be deemed 'easy.' Halting all such tokens simultaneously, however, risks losing contextual information needed in later layers. To adjust for this issue, some tokens of specific classes are always retained, ensuring that the model maintains sufficient levels of contextual information and has shown to improve overall accuracy of semantic segmentation tasks. [8, 45]

Testing shows DToP can reduce on average 20% to 30% in computational cost without notable accuracy degradation [8]. However, the addition of the expensive attention mechanisms adds a lot of computational complexity [2], partially negating the efficiency gains from pruning tokens within the vision transformers.

Dynamic Token pruning does not consider the potential of previous predictions. While in single images, there is only the possibility to prune easy tokens, a stream of images opens the opportunity to prune tokens that have

been previously predicted, which may improve the efficiency further.

2.5.3 Dynamic Token Pass

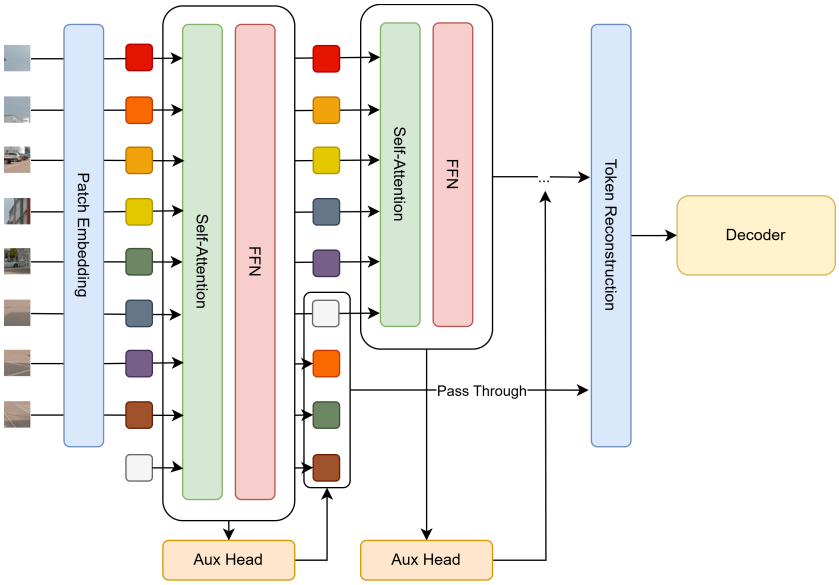

Figure 2.12: Dynamic Token Pass Architecture. Adapted from [9]

Dynamic Token Pass [9] is an architecture which, like DToP [8], reduces the computational complexity of a transformer architecture by halting 'simple' tokens early before sending them to deeper layers. This is achieved through an auxiliary head that is data-aware and dynamically adjusts the model's complexity.

The decision to stop a token is made by the decision layers. A decision layer consists of a self-attention layer. There are D decision layers present, which divide the L layers of the backbone into D+1 different blocks with every block except the last containing an auxiliary head with the decision layer. [9]

After going through the entire transformer encoder, a token reconstruction takes place, which then serves as input to the decoder that makes a prediction for semantic segmentation.

The auxiliary head however is still a computationally expensive algorithm. While testing shows a decrease of about 13 to 32 % in GFLOPs [9], a lot of gains in efficiency, similar to DToP, are negated by the attention mechanism. Dynamic Token Pass does not contain any image stream optimizations.

2.5.4 Token Merging

Token Merging (ToMe) [44] is another approach for reducing the computational complexity in transformer models by merging redundant or similar tokens, thereby reducing the number of tokens processed at each layer in the

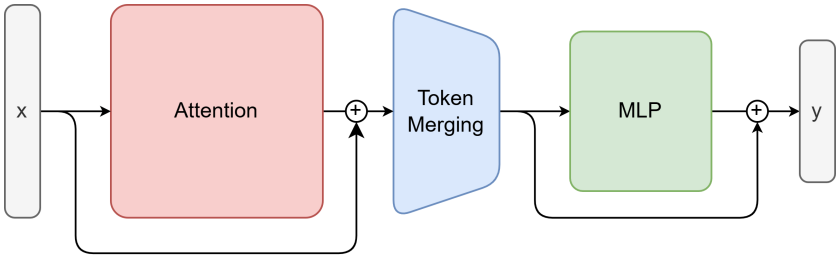

Figure 2.13: Token Merging architecture

transformer encoder. This method involves a module that can be integrated into an existing vision transformer without retraining of the original model.

ToMe reduces the number of tokens by q fixed amount, r, for each layer, where r represents a specific number of tokens rather than a ratio. Over the entire transformer, consisting of L layers, the total reduction amounts to Lr. The token merging occurs between the self-attention mechanism and the fully connected neural network [44].

Tokens are merged based on their similarity, which is determined using the dot product between two token embeddings. Unlike clustering algorithms such as kmeans clustering (Lloyd, 1982), which may introduce additional complexity and potentially negating computational gains, ToMe uses the Bipartite Soft Matching algorithm to improve performance. This algorithm divides all tokens into 2 sets, A and B. For every token in set A, an edge is drawn to the most similar token in set B. Every combination has a similarity score, the result from the dot product. The r most similar token combinations are merged, (for example by averaging their features), reducing the token count by at least r. The merged tokens are subsequently concatenated back together for processing in the MLP block. [44]

While token merging does not directly contain a way to process image stream data, experiments were also executed on video data. [44] These experiments showed an increase in throughput of about 2.2 times, with a negligible accuracy drop.

This model does not need retraining and can be applied to any off-the shelf vision transformer backbone.

2.5.5 Semantic aware Temporal Accumulation

Semantic-Aware Temporal Accumulation token pruning is a method to reduce the number of tokens in transformer models for video processing. The primary goal of STA is to prune redundant tokens that have similar properties within a sequence of images. The algorithm evaluates tokens based on two key factors: temporal similarity and semantic awareness. Temporal similarity refers to the resemblance of tokens across consecutive frames, removing tokens that have already appeared in earlier frames, thus reducing redundancy. [10]

The semantic awareness component adds a weighting mechanism to the temporal similarity score, ensuring that

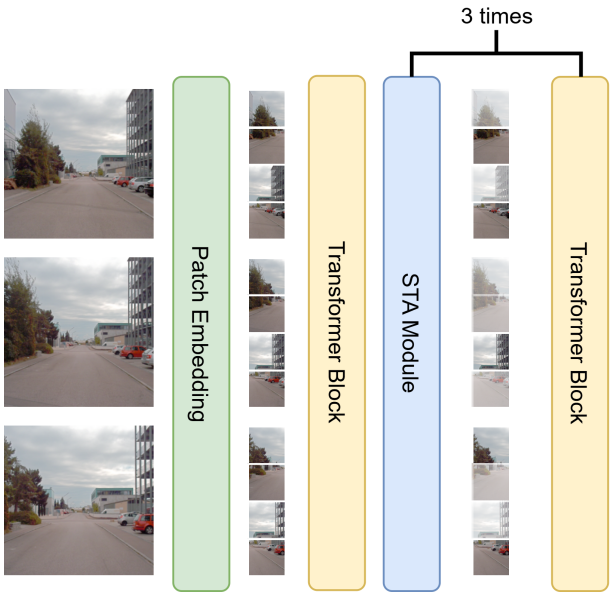

Figure 2.14: Conceptual pipeline of video Transformer with STA module, adapted from [10]

tokens are not pruned uniformly. Tokens that provide less information about the class attributes are assigned a higher weight and are more likely to be pruned, while tokens with higher semantic relevance are given a lower weight and are preserved. [10]

While STA offers promising benefits in token reduction, it presents two significant limitations when applied to image streams in the context of semantic segmentation.

First, STA was originally developed for classification tasks, where removing tokens is more feasible. In contrast, semantic segmentation requires detailed information at every stage, making the pruning of tokens less suitable.

Second, STA uses entire video sequences, rather than processing individual frames in real-time. This limitation is a challenge for vehicles, where the camera moves continuously, making it difficult to apply STA effectively. Furthermore, STA has not been thoroughly tested in scenarios involving moving cameras, raising concerns about its effectiveness in such environments.

2.6 Datasets

An ideal dataset for this study would feature long, segmentation-annotated videos with high frame rates, captured across diverse scenarios. These characteristics ensure robustness during model evaluation.

2.6.1 Cityscapes

Cityscapes [13] is a well-established dataset designed for "semantic urban scene understanding". It contains a total of 25000 RGB images, including 5000 fine-annotated frames and 20000 coarse-annotated frames. The dataset includes a variety of urban scenes and video sequences. However, the relatively small size of the dataset makes it unsuitable for this study, as it lacks a sufficiently large collection of long video segments.

2.6.2 A2D2

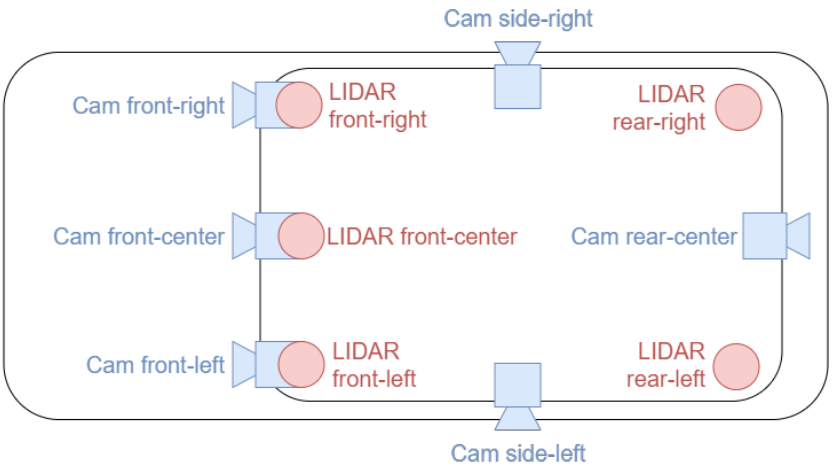

Figure 2.15: Schematic top view of the sensor carrier with sensors used to create the A2D2 dataset. Adapted from [1]

The A2D2 dataset [1], created by AUDI, is captured from a driving car in various urban environments across multiple cities in Germany. This dataset includes both LIDAR and RGB data, providing a diverse set of recordings from different driving conditions. The A2D2 dataset is split into two parts: 41277 non-sequential annotated frames and 392556 unannotated sequential frames.

While the non-sequential frames are annotated for semantic segmentation, the sequential frames are not. This creates a challenge, as it prevents direct evaluation of the model's output against ground-truth annotations for these video segments. Nonetheless, the sequential frames offer a substantial testing base that allows us to assess the model's performance in various driving conditions, comparing it against a baseline model.

The sensors used in the creation of the A2D2 dataset are a combination of 6 RGB cameras and 5 LIDAR sensors. The sensors are positioned at various locations on the vehicle to provide comprehensive coverage of the environment. The dataset's frames are captured at a resolution of 1920×1208 pixels, and the data is recorded at 30 frames per second.

As shown in Figure 2.15, the front of the vehicle is equipped with three forward-facing cameras, referred to as cam front-right, cam front-center, and cam front-left, which have overlapping fields of view. Each of these front-facing cameras is paired with a corresponding LiDAR sensor. Additionally, the vehicle features two side-mounted cameras (left and right) and a rear-center camera. LiDAR sensors are also installed at the rear, on the rear-right and rear-left positions.

Although the dataset is extensive, it lacks information such as vehicle speed and varying weather and lighting conditions, which could further enrich the evaluation.

2.6.3 Waymo Open Dataset

The Waymo Open Dataset [12], specifically the Perception set, provides annotations for 3D segmentation, making it suitable for semantic segmentation tasks. However, this dataset is somewhat limited in terms of the total amount of available data, consisting of only 1,150 scenes, each approximately 20 seconds long. Additionally, the dataset's frame rate is lower compared to A2D2 [1], with data captured at just 10 frames per second. While the dataset's annotations are valuable for segmentation tasks, its limited size and lower frame rate make it less ideal for the purposes of this study.

3

Methodology

3.1 Focus Area in Dataset

As discussed in Section 2.6.2, the dataset contains data for 6 RGB cameras. In this study, the focus lies on the front center camera. To ensure consistency and optimize model performance, a center crop is applied to the images, resulting in square images with dimensions of 1208×1208 pixels. This focuses on the central region of the scene, where the main action typically occurs. It also aligns with the preprocessing used during the model's pretraining.

3.2 Model

The focus of this paper is to propose an architecture tailored to optimize the inference performance of Vision Transformer based semantic segmentation models, specifically in the automotive domain. This optimization is achieved through the integration of adaptive computation techniques within the architecture.

To accomplish this, a weightless module is incorporated into the transformer block. Since this module does not introduce trainable parameters, it allows for a seamless transfer of weights from pretrained models to the enhanced architecture. This approach preserves the original model's pretrained capabilities while enabling more efficient and adaptable inference.

The SegVit [6, 23] architecture was selected due to its superior results in the field of semantic segmentation, particularly in the context of vision transformers [24, 6]. SegVit employs a standard vision transformer encoder and the ATM-decoder head, as described in Section 2.4.4. SegVit offers several options for pretrained models, including models trained on the ADE20K [46], COCOStuff10K [47], and PASCAL-Context [48] datasets.

The model trained on the COCOStuff dataset serves as the foundation for this work due to its specific relevance to outdoor scenarios, which closely align with the A2D2 dataset. The COCOStuff10K dataset contains classes for various outdoor objects, such as vehicles, people, plants, and different types of ground surfaces. Additionally, it

includes annotations for contexts like the sky and other outdoor environments, which overlap with the classes and objects present in the A2D2 dataset [47, 1]. This choice allows for more effective model transfer and better performance in the target domain.

3.3 Proposed Architecture

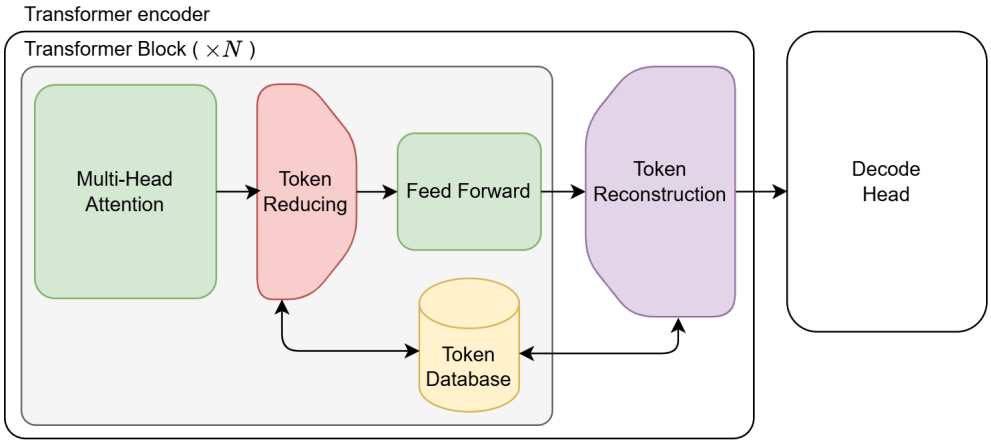

Figure 3.1: Proposed Architecture

This master's dissertation proposes a new architecture that is inspired on elements of both token pruning methods as token merging methods.

The architecture tries to optimize the transformer blocks by limiting the tokens that are input in the transformer blocks. Since the multi-head attention is a computational exponential increasing algorithm, limiting the input should decrease the time to compute. This multi-head attention is calculated multiple times, since the transformer block is executed N times.

Therefore we introduce a token reducing module. This module dynamically reduces the amount of tokens send to the next layer, thereby not changing the exponential nature of the multi-head attention, but lowering the computational load by decreasing the amount of tokens that are served as input to the next transformer layer. This module is situated in each transformer block between the multi-Head attention block and the feed forward neural network, and is situated as a red block in Figure 3.1. This module uses both techniques from token merging, i.e. the part where we want to look what tokens are similar to each other, and parts from token pruning mechanisms, by instead of merging the tokens, completely removing them from the set for the next forward pass. The token reducing method uses a database to save data about these tokens for later use. The complete process is explained in depth in Section 3.3.3

To keep the decode head unchanged, the token set of the last transformer layer needs to be rebuild to a token set

that the decode head can use. This is done in a token reconstruction module, shown in purple in Figure 3.1. This builds a token set compatible with the decode head using both output of the transformer layers and a history. This process is explained in detail in Section 3.3.4

3.3.1 Definitions

- **Transformer Layer (L):** A single layer in a transformer model.
- **ViT Transformer Encoder:** Comprises multiple transformer layers arrange sequentially. Each layer is assigned a unique index i, allowing a specific layer to be identified as L_i.
- **Input to the Token Reducing Block (I_i):** The input token set provided to the token reducing block within the layer L_i.
- **Reduced Token Set (R_i):** The output token set produced by the token reducing block in layer L_i, containing fewer tokens than I_i. R_i is also referred to as the reduced token set.
- **Token History (D_{i_hist}):** The set of tokens from previous frames. Stored in a database contained in L_i.
- **Processed Token History (D_{i_proc}):** The set of processed versions of the tokens in D_{i_hist}.
- **Source Tokens (SRC_i):** The subset of tokens in I_i that are excluded from R_i during the token reduction process, leading to the shrinking of the token set.
- **Destination Tokens (DST_i)**: The tokens in the database that are most similar to the source tokens (SRC_i) and are thus linked to them during processing. This is a subset of D_{i_proc}.
- **Reduction size (r_i)**: The size of the reduction. This is equal to the length of SRC_i and DST_i.
- **Output set (O)**: The output set of the reconstruction process, also referred to as the reconstructed token set.

3.3.2 Database Design

Each transformer layer has its own database to efficiently manage the history of tokens. The database design, as illustrated in Figure 3.2, consists of three key elements, maintaining a structured token reduction workflow.

The first element is the Token History tensor, denoted as D_{i_hist} which stores tokens generated during the token reduction step of the previously processed frames. Specifically, this tensor holds the outputs of the token reduction in the current layer during previous inferences. This tensor has a circular design, where tokens are written sequentially in a circular manner. Once the tensor reaches its capacity, older tokens are overwritten by newer ones.

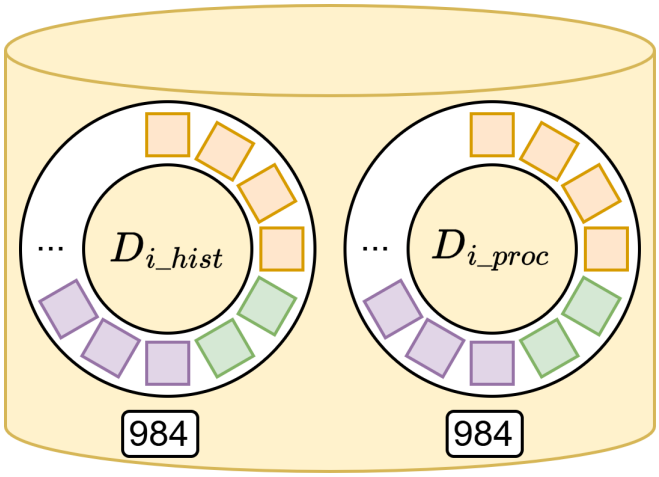

Figure 3.2: Token Database Design

The second element is the Processed Token tensor, denoted as D_{i_proc}, which contains the final processed tokens corresponding to the output layer. Each token in this tensor maps one-to-one with a token in the Token History tensor. Essentially, this tensor holds the version of the mapped token in the token history that was fully processed, i.e. have run through all transformer layers. This token also follows the same circular design as D_{i_hist}

The third element exists of two indices. These point to the position on the circle where new tokens should be added.

3.3.3 Token Reducing

The token reduction block, as seen in Figure 3.3, processes I_i, the output from the multi-head attention block in layer L_i, with the primary objective of reducing its size. This is achieved through the following approach:

The process identifies tokens in I_i that closely resemble tokens stored in the database, D_{i_hist}. To determine these similarities, the bipartite soft matching algorithm is utilized. While this algorithm does not provide optimal results like clustering, it offers a practical trade-off between computational efficiency and accuracy

The algorithm begins by treating each token in I_i as a node, collectively forming the set a, while the tokens in D_{i_hist} are treated as another set of nodes, b. For each token in a, an edge is drawn to the most similar token in b. The similarity between tokens is determined using cosine similarity (Figure 3.4), and the similarity score serves as the weight of the edge.

After establishing these connections, the algorithm filters the tokens in a based on a predefined threshold for their edge weights. Only tokens with an edge weight exceeding this threshold are retained, while the others are discarded. The retained tokens, which remain connected to database tokens, are referred to as SRC_i or source tokens, and the tokens in b connected to these source tokens are designated as destination tokens, or DST_i.

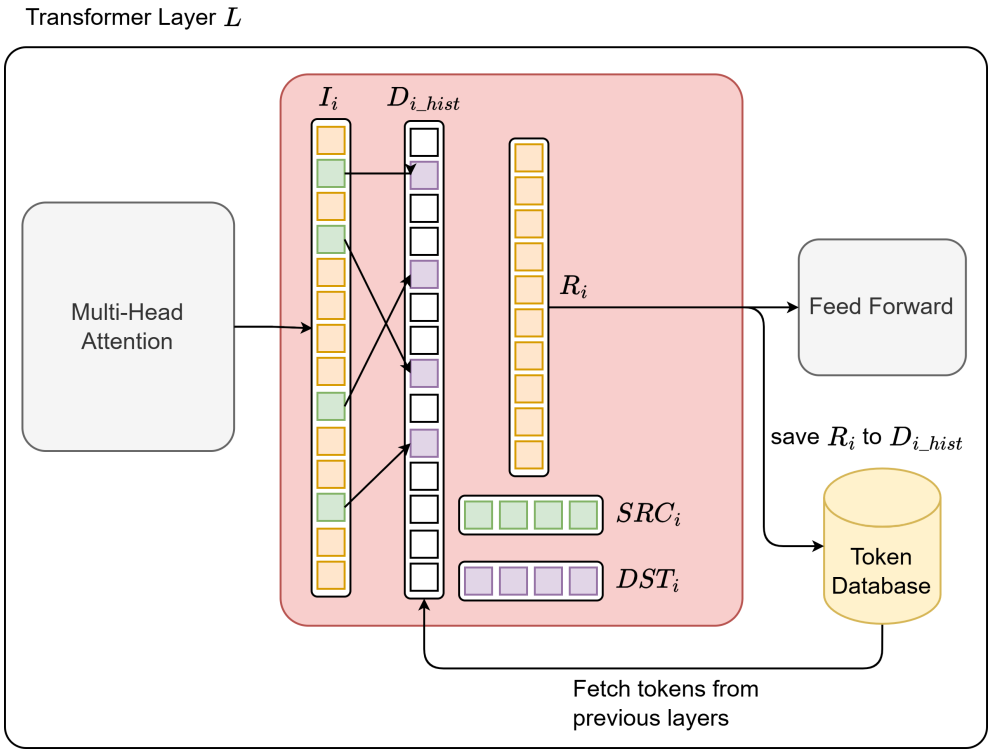

Figure 3.3: Token reducing block

$$\frac{\vec{a} \cdot \vec{b}}{||\vec{a}||\,||\vec{b}||}$$

Figure 3.4: Equation of Cosine Similarity, where $\vec{a}$ is a token from I_i and $\vec{b}$ is a token from D_{i_hist}

Finally, any tokens in a that fail to meet the similarity threshold are classified as reduced tokens, R_i.

R_i is saved to the token history D_{i_hist}. This update introduces a temporary imbalance with the processed token history, as the newly added tokens lack a corresponding processed version. This imbalance is expected and will be resolved during the reconstruction step.

Example

An example is illustrated in Figure 3.5, where a set of 5 tokens from a is processed. Each token in a is connected to its most similar token in the b set, with the similarity represented by a weighted edge. The weight ranges from -1 to 1, where 1 indicates a perfect match, and -1 signifies complete dissimilarity or the opposite of the token.

In this case, the similarity threshold is set to 0.80.

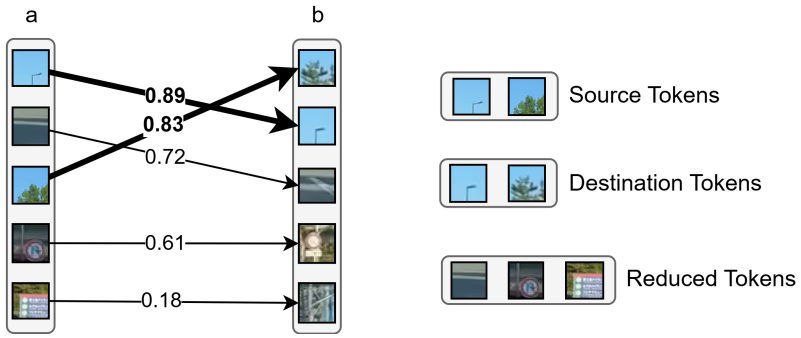

Figure 3.5: Example of the matching algorithm.

Among the 5 connections, only 2 have weights exceeding the threshold of 0.80. These connections are retained, while the remaining 3, with weights below the threshold, are discarded. As a result, the lower three tokens in a in Figure 3.5 lose their connections to tokens in b. These disconnected tokens are categorized as reduced tokens and form the output of the token reduction block. The two tokens in a that exceed the threshold are referred to as source tokens, and their corresponding matches in b are designated as destination tokens.

3.3.4 Token Reconstruction

Token reconstruction occurs during the final stage of the modified Vision Transformer encoder, as illustrated in Figure 3.1. This step is critical for compatibility with the ATM-decode head from SegVit, as it restores the reduced token set to its original size. Since the model should not be retrained, the output of the transformer encoder must be identical to that of an unmodified encoder. This is achieved by iteratively reversing the token-reducing process applied during inference.

The reconstruction process aims to build an output set of tokens that is the same size as the token set produced by an unmodified vision transformer. As shown in Figure 3.6, the process begins with the reduced token set produced by the final transformer layer L_n, where n is the index i of the last layer in the transformer encoder. This set, denoted as R_n, significantly smaller than the input token set due to the token-reduction process (described in Section 3.3.3), forms the initial output O and is iteratively expanded during the reconstruction.

Each layer in the modified transformer encoder retains metadata, such as SRC_i and DST_i, that guides the reconstruction. The reconstruction process proceeds layer-by-layer in reverse order. For each layer L at index i, tokens pruned during the forward pass (SRC_i), are reintroduced into the output token set O. The metadata associated with layer L_i provides information about the source tokens and their original positions. Using this data, the reconstruction process identifies the precise locations in O where the reintroduced tokens should be placed.

However, directly using the token sets SRC_i or DST_i in these slots is insufficient. While the tokens may

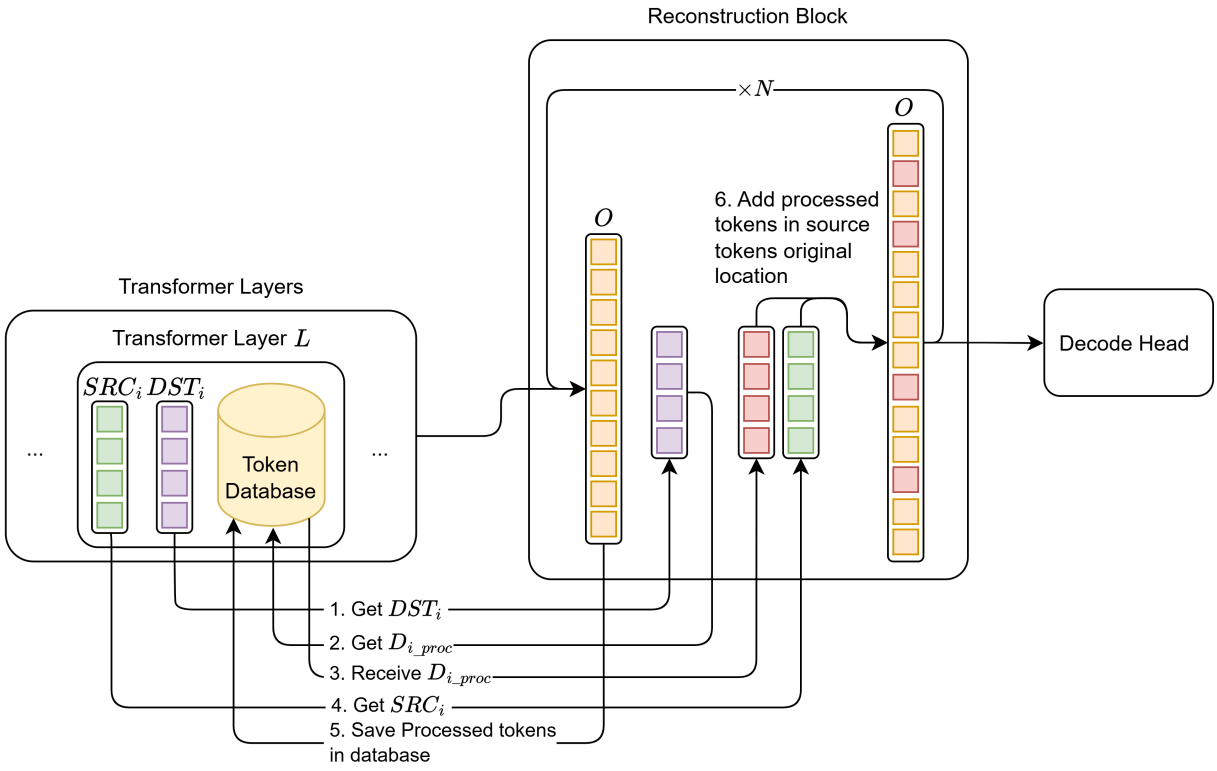

Figure 3.6: Architecture of the token reconstruction block.

have been similar to each other during reduction, their relevance diminishes as they were processed by fewer transformer layers compared to tokens that have gone through all layers. To address this, a fully processed version of each destination token (D_{i_proc}) is used during the reconstruction of the previous frame. The fully processed tokens corresponding to the tokens in DST_i are then used to populate the identified slots in O. This ensures that every token in the reconstructed output has undergone consistent processing through the entire transformer encoder.

As the process advances through each layer, the output token set expands by the same number of tokens pruned (r) during the forward pass. After reconstructing a given layer L_i, the token set matches the original size of input I_i of that layer, equivalent to the reduced set $R_{(i-1)}$ of layer $L_{(i-1)}$.

Finally, the database of L_i is updated with the processed tokens. Fully processed tokens from the reconstruction, the current state of O, are populated into D_{i_hist}. This ensures a one-to-one correspondence between the reduced tokens D_{i_hist} and their processed version D_{i_proc}.

This process repeats until all layers have been reconstructed, producing an output token set identical in size to the transformer's original input. This fully reconstructed token set is then ready for processing by the semantic segmentation decode head.

Example

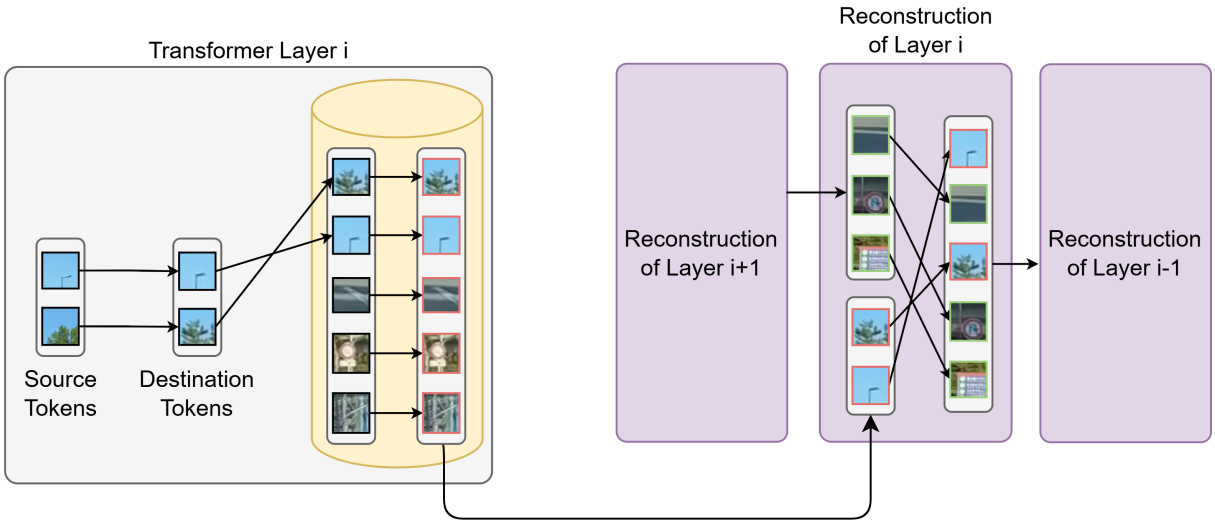

Figure 3.7: Example of a reconstruction

This example reconstructs the reduction from the example in Section 3.3.3.

As previously discussed, each transformer layer L_i that reduces the token set must undergo a reconstruction. The reconstruction process proceeds in reverse order, starting from the final transformer layer and progressing sequentially to the first layer. In this context, the example assumes that L_i is neither the first nor the last layer. Consequently, the input to this reconstruction step originates from the reconstruction of layer L_{i+1} and its output is forwarded to the reconstruction of layer L_{i-1}.

The input of the reconstruction process is a set of tokens O, which may consist of tokens that were reconstructed in higher layers or tokens that passed through the entire transformer encoder without being removed. These tokens are represented as green-bordered tokens in Figure 3.7. At this stage, the specific origin or composition of O is irrelevant to this step. The primary focus is to expand the size of O by r_i.

To achieve this, the reconstruction process retrieves the source tokens SRC_i and their corresponding destination tokens DST_i from L_i. The DST_i tokens are looked up in the token history D_{i_hist} to obtain their processed equivalents from the processed token history D_{i_proc}. In Figure 3.7, these processed tokens are depicted with a red border.

Once the processed equivalents of DST_i are retrieved, they are reintroduced into O at the original location of SRC_i. This operation expands O by precisely r_i, restoring the token set size to its original state prior to reduction.

Since the reconstruction is an iterative process, the updated O now serves as the input for the reconstruction of the next layer, L_{i-1}. This iterative reconstruction continues until the first layer is reached.

3.3.5 Startup Procedure

When the model starts from scratch, all transformer layers begin with an empty database. As a result the first frame, no tokens can be matched to tokens in D_{i_hist}, and no reduction occurs. In this scenario, $r_i = 0$, and both SRC_i and DST_i are empty lists. However, the rest of the token processing pipeline remains intact, ensuring all tokens from the first frame are added to the database D_{i_hist}.

From the second frame onward, tokens stored in D_{i_hist} become available for matching. Since the database may not have reached its maximum capacity at this stage, r_i might be smaller compared to a fully warmed-up model. Over time, the database dynamically builds itself based on the current context, adapting to the environment or situation the model operates in.

Using a predefined database at initialization might seem like an alternative to a cold start. However, this approach could be unsuitable if the predefined database does not align with the current context. By building the database dynamically from scratch, the model ensures that D_{i_hist} remains relevant and tailored to the vehicle's immediate surroundings.

3.3.6 Token Lifecycle

The life cycle of a token is viewed from the perspective of the layers themselves. It begins with the token's "birth" when an input item is introduced into the layer. The token first passes through the multi-head attention block, where it gathers contextual information by interacting with other tokens. After this, it proceeds to the token-reducing block.

In the token-reducing block, the token is either terminated or continues. If it matches an entry in the database (D_{i_hist}), its life cycle ends, and the token is halted. If no match is found, the token continues to live, is added to D_{i_hist} and becomes part of the input for subsequent layers.

Once all layers are completed and the model enters its reconstruction phase, the token is paired with a fully processed version, which will be stored in D_{i_proc}. This token pair remains in the database until it is eventually overwritten by a new token pair.

3.4 Implementation Details

The architecture used builds upon the framework established by SegVit[6], which employs a plain Vision Transformer integrated within the MMSegmentation[49] and MMEngine[50] frameworks. Two additional modules, the token reduction block and token reconstruction block, were developed to enhance the architecture.

3.4.1 Decode head

In previous sections, a simplified description of the ATM-decode head was provided. The actual implementation requires tokens from three distinct stages of the encoder: the outputs of layers $8, 16$, and 24. The implementation ensures that tokens from each encoder layer are reconstructed independently, with the processed tokens from each stage contributing to their respective reconstructions.

For example, at layer 8, tokens are reconstructed to their original size using processed tokens from layer 8. This process is repeated for layer 16 and subsequently for layer 24, ensuring that all three stages provide their respective reconstructed token sets. Sets don't get mixed.

3.4.2 Matching Algorithm

The matching algorithm used in this work is inspired by the methodology in ToMe[44]. In ToMe, tokens are divided into two groups, a and b, where b is transposed (b^T). The dot product of a and b^T produces a tensor representing the similarity between token pairs, and the maximum value is selected for each token in a.

In this implementation, tokens are not split. Instead, a represents tokens from the current image, while b represents tokens from D_{i_hist}. In addition to the dot product between a and b^T, normalized versions of a and b, referred to as a_{norm} and b_{norm}, are computed to represent token vector lengths. The cosine similarity for each possible token pair is calculated as:

$$\frac{a \cdot b^T}{a_{norm} \cdot b^T_{norm}}$$

For each token in a, the maximum similarity value is identified, linking tokens from the image to their closest matches in D_{i_hist}. These links are stored as `src_idx` and `dst_idx`, representing the indices of matched tokens.

Unlike ToMe, this implementation employs a dynamic reduction method. A threshold is applied to filter tokens based on similarity, creating a mask over the tensor and forming the SRC_i set of tokens exceeding the threshold.

This code can be found in Code Fragment 1

3.4.3 Database

As described in Section 3.3.2, the database is designed with a circular structure that continuously overwrites itself to maintain efficiency. Implemented as a tensor, it is kept in GPU memory to avoid latency from CPU-GPU data transfers. This design ensures minimal bottlenecking during token storage and retrieval.

Each index in D_{i_hist} directly corresponds to the same index in D_{i_proc}, eliminating the need for explicit pointers. This correspondence simplifies reconstruction processes.

The circular design resolves memory management issues encountered in alternative designs that concatenated partial tensors. Such approaches caused memory fragmentation and GPU garbage collection challenges. By contrast, the circular design ensures consistent performance and memory efficiency.

As detailed in Section 3.4.1, the database includes three tensors corresponding to the processed tokens from layers 8, 16, and 24.

3.4.4 Reconstruction

Reconstruction relies on the `src_idx` and `dst_idx` indices derived in Section 3.4.2. Tokens at `dst_idx` in D_{i_proc} are mapped to the corresponding `src_idx` positions. The one-to-one correspondence assumption, described in Section 3.4.3, facilitates efficient reconstruction of token sets for each encoder layer. This code can be found in Code Fragment 3.

4

Experimental Results

4.1 Evaluation Metrics

The performance of the proposed model is evaluated using the following key metrics:

- Inference Time: The duration required to execute both the encoder and decoder components of the model.
- Pixel-Wise Loss: The difference between the optimized architecture and the original model's outputs.
- Computational Efficiency: The number of FLOPs required by the model, which reflects its computational intensity.

The evaluation was done using the A2D2[1] dataset, specifically the videos captured in Gaimersheim, Ingolstadt, and Munich. For the purpose of this thesis, the benchmarking visualizations are based on the Gaimersheim video. While results from other videos in the dataset are comparable, they may vary depending on the complexity of the environment. For instance, videos with more complex scenes may result in slightly reduced performance, whereas simpler scenes show better outcomes due to the model's adaptive nature.

All GPU benchmarking experiments were performed on imec's GPULab infrastructure, specifically the 900 cluster. The hardware configuration included:

- GPU: Nvidia GeForce GTX 1080 TI
- Memory: 16 GB of CPU memory
- Processors: 4 CPUs

4.2 Benchmarking of the current architecture

The current architecture consists of a plain Vision Transformer Encoder[3] and the ATM decode head from SegVit[6].

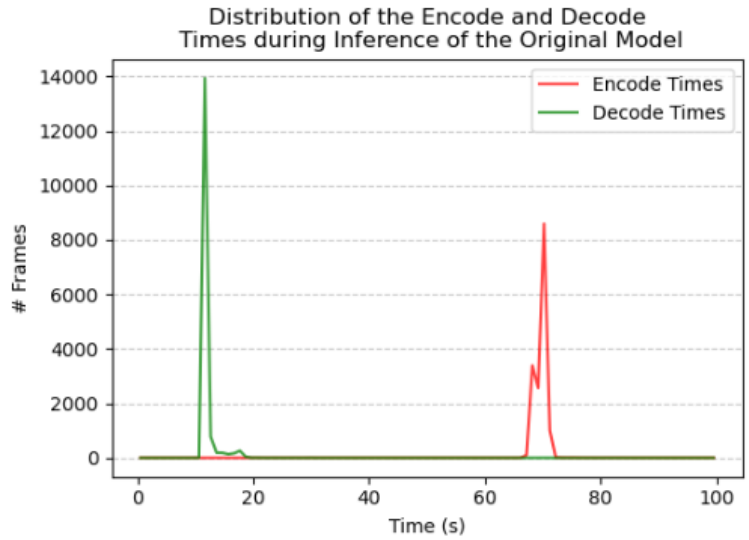

Figure 4.1: Distribution of Encode and Decode Times during Inference of the Original Model

A benchmarking analysis of inference time for a single image shows that the Vision Transformer Encoder requires approximately $70ms$, where as the decode head completes in approximately $12ms$. This indicates that the Encoder is almost 6 times slower than the Decoder, as visualized in Figure 4.1

These findings highlight that the Encoder presents the greatest potential for optimization.

Table 4.1 summarizes the metrics for inference time of the original model on GPU.

Table 4.1: Metrics on Encode and Decode Inference Time of the Original Model (GPU)

	Mean (ms)	Median (ms)	Standard Deviation (ms)	Variance
Encode Time	69.696	69.987	2.109	4.447
Decode Time	11.912	11.600	1.174	1.379

4.3 Experiments and Analysis

4.3.1 Varying Thresholds for Cosine Similarity

To evaluate the impact of varying thresholds in the token matching algorithm, a series of experiments were conducted to test different parameter settings, specifically the threshold for the cosine similarity. It was observed that lower threshold values significantly affect the pixel-wise accuracy. As can be observed in Figure 4.2 thresh-

olds below 0.94 resulted in outcomes with a median pixel-wise loss higher than 20%, and with thresholds under 0.84 leading to a median pixel-wise loss of 75% compared to the original model.

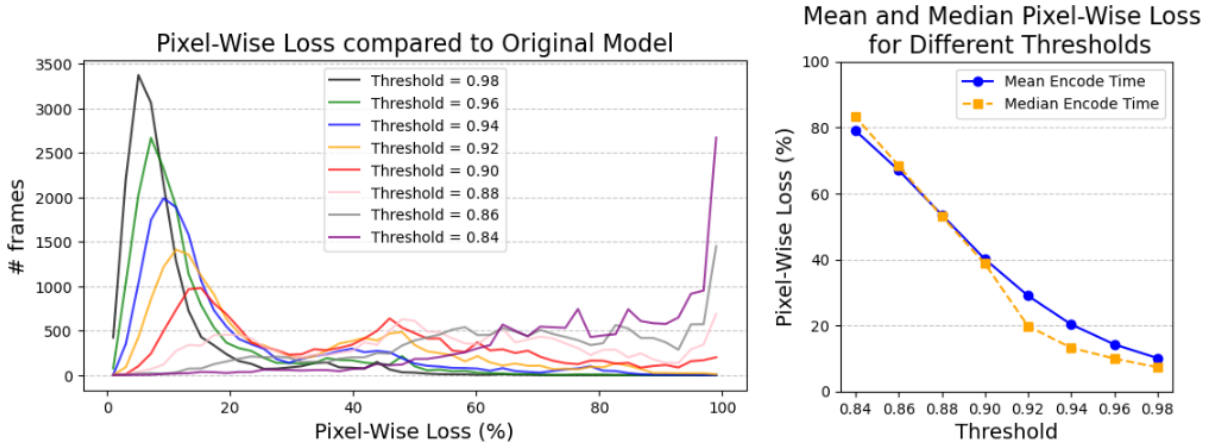

Figure 4.2: (left) Distribution of the Pixel-Wise Loss from the output of the Token Reduction Model compared to the output of the Original Model. (Right) Mean and Median Pixel-Wise Loss at different Cosine Similarity Thresholds

This aligns with expectations. When the cosine similarity threshold decreases, a higher number of tokens are matched to less similar counterparts, which reduces the accuracy of the predictions. This decline shows the importance of maintaining a high threshold to ensure precise token matching.

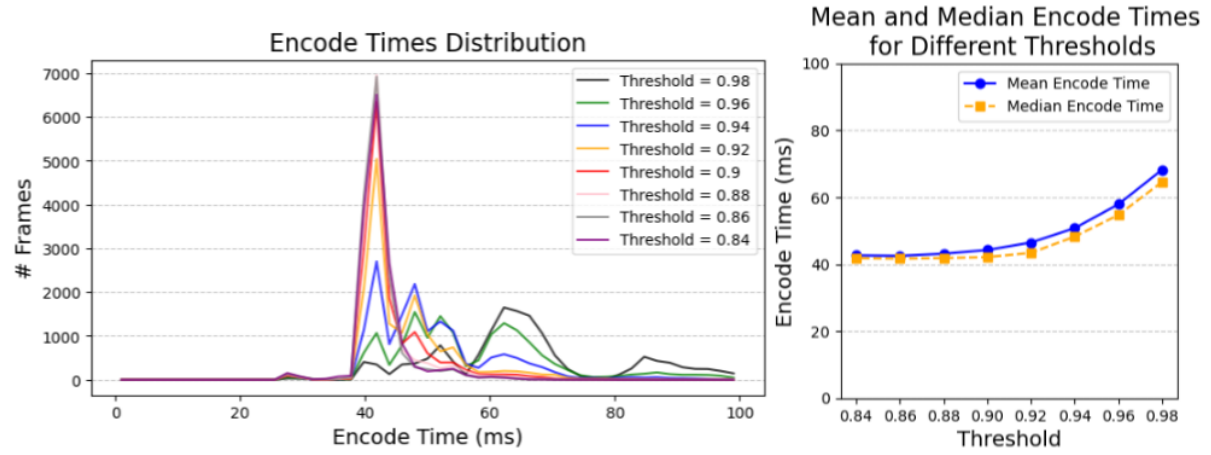

Figure 4.3: (left) Distribution of encode times of the token reduction model at different fixed thresholds, with a reduction in every layer. (right) Mean and median encode times of the token reduction model at different fixed thresholds.

Additionally, lowering the threshold has the benefit of increasing the number of tokens matched early in the processing pipeline, thereby reducing the computational load on the deeper layers of the vision transformer encoder. This can be observed in Figure 4.3 This leads to shorter decode times. However, balancing accuracy and processing time is critical to achieving optimal performance.

Furthermore, this experiment allows evaluation of the token reduction dynamics across layers. By plotting the frame index on the x-axis and the number of reduced tokens on the y-axis (max of 1024), as depicted in Figure 4.4, the results reveal predictable patterns, with lower thresholds pruning significantly more tokens than those with higher thresholds.

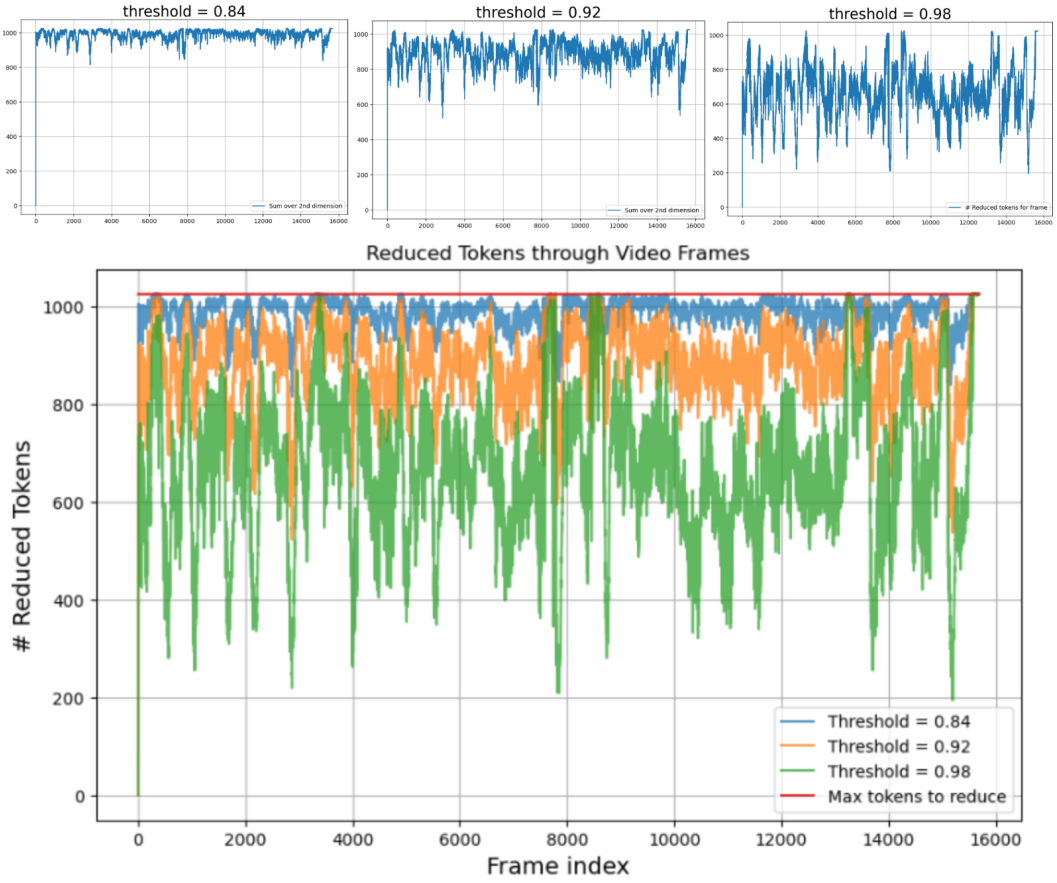

Figure 4.4: Total amount of reduced tokens for each frame in the Gaimersheim video for three different fixed thresholds.

Interestingly, in the graph shown in Figure 4.4 another conclusion can be made. There are sections in the video where the model generally performs well, where tokens are reduced in high quantities. Even at threshold of 0.98, some sections demonstrate extreme reductions where all 1024 tokens are reduced, requiring no recalculations for this frame. But there are also sections where the model has a lower token reduction, regarding of the threshold. This adaptive behavior indicates that the model effectively adjusts to the environment of the vehicle, demonstrating its dynamic processing capabilities.

When examining the layers of the Vision Transformer where token matching predominantly occurs, it becomes evident that most matching is concentrated in the initial layer of the model, as depicted in Figure 4.5. This is followed by several layers with little to no matching activity. This behaviour suggests potential inefficiencies, as

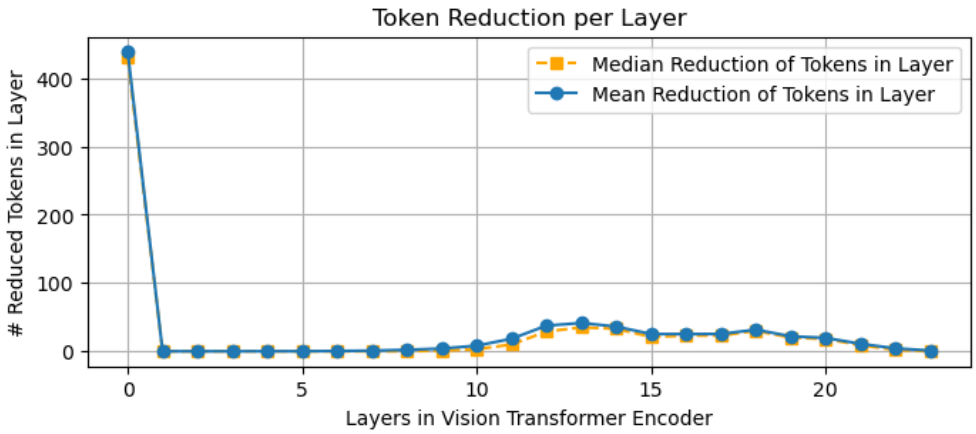

Figure 4.5: Mean and Median amount of reduced tokens for each layer of the Vision Transformer encoder.

significant computations, such as the matching algorithm, are performed without any substantial benefits. To address these inefficiencies, further optimizations can be considered:

- Limiting the number of layers where token reduction is applied
- Transitioning from a fixed threshold to a dynamic approach, such as a linear method where thresholds start at a high value and decrease progressively through subsequent layers.

4.3.2 Linear Threshold

The experiments described in Section 4.3.1 revealed that using a fixed threshold throughout all layers might not be ideal. A significant amount of tokens are matched in the initial layers of the vision transformer encoder, indicating inefficiencies in token processing.

In this follow-up experiment, the token matching process was adjusted to reduce the number of layers where a reduction occurs. Specifically, reductions were applied every six layers, resulting in four reductions overall. Additionally, a new dynamic threshold was introduced next to the fixed ones. It is recalculated for every layer using a linear decreasing approach. This threshold starts at 0.995 in the initial layer and decreases to 0.93 in the last layer. The goal is to make token matching more selective at the beginning and more permissive as processing progresses through the model.

Benchmarking results for this approach were surprising, as seen in Figure 4.6. The linear threshold produced better pixel-wise loss values than the highest fixed threshold tested (0.98), while also reducing processing time compared to the 0.98 threshold. When compared to a fixed threshold of 0.94, the linear approach requires slightly more processing time per frame but delivered significantly improved accuracy.

These findings suggest that a linear threshold approach might offer a more balanced approach offers a more

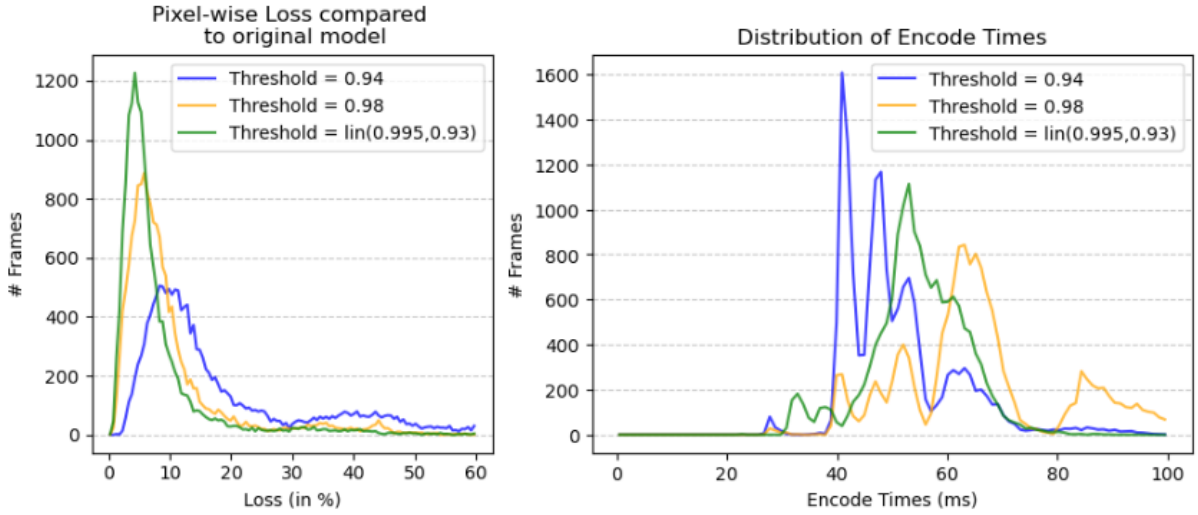

Figure 4.6: (left) Distribution of encode times at different thresholds. (right) Distribution of the Pixel-Wise loss at different thresholds.

balanced approach to the tradeoff between speed and accuracy.

4.3.3 Varying Reduction Intervals

As discussed in Section 4.3.1, applying token reductions in every layer can introduce computational overhead without meaningful benefits. To address this, the linear threshold introduced in Section 4.3.2 was tested with different reduction intervals. For consistency, all intervals tested were divisors of 24.

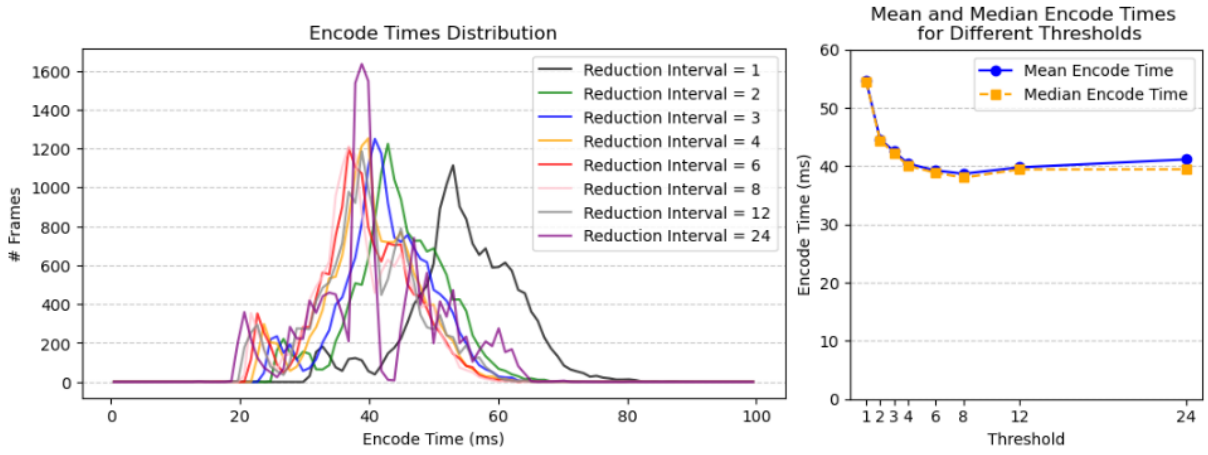

Figure 4.7: (left) Distribution of the duration of the Modified Vision Transformer Encoder using a Linear Threshold from 0.995 to 0.93 with varying Reduction Intervals. (Right) Visualization of the mean and median duration of the encoder on different Reduction Intervals.

Processing time results indicate that most models perform similarly except for the interval of 1, where reductions occur too frequently. This configuration added approximately $0.015s$ to the processing time of the encoder compared to an interval of 8, which shows the lowest mean and median times.

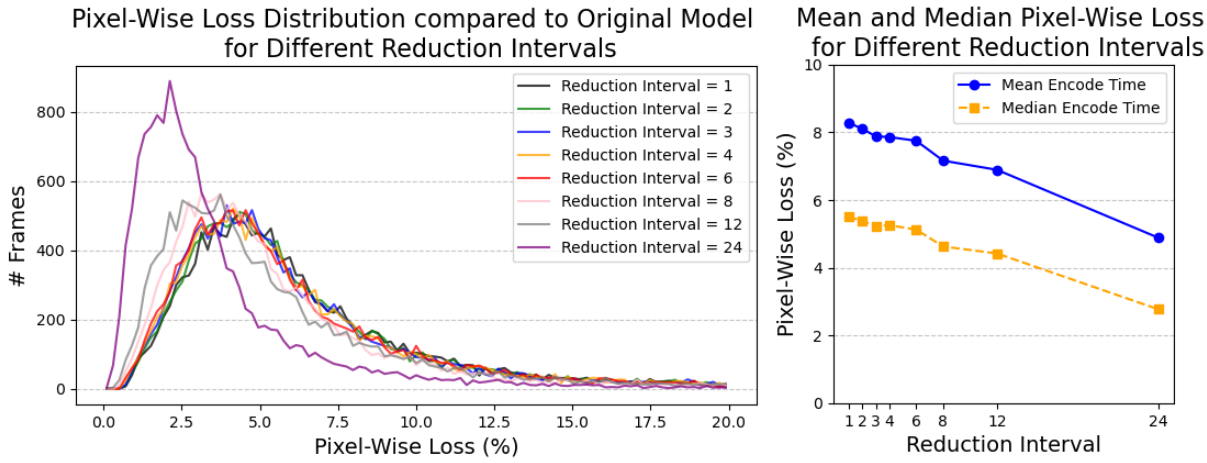

Figure 4.8: (left) Distribution of the Pixel-Wise Loss at different reduction intervals. (right) Visualization of the mean and median duration of the Pixel-Wise Loss for different reduction intervals.

In terms of pixel-wise loss, applying reductions in every layer produced worse results compared to configurations with fewer reductions. This is because retaining more tokens leads to greater computational precision, generating outputs closer to those of an untouched model. However, when reduction occur too frequently, there is an increased risk of output drift.

An interval of 8 appears to strike a good balance between processing efficiency and accuracy, though this optimal interval may vary depending on the application. In some applications it might be better to reduce only once due to the lower loss.

4.3.4 Spatial Distribution of Matched Tokens

Figure 4.9 presents a visualization of the spatial distribution where most tokens are reduced during the token reduction process. This analysis provides insights into the reduction strategy.

The visualization reveals that the model shows a clear preference for reducing tokens in the upper and lower sections of the frame while reducing fewer tokens from the left, right, and central regions. This behavior is caused by a few factors.

Tokens in the upper part of the frame predominantly represent sky or clouds, which are consistent across frames, allowing for efficient matching and reduction. The lower part of the frame typically contains road surfaces, which also display consistent features across consecutive frames. As a result, the model can reliably match road related

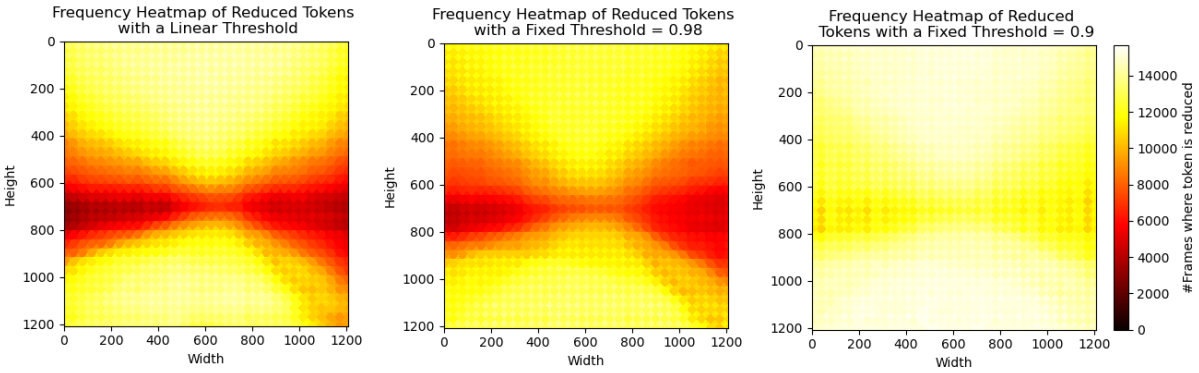

Figure 4.9: Spatial distribution of reduced tokens across different reduction parameters. (Left) A linear threshold from 0.995 to 0.93. (Center) A fixed threshold of 0.98. (Right) A fixed threshold of 0.90. Darker colors indicate regions where tokens are less likely to be reduced, while lighter shows higher reductions.

tokens, leading to frequent reductions in this regions. The middle and side regions of the frame often capture rapidly changing environments such as passing buildings, vehicles, and other dynamic elements. These areas show significant variability, making token matching more challenging and leading to less frequent reductions.

Another interesting insight from the visualization in Figure 4.9 is the lack of vertical line symmetry in the pruning regions. The reduction region on the left side of the frame is slightly larger than that on the right. This asymmetry can be explained by the driving context of the vehicle. Since the car operates on the right side of the road, it is closer to complex and dynamic environments such as buildings on the side. Conversely, on the left side of the vehicle, complex environments such as buildings are farther away from the vehicle, with another lane in between.

4.3.5 Analysis of Matched Tokens

In this section, a more detailed analysis is provided of the tokens matched during the reduction process. When the model determines that a current patch is a match, its token is replaced with that of the previous frame. Ideally, the matched tokens are visually similar. If they differ enormously, they are unlikely to be a good match.

Figure 4.10 illustrates a subset of matched sky tokens. These tokens are generally easy to match, as they show minimal variation over time. While slight differences may arise, such as changes in cloud position or minor noise and artifacts from the camera, the model demonstrates a robust ability to handle these variations effectively. The visual similarity observed in these tokens supports the conclusion that sky patches are inherently easier to match due to their stable and predictable nature, as discussed in Section 4.3.4.

As depicted in Figure 4.11, the matched ground tokens show a similar level of consistency as the sky tokens. While slight variations such as translations or differences in texture may occur, the overall visual similarity is maintained. This consistency is likely due to the uniform nature of ground patches, which often feature repetitive patterns or

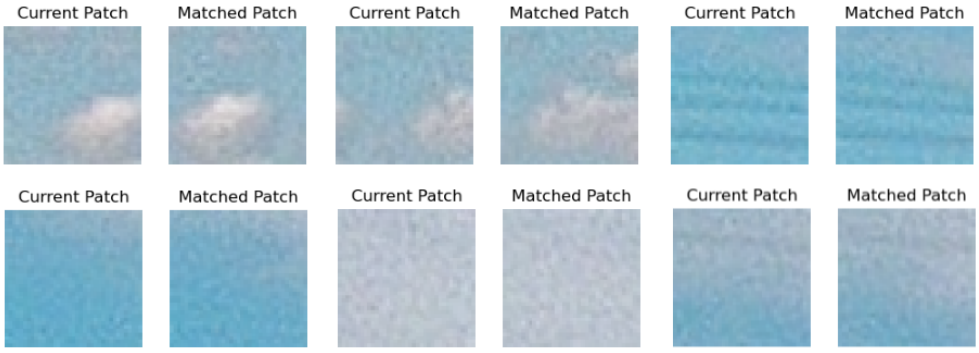

Figure 4.10: Sky Tokens from frame 2125 matched to previous frames.

Figure 4.11: Ground Tokens from frame 2125 matched to previous frames.

textures, making them straightforward for the model to match.

Figure 4.12 focuses on tokens from the left and right sections of the frame. These tokens validate the findings discussed in Section 4.3.4, where it was observed that areas at the sides of the frame are less stable and undergo more frequent changes. Tokens in these regions contain a greater diversity of colors, textures, and objects, leading to increased difficulty in achieving accurate matches.

In these areas, objects may shift position significantly due to rapid motion or changing perspectives. Additionally, parts of previously obstructed objects may become visible, introducing further variability. Despite these challenges, the model is often able to match visually similar objects, though with potentially reduced accuracy compared to the more stable regions like the sky or ground.

4.3.6 Visualization of High and Low Token Reductions

The behavior of the token reduction process is influenced by the environment of the vehicle. This section explores two contrasting scenarios: one with a significant dip in the token reduction, and one with a high reduction rate.

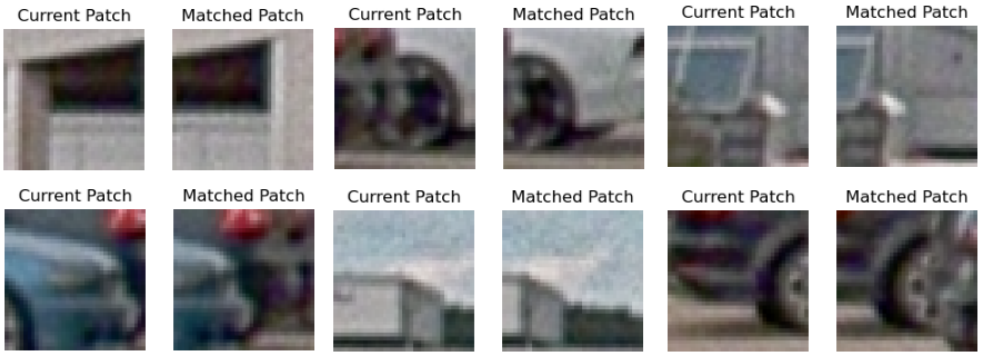

Figure 4.12: Miscellaneous Tokens from frame 2125 matched to previous frames.

Low Token Reduction

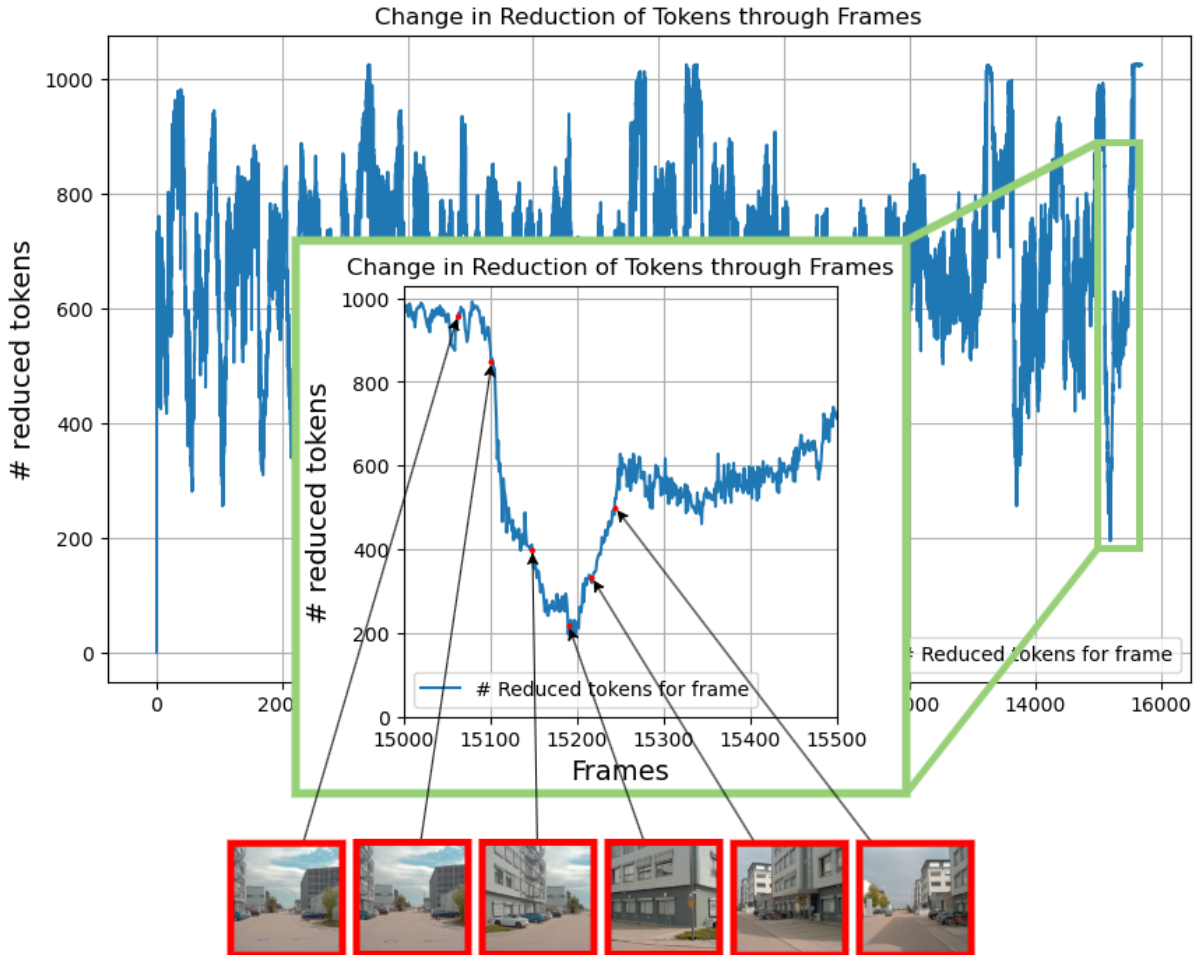

Figure 4.13: Section of the Gaimersheim video where there is a low reduction in tokens

Figure 4.13 illustrates a period during which the model experiences a decrease in token reduction. This occurs when the model struggles to match the current environment with its previously learned context. When examining

the corresponding video segment, it becomes clear that this dip corresponds with a turning maneuver by the vehicle. During this turn, the environment captured by the camera undergoes substantial changes such as the field of view shifting predominantly to a building, decreased road visibility, and changes in the sky from blue with scattered clouds to dense gray clouds after the turn. These rapid environmental changes challenge the model's ability to update its token database in real-time, leading to a temporary reduction in token matching. However, as the vehicle completes the turn and the environment stabilizes, the model adjusts and resumes efficient token reduction.

High Token Reduction

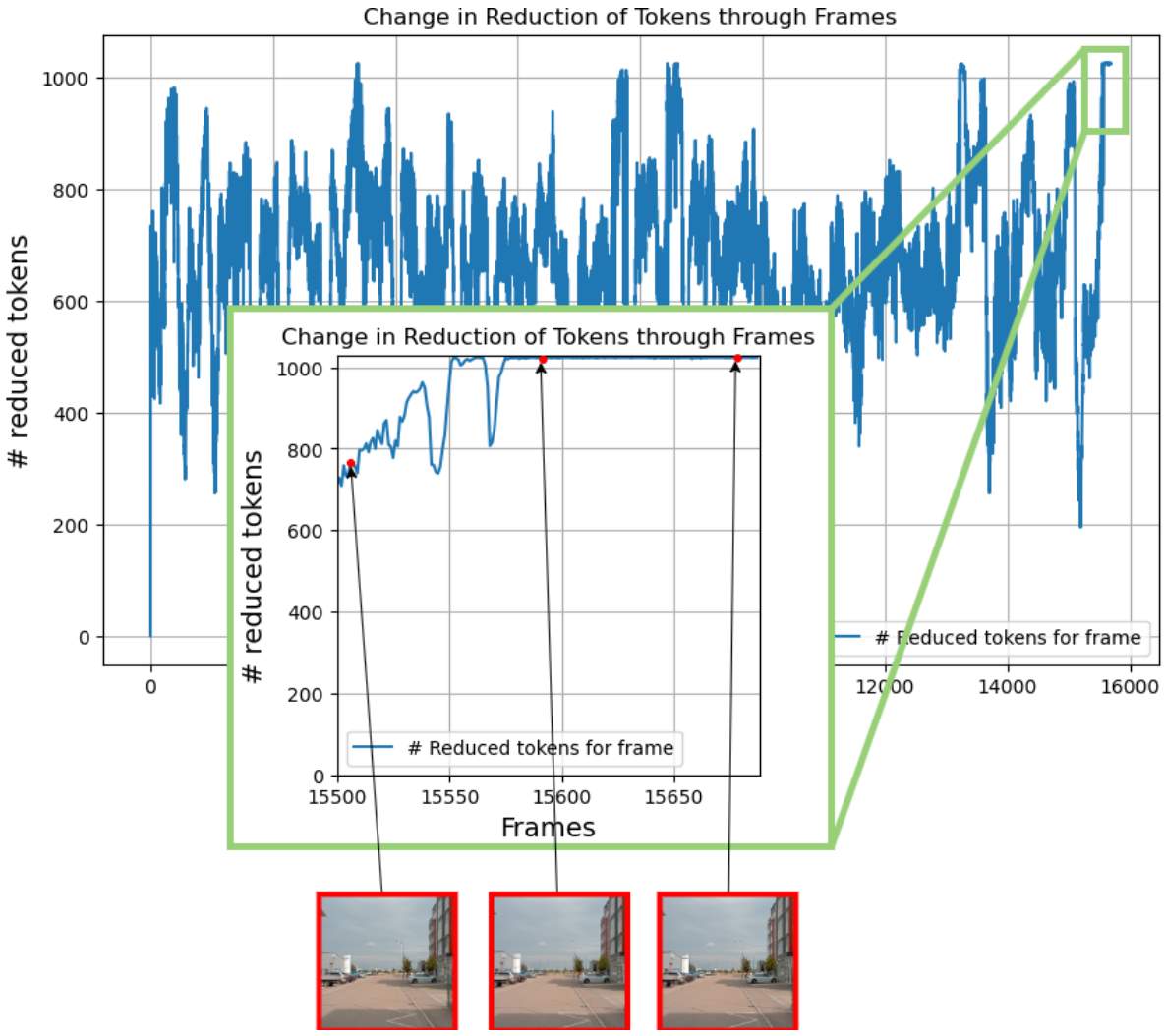

Figure 4.14: Section of the Gaimersheim video where there is a high reduction in tokens

In contrast, Figure 4.14 depicts a scenario with a high token reduction rate. In this segment, the model achieves complete token pruning, eliminating the need for recalculations. Analyzing the associated video frames reveals

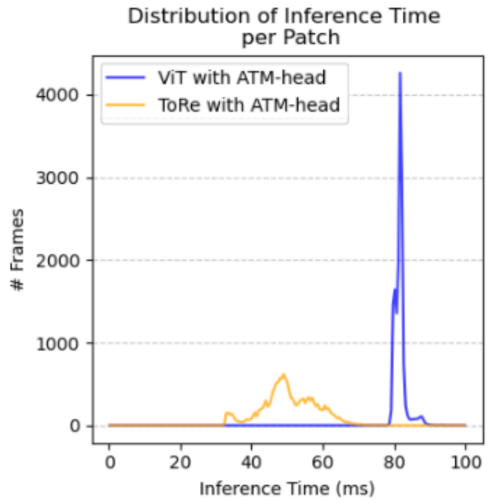

Figure 4.15: Distribution of Inference Time in the Token Reducing Model compared to the Original Model on GPU

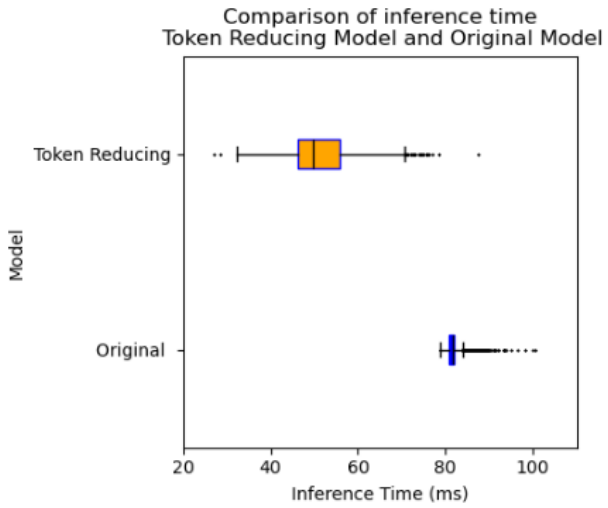

Figure 4.16: Boxplot comparison of inference times of the Token Reducing Model and the Original Model on GPU

that this occurs when the vehicle comes to a stop and the observed environment remains virtually unchanged across consecutive frames. The static nature of the scene allows the model to learn and store the tokens representing the environment in its database and reliably match all tokens from previous frames to the current one without recalculations. This scenario showcases the model's capability to efficiently adapt to static environments, leveraging the lack of changes to optimize its processing.

4.3.7 Optimized Token Reduction Model

This section provides an in-depth benchmarking of an optimized Token-Reducing Model against the Original Model, focusing on inference performance and resource utilization. Key metrics such as inference time, floating-point operations (FLOPs), and Pixel-Wise Loss are analyzed on both GPU and CPU. In the optimized model, a Reduction Interval of 8, identified as optimal in Section 4.3.3, is utilized. Furthermore, the Cosine Similarity threshold decreases linearly from 0.995 to 0.93, resulting in performance gains as discussed in Section 4.3.2.

Inference Performance on GPU

Figure 4.15 shows the distribution of inference times for the original and Token-Reducing models. The original model exhibits a dense peak around $0.08s$, with a mean and median inference time of $0.08159s$ and $0.08158s$, respectively. In contrast, the Token-Reducing Model demonstrates a broader distribution ranging from $0.035s$ to $0.07s$, showing its ability to dynamically adapt to the complexity of input images. The mean inference time for the Token-Reducing Model is $0.05204s$, and the median is $0.051318s$, resulting in a 37% reduction in inference time compared to the original model.

Table 4.2 summarizes the metrics for inference time on GPU.

Table 4.2: Metrics on Inference Time of the Reducing Model compared to Original Model (GPU)

	Mean (ms)	Median (ms)	Standard Deviation (ms)	Variance
ToRe	50.566	49.836	7.594	57.667
Original	81.588	81.584	3.491	12.190

As shown in Figure 4.17, the Token-Reducing Model adapts its computational requirements based on input complexity. On average, it requires 49% fewer FLOPs than the original model, which uses a fixed amount of computations regardless of input characteristics. Figure 4.18 further illustrates the relationship between FLOPs and tokens calculated for that frame, confirming the model's dynamic efficiency.

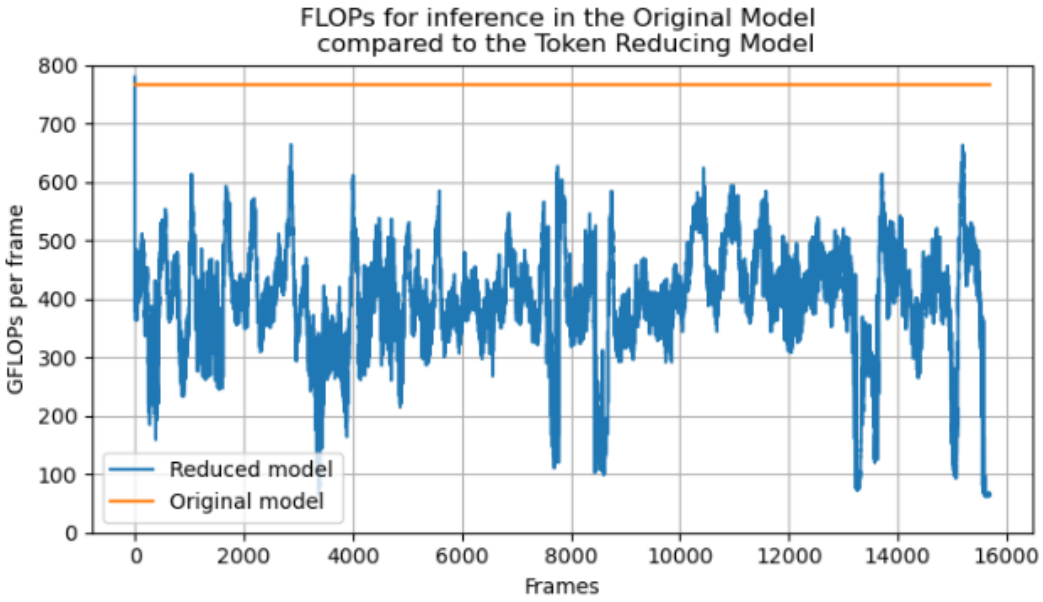

Figure 4.17: FLOPs for inference in the original model compared to the Token Reducing Model.

Table 4.3 provides metrics for FLOPs required per frame.

Table 4.3: Metrics on FLOPS required to Calculate a Frame

	Mean (GFLOPs)	Median (GFLOPS)	Standard Deviation (GFLOPS)
ToRe	393.2502	398.8142	93.6639
Original	766.6132	766.6132	0

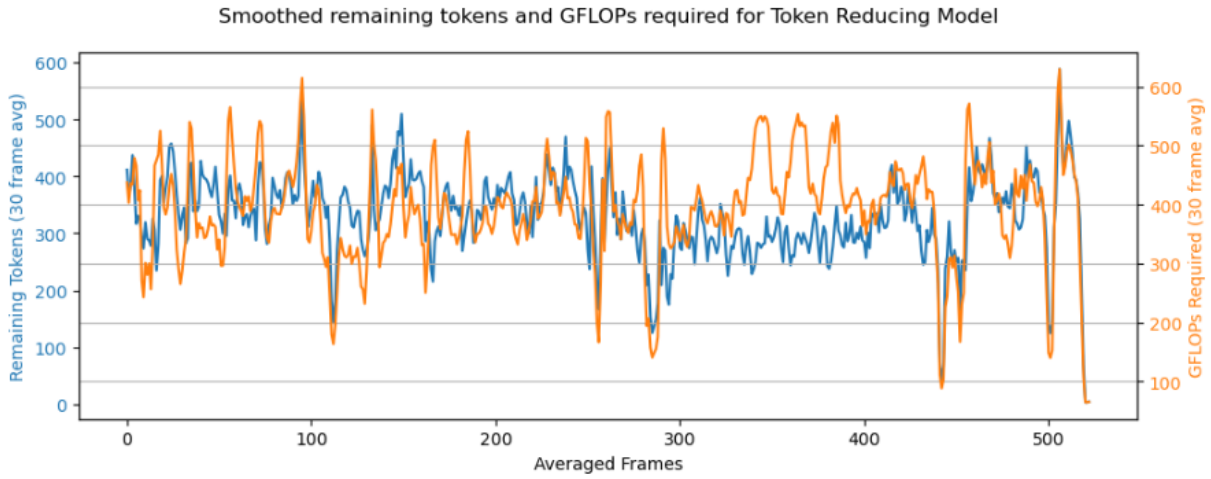

Figure 4.18: Relationship between the tokens not matched in the reduction and the flops required for the inference of that frame.

Figure 4.19 depicts the distribution of pixel-wise loss between the two models. The Token-Reducing Model achieves a mean pixel-wise loss of 7.16026% and a median of 4.62452%. While some outliers exhibit higher losses up to 79.5%, the overall accuracy remains acceptable.

Table 4.4 summarizes the metrics for inference time on GPU.

Table 4.4: Metrics on Loss ToRe Model compared to Original Model

	Mean (%)	Median (%)	Standard Deviation (%)	Variance
Loss	7.16026	4.62452	8.96581	80.38578

Inference Performance on CPU

Figure 4.20 and Figure 4.21 compare the inference time distributions of the original and Token-Reducing models on CPU. The original model shows a dense range between $4s$ and $5s$, with a mean of $4.49216s$ and a median of $4.38176s$. In contrast, the Token-Reducing Model achieves a wider distribution between $1.7s$ and $4s$, with a mean of $2.74732s$ and a median of $2.61663s$, representing a 40% reduction in inference time. These gains are comparable to GPU performance.

Table 4.5 provides the CPU inference time metrics.

The Token-Reducing Model retains the same accuracy and computational efficiency on the CPU as observed on the GPU, provided the model initialization is identical, and no randomness is introduced.

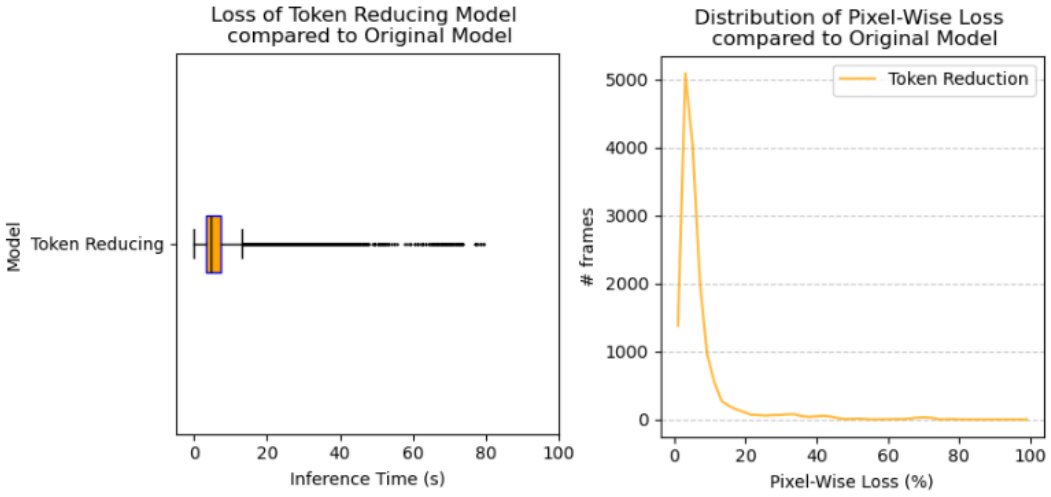

Figure 4.19: (left) Boxplot of the Pixel-Wise Losses of every frame in the Gaimersheim video (right) Distribution of the Pixel-Wise Losses in the frames of the same video

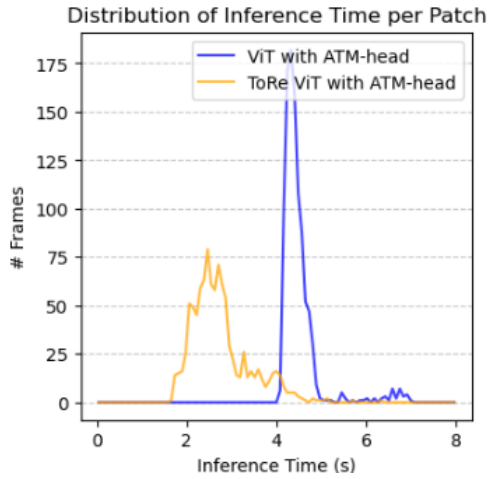

Figure 4.20: Distribution of Inference Time in the Token Reducing Model compared to the Original Model on CPU

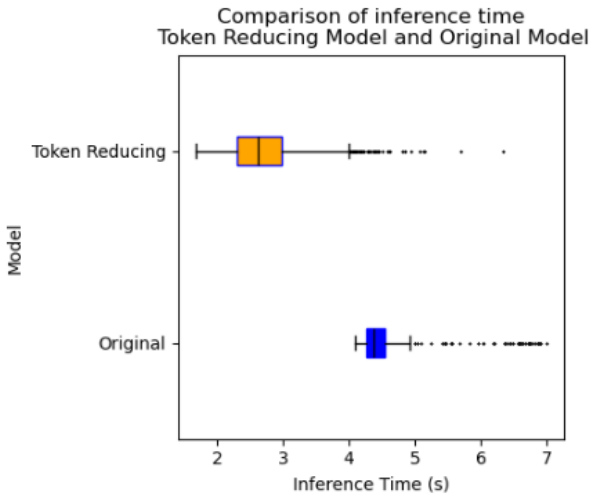

Figure 4.21: Boxplot comparison of inference times of the Token Reducing Model and the Original Model on CPU

4.3.8 Frame Processing Speed and Vehicle Movement

The processing times of the original model and the token-reducing model allow for an estimation of the distance a vehicle travels between the calculation of consecutive frames. This relationship is visualized in Figure 4.22. The results indicate that the token reduction model can save approximately one meter of distance traveled between frames at a speed of 120 km/h compared to the original model.

This efficiency gain presents several potential benefits. The saved computation time could be used to increase the frame rate, allocate GPU resources for other critical tasks, or conserve energy. These options allow for dynamic

Table 4.5: Metrics on Inference Time of the Reducing Model compared to Original Model (CPU)

	Mean (s)	Median (s)	Standard Deviation (s)	Variance
ToRe	2.74732	2.61663	0.64677	0.41831
Original	4.49216	4.38176	0.45355	0.20571

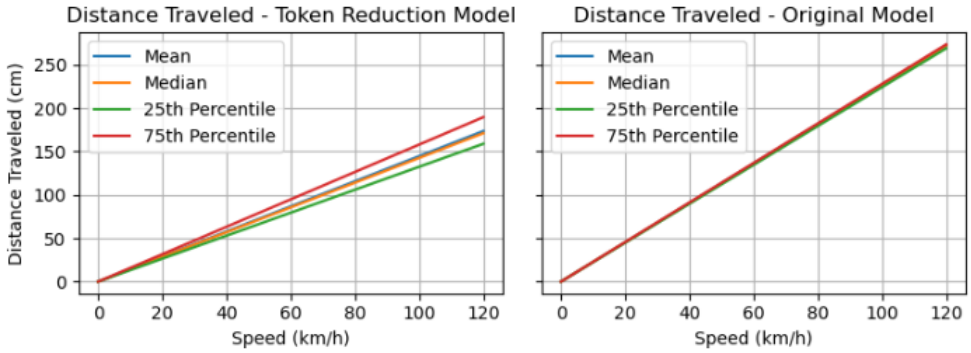

Figure 4.22: Comparison of frame processing times between the original encoder and decoder, and the token-reducing encoder with the original decoder, highlighting the potential distance saved during processing at different speeds.

optimization tailored to the requirements of the driving scenario.

However, it is important to note that this conclusion is speculative to some extent. The model's performance at higher speeds remains unverified, as the dataset used for evaluation does not include speed-related data. Consequently, further testing with speed-variable datasets would be necessary to validate the model's performance across different speeds.

4.4 Limitations

This thesis is subject to several limitations that influence the scope and applicability of its findings.

4.4.1 Benchmarking Constraints

The absence of ground truth data in the dataset used for benchmarking results in that pixel-wise accuracy could not be directly compared to a reference ground truth. Instead, comparisons were made to the original model's outputs, using pixel-wise loss to evaluate performance differences. While this approach is reasonable given that the thesis focuses on optimizing the existing architecture rather than retraining. However, it does not provide a direct measure of how well the token-reducing model would perform against a ground truth. This restricts the ability to evaluate absolute performance and instead highlights relative performance.

4.4.2 Vehicle Speed Considerations

The dataset lacks information about the vehicle's speed, which is another potential limitation. The models performance may vary at different speeds, as higher speeds introduce rapid environmental changes that could impact token matching and reduction. Without testing across varying speeds, it is difficult to generalize the model's performance gains for real-world scenarios where speed changes are common.

4.4.3 Limited Environmental Variability

The dataset primarily includes videos captured under similar weather conditions, such as clear skies or overcast days. Consequently, the model was not evaluated in diverse conditions such as nighttime, rain, or snow. These conditions could present unique challenges for token matching and reduction, potentially affecting models performance.

4.4.4 Multi-Source Inference

The current implementation is optimized for processing a single frame at a time from one camera feed. When analyzing multiple frames from multiple cameras simultaneously, separate model instances would be required to avoid database conflicts. The current implementation lacks support for multi-source inference using a shared model, which could limit scalability in multi-camera systems.

4.4.5 Image Dimensions and Scaling

The model used is trained on square images of 512×512, limiting its ability to process images with larger dimensions or different aspect ratios without retraining. Inferencing images at their original size results in unusable outputs due to mismatched class embeddings. To address this, a center crop and downscale to 512×512 was applied. While this workaround ensures functionality by focusing on the central region where most relevant action occurs, it restricts the model's ability to leverage the full spatial context of larger images.

5

Conclusion

This master's dissertation researched how semantic segmentation models using vision transformers can be made more efficient, with a focus on niche deployments: automotive applications. Unlike other scenarios, the frames processed in this context tend to be highly similar in consecutive frames, as a vehicle's position and surroundings change incrementally over time. This research explored whether previously calculated information could be used to reduce computational load and inference time while maintaining accuracy comparable to the original model. Importantly, the approach was designed to enhance inference without retraining the model, focusing on improvements at the inference stage. This makes the solution more practical and accessible for deployment.

In this study, a token reduction and reconstruction block was proposed. This block prunes tokens passing through the encoder by matching them against a token database. When a token matches a token in the database, a processed version of that token is reintroduced to the token set before it proceeds to the decoder. This method effectively reduces the number of tokens processed by the encoder.

The study found that this approach reduced the median inference time of the encoder by approximately 37%, alongside a 49% reduction in FLOPS required, compared to baseline model without token reducing. Additionally, the method achieved a median pixel-wise loss of 4.62% compared to the original output, demonstrating a promising balance between computational efficiency and output accuracy.

These results suggest that leveraging a database of previously calculated context is a viable strategy for optimizing vision transformers in the automotive domain. The potential for reducing inference time and computational resources while preserving accuracy indicates that this is a promising direction for further exploration.

While this research provided valuable insights, several limitations should be acknowledged. The dataset used was relatively limited and lacked diversity, as it did not include challenging weather conditions such as snow, storms, or mist, nor did it account for varying daylight scenarios such as night, dusk, or dawn. Additionally, the model's performance at different vehicle speeds was not evaluated, leaving an open question about its adaptability to such variations. Furthermore, absolute accuracy could not be calculated due to the absence of ground truth for the video sequences.

Future research should aim to overcome these limitations by evaluating the model across a broader range of conditions, including diverse weather scenarios, lighting conditions, and vehicle speeds. Another valuable addition would be to extend the approach to handle multi-input configurations, enabling the simultaneous processing of video feeds from multiple cameras.

Finally, this research highlights the importance of adaptive architectures in vision transformer models, as they have the potential to reshape how machine learning models are designed for various applications. By tailoring architectures to the specific requirements of a task, significant gains in efficiency and performance can be achieved. Furthermore, this adaptability could influence future trends in machine learning, encouraging a shift toward more flexible and resource-efficient model designs, particularly in domains where computational resources are limited or real-time processing is critical. It demonstrates the need to revisit fixed architectures to optimize models for specific applications. By doing so, substantial computational resources can be saved without compromising the quality of results.

Future Work

The database implemented uses a cold-start mechanism. This implies that, upon initialization, the database is empty. To address this, a shutdown process could be implemented to save the current state of the database to permanent storage, enabling it to be reloaded during the next initialization. This approach assumes that the environment of the car remains unchanged (for instance, its location, weather conditions, and time of day).

Exploring alternative database designs could result in significant improvements. The current design implements a circular buffer that is overwritten every few frames. However, a priority queue design might offer advantages by organizing tokens based on specific criteria. For example, tokens could be prioritized by age, with newer tokens ranked higher in the queue, or by a counter tracking their matching frequency, allowing tokens that remain unmatched for several frames to be removed.

The database architecture could also be adapted to support multi-camera feed inference. This would involve creating separate databases for each camera feed, which is not currently implemented but could improve scalability. Additionally, the matching algorithm would need to be modified to process inputs from multiple tensors, ensuring that each is compared only to its corresponding database. A larger single database could also work, but might introduce a longer matching process.

Retraining or fine-tuning the model has the potential to improve its overall accuracy. Testing the architecture with more diverse datasets could provide valuable insights into its performance and limitations.

Investigating the architecture's behavior under varying speeds could reveal interesting findings. Parameters such as the token reduction interval and matching threshold could be adjusted based on speed or environmental conditions, optimizing performance for different scenarios.

The current architecture demonstrates the ability to reduce inference time and computational load. However, further research is needed to determine the optimal utilization of these efficiency gains. Potential applications include processing more frames, analyzing additional information, or conserving resources. Dynamic adjustments to these applications based on the car's environment could also be explored.

Sustainability reflection

Broader Societal Context

This thesis addresses the integration of artificial intelligence models in the automotive industry, particularly in computer vision and AI transformation. The primary objective is to optimize existing AI architectures for applications in vehicles, thereby contributing to the ongoing digital transformation within the sector. These developments not only enhance automotive functionality but also reflect the broader societal shift toward automation and intelligent systems, supporting advancements in mobility and safety.

Contribution to Sustainable Development Goals (SDGs)

This project aligns with the United Nations Sustainable Development Goals (SDGs), specifically:

- Goal 9: Industry, Innovation, and Infrastructure: By improving AI efficiency and optimizing its implementation in vehicles, this work supports innovative industrial practices, contributing to smarter, more sustainable transportation systems.[51]
- Goal 13: Climate Action: Energy-efficient AI models are critical to enhancing the viability of electric vehicles. Reduced power consumption directly impacts the range and sustainability of electric cars, thereby supporting efforts to mitigate climate change.[51]

Environmental Impact

As AI models become increasingly common in modern vehicles, their energy efficiency is paramount. Optimizing these models ensures their viability, particularly for electric vehicles, where power consumption is a critical factor. By minimizing energy usage, this project not only advances technological innovation but also promotes environmental sustainability.

Privacy and GDPR Compliance

The proposed AI model adheres to the General Data Protection Regulation (GDPR) standards [52, 53, 54]. While the system captures identifiable data such as faces and license plates, several measures ensure compliance:

- Data Minimization: Only the necessary RGB data is captured, and photos are discarded immediately after tokenization. Tokens are stored temporarily and overwritten as new data is processed or the model is shut down.

- Legitimate Interests: The data usage serves legitimate interests without infringing on individuals' rights and freedoms.
- Transparency: Individuals must be informed about the data processing activities. This is out of scope for this thesis.
- Data Protection Impact Assessment (DPIA): Conducting a DPIA ensures thorough evaluation and compliance with privacy standards. This is out of scope for this thesis

Additionally, this thesis complies with Belgian privacy laws, as no captured material is stored or shared with third parties, ensuring that personal data remains secure.[55]

Explainability

To foster trust and informed decision-making, users must understand the trade-offs inherent in the model's architecture. Clear documentation of its limitations and benefits enables stakeholders to assess its suitability for specific applications.

Ethical Considerations

While the model is designed to enhance efficiency in vehicles, its potential for dual-use scenarios must be acknowledged. For instance, the architecture could be repurposed for surveillance applications, raising ethical concerns.

The model achieves energy savings at the cost of a minor reduction in accuracy. Determining whether this trade-off is acceptable requires careful evaluation, especially in safety-critical applications.

If further testing reveals poor performance under certain conditions, deploying the model in production vehicles may not be ethically justifiable.

Use of Generative AI

This master dissertation used generative AI tools to assist in some aspects of the research and writing process.

ChatGPT, Copilot

- Generating tables and images in LaTeX format
- Formatting LaTeX content, formulas, ...
- Exploring synonyms, alternative descriptions, different writing styles
- Explaining complex concepts such as efficient manipulation of tensors on GPU, the MMEngine platform and its configuration, ...

Github Copilot

- Formatting result tables for analysis

DeepL

- Translating individual words or sentences from dutch to english or vice-versa.

Grammarly

- Grammar and spelling check

References

[1] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Mühlegg, S. Dorn, T. Fernandez, M. Jänicke, S. Mirashi, C. Savani, M. Sturm, O. Vorobiov, M. Oelker, S. Garreis, and P. Schuberth, "A2D2: Audi Autonomous Driving Dataset," 2020. [Online]. Available: https://www.a2d2.audi

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[4] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, "Segmenter: Transformer for semantic segmentation," *CoRR*, vol. abs/2105.05633, 2021. [Online]. Available: https://arxiv.org/abs/2105.05633

[5] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "Segformer: Simple and efficient design for semantic segmentation with transformers," in *Neural Information Processing Systems (NeurIPS)*, 2021.

[6] B. Zhang, Z. Tian, Q. Tang, X. Chu, X. Wei, C. Shen, and Y. Liu, "Segvit: Semantic segmentation with plain vision transformers," *NeurIPS*, 2022.

[7] H. Yin, A. Vahdat, J. M. Álvarez, A. Mallya, J. Kautz, and P. Molchanov, "Adavit: Adaptive tokens for efficient vision transformer," *CoRR*, vol. abs/2112.07658, 2021. [Online]. Available: https://arxiv.org/abs/2112.07658

[8] Q. Tang, B. Zhang, J. Liu, F. Liu, and Y. Liu, "Dynamic token pruning in plain vision transformers for semantic segmentation," 2023. [Online]. Available: https://arxiv.org/abs/2308.01045

[9] Y. Liu, Q. Zhou, J. Wang, F. Wang, J. Wang, and W. Zhang, "Dynamic token-pass transformers for semantic segmentation," 2023. [Online]. Available: https://arxiv.org/abs/2308.01944

[10] S. Ding, P. Zhao, X. Zhang, R. Qian, H. Xiong, and Q. Tian, "Prune spatio-temporal tokens by semantic-aware temporal accumulation," 2023. [Online]. Available: https://arxiv.org/abs/2308.04549

[11] R. I. of Carnegie Mellon University, "Navlab," 2025, accessed: 2024-08-29. [Online]. Available: https://www.ri.cmu.edu/robotics-groups/navlab/

[12] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

5 References

[13] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[14] A. Ayanzadeh, "A study review: Semantic segmentation with deep neural networks," 03 2018.

[15] S. Bisoy, q. sellot, R. Priyadarshini, D. R. Barik, A. Vidyarthi, and S. Kautish, "Intelligent semantic segmentation for self-driving vehicles using deep learning," 01 2022.

[16] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, L. Màrquez, C. Callison-Burch, and J. Su, Eds. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 1412–1421. [Online]. Available: https://aclanthology.org/D15-1166

[17] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs." in *ICLR (Poster)*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://dblp.uni-trier.de/db/conf/iclr/iclr2015.html#ChenPKMY14

[18] M. Yang, K. Yu, C. Zhang, Z. Li, and K. Yang, "Denseaspp for semantic segmentation in street scenes," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3684–3692.

[19] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," 2016. [Online]. Available: https://arxiv.org/abs/1511.07122

[20] Z. Huang, X. Wang, Y. Wei, L. Huang, H. Shi, W. Liu, and T. S. Huang, "Ccnet: Criss-cross attention for semantic segmentation," *CoRR*, vol. abs/1811.11721, 2018. [Online]. Available: http://arxiv.org/abs/1811.11721

[21] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Learning a discriminative feature network for semantic segmentation," *CoRR*, vol. abs/1804.09337, 2018. [Online]. Available: http://arxiv.org/abs/1804.09337

[22] Z. Zhong, Z. Q. Lin, R. Bidart, X. Hu, I. B. Daya, J. Li, and A. Wong, "Squeeze-and-attention networks for semantic segmentation," *CoRR*, vol. abs/1909.03402, 2019. [Online]. Available: http://arxiv.org/abs/1909.03402

[23] B. Zhang, L. Liu, M. H. Phan, Z. Tian, C. Shen, and Y. Liu, "Segvitv2: Exploring efficient and continual semantic segmentation with plain vision transformers," *IJCV*, 2023.

[24] H. Thisanke, C. Deshan, K. Chamith, S. Seneviratne, R. Vidanaarachchi, and D. Herath, "Semantic segmentation using vision transformers: A survey," 2023. [Online]. Available: https://arxiv.org/abs/2305.03273

[25] A. Gillioz, J. Casas, E. Mugellini, and O. Abou Khaled, "Overview of the transformer-based models for nlp tasks," 09 2020, pp. 179–183.

5 References

[26] S. Singh and A. Mahmood, "The nlp cookbook: Modern recipes for transformer based deep learning architectures," *IEEE Access*, vol. 9, pp. 68 675–68 702, 2021.

[27] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014. [Online]. Available: http://arxiv.org/abs/1409.3215

[28] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[29] K. S. Kalyan, A. Rajasekharan, and S. Sangeetha, "AMMUS : A survey of transformer-based pretrained models in natural language processing," *CoRR*, vol. abs/2108.05542, 2021. [Online]. Available: https://arxiv.org/abs/2108.05542

[30] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[31] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, and Z. Chen, "Gshard: Scaling giant models with conditional computation and automatic sharding," *CoRR*, vol. abs/2006.16668, 2020. [Online]. Available: https://arxiv.org/abs/2006.16668

[32] S. Islam, H. Elmekki, A. Elsebai, J. Bentahar, N. Drawel, G. Rjoub, and W. Pedrycz, "A comprehensive survey on applications of transformers for deep learning tasks," 2023. [Online]. Available: https://arxiv.org/abs/2306.07303

[33] X. Chen, Y. Wu, Z. Wang, S. Liu, and J. Li, "Developing real-time streaming transformer transducer for speech recognition on large-scale dataset," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 5904–5908.

[34] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented transformer for speech recognition," 2020. [Online]. Available: https://arxiv.org/abs/2005.08100

[35] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, and A. Ku, "Image transformer," *CoRR*, vol. abs/1802.05751, 2018. [Online]. Available: http://arxiv.org/abs/1802.05751

5 References

[36] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, "Vivit: A video vision transformer," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 6816–6826.

[37] Z. Liu, J. Ning, Y. Cao, Y. Wei, Z. Zhang, S. Lin, and H. Hu, "Video swin transformer," *CoRR*, vol. abs/2106.13230, 2021. [Online]. Available: https://arxiv.org/abs/2106.13230

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[39] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, cite arxiv:1607.06450. [Online]. Available: http://arxiv.org/abs/1607.06450

[40] P. Dierickx, A. Damme, N. Dupuis, and O. Delaby, "Comparison between cnn, vit and cct for channel frequency response interpretation and application to g.fast," *IEEE Access*, vol. 11, pp. 24 039–24 052, 01 2023.

[41] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. S. Torr, and L. Zhang, "Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers," *CoRR*, vol. abs/2012.15840, 2020. [Online]. Available: https://arxiv.org/abs/2012.15840

[42] A. Kirillov, R. B. Girshick, K. He, and P. Dollár, "Panoptic feature pyramid networks," *CoRR*, vol. abs/1901.02446, 2019. [Online]. Available: http://arxiv.org/abs/1901.02446

[43] Z. Shen, M. Zhang, S. Yi, J. Yan, and H. Zhao, "Factorized attention: Self-attention with linear complexities," *CoRR*, vol. abs/1812.01243, 2018. [Online]. Available: http://arxiv.org/abs/1812.01243

[44] D. Bolya, C.-Y. Fu, X. Dai, P. Zhang, C. Feichtenhofer, and J. Hoffman, "Token merging: Your ViT but faster," in *International Conference on Learning Representations*, 2023.

[45] C. Yu, J. Wang, C. Gao, G. Yu, C. Shen, and N. Sang, "Context prior for scene segmentation," *CoRR*, vol. abs/2004.01547, 2020. [Online]. Available: https://arxiv.org/abs/2004.01547

[46] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ADE20K dataset," *CoRR*, vol. abs/1608.05442, 2016. [Online]. Available: http://arxiv.org/abs/1608.05442

[47] H. Caesar, J. R. R. Uijlings, and V. Ferrari, "Coco-stuff: Thing and stuff classes in context," *CoRR*, vol. abs/1612.03716, 2016. [Online]. Available: http://arxiv.org/abs/1612.03716

[48] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[49] OpenMMLab, "Mmsegmentation," 2025, accessed: 2024-04-23. [Online]. Available: https://github.com/open-mmlab/mmsegmentation

[50] ——, "Mmengine," 2025, accessed: 2024-04-23. [Online]. Available: https://github.com/open-mmlab/mmengine

[51] U. Nations, "Sustainable development goals," 2025, accessed: 2025-01-02. [Online]. Available: https://sdgs.un.org/goals

[52] europa.eu, "edps.europa.eu," 2025, accessed: 2025-01-02. [Online]. Available: https://www.edps.europa.eu/data-protection_en

[53] ICO, "Information commisioner's office," 2025, accessed: 2025-01-02. [Online]. Available: ico.org.uk

[54] EU, "Official journal of the european union," 2025, accessed: 2025-01-02. [Online]. Available: https://eur-lex.europa.eu/legal-content/NL/TXT/HTML/?uri=CELEX:32016R0679&from=EN

[55] Secunews, "Politie.be," 2025, accessed: 2025-01-02. [Online]. Available: https://www.politie.be/5415/vragen/verkeer/dashcams-wat-mag-in-belgie

Appendices

Attachment A: Matching Algorithm

```
a = metric        # tokens from the image, Shape: (1, x, 1024)
b = token_history # tokens from database,  Shape: (1, x, 1024)

a_norm = a.norm(dim=-1, keepdim=True)  # Length of each token as vector, Shape: (1, x, 1)
b_norm = b.norm(dim=-1, keepdim=True)  # Length of each token as vector, Shape: (1, x, 1)

dot_product  = a @ b.transpose(-1, -2)           # above cosine similarity
norm_product = a_norm @ b_norm.transpose(-1, -2) # below cosine similarity

norm_product = norm_product + 1e-8  # Introduce epsilon to prevent division by 0

cosine_similarity = dot_product / norm_product # Cosine Similarity

# Find max tokens from database for each token from image
node_max, node_idx = cosine_similarity.max(dim=-1)

# apply maks to tokens exceeding threshold
mask = node_max > r

if torch.any(mask): # Token Reduction
    src_idx = torch.where(mask)[1]
    src_idx = src_idx.unsqueeze(0).unsqueeze(-1)

    dst_idx = node_idx[..., None].gather(dim=-2, index=src_idx)
else: # No Token Reduction
    src_idx = None
    dst_idx = None

unm_idx = torch.where(~mask)[1]
unm_idx = unm_idx.unsqueeze(0).unsqueeze(-1)
```

Code Fragment 1: Matching algorithm of tokens using cosine similarity.

Attachment B: Reduction Algorithm

```
def reduce(x: torch.Tensor) -> torch.Tensor:

    if(src_idx is not None): # If there were tokens matched

        src = x # the source of tokens is just x

        n, t1, c = src.shape

        _, t2, _ = unm_idx.shape

        # return all tokens that haven't been matched, this becomes the 'reduced token set'
        unmatched_tokens = src.gather(dim=-2, index=unm_idx.expand(n, t2, c))
    else:
        # no matches, no reduction
        unmatched_tokens = x
    return unmatched_tokens, unm_idx, src_idx, dst_idx
```

Code Fragment 2: Reduction of the token set after the matching algorithm.

Attachment C: Reconstruction Algorithm

```
def reconstruct(
    x: torch.Tensor,
    unm_idx: torch.Tensor,
    src_idx: torch.Tensor,
    dst_idx: torch.Tensor,
    out_layer):

    if src_idx is not None: # If there were tokens matched

        unm = x # origianl tensor to be extended
        dst = token_history_processed[out_layer] # processed equivalents of the tokens matched

        n, unm_len, c = unm.shape  # Shape of the input
        r = dst_idx.size(1)        # Reduction size
        total_len = unm_len + r    # Length of the output

        # Gather the src tokens from `dst` at positions specified by `dst_idx`
        src = torch.index_select(dst, dim=1, index=dst_idx.view(-1)).view(n, r, c)

        # Concatenate the full unm tokens and the source tokens
        # This tensor has the size of the output tensor needed
        combined = torch.cat((unm, src), dim=1)

        # Concatenate the index tensors.
        indices = torch.cat((unm_idx, src_idx), dim=1).expand(n, total_len, c)

        # Use the indeces to fill in the full tokens in the right place
        out = torch.zeros_like(combined, device=x.device).scatter_(
            dim=1, index=indices, src=combined)
        return out
    else: # No reconstruction required
        return x
```

Code Fragment 3: Reconstruction of the token set at the end of the Vision Transformer.