

Vision Transformer Interpretability for Video-based Action Recognition

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Data Science

eingereicht von

Raquel Panadero Palenzuela, BSc

Matrikelnummer 12231231

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof.in Dipl.-Ing.in Mag. Dr.in Margrit Gelautz

Mitwirkung: Projektass. Dipl.-Ing. Dominik Schörkhuber

Wien, 21. September 2025

Raquel Panadero Palenzuela

Margrit Gelautz



Vision Transformer Interpretability for Video-based Action Recognition

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Data Science

by

Raquel Panadero Palenzuela, BSc

Registration Number 12231231

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof.in Dipl.-Ing.in Mag. Dr.in Margrit Gelautz

Assistance: Projektass. Dipl.-Ing. Dominik Schörkhuber

Vienna, 21st September, 2025

Raquel Panadero Palenzuela

Margrit Gelautz



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Raquel Panadero Palenzuela, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 21. September 2025

Raquel Panadero Palenzuela



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Danksagung

Zuallererst möchte ich mich ganz besonders bei Prof. Margrit Gelautz für ihre Unterstützung, Begleitung und Beratung beim Schreiben dieser Arbeit bedanken. Ich bin wirklich dankbar für die Möglichkeit, meine Arbeit unter ihrer Leitung durchgeführt haben zu dürfen. Ein besonderer Dank geht an Dominik Schörkhuber für seine Beratung und Unterstützung in den letzten Monaten.

Ich möchte meiner Familie und meinen Freunden in Barcelona von ganzem Herzen für ihre Unterstützung aus der Ferne danken. Ein besonderer Dank geht an meine Yayita – dafür, dass sie mich bei meinen Besuchen immer mit meinen Lieblingsgerichten empfängt, bedingungslos an mich glaubt und mich so liebt, wie ich bin. Meiner Cousine danke ich für das Lachen und die Tränen, die wir während unseren endlosen FaceTime-Gespräche geteilt haben. Deine Liebe und Ermutigung haben mir viel bedeutet.

Ich bin auch meinen Freunden in Wien zutiefst dankbar, die diese letzten Jahre wirklich unvergesslich gemacht haben. Und meiner besten Freundin und der besten Partnerin, die ich mir wünschen kann – danke, Verena, für deine Unterstützung, deine Hilfsbereitschaft, deine sanften Umarmungen und dafür, dass ich mich bei dir immer wie zu Hause fühle. Ich bin so dankbar, dich in meinem Leben zu haben.

Diese Diplomarbeit wurde größtenteils durch das Projekt SyntheticCabin (Projektnummer 884336) finanziert, welches im Rahmen des FTI-Programms Mobilität der Zukunft durch das Bundesministerium für Klimaschutz (BMK) gefördert und von der Österreichischen Forschungsförderungsgesellschaft (FFG) abgewickelt wird. In der Endphase wurde die Diplomarbeit durch die Wirtschaftsagentur Wien unter dem Projekt Empathic Vehicle (Projektnummer 4998519, Call Tech4People) unterstützt. Wir danken Projektpartner emotion3D für die Zusammenarbeit und die zur Verfügung gestellten Datensätze und Bilder.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

First and foremost, I would especially like to thank Prof. Margrit Gelautz for her support, guidance and advice during this thesis. I am truly grateful for the opportunity to conduct my thesis within her group. Special thanks to Dominik Schörkhuber for his advice and assistance during these last months.

I would like to express my heartfelt gratitude to my family and friends in Barcelona for their support from afar throughout this journey. A special thank you to my yayita – for always welcoming me with my favorite meals when I come visit, for believing in me unconditionally, and for loving me just as I am. To my cousin, thank you for sharing both laughter and tears during our endless FaceTime conversations. Your love and encouragement have meant the world to me.

I am also deeply thankful to my friends in Vienna, who have made these past years truly unforgettable. And to my best friend and the best partner I could ever ask for – thank you, Verena, for your support, your kindness, your soft hugs, and for always making me feel at home. I am so grateful to have you in my life.

This thesis was supported for the most part by the project SyntheticCabin (Project Number 884336), which was funded through the Austrian Research Promotion Agency (FFG) on behalf of the Austrian Ministry of Climate Action (BMK) via its Mobility of the Future funding programme. During the final stage of the thesis, the work received support from Vienna Business Agency under the project Empathic Vehicle (Project Number 4998519, Call Tech4People). We thank project partner emotion3D for the collaboration and the provided datasets and images.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

Die Erkennung von Verhaltensweisen von Autofahrer*innen ist ein wichtiger Bestandteil intelligenter Überwachungssysteme im Fahrzeuginnenraum, die darauf abzielen, die Verkehrssicherheit durch die Erkennung riskanter Verhaltensweisen wie Müdigkeit, Ablenkung oder Telefonieren zu verbessern. Transformer-Modelle haben sich in letzter Zeit als vielversprechende Lösung für diese videobasierte Aufgabe herausgestellt, da sie komplexe räumlich-zeitliche Muster modellieren können. Ihre Einführung in sicherheitskritische Bereiche wie die Fahrer*innenüberwachung wird jedoch durch zwei wesentliche Einschränkungen behindert: (1) eine schlechte Interpretierbarkeit, die es erschwert, die Entscheidungsfindung des Modells zu verstehen, Transparenz zu gewährleisten und potenzielle Verzerrungen im Ergebnis zu identifizieren, (2) sowie lange Rechenzeiten, die praktische Herausforderungen für den Echtzeit-Einsatz in ressourcenbeschränkten Umgebungen mit sich bringen. In dieser Arbeit befassen wir uns mit diesen beiden Herausforderungen, indem wir bestehende Interpretierbarkeitstechniken anwenden und einen Layer-Pruning-Ansatz einführen, der sich an der Wichtigkeit der Attention Heads orientiert. Unsere qualitative Analyse der Attention Heads über alle Netzwerk-Schichten hinweg zeigt, wie Transformer räumlich-zeitliche Merkmale schrittweise kodieren, wobei Heads in tieferen Schichten sich auf Merkmale spezialisieren, die für das Fahrverhalten relevant sind. Diese Analyse unterstreicht die Wirksamkeit der angewandten Metriken zur Erkennung der Bedeutung der Heads bei der Identifizierung der entscheidenden Attention Heads und gibt Aufschluss über die wichtigsten visuellen Hinweise, welche die Vorhersagen des Modells leiten. Quantitative experimentelle Ergebnisse zeigen, dass unsere vorgeschlagene Pruning-Technik eine erhebliche Reduzierung der Rechenzeit bei minimaler Leistungseinbuße durch das Entfernen von Schichten mit geringer Relevanz erzielt. Konkret erreichen wir auf unserem DriverActionInsight (DAI)-Datensatz eine FLOPs-Einsparung von 23,5% bei der Komprimierung von Video Swin mit einer Verringerung der Top-1-Genauigkeit von weniger als 1%. Diese Ergebnisse zeigen das Potenzial unseres Ansatzes auf, die Interpretierbarkeit und Effizienz von Video-Transformer-Modellen zu erhöhen und dadurch ihren Echtzeiteinsatz in Fahrer*innenassistenzsystemen zu ermöglichen.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

Driver action recognition is a critical component of intelligent in-cabin monitoring systems aimed at enhancing road safety by detecting risky behaviors such as drowsiness, distraction, or phone use. For this video-based task, transformer models have recently emerged as a promising solution, thanks to their ability to model complex spatio-temporal patterns. However, their adoption in safety-critical domains like driver monitoring is hindered by two major limitations: (1) poor interpretability, which makes it difficult to understand model decision-making, ensure transparency, and identify potential biases; and (2) high computational cost, which poses practical challenges for real-time deployment in resource-constrained environments. In this work, we address both challenges by building upon existing interpretability techniques and introducing a novel layer-pruning approach guided by attention head importance. Our qualitative analysis of attention heads across layers reveals how transformers progressively encode spatio-temporal features, with deeper-layer heads specializing in features relevant to driver behaviors. This analysis underscores the effectiveness of the applied head importance metrics in pinpointing the crucial attention heads, shedding light on the key visual cues that guide the model's predictions. Quantitative experimental results demonstrate that our proposed pruning technique achieves substantial reductions in computational costs with minimal performance degradation when removing low-relevance layers. Specifically, on our DriverActionInsight (DAI) dataset, we achieve a 23.5% FLOPs saving in compressing Video Swin with less than a 1% decrease in Top-1 accuracy. These findings highlight the potential of our approach to make video transformers more interpretable and efficient, facilitating real-time deployment in driver assistance systems.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Kurzfassung	xi
Abstract	xiii
Contents	xv
1 Introduction	1
1.1 Motivation and Problem Statement	1
1.2 Aim and Contributions of the Work	2
1.3 Structure	3
2 Background	5
2.1 Vision Transformers	5
2.2 Video Transformers	8
3 Related Work	15
3.1 Driver Action Recognition	15
3.2 Vision Transformers for Action Recognition	16
3.3 Interpretability of Transformers	17
3.4 Pruning for Neural Networks	17
3.5 Datasets	18
4 Methods	23
4.1 Attention Maps Computation	23
4.2 Head Importance Metrics	29
4.3 Layer Pruning	33
5 Experiments and Results	35
5.1 Training	35
5.2 Quantitative Results	35
5.3 Qualitative Results	41
6 Conclusions	55
6.1 Discussion	55
	xv

6.2 Future work	56
Overview of Generative AI Tools Used	57
List of Figures	59
List of Tables	63
Acronyms	65
Bibliography	67

Introduction

1.1 Motivation and Problem Statement

Video-based action recognition is a crucial task in computer vision with widespread applications, including surveillance, human-computer interaction, and in-cabin driver action recognition within assisted driving systems to enhance road safety. In the context of driver action recognition, understanding and predicting driver behavior is essential for developing intelligent systems that can detect and respond to potentially distracting actions, like phone use or drowsiness, thereby enhancing road safety. To achieve accurate action recognition, we leverage transformers, a paradigm-shifting approach in artificial intelligence. Transformers have demonstrated remarkable success across various domains, from Natural Language Processing (NLP) to computer vision tasks such as image classification [DBK⁺21], object detection [CMS⁺20, AA23], and action recognition [ADH⁺21, BWT21, FXM⁺21]. Recently, they have become a powerful alternative to traditional CNN-based architectures [TBF⁺15, CZ17], particularly gaining attention in video-based action recognition due to their effective handling of spatio-temporal dependencies [BWT21, ADH⁺21, LNC⁺21].

Despite transformers' success, interpretability of vision transformers remains limited, particularly in video applications where the temporal dimension introduces additional complexity. While interpretability tools exist for NLP transformers [DWB20, LGB⁺19], such as BERT [DCLT19], similar methods for vision transformers are scarce and typically focus on single-image analysis, missing out on temporal aspects critical to action recognition [CGW20, LWD⁺23]. This creates significant challenges in interpreting model predictions for critical tasks such as driver action recognition, where understanding the model's focus can be essential for safety applications. Additionally, transformers face notable limitations, including substantial model sizes and high computational demands. Video transformers, in particular, require even greater resources due to the temporal

dimension, making real-time application and deployment in resource-intensive scenarios, like in-cabin driver monitoring systems, more complex.

To address these challenges, we adapt interpretability techniques based on head importance metrics to gain deeper insights into the self-attention mechanisms of state-of-the-art video transformers. We apply these techniques to two datasets: a general action dataset and a domain-specific dataset for in-cabin driver action recognition, where understanding driver behavior is key for enhancing assisted driving systems. Our experiments reveal two major advantages of this approach: First, by identifying the importance of individual heads, we can highlight specific regions, movements, or moments in time that relevant heads focus on, offering valuable insights into the visual cues most critical for predicting driver behavior. Second, we introduce a layer pruning strategy informed by these importance metrics, selectively removing less critical layers to improve model efficiency and reduce computational load, ultimately making video transformers more practical for real-time applications.

1.2 Aim and Contributions of the Work

The aim of this thesis is twofold. First, we aim to investigate the self-attention mechanism of state-of-the-art video transformer models, examining what these models perceive and prioritize when recognizing actions in videos. This involves analyzing attention head behavior, identifying attention patterns, and determining head importance to pinpoint the key image regions, movements, or time frames that significantly influence action prediction.

Second, the thesis focuses on enhancing the efficiency of video transformers for action recognition by reducing their complexity and resource usage without sacrificing accuracy. We adapt and evaluate existing importance metrics for transformers in the context of video data, and propose a layer pruning method based on these relevance scores to optimize performance.

1.2.1 Research Questions

Therefore, this thesis will address the following research questions:

- What insights can be gained from the self-attention modules of these transformer models in understanding how they recognize driver actions in videos?
 - How do various attention heads within these models exhibit different attention patterns?
 - What is the importance of individual heads?
 - Which layers in the transformer are the most and least relevant?
- Which image areas, movements, or moments in time play the most significant role for accurate driver action prediction?

- To what extent can our proposed layer pruning method lead to improvements in model performance for driver action recognition tasks?

1.2.2 Contributions

The main contributions of this work are the following:

- We perform a head-level analysis to identify specific visual cues encoded by influential heads, offering insights into the decision-making process for driver action recognition.
- We introduce a layer pruning technique based on the applied importance metrics to reduce computational complexity of state-of-the-art transformer models while maintaining high performance.
- We conduct extensive experiments across two different datasets and multiple architectures, and demonstrate the effectiveness of the importance metrics by using layer pruning strategies.

1.3 Structure

The structure of this thesis is as follows:

Chapter 2. This chapter provides the technical foundation for the thesis, detailing the architecture of video transformers and the theory behind attention mechanisms.

Chapter 3. Once a preliminary background has been presented, this chapter focuses on the previous work that is directly relevant to our thesis. Specifically, we first summarize related research findings in the domain of driver action recognition, and vision transformers applied to action recognition tasks. We then present an overview of existing interpretability and pruning techniques specific to these architectures. Finally, we provide an overview of the available datasets that will be used for evaluating these methods.

Chapter 4. The method chapter presents our approach for computing attention maps on video transformer models, details the importance metrics we build upon, and introduces our novel pruning method alongside the procedure for assessing model interpretability.

Chapter 5. In this chapter, we report the experiments conducted in this study. We begin by providing an overview of the training framework and parameters used. The experimental results are divided into two parts. The first part focuses on the quantitative analysis of the pruning method and the ablation study on importance metrics. The second part presents the interpretability analysis, highlighting the key insights gained from attention maps and the generalization of attention heads across datasets.

Chapter 6. Finally, we conclude by discussing the main findings derived from the experiments and open challenges for future work that could potentially improve the current results.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Background

In this chapter, we discuss the foundational concepts and technologies relevant to this thesis. We begin by introducing Vision Transformers (ViTs), the foundational model from which video transformers are derived. We then explain how video transformers adapt the principles of ViTs to handle spatiotemporal data and compare the three key architectures used in this research (Time-Space Transformer (TimeSformer), Video Swin, and Multiscale Vision Transformer (MViT)) highlighting their differences both from ViT and from one another.

2.1 Vision Transformers

Convolution-based backbone architectures have served as the primary backbones for computer vision tasks, including image recognition [HZRS16], object detection [RHGS17], and video classification [TBF⁺15]. However, the introduction of the ViT [DBK⁺21] has marked a significant turning point in the field. Unlike traditional architectures, ViT uses a transformer-based model architecture, originally designed for Natural Language Processing (NLP) tasks. This architecture outperformed existing state-of-the-art Convolutional Neural Networks (CNNs), and marked the beginning of a shift in backbone architectures from CNNs to transformers.

The core idea behind ViT is to treat an image as a sequence of patches, similar to how a sentence is treated as a sequence of words in NLP. This sequence of patches is then fed into a transformer model, which uses self-attention mechanisms to capture long-range dependencies between different parts of the image. The self-attention module, located in each layer of the transformer encoder, functions by assigning weights to various patches according to their relevance to one another, as explained below. The process is illustrated in Fig. 2.1. The key components of ViT are:

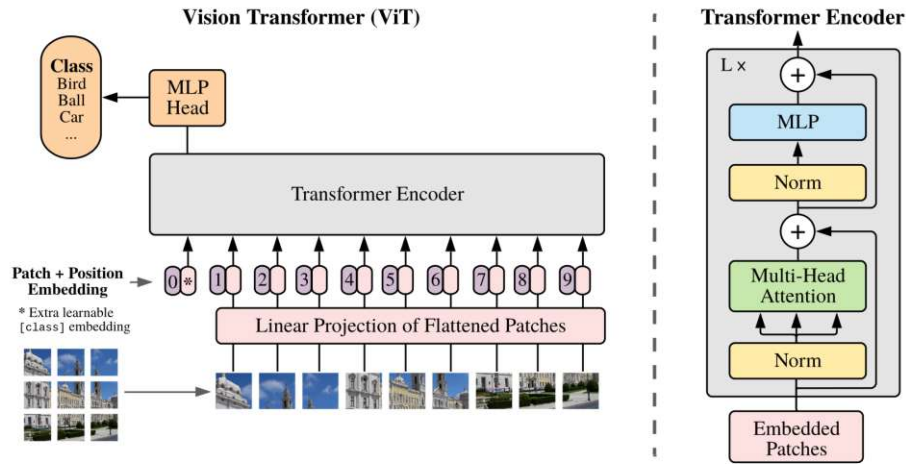


Figure 2.1: ViT model overview. Figure taken from [DBK⁺21].

1. **Image Patching:** The input image is divided into non-overlapping patches (e.g., 16×16 pixels). Assuming that the input image size is $H \times W$, and $P \times P$ is the patch size, the number of patches will be $N = HW/P^2$. Each patch is treated as a token and is flattened into a one-dimensional vector.
2. **Token Embeddings:** These patch vectors are mapped to D dimensions with a trainable linear projection to obtain the patch embeddings, which serve as a concise representation of their respective image patches. Additionally, inspired by BERT’s architecture [DCLT19], a learnable *[class]* embedding (commonly referred to as the CLS token) is concatenated to the patch embeddings, resulting in a $(1 + N) \times D$ matrix. The *[class]* (or CLS) token is designed to aggregate class-related features, which are used to generate the final class probability. Finally, position embeddings are added to the patch embeddings to preserve positional information.
3. **Transformer Encoder:** The transformer encoder consists of L stacked attention layers, as indicated by the “ $L \times$ ” notation within the transformer encoder architecture shown in Fig. 2.1, signifying that there are L layers (or blocks) with an identical structure. Each layer incorporates a multi-head self-attention (MHSA) mechanism, which is the key component in transformer-based architectures.

MHSA mechanism. This mechanism operates by firstly splitting the input embeddings into n different heads and performing the self-attention process simultaneously, allowing the model to focus on different parts of the input [VSP⁺17]. Within each head, the input embeddings are linearly projected onto three learnable weight matrices to obtain: query (Q), key (K), and value (V). Then, the self-attention mechanism calculates a score for each pair of elements in the input sequence by performing the dot product for every possible pair of queries and keys, which is equivalent to the matrix multiplication below (Eq. (2.1)):

$$\text{Unnormalized scores} = QK^T \quad (2.1)$$

The raw attention value from token i to j reflects the similarity between the query Q_i and key K_j . A higher score suggests that the model should pay more attention to the key vector K_j corresponding to the current query Q_i , suggesting that the token represented by the key is relevant for processing the query.

The scores are generally scaled by dividing them by the square root of the Key vector dimension (d_k), which is equivalent to the original input embeddings dimension D divided by the number of heads n . This scaling helps to prevent excessively high values that might lead to saturation in the subsequent softmax function. Once scaled, the scores are fed into the softmax function, transforming them into probabilities (see Eq. (2.2)), thereby highlighting their relative differences. This process converts the unnormalized scores into attention weights, which are organized into an attention matrix A of size $N \times N$, where N represents the number of tokens. Each entry a_{ij} in this matrix corresponds to the attention weight from the i^{th} token to the j^{th} token. The attention matrix A is crucial for interpretability purposes, as we can gain insights into how the model attends to different parts of the input by representing the attention weights visually. These visual representations are known as attention maps.

$$A = \text{softmax}(QK^T / \sqrt{D/n}) \quad (2.2)$$

The attention weights are then used to compute a weighted sum of the value vectors V . This results in a new updated representation H for each token in the sequence, which captures contextual information based on the relationships between tokens (see Eq. (2.3)). Finally, the updated representations from all heads $h = \{1, \dots, n\}$ are concatenated and linearly transformed with new learnable parameters W_o and b_o , to obtain a combined representation denoted as z , as shown in Eq. (2.4).

$$H = A \cdot V = \text{softmax}(QK^T / \sqrt{D/n}) \cdot V \quad (2.3)$$

$$z = \text{Concat}(H_0, H_1, \dots, H_n) \cdot W_o + b_o \quad (2.4)$$

This process is repeated for L layers, resulting in $L \times n$ heads learning diverse relationships between tokens.

4. **Classification Head:** In this step, the *[class]* embedding is separated from the patch tokens, and transformed into class logits through a Multilayer Perceptron (MLP). Class logits are the raw output scores produced by a model, specifically indicating the model's confidence about a specific class before normalization. These logits can be interpreted as unnormalized likelihoods for each class, indicating how strongly the model associates the input with each potential category.

2.2 Video Transformers

Video transformers adapt the principles of ViTs by applying self-attention across both spatial and temporal dimensions. Instead of treating an image as a sequence of patches, video transformers treat a video as a sequence of frames, with each frame divided into patches. The key challenge is to efficiently capture both intra-frame spatial dependencies and inter-frame temporal dependencies. To achieve this, video transformers typically modify the attention mechanism to handle the temporal aspect of video data. Below, we discuss how three prominent video transformer architectures (TimeSformer, Video Swin, and MViT) design their attention mechanisms for video data. Table 2.1, located at the end of this chapter, provides an overview of the attention matrices, token sizes, and output dimensions of the MHSA layer for these architectures.

TimeSformer

TimeSformer was one of the first models to adapt the ViT architecture for video action recognition. This architecture decouples spatial and temporal attention in its MHSA. Unlike Video Swin and MViT, which use 3D patches, TimeSformer operates with 2D patches (tokens) of size $P \times P$ for each frame. Each patch is projected into an embedding $z_{(p,t)} \in \mathbb{R}^D$, where $p = \{1, \dots, N\}$ denote spatial locations, and $t = \{1, \dots, T\}$ represent temporal indices.

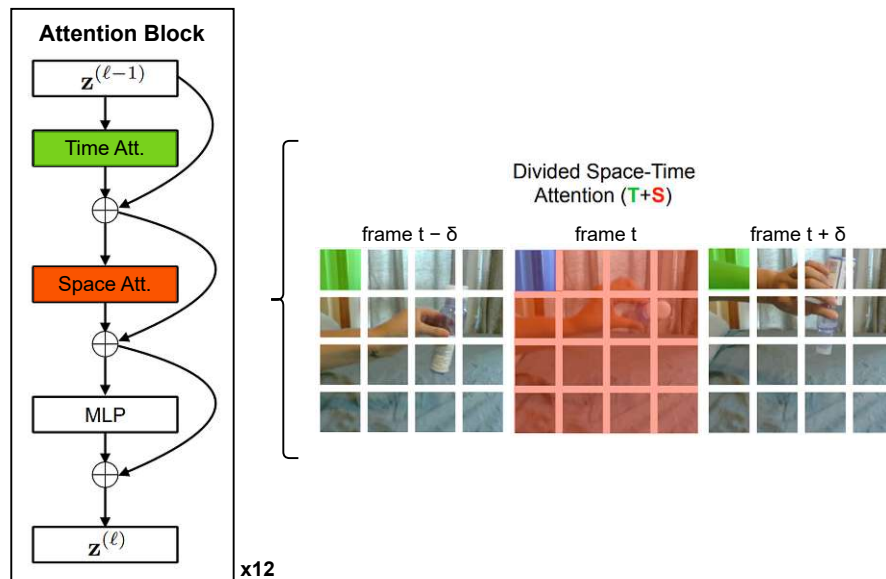


Figure 2.2: Visualization of the TimeSformer attention mechanism. On the right, a representation of the attention mechanism in TimeSformer is depicted, illustrating that this block is repeated 12 times (the model has 12 equivalent layers). On the left, the attention visualization is applied to a video clip. Figure adapted from [BWT21].

TimeSformer consists of 12 layers, where in each layer, temporal and spatial attention

are applied separately, each with its own set of learnable parameters. First, temporal attention compares patches at the same spatial location across frames $t = \{1, \dots, T\}$, producing a temporal attention matrix of size $T \times T$ for each patch p . Afterwards, spatial attention is applied within each frame independently (treated as a standalone image), comparing all patches (plus the *[class]* token) at a fixed time step t . This results in a $(1 + N) \times (1 + N)$ attention matrix. Fig. 2.2 illustrates how the attention mechanism in TimeSformer is applied to video frames.

Video Swin Transformer

The Video Swin Transformer [LNC⁺21] builds upon the Swin Transformer architecture [LLC⁺21], extending its capabilities to the spatio-temporal domain. Similar to ViT [DBK⁺21], the input video is initially decomposed into N non-overlapping 3D patches, each mapped to an embedding vector of dimension D using a 3D convolution. Video Swin employs a hierarchical structure, consisting of four stages (or blocks), each with increasing spatial-temporal abstraction. Initially, the input video is converted to very small 3D tokens (e.g. tokens of size $2 \times 4 \times 4$) for high-resolution spatial processing. After each block, a pooling or merging layer combines adjacent patches, reducing their spatial dimensions while doubling the feature channels.

In each block of Video Swin, multiple layers are stacked, and each of these layers incorporates the MHSA mechanism. Unlike global self-attention that spans the entire input, the MHSA in Video Swin focuses on local windows to compute attention. Specifically, attention is computed within non-overlapping windows of size $W_t = 8 \times 7 \times 7$ tokens, resulting in an attention matrix of size $W_t \times W_t$. This attention mechanism compares how tokens within a window attend to each other. As the model progresses into deeper layers, the token size increases, and attention computation transitions from highly localized areas to broader regions. This hierarchical arrangement allows the model to capture features at different scales, enhancing its ability to represent both fine and coarse-grained details in the video data.

Fig. 2.3 illustrates the token and attention window sizes across the four stages. In Stage 1, due to the small token size of $2 \times 4 \times 4$ pixels, 64 windows are required to cover the entire video sequence ($16 \times 224 \times 224$). Each window, consisting of $8 \times 7 \times 7$ tokens, corresponds to an area of $16 \times 28 \times 28$ pixels, and when arranged in a $1 \times 8 \times 8$ grid spatially, covers the full video resolution of $16 \times 224 \times 224$. As the network deepens, the number of windows decreases while token size grows.

Notably, Video Swin omits the use of a *[class]* token. Instead, in the classification head, it performs global average pooling across all token feature vectors, consolidating all information into a single vector. This resulting vector is then passed into the MLP.

Additionally, the original paper [LNC⁺21] introduces the concept of 3D shifted windows. Since the MHSA operates within non-overlapping 3D windows, this design limits connections between different windows. To overcome this limitation, the authors propose shifting the window configuration along the temporal, height, and width axes in odd

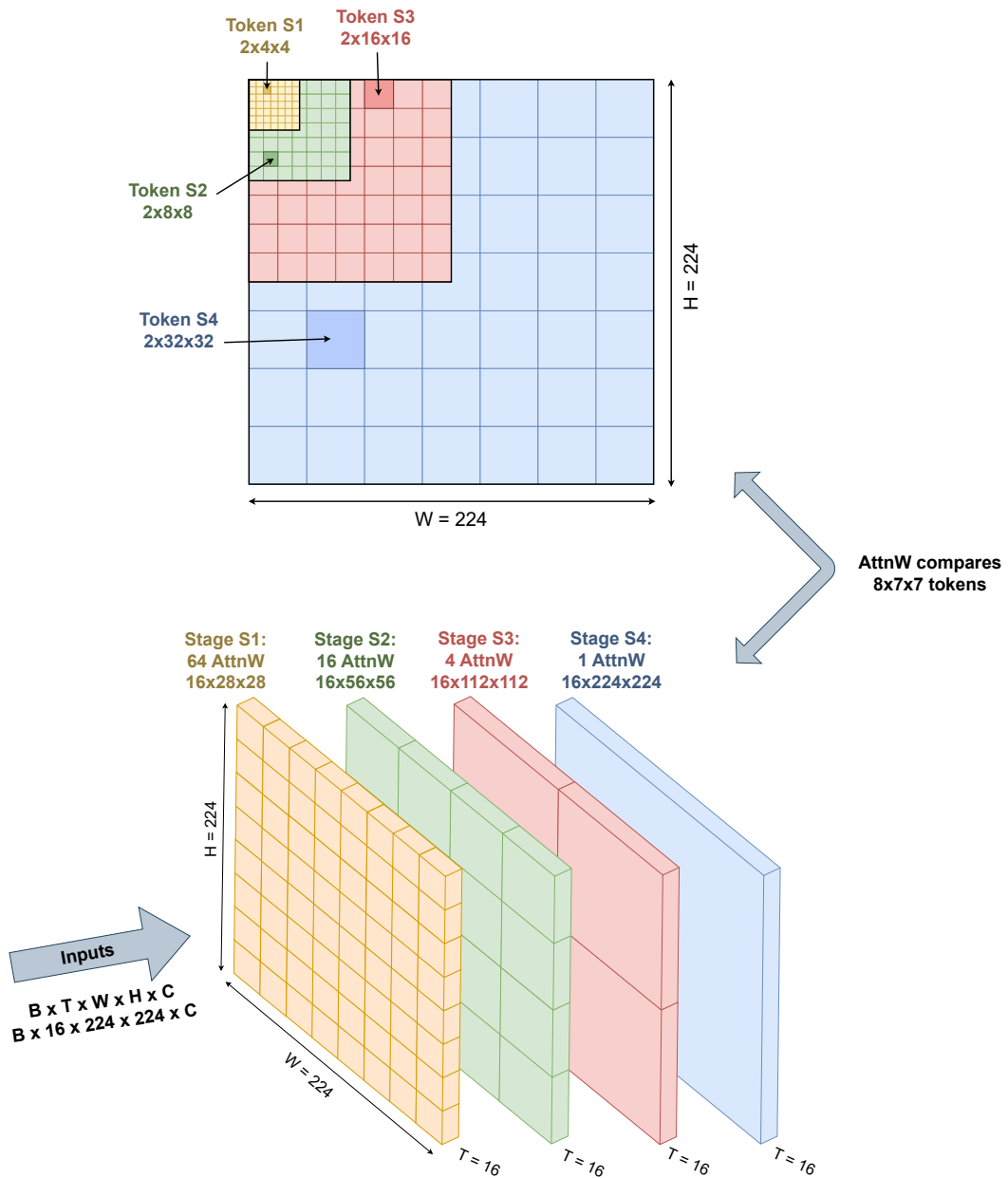


Figure 2.3: Illustration of token sizes and attention window (AttnW) sizes across the four stages (S1-S4) of Video Swin. The top shows token sizes at each stage, while the bottom displays attention window sizes, including the number of windows for an input of 16 frames ($T = 16$) at a resolution of 224×224 pixels, with batch size B and channel dimension C . For example, in Stage 2 (S2), the token size is $2 \times 8 \times 8$ pixels, resulting in a window size of $16 \times 56 \times 56$ pixels (since the window size W_t is consistently $8 \times 7 \times 7$ tokens). By Stage 4, the token size reaches $2 \times 32 \times 32$ pixels, with the attention window covering the entire video sequence ($16 \times 224 \times 224$ pixels), allowing for global attention computation.

layers, allowing the model to capture additional contextual information. However, our experiments do not show a significant performance improvement when applying this shifting mechanism. Therefore, we opt to apply the same window configuration across all MHSA blocks. This decision simplifies model understanding and also aids interpretability, as it allows for a clearer mapping of attention weights to their corresponding spatial and temporal positions in the input video.

MViT

Like Video Swin, MViT uses a multi-scale approach, computing attention across various resolutions through pooling techniques. Early stages in MViT handle high-resolution tokens, while later stages aggregate information from larger patches or regions. However, unlike Video Swin, which applies pooling layers between stages, MViT integrates a pooling attention mechanism that uses strided convolutions to reduce the dimensionality of key, value, and query vectors.

This pooling attention mechanism in MViT has the following implications:

- Query Pooling:** The query tensor Q is pooled to reduce resolution between different stages of the model. As shown in Fig. 2.4, the resolution of the output from the pooling attention mechanism (matrix H) is determined entirely by the pooling of Q , producing a new resolution of $t' \times h' \times w'$, calculated as $\frac{t}{s_t^q} \times \frac{h}{s_h^q} \times \frac{w}{s_w^q}$. In the default configuration, the stride along the temporal dimension s_t^q remains fixed at 1 at all times, while the stride in the spatial dimensions increases to 2 only in the first layer of each stage (except the first stage). In the subsequent layers of each stage, the stride is kept at $S_Q = (1, 1, 1)$. This means that spatial resolution is halved only at the start of each stage and remains constant afterward. The corresponding output sizes are detailed in Table 2.1b, showing that within a block, the output dimensions remain consistent, while at the start of each stage, the spatial dimensions are halved, and the channel dimension is doubled.
- Key and Value Pooling:** Pooling the keys (K) and values (V) reduces the number of tokens the model attends to. As illustrated in Fig. 2.4, self-attention is computed on $t' \times h' \times w'$ tokens by attending to $t'' \times h'' \times w''$ tokens. By pooling K and V , the number of tokens decreases from $t \times h \times w$ to $t'' \times h'' \times w''$, where $t'' \times h'' \times w''$ is typically much smaller (e.g., $8 \times 7 \times 7$), resulting in smaller attention maps and reduced computational complexity.
- Attention Matrix:** As a result of pooling K and Q , the attention matrix A is no longer squared. While this pooling significantly reduces the spatial or temporal resolution of the K and V vectors, potentially causing a loss of fine-grained details, the query (Q) retains a higher resolution, gradually decreasing until K , Q , and V reach the same resolution (e.g., $8 \times 7 \times 7$). The key idea behind this approach is that lower-resolution tokens (in K) attend to higher-resolution tokens (in Q), and vice versa. For classification, the attention weights used for prediction reflect

2. BACKGROUND

the attention of the *[class]* token toward the lower-resolution tokens, rather than high-resolution ones, to minimize the number of tokens processed ($t'' \times h'' \times w''$ tokens).

- **Output:** When the attention matrix is multiplied by the value vectors, the output represents the weighted sum of the values V associated with the keys that the queries attend to. This means that the information passed to the next layer integrates features from different scales, as V is scaled according to how high-resolution tokens focus on low-resolution tokens.

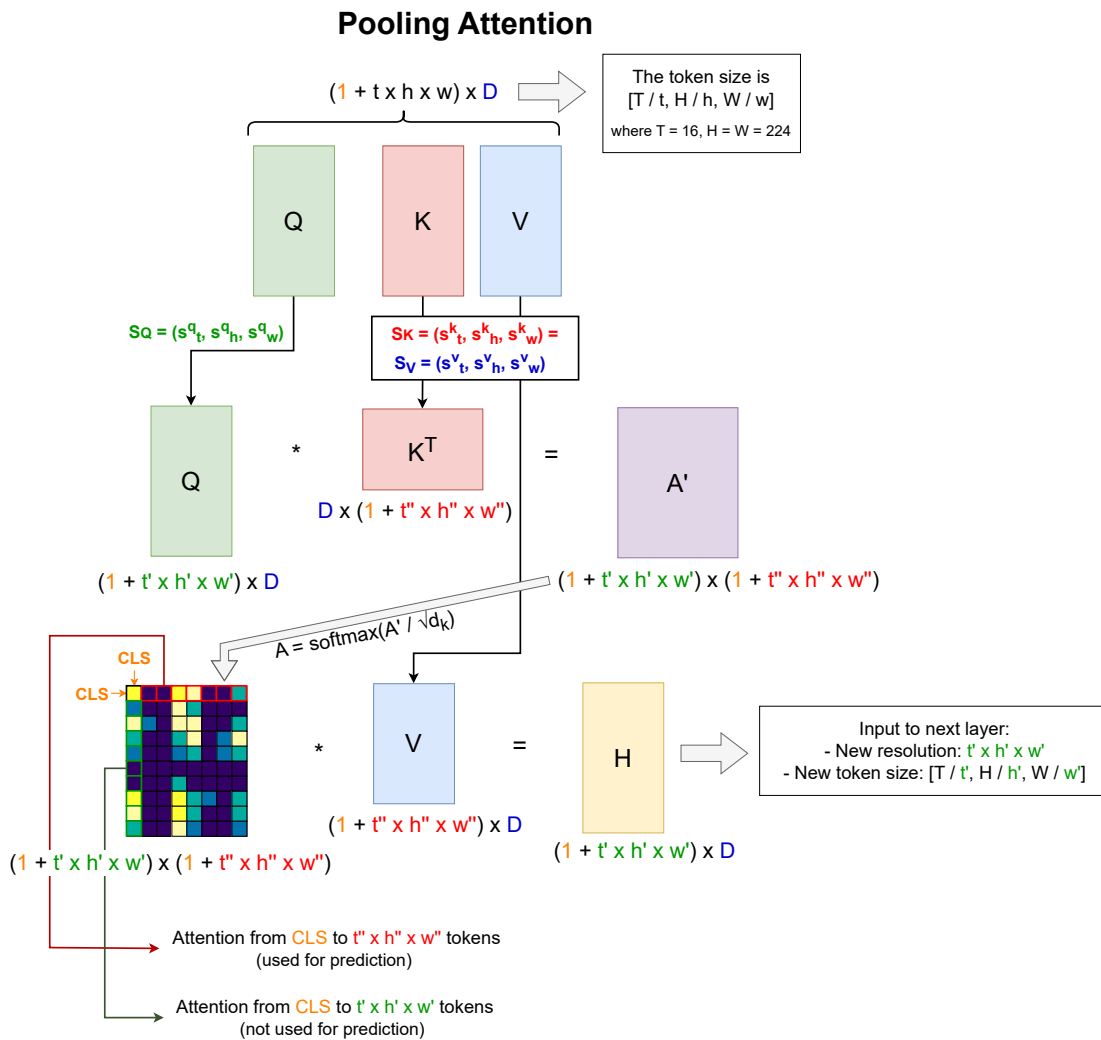


Figure 2.4: Overview of the pooling attention mechanism in MViT within a general layer. This process, which includes pooling and self-attention computation, is consistent across all layers of the architecture. The main differences lie in the strides applied to the key, query, and value vectors (S_K , S_Q , and S_V), as well as the input resolution $t \times h \times w$, which are progressively reduced as the model depth increases.

Stage	Operators	Output dimensions
patch ₁	$1 \times 16 \times 16$, 768 stride $1 \times 16 \times 16$	$(1+16 \times 14 \times 14) \times 768$
block ₁	$\left[\begin{array}{l} \text{MHSA-T } 16 \times 16 \\ \text{MHSA-S } (1+14 \times 14) \times (1+14 \times 14) \end{array} \right] \times 12$	$(1+16 \times 14 \times 14) \times 768$

(a) TimeSformer

Stage	Operators	Output dimensions
patch ₁	$3 \times 7 \times 7$, 96 stride $2 \times 4 \times 4$	$(1+8 \times 56 \times 56) \times 96$
block ₁	$\left[\begin{array}{l} \text{MHSA } (1+8 \times 56 \times 56) \times (1+8 \times 7 \times 7) \\ \text{token sizes } (2 \times 4 \times 4, 2 \times 32 \times 32) \end{array} \right] \times 1$	$(1+8 \times 56 \times 56) \times 96$
block ₂	$\left[\begin{array}{l} \text{MHSA } (1+8 \times 28 \times 28) \times (1+8 \times 14 \times 14) \\ \text{token sizes } (2 \times 8 \times 8, 2 \times 16 \times 16) \end{array} \right] \times 1$	$(1+8 \times 28 \times 28) \times 192$
	$\left[\begin{array}{l} \text{MHSA } (1+8 \times 28 \times 28) \times (1+8 \times 7 \times 7) \\ \text{token sizes } (2 \times 8 \times 8, 2 \times 32 \times 32) \end{array} \right] \times 1$	
block ₃	$\left[\begin{array}{l} \text{MHSA } (1+8 \times 14 \times 14) \times (1+8 \times 14 \times 14) \\ \text{token sizes } (2 \times 16 \times 16, 2 \times 16 \times 16) \end{array} \right] \times 1$	$(1+8 \times 14 \times 14) \times 384$
	$\left[\begin{array}{l} \text{MHSA } (1+8 \times 14 \times 14) \times (1+8 \times 7 \times 7) \\ \text{token sizes } (2 \times 16 \times 16, 2 \times 32 \times 32) \end{array} \right] \times 10$	
block ₄	$\left[\begin{array}{l} \text{MHSA } (1+8 \times 7 \times 7) \times (1+8 \times 14 \times 14) \\ \text{token sizes } (2 \times 32 \times 32, 2 \times 16 \times 16) \end{array} \right] \times 1$	$(1+8 \times 7 \times 7) \times 768$
	$\left[\begin{array}{l} \text{MHSA } (1+8 \times 7 \times 7) \times (1+8 \times 7 \times 7) \\ \text{token sizes } (2 \times 32 \times 32, 2 \times 32 \times 32) \end{array} \right] \times 1$	

(b) MViT

Stage	Operators	Output dimensions
patch ₁	$2 \times 4 \times 4$, 96 stride $2 \times 4 \times 4$	$8 \times (56 \times 56) \times 96$
block ₁	$\left[\begin{array}{l} \text{MHSA } (8 \times 7 \times 7) \times (8 \times 7 \times 7) \\ \text{token size } 2 \times 4 \times 4 \end{array} \right] \times 2$	$8 \times (56 \times 56) \times 96$
block ₂	$\left[\begin{array}{l} \text{MHSA } (8 \times 7 \times 7) \times (8 \times 7 \times 7) \\ \text{token size } 2 \times 8 \times 8 \end{array} \right] \times 2$	$8 \times (28 \times 28) \times 192$
block ₃	$\left[\begin{array}{l} \text{MHSA } (8 \times 7 \times 7) \times (8 \times 7 \times 7) \\ \text{token size } 2 \times 16 \times 16 \end{array} \right] \times 18$	$8 \times (14 \times 14) \times 384$
block ₄	$\left[\begin{array}{l} \text{MHSA } (8 \times 7 \times 7) \times (8 \times 7 \times 7) \\ \text{token size } 2 \times 32 \times 32 \end{array} \right] \times 2$	$8 \times (7 \times 7) \times 768$

(c) Video Swin Transformer

Table 2.1: Overview of model architectures. For each model – (a) TimeSformer, (b) Video Swin, and (c) MViT– we detail the different stages (or blocks). Each subtable presents information on the number of layers using the MHSA mechanism, the token sizes at each layer, the dimensions of the attention matrix, and the output dimensions of the MHSA for that block.

Related Work

In this section, we first summarize related research findings in the application domain of driver action recognition, followed by relevant works on action recognition using vision transformers. We then present an overview of existing interpretability and pruning techniques specific to these architectures, setting the stage for our novel contributions in adapting and enhancing these methods for improved efficiency and understanding. Lastly, we introduce the datasets used in this thesis.

3.1 Driver Action Recognition

Driver action recognition has been an active area of research in recent years due to its importance for improving road safety. In early years, the dominant approach involved using 2D-CNNs and 3D-CNNs to classify driver actions [BGT18, XTL⁺18, XLW⁺19], which were widely adopted for their ability to extract robust spatial features from visual data. However, recent advancements have introduced a variety of innovative methods.

Recognizing the importance of temporal dynamics in driver actions, researchers have integrated Recurrent Neural Networks (RNNs) into their CNN-based models to capture this temporal aspect. For instance, Kidu *et al.* [KSS⁺24] presented a Long-term Recurrent Convolutional Network (LRCN) that combines the spatial feature extraction capabilities of CNNs with the temporal sequence learning of LSTMs. This approach demonstrated superior performance in recognizing complex driver actions.

Beyond traditional CNN-RNN architectures, other studies have explored the use of human pose estimation to improve driver action recognition. These approaches leverage keypoints (e.g., hand, arm, head positions) to provide a detailed representation of the driver's body movements. Graph Convolutional Networks (GCNs) have been particularly effective in this regard, as they can model the relationships between different body joints to improve recognition accuracy [LLZ⁺19]. These pose-based methods have proven useful

for identifying actions like reaching for objects or turning the steering wheel, which are not easily discernible from raw image data alone.

Most recently, transformer-based architectures have entered the scene, showcasing their effectiveness in handling complex driver action recognition tasks. These models have gained attention for their ability to handle long-range dependencies and capture both spatial and temporal features more effectively than traditional CNNs or RNNs. Recent works, such as [ZQJM23, LZZ⁺22], have demonstrated that transformers can outperform previous models by leveraging self-attention mechanisms to focus on relevant parts of the video sequence.

3.2 Vision Transformers for Action Recognition

Convolution-based architectures have served as the primary backbones for computer vision tasks, including image recognition, object detection, and video classification. However, the introduction of the Vision Transformer (ViT)[DBK⁺21] has marked a significant turning point in the field. The ViT processes images by splitting them into non-overlapping patches, treating them as a sequence of tokens, and then using a transformer encoder to learn relationships between these patches, enabling image understanding. This architecture outperformed existing state-of-the-art CNNs, and marked the beginning of a shift in backbone architectures from CNNs to transformers.

In the context of action recognition, recent research has adapted the ViT concept to video analysis by extending the self-attention mechanism, the core element of transformers, from the spatial domain of images to the space-time 3D volume of video sequences. This adaptation has led to models like TimeSFormer [BWT21] and ViViT [ADH⁺21], which explore various strategies for factorizing spatial and temporal dimensions. TimeSFormer implements separate temporal and spatial attention within each block, while ViViT advocates for a *late fusion* approach, adding a temporal attention encoder on top of the spatial encoder. However, these models suffer from considerable computational inefficiency as complexity scales quadratically with the number of tokens.

To address this, recent research has explored hierarchical structures in video transformers, exemplified by models like MViT [FXM⁺21, LWF⁺21] and Video Swin Transformer [LNC⁺21], aiming to represent a video more efficiently. MViT employs pooling techniques to reduce spatial resolution while increasing channel capacity, while Video Swin Transformer, based on its 2D counterpart [LLC⁺21], builds hierarchical feature maps by merging image patches in deeper layers and limits self-attention computations to non-overlapping local windows instead of costly global calculations to improve efficiency.

Despite these advancements, further research is needed to mitigate complexity. In this thesis, we examine TimeSFormer, MViT, and Video Swin Transformer to analyze and demonstrate their efficiency and potential for pruning.

3.3 Interpretability of Transformers

The main component of transformer networks is the self-attention mechanism, and several attention visualization techniques have been explored to interpret it. Interpreting raw attention as relevancy scores is a common practice [CMS⁺20, CTM⁺21], however, averaging these weights across heads or layers can overlook the distinct roles of each. Enhanced techniques include attention rollout [AZ20], which adjusts attention weights based on linear combinations across layers, and Grad-CAM [SCD⁺17], which integrates input features and gradients to identify relevant image regions for a specific target. Recently, [CGW20] introduced a method based on Layer-wise Relevance Propagation (LRP) to calculate and propagate relevancy scores throughout the network. However, while these methods provide valuable insights into overall attention distribution, they still lack analysis at the individual head level.

Conversely, in the realm of NLP, other studies delve into exploring the importance of individual heads within the network [HDWX21, VTM⁺19, LXW⁺21] and examining model behavior by visualizing attention weights and patterns for the most relevant heads, thus narrowing the scope of model analysis. Similarly, as foundational research for our work, [LWD⁺23] delves into the ViT model, investigating the identification of important heads through various pruning-based metrics and subsequently examining different attention patterns and maps by selecting the most relevant heads. However, interpretability for video transformers remains largely unexplored. In this thesis, we adapt the importance metrics from [LWD⁺23] to video transformers and introduce a novel layer pruning method based on these metrics.

3.4 Pruning for Neural Networks

Reducing computational complexity in neural networks has often been achieved through pruning techniques. Recent works have explored structured pruning methods, which selectively remove in an organized manner less critical components from the model while maintaining its overall structure, leading to a more efficient architecture. In CNNs, this often involves the removal of entire filters [YYY⁺19] or channels [LZK⁺21], while in transformers, attention heads are frequently targeted for pruning, given that some heads contribute more to the model's predictions than others. However, layer pruning remains relatively unexplored; a recent survey on deep neural network pruning mentions only one work addressing layer pruning in CNNs, with no studies focusing on video transformers [CZS24].

Research in the realm of NLP has proved that pruning most attention heads in transformers has minimal impact on performance [HDWX21, VTM⁺19, MLN19], with some works even suggesting layers can be reduced to only one head [VTM⁺19], indicating the potential for whole layer pruning. Additionally, as a source of inspiration for our work, Fan *et al.* [FGJ20] proposed structured layer pruning in NLP, demonstrating the feasibility of selecting sub-networks of a trained model during inference time with limited

impact on performance.

Additionally, as a source of inspiration for our work, Fan *et al.* [FGJ20] proposed structured layer pruning, which selects sub-structures of a model during inference time. They demonstrated the feasibility of selecting sub-networks of any depth from one large network without the need for fine-tuning and with limited impact on performance. Nevertheless, this work is primarily focused on the NLP domain.

Furthermore, Yu *et al.* [YHW⁺22] explored block pruning for vision transformers, introducing classifiers using intermediate information. They found their approach achieved higher parallelism efficiency than unstructured pruning techniques (e.g. weight-wise pruning on attention matrix), which motivates our research on layer pruning methods.

Unlike prior work, which often treats efficiency and interpretability as separate objectives, our approach leverages importance metrics to address both simultaneously in the context of video transformer pruning. As shown, existing methods typically focus on optimizing performance through architectural compression, with little attention to what these changes reveal about the model’s internal behavior. In contrast, we use importance scores not only to identify and remove less relevant layers, improving computational efficiency, but also to uncover how different layers contribute to spatio-temporal representation learning. To our knowledge, this is the first work to apply layer pruning in video transformers using importance metrics, introducing a dual-purpose framework that jointly targets efficiency and interpretability.

3.5 Datasets

In this subsection, we present the datasets relevant to our work, which focuses on action recognition from video data. Our interest lies particularly in datasets that include annotated action recognition labels, covering both general-purpose and driver-specific domains. While there are many driver-related datasets available, not all are suitable for our purposes. For instance, DriPE [GCJT21] is designed for driver pose estimation and lacks explicit action recognition labels, making it unsuitable for our use case.

Dataset	Driver	Size	Classes	Scenario
Drive&Act [MRH ⁺ 19]	Yes	12H video	34	Acted
DMD [OKCn ⁺ 20]	Yes	41H video	13	Acted
3MDAD [JBKAM20]	Yes	574K frames	16	Naturalistic
AIDE [YHX ⁺ 23]	Yes	522K frames	7	Naturalistic
DAI (Ours)	Yes	1.3K clips	20	Acted
HMDB-51 [KJG ⁺ 11]	No	3.3K clips	51	Naturalistic
UCF101 [SZS12]	No	2.5K clips	101	Naturalistic
NTU RGB+D [SLNW16]	No	56.9K clips	60	Naturalistic
Kinetics-400 [CZ17]	No	306.3K clips	400	Naturalistic

Table 3.1: Overview of relevant datasets for action recognition from video data.

Table 3.1 provides a summary of the datasets considered. Among the driver-focused datasets, we include four well-established benchmarks (Drive&Act, DMD, 3MDAD, and AIDE) as well as our proprietary dataset, DriverActionInsight (DAI). The first two, Drive&Act and DMD, consist of acted scenarios with participants performing scripted driver actions in controlled environments. 3MDAD and AIDE, in contrast, provide naturalistic driver behavior captured in real-world settings, which better reflect spontaneous, in-the-wild activity.

In addition to the driver-focused datasets, we include several general human action recognition datasets (HMDB-51, UCF101, NTU RGB+D, and Kinetics-400) that are widely used benchmarks in the broader computer vision community. These datasets are not specific to driving but contain diverse and richly annotated human actions in naturalistic settings, making them valuable for training or pretraining general-purpose models that can be adapted to the driver context.

After providing a high-level overview, we further describe in more detail the specific datasets used in this work in the following subsections.

Kinetics-400 The Kinetics-400 dataset is a prominent benchmark for video action recognition, consisting of over 300,000 video clips categorized into 400 distinct action classes. These classes range from everyday activities to more specialized actions. The dataset is designed to capture a wide variety of movements and scenarios, making it a rich resource for training models that can generalize across different types of actions. An overview of the Kinetics-400 action classes can be found in Fig. 3.1.

Due to its large size, Kinetics-400 will not be used to evaluate our proposed layer pruning method. However, it serves as the foundation for pre-training all our video transformer models. This pre-training enables the models to learn robust feature representations that generalize well across various tasks. Subsequently, these models can be fine-tuned on smaller, task-specific datasets, resulting in enhanced performance.



Figure 3.1: Sample frames for 4 action classes of Kinetics-400. Figure reproduced from [CZ17].

UCF101 The UCF101 dataset is a widely used benchmark for general human action recognition, containing 13,320 video clips across 101 action classes. These classes span a diverse range of activity types, including Human-Object Interactions (e.g., Apply Lipstick,

3. RELATED WORK

Brushing Teeth), Body-Motion Only actions (e.g., Pull ups, Swing), Human-Human Interactions (e.g., Haircut, Salsa Spin), Playing Musical Instruments (e.g., Playing Cello), and various Sports (e.g., High Jump, Kayaking).

The videos are collected from YouTube and exhibit high variability in terms of camera motion, background clutter, and lighting, making UCF101 a challenging and realistic dataset. In our work, we use UCF101 to assess how well our proposed pruning method generalizes beyond the driver-focused domain. The variety and scale of the dataset allow us to test the robustness and efficiency of our approach on large, unconstrained action recognition tasks.

Sample frames from the UCF101 dataset are shown in Fig. 3.2, illustrating the diversity of actions and environmental contexts covered in the dataset.



Figure 3.2: Sample frames of action classes from the UCF101 dataset.

Drive&Act The Drive&Act dataset is a large-scale benchmark for in-vehicle activity recognition, capturing over 12 hours of multi-modal video data annotated with a hierarchical taxonomy of driver behaviors. In our study, we make use of the infrared modality from the inner mirror view, as it most closely resembles the data available in our own dataset in terms of camera placement and lighting conditions.

While Drive&Act offers a wide range of annotations, including coarse tasks and atomic action units, we focus exclusively on the fine-grained activity level, which contains 34 defined action classes (e.g., fastening seatbelt, drinking, interacting with phone). Samples from the dataset are illustrated in Fig. 3.3. These actions closely resemble those defined in our proprietary DAI dataset, making Drive&Act a strong candidate for evaluating cross-dataset generalization. By aligning the activity label granularity, we ensure that both datasets are comparable in terms of driver behavior categories, allowing us to directly test the transferability of attention specialization without further fine-tuning.

DriverActionInsight (DAI) Our proprietary DriverActionInsight (DAI) dataset comprises 20 driver-specific action classes and was recorded in a controlled environment with 20 subjects and a pre-defined catalog of actions. It contains 1,302 annotated

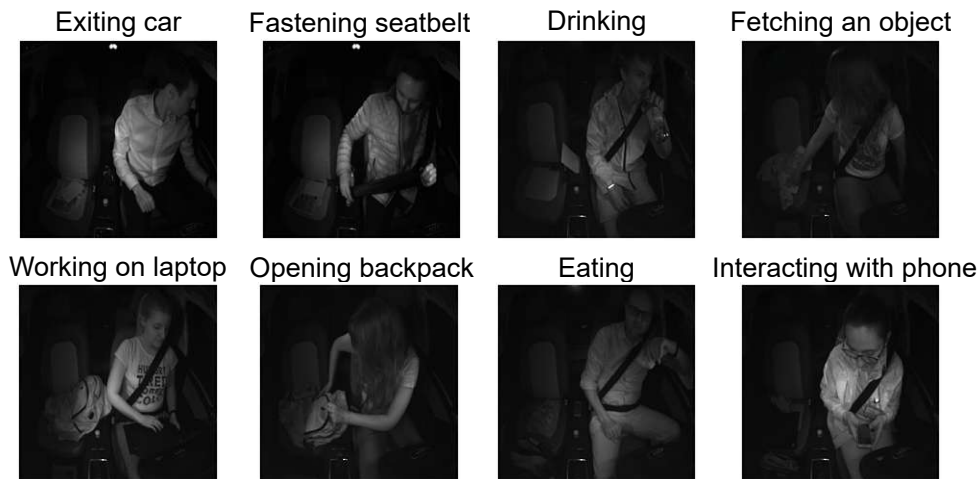


Figure 3.3: Sample frames for 8 action classes of Drive&Act.

sequences recorded with a near-infrared camera mounted near the rear-view mirror. The dataset is partitioned into train, validation, and test subsets with a ratio of 70-15-15, and stratified sampling is applied to ensure that each subject is contained in only one subset. Samples from the dataset are illustrated in Fig. 3.4, where faces are blurred for privacy protection.

This small, application-specific dataset is used to evaluate how models fine-tune to specialized tasks like driver action recognition, to examine whether the learned representations (head attention patterns) differ from those in general action recognition datasets (e.g. UCF101), and to assess the performance of our pruning method on a reduced dataset.



Figure 3.4: Sample frames for 8 action classes of our proprietary DAI dataset.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Methods

This chapter describes the methods developed in this work, covering the key techniques used to achieve the results. First, we present a new approach for computing attention maps in video transformer models, followed by a description of the selected head importance metrics designed to evaluate the relevance of individual attention heads within a video transformer model. Then, we propose a novel layer pruning method implemented to optimize model performance while maintaining accuracy by removing low-relevance layers.

4.1 Attention Maps Computation

In this section, we describe the computation of attention maps for the three video transformers used in our experiments: TimeSformer, Video Swin, and MViT. Each architecture employs a different method for generating attention maps, reflecting the significant variations in their MHSA mechanisms, as discussed in Section 2.2. It is also important to emphasize that the methods presented here are designed for visualizing individual heads of the transformer. We do not average the weights across heads, since each head captures different patterns, and averaging would mask their unique contributions, as noted in Section 3.3. Furthermore, while alternative visualization techniques, such as attention rollout [AZ20], are available, implementing methods that track attention flow through all transformer layers can be challenging, especially for hierarchical models like Video Swin, and even more so for MViT, which uses non-squared attention matrices.

TimeSformer

Visualizing the attention weights in TimeSformer is a straightforward process, as all layers share the same structure, ensuring consistent token sizes and attention matrix dimensions throughout the architecture. This consistency is detailed in Table 2.1a, which illustrates that the attention matrices in the MHSA are identical for each layer. Additionally, as

noted in Section 2.2, the token size remains fixed at $P \times P$, with P typically set to 16. However, since TimeSformer separates the computation of temporal and spatial attention, it is necessary first to create decoupled attention representations for both space and time:

- Temporal attention:** As discussed in Section 2.2, temporal attention in TimeSformer compares patches across time that are located in the same spatial positions. For each spatial position in the grid, which corresponds to a token of size $p \times p$ (with the frame divided into N such tokens), we obtain a temporal attention matrix of size $T \times T$. Since this matrix does not include a `[class]` token, we aggregate the temporal attention weights vertically, resulting in relevance scores for each frame $t = \{1, \dots, T\}$ for a given token. This process is repeated for every token. Frames with higher aggregated values are considered more relevant for that specific token. To convert these token-level scores into pixel-level scores, we interpolate the aggregated weights for each token using a scale factor equal to the patch size (1, 16, 16), as each token represents $1 \times 16 \times 16$ pixels. Finally, we reshape the attention maps to match the original input dimensions of $16 \times 224 \times 224$. To interpret temporal attention, we need to compare the scores of individual pixels (or spatial regions) across frames. As shown in Fig. 4.1 (bottom left), if a specific spatial area within a frame is highlighted, it indicates that this region holds greater importance or relevance for the prediction compared to the same spatial area in other frames.
- Spatial attention:** As discussed in Section 2.2, spatial attention compares tokens (patches) within individual frames, producing T attention matrices, one for each frame, with dimensions $(1 + N) \times (1 + N)$, where N denotes the number of tokens and the `[class]` token is included. To generate spatial attention maps, for each attention matrix, we directly use the attention weights that represent the attention the `[class]` token allocates to the $p \times p$ tokens. Since these scores are initially mapped to tokens, interpolation is required to transition from token space to pixel space. This interpolation is performed using a scale factor equivalent to the token size (16×16), yielding a 224×224 attention map for each frame. Interpreting spatial attention is straightforward – it focuses on each frame independently, as shown in Fig. 4.1 (bottom right). Within each frame, the highlighted areas correspond to regions of greater importance compared to those not highlighted.

To create a combined visualization using both temporal and spatial attention maps, we first select the transformer heads to visualize for each type of attention, which can come from different layers. Once the heads are selected, we apply element-wise multiplication to merge the two attention maps. For each spatial token (e.g., a patch at a specific location) and its corresponding temporal attention scores across frames, the combination is done by multiplying the temporal attention values for that token with the spatial attention values within each frame. This element-wise multiplication maintains the original dimensionality of the attention maps. The resulting combined attention reflects

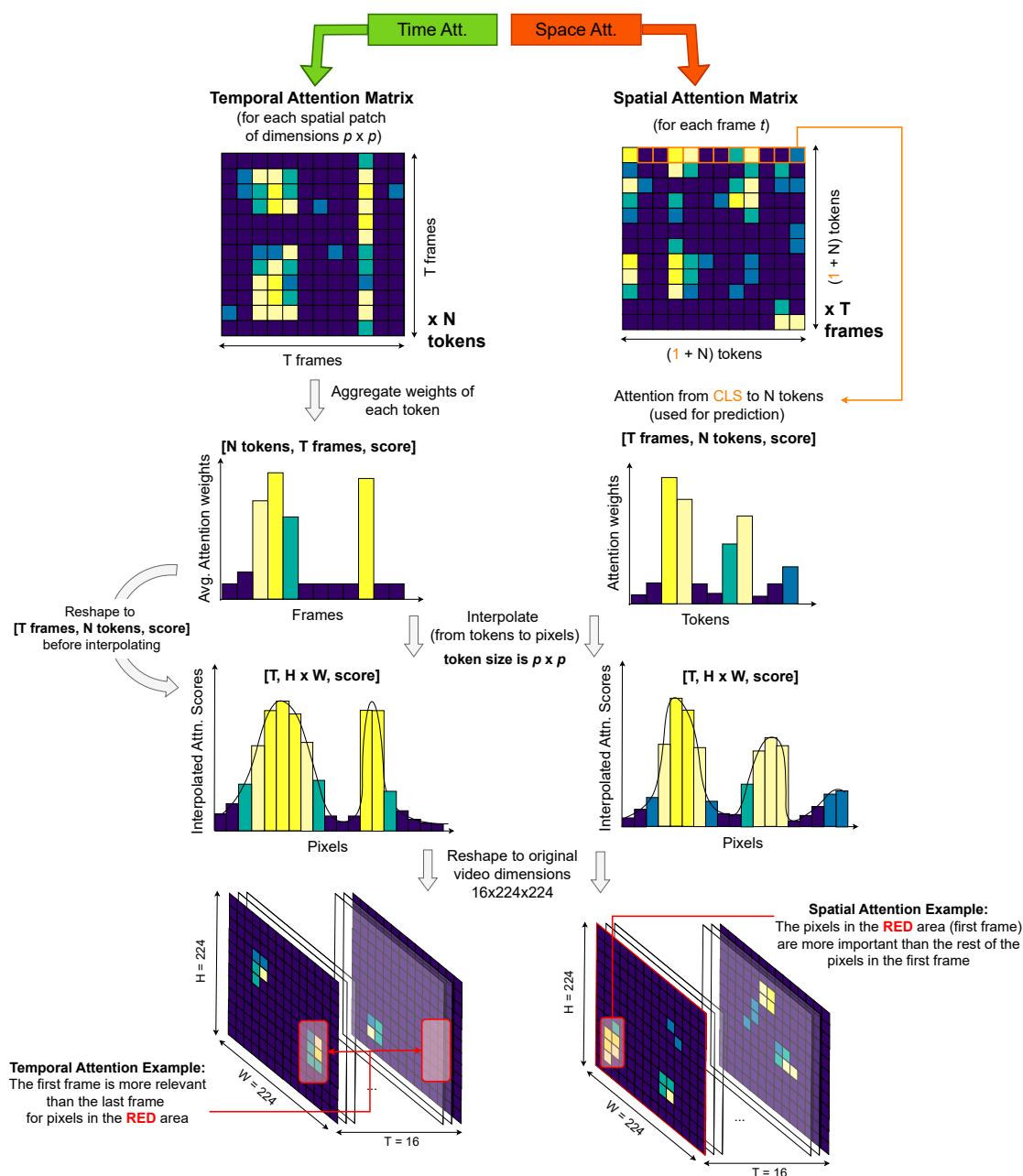


Figure 4.1: Illustration of the process for computing attention weights and visualizing attention maps in TimeSformer. This procedure is consistent across all layers of the architecture. The green and red colors at the top represent temporal and spatial attention, respectively, ensuring consistency with Fig. 2.2. The CLS token refers to the *[class]* token (in orange), which is defined in Section 2.1. The colors used to represent attention weights are approximated from the viridis color palette, which will be employed in our qualitative experiments. In this palette, yellow indicates high attention, while dark blue and purple signify low attention.

both the temporal importance (how significant a frame is for that token) and the spatial importance (how relevant the token is within a specific frame).

For example, if the token is temporally important but not spatially relevant within a frame, its overall score will be relatively low, and vice versa. Therefore, regions that receive high combined scores represent areas that are both spatially and temporally important for each token, providing a more comprehensive view of attention for each token across the video.

Video Swin Transformer

As discussed in Section 2.2, Video Swin computes local attention within non-overlapping windows in a video, resulting in a separate attention matrix for each window. Based on this, the idea is to extract the attention weights, generate a local attention map for each 3D window, and finally stitch together the local attention maps from all windows to form a global attention map. Fig. 4.3 provides an overview of this process. While the shifted window partitioning used in alternate layers typically requires consideration, in our qualitative experiments for transformer interpretability, we did not apply the shifted windows. This decision was made because the accuracy remained similar, and avoiding the shift simplified the construction of attention maps.

To generate the local attention map for each window, since Video Swin does not include a dedicated *[class]* token, we assess the importance of each token by averaging vertically the attention weights in the attention matrix for each token. This idea is inspired by [LWD⁺23], where different types of attention patterns are explained. Essentially, the most relevant patterns are the content-relevant ones (see Fig. 4.2). In the content-agnostic patterns, we observe cases where all tokens either attend to themselves or to their neighbors. In a general sense, this means that all tokens analyzed are important (or share the same importance), as they focus on either themselves or neighboring tokens. By averaging the attention weights vertically, we obtain a similar average attention score for each token.

However, in content-relevant patterns, only one or a few tokens are particularly important. In a vertical pattern (example *L* in Fig. 4.2), a group of different tokens attends to one specific token, indicating that this token holds significant relevance, and its content should be enhanced in the next layer. In the block pattern (example *M* in Fig. 4.2), all tokens within the block are equally important, as they attend to each other, highlighting a broader region that needs emphasis. In both cases, averaging the attention weights vertically preserves the important information. Whether in a vertical or block pattern, the attended tokens will receive a higher average score.

After computing the aggregated (average) attention weights, we obtain an attention score for each 8x7x7 token. The next step involves mapping these scores from the token space to the pixel space. This is achieved by interpolating the attention scores using a scale factor that corresponds to the token size. Since the token dimensions vary across different stages of the model, it is needed to apply the appropriate interpolation factor at each

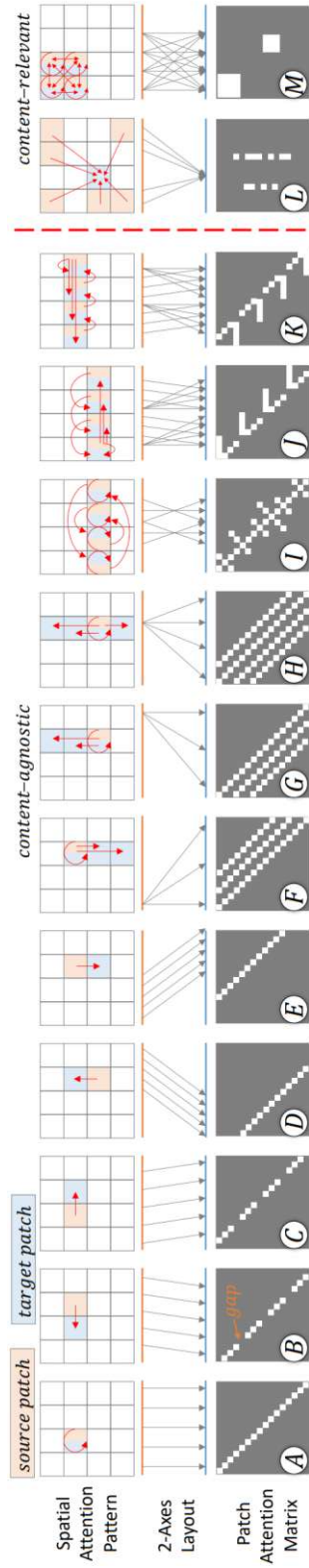


Figure 4.2: The 13 possible attention patterns between image patches presented in the (top) image space, (middle) two-axes, and (bottom) heatmap visualization. Each pattern is a mix of one/multiple of the four basic patterns: diagonal (A-K), horizontal (J, K), vertical (L), and block (M). The first two are *content-agnostic* and often appear in lower-layer heads; the latter two are *content-relevant* and often occur in higher-layer heads. Figure taken from [LWD⁺23].

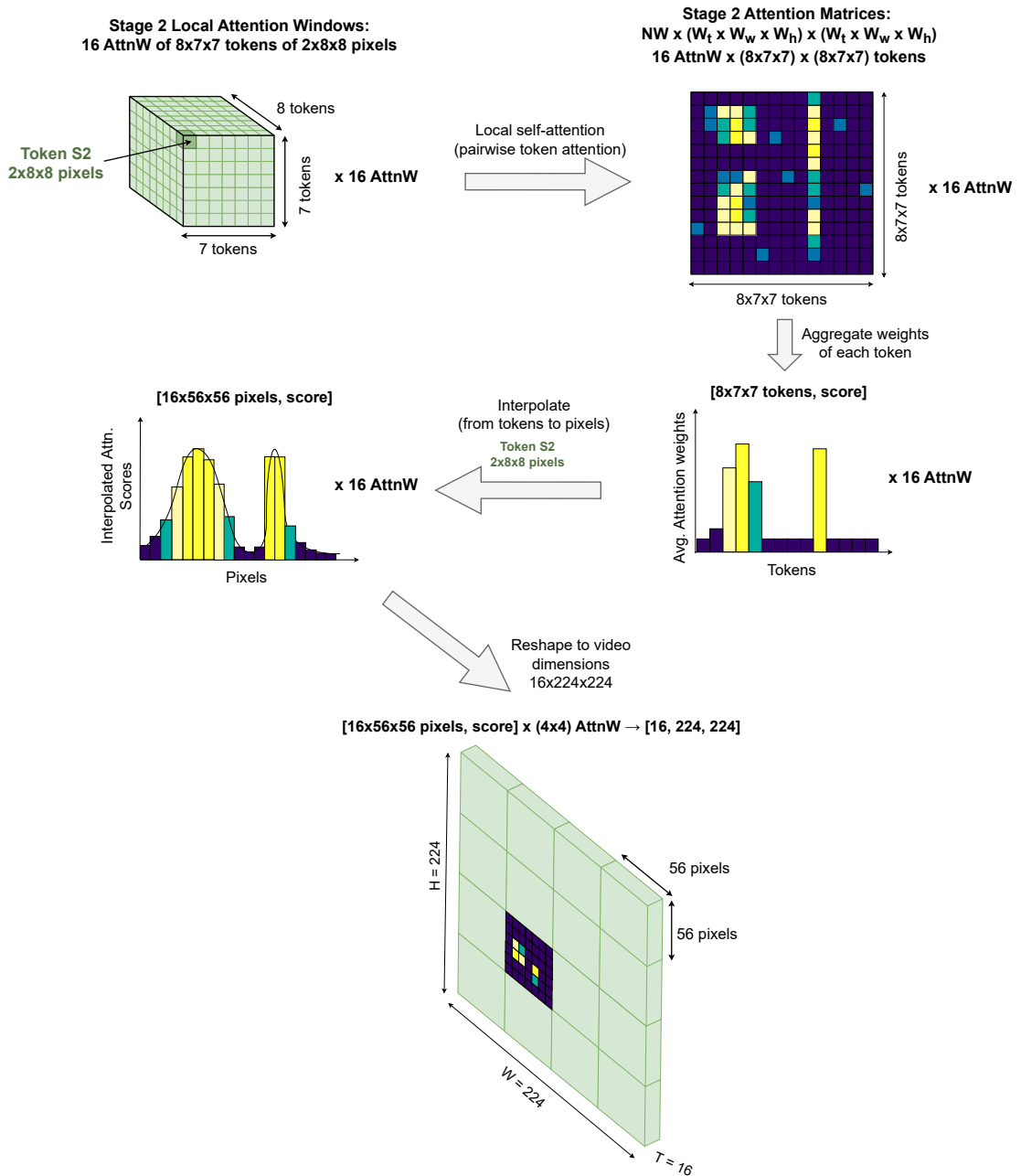


Figure 4.3: Illustration of the process for computing attention weights and visualizing attention maps in Stage 2 of the Video Swin Transformer. The same procedure applies to all other stages, with differences only in token size and the number of attention windows.

stage to ensure accurate mapping from tokens to pixels. After interpolating, we obtain the local attention map, which must be reshaped to conform to the three-dimensional

spatial structure. This local attention map has the same dimensions as the corresponding attention window in pixel space (see bottom part of Fig. 4.3).

Finally, to create the global attention map, it is important to note that for each layer l , there are NW windows arranged in a $1 \times \sqrt{NW} \times \sqrt{NW}$ grid (as shown in the bottom part of Fig. 2.3 for each stage). Each 3D local attention map is then placed in the corresponding position within this grid, based on the index of its associated attention window.

MViT

Computing the attention maps for MViT is relatively straightforward due to the presence of a dedicated `[class]` token. The key consideration in this process is the resolution of the tokens associated with the attention weights. As illustrated in Fig. 2.4, the attention weights used for visualizing the attention maps, and also fed into the classification head after the last layer of the transformer for predictions, represent the attention that the `[class]` token allocates to lower-resolution tokens, which are derived from the pooling operation of K . Therefore, we need to reverse the pooling process to obtain the original spatial dimensions of the input video.

To achieve this, we find that interpolation with an appropriate scale factor is the most effective method for restoring the original video dimensions. The input to the pooling attention mechanism in a general layer l is structured as $(t \times h \times w) \times D$. It is important to note that this input has already undergone downsampling from $T \times H \times W$ to $t \times h \times w$ due to the Q pooling operation in previous layers. Moreover, the input is further downsampled by the pooling of K , reducing it from $t \times h \times w$ to $t'' \times h'' \times w''$. Both of these downsampling factors are integrated into the scale factor applied during interpolation, specifically $\frac{T}{t''} \times \frac{H}{h''} \times \frac{W}{w''}$. This scale factor corresponds to the dimensions in pixels of the pooled tokens in K . After applying the interpolation, we reshape the resulting data into a three-dimensional format to obtain the attention maps in the original video dimensions. An illustration of this entire process can be found in Fig. 4.4.

4.2 Head Importance Metrics

Attention heads in transformer architectures are known to specialize in learning different patterns [PNJ⁺19, KWD⁺21, LWD⁺23]. To better understand their individual contributions, we compute head importance metrics, which help identify which heads play a more crucial role in making predictions. One common technique for evaluating the importance of individual attention heads, as used in [MLN19, LWD⁺23], is *one-head ablation*. This approach systematically disables each attention head one at a time and measures how the model's predictions change as a result. The goal is to isolate the functional contribution of each head from the overall behavior of the model.

To perform the ablation, the attention mechanism is modified at the level of a specific head. Recall that in MHSA, each head h computes an attention matrix A_h , which captures

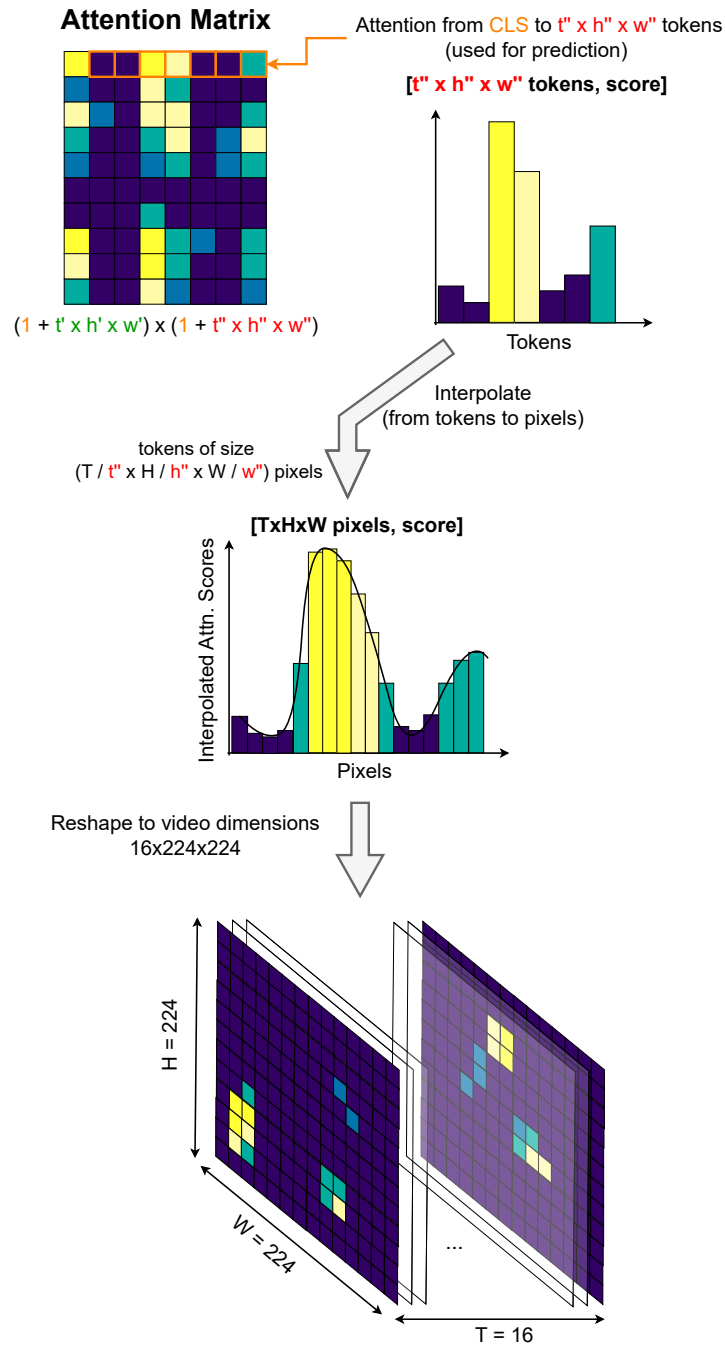


Figure 4.4: Illustration of the process for computing attention weights and visualizing attention maps in MViT. This procedure is consistent across all MViT layers, with variations in the stride used for pooling K resulting in different dimensions (t'' , h'' , w'') at each layer. Consequently, the interpolation factor $\left(\frac{T}{t''}, \frac{H}{h''}, \frac{W}{w''}\right)$ must be adjusted accordingly.

pairwise dependencies between tokens based on learned queries and keys. This matrix is then used to compute a weighted sum of value vectors, producing the head-specific output H_h (as in Eq. (2.3)). In the ablation process, we effectively prune head h by setting its attention matrix A_h to a matrix of zeros:

$$A_h = 0$$

As a result, the corresponding head output $H_h = A_h V_h$ becomes zero, effectively removing head h 's contribution to the final multi-head attention output. Since H_h appears as a zero block in the concatenation used to compute z (see Eq. (2.4)), the pruned head no longer influences the model's output. Moreover, all other heads remain intact, allowing us to isolate the effect of removing a single head without altering the architecture or requiring retraining.

In contrast to [MLN19], where head importance is estimated by directly measuring the change in accuracy after pruning, the approach in [LWD⁺23] differentiates between two types of evaluation: model-level and layer-level. Our work adopts the model-level view, which directly measures how pruning alters the model's final output distribution.

To evaluate model-level impact, we consider how the output probability distribution changes when a particular head is removed. Following Li *et al.* [LWD⁺23], we use two metrics to quantify this change:

- **True Class Probability Change (ΔProb):** This metric captures the absolute change in the predicted probability assigned to the ground truth class label. It is computed as:

$$\Delta\text{Prob}_{l,h} = |P[\text{true class}] - P_{l,h}[\text{true class}]| \quad (4.1)$$

where P represents the model's original predicted distribution, and $P_{l,h}$ denotes the distribution after removing head h from layer l . This allows us to directly assess how pruning a head influences the model's confidence in its correct prediction.

- **Jensen-Shannon Divergence (JSD):** The JSD is a symmetric measure used to quantify the similarity between two probability distributions [MPPP97]. Given two distributions P and Q , the JSD is defined as:

$$\text{JSD}(P||Q) = \frac{1}{2}KL(P||M) + \frac{1}{2}KL(Q||M), \quad (4.2)$$

where $KL(\cdot||\cdot)$ denotes the Kullback–Leibler divergence, and M is the midpoint distribution defined as the average of P and Q :

$$M = \frac{1}{2}(P + Q).$$

In the context of evaluating attention head importance through pruning, JSD measures the divergence between the original predicted probability distribution P and the altered distribution $P_{l,h}$ obtained after pruning head h in layer l :

$$\text{JSD}_{l,h} = \text{JSD}(P||P_{l,h}). \quad (4.3)$$

Compared to KL divergence, JSD has several desirable properties [Nie19]:

- **Symmetry:** Unlike KL divergence, which is asymmetric, JSD treats P and Q equally, ensuring that $JSD(P||Q) = JSD(Q||P)$.
- **Boundedness and Normalization:** JSD is bounded within the range $[0, 1]$, where 0 indicates identical distributions and 1 corresponds to maximally different distributions.
- **Stability:** JSD handles zero probabilities more gracefully, avoiding the infinite or undefined values that can arise in KL divergence when comparing distributions with non-overlapping support.

These properties make JSD a robust and reliable metric for quantifying changes in model predictions due to pruning individual attention heads.

These two metrics (ΔProb , JSD) serve complementary purposes. While ΔProb focuses on the true class and thus directly ties importance to predictive performance, JSD gives a broader view of how much the model’s distributional output shifts overall. In our empirical evaluations, both metrics tended to produce similar patterns across heads and samples. Since they are normalized to the same range $[0, 1]$ and highly correlated, we combine them into a single relevance score, defined as the average between JSD and ΔProb :

$$R = \frac{1}{2}(\text{JSD} + \Delta\text{Prob}) \quad (4.4)$$

This aggregated relevance score R serves as our primary metric for identifying important attention heads. We apply it at varying levels of granularity:

- **Individual samples:** For each input video clip, we compute R to determine which heads are most influential for that specific sample, enabling the identification of the key visual cues driving the model’s prediction.
- **Class-level analysis:** By aggregating relevance scores across all samples belonging to a particular class, we identify heads that consistently contribute to predictions for that class, revealing whether specific visual elements are commonly emphasized.
- **Dataset-level analysis:** Averaging across the entire dataset allows us to detect heads that play a broadly important role across diverse inputs.

When performing interpretability analyses at the class or dataset level, the relevance scores are averaged over all samples in the group, yielding an importance score for each head in each transformer layer.

In [LWD⁺23], layer-level impact is assessed by measuring changes in the activations of subsequent layers after pruning a specific attention head, typically using cosine distance.

However, we choose not to adopt this approach in our study due to the hierarchical design of the models we investigate (specifically, the Video Swin Transformer and MViT). These architectures introduce varying token resolutions across layers, which can complicate the comparison of activations before and after pruning. The changes in token dimensions disrupt the alignment of intermediate representations, making layer-level metrics unreliable for capturing the true effect of pruning. Consequently, we restrict our analysis to model-level impact metrics, which remain consistent regardless of the architecture used.

4.3 Layer Pruning

To address the potential redundancy in transformer architectures and to validate the effectiveness of our importance metrics, we propose a layer pruning approach. Unstructured pruning methods, such as weight-level pruning that zeros out individual attention weights, typically require specialized hardware or libraries to realize speed gains [YHW⁺22, CZS24]. In contrast, our method removes entire layers post-training, leading to a meaningful reduction in model parameters and improvements in inference efficiency.

Our approach begins by evaluating the relevance of each layer through a one-layer-at-a-time ablation strategy, conceptually similar to the method used for assessing head importance. Specifically, we remove one transformer layer at a time while keeping the rest of the model intact. For each training sample, we compute the model’s output probability distribution before and after the removal of the layer. Using these distributions, we calculate the corresponding relevance score R , as defined in Eq. (4.4). By averaging the R scores across all training samples, we obtain an overall importance value for each layer. A schematic overview of this process is shown in Fig. 4.5. We choose R as our primary importance metric because it showed slightly better performance in our ablation study, where we compared different options (JSD, ΔProb , and R), as detailed in Section 5.2.

Our proposed layer pruning strategy offers clear advantages in terms of both computational efficiency and deployment flexibility during inference. Once a model is trained with the full set of layers, it can be pruned to any target depth as needed without any additional fine-tuning, as we found that further training did not significantly improve the results and therefore increased runtime unnecessarily. This enables dynamic on-the-fly adjustment of the model’s complexity depending on available computational resources, real-time latency constraints, or desired accuracy. Moreover, the pruning process is computationally efficient, as it scales linearly with respect to both the number of layers in the model and the number of samples in the dataset. This makes our method a practical solution for optimizing large-scale video transformer architectures.

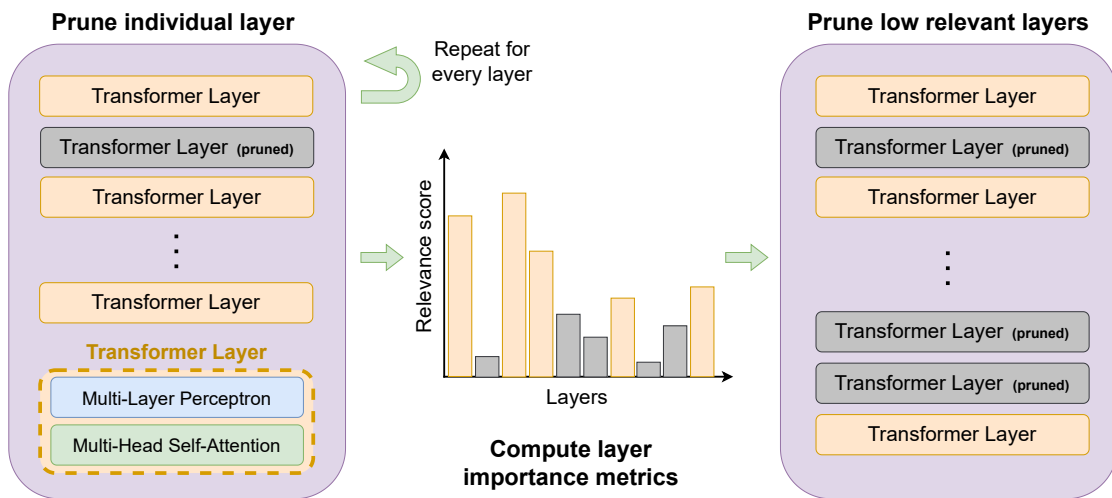


Figure 4.5: Pipeline of our proposed layer pruning method. We initially train a video transformer and then evaluate each layer by removing it and calculating its relevance score. After obtaining all relevance scores, we sort them and prune the model to the desired depth by selecting the top-k layers based on the relevance criteria.

Experiments and Results

This section presents our experiments and results. We begin by introducing the training settings and evaluation metrics. The experimental results are then divided into two parts. In the first part, we provide a quantitative analysis of the pruning method and conduct an ablation study on the selected importance metrics. In the second part, we focus on the interpretability analysis of the video transformer, emphasizing key insights obtained from the attention maps.

5.1 Training

The training process was conducted on an NVIDIA GeForce RTX 4090 GPU utilizing the PyTorch framework. All three networks (TimeSformer, MViT, and Video Swin Transformer) were initialized with pre-trained weights from Kinetics-400 [CZ17], and subsequently fine-tuned separately on the DAI and UCF101 datasets. This fine-tuning phase consisted of 40 epochs, with a batch size of 8 and a learning rate of $2 \cdot 10^{-5}$. Each epoch iterated over the entire training dataset and concluded with a validation step. Adam [KB15] was employed as the optimizer throughout training, and the Cross-Entropy Loss function [ZS18] was chosen due to its suitability for multi-class classification tasks. For evaluation, we use the Top-1 accuracy metric to assess the performance of the different architectures.

5.2 Quantitative Results

In this section, we present the quantitative results of our experiments with the models TimeSFormer, MViT and Video Swin Transformer on our DAI dataset and UCF101. We analyze the impact of layer pruning and discuss model-specific observations.

To validate the utility of our selected importance metrics, we establish two pruning modes, based on the importance results, for comparison:

- **High-Relevance Pruning:** We target layers with the highest relevance scores for pruning.
- **Low-Relevance Pruning:** We identify and prune layers with the lowest relevance scores.

By comparing the outcome of these two pruning modes, we demonstrate the influence of targeting different relevance levels on model performance, and thus confirm the validity of our proposed approach.

To evaluate the effectiveness of our layer pruning method, we used the following metrics, comparing the results before and after pruning a specified number of layers:

- **Accuracy:** The overall top-1 classification accuracy of the model on the test set.
- **FLOPs:** The number of floating-point operations, which serves as a measure of computational complexity.
- **Model size:** The total number of parameters.

Impact of Pruning Mode

Table 5.1 presents the performance comparison of the unpruned model against various pruned models at different depths across three baselines. Results show a substantial contrast between pruning layers with high relevance versus those with low relevance. This trend is consistent across all investigated architectures and datasets.

Specifically, when pruning layers of low relevance, the decline in performance is less severe, particularly noticeable in the slower decrease of accuracy for Video Swin Transformer and MViT models. Conversely, removing the most crucial layers, identified based on the computed relevance score defined in Eq. (4.4), results in a significant and rapid drop in accuracy. Even with minimal parameter reduction, removing only one layer (which equates to 3.6% of parameters for Video Swin Transformer and 0.4% for MViT), leads to a significant accuracy drop. For instance, comparing to the baseline models without any pruned layers, in the DAI dataset, there is a 30.43% accuracy decrease for Video Swin Transformer and 14.67% decrease for MViT, while in UCF101, there is a 9.04% decrease for Video Swin Transformer and 17.1% decrease for MViT. The TimeSformer exhibits even more drastic reductions, with accuracy plummeting to 9.78% in DAI and 6.85% in UCF101 after pruning just one layer. These results emphasize the critical role of high-relevance layers in facilitating accurate action recognition, highlighting the importance of their preservation.

However, removing a layer lacking relevance, despite potentially containing more parameters, leads to an almost negligible decrease (sometimes even slight increase in DAI) in accuracy across all cases. In TimeSformer, the change in accuracy is +4.21% and -3.7%

Model	Pruned Layer Relevance	# Layers Pruned	Top-1 Accuracy		Params. ($\Delta\%$)	FLOPs ($\Delta\%$)
			DAI	UCF101		
TimeSformer	-	0	85.87	91.01	121.28M	392.51G
	Lowest	1	90.08	87.31	-8.3%	-8.5%
		2	74.46	77.56	-16.6%	-16.8%
		3	52.75	58.34	-24.8%	-24.9%
	Highest	1	9.78	6.85	-8.3%	-8.5%
		2	8.15	4.73	-16.6%	-16.8%
3		2.17	1.24	-24.8%	-24.9%	
Video Swin Transformer	-	0	88.97	91.22	49.52M	83.06G
	Lowest	1	90.26	90.96	-2.4%	-3.9%
		2	90.06	90.72	-7.3%	-7.8%
		3	88.56	89.69	-10.9%	-11.7%
		6	88.04	86.07	-21.8%	-23.5%
		8	83.70	79.01	-29.1%	-31.3%
	Highest	1	58.15	82.18	-3.6%	-5.6%
		2	14.67	62.62	-4.8%	-11.3%
		3	13.59	46.58	-29.1%	-12.9%
		6	3.26	4.62	-31.2%	-27.5%
8		3.26	1.82	-38.5%	-35.4%	
MViT	-	0	90.76	94.00	34.31M	64.46G
	Lowest	1	90.32	93.94	-5.2%	-5.1%
		2	90.29	93.81	-10.4%	-10.2%
		3	90.21	92.84	-15.6%	-15.4%
		5	86.96	91.11	-26.1%	-25.6%
	Highest	1	76.09	76.90	-0.4%	-7.5%
		2	51.09	50.04	-1.8%	-13.4%
		3	37.50	41.51	-7.0%	-18.5%
		4	37.50	41.51	-7.0%	-18.5%
5		25.00	28.07	-32.9%	-28.4%	

Table 5.1: Main results on the test set for DAI and UCF101 datasets. Top-1 accuracy, parameter counts and FLOPs are presented for three video transformers. The model with 0 layers pruned serves as the unpruned baseline, while the remaining models indicate the number of pruned layers and their relevance levels (Lowest or Highest).

in DAI and UCF101, respectively. For the Video Swin Transformer, the change is +1.29% and -0.26% in DAI and UCF101, respectively, while for MViT, the drop is -0.44% and -0.06% in DAI and UCF101, respectively. However, the accuracy outcomes in the DAI dataset exhibit less consistency compared to those in UCF101, attributed to the relatively smaller size of the DAI test set. Additionally, it is important to highlight that the change in parameters remains consistent across both datasets, as layers of equivalent size are removed. The layers that are dropped can be found in Table 5.2, while the parameters for each layer are detailed in Table 2.1. For instance, in the case of VideoSwin Table 2.1c, all layers in Block 3 (comprising 18 layers indexed from 4 to 21, since indexing begins at 0 in Block 1) have the same number of parameters because they maintain the same attention matrix resolution in the MHSA mechanism.

Model-Specific Observations

We assess how the impact of pruning on accuracy varies across the three models. Examining the low-relevance pruning results, the TimeSformer model exhibits a substantial decrease in accuracy with each layer pruned. This observation suggests that with only 12 structurally identical layers, TimeSformer might not have highly specialized layers. Instead, each layer likely contributes crucial information for accurate predictions. This implies broad information distribution across all layers.

On the other hand, Video Swin Transformer and MViT show greater resilience, indicating a higher level of prunability. For example, Video Swin Transformer can tolerate a 22% parameter reduction with only a 7.15% accuracy drop on UCF101. MViT performs even better, enduring a 26% reduction with less than a 3% accuracy decrease on the same dataset. Possible explanations for this resilience are that the heads in less important layers are either highly specialized and attend to non-relevant patterns for predictions, or alternatively, they might focus on recurrent general patterns captured elsewhere in the architecture. As a result, the removal of information from these layers does not lead to a significant drop in accuracy.

Layer Importance

Table 5.2 provides insights into the specific layers pruned using both high-relevance and low-relevance pruning strategies across the three models. Notably, we refrained from pruning layers {1, 3, 14} in MViT. As seen in Table 2.1b, these layers represent the beginning of a new block (with layer numbering starting from 0, meaning the only layer in Block 1 is layer 0). Specifically, layer 1 is the first layer in Block 2, layer 3 is the first layer in Block 3, and layer 14 is the first layer in Block 4. As discussed in Section 2.2, each of these layers serves as a transition point between blocks, where the model adjusts the feature representations by lowering their resolution, performing pooling on the Q matrix within their Multi-Head Pooling Attention module. Removing these layers would disrupt the architectural flow, causing dimension inconsistencies in the layers that follow. This could lead to critical failures in the model’s ability to process data, as the subsequent layers expect inputs of specific dimensionality.

General Observation. Interestingly, all three models exhibit shared dropped layers, particularly following a consistent pattern with high-relevance layers. These high-relevance layers are consistently observed as the first layers of the transformer (TimeSformer: [0, 1], Video Swin Transformer: [0, 1], MViT: 0), proving their key role in capturing essential, high-level information from the video sequence that serves as the foundation for understanding. Subsequently, as more high-relevance layers are pruned, the last layers of the transformer are affected (TimeSformer: 11, Video Swin Transformer: [22, 23], MViT: 15), which typically encode fine-grained details specific to the action, crucial for accurate prediction.

On the other hand, low-relevance pruning exhibits a wider range of dropped layers across models, indicating greater diversity and distribution compared to high-relevance

Model	# layers pruned	High-relevance layers		Low-relevance layers	
		DAI	UCF101	DAI	UCF101
TimeSformer (12 layers)	1	0	0	10	5
	2	0, 11	0, 11	8, 10	5, 8
	3	0, 1, 11	0, 1, 11	7, 8, 10	5, 6, 8
Video Swin Trans- former (24 layers)	1	0	0	7	9
	2	0, 1	0, 1	7, 19	9, 15
	3	0, 22, 23	0, 22, 23	7, 19, 21	9, 15, 17
	6	$\begin{bmatrix} 0, 1, 2, \\ 3, 22, 23 \end{bmatrix}$	$\begin{bmatrix} 0, 1, 2, \\ 3, 22, 23 \end{bmatrix}$	$\begin{bmatrix} 7, 9, 15, \\ 17, 19, 21 \end{bmatrix}$	$\begin{bmatrix} 7, 9, 13, \\ 15, 19, 21 \end{bmatrix}$
8	$\begin{bmatrix} 0, 1, 2, 3, \\ 4, 6, 22, 23 \end{bmatrix}$	$\begin{bmatrix} 0, 1, 2, 3, \\ 6, 8, 22, 23 \end{bmatrix}$	$\begin{bmatrix} 7, 9, 13, 15, \\ 17, 18, 19, 21 \end{bmatrix}$	$\begin{bmatrix} 5, 7, 9, 13, \\ 15, 17, 19, 21 \end{bmatrix}$	
MViT (16 layers)	1	0	0	11	12
	2	0, 2	0, 2	7, 11	6, 12
	3	0, 2, 4	0, 2, 4	7, 11, 12	6, 8, 12
	5	0, 2, 4, 8, 15	0, 2, 4, 7, 15	5, 7, 10, 11, 12	5, 6, 10, 11, 12

Table 5.2: Layers dropped in a single experimental run. The number of layers pruned in each case is displayed, along with the indices of the pruned layers in both low-relevance and high-relevance pruning modes, based on the computed relevance score. Consistent results are observed across multiple runs.

layers. Nevertheless, a common characteristic among these layers is their placement as intermediate layers within the model architecture, suggesting that they might contain a mix of redundant and less critical information.

The relevance scores for each architecture in both datasets are depicted in Fig. 5.1 using boxplots. These boxplots show the relevance score for each layer, considering all samples from the training set. As previously mentioned, a consistent trend emerges across models: the first and last layers exhibit the highest relevance scores, while the intermediate layers tend to have lower scores.

Video Swin Transformer. Among the models, Video Swin stands out for its minimal variability in relevance scores, both across datasets and within individual layers. Notably, the first two layers are highly crucial, followed by the last layers, with a significant gap in relevance between them. The intermediate layers, however, contribute very little, making them prime candidates for pruning without a significant impact on performance. Interestingly, among these intermediate layers, the odd-numbered ones are slightly less relevant than the even-numbered ones, though the difference is minimal. The consistently low relevance in these intermediate layers across samples makes them ideal for removal without losing significant information.

TimeSformer. On the other hand, TimeSformer, particularly on the DAI dataset, shows much higher variability across samples. This suggests that even when pruning low-relevance layers in TimeSformer, some of those layers may still carry important information for certain samples, which explains the higher accuracy drop observed. The variation in relevance across samples makes low-relevance pruning more challenging for this model.

MViT. For MViT, we see that the layers we opted not to prune (layers 1, 3, and 14) due to their role in dimensionality changes are indeed some of the most relevant, along with the first (layer 0) and last layer (layer 15). The other layers exhibit low relevance scores and little variability across samples, meaning their contribution remains consistently low. This consistency makes these layers ideal candidates for pruning, as they can be removed without losing significant information across different samples.

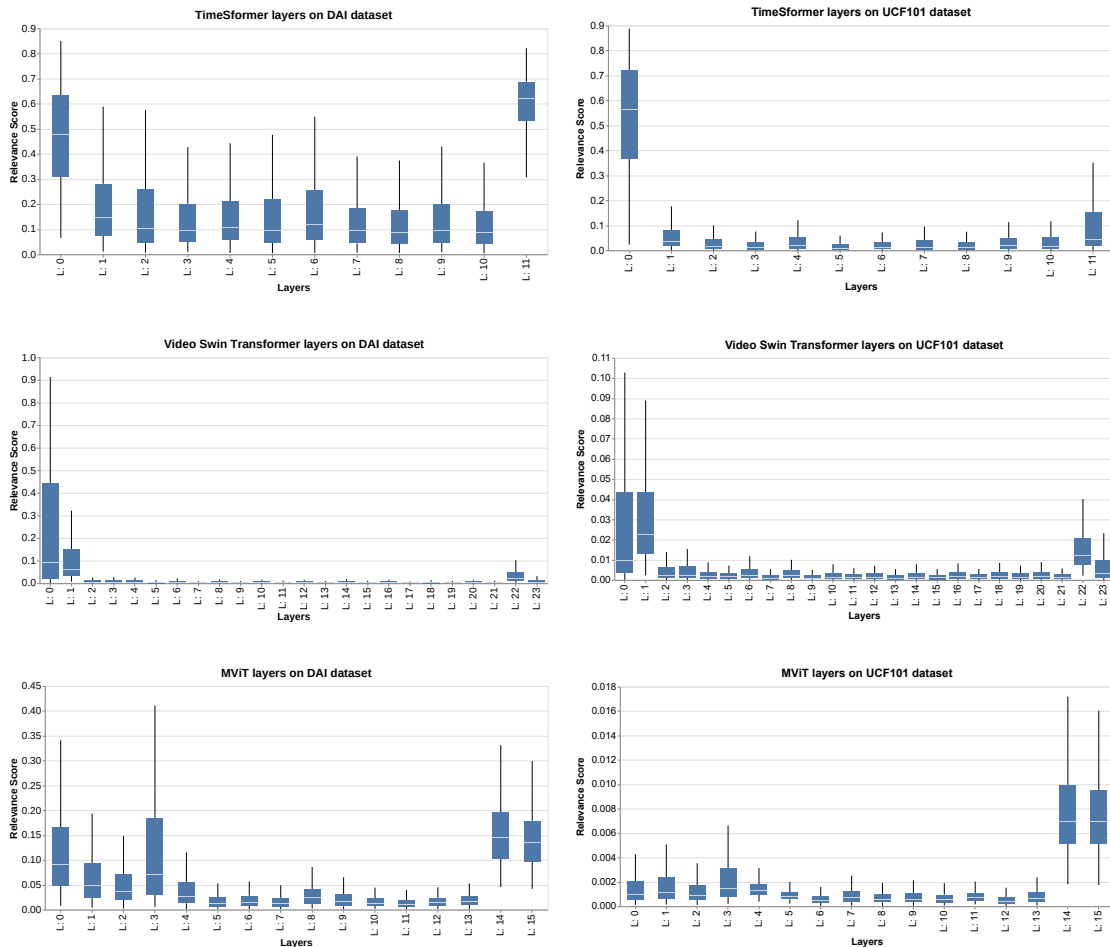


Figure 5.1: Boxplots of layer relevance scores for the TimeSformer, Video Swin Transformer, and MViT architectures on both DAI and UCF101 datasets. These scores are calculated on the training sets to guide the selection of layers for further pruning. To enhance visualization, the vertical axis range has been adjusted, excluding some outliers.

Ablation Study on Importance Metric

To determine the most effective metric for quantifying layer relevance, we conducted an ablation study comparing JSD, ΔProb , and the R score, which is the average of

the first two metrics. As shown in Table 5.3, the accuracy values across different cases, particularly in the DAI dataset, are quite similar regardless of the metric used. This consistency is especially evident when pruning high-relevance layers, where all three metrics yield nearly identical accuracy scores across different datasets and architectures. This is because all three metrics effectively identify and agree on the high-relevance layers. However, a slight difference emerges when removing low-relevance layers: JSD tends to perform the worst, while the R score delivers the best results, closely followed by the Δ Prob metric. These findings suggest that while all metrics are reliable for identifying critical layers, the R score offers a slight edge in preserving accuracy when pruning less important layers.

Model	Pruned Layer Relevance	Importance Metric	UCF101	DAI
TimeSformer (3 layers pruned)	Lowest	JSD	57.88	52.75
		Δ Prob	58.21	52.75
		R	58.43	52.75
	Highest	JSD	1.24	2.17
		Δ Prob	1.24	2.17
		R	1.24	2.17
Video Swin (6 layers pruned)	Lowest	JSD	83.08	88.04
		Δ Prob	85.12	88.04
		R	86.07	88.04
	Highest	JSD	4.62	3.26
		Δ Prob	4.62	3.26
		R	4.62	3.26
MViT (5 layers pruned)	Lowest	JSD	90.11	86.15
		Δ Prob	91.11	86.96
		R	91.11	86.96
	Highest	JSD	30.11	25.00
		Δ Prob	28.07	25.00
		R	28.07	25.00

Table 5.3: Comparison of Top-1 accuracy outcomes when pruning layers based on different importance metrics (JSD, Δ Prob, and R score) across various datasets and architectures.

5.3 Qualitative Results

In this chapter, we explore the interpretability of video transformers by examining the differences in attention maps across various architectures and also analyze how these maps evolve with layer depth within each model. Additionally, we identify specialized attention heads within the datasets and discuss their roles and generalization capabilities.

Attention Head Relevance Across Transformer Layers

To demonstrate the efficacy of our selected metrics, we employ attention visualization based on the significance of individual heads within the model. By comparing the focus areas of high-importance heads with those of low-importance heads, we can gain insights

into the model’s decision-making processes. Fig. 5.2 illustrates this with different test samples, showing attention weights of the heads with the highest and lowest importance at various depth levels across the three video transformers: TimeSformer, Video Swin, and MViT. For visualization purposes, we select single frames that are empirically determined to be the most significant within the video sequence.

For TimeSformer, we visualize the spatial attention maps based on the `[class]` token scores found on the spatial attention matrix (as described in Section 4.1. Although we are not considering the temporal attention here, the spatial attention alone is sufficient to highlight key regions within individual frames. This approach allows for a meaningful comparison of TimeSformer’s attention maps with those of Video Swin and MViT, as in this case all three models are evaluated based on their ability to capture important spatial features within a frame. However, in the following sections, we delve deeper into the role of temporal attention in Video Swin (and similarly in MViT) by analyzing its application across entire video sequences. For the TimeSformer architecture, we extend our analysis to not only examine temporal attention weights but also explore how both spatial and temporal attention can be combined.

In Fig. 5.2, we observe both similarities and differences across the architectures in how heads at different depths pay attention to patterns. A common trend across all models is that in the early layers, heads (whether relevant or not) tend to focus on more general patterns. However, in deeper layers, attention narrows to specific regions or objects critical for the prediction. One noticeable difference is the hierarchical structure of Video Swin, visible in the early layers where a mesh-like attention pattern emerges due to the small token sizes (2x4x4), capturing global features like gradients or contrasts.

At intermediate levels, high-importance heads across all architectures begin focusing on key areas for action prediction, but the attention is less refined than in the deeper layers. For example, in the “Apply Lipstick” action (first row of TimeSformer in Fig. 5.2), the intermediate layer head focuses on the person’s face but also spreads to the background, introducing some noise. However, in the deepest layer, the attention becomes more specialized, focusing solely on the face contour. Similarly, in the “Frisbee Catch” action (first row of Video Swin in Fig. 5.2), the intermediate layer of Video Swin still pays attention to the background, but in the deeper layers, attention shifts specifically to the players’ legs, which are key to identifying the action. Another example can be seen in MViT for the “Eating-drinking” class (first row of MViT in Fig. 5.2), where the intermediate layer focuses on the driver’s arm and bottle, but in the deep layers, the attention sharpens to focus only on the bottle, emphasizing the critical visual cue for that specific action.

The distinction between the highest and lowest relevance heads is clear, as the lowest relevance heads frequently provide little useful information for class prediction, regardless of the layer depth. For instance, in TimeSformer, the lowest relevance head in the early layers (first row in Fig. 5.2) focuses on the background rather than the subject, while in the second example, the intermediate layer’s least relevant head attends to the background instead of the driver’s hand for the “Sunblinds” action. Similarly, in the deepest layer



Figure 5.2: Visualization of the attention maps produced by the highest and lowest relevance heads for TimeSformer, Video Swin and MViT. Red boxes indicate the lowest relevance heads within the corresponding layers, while green boxes indicate the highest relevance heads within the corresponding layers. Early layers refer to the first layer for TimeSformer, and first two layers for both Video Swin and MViT, while deep layers correspond to the last one in TimeSformer, and last two layers in Video Swin and MViT. Intermediate layers encompass those in between. Colormap: viridis (dark blue: low attention, bright yellow: high attention). Best viewed in color.

(last row of the TimeSformer section in Fig. 5.2), the lowest relevance head focuses on the driver’s head rather than the phone in the “Texting-on-phone” action. This trend holds across architectures. For Video Swin (see last row corresponding to Video Swin model in Fig. 5.2), the lowest relevance head in the deep layer focuses on the steering wheel rather than the phone, which is irrelevant for recognizing the “Texting-on-phone” action. Similarly, in MViT, the lowest relevance heads misdirect attention to irrelevant regions like the seatbelt instead of focusing on the bottle or the arm for the “Eating-drinking” action (see first row of MViT in Fig. 5.2).

From these observations, we find that deeper layers contain specialized heads that focus on essential visual cues for accurate predictions. Additionally, the chosen metrics effectively identify the key heads that provide the most relevant information for prediction.

Specialized Heads

Building on these findings, we further investigate the attention mechanisms within the deeper layers of the Video Swin architecture, specifically analyzing the DAI dataset. Although all models demonstrate specialized heads, we focus on Video Swin due to its larger number of heads – 24 heads in each of the last two layers – allowing for a broader diversity of specialized heads. Additionally, we choose the DAI dataset because of its smaller size and limited class range, making it more practical to explore the similarities and differences among classes.

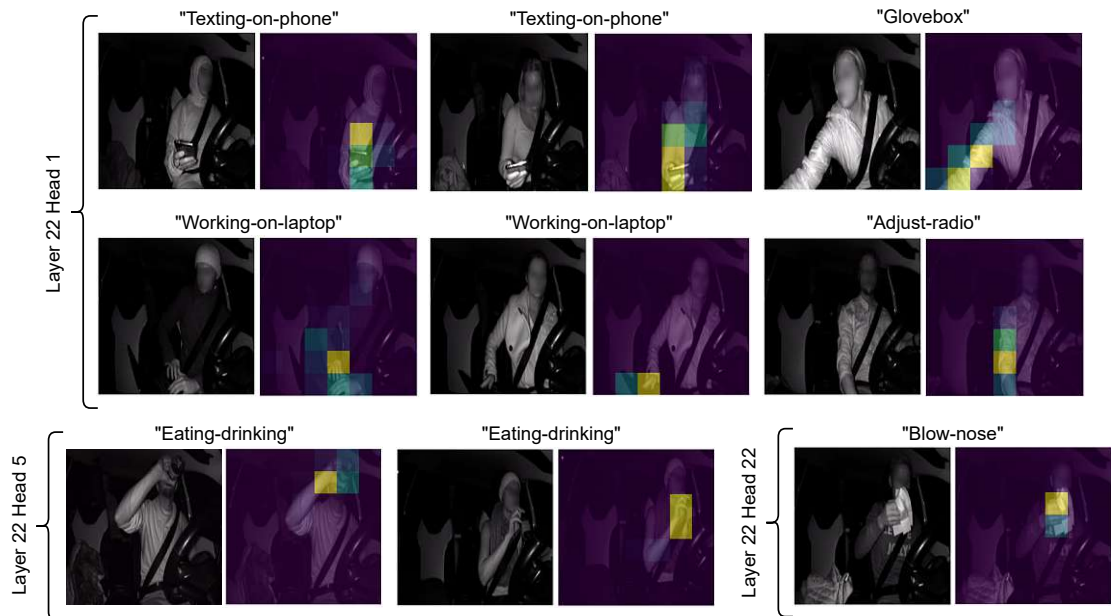


Figure 5.3: Attention maps of specialized relevant heads using Video Swin for various test samples on the DAI dataset, revealing that some classes share a specialized head.

For this analysis, we identify the most relevant specialized head within the final block,

Class action	Most relevant head
Texting-on-phone Working-on-laptop Put-onoff-jacket Glovebox Adjust-radio	L: 22, H: 1
Coughing-yawning-sneezing Talking Adjust-rear-mirror Talking-on-phone Sunglasses	L: 22, H: 4
Eating-drinking	L: 22, H: 5
Steering Sunblinds Switch-gear Read-paper	L: 22, H: 11
Reach-objects	L: 22, H: 15
Drowsy Adjust-side-mirror Seatbelt	L: 22, H: 17
Blow-nose	L: 22, H: 22

Table 5.4: Most relevant specialized head by class in the DAI dataset based on the relevance score computed for all training samples among heads on the last two layers of the Video Swin (layers 22 and 23). All identified specialized heads are located in layer 22, making it more relevant.

which comprises layers 22 and 23, for each class in the DAI dataset. These two last layers are highly relevant, as evidenced by results in Table 5.2. Table 5.4 displays the most relevant head for each video class, revealing instances where classes with similar movements or patterns share common specialized heads. The aggregation of classes in this table results logical and insightful. For instance, head 4 in layer 22 consistently focuses on the driver’s head, making it particularly relevant for classes where actions occur near this region. Actions like “Talking” or “Talking-on-phone” naturally involve proximity to the driver’s head, as individuals typically position their phones against their ears during these activities. Similarly, the class “Sunglasses” also benefits from the attention of head 4 in layer 22, as the model examines the driver’s face to determine whether they are wearing sunglasses. Examples of the attention maps corresponding to this head are illustrated in Fig. 5.4 (right part).

Another interesting example is layer 22, head 1. This head is interesting because it identifies various patterns within the same spatial regions where actions take place. For instance, in the cases of “Texting-on-phone” and “Working-on-laptop”, both involve the same spatial region where the action occurs, with the head tracking the driver’s fingers either grabbing the phone or typing on the laptop. Similarly, for actions like “Glovebox”, “Adjust-radio”, and “Put-onoff-jacket”, the head tracks the driver’s arms, which move

within a similar spatial area. Examples of this behavior can be seen in the first two rows of Fig. 5.3.

On the other hand, there are classes with very specific actions that do not share specialized heads with any other class. This is primarily because these heads track unique items that are not present in other classes. For example, layer 22, head 5 specifically monitors the bottle and the driver’s hand as it approaches the mouth for the “Eating-drinking” action. Likewise, layer 22, head 22 is dedicated to tracking the paper tissue used in the “Blow-nose” action. Attention maps depicting these cases are illustrated in Fig. 5.3 (last row).

Temporal Attention

As previously discussed, Video Swin utilizes a joint spatial-temporal attention mechanism using 3D tokens. This allows the model to focus not only on specific regions within frames but also on key moments across the video sequence. This temporal specialization becomes especially prominent in the deeper layers, where specialized heads focus on a select few frames that capture the most critical aspects of the action. These heads exhibit both spatial and temporal specialization, concentrating on specific patterns in certain regions while attending to only a limited number of frames.

As shown in Fig. 5.4, the model effectively isolates key frames that contain the most relevant visual information. For instance, when the driver is texting, the model focuses solely on frames where the phone is visible, ignoring other irrelevant frames. Similarly, in the “Eating-drinking” action, attention narrows down to the specific frame where the drinking motion occurs, while other frames receive minimal or no attention. This temporal focus reduces noise and enables the model to prioritize only the essential moments for action recognition.

We also observe a distinct temporal behavior in certain specialized heads, specifically in layer 22 head 4 of the DAI dataset for Video Swin. This head, which we previously highlighted for its focus on the driver’s head, stands out not only for actions occurring around the face but also for its consistently high relevance across all test samples. In fact, it holds the highest relevance score across the entire DAI dataset for Video Swin, considering only the last two layers. Beyond spatially identifying the driver’s head, this head demonstrates strong temporal consistency, tracking the driver’s head across every frame, regardless of the video class (see bottom part in Fig. 5.4). While other heads concentrate on specific key frames, layer 22 head 4 attends to the driver’s head continuously throughout the video. Even in classes like “Working-on-laptop”, where it is not the most relevant head for the action, it still tracks the head in every frame, showing a unique and stable behavior across all samples.

TimeSformer Attention Analysis

As discussed earlier, Video Swin (and similarly MViT) utilize joint spatiotemporal attention mechanisms, allowing us to analyze how temporal attention operates over entire

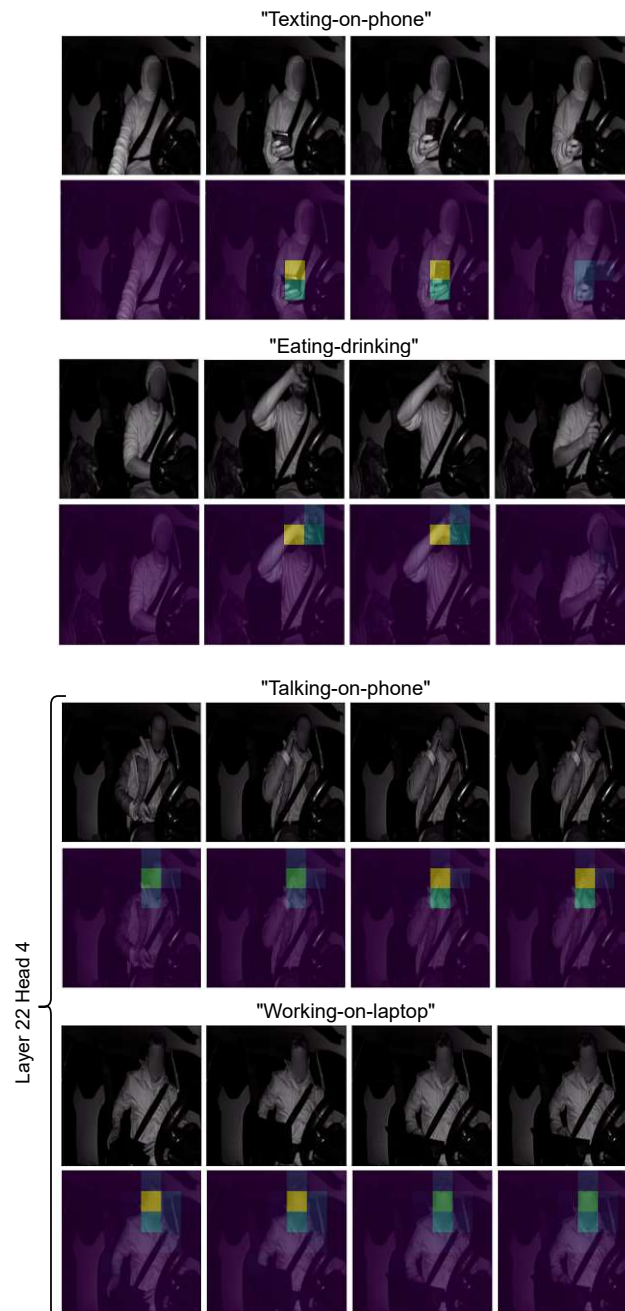


Figure 5.4: Attention maps for various DAI samples across a temporal sequence. On the first two examples, specialized heads that focus on specific objects (e.g., phone) demonstrate high temporal focus by highlighting only one or two frames. On the bottom, head 4 in layer 22 consistently tracks the driver’s head across multiple frames. Although the full video sequence in the experiments contains 16 frames, we display only the 4 consecutive frames where the main action occurs and attention is most concentrated.

video sequences. However, we cannot directly compare these models to TimeSformer due to its divided attention mechanism, which separates spatial and temporal attention. Therefore, we analyze TimeSformer individually. Since TimeSformer maintains square matrices and consistent token sizes across layers, we can visualize attention maps for the most relevant heads and compare this with results from the attention rollout method [AZ20] to evaluate which technique offers more insightful qualitative results.

As shown in Fig. 5.5 (second row), we first plot the spatial attention of the most relevant head of the last two layers (specialized head). This specialized head, as previously discussed, tracks the specific regions in each frame that are critical for the prediction. For instance, in the “Blow-nose” sample, this specialized head tracks the driver’s head, anticipating the action around the face. Likewise, in the “Read-paper” sample, it consistently tracks the newspaper throughout the video sequence, indicating its importance in recognizing the action.

Next, we visualize the temporal attention to examine which frames are most significant for the model’s prediction (see Fig. 5.5, third row). By comparing the highlighted areas across frames, we can identify the moments that the model finds most relevant. For instance, in the “Blow-nose” action, frames 2 and 4 stand out as the most important ones temporally, particularly in areas where the paper tissue is visible. Since the tissue’s spatial position varies between frame 2 and frame 4, the temporal attention highlights the tissue area in each. If the tissue had remained in the same position across both frames, the model might either concentrate more on one frame or distribute its attention evenly between them, depending on how much information each frame contributes to the prediction. On the other hand, frames 1 and 3 show low attention, as they contribute less due to the tissue not being in focus.

For the “Read-paper” action, the model’s attention is heavily concentrated on frame 1, specifically on the hand and the newspaper (see Fig. 5.5, third row, right part). In subsequent frames, only a few patches around the newspaper are highlighted. This suggests that the model gathers most of the information it needs from the first frame, where the newspaper is already fully visible. Since the spatial position of the newspaper remains largely unchanged throughout the sequence, the model does not need to focus on different spatial regions in later frames, indicating that frame 1 alone is sufficient for the prediction. This comparison between actions illustrates how the model adapts its temporal attention based on the spatial dynamics and visibility of key objects in the video.

As seen in Fig. 5.5 (row 4), applying element-wise multiplication to combine the spatial and temporal attention maps highlights only the most specific and relevant areas of the image. This approach filters out noise both spatially and temporally, resulting in a much cleaner and more precise attention map. The focus narrows only to the critical regions that contribute most to action prediction, enhancing the model’s interpretability by isolating key moments and regions. In contrast, attention rollout offers a broader view by aggregating attention across all heads and layers (see Fig. 5.5, last row). While this provides insight into how the model distributes its focus over time, it can obscure the finer

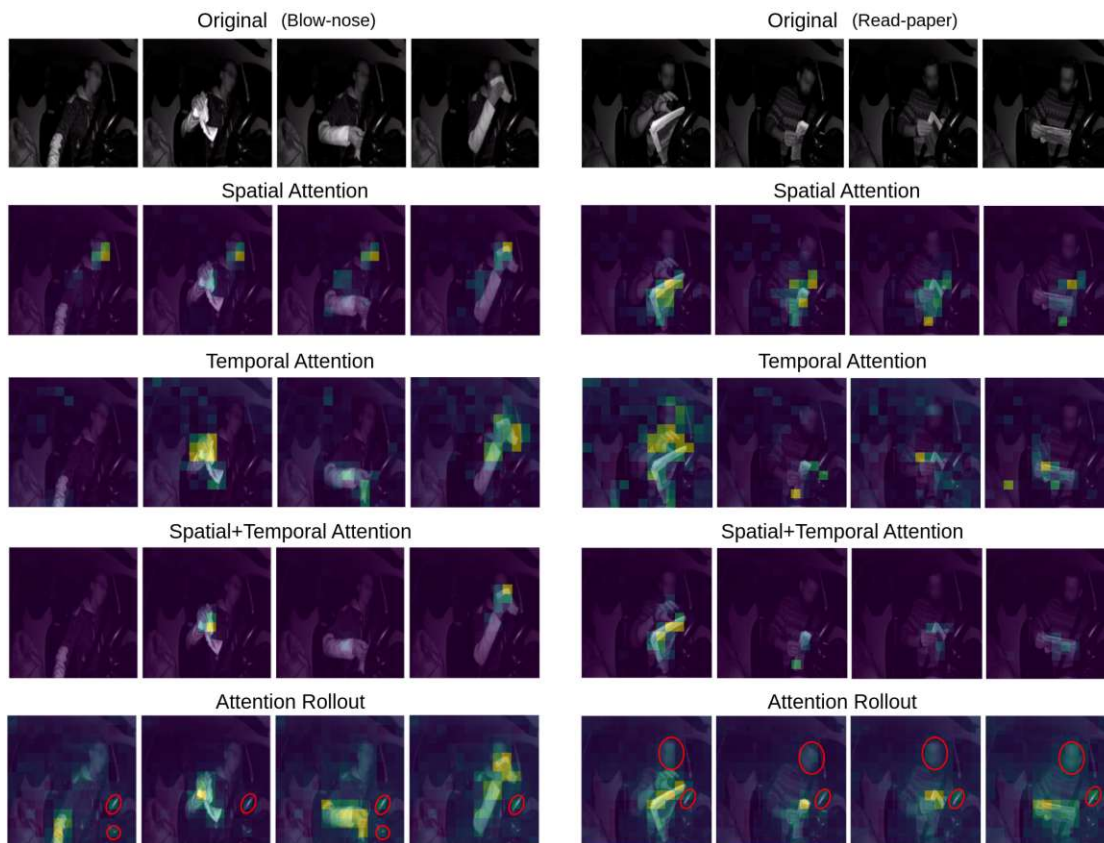


Figure 5.5: Attention maps for two DAI samples using TimeSformer. The original video frames (first row) are shown alongside the spatial attention from the most relevant head in the final layer (second row), the temporal attention from the most important temporal head (third row), and the combined spatial-temporal attention through element-wise multiplication (fourth row). The attention rollout is provided for comparison (last row), with irrelevant areas highlighted by this method (e.g., brightness bias) circled in red.

details captured by specialized heads. In addition, a closer inspection of the rollout maps reveals some irrelevant tokens being highlighted, in particular bright spots on the steering wheel, which are present in both samples analyzed and consistently appear across the entire test set. This brightness issue arises from the characteristics of the DAI dataset, which employs NIR images where the background is generally darker than the driver. Since focusing on the driver is crucial for action prediction, there is an implicit bias favoring brighter areas over darker ones. This phenomenon, referred to as brightness bias [KKK⁺19], is particularly evident in the early layers of TimeSformer, where attention heads tend to prioritize general patterns such as bright-dark contrasts rather than action-specific details. When using the attention rollout method, attention to these irrelevant tokens in the initial layers is carried through to subsequent layers, resulting in their presence in the final attention maps. However, by visualizing only the most relevant

heads, this issue is mitigated, as these heads typically do not focus on such bright spots. Therefore, our approach may offer a more effective way to study interpretability by reducing biases and enhancing the model’s focus on relevant information.

Specialized Head Generalization on Drive&Act

As established in earlier analyses, we identified the most relevant specialized heads within the final block of the Video Swin Transformer for each action class in the DAI dataset. Leveraging these findings, we aim to assess whether the specialized heads identified in the DAI dataset can semantically generalize to a different but closely related driver-specific dataset (the Drive&Act dataset) without any additional fine-tuning. This evaluation is performed in a zero-shot manner using the DAI-trained model as a black box. To do this, we perform a forward pass using the Video Swin model trained on the DAI dataset with samples from the Drive&Act dataset, extracting the attention maps from the corresponding specialized heads for those specific samples.

Given that the DAI and Drive&Act datasets differ in their class taxonomies (20 classes in DAI versus 34 in Drive&Act) and lack a complete one-to-one mapping, we manually align a subset of semantically similar actions to assess the generalization behavior of specialized attention heads.

As illustrated in Fig. 5.6, as a first example, we focus on head 1 in layer 22, which was identified as the most relevant head for the actions like “Texting-on-phone” and “Working-on-laptop” in the DAI dataset. The “Texting-on-phone” class is mapped to “Interacting-with-phone” in Drive&Act, while “Working-on-laptop” exists in both datasets. When we apply the DAI-trained model to corresponding samples in the Drive&Act dataset, we observe that this head generalizes in a consistent manner: its attention continues to focus on the driver’s hands and the object being manipulated (either a phone or a laptop) despite no fine-tuning being performed.

As a second example, we examine the “Eating-drinking” action class in the DAI dataset. In DAI, these two activities are grouped into a single class and are associated with a highly specialized attention head (head 5 in layer 22) which was not significantly relevant for any other class. This exclusivity is likely due to the distinctive motion patterns involved in bringing objects (e.g., food or a bottle) toward the mouth, making it a strong visual signal for this specific class.

In contrast, the Drive&Act dataset separates this activity into two distinct classes: “eating” and “drinking”. Despite this difference in taxonomy, head 5 in layer 22, trained solely on the combined class in DAI, exhibits robust generalization when applied to Drive&Act. As illustrated in Fig. 5.6, this head continues to capture meaningful visual cues in both separated classes. Specifically, it focuses on the bottle or the driver’s fingers grasping the bottle in the “drinking” samples, and similarly attends to the hand-to-mouth motion or fingers holding food in the “eating” samples.

As a final example, we analyze a specialized attention head (head 15 in layer 22) which, in the DAI dataset, was found to be exclusively activated for the class “Reach-objects”.

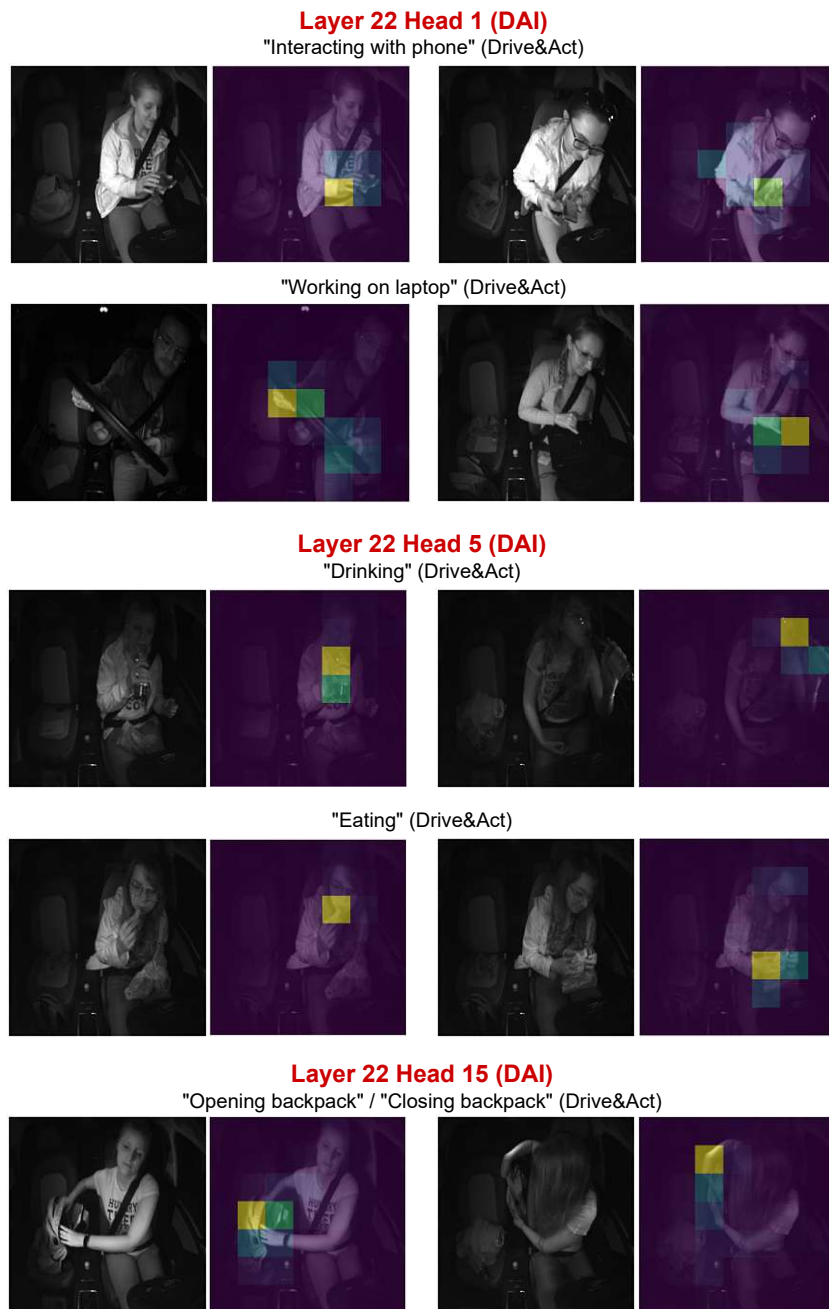


Figure 5.6: Attention maps from specialized heads of the Video Swin model trained on the DAI dataset, demonstrating generalization to Drive&Act. For each example, the corresponding specialized head (identified from DAI) is indicated, along with the class label from the Drive&Act dataset for the visualized sample.

Initially, we attempted to map this class to the “fetching an object” action in Drive&Act. However, this mapping proved suboptimal due to a mismatch in the scope of the actions across datasets. In DAI, the “Reach-objects” class was predominantly associated with a narrow set of objects (e.g., backpacks, as shown in the second row of Video Swin examples in Fig. 5.2). In contrast, the “fetching an object” class in Drive&Act expands the range of objects being interacted with (e.g. seatbelts, jackets, food, multimedia displays, or laptops) because it treats fetching for these objects as distinct actions from the subsequent interactions – for example, reaching for a laptop is different from working on it. Conversely, DAI combines these sequences into a single “Working-on-laptop” action.

Upon further inspection, we refined the mapping by aligning DAI’s “Reach-objects” with the Drive&Act actions “opening backpack” and “closing backpack”. This updated mapping proved to be much more successful: as illustrated in the last row of Fig. 5.6, head 15 in layer 22 accurately localized attention around the backpack and the driver’s hand during the opening and closing motions.

All of these results demonstrate that specialized attention heads trained on the DAI dataset can meaningfully generalize to semantically related actions in the Drive&Act dataset, even without fine-tuning, which shows the potential for cross-dataset transfer in driver action recognition.

Specialized Head Generalization on UCF101

Extending our investigation of generalization, we acknowledge that transferring from a driver-specific dataset to a broad, general action recognition dataset is inherently challenging. However, as discussed earlier, head 4 in layer 22 of Video Swin, trained on the DAI dataset, consistently tracks the driver’s head across all test samples (see right part of Fig. 5.4). Motivated by this, we examine the attention patterns of this specialized head to evaluate its ability to generalize to the diverse actions in the UCF101 dataset. This evaluation is conducted in the same zero-shot, out-of-the-box manner as before.

As illustrated in Fig. 5.7, the attention patterns of this head exhibit remarkable generalization to the UCF101 dataset. For instance, within UCF101, layer 22, head 7 is identified as the most significant for the “PlayingViolin” class, focusing specifically on the violins (see lower part, second column in Fig. 5.7). However, when we apply the model trained on DAI, using the frozen weights of layer 22, head 4, which is specialized in concentrating on the person’s head, we observe successful generalization in the attention patterns (see lower part, last column in Fig. 5.7).

This result highlights the robustness of specialized attention heads in capturing semantically meaningful features that extend beyond their original training domain, suggesting that such heads can serve as transferable components for understanding key visual cues across diverse action recognition datasets.

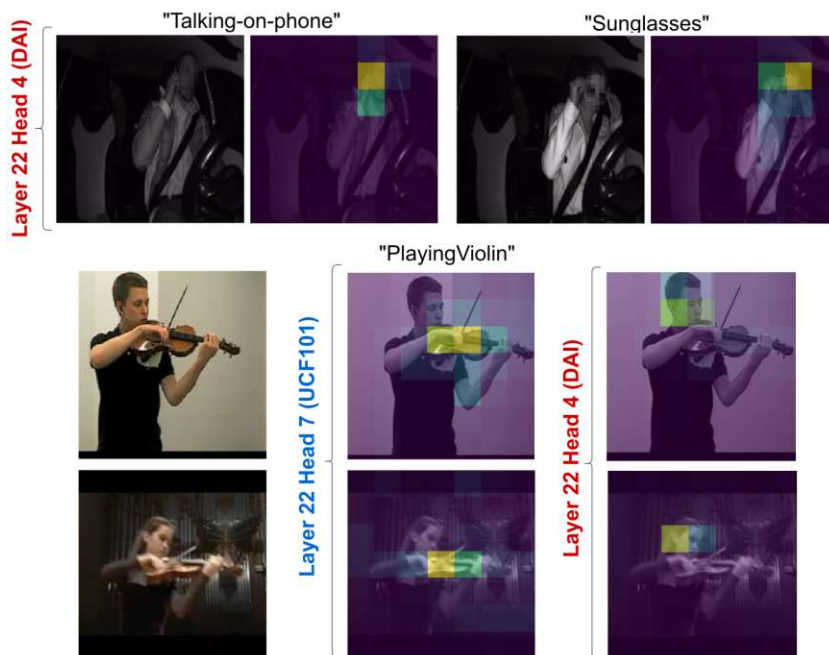


Figure 5.7: Attention maps of a specialized head using Video Swin that generalizes across datasets. In the upper part (first row), a specialized head on DAI consistently focuses on the driver’s head, regardless of the video class. Additionally, in the lower part, second column, we show a head trained on UCF101 to detect violins. For the same violin samples, the DAI-trained head successfully generalizes by detecting the person’s head, as shown in the lower part, last column.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Conclusions

6.1 Discussion

In this thesis, we focused on enhancing driver action recognition systems by investigating the self-attention mechanisms of state-of-the-art video transformers. Understanding these mechanisms is crucial for developing driver assistance technologies that rely on accurate action recognition. Our qualitative analysis highlighted the effectiveness of the applied head importance metrics, demonstrating that identifying specific attention heads within these models provides valuable insights into the visual cues, such as key regions, movements and moments in time, that influence the model's predictions both spatially and temporally. Our findings highlight how individual attention heads capture distinct patterns at varying depths of the transformer, allowing the model to effectively learn relevant spatio-temporal features for detecting driver actions. Notably, specialized heads, particularly those located in the deeper layers, demonstrate unique functions tailored to specific driver behaviors, capturing fine-grained and semantically meaningful visual cues. Specifically, some of these heads not only excel in recognizing patterns within the in-domain DAI dataset but also show impressive generalization capabilities across in-domain datasets such as Drive&Act and even out-of-domain like UCF101 without additional fine-tuning.

Furthermore, to address the computational challenges inherent in deploying video transformers in real-time driver assistance systems, we proposed a layer pruning technique guided by head importance metrics. Our quantitative results demonstrated that selectively pruning low-relevance layers significantly reduces computational complexity while maintaining strong performance across both the DAI and UCF101 datasets, as well as three state-of-the-art transformer architectures. For instance, on our DAI dataset, we achieved a 23.5% reduction in FLOPs when compressing the Video Swin model, with less than a 1% drop in Top-1 accuracy. Similarly, on the UCF101 dataset, we obtained nearly a 26% reduction in FLOPs for the MViT model, with a loss of less than 3% in

Top-1 accuracy. In contrast, pruning high-relevance layers led to severe performance degradation, highlighting the critical role these layers play in maintaining model accuracy. For example, removing just a single high-relevance layer resulted in an accuracy drop of up to 30% for the Video Swin Transformer on the DAI dataset, underscoring the importance of preserving these essential layers. This contrast confirms the robustness of our importance metrics and the effectiveness of our pruning strategy for optimizing video transformers for action recognition tasks.

Overall, the insights gained from this research contribute to a deeper understanding of how video transformers operate and how their efficiency can be improved. This work lays the groundwork for future applications of transformers in diverse domains, particularly in areas where real-time action recognition is critical, such as in assisted driving systems.

6.2 Future work

While this thesis has provided valuable insights into the interpretability of self-attention mechanisms of video transformers and introduced a method to enhance their efficiency, several avenues for future research remain.

Given the observable prunability of transformers, future research could explore optimizing transformer architectures based on data attributes. This could involve dynamically adjusting layers, channels, or heads to align with the complexity and size of the dataset. For instance, simpler datasets could benefit from a streamlined architecture with fewer layers and heads, while more complex datasets might require a richer configuration to capture intricate patterns effectively. This adaptability could lead to significant improvements in resource efficiency, as models would not be over-engineered for simpler tasks, thus reducing computational costs without sacrificing performance.

While our current work focused primarily on action recognition, another promising avenue for future investigation could explore the application of our interpretability and pruning techniques to other video-based tasks. Areas such as video captioning, temporal action localization, or video question answering could benefit from the insights and methodologies developed in this study. By adapting our approaches to these diverse tasks, we could gain a more comprehensive understanding of how video transformers process and interpret temporal information across various video understanding challenges. This broader application could lead to more general principles for designing and optimizing video transformer architectures across a wide range of computer vision tasks.

Overview of Generative AI Tools Used

I hereby declare that I used the AI tool ChatGPT¹ solely for rephrasing purposes in the writing process of this thesis. The core content and ideas presented are entirely my own; ChatGPT was used only to suggest alternative wordings or phrasing options (e.g., to find suitable connectors or improve sentence flow). I carefully reviewed all suggestions and adapted them to my own writing style and judgment.

Additionally, I employed the AI-based spelling and grammar checker integrated within Overleaf² to correct typographical and spelling errors.

Examples of Prompts Used with AI Tools

Below are examples of prompts I used with ChatGPT for rephrasing and phrasing suggestions during the writing of this thesis:

- **Rephrasing a sentence:**
“Can you suggest alternative ways to write this sentence: ‘The results indicate a significant improvement in performance.’?”
- **Improving sentence flow:**
“How can I rewrite this paragraph to improve readability and flow?”
- **Making text more formal:**
“Can you help me make this sentence sound more formal: ‘We found out that the method works better.’?”
- **Suggesting synonyms:**
“What are some synonyms for the word ‘significant’ that fit academic writing?”
- **Improving academic tone:**
“How can I rephrase this sentence to sound more academic and precise?”

¹<https://chatgpt.com/>

²https://www.overleaf.com/learn/how-to/AI_Assist



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Figures

2.1	ViT model overview. Figure taken from [DBK ⁺ 21].	6
2.2	Visualization of the TimeSformer attention mechanism. On the right, a representation of the attention mechanism in TimeSformer is depicted, illustrating that this block is repeated 12 times (the model has 12 equivalent layers). On the left, the attention visualization is applied to a video clip. Figure adapted from [BWT21].	8
2.3	Illustration of token sizes and attention window (AttnW) sizes across the four stages (S1-S4) of Video Swin. The top shows token sizes at each stage, while the bottom displays attention window sizes, including the number of windows for an input of 16 frames ($T = 16$) at a resolution of 224×224 pixels, with batch size B and channel dimension C . For example, in Stage 2 (S2), the token size is $2 \times 8 \times 8$ pixels, resulting in a window size of $16 \times 56 \times 56$ pixels (since the window size W_t is consistently $8 \times 7 \times 7$ tokens). By Stage 4, the token size reaches $2 \times 32 \times 32$ pixels, with the attention window covering the entire video sequence ($16 \times 224 \times 224$ pixels), allowing for global attention computation.	10
2.4	Overview of the pooling attention mechanism in MViT within a general layer. This process, which includes pooling and self-attention computation, is consistent across all layers of the architecture. The main differences lie in the strides applied to the key, query, and value vectors (S_K , S_Q , and S_V), as well as the input resolution $t \times h \times w$, which are progressively reduced as the model depth increases.	13
3.1	Sample frames for 4 action classes of Kinetics-400. Figure reproduced from [CZ17].	19
3.2	Sample frames of action classes from the UCF101 dataset.	20
3.3	Sample frames for 8 action classes of Drive&Act.	21
3.4	Sample frames for 8 action classes of our proprietary DAI dataset.	21
		59

4.1	Illustration of the process for computing attention weights and visualizing attention maps in TimeSformer. This procedure is consistent across all layers of the architecture. The green and red colors at the top represent temporal and spatial attention, respectively, ensuring consistency with Fig. 2.2. The CLS token refers to the <i>[class]</i> token (in orange), which is defined in Section 2.1. The colors used to represent attention weights are approximated from the viridis color palette, which will be employed in our qualitative experiments. In this palette, yellow indicates high attention, while dark blue and purple signify low attention.	25
4.2	The 13 possible attention patterns between image patches presented in the (top) image space, (middle) two-axes, and (bottom) heatmap visualization. Each pattern is a mix of one/multiple of the four basic patterns: diagonal (A-K), horizontal (J, K), vertical (L), and block (M). The first two are <i>content-agnostic</i> and often appear in lower-layer heads; the latter two are <i>content-relevant</i> and often occur in higher-layer heads. Figure taken from [LWD ⁺ 23].	27
4.3	Illustration of the process for computing attention weights and visualizing attention maps in Stage 2 of the Video Swin Transformer. The same procedure applies to all other stages, with differences only in token size and the number of attention windows.	28
4.4	Illustration of the process for computing attention weights and visualizing attention maps in MViT. This procedure is consistent across all MViT layers, with variations in the stride used for pooling K resulting in different dimensions (t'', h'', w'') at each layer. Consequently, the interpolation factor $(\frac{T}{t''), \frac{H}{h''), \frac{W}{w''})$ must be adjusted accordingly.	30
4.5	Pipeline of our proposed layer pruning method. We initially train a video transformer and then evaluate each layer by removing it and calculating its relevance score. After obtaining all relevance scores, we sort them and prune the model to the desired depth by selecting the top-k layers based on the relevance criteria.	34
5.1	Boxplots of layer relevance scores for the TimeSformer, Video Swin Transformer, and MViT architectures on both DAI and UCF101 datasets. These scores are calculated on the training sets to guide the selection of layers for further pruning. To enhance visualization, the vertical axis range has been adjusted, excluding some outliers.	40

5.2	Visualization of the attention maps produced by the highest and lowest relevance heads for TimeSformer, Video Swin and MViT. Red boxes indicate the lowest relevance heads within the corresponding layers, while green boxes indicate the highest relevance heads within the corresponding layers. Early layers refer to the first layer for TimeSformer, and first two layers for both Video Swin and MViT, while deep layers correspond to the last one in TimeSformer, and last two layers in Video Swin and MViT. Intermediate layers encompass those in between. Colormap: viridis (dark blue: low attention, bright yellow: high attention).	43
5.3	Attention maps of specialized relevant heads using Video Swin for various test samples on the DAI dataset, revealing that some classes share a specialized head.	44
5.4	Attention maps for various DAI samples across a temporal sequence. On the first two examples, specialized heads that focus on specific objects (e.g., phone) demonstrate high temporal focus by highlighting only one or two frames. On the bottom, head 4 in layer 22 consistently tracks the driver’s head across multiple frames. Although the full video sequence in the experiments contains 16 frames, we display only the 4 consecutive frames where the main action occurs and attention is most concentrated.	47
5.5	Attention maps for two DAI samples using TimeSformer. The original video frames (first row) are shown alongside the spatial attention from the most relevant head in the final layer (second row), the temporal attention from the most important temporal head (third row), and the combined spatial-temporal attention through element-wise multiplication (fourth row). The attention rollout is provided for comparison (last row), with irrelevant areas highlighted by this method (e.g., brightness bias) circled in red.	49
5.6	Attention maps from specialized heads of the Video Swin model trained on the DAI dataset, demonstrating generalization to Drive&Act. For each example, the corresponding specialized head (identified from DAI) is indicated, along with the class label from the Drive&Act dataset for the visualized sample.	51
5.7	Attention maps of a specialized head using Video Swin that generalizes across datasets. In the upper part (first row), a specialized head on DAI consistently focuses on the driver’s head, regardless of the video class. Additionally, in the lower part, second column, we show a head trained on UCF101 to detect violins. For the same violin samples, the DAI-trained head successfully generalizes by detecting the person’s head, as shown in the lower part, last column.	53



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Tables

2.1	Overview of model architectures. For each model – (a) TimeSformer, (b) Video Swin, and (c) MViT– we detail the different stages (or blocks). Each subtable presents information on the number of layers using the MHSA mechanism, the token sizes at each layer, the dimensions of the attention matrix, and the output dimensions of the MHSA for that block.	14
3.1	Overview of relevant datasets for action recognition from video data. . . .	18
5.1	Main results on the test set for DAI and UCF101 datasets. Top-1 accuracy, parameter counts and FLOPs are presented for three video transformers. The model with 0 layers pruned serves as the unpruned baseline, while the remaining models indicate the number of pruned layers and their relevance levels (Lowest or Highest).	37
5.2	Layers dropped in a single experimental run. The number of layers pruned in each case is displayed, along with the indices of the pruned layers in both low-relevance and high-relevance pruning modes, based on the computed relevance score. Consistent results are observed across multiple runs. . . .	39
5.3	Comparison of Top-1 accuracy outcomes when pruning layers based on different importance metrics (JSD, Δ Prob, and R score) across various datasets and architectures.	41
5.4	Most relevant specialized head by class in the DAI dataset based on the relevance score computed for all training samples among heads on the last two layers of the Video Swin (layers 22 and 23). All identified specialized heads are located in layer 22, making it more relevant.	45



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acronyms

- CNN** Convolutional Neural Network. 5
- MHSA** multi-head self-attention. 6, 8, 9, 11, 14, 23, 29, 37, 63
- MLP** Multilayer Perceptron. 7, 9
- MViT** Multiscale Vision Transformer. 5, 11, 13, 23, 59
- NLP** Natural Language Processing. 5
- RNN** Recurrent Neural Network. 15
- TimeSformer** Time-Space Transformer. 5, 8, 9, 23, 24
- ViT** Vision Transformer. 5, 6, 8, 9, 59



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

- [AA23] Ayoub Benali Amjoud and Mustapha Amrouch. Object detection using deep learning, CNNs and vision transformers: A review. In *IEEE Access*, volume 11, pages 35479–35516, 2023.
- [ADH⁺21] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lucic, and Cordelia Schmid. ViViT: A video vision transformer. In *International Conference on Computer Vision (ICCV)*, pages 6816–6826, 2021.
- [AZ20] Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4190–4197, 2020.
- [BGT18] Bhakti Baheti, Suhas S. Gajre, and Sanjay N. Talbar. Detection of distracted driver using convolutional neural network. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1145–11456, 2018.
- [BWT21] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *International Conference on Machine Learning (ICML)*, pages 813–824, 2021.
- [CGW20] Hila Chefer, Shir Gur, and Lior Wolf. Transformer interpretability beyond attention visualization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 782–791, 2020.
- [CMS⁺20] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*, pages 213–229, 2020.
- [CTM⁺21] Mathilde Caron, Hugo Touvron, Ishan Misra, Herv'e J'egou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *International Conference on Computer Vision (ICCV)*, pages 9630–9640, 2021.

- [CZ17] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6299–6308, 2017.
- [CZS24] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pages 1–20, 2024.
- [DBK⁺21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, pages 1–21, 2021.
- [DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, volume 1, pages 4171–4186, 2019.
- [DWB20] Joseph F DeRose, Jiayao Wang, and Matthew Berger. Attention flows: Analyzing and comparing attention mechanisms in language models. In *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, volume 27, pages 1160–1170, 2020.
- [FGJ20] Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations (ICLR)*, pages 1–16, 2020.
- [FXM⁺21] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *International Conference on Computer Vision (ICCV)*, pages 6804–6815, 2021.
- [GCJT21] Romain Guesdon, Carlos Crispim-Junior, and Laure Tougne. Dripe: A dataset for human pose estimation in real-world driving settings. In *International Conference on Computer Vision Workshops (ICCVW)*, pages 2865–2874, 2021.
- [HDWX21] Yaru Hao, Li Dong, Furu Wei, and Ke Xu. Self-attention attribution: Interpreting information interactions inside transformer. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, pages 12963–12971, 2021.

- [HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [JBKAM20] Imen Jegham, Anouar Ben Khalifa, Ihsen Alouani, and Mohamed Mahjoub. A novel public dataset for multimodal multiview and multispectral driver distraction analysis: 3mdad. *Signal Processing Image Communication*, 88:115960, 2020.
- [KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, pages 1–15, 2015.
- [KJG⁺11] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A large video database for human motion recognition. In *International Conference on Computer Vision (ICCV)*, pages 2556–2563, 2011.
- [KKK⁺19] Byungju Kim, Hyunwoo Kim, Kyungsu Kim, Sungjin Kim, and Junmo Kim. Learning not to learn: Training deep neural networks with biased data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [KSS⁺24] Thomas Kidu, Yongjun Song, Kwang-Won Seo, Sunyong Lee, and Taejoon Park. An intelligent real-time driver activity recognition system using spatio-temporal features. *Applied Sciences*, 14(17):1–25, 2024.
- [KWD⁺21] Kyungmin Kim, Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Zhicheng Yan, Peter Vajda, and Seon Kim. Rethinking the self-attention in vision transformers. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3065–3069, 2021.
- [LGB⁺19] Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. Linguistic knowledge and transferability of contextual representations. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, volume 1, pages 1073–1094, 2019.
- [LLC⁺21] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *International Conference on Computer Vision (ICCV)*, pages 9992–10002, 2021.
- [LLZ⁺19] Peng Li, Meiqi Lu, Zhiwei Zhang, Donghui Shan, and Yang Yang. A novel spatial-temporal graph for skeleton-based driver action recognition. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3243–3248, 2019.

- [LNC⁺21] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3192–3201, 2021.
- [LWD⁺23] Yiran Li, Junpeng Wang, Xin Dai, Liang Wang, Chin-Chia Michael Yeh, Yan Zheng, Wei Zhang, and Kwan-Liu Ma. How does attention work in vision transformers? a visual analytics attempt. In *IEEE Transactions on Visualization and Computer Graphics*, volume 29, pages 2888–2900, 2023.
- [LWF⁺21] Yanghao Li, Chaoxia Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. MViTv2: Improved multiscale vision transformers for classification and detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4794–4804, 2021.
- [LXW⁺21] Raymond Li, Wen Xiao, Lanjun Wang, Hyeju Jang, and Giuseppe Carenini. T3-Vis: visual analytic for training and fine-tuning transformers in nlp. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 220–230, 2021.
- [LZK⁺21] Liyang Liu, Shilong Zhang, Zhanghui Kuang, Aojun Zhou, Jing-Hao Xue, Xinjiang Wang, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Group fisher pruning for practical network compression. In *International Conference on Machine Learning (ICML)*, volume 139, pages 7021–7032, 2021.
- [LZZ⁺22] Junwei Liang, He Zhu, Enwei Zhang, Jun Zhang, and Tencent Youtu Lab. Stargazer: A transformer-based driver action detection system for intelligent transportation. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3159–3166, 2022.
- [MLN19] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 1–11, 2019.
- [MPPP97] M.L. Menéndez, J.A. Pardo, L. Pardo, and M.C. Pardo. The jensen-shannon divergence. *Journal of the Franklin Institute*, 334:307–318, 1997.
- [MRH⁺19] Manuel Martin, Alina Roitberg, Monica Haurilet, Matthias Horne, Simon Reiß, Michael Voit, and Rainer Stiefelhagen. Drive&act: A multi-modal dataset for fine-grained driver behavior recognition in autonomous vehicles. In *International Conference on Computer Vision (ICCV)*, pages 2801–2810, 2019.
- [Nie19] Frank Nielsen. On the jensen–shannon symmetrization of distances relying on abstract means. *Entropy*, 21(5):485, 2019.

- [OKCn⁺20] Juan Diego Ortega, Neslihan Kose, Paola Cañas, Min-An Chao, Alexander Unnervik, Marcos Nieto, Oihana Otaegui, and Luis Salgado. DMD: A large-scale multi-modal driver monitoring dataset for attention and alertness analysis. In *European Conference on Computer Vision Workshops (ECCVW)*, page 387–405, 2020.
- [PNJ⁺19] Cheonbok Park, Inyoup Na, Yongjang Jo, Sungbok Shin, Jae Won Yoo, Bum Chul Kwon, Jian Zhao, Hyungjong Noh, Yeonsoo Lee, and Jaegul Choo. SANVis: Visual analytics for understanding self-attention networks. In *IEEE Visualization Conference (VIS)*, pages 146–150, 2019.
- [RHGS17] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 39(6):1137–1149, 2017.
- [SCD⁺17] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *International Conference on Computer Vision (ICCV)*, pages 618–626, 2017.
- [SLNW16] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1010–1019, 2016.
- [SZS12] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *Center for Research in Computer Vision, University of Central Florida*, pages 1–7, 2012.
- [TBF⁺15] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015.
- [VSP⁺17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 1–11, 2017.
- [VTM⁺19] Elena Voita, David Talbot, F. Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 5797–5808, 2019.
- [XLW⁺19] Yang Xing, Chen Lv, Huaji Wang, Dongpu Cao, Efstathios Velenis, and Fei-Yue Wang. Driver activity recognition for intelligent vehicles: A deep

learning approach. *IEEE Transactions on Vehicular Technology*, 68(6):5379–5390, 2019.

- [XTL⁺18] Yang Xing, Jianlin Tang, Hong Liu, Chen Lv, Dongpu Cao, Efstathios Velenis, and Fei-Yue Wang. End-to-end driving activities and secondary tasks recognition using deep convolutional neural network and transfer learning. In *IEEE Intelligent Vehicles Symposium*, pages 1626–1631, 2018.
- [YHW⁺22] Fang Yu, Kun Huang, Meng Wang, Yuan Cheng, Wei Chu, and Li Cui. Width & depth pruning for vision transformers. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 36, pages 3143–3151, 2022.
- [YHX⁺23] Ding kang Yang, Shuai Huang, Zhi Xu, Zhenpeng Li, Shunli Wang, Mingcheng Li, Yuzheng Wang, Yang Liu, Kun Yang, Zhaoyu Chen, Yan Wang, Jing Liu, Peixuan Zhang, Peng Zhai, and Lihua Zhang. Aide: A vision-driven multi-view, multi-modal, multi-tasking dataset for assistive driving perception. In *International Conference on Computer Vision (ICCV)*, pages 20402–20413, 2023.
- [YYY⁺19] Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.
- [ZQJM23] Wei Zhou, Yinlong Qian, Zequn Jie, and Lin Ma. Multi view action recognition for distracted driver behavior localization. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 5375–5380, 2023.
- [ZS18] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 1–11, 2018.