Universidad Internacional de la Rioja (UNIR)

Escuela Superior de Ingeniería y Tecnología

Máster en Inteligencia Artificial

Pose-based Gesture Recognition in Autonomous Vehicles using Speech Recognition Networks

Trabajo Fin de Estudios

Autor: Pablo Pardo Decimavilla

Tutores: Jose María Escalante y Luis M. Bergasa Pascual

Ciudad: Alcalá de Henares

Fecha: 10 de julio de 2024

Contents

List of Figures

List of Tables

Resumen

Reconocer distracciones en la carretera es esencial para reducir accidentes de tráfico. Las redes basadas en video suelen usarse, pero tienen un alto costo computacional y son vulnerables a cambios de perspectiva. Este artículo propone un enfoque novedoso para clasificar acciones del conductor basado en poses, utilizando redes de reconocimiento de voz, que son más ligeras y resistentes a los cambios de perspectiva. La similitud en la codificación entre datos de audio y poses se aprovecha representando poses como puntos clave a lo largo del tiempo. Nuestra arquitectura se basa en Squeezeformer (Kim et al., 2022), una red de reconocimiento de voz eficiente y basada en atención. Implementamos técnicas de aumento de datos para mejorar la generalización. Los experimentos con el conjunto de datos Drive&Act muestran un rendimiento superior frente a métodos de última generación. Además, desarrollamos un conjunto de datos real para ajustar el modelo, permitiendo su uso en entornos personalizados. Los resultados destacan la eficacia y robustez de estas redes en la clasificación de acciones basadas en poses.

Palabras Clave: Conducción Autónoma, Sistemas Avanzados de Advertencia de Distracción del Conductor, Aprendizaje Profundo, Transformers.

Abstract

Recognizing distractions on the road is crucial to reduce traffic accidents. Video-based networks are typically used, but are limited by their computational cost and are vulnerable to viewpoint changes. In this paper, we propose a novel approach for pose-based driver action classification using speech recognition networks, which is lighter and more viewpoint invariant that video-based one. We leverage the similarity in the encoding of information between audio and pose data, representing poses as key points over time. Our architecture is based on Squeezeformer (Kim et al., 2022), an efficient attention-based speech recognition network. We introduce a selection of data augmentation techniques to enhance generalization. Experiments on the Drive&Act dataset demonstrate superior performance compared to state-of-the-art methods. Additionally, we have developed a real dataset to finetune the model, enabling deployment in a custom environments. Our results highlight the effectiveness and robustness of speech recognition networks in pose-based action classification.

Keywords: Autonomous Driving, Advanced Driver Distraction Warning Systems, Deep Learning, Transformers.

1. Introduction

Every day in the United States, nine people lose their lives in accidents that are reported to involve a distracted driver (for Disease Control & Prevention, n.d.). The organisation states that "anything that takes your attention away from driving can be a distraction". In Europe, near 25% of all crashes are due to lack of attention while driving(for Transport, 2022). Actions such as talking on the phone, eating or operating the multimedia screen distract the driver and slow down his reaction time. The implementation of systems that identify and notify these actions has the potential to reduce the occurrence of accidents. Furthermore, with the advent of partially autonomous vehicles on the market, it is imperative to ascertain the driver's status in order to facilitate a safe transition from autonomous to manual control. The objective of this research is to classify actions that may divert the driver's attention. This will enable an alert system for manual driving and an secure transition between modes in autonomous vehicles.

Driver action classification networks are derived from Human Action Recognition (HAR) architectures. Both can be broadly classified into those based on RGB images and those based on human pose (Sun et al., 2022). Other references in the literature combine both modalities (Guo et al., 2023). Image-based networks are able to capture fine details of the video and extract relevant features from each class (Feichtenhofer et al., 2019) (Liu et al., 2021). Pose-based architectures extract the 2D or 3D pose of the driver for further processing with a neural network (Holzbock et al., 2022). These last are computationally lighter than vision-based ones and they are agnostic to viewpoint, background or lighting changes, resulting in interesting networks for automation purposes (Martin et al., 2023).

Often, drawing inspiration from other fields of artificial intelligence can yield good results. As is the case with Transformers (Vaswani et al., 2017) originally designed for text processing and successfully used for image processing. This work explores the use of speech recognition networks for pose-based driver distractions classification. The relationship between these two concepts arises from the similarity in the encoding of information. As seen in the Figure 1.1, audio is encoded in a spectrogram as a number of features (frequencies) sampled over time, while the pose can be represented in the same way, with those features being keypoints of the driver. Our interest lies in audio recognition encoders that can identify temporal patterns and relationships through complex sequences.

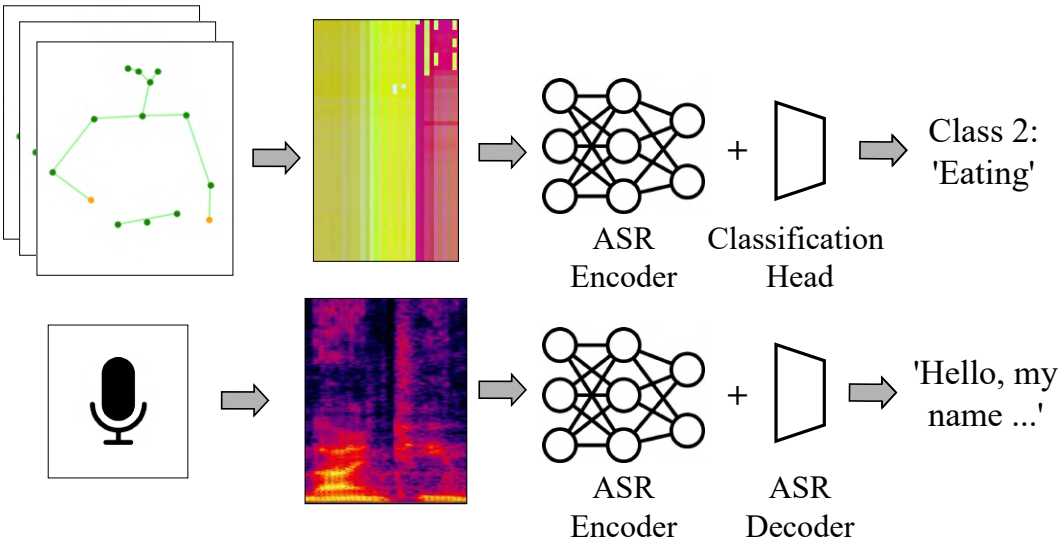

Figure 1.1: Graphical abstract.

Our architecture comprises three sections. First, a stem layer adjusts the size of the input to be processed by the encoder. We use a Transformer-based encoder inspired by Squeezeformer (Kim et al., 2022), a novel and efficient architecture imported from the audio recognition domain. In contrast to speech networks, we aim to classify actions, therefore we use a classification head as decoder.

We propose a selection of data augmentation techniques to enhance the network's generalisation and avoid overfitting during training. Three sets of techniques are proposed, one focusing on the temporal dimension, one on the spatial dimension and one combining both. We train on Drive&Act (Martin et al., 2019), a challenging dataset for driver distraction classification. The actions are typical distractions in the field of both manual and autonomous driving. Our experiments show superior performance compared to other state-of-the-art techniques. Furthermore, three ablation studies were conducted to understand the impact of data augmentation techniques, the implementation of objects to the network and finally the robustness to viewpoint changes.

Finally, we proceeded to implement our model in a real-world scenario. This involved the electric autonomous car developed by the Robesafe research group[1]. A camera will be installed under the centre rear-view mirror of the vehicle to monitor the driver's activities. A set of data is collected in order to test the efficacy of the network in our custom use

[1] `http://www.robesafe.es/`

case. With this data, a fine-tuning process is employed to measure the effectiveness of the network in the given environment.

The main contributions of this TFM are:

- Development of an architecture based on speech recognition networks for pose-based driver activity classification.
- Development of data augmentation techniques focusing on spatial and temporal features. Through an ablation study, we demonstrate the contribution of each group of techniques.
- Validation of the network in the Drive&Act dataset over-performing the state-of-the-art.
- Inclusion experiment of information about objects of interest in the scene, which resulted in significant improvements.
- Robustness study of our network to viewpoint changes compared to video-based networks.
- The application of the network in a real autonomous vehicle and the subsequent study to assess the domain adaptation of the network to new environments.

In summary, this paper proposes an architecture based on speech recognition networks that, together with a selection of data augmentations, provides results that outperform the state of the art over an open-source dataset and is able to be adapted in a real-world use case. Our code is publicly available[2].

[2] `https://github.com/pablopardod/dyalyt`

2. Theorical Background

2.1 Multi-Head Attention Module

2.1.1 Attention Mechanism

The attention mechanism is a key innovation in deep learning, particularly for sequence-based tasks like natural language processing. At its core, the attention mechanism allows the model to focus on different parts of the input sequence when producing each element of the output sequence. This is particularly useful in tasks where the context of the sequence is important, such as automatic speech recognition.

Mathematically, the attention mechanism can be described as a process of computing a weighted sum of input values, where the weights are determined by a compatibility function that compares the query with each key. Given an input sequence of vectors $X = [x_1, x_2, ..., x_n]$, the attention mechanism operates as follows:

1. **Compute Query, Key, and Value matrices**: These are linear transformations of the input:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

where W^Q, W^K, and W^V are learned weight matrices.

2. **Compute attention scores**: The attention scores are computed as the dot product of the query and key vectors, scaled by the square root of the dimension of the key vectors:

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

where d_k is the dimension of the key vectors.

3. **Apply softmax**: The softmax function is applied to the attention scores to obtain the attention weights, which are then used to compute the weighted sum of the value vectors.

2.1.2 Multi-Head Attention

Multi-head attention extends the basic attention mechanism by allowing the model to jointly attend to information from different representation subspaces at different positions. This is achieved by using multiple attention heads, each with its own set of parameters.

The multi-head attention mechanism can be described as follows and is illustrated in Figure 2.1:

1. **Linear projections**: For each head i, the input sequence X is linearly projected to a query, key, and value matrix:

$$Q_i = XW_i^Q, \quad K_i = XW_i^K, \quad V_i = XW_i^V$$

where W_i^Q, W_i^K, and W_i^V are the weight matrices for head i.

2. **Scaled dot-product attention**: Each head performs the scaled dot-product attention function independently:

$$\text{head}_i = \text{Attention}(Q_i, K_i, V_i)$$

3. **Concatenation and linear projection**: The outputs of all heads are concatenated and projected through a final linear transformation:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, ..., \text{head}_h)W^O$$

where W^O is a learned projection matrix.

By using multiple heads, the model can capture a richer set of dependencies and interactions in the data, improving its ability to understand complex patterns. The multi-head attention mechanism is a fundamental building block of the Transformer architecture, which has achieved state-of-the-art performance on numerous tasks.

2.2 Feed Forward Module

The Feed Forward Module is a crucial component in Transformer architectures, following the attention mechanisms to further process the information. This module is applied independently to each position in the sequence, allowing for non-linear transformations of the input. Its purpose is to introduce more complex transformations and interactions between the components of the input sequence.

2.2.1 Architecture

The Feed Forward Module consists of two linear transformations with a ReLU activation in between. Mathematically, it can be represented as follows:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

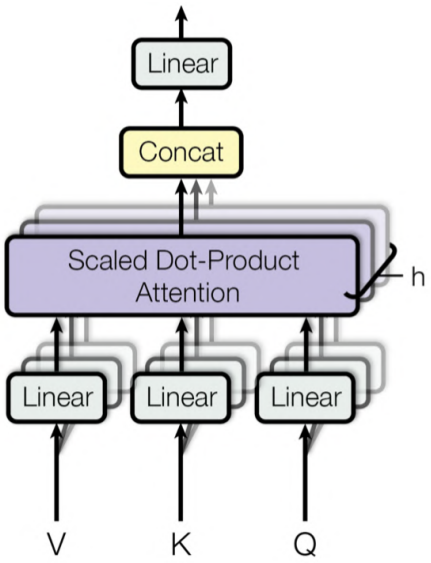

Figure 2.1: Illustration of the Multi-Head Attention mechanism.
Source: https://paperswithcode.com

where: - x is the input vector. - W_1 and W_2 are weight matrices. - b_1 and b_2 are bias vectors. - $\max(0, x)$ denotes the ReLU activation function.

2.2.2 Detailed Explanation

The Feed Forward Module operates in the following steps:

1. **Linear Transformation**: The input vector x is first linearly transformed using the weight matrix W_1 and bias b_1. This transformation increases the dimensionality of the input vector to a higher-dimensional space, typically referred to as the hidden layer.

$$h = xW_1 + b_1$$

2. **ReLU Activation**: The output of the linear transformation is then passed through a ReLU activation function. The ReLU function introduces non-linearity to the model, allowing it to learn more complex functions. Mathematically, the ReLU activation is defined as:

$$\text{ReLU}(h) = \max(0, h)$$

3. **Second Linear Transformation**: The activated output is then linearly transformed again using the weight matrix W_2 and bias b_2. This transformation projects the

high-dimensional data back to the original input dimension.

$$y = \text{ReLU}(h)W_2 + b_2$$

4. **Residual Connection and Layer Normalization**: In practice, the output of the Feed Forward Module is often added to the input via a residual connection, followed by layer normalization to stabilize and accelerate the training process.

$$\text{Output} = \text{LayerNorm}(x + y)$$

The Feed Forward Module's combination of linear transformations and non-linear activation enables the model to learn and represent complex patterns in the data, complementing the attention mechanism by providing additional capacity for the model to process and understand the input sequence.

2.3 Convolutional Block

A typical convolutional layer applies a set of filters to the input data, where each filter slides over the input data to produce feature maps. These feature maps are then passed through an activation function to introduce non-linearity, enabling the model to learn complex patterns.

2.3.1 1D Convolutional Layers

In the context of sequential data, such as time-series or textual data, 1D convolutions are commonly used. A 1D convolutional layer slides filters over one dimension of the input data, typically the temporal or sequential dimension. This allows the model to capture temporal dependencies and patterns effectively.

Mathematically, the output of a 1D convolutional layer can be expressed as follows:

$$y(t) = \sigma\left(\sum_{i=0}^{k-1} x(t+i) \cdot w_i + b\right)$$

where:

- $y(t)$ is the output at position t
- $x(t)$ is the input at position t
- w_i are the weights of the filter

- b is the bias term
- k is the size of the filter
- σ is the activation function

2.4 Activation Functions

2.4.1 Introduction

Activation functions play a crucial role in neural networks by introducing non-linearity into the model, allowing it to learn complex patterns. In recent years, newer activation functions have been proposed, such as Swish and GLU (Gated Linear Unit), which have shown promising results in various deep learning tasks.

2.4.2 Swish Activation Function

The Swish activation function, proposed by researchers at Google, is defined as:

$$\text{Swish}(x) = x \cdot \sigma(x)$$

where $\sigma(x)$ is the sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Swish is a smooth, non-monotonic function that has been shown to outperform ReLU on deeper models. One of the key advantages of Swish is that it allows small negative values instead of truncating them to zero, which can lead to better gradient flow and improved performance.

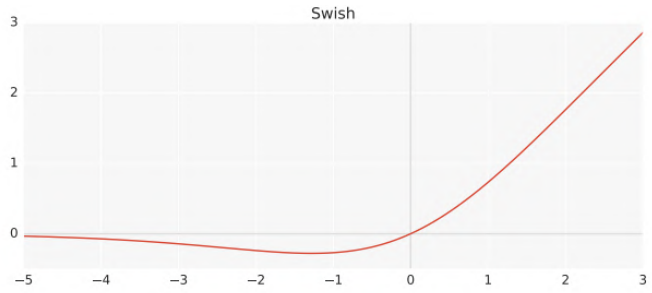

Figure 2.2: Swish Activation Function

Source: https://medium.com

2.4.3 Gated Linear Unit (GLU)

The Gated Linear Unit (GLU) is another advanced activation function designed to improve the representational capacity of neural networks. It is particularly useful in natural language processing tasks. The GLU activation is defined as:

$$\text{GLU}(a, b) = a \cdot \sigma(b)$$

where a and b are the outputs from the previous linear transformations, and σ is the sigmoid activation function.

The GLU effectively gates one of the linear transformations with the sigmoid of another, allowing the model to control the flow of information. This gating mechanism helps in capturing complex dependencies in the data.

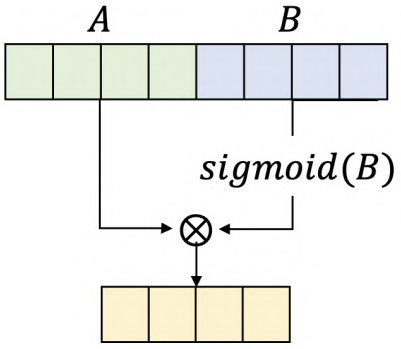

Figure 2.3: Gated Linear Unit (GLU) Activation Function
Source: https://www.researchgate.net

3. Related Works

Over the past few years, there has been a growing interest in driver monitoring. Some studies focus on analysing facial pose to classify the driver's gaze direction (Hari & Sankaran, 2021), in order to ascertain whether the driver is paying sufficient attention to the road. Others focus on examining the complete posture of the driver in order to recognise and classify driver videos into predetermined categories. The actions of interest can occur while performing manual driving of the vehicle or while being a passenger in an automated environment. Actions such as talking on the phone allow us to look at the road but distract us mentally, slowing our ability to react to unexpected events. In this work we will focus on recognising actions using automatic speech recognition (ASR) networks. Therefore, on the one hand we will examine the state of the art of pose-based driver action recognition and on the other hand we will study the different ASR models that exist in the literature.

3.1 Pose-based Driver Action Recognition

Human Action Recognition (HAR) through the human pose pose is a modality that has been widely studied. Usually these actions are general, focusing on sports, body movements, gestures, etc. The application to autonomous driving is direct through the integration of actions that lead to distraction.

Convolutional neural networks are utilized to classify actions based on pose input (B. Li et al., 2017). The pose is represented by an image in which the three coordinate components (x, y, z) of each joint are represented by the corresponding three components (R, G, B) of each pixel. Following the application of translation and scale normalisation, the image is fed into a multi-scale convolutional neural network (CNN) classifier, which will then classify the action among the predefined options. As seen in the Figure 3.1 the multi-scale model transforms the image into different scales, features are extracted with a shared model, and post-processing is applied for each scale. This allows objects to be captured that may appear at different sizes in the image.

Mingyan Wu et al. (Wu et al., 2021) combine spatial features obtained with a CNN with geometric features to predict the corresponding driver action. As seen in Figure 3.2 the model comprises three branches, with the image and its extracted pose serving as the input. The top two branches are responsible for extracting visual features through a

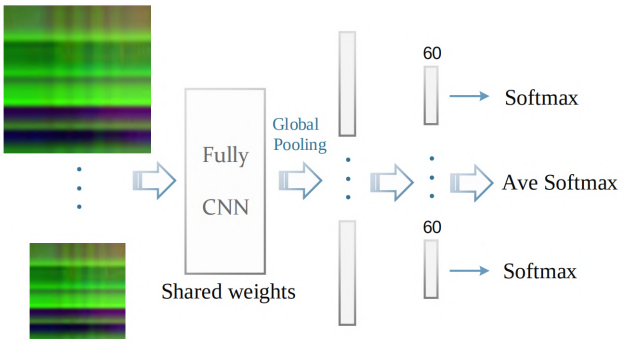

Figure 3.1: Multi-scale CNN model

Source: (B. Li et al., 2017)

CNN. The first extracts features from the whole image, and the second from the cropped hand. The lower branch extracts the features of the previously extracted pose. The three extracted features are fused in later stages. First, the two visual features are concatenated and then fused with the pose features. Finally, a classification head indicates the detected class.

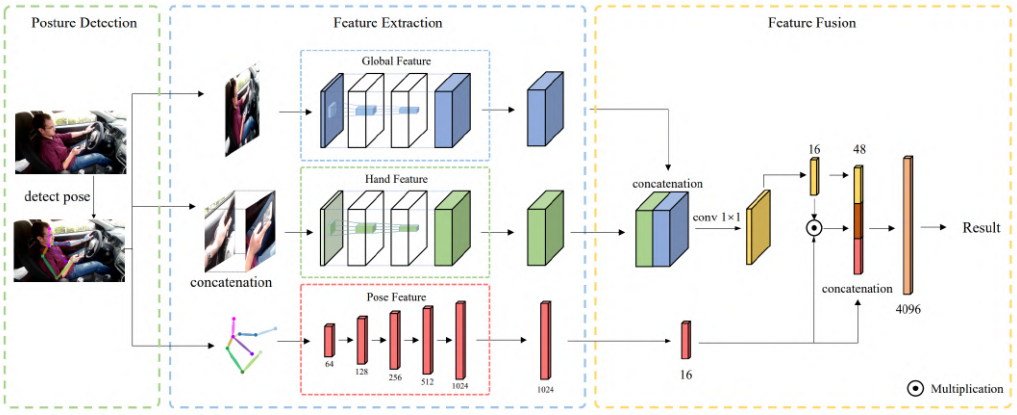

Figure 3.2: Proposed architecture by Mingyan Wu et al.

Source: (Wu et al., 2021)

The two-stream RNN architecture proposed by (Wang & Wang, 2017) models both temporal and spatial dynamics through a Recurrent Neural Networks (RNN). As seen in Figure 3.3, the temporal RNN models the temporal dynamics of skeletal actions by processing the sequential coordinates of joints over time. Two structures are explored: stacked RNN, which concatenate the coordinates of all joints at each time step, and hierarchical RNN (right of the Figure 3.3), which separate the joints in five main parts of the body.

The spatial RNN models the spatial dependencies among joints in the skeletal structure by converting the spatial graph into a sequence of joints and processing it through an RNN. Two methods for converting the graph into a sequence are proposed: chain sequence and traversal sequence. Finally the features of each branch is fed into a fully connected layer with a softmax activation function. The fusion is performed by combining the softmax class posteriors from the two nets.

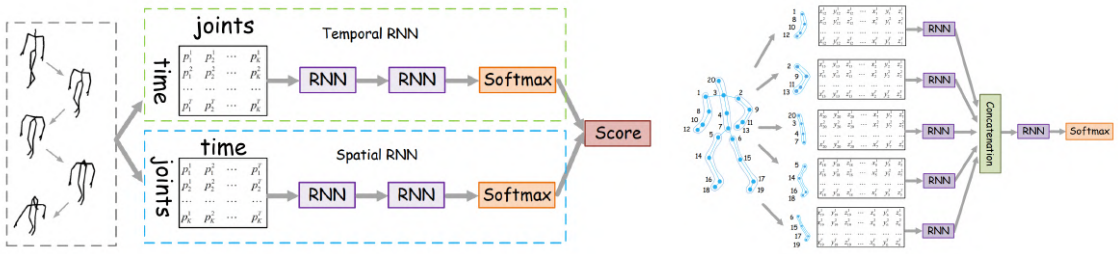

Figure 3.3: On the left the model proposed by Wang et al. and on the right the pose hierarchical RNN

Source: (Wang & Wang, 2017)

A third stream is introduced in (Martin et al., 2018) with information about the driver's environment. The model of the car interior is represented by bounding volumes, which are employed to calculate the distance from the wrist to each element. The distances of each frame are stacked to form a new input vector. As illustrated in Figure 3.4, a new branch is added to the architecture proposed by (Wang & Wang, 2017) to provide context to the sequence. The pipeline is another recurrent neural network (RNN) comprising two stacked long short-term memory (LSTM) layers.

Advanced networks unify the study by processing the 3D skeleton in the spatial and temporal domain in a single stream. Graph neural networks (S. Yan et al., 2018) are able to learn spatial and temporal patterns from data simultaneously. The methodology involves constructing a spatial-temporal graph from skeleton sequences and applying spatial graph convolutional neural networks. ST-GCN represents skeleton sequences as spatial-temporal graphs, where each node corresponds to a joint. A Graph Convolutional Neural Network (GCNN) is a type of neural network designed to operate on graph-structured data. In essence, it extends the principles of convolutional neural networks (CNNs) to graphs rather than regular grids, such as images. This enables to capture more complex relationships and dependencies within the graph structure, leading to improved performance on various

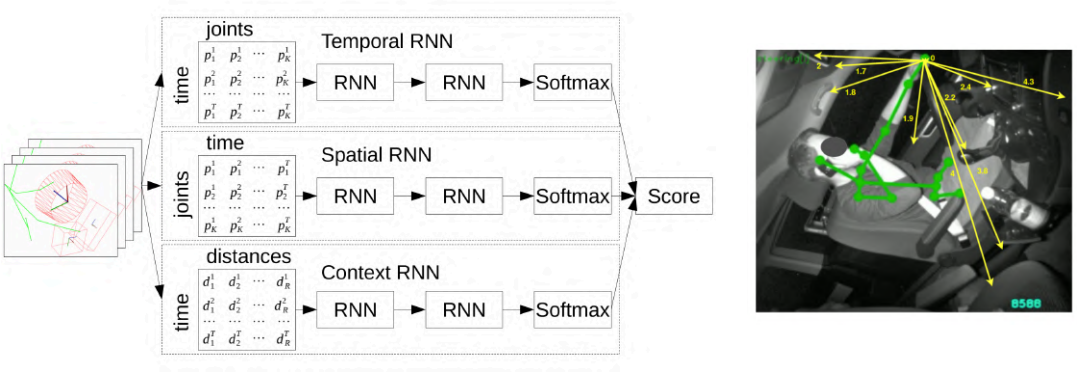

Figure 3.4: On the left the architecture proposed by Marting et al. and on the right an example of distance calculation from the wrist to different interior elements

Source: (Martin et al., 2018)

graph-based tasks.

Martin et al. (Martin et al., 2020) use this approach to integrate additional input modalities like interior elements and objects. In addition to the existing nodes formed by the 3D key points of the person, there are also those of the parts of the car interior and the detectable objects, previously annotated. The authors postulate that an object closer to the driver is more likely to be interacting with him. This is why, through the use of a distance-aware and threshold setting, they regulate how many edges are created and therefore also which nodes are added to the final graph. In the Figure 3.5 we find a sample interaction graph for the class "fetching an object". The diagram shows joints marked in red, interior elements in blue, and objects in green. Blue lines indicate connections between joints. Green lines represent connections to objects and orange lines show connections to interior elements.

With the advent of transformers, an efficient alternative called MLP-Mixers (Tolstikhin et al., 2021), inspired by CNNs and transformers, is proposed. It processes image patches instead of the whole image like Transformers. It contains channel-mixing blocks for each image patch (token) to capture spatial and per-channel features. St-MLP (Holzbock et al., 2022) explores the application of MLP-Mixers on 3D body pose skeletons overtime for gesture recognition in automated driving. As the body skeletons are defined in space and time, inside the mixing block they define two type of mixing operations; temporal-mixing and spatial-mixing. Also a Squeeze-and-Excitation (SE) block is included to weight the importance of each time step in the model. This helps the network to prioritize recent

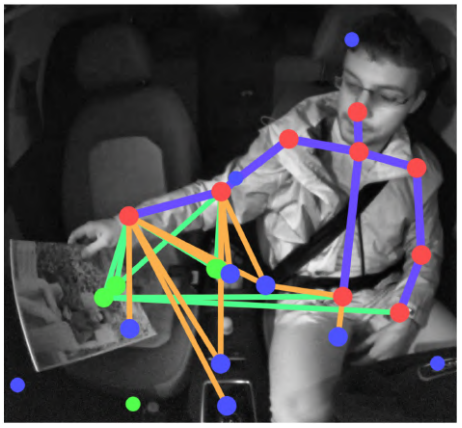

Figure 3.5: Sample interaction graph for the class "fetching an object" of the network proposed by Martin et al.

Source: (Martin et al., 2020)

time steps over earlier ones. As seen in the Figure 3.6, the final architecture consist of several mixing blocks concatenated. They are preceded by a block that adapts the size and format of the input data. A classification head is ultimately applied in order to infer the corresponding class. This approach demonstrates competitive results in pose-based action classification by adapting this architecture.

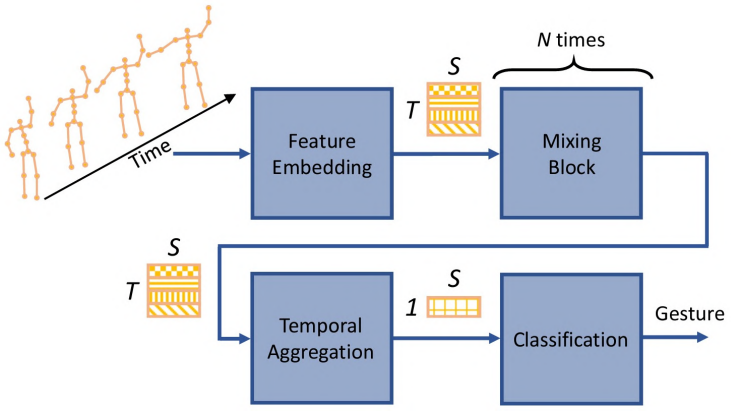

Figure 3.6: The architecture proposed in St-MLP

Source: (Holzbock et al., 2022)

Some methods struggle with limited receptive fields and computational inefficiencies when capturing correlations between all joints and frames. To address these limitations SkateFormer (Do & Kim, 2024) partitions joints and frames based on different skeletal-temporal relations and employs partition-specific self-attention within each partition. This

method efficiently captures critical joints and frames essential for action recognition, significantly reducing computational complexity. As seen in the Figure 3.7, it's key innovation lies in its partition-specific attention strategy, categorized into four skeletal-temporal relation types: neighboring joints with local motion, distant joints with local motion, neighboring joints with global motion, and distant joints with global motion. This approach allows the model to selectively focus on important skeletal and temporal features, leading to improved action recognition performance.

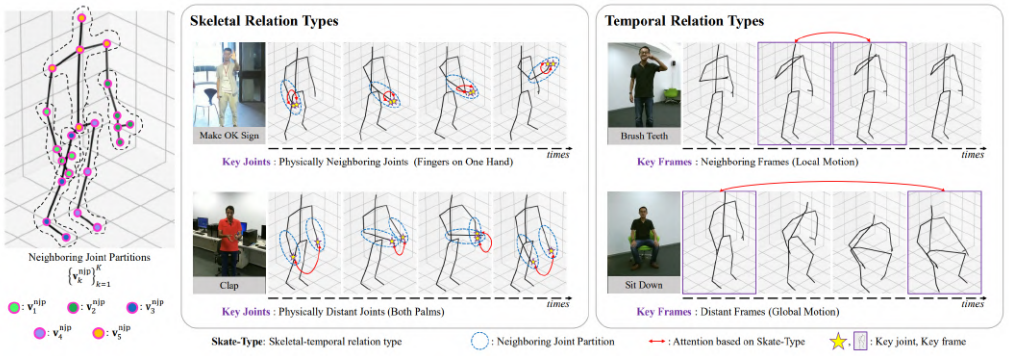

Figure 3.7: SkateFormer's partition-specific attention strategy for skeleton-based action recognition.

Source: (Do & Kim, 2024)

Given the increasing success of language models and natural language processing, we see efforts to integrate these models to enhance action recognition. (Xiang et al., 2023) introduces the Generative Action-description Prompts (GAP) for skeleton-based action recognition. As seen in the Figure 3.8 it utilizes a pre-trained large-scale language model to automatically generate detailed text descriptions of body part movements for various actions. These descriptions are then used in a multi-modal training framework, where a text encoder generates feature vectors from these descriptions to supervise the skeleton encoder, thereby improving action representation learning. The GAP method is designed to enhance the performance of skeleton-based action recognition models without increasing computational costs during inference. By integrating knowledge from detailed action descriptions, the GAP framework achieves significant improvements over existing baseline models. The authors highlight that this is the first work to leverage generative prompts for this purpose, offering a new multi-modal training paradigm that effectively uses textual information to guide the learning process of skeleton-based action recognition models.

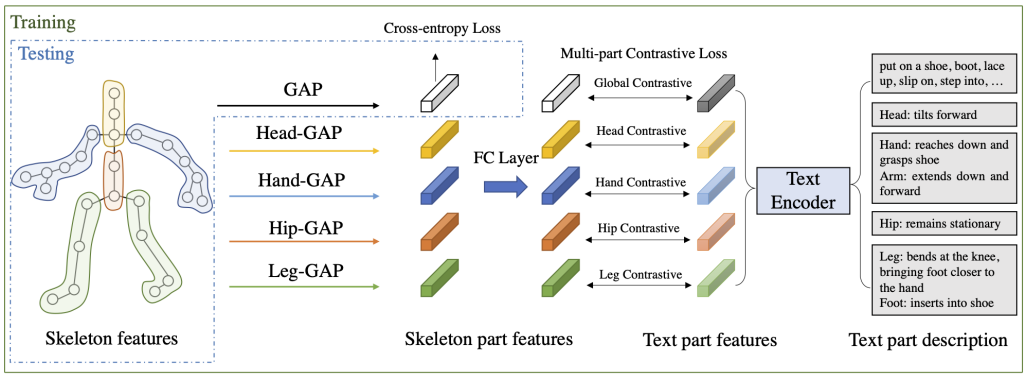

Figure 3.8: Architecture of GAP

Source: (Xiang et al., 2023)

3.2 Speech Recognition Networks

End-to-end automatic speech recognition (ASR) models typically comprise an encoder that processes a speech signal (a sequence of speech frames) and extracts high-level acoustic features. The encoder is complemented by a decoder that converts the extracted features into a text sequence. In all ASR models, the most important component is the encoder which converts speech input sequences into high-level feature representations. Our study will focus exclusively on ASR encoders.

In the initial ASR research, LSTM was the preferred model unit. The encoder could be configured as either a multi-layer unidirectional LSTM-RNN or a multi-layer bidirectional LSTM (BLSTM)-RNN like in (Sak et al., 2014). The encoder, which is a stack of LSTM layers, processes the input sequence to produce a fixed-size context vector that captures temporal dependencies in the speech signal. This context vector is then fed into a decoder, which generates the final transcription. The primary weakness of LSTM networks in ASR is their limited ability to capture long-range dependencies and parallelize computations due to their sequential nature. LSTMs process one timestep at a time, which can lead to inefficiencies, especially with long sequences. Transformer networks address this issue by using self-attention mechanisms, which allow them to process entire sequences in parallel and capture long-range dependencies more effectively outperforming LSTM-based methods.

Convolutional neural networks (CNN) are often used as backbones. The VGG convolutional architecture is employed as the backbone of an ASR model by (Beckmann et

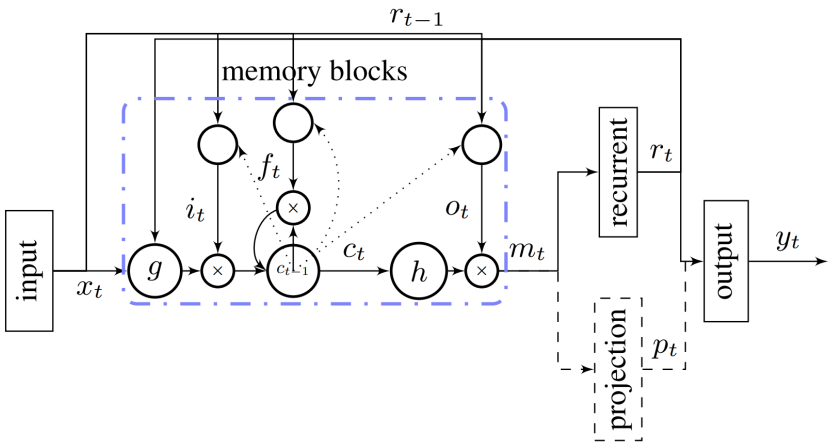

Figure 3.9: LSTM based RNN architecture for ASR

Source: (Sak et al., 2014)

al., 2019). This architecture is demonstrated to be capable of extracting relevant features from the spectrogram and performing satisfactory speech recognition. Furthermore, the pre-trained network is leveraged for transfer learning purposes. Jasper (J. Li et al., 2019) proposes a block architecture. Each block is comprised of a series of sub-blocks, each of which contains a one-dimensional convolution, batch normalization, rectified linear unit (ReLU) activation function, and dropout. Residual connections are used to connect the input with the last sub-block. The input audio is fed into the network as an audio spectograms. It is a visual representation of the spectrum of frequencies of a signal as it varies with time Each spectrogram is essentially a 2D matrix of numerical values, where one axis represents time, another represents frequency, and the values represent the intensity of each frequency component at each time step. QuartzNet's (Kriman et al., 2019) design is based on Jasper. They replaced the 1D convolutions with 1D convolutions that operate on each channel across K time frames and a pointwise convolutional layer that operates on each time frame across all channels. This results in a significant reduction in the number of parameters in the network, thereby reducing its size and inference time. In both models, a CTC (continuous speech recognition) decoder is employed to generate the output text. Citrinet (Majumdar et al., 2021) and ContextNet (Han et al., 2020) enhance the architecture with 1D Squeeze-and-Excitation (SE) context modules. This enhance the architecture by introducing attention mechanisms, allowing the network to focus on important features while suppressing irrelevant ones. These modules dynamically recalibrate the feature maps across channels, improving the model's ability to capture long-range

dependencies and enhance feature representations. As a result, Citrinet and ContextNet achieve better performance in various speech recognition tasks by effectively leveraging contextual information within the input signals.

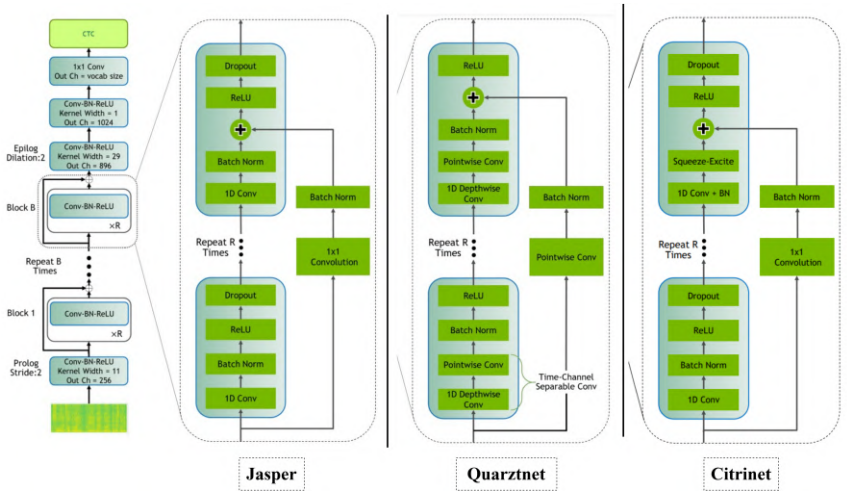

Figure 3.10: Comparison between three CNN blocks used in speech recognition models (Jasper, Quarztnet and Citrinet)

Source: (J. Li et al., 2019) and (Kriman et al., 2019)

Convolutional blocks represent an advancement over LSTM networks. However, the primary limitation persists in the inability to capture the global context. With the advent of Transformers, new models adopted this architecture to address this issue (Karita et al., 2019). They leverage self-attention mechanisms to model the relationships between all elements in a sequence simultaneously. In a typical Transformer-based ASR system, the encoder consists of several layers of self-attention and feed-forward networks, which process the input sequence of acoustic features to create a contextual representation of the speech signal. This representation is then passed to the decoder, which also uses self-attention layers to generate the final transcription. Residual connections and layer normalization are used to connect different layers and blocks. Multi-head self-attention (MHSA) enhances model capacity by applying multiple parallel self-attention mechanisms to the input sequence and then concatenating the outputs of each attention module. While the Transformer excels at capturing global context, it is less effective at extracting local patterns. To enhance its modeling capability, convolutional neural networks (CNN), which specialize in local information, are combined with the Transformer to create the Conformer (Gulati et al., 2020). Figure 3.12 illustrates the differentiation between the Transformer

and its evolution to a Conformer.

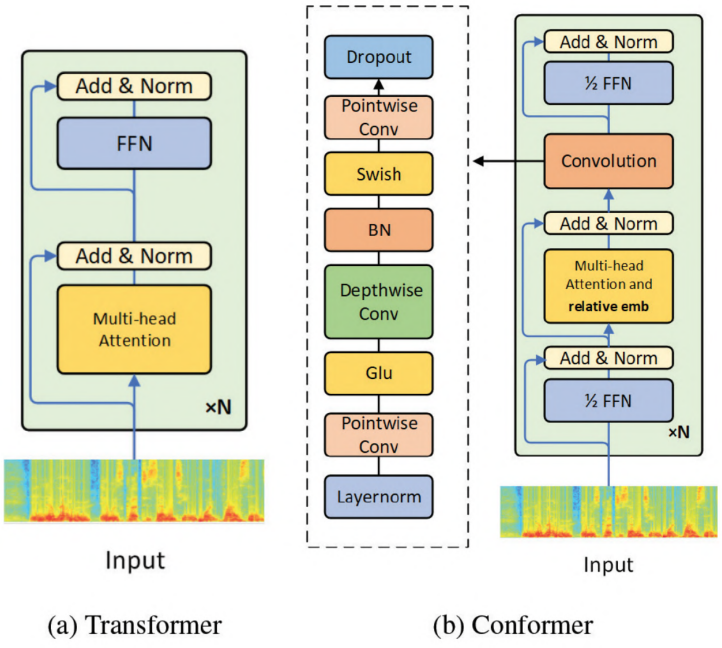

Figure 3.11: Transformer vs Conformer

Source: (J. Li, 2022)

As seen on the left of the Figure 3.12 the Conformer encoder first processes the input with a convolution subsampling layer to reduce the input size. This is followed by a concatenation of Conformer blocks. They are composed of four modules stacked together; a feed-forward module, a self-attention module, a convolution module, and a second feed-forward module in the end. The combination of attention blocks with convolutional blocks scores robust metrics while maintaining a low inference time. Recently, Squeezeformer (Kim et al., 2022) has been introduced, which makes a deep study and solves some design problems of the Conformer resulting in improved efficiency and performance. On the right of the Figure 3.12 the Squeezeformer architecture is observed. It redesigns the hybrid attention-convolution architecture, simplifying it for better performance. Additionally, the micro-architecture is refined by unifying activations, simplifying layer normalization, and incorporating depthwise separable convolutions, all contributing to enhanced accuracy and efficiency.

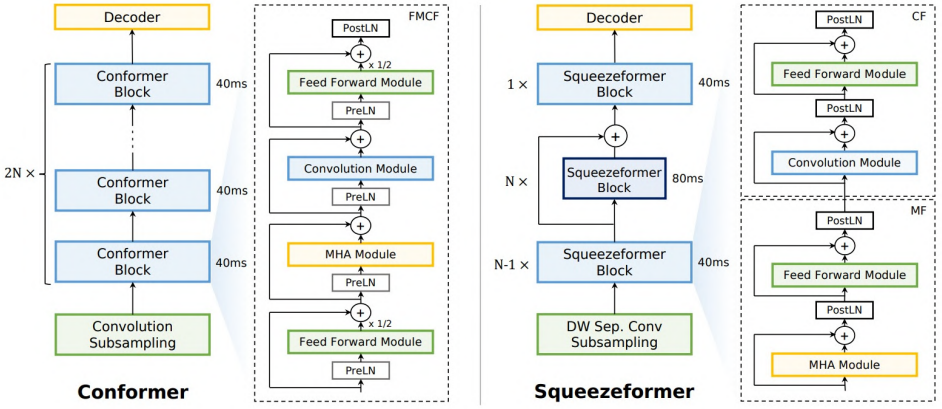

Figure 3.12: Comparison between the Conformer and Squeezeformer block

Source: (Kim et al., 2022)

3.3 Background Summary

From all of the above, we can see a clear trend towards the implementation of attention-based models in the latest models. They have been a clear revolution in all disciplines and the most powerful models include this technology. We can also observe that the use of mixed models of transformers and CNNs is a very studied line with good results. On the one hand, they exploit the advantages of transformers, such as global context capture, but without abandoning models with limited inference times, such as CNNs. This is the case of st-mlp (Holzbock et al., 2022), which achieves the best results on the Drive&Act datasets using this technology. Therefore, it seems clear to follow this line by including speech recognition networks. As previously studied, the SqueezeFormer model is integrated in this line with the combination of convolutional modules and modules with attention, which indicates that it is a good line to be followed. In addition, the architecture of the SqueezeFormer allows us to easily configure different sizes as it is divided into blocks and adapt it to our problems and data distribution.

After studying the state of the art, using SqueezeFormer blocks to classify driver actions is the chosen technique to be followed. Throughout this work we will investigate and develop how to implement this network in our context and how to overcome the gap between speech and actions. We will also use ST-MLP as a reference, as it currently ranks first in Drive&Act and uses similar technology.

4. Objectives and Working Methodology

4.1 Objectives

The objective of this work is to explore and verify the suitability of networks designed for automatic speech recognition for the classification of distracted driving. To this end, we will first conduct a study of the state of the art, after which we will select a network as baseline which we will adapt and apply to the aforementioned problem. We will train our network proposal with the Drive&Act dataset (Martin et al., 2019), which provides us with the 3D pose of the driver and collects 12 of the main actions identified in Drive&Act, listed in Table 4.1, that are executed during autonomous driving. By taking the state of the art as a reference, our goal is to achieve better results in the dataset and a more efficient inference time. st-MLP (Holzbock et al., 2022) currently supports the best results in this dataset with a 40.56 in validation and 34.61 in macro accuracy test. For the purposes of this application, an operating frequency above 30 Hz will be sufficient for a real-time working.

Table 4.1: Classes on Drive&Act Dataset

Nº	**Class**	**Nº**	**Class**
0	Eating / Drinking	6	Read magazine
1	Driving preparation	7	Read newspaper
2	Park car and exit	8	Take off jacket
3	Take over steering	9	Take off sunglasses
4	Put on jacket	10	Watch video
5	Put on sunglasses	11	Work on laptop

Once the model has been trained, adapted and tested in the Drive&Act database, it will be implemented in real life. A set of real data will be collected for the purpose of implementing this system. In this instance, a stereo camera will be installed inside the vehicle of the RobeSafe research group, where the system will be implemented in its final form. Following the format of the Drive&Act dataset, participants will simulate the

performance of the aforementioned actions. The dataset will then be fine-tuned in order to adapt the network weights to the real working environment and to measure performance. Given the limited number of participants, the finetuning will be conducted using the cross-validation technique. The project will be concluded with the presentation of the validation results and an analysis of the inference time on a real embedded system.

4.2 Methodology

The CRISP-DM methodology, which is widely used in the field of data science and is also applicable to deep learning projects, is proposed for the development of this work. It is an iterative process that allows us to return to previous stages thanks to feedback from later stages. As shown in Figure 4.1, there are six stages:

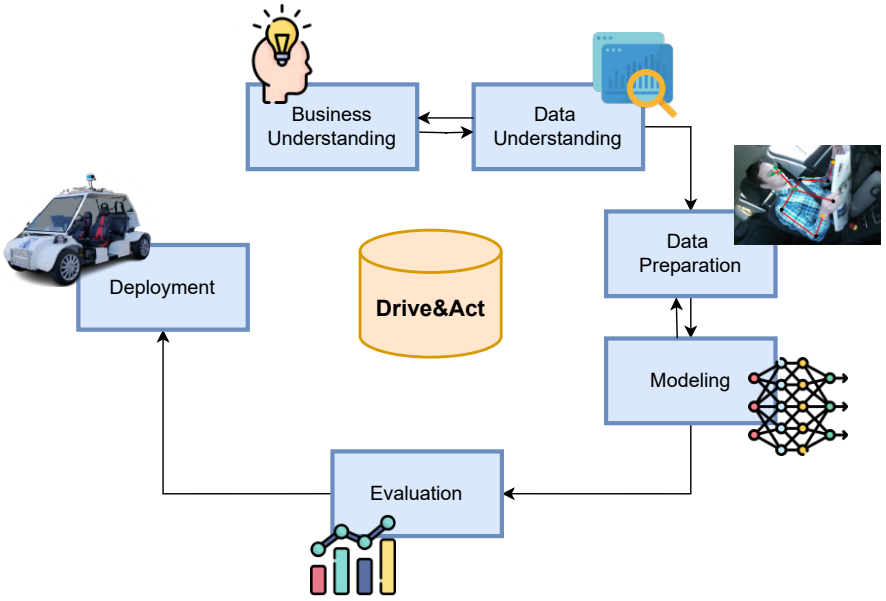

Figure 4.1: CRISP-DM methodology

- **Understanding the business**: It is essential to comprehend the objectives of the project and to study the potential contribution of deep learning techniques to those objectives.
- **Understanding the data**: It is necessary to examine the Drive&Act database in order to gain an understanding of the file system structure and the manner in which the information has been encoded. In this case, it will be important to know the key

points collected in the database and the format of the 3D coordinates. Furthermore, exploratory data analysis (EDA) will be conducted to understand the extent of any imbalances in the dataset, identify any potential issues with the data cleaning process, and assess the presence of any biases.

- **Data preparation**: This stage involves the preparation of the data to be used in the chosen model. The necessary pre-processing must be carried out and the data format must be adjusted to fit the input size of the deep learning model.

- **Modelling**: The model is trained with the dataset. This is the most time-consuming aspect of the process, as it necessitates the performance of different experiments to adjust the hyperparameters of the model. These are responsible for indicating the size of the transformer, how it learns, and it is vital to adapt to the distribution of the data.

- **Evaluation**: The performance of the model is evaluated using the appropriate metrics for the problem. This enables the measurement of whether the model meets the requisite objectives and the comparison with the state of the art.

- **Deployment**: The model is implemented in the production environment and further testing is performed to ensure that it functions as intended.

5. Method

5.1 Problem Formulation

The input data is represented as a three-dimensional matrix. Let $X \in \mathbb{R}^{3\times F\times K}$, where the first dimension encompasses the spatial coordinates (x, y, z) for each keypoint, F denotes the number of frames and K represents the count of keypoints in the pose. Let C be the class label associated with the i-th video segment, where i denotes the segment index within the video. Formally, $C = \{C_1, C_2, \ldots, C_M\}$, where M is the total number of distinct classes. The dataset D is a collection of multiple videos, $D = \{V_1, V_2, \ldots, V_N\}$ where N represents the total number of videos. Each video V_j is partitioned into segments X_i of duration F and the associated class labels C.

Our objective is to formulate a mapping function $f : \mathbb{R}^{3\times F\times K} \rightarrow \mathbb{N}$. Given a tensor X_i, our goal is to predict the corresponding class C with the following architecture.

5.2 Our Approach

The aim of this study is to investigate the application of automatic speech recognition in pose-based action recognition. The encoder is based on the Squeezeformer (Kim et al., 2022) block. It is a hybrid attention-convolution architecture that builds on the Conformer architecture (Gulati et al., 2020) to improve performance, making an important FLOPS reduction by solving some design problems. Considering the current state of the art and the design paradigms adopted by other solutions in the field, we propose the architecture shown in Figure 5.1. The foundations of this architecture where published in the IEEE Intelligent Vehicles Symposium (IV 2024).

The initial layer is a convolutional block, designed to conform the input data to the specifications of the encoder. The layer transforms the input's three (x,y,z) channels into the number of tokens per channels required by the encoder. The utilized kernel selectively operates on the spatial dimensions of the input data without temporal integration. This means that we will squeeze the input tensor without mixing the features in the temporal dimension. This operation guarantees that the number of frames matches the token length. Following the convolutional layer, a subsequent normalization block is incorporated, followed by the application of the activation function (SiLU). We have switched from an

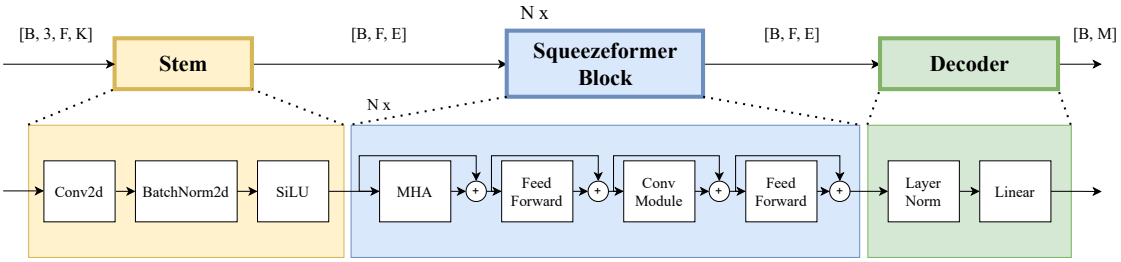

Figure 5.1: Architecture overview. **B:** Batch size; **F:** Number of frames; **K:** Number of keypoints; **E:** Channels per token; **M:** Number of classes; **N:** Number of encoder blocks.

image representation to a token like that can be consumed by a transformer. Compared to the audio data, ours have similar characteristics but in smaller dimensions. Due to this it is not necessary to temporarily reduce the input or apply feature extraction techniques. As summary, this first section transforms the input tensor $\mathbb{R}^{3\times F\times K}$ into $\mathbb{R}^{F\times E}$, where F is the token length (number of frames) and E is the channels per token size.

Next, we continue with the Transformer encoder with multiple Squeezeformer blocks, which are repeated N times. The input and output sizes are identical ($\mathbb{R}^{F\times E}$), allowing for seamless chaining of the blocks without any intermediate adaptation. Each Squeezeformer block comprises four modules. Following each block, the residual sum is calculated by adding the output of a module to its input. This technique prevents the problem of gradient fading and enables the network to learn identity functions. Additionally, after each sum, a post-layer normalization (PostLN) is applied to normalize the output of the modules.

5.2.1 Multi-Head Attention Module

The first module is a multi-head attention module (MHA). This module enables the model to simultaneously focus on various parts of the input sequence, which is crucial for capturing intricate dependencies between key points. The *MultiHeadedAttentionModule* combines relative positional encoding with multi-head attention. The *RelPositionalEncoding* is responsible for generating relative positional encodings for the input sequences. These encodings help the model to capture the relative positions of elements in the sequence, which is crucial for not losing the time dimension. The *RelativeMultiHeadAttention* performs the multi-head attention operation. It first projects the input sequences into query, key, and value vectors using linear transformations. It also projects the positional embed-

dings. The attention scores are computed by combining content-based and position-based scores. The content-based scores are derived from the dot product of queries and keys, adjusted by a bias term, while the position-based scores are computed from the dot product of queries and positional embeddings. These scores are then combined, normalized, and used to compute the weighted sum of the value vectors, resulting in the attended output.

5.2.2 Feed Forward Module

The next module is the Feed Forward Module, which applies transformations to the input data processed by the attention to enhance the model's ability to represent relationships in the data. This module is structured to follow the pre-norm residual pattern and employs layer normalization, Swish activation, and dropout to enhance the model's performance and regularization. As seen in the Figure 5.2, the first linear layer expands the input dimension by the expansion factor, creating a higher-dimensional space where complex representations can be learned. The second linear layer reduces the expanded dimension back to the original input size. Between these layers, the Swish activation function is applied to introduce non-linearity, and dropout layers are used to prevent overfitting by randomly zeroing some elements during training.

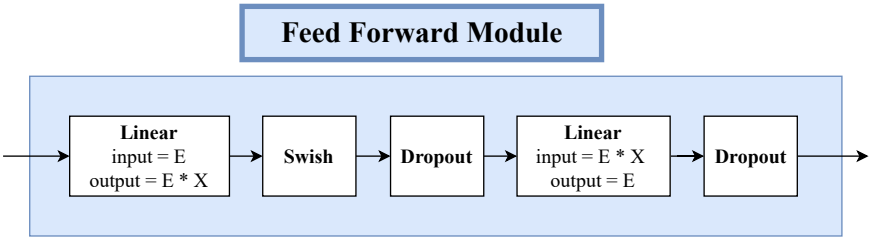

Figure 5.2: Feed Forward Module of the Squeezeformer Block. **E**: Channels per token; **X**: Expansion factor

5.2.3 Convolutional Module

The next module is the Convolution Module, which captures local features and helps the model understand the temporal structure of the data. As seen in the Figure 5.3 it combines pointwise convolution, GLU activation, depthwise convolution, batch normalization, and dropout operations to process input sequences efficiently while maintaining the necessary dimensionality and preserving spatial relationships within each channel. This makes

it suitable for capturing both local and global dependencies. The module comprises two distinct configurations of the torch convolutional block. By adjusting the module's parameters in specific ways, two markedly different effects can be achieved. The two versions are:

- The **PointwiseConv1d** class implements a 1-dimensional convolution where the kernel size is 1. This operation is referred to as pointwise convolution because it applies a linear transformation to each point (or channel) independently across the temporal dimension of the input. It's particularly useful for adjusting the number of channels or dimensions without changing the spatial dimensions (time in this case). This is achieved through a Conv1d layer with kernel_size=1.
- The **DepthwiseConv1d** class performs a 1-dimensional depthwise convolution. Unlike traditional convolutions that operate on all input channels simultaneously, depthwise convolution processes each input channel independently using a separate kernel for each channel. This helps capture spatial dependencies within each channel of the input sequence without mixing information across channels. The convolutional operation is applied with groups=in_channels in Conv1d, which ensures that each channel is convolved separately.

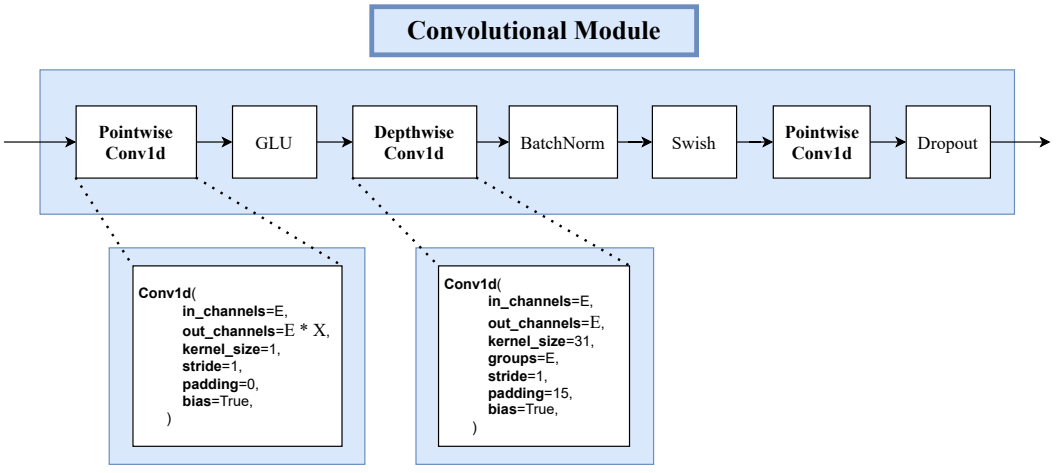

Figure 5.3: Convolutional Module of the Squeezeformer Block. **E**: Channels per token; **X**: Expansion factor

Batch normalization stabilizes and accelerates training by normalizing layer inputs and improving gradient flow. Dropout prevents overfitting by introducing noise and promot-

ing more robust feature learning. When used together, they complement each other in regularization, leading to better training dynamics and improved generalization.

The Squeezeformer block concludes with another Feed Forward Module. Finally, the decoder is presented. It consists of a normalization layer followed by a linear layer. This block transforms the output of the last Squeezeformer block into a vector of size $\mathbb{R}^M$, which indicates the confidence of belonging to each class.

5.3 Data augmentation

Data augmentation plays a crucial role in training deep learning models. This technique involves generating diverse variations of the existing dataset by applying transformations. The primary objective of data augmentation is to enhance the model's generalization capability by exposing it to a wider range of scenarios and variations in the input data. We propose the selection and adaptation of techniques that can be divided into four categories:

- **Temporal augmentations:** Resampling functions have been implemented to adjust the sequence length by either interpolating frames or repeating them to modify the sampling frequency. A random timeshift is also applied to the sequence, along with an augmentation that randomly removes the beginning and end of the sequence. Additionally, time masks are randomly applied to hide certain time periods in the sequences.

- **Keypoint drop:** Spatial masks are applied to the encoded pose, which hide a random set of keypoints for a random period of time. *FaceDrop* and *PoseDrop* are proposed to remove facial and bodily landmarks, respectively. In this way the occlusions of the pose are modelled.

- **Skeleton augmentations:** Based on the data augmentation methods proposed in (Chen et al., 2021), three modifications to the global driver skeleton are suggested. The height can be modified by stretching or shrinking the skeleton, the width by enlarging or narrowing it, and the rotation of the skeleton about the back axis can also be adjusted. These modifications enable the generation of diverse bodies.

- **CutMix-based augmentations:** CutMix (Yun et al., 2019) is a data augmentation technique that involves cutting and pasting square patches from different images

during training to create new training samples. This technique will be applied by mixing a contiguous set of keypoints between two sequences for a certain time to form a square. Additionally, as shown in Figure 5.4, we propose two modifications that are adapted to our type of data:

- **Spatial mixing:** A random set of contiguous keypoints are selected and spread over the entire sequence.
- **Temporal mixing:** During a random time duration all keypoints will be mixed.

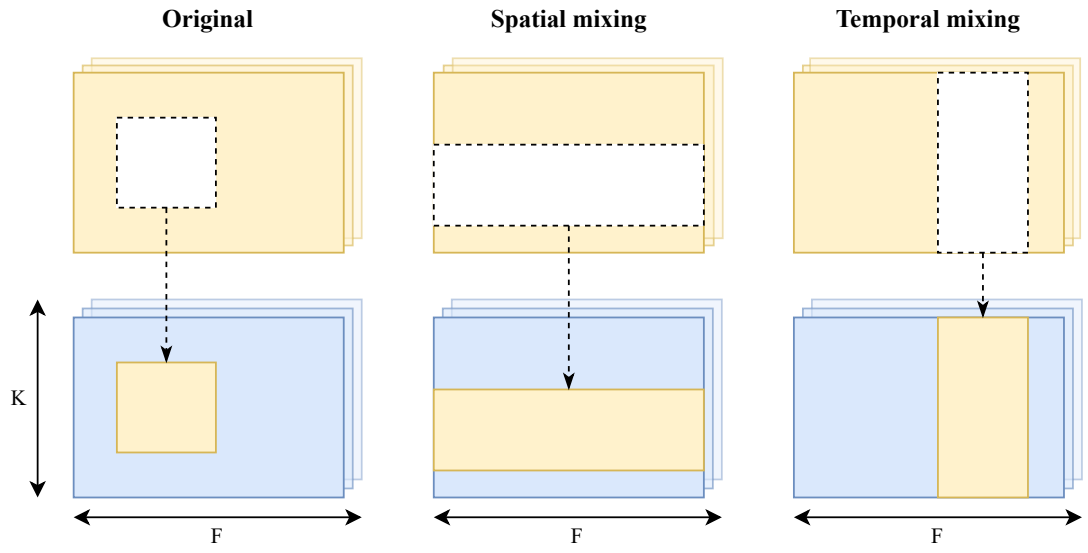

Figure 5.4: Original CutMix and proposed modifications. Blue and yellow represent two different sequences that belong to the same batch. **F**: number of frames. **K**: number of keypoints.

The labels of the pasted patches are mixed based on the area ratios of the patches, providing a smooth combination of multiple classes in the augmented training samples. Let $\mathbf{x}_i$ represent the i-th input sequence patch, $\mathbf{y}_i$ its corresponding ground truth label, and $\mathbf{x}_{i,j}$ denote the patch from a randomly selected second sequence overlaid onto $\mathbf{x}_i$. The label for the augmented sequence $\tilde{\mathbf{x}}_i$ is calculated as follows:

$$\tilde{\mathbf{y}}_i = (1 - \lambda)\mathbf{y}_i + \lambda \mathbf{y}_{i,j} \tag{5.1}$$

where λ is the area ratio of the overlaid patch $\mathbf{x}_{i,j}$ to the original sequence $\mathbf{x}_i$.

6. Dataset

In order to train and evaluate our model, we will utilise two datasets. The first is the Drive&Act dataset, which is widely used in the field of computer vision and is focused on the recognition of driver actions through the analysis of pose. This dataset will allow us to optimise and evaluate the proposed model, as well as enabling us to compare it with the state of the art. This will enable us to ascertain the suitability of the approach. Once the final model has been obtained, a new dataset will be prepared for production. The dataset will be collected using the same hardware and in the same environment in which the system is to be deployed. This will enable the system to be evaluated offline in a realistic setting. The composition and characteristics of the two datasets are detailed below:

6.1 Drive&Act

Drive&Act (Martin et al., 2019) is an action recognition dataset for autonomous vehicles in which 15 different people perform actions. The data was collected from six synchronised viewpoints. Figure 6.1 provides an example of each viewpoint for a specific frame. It contains common distractions such as *eating* or *drinking*, but also include actions more typical of autonomous cars such as *reading a newspaper* or *working with a laptop*. The data is collected in a static driving simulator. During the recording the driver is instructed to complete twelve different tasks. Most of the tasks are completed while driving autonomously. Each participant will carry out the action at their own discretion and as they would normally do. During every session, four unexpected take over requests are triggered. It also provides the 3D skeleton of the driver that has been extracted with OpenPose (Cao et al., 2019) and triangulated with the 2D poses from 3 frontal views (right-top, front-top, left-top). Figure 6.2 displays four class examples along with the driver's pose. The skeleton is formed by 13 landmarks corresponding to the driver's upper body. A 3D model of the car interior is also provided.

Three simultaneous annotations are defined for all sequences, from higher to lower level of abstraction:

- **Coarse scenarios/tasks**: Representing the highest level of abstraction, these encompass 12 general tasks that actors must complete during recording.

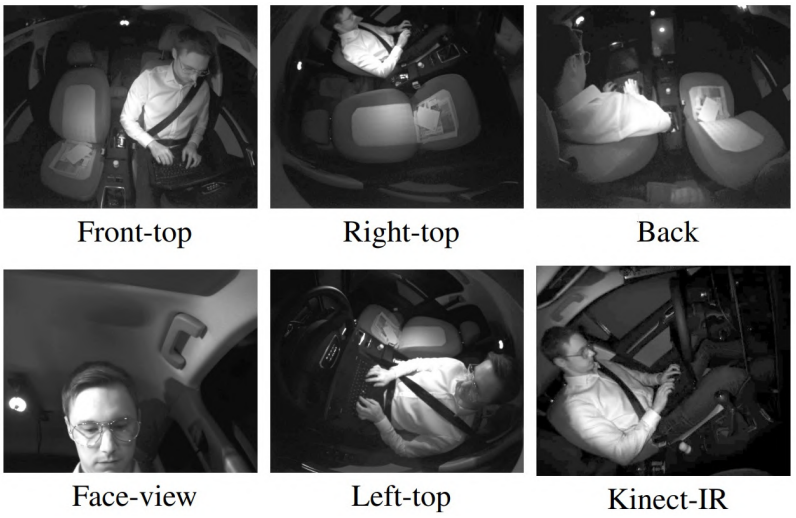

Figure 6.1: Example images of the working on laptop activity for the 6 different views
Source: (Martin et al., 2019)

- **Fine-grained activities**: At the second level of hierarchy, these activities further break down into 34 classes. Each one is performed uniquely by drivers based on individual preferences. These concise classes distinguish actions like opening and closing a water bottle.

- **Atomic action units**: Describing the most precise level of detail, this level encompasses 5 action types, 17 object categories, and 14 car locations. This results in 372 distinct combinations or classes, explicitly associated with object interactions.

As we are mainly concerned with the pose, the most appropriate annotation is the coarse. Then, coarse scenarios will be used for training and validation in the results section. The other annotations refer to specific object-related classes or classes where the driver does not appear (when entering or exiting the vehicle). The predefined splits, for training, validation, and testing, will be utilized and our chosen metric will be the mean per-class accuracy (macro-accuracy), established as the standard criterion. Our task will be to classify the entire sequence, already defined by the dataset, with the label corresponding to the class.

6.1.1 Exploratory Data Analysis (EDA)

It is crucial to understand the data before training the network. By exploring the data, we can identify potential imbalances, such as classes that are over-represented or under-

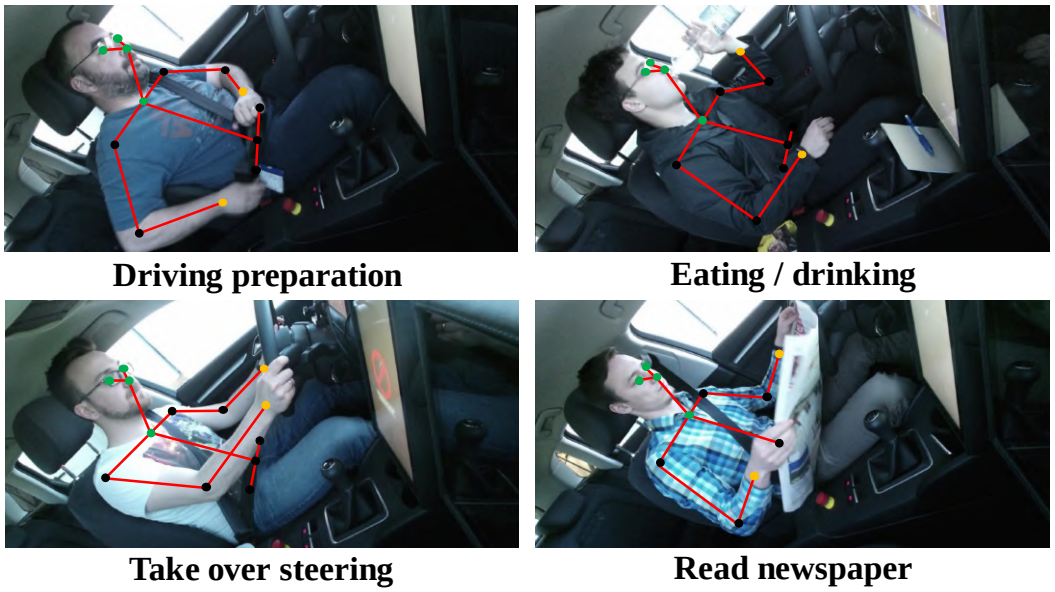

Figure 6.2: Example of four frames, including the driver's pose and corresponding class.

represented, bias, inconsistency in the data, and so on. The table shows the coarse classes of the dataset. The first study will be to analyse whether they are all equally represented in the dataset. Table 6.1 illustrates the frame-wise frequency analysis of the coarse annotations. There is a clear imbalance in the classes with the most represented being *watch a video* with 23% and the least represented being *take off sunglasses* and *put on sunglasses* with 2%. As previously stated, all actions are performed once per participant. Therefore, this imbalance is due to the natural duration of the actions. The actions least represented are those that generally take less time to perform and therefore, from a frame-wise frequency point of view, are in the minority. This characteristic must be taken into account when training the network.

Action Name	**Percentage**	**Action Name**	**Percentage**
Eating/drinking	15%	Read magazine	14%
Driving preparation	4%	Read newspaper	12%
Park car and exit	4%	Take off jacket	3%
Take over steering	2%	Take off sunglasses	2%
Put on jacket	5%	Watch video	23%
Put on sunglasses	2%	Work on laptop	14%

Table 6.1: Distribution of the coarse classes in the Drive&Act dataset.

Once we have established the overall contribution of each class, we will then examine the duration of the actions in more detail. This analysis will examine the distribution of the duration of each class segment. Figure 6.3 presents a box plot for each class under consideration. As previously noted, certain actions tend to last longer than others on average. Each individual performs actions in a manner that is influenced by their age, mobility, and character. This is particularly evident in actions such as *fastening the seat belt* or *putting on a jacket*, where the dispersion of actions is greater and we find a greater number of outliers. In general, all classes follow a normal distribution with no clear deviation to the right or left. This indicates that the data in the dataset are of high quality and do not present a clear bias.

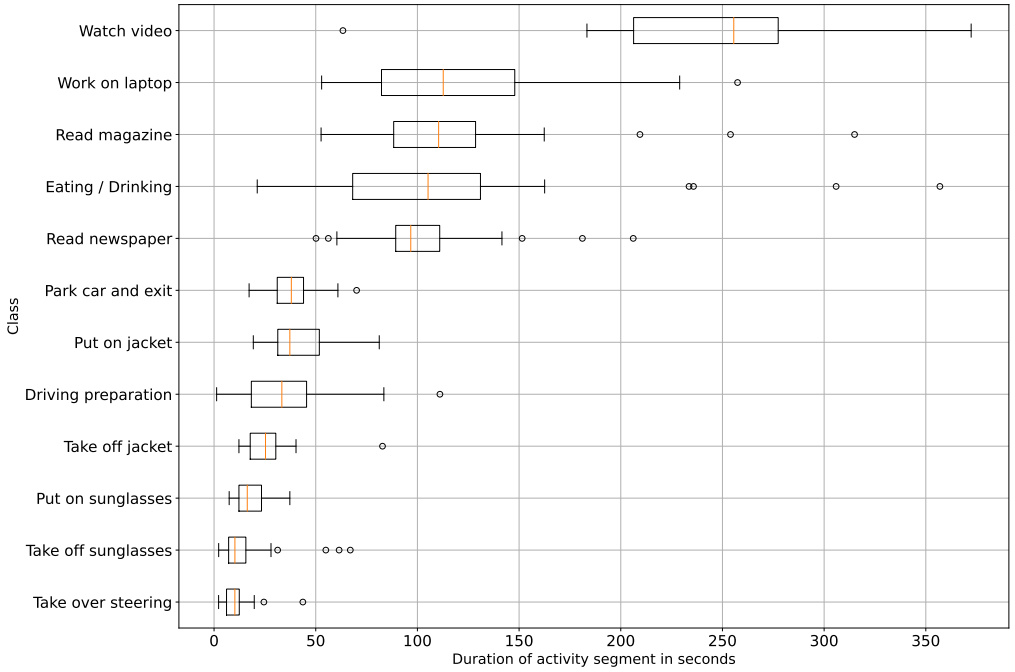

Figure 6.3: Duration in seconds statistics of the coarse activities as boxplot of the Drive&Act dataset.

The dataset presents the actions divided into 90-frame fragments. As previously stated, our network requires a fixed duration of input data. This, along with the fact that the actions have very different durations, will result in an unbalanced representation of each class in the total data set for each epoch. As outlined below, we will utilise a metric that accounts for the class imbalance and implement techniques to address this imbalance during the training process. Table 6.2 provides an overview of the number of occurrences, mean, median, maximum and minimum duration in frames for each class. The final column

Table 6.2: EDA table; nº Seg: Number of 90-frame segments of each class.

Class	**Count**	**Mean (s)**	**Median (s)**	**Max (s)**	**Min (s)**	**nº Seg.**
Eating / Drinking	33	117.1	105.2	356.9	21.2	1892
Driving preparation	29	36.1	33.3	111.1	1.3	1277
Park car and exit	27	38.2	38.0	70.1	17.2	1214
Take over steering	33	11.4	10.2	43.7	2.2	1181
Put on jacket	29	41.1	37.3	81.2	19.3	1023
Put on sunglasses	29	18.3	16.3	37.3	7.4	412
Read magazine	29	121.0	110.4	315.0	52.6	362
Read newspaper	29	104.4	96.7	206.1	50.1	352
Take off jacket	27	27.1	25.3	82.8	12.2	261
Take off sunglasses	28	16.5	10.2	67.0	2.3	193
Watch video	23	245.6	255.6	372.3	63.4	171
Work on laptop	30	120.1	112.7	257.4	52.9	141

displays the number of 90-frame fragments for each class included in the dataset. It is to be expected that the actions that on average have longer durations will have a higher number of segments. In total, the dataset consists of 8479 segments, which are divided into three sets: train, validation and test. The division was determined by the dataset creator himself and will be used. This also helps us to make a fairer comparison with the state of the art.

6.2 Test Dataset Creation

In order to test and adapt our model to the described use case, it is necessary to generate a real-world data set. For this purpose, the same hardware and element layout will be used as in the final implementation. The actors will take the actions as real-world as possible in order to replicate the behaviour of the system's users. The following section outlines the selection and configuration of the data capture hardware, the software used to collect and process the data, and the setup of the volunteers.

6.2.1 Hardware implementation

As illustrated in Figure 6.4, the data capture setup is comprised of three components:

- **ZED X Mini[1]:** is a stereo camera device developed by Stereolabs. It is compact and designed for machine vision and augmented reality applications. It provides high-quality stereoscopic images and real-time depth, rendering it an optimal choice for object detection, pose estimation, 3D mapping, and autonomous navigation applications. Highly accurate depth perception from 0.1 to 8m, suitable for vehicle interiors. This camera is the data entry point of our system and is installed inside the vehicle, facing the driver.

- **Zed Box Orin NX 16GB[2]:** is an embedded computing system developed by NVIDIA. The device is designed for use in artificial intelligence and deep learning applications at the edge. The device incorporates the processing power of the NVIDIA Orin GPU architecture, rendering it well-suited to data-intensive tasks such as image recognition, video analysis, and other real-time deep learning projects. In addition to processing the depth of the stereo cameras, several accelerated pose estimation and object detection networks have been implemented for this system. These networks, when combined with the depth data, can extract 3D detections and poses.

- **Laptop:** This is a generic laptop computer that will run a Docker container with the necessary software to connect to the Zed Box. From this computer, data collection for the dataset will be controlled. Additionally, the RVIZ and a plugin will permit the real-time visualisation of the data collected by the device, like the RGB image, the depth point cloud or the 3D pose. Further details about the software implementations can be found in the subsequent section.

As illustrated in the image, the camera is linked to the ZED Box via a FAKRA port and a secure GMSL2 connection. The computer is connected to the ZED Box via Ethernet. It should be noted that this configuration will be employed in both the data collection phase and the implementation phase.

6.2.2 Software implementation

ROS 2 (Robot Operating System 2) is a vital component of our system's communication infrastructure. ROS2 is a flexible framework for building robot software. It provides tools and libraries to help developers create complex robotic systems. In ROS2, a "topic" is a

[1] https://store.stereolabs.com/products/zed-x-mini-stereo-camera

[2] https://store.stereolabs.com/products/zed-box-nvidia-jetson-orin-nx-16gb

Figure 6.4: Experimental hardware setup.

communication channel where nodes exchange data. Nodes can publish information to a topic or subscribe to receive updates from it. Topics facilitate the exchange of messages between different components of a robotic system, enabling collaboration and coordination. A "message" in ROS2 is the data structure sent over a topic. It defines the format and content of the information being exchanged between nodes. Messages can represent various types of data, such as sensor readings, motor commands, or other forms of information relevant to the robot's operation. In our case, the ZED Box will communicate with the laptop through messages. In the case of the human pose, we will utilise custom messages developed by Stereolabs, which will require the installation of the ZED SDK[3].

In order to facilitate the data capture process, a customised visualisation environment was created utilising the Python library Pygame. As illustrated in Figure 6.5, the RGB image of the camera, with the 3D pose of the driver projected onto the image plane, is displayed on the left. The last 90 frames captured of the pose, encoded as previously detailed, are displayed at the bottom of the page. Finally, the information provided by the Zed Box inside the node is displayed on the right-hand side, including whether the tracking of the person is activated and the number of keypoints generated. Furthermore, this visualisation enables real-time inference, allowing other fields to indicate whether the model is operational and the predicted class.

6.2.3 Data generation

A total of 6 individuals participated in the generation of the test data. The software developed for data capture will randomly select actions to be performed by the participants.

[3]https://www.stereolabs.com/developers/release

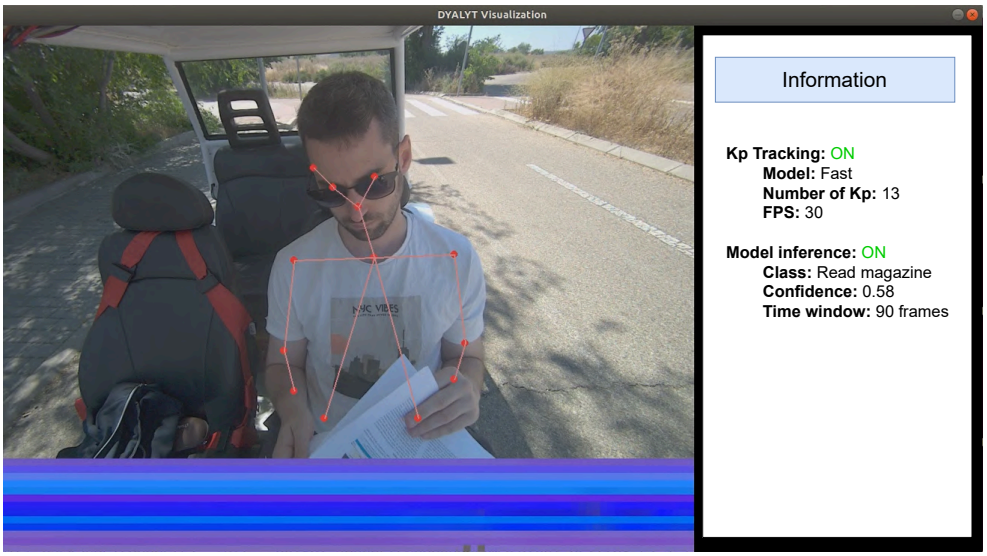

Figure 6.5: Custom made python visualization with the RGB camera, keypoints and relevant information. **Kp**: Keypoint

Each participant will perform each of the actions listed in Table 4.1 at least once, during the time he or she determines. The first action will always be *driving preparation*, and the last action will always be *park car and exit*. Furthermore, the actions *put on sunglasses* and *put on jacket* will be allways be selected first than the corresponding *take off* action. In order to guarantee the security of the data collection process, the vehicle was stationary while users simulated fully autonomous driving. The data annotation was conducted in real time, with the precise start and end of each action being recorded. This methodology permitted the data to be gathered and labelled in real time, thereby ensuring the quality and randomness of the data. In addition, the *idle* class has been added. This class corresponds to non-distraction, i.e. any time when no action is taking place. This is necessary to allow the model to work by itself without the need for an earlier stage.

6.2.4 Exploratory Data Analysis (EDA)

Once the data has been collected, an exploratory analysis is performed to ascertain the distribution of the data. As with the Drive&Act dataset, individuals perform the actions in the way they want and during the desired time. Figure 6.6 illustrates the distribution of each action. Despite the limited number of individuals, a normal distribution is observed in the data. As with the Drive&Act dataset, we observe that certain actions tend to be completed in a shorter time due to their inherent nature. The order of the actions is not

identical, but actions such as *watching video* or *reading newspaper* are situated in high durations, while actions such as *put on sunglasses* or *take off sunglasses* are placed in low durations. We also observe that longer actions have a greater degree of dispersion, due to the nature of the actions and the wide possibility of performing them. In contrast, shorter actions are very restricted actions with a very low dispersion.

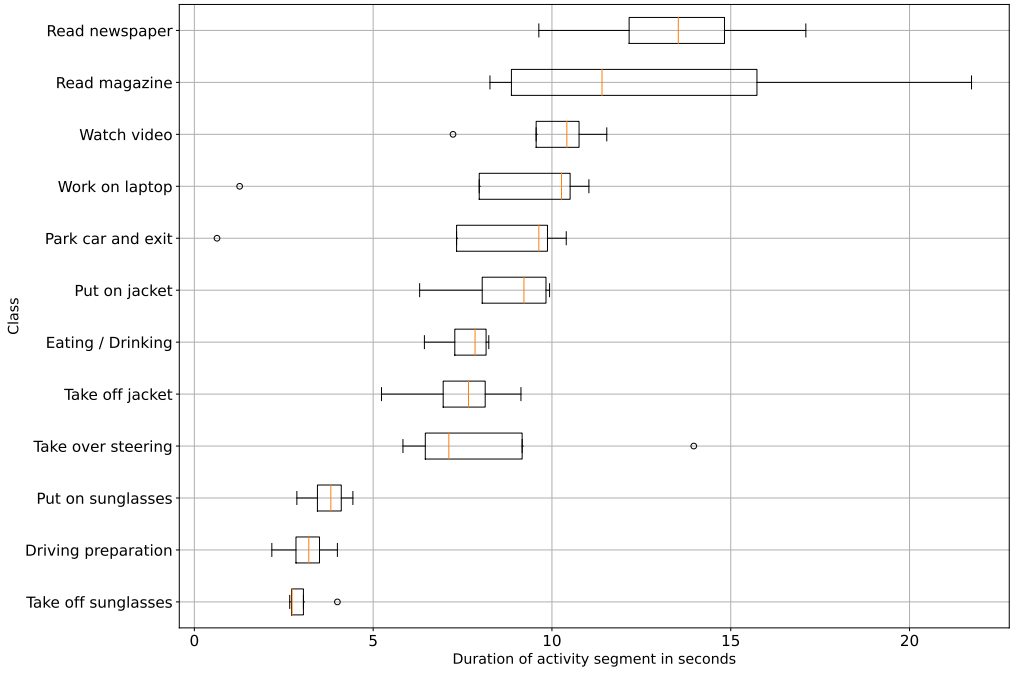

Figure 6.6: Duration in seconds statistics of the coarse activities as boxplot of the own-made test set.

Table 6.3 illustrates the distribution of the three-second segment samples in the dataset. As previously observed, the dataset is imbalanced. As in Drive&Act, actions with longer average duration contain a greater number of segments and therefore have a greater representation in the dataset. Finally, we observe the 2d representation in the image plane of the pose obtained by the ZEDX mini for different actions. Figure 6.7 shows an example frame of six different classes.

Action Name	**Percentage**	**Action Name**	**Percentage**
Eating/drinking	8%	Read magazine	14%
Driving preparation	3%	Read newspaper	15%
Park car and exit	8%	Take off jacket	8%
Take over steering	9%	Take off sunglasses	3%
Put on jacket	9%	Watch video	11%
Put on sunglasses	4%	Work on laptop	8%

Table 6.3: Distribution of the coarse classes of the own-made test set.

Figure 6.7: Example frame for the 2d representation in the image plane of the pose obtained by the ZEDX mini for six different actions.

7. Experiments

7.1 Implementations Details

Our architecture has been implemented with PyTorch (Paszke et al., 2019) and PyTorch Lightning (Falcon, 2019). The model has been trained for 100 epochs with a balanced batch of 256 sequences. This process involves selecting samples in a way that ensures each batch contains approximately the same number of examples of each class, regardless of their frequency in the full dataset. As previously discussed, the dataset is unbalanced and this technique allows us to counteract the problem. It helps to reduce bias towards majority classes, improve the stability of the training process and enhance the model's convergence. Regarding the optimization, AdamW with a weight decay of 0.24 has been used. Furthermore, cosine annealing has been employed as the learning rate scheduler with a warm-up value of 60% and a learning rate of 0.005. AdamW was used for optimization and cosine annealing was employed as the learning rate scheduler.

In the pursuit of optimizing model performance, a systematic exploration of hyperparameters was conducted employing Wandb Sweeps (Biewald, 2020). The network size was investigated by varying the number of encoder blocks and token channels to identify an optimal configuration. Various dropout values for the blocks were also included in the exploration. Parameters pertinent to the learning rate scheduler, such as learning rate (lr), warmup steps, and weight decay, were incorporated into the search. Detailed information regarding these parameter values can be found in the code repository. The search is performed using bayesian optimization seeking to minimize the validation loss function (cross entropy loss). Figure 7.1 illustrates the evolution and combination of hyperparameters selected by Wandb. Each dot represents a completed training. The X-axis represents time, while the Y-axis represents the best validation result of the experiment. The hyperparameters are randomly initialised and training is then carried out. A target variable is established, which in this case is the macro-accuracy in validation. In this manner, the optimiser will vary the different parameters in such a way that it will identify relationships and dependencies within them that will maximise the target metric. Due to the size of the network, 385 training runs were conducted over a period of approximately 24 hours on an NVIDIA GeForce RTX 2080 Ti GPU. Results of the best model are shown in the following section.

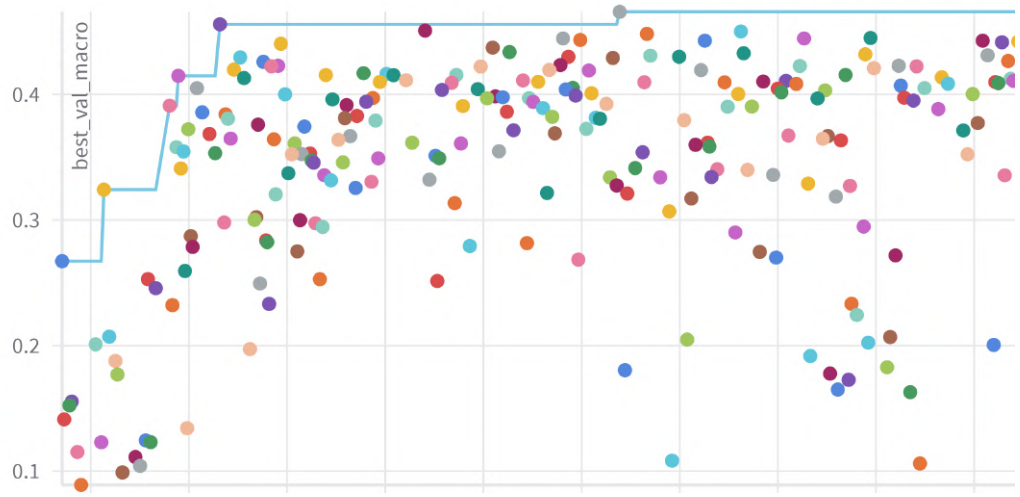

Figure 7.1: Evolution over time of macro accuracy in the validation set during the hyper-parameter optimisation.

Source: (Biewald, 2020)

7.2 Metric

Macro-accuracy or mean per-class accuracy is a metric commonly used in classification tasks to evaluate the performance of a model, especially when dealing with unbalanced datasets. It calculates the accuracy for each class separately and then takes the average of these accuracies. In this study, we will categorise 90-frame segments into one of several predefined classes. This will allow us to calculate the accuracy of each class's classification and subsequently determine the average. The formula for macro-accuracy can be expressed as follows:

$$\text{Macro-accuracy} = \frac{1}{N} \sum_{i=1}^{N} \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i}$$

Where:

- N is the total number of classes.
- TP_i is the number of true positives for class i.
- FP_i is the number of false positives for class i.

In contrast, the micro-accuracy calculates the mean of all segments. The main issue is that in datasets with an imbalanced distribution, as is our case, minority classes are

underrepresented. Even if all instances of these classes are misclassified, the final metric will not be affected. This phenomenon can be observed in our dataset, where classes that represents 1%-2% of the overall segments might be ignored and lose importance.

7.3 Results

We evaluate our architecture in the Drive&Act dataset (Martin et al., 2019) for the coarse modality and in our own-made test set with the same classes. First, we train in Drive&Act to obtain some initial weights in order to compare our proposal with the state of the art. Then, we evaluate the trained model on our test set. Finally, we perform finetuning based on the best trained model from the Drive&Act dataset to adapt it specifically to our environment through a cross-validation finetuning process.

7.3.1 Drive&Act

The first experiment is conducted on the Drive&Act dataset. During training, hyperparameter tuning was performed to optimize the model's performance. The results presented below correspond to the best model trained through the tuning process. To deal with the unbalance problem and to perform a fair comparison without modifying the input dataset, we apply weighted or balanced batch. This process adjusts the influence of each sample in the batch according to the class distribution of the dataset. This approach helps mitigate class imbalance issues during model training without directly modifying the dataset itself.

Figure 7.2 displays the training and validation loss over epochs, with the training loss depicted in orange and the validation loss in blue. Notably, the training loss is consistently lower than the validation loss. Both losses decrease in parallel until reaching a minimum point where they converge and stabilize, marking the point where early stopping is applied to stop the training process. The Figure 7.3 shows the validation macro-accuracy, which achieves a maximum of 44.6% when stabilizing. This indicates that the model has reached its optimal performance in terms of accuracy as well.

Fig. 7.4 shows the accuracy per class in the test set. The activity with the highest classification accuracy is *driving preparation*, reaching approximately 0.78. *Take over steering* and *watch video* also show relatively high accuracy, with values around 0.60 and 0.58, respectively. Activities such as *park car and exit*, *read newspaper*, *eating/drinking*, and *put on jacket* have moderate accuracies, ranging from about 0.45 to 0.48. These activities

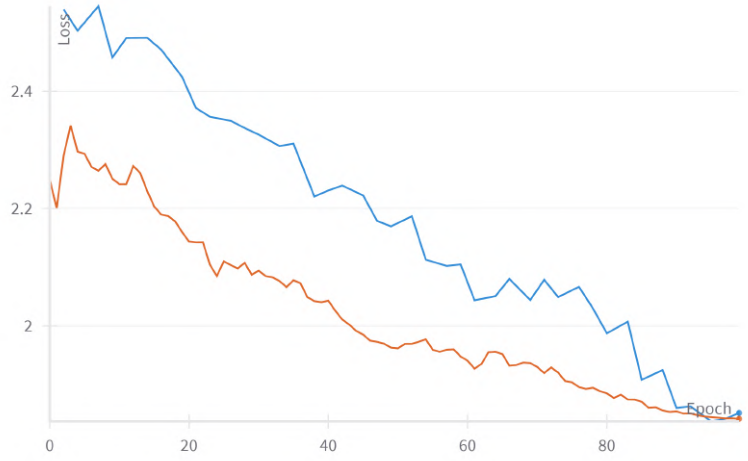

Figure 7.2: Training (orange) and validation (blue) loss over epochs. The training is stopped early using early stopping once the validation loss stabilizes.

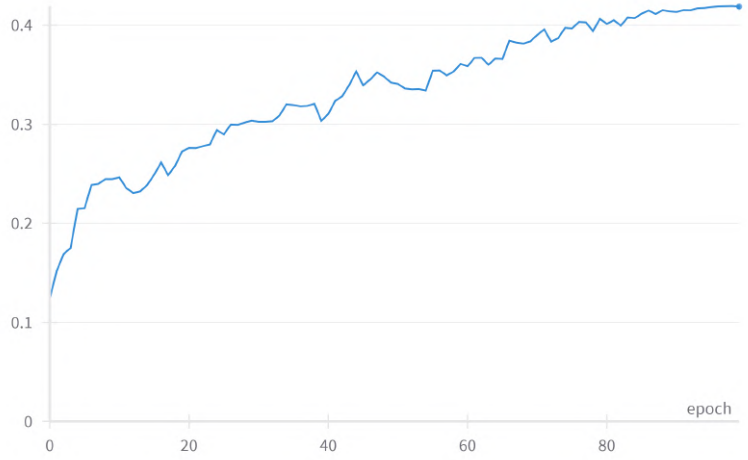

Figure 7.3: Validation macro-accuracy over epochs.

are clustered in the middle of the chart, indicating that the network performs reasonably well but with room for improvement. The activities with the lowest classification accuracy are *put on sunglasses* and *take off sunglasses*, with accuracies close to 0.20 and 0.10, respectively. *Work on laptop* and *take off jacket* also have relatively low accuracies.

The network excels in classifying activities that likely have more distinct and recognizable patterns, such as *driving preparation* and *take over steering*. Lower accuracies in activities like *put on sunglasses* and *take off sunglasses* suggest that these actions might have more subtle or less distinguishable features, making them harder to classify accurately. In addition, the underrepresentation of these classes due to their short duration

makes it more difficult for the network to generalise and match them in the test set. Finally, these classes behave in a similar way, which is particularly complicated when only pose is taken into account.

Table 7.1 displays the macro-accuracy results comparing with other state-of-the-art architectures. The data column indicates the input used in the network. 'P' represents the driver pose, 'I' represents the 3D model of the vehicle interior and 'O' the objects. Our encoder consists of two Squeezeformer blocks with 120 channels per token. Figure 7.5 presents a graphical representation of the validation and test results of our model in comparison to the state of the art.

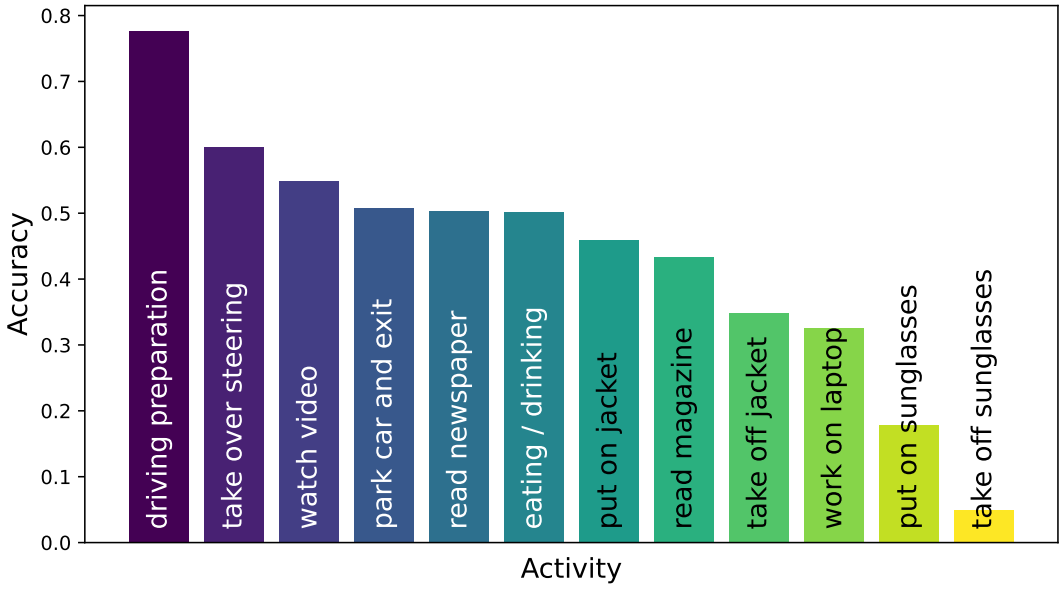

Figure 7.4: Per class accuracy of the test set of the Drive&Act dataset.

The architecture demonstrates superior performance compared to other proposals, achieving better results in both validation and test. Performance is superior even when compared to networks that use extra data (I,O). The major improvement lies in the test suite. The values obtained are more balanced, demonstrating a higher generalisation capability. When compared to st-MLP (Holzbock et al., 2022), a solution also based on attention, we improved results by 4.04 in validation and 8.98 in the test set. The number of parameters in our network is 538k, while st-MLP has 588k, resulting in a reduction of 50k parameters.

Table 7.1: Evaluation of coarse scenarios/tasks on the Drive&Act dataset using macro-accuracy. **P:** Pose; **I:** Interior; **O:** Object

Data	**Method**	**Validation**	**Test**
-	Random	8.33	8.33
P + I	Interior (Martin et al., 2019)	35.76	29.75
P + I	Three-Stream (Martin et al., 2018)	41.70	35.45
P + I + O	GNN (Martin et al., 2020)	42.82	37.84
P	Pose (Martin et al., 2019)	37.18	32.96
P	Two-Stream (Wang & Wang, 2017)	39.37	34.81
P	st-MLP (Holzbock et al., 2022)	40.56	34.61
P	**ours**	**44.60** (+4.04)	**43.59** (+8.98)

7.3.2 Real Test Set

After training the model on Drive&Act, we take the best weights and evaluate it directly on our custom dataset. It achieves a macro accuracy of **32.25**, which, although representing approximately a 25% reduction in performance compared to the original dataset, remains commendable considering the substantial differences between the setups.

Once evaluated on the Drive&Act trained model in our test set, we will perform fine-tuning on our custom dataset to maximize its performance and adapt it to our use case. We will take the best model trained in the upper section as initial weights. Due to the small size of our data, we will apply cross-validation techniques. Cross-validation is a statistical method used to estimate the performance of machine learning models. It involves partitioning the data into subsets, training the model on some subsets (training set), and validating it on the remaining subsets (validation set). This process helps in assessing the model's ability to generalize to an independent dataset. The primary goal is to ensure that the model performs well not just on the data it was trained on, but also on unseen data. In our case, we will employ two types of cross-validation to ensure robust evaluation:

- **Leave-One-Person-Out Cross-Validation (LOPO-CV):** In this approach, we

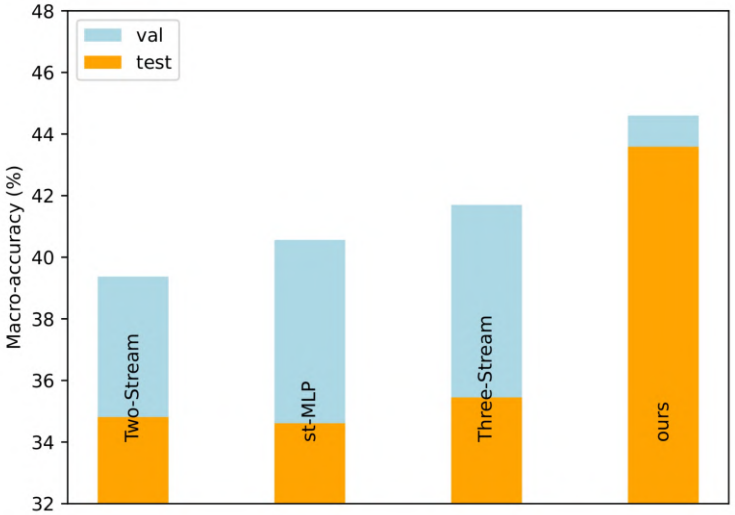

Figure 7.5: Comparison of macro accuracy between validation (blue) and test (orange) for state-of-the-art and Our model.

create separate "bags" of data for each individual in the dataset. During each iteration of cross-validation, the model is trained on the data from all but one individual and tested on the data from the excluded individual. This technique helps evaluate the model's performance in scenarios where it encounters actions performed by people not seen during training. It is particularly useful for assessing how well the model generalizes to new subjects.

- **Stratified k-Fold Cross-Validation:** Here, we mix all the actions in the dataset and create "bags" by splitting the data into k equally sized folds, ensuring that each fold has a proportional representation of all action classes. In each iteration, k-1 folds are used for training, and the remaining fold is used for validation. This process is repeated k times, with each fold used exactly once as the validation data. Stratified k-Fold Cross-Validation ensures that each fold is representative of the entire dataset, maintaining the balance of action classes, which is crucial for evaluating the model's performance on different actions uniformly.

By using these cross-validation techniques, we aim to achieve a comprehensive understanding of our model's performance in a real scenario, both in terms of its ability to generalize across different individuals and its robustness in recognizing various actions. This will

provide a more reliable assessment of the fine-tuned model before deployment.

Quantitative results

We present the quantitative results of our cross-validation experiments. We applied the two different types of cross-validation: Leave-One-Person-Out Cross-Validation (LOPO-CV) and Stratified k-Fold Cross-Validation. The performance of the model in each fold is reported to provide a comprehensive evaluation. In LOPO-CV, the model was trained on the data from all but one individual and tested on the data from the excluded individual. This process was repeated for each individual in the dataset. Table 7.2 shows the validation results for each fold.

Table 7.2: Leave-One-Person-Out Cross-Validation finetuning

Person	**1**	**2**	**3**	**4**	**5**	**6**	**Mean**
Macro accuracy	47.5	39.7	45.5	41.0	44.3	42.9	**43.48**

In Stratified k-Fold Cross-Validation, the dataset was split into k folds, ensuring that each fold had a proportional representation of all action classes. The model was trained on k-1 folds and validated on the remaining fold, repeating the process k times. Table 7.3 shows the validation results for each fold.

Table 7.3: Stratified k-Fold Cross-Validation finetuning

Fold	**1**	**2**	**3**	**4**	**5**	**6**	**Mean**
Macro accuracy	45.9	49.5	51.4	48.3	45.4	47.2	**47.95**

The results from both types of cross-validation provide a robust evaluation of our model's performance. On the one hand, the LOPO-CV shows that there are sets of people who better shape the behaviour of others. We observe unstable results due to the fact that there are folds that model better behaviour than the rest. In addition, there are people whose way of performing actions diverges more from the average. On the other hand, the Stratified k-Fold Cross-Validation shows greater stability between folds due to the mixing of the people, as well as better average results. However, it should be noted that the validation data is not entirely new, as all individuals participate in both the training and validation sets.

It can be concluded that the finetuning process achieves satisfactory and comparable results to those obtained from the reference dataset. This is possible because it makes use of the knowledge gained from training the weights on a larger dataset, which is then applied to the specific context of our use case. By having a similar setup with the same actions, we are able to achieve competitive results. For the deployment, we will train with stratified k-Fold, as this allows us to have a validation set to monitor the training status, while including all participants.

Qualitative results

We provide a qualitative analysis of the model's performance by showcasing examples of correctly and incorrectly classified actions. This analysis helps us understand the strengths and limitations of our model beyond quantitative metrics. For each example, three equidistant frames are observed within the action. The image is not merely a visual representation; rather, it serves to help us to understand the data. The projection of the three-dimensional points on the image plane is superimposed on the image. Finally, under each frame, we can see the encoded data that feeds the network. The last 90 frames prior to the current captured moment are represented. Figure 7.6 to 7.12 illustrate correctly and incorrectly classified examples of sequences.

Figure 7.6: Example sequence of the real test set. **Predicted:** Driving preparation
Ground Truth: Driving preparation

Inference time

Inference time is a critical aspect of our model's performance, particularly because it will be used in real-time safety applications. Ensuring an appropriate inference time is paramount to provide timely and accurate detection of actions while driving. In our experiments, we

Figure 7.7: Example sequence of the real test set. **Predicted:** Read magazine

Ground Truth: Read newspaper

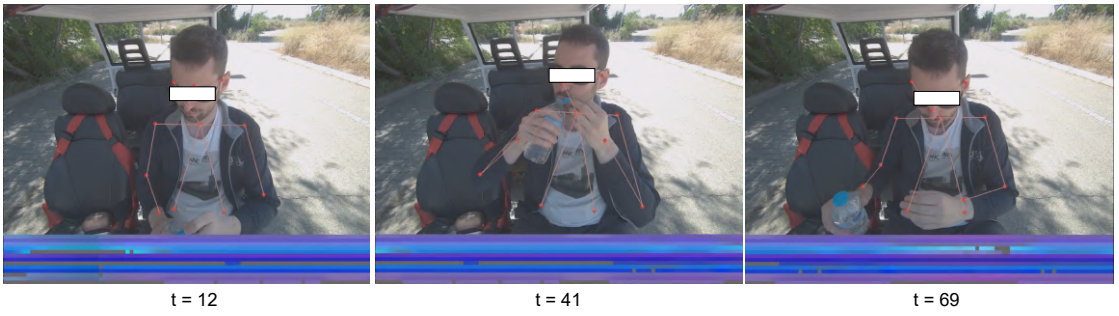

Figure 7.8: Example sequence of the real test set. **Predicted:** Eating / Drinking

Ground Truth: Eating / Drinking

Figure 7.9: Example sequence of the real test set. **Predicted:** Put on sunglasses

Ground Truth: Put on sunglasses

tested the inference time of the network on both a high-performance desktop GPU, the NVIDIA 2080 Ti, and an embedded system, the Jetson Orin NX with 16GB of memory. For the inference process, a sliding window approach is employed. We take a window of the last 90 frames captured at the rate of inference. The model then processes these frames to predict the current action being performed. This method allows the system to continuously monitor and analyze the driver's actions in real-time. To evaluate the

Figure 7.10: Example sequence of the real test set. **Predicted:** Iddle

Ground Truth: Iddle

Figure 7.11: Example sequence of the real test set. **Predicted:** Iddle

Ground Truth: Watch video

Figure 7.12: Example sequence of the real test set. **Predicted:** Work on laptop

Ground Truth: Work on laptop

real-time performance of our model, we measured the inference time on both the NVIDIA 2080 Ti and the Jetson Orin NX. The results are summarized in the Table 7.4.

The inference times indicate that the model performs efficiently on both types of hardware, making it suitable for real-time applications. The NVIDIA 2080 Ti, being a high-performance desktop GPU, provides faster inference times, which is expected due to its superior processing capabilities. On the other hand, the Jetson Orin NX, designed for em-

Hardware	Inference time	FPS
NVIDIA 2080 Ti	12.8 ms	78.1
Jetson Orin NX 16GB	29.2 ms	34.2

Table 7.4: Inference time results on different hardware.

bedded systems, also achieves acceptable inference times, demonstrating its feasibility for deployment in in-vehicle systems where space and power efficiency are critical. Our evaluation demonstrates that the model consistently achieves inference times that meet and exceed the minimum requirement of operating at more than 30 Hz on both the NVIDIA 2080 Ti and Jetson Orin NX 16GB. This performance benchmark ensures that our system can function effectively in real-time, providing reliable and timely action recognition that is crucial for enhancing vehicle safety systems.

7.4 Ablation Study

In this section, we aim to explore various features of our deep learning model to try to draw further conclusions and future lines of research. The primary focus is to evaluate the impact of the different sets of data augmentation techniques. By applying these techniques individually and in combination, we assess their effects on the model's performance on the test set. Additionally, we implement the detection of specific objects to understand how the network adapts to the inclusion of this data. This exploration provides insights that could lead to new advancements and improvements of the work. Finally, we assess the robustness of the network when the camera viewpoint is modified, in order to measure its ability to adapt to varying conditions.

7.4.1 Data augmentation

Data augmentation helps to improve the generalisation and robustness of data, particularly when the dataset is small. This leads to a smaller difference between the results of the validation and test sets. An ablation study has been proposed to test the contribution of these techniques to the final result. The study uses the configuration outlined in the result section and is conducted with the four data augmentation groups described above. The individual contribution of each group of used techniques are depicted in Table 7.5.

We can conclude that the utilization and combination of data augmentation techniques

Table 7.5: Ablation study in data augmentation. **O:** Original; **S:** Spatial; **T:** Temporal

Data augmentation techniques						
Temporal	**Skeleton**	**CutMix-based**			**Val**	**Test**
		O	**S**	**T**		
-	-	-	-	-	33.44	30.88
✓	-	-	-	-	38.17	33.25
-	✓	-	-	-	36.17	28.62
-	-	✓	-	-	34.61	32.68
-	-	-	✓	-	35.21	35.28
-	-	-	-	✓	35.70	34.79
-	-	✓	✓	✓	39.42	37.11
✓	✓	✓	✓	✓	**44.60**	**43.59**

improves the performance of our network. The test results benefit the most, with a greater improvement compared to the model without data augmentation. Among all the techniques, the CutMix-based technique stands out. Each contributes to enhancing the baseline, but integrating them provides a more significant improvement.

7.4.2 Objects implementation

We observed that classes that involve the interaction with specific objects exhibit a poorer performance. Previous studies (Weyers et al., 2019) demonstrated that combining these objects with the pose improves the results of related classes. Martin et al. (Martin et al., 2020) combine the pose information with manually annotated objects, resulting in a 7.54 improvement over the pose-only model. In order to validate this hypothesis we propose including the objects as a keypoints. This means that dimension K will be increased by O, which represents the number of detectable objects in the scene. The vector's value will indicate the confidence of the detection and will be repeated to expand in the three dimensions (x, y and z). Figure 7.13 illustrates what the input data would look like when poses and objects are combined. The colour red indicates that this object has been identified in the corresponding frames. Each row corresponds to a different previously defined object. These rows are concatenated with the encoding of the pose (striped lines

of the image) used in the work. This new encoding will be provided to the network and will result in the following outcomes.

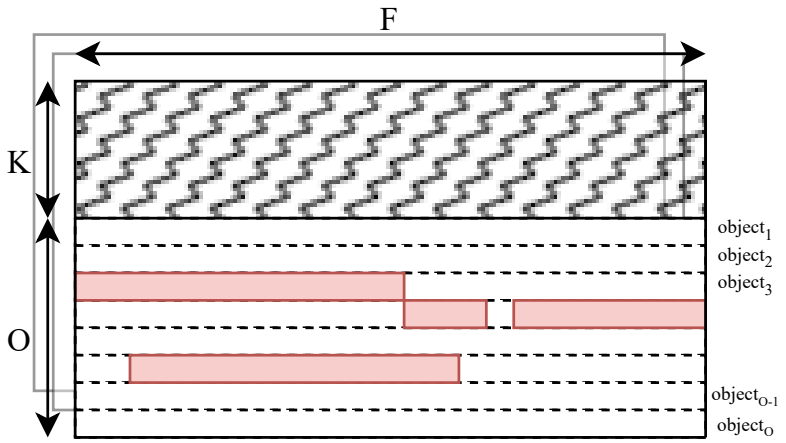

Figure 7.13: Illustration of the input data with combined poses and objects. The color red indicates identified objects in the corresponding frames. Each row represents a different predefined object, concatenated with the encoding of the pose (striped lines in the image) used in this work. **K**: number of keypoints; **O**: number of objects; **F**: number of frames in the segment.

The dataset provides annotations for the objects that the driver interacts with. This information is used to conduct our experiment. Information regarding the object is encoded as previously described. Consequently, the network is trained to classify actions based on the pose and objects information. The experiment results for fine-grained annotations using the macro-accuracy metric are presented in Table 7.6.

Table 7.6: Evaluation for fine-grained activities on the Drive&Act Dataset using macro-accuracy.

Type	**Method**	**Validation**	**Test**
Video	P3D ResNet (Qiu et al., 2017)	55.04	45.32
Video	I3D Net (Tran et al., 2015)	**69.57**	63.64
Pose + Objects	Graphs (Martin et al., 2020)	65.87	58.8
Pose + Objects	ours	66.21	**64.83**

Significant improvement is achieved through the inclusion of objects, reaching compara-

ble results to state-of-the-art video networks based on transformers. We require additional processing for pose extraction and object detection. It is important to note that results are based on perfect annotations. Overall, this experiment shows the potential of our proposed architecture including additional information, such as objects, with pose to improve classification performance at the cost of introducing a higher computational load, similar to the vision-based approaches.

7.4.3 Robustness in viewpoint change

Achieving a robust network that handles changes in viewpoint minimize data collection efforts and facilitate data reuse. This issue is important in the automotive field since multiple cameras have already been introduced in modern cars, but their mounting position can change between vehicle models. Manuel Martin et al. (Martin et al., 2023) measured the behavior of their network in response to changes in viewpoints. We replicate their experiment to assess the robustness of our architecture. The 3D pose is extracted from the triangulation of three frontal views (right-top, front-top, left-top). The dataset includes a Kinect camera with depth estimation located at the top-left position. The 3D pose is extracted combining OpenPose 2D pose along with the depth image. A cross-view test is conducted by training in one modality and evaluating in the opposite one.

Table 7.7: Cross-view evaluation for I3D and the proposed architecture on the validation set of fine-grained activities of Drive&Act. **CM:** Center mirror; **KIR:** Kinect IR

Train	**Test**			
	I3D Net		**Ours**	
	CM	KIR	CM	KIR
CM	69.6	6.8	52.88	35.58
KIR	6.7	72.9	43.77	49.67
Variation	-90.37%	-90.67%	-17.22%	-28.36%

Results are compared with I3D, a video-based end-to-end model. Results of I3D are taken from (Martin et al., 2023). Table 7.7 displays the cross-view experiment results for both I3D and our pose-based architecture in the fine-grained activities of the Drive&Act dataset. We find two different viewpoints; Center mirror (CM) and Kinect IR (KIR). Both networks are trained and evaluated from a single viewpoint, and then tested on the same

and on the opposite. This approach highlights the performance degradation when weights trained from one perspective are applied to another. It is important to highlight that these actions are being carried out in the same environment and with the same distribution of data. The only modification is the type of sensor and the position of capture.

The performance of the video-based network is significantly impacted by changes in camera perspective, resulting in a 90% reduction. In contrast, our proposal based on pose data exhibits greater robustness to sensor changes, with an average reduction of 22.79%, indicating a significant improvement on viewpoint robustness.

8. Conclusion

The research presented in this work has explored the integration of speech recognition networks with pose-based gesture recognition for application in autonomous vehicles. This innovative approach leverages the strengths of transformer-based architectures and convolutional neural networks (CNNs), to enhance the accuracy and robustness of driver action classification systems.

8.1 Summary of Findings

- **Attention-based models:** The study highlights the significant impact of attention-based models in contemporary machine learning applications. These models, particularly those utilizing transformer architectures, have revolutionized various fields by effectively capturing global context in data sequences. The ability of transformers to process and relate all elements in a sequence simultaneously has proven beneficial in the domain of speech and gesture recognition.

- **Combination of CNNs and Transformers:** The research demonstrates the effectiveness of hybrid models that combine CNNs with transformers. This combination exploits the local feature extraction capabilities of CNNs while leveraging the global context understanding of transformers. The Conformer model, an evolution of the transformer, incorporates convolutional layers to improve the capture of local patterns, thereby enhancing overall model performance.

- **Driver action classification:** The proposed approach for driver action classification, which integrates 3D pose data with speech recognition networks, has shown promising results. By incorporating object interaction recognition and addressing viewpoint robustness, the system achieves a comprehensive understanding of driver activities. This is crucial for developing reliable and efficient autonomous vehicle systems.

8.2 Contributions

This study contributes to the field of artificial intelligence and autonomous driving in several key ways:

- **Novel approach:** This work introduces a novel approach by integrating speech recognition networks into the driver action recognition process. This has enabled us to successfully investigate this field of artificial intelligence and adapt Conformer networks to this task. The network, in conjunction with data augmentation techniques, has facilitated advancement beyond the state of the art, demonstrating the viability of these networks. This work provides a foundation for future integrations of novel models in the field of speech recognition.

- **Selection and adaptation of data augmentation techniques:** Recognizing the challenges posed by limited data, this research carefully selects and adapts various data augmentation techniques to enhance the robustness and generalizability of the models. These techniques, such as temporal and spatial transformations, help mitigate the issue of small datasets by artificially expanding the training data. This approach ensures that the model can learn effectively even from limited examples, leading to more accurate and reliable action recognition.

- **Enhanced robustness:** The focus on robustness to viewpoint changes addresses a critical challenge in the deployment of autonomous systems. By ensuring that the model performs consistently across different camera perspectives, the research paves the way for more adaptable and resilient autonomous driving solutions.

- **Advancing benchmark performance:** This research contributes to advancing the state-of-the-art in the Drive&Act dataset, a widely used benchmark for driver action recognition. Through meticulous model design and training strategies, the proposed approach achieves superior performance compared to existing methods. This improvement not only demonstrates the efficacy of the proposed system but also sets a new standard for future research in this area.

- **Incorporating object information:** We conducted an experiment involving the inclusion of information about objects of interest in the scene, which led to a notable improvement. This serves as a foundation for future implementations of an object detector and a later fusion with the pose.

- **Fine-tuning and evaluation on a real-world dataset:** The work emphasizes the importance of real-world applicability by fine-tuning and evaluating the proposed models on an custom dataset. This step ensures that the system can be deployed

in our own use case. The evaluation on real-world data demonstrates the model's robustness and reliability, reinforcing its potential for deployment in real-world autonomous vehicles.

This work has resulted in a publication that has been accepted for presentation at the IEEE Intelligent Vehicles Symposium (IV 2024). The presentation took place in June 2024 at Jeju Island, Korea. Titled *Do You Act Like You Talk? Exploring Pose-based Driver Action Classification with Speech Recognition Networks*, the paper highlights the innovative use of speech recognition networks in driver action recognition.

9. Future Work

The research conducted in this master thesis lays a strong foundation for advancing driver action recognition systems, yet several avenues for future exploration remain to be pursued:

- **Object fusion with the pose:** Future work should focus on integrating a suitable object detector into the current system. This will enable the fusion of object recognition data with existing gesture, providing a more comprehensive understanding of the driver's environment and interactions. By incorporating object detection mechanisms, the system can better interpret the driver's actions in context. For instance, recognizing objects that the driver interacts with can enhance the accuracy of action classification and improve decision-making processes in autonomous driving.

- **Driver's gaze and car interior information:** Incorporating the driver's gaze into the recognition system is a crucial next step. Gaze tracking can provide insights into the driver's focus and attention, further informing the action recognition system and enhancing its ability to predict and respond to driver behaviors. Adding information about the car's interior, such as seat positions, control layouts, and interior lighting conditions, can improve the system's contextual awareness. This future work will allow for more precise action recognition and adaptive responses based on the driver's specific environment.

- **Dataset expansion with more participants:** Expanding the dataset to include a broader range of participants is essential for improving the generalizability of the model. This includes drivers of different ages, genders, and driving habits to ensure the system is robust across a diverse population. A larger dataset will capture a wider array of driving behaviors and scenarios, enabling the model to learn from a more comprehensive set of examples and perform better in real-world conditions.

- **Integration with the modular architecture of the car:** Future research should focus on seamlessly integrating the driver action recognition system with the modular architecture of autonomous vehicles. This involves ensuring compatibility with other vehicle systems, such as navigation, safety, and infotainment modules. Developing a unified framework that allows for efficient communication and data sharing between the driver recognition system and other vehicle modules will enhance overall vehicle

intelligence and responsiveness.

- **Real-world deployment and testing:** Extensive testing in real-world driving conditions is necessary to validate the system's performance and reliability. This includes running with the vehicle moving and handling various driving scenarios, weather conditions, and traffic environments. Implementing continuous learning mechanisms will allow the system to adapt and improve over time based on new data collected from real-world usage.

- **Personalized models:** Developing personalized models that adapt to individual drivers' profiles and preferences can further enhance the system's effectiveness. This involves using machine learning techniques to learn and adapt to the unique driving styles and behaviors of different users. Also creating adaptive user interfaces that respond to individual driver needs and preferences will improve the overall user experience and safety.

- **Regulatory and ethical considerations:** Addressing regulatory and ethical issues related to data privacy and consent is crucial. Future research should ensure that the system complies with data protection regulations and incorporates mechanisms for obtaining and managing user consent. Exploring the ethical implications of using surveillance technologies in autonomous vehicles will be important to ensure that these systems are used responsibly and do not infringe on drivers' privacy or rights.

References

Beckmann, P., Kegler, M., Saltini, H., & Cernak, M. (2019). Speech-vgg: A deep feature extractor for speech processing. *ArXiv*, *abs/1910.09909*. Retrieved from `https://api.semanticscholar.org/CorpusID:204823939`

Biewald, L. (2020). *Experiment tracking with weights and biases.* Retrieved from `https://www.wandb.com/` (Software available from wandb.com)

Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E., & Sheikh, Y. (2019). *Openpose: Realtime multi-person 2d pose estimation using part affinity fields.*

Chen, J., Yang, W., Liu, C., & Yao, L. (2021). A data augmentation method for skeleton-based action recognition with relative features. *Applied Sciences*, *11*(23). Retrieved from `https://www.mdpi.com/2076-3417/11/23/11481` doi: 10.3390/app112311481

Do, J., & Kim, M. (2024). *Skateformer: Skeletal-temporal transformer for human action recognition.*

Falcon, W. (2019, March). *PyTorch Lightning.* Retrieved from `https://github.com/Lightning-AI/lightning` doi: 10.5281/zenodo.3828935

Feichtenhofer, C., Fan, H., Malik, J., & He, K. (2019). *Slowfast networks for video recognition.*

for Disease Control, C., & Prevention. (n.d.). *Distracted driving.* `https://www.cdc.gov/transportationsafety/distracted_driving/index.html`. ((Accessed on 01/17/2024))

for Transport, D. G. (2022). *Road safety thematic report – driver distraction* (Vol. European Road Safety Observatory). European Commission.

Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., ... Pang, R. (2020). *Conformer: Convolution-augmented transformer for speech recognition.*

Guo, F., Jin, T., Zhu, S., Xi, X., Wang, W., Meng, Q., ... Zhu, J. (2023). B2c-afm: Bi-directional co-temporal and cross-spatial attention fusion model for human action recognition. *IEEE Transactions on Image Processing*, *32*, 4989-5003. doi: 10.1109/TIP.2023.3308750

Han, W., Zhang, Z., Zhang, Y., Yu, J., Chiu, C.-C., Qin, J., ... Wu, Y. (2020). *Contextnet: Improving convolutional neural networks for automatic speech recognition with global context.*

Hari, C., & Sankaran, P. (2021). Driver distraction analysis using face pose cues. *Expert Systems with Applications*, *179*, 115036. Retrieved from https://www.sciencedirect.com/science/article/pii/S0957417421004772 doi: https://doi.org/10.1016/j.eswa.2021.115036

Holzbock, A., Tsaregorodtsev, A., Dawoud, Y., Dietmayer, K., & Belagiannis, V. (2022, June). A spatio-temporal multilayer perceptron for gesture recognition. In *2022 ieee intelligent vehicles symposium (iv)*. IEEE. Retrieved from http://dx.doi.org/10.1109/IV51971.2022.9827054 doi: 10.1109/iv51971.2022.9827054

Karita, S., Chen, N., Hayashi, T., Hori, T., Inaguma, H., Jiang, Z., ... Zhang, W. (2019, December). A comparative study on transformer vs rnn in speech applications. In *2019 ieee automatic speech recognition and understanding workshop (asru)*. IEEE. Retrieved from http://dx.doi.org/10.1109/ASRU46091.2019.9003750 doi: 10.1109/asru46091.2019.9003750

Kim, S., Gholami, A., Shaw, A., Lee, N., Mangalam, K., Malik, J., ... Keutzer, K. (2022). *Squeezeformer: An efficient transformer for automatic speech recognition.*

Kriman, S., Beliaev, S., Ginsburg, B., Huang, J., Kuchaiev, O., Lavrukhin, V., ... Zhang, Y. (2019). *Quartznet: Deep automatic speech recognition with 1d time-channel separable convolutions.*

Li, B., Dai, Y., Cheng, X., Chen, H., Lin, Y., & He, M. (2017). Skeleton based action recognition using translation-scale invariant image mapping and multi-scale deep cnn. In *2017 ieee international conference on multimedia & expo workshops (icmew)* (p. 601-604). doi: 10.1109/ICMEW.2017.8026282

Li, J. (2022). *Recent advances in end-to-end automatic speech recognition.*

Li, J., Lavrukhin, V., Ginsburg, B., Leary, R., Kuchaiev, O., Cohen, J. M., ... Gadde, R. T. (2019). *Jasper: An end-to-end convolutional neural acoustic model.*

Liu, Z., Ning, J., Cao, Y., Wei, Y., Zhang, Z., Lin, S., & Hu, H. (2021). *Video swin transformer.*

Majumdar, S., Balam, J., Hrinchuk, O., Lavrukhin, V., Noroozi, V., & Ginsburg, B. (2021). *Citrinet: Closing the gap between non-autoregressive and autoregressive end-to-end models for automatic speech recognition.*

Martin, M., Lerch, D., & Voit, M. (2023). Viewpoint invariant 3d driver body pose-based activity recognition. In *2023 ieee intelligent vehicles symposium (iv)* (p. 1-6). doi: 10.1109/IV55152.2023.10186682

Martin, M., Popp, J., Anneken, M., Voit, M., & Stiefelhagen, R. (2018). Body pose and context information for driver secondary task detection. In *2018 ieee intelligent vehicles symposium (iv)* (p. 2015-2021). doi: 10.1109/IVS.2018.8500523

Martin, M., Roitberg, A., Haurilet, M., Horne, M., Reiß, S., Voit, M., & Stiefelhagen, R. (2019). Drive&act: A multi-modal dataset for fine-grained driver behavior recognition in autonomous vehicles. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2801-2810. Retrieved from `https://api.semanticscholar.org/CorpusID:201624289`

Martin, M., Voit, M., & Stiefelhagen, R. (2020). Dynamic interaction graphs for driver activity recognition. In *2020 ieee 23rd international conference on intelligent transportation systems (itsc)* (p. 1-7). doi: 10.1109/ITSC45102.2020.9294520

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc. Retrieved from `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`

Qiu, Z., Yao, T., & Mei, T. (2017). *Learning spatio-temporal representation with pseudo-3d residual networks.*

Sak, H., Senior, A., & Beaufays, F. (2014). *Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition.*

Sun, Z., Ke, Q., Rahmani, H., Bennamoun, M., Wang, G., & Liu, J. (2022). Human action recognition from various data modalities: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–20. Retrieved from `http://dx.doi.org/10.1109/TPAMI.2022.3183112` doi: 10.1109/tpami.2022.3183112

Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., ... Dosovitskiy, A. (2021). *Mlp-mixer: An all-mlp architecture for vision.*

Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2015). *Learning spatiotemporal features with 3d convolutional networks.*

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Wang, H., & Wang, L. (2017). *Modeling temporal dynamics and spatial configurations of actions using two-stream recurrent neural networks.*

Weyers, P., Schiebener, D., & Kummert, A. (2019). Action and object interaction recognition for driver activity classification. In *2019 ieee intelligent transportation systems conference (itsc)* (p. 4336-4341). doi: 10.1109/ITSC.2019.8917139

Wu, M., Zhang, X., Shen, L., & Yu, H. (2021). Pose-aware multi-feature fusion network for driver distraction recognition. In *2020 25th international conference on pattern recognition (icpr)* (p. 1228-1235). doi: 10.1109/ICPR48806.2021.9413337

Xiang, W., Li, C., Zhou, Y., Wang, B., & Zhang, L. (2023). *Generative action description prompts for skeleton-based action recognition.*

Yan, H., Liu, Y., Wei, Y., Li, Z., Li, G., & Lin, L. (2023). *Skeletonmae: Graph-based masked autoencoder for skeleton sequence pre-training.*

Yan, S., Xiong, Y., & Lin, D. (2018). *Spatial temporal graph convolutional networks for skeleton-based action recognition.*

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., & Yoo, Y. (2019). *Cutmix: Regularization strategy to train strong classifiers with localizable features.*