

École polytechnique de Louvain

Predicting car collisions based on footage recorded by dashboard cameras

Participating in the Nexar Dashcam Crash Prediction Challenge on Kaggle

Author: **Siénou LAMIEN**
Supervisor: **Charles PECHEUR**
Readers: **Hélène VERHAEGHE, Benoît RONVAL**
Academic year 2025–2026
Master [120] in Computer Science

Disclaimer

The writing of this thesis made use of generative AI tools and DeepL exclusively to improve the clarity of the report, organization and readability. Their use was limited mainly to rephrasing sentences for fluency and translating sections from French to English. All analyses, experiments and conclusions remain entirely the result of my own work.

Abstract

Predicting traffic accidents is a longstanding challenge with major implications for road safety and the development of advanced driver-assistance systems (ADAS). With approximately 1.19 million road deaths worldwide each year, the ability to predict imminent collisions from dashboard camera footage represents a crucial step toward safer and more autonomous vehicles. This thesis addresses this problem as part of the Nexar Dashcam Crash Prediction Challenge hosted on the Kaggle platform. The challenge provides a real-world dataset composed of 1500 training videos and 1344 test videos. The objective is to anticipate collisions in dashcam recordings ending 0.5, 1 or 1.5 seconds before the event occurs, with models evaluated on their ability to predict accidents as early as possible. To address this challenge, multiple deep learning architectures are studied, including hybrid CNN-LSTM models, 3D convolutional networks (3D-CNN) and video transformers (ViViT and VideoMAE). The most effective approach is built on the VideoMAE vision transformer, pre-trained in a self-supervised manner on the Kinetics-400 dataset and fine-tuned on 16-frame clips extracted from the competition videos. Performance is measured by mean Average Precision (mAP). The best configuration reaches 0.827 mAP on the public test set and 0.849 on the private test set, ranking 5th out of 237 teams in the final leaderboard.

Contents

1	Introduction	6
1.1	Objectives of the Thesis	6
1.2	Structure of the Thesis	7
2	Background	8
2.1	Convolutional Neural Networks	8
2.1.1	Architecture	8
2.2	Sequential Modeling for Video	9
2.2.1	RNN and LSTM	9
2.2.2	Hybrid CNN-LSTM Architectures	9
2.2.3	3D Convolutional Networks	10
2.3	Vision Transformers	10
2.3.1	Fundamental principle	11
2.3.2	Architecture	11
2.3.3	Benefits for vision	13
2.4	Video Transformers	14
2.4.1	Temporal challenges	14
2.4.2	Video Vision Transformer	14
2.4.3	VideoMAE	16
2.5	The Nexar Dataset and Evaluation Framework	18
2.5.1	Background of the Competition	18
2.5.2	Dataset	18
2.5.3	Challenges of the dataset	22
3	Related Work	23
3.1	Solutions in the literature	23
3.1.1	Foundational Work	23
3.1.2	Relational Approaches	23
3.1.3	Reinforcement Learning and Human Cognition	24
3.1.4	Transformers	24
3.2	Solutions from the Challenge	24

3.2.1	Data-centric approach (winner)	24
3.2.2	VideoMAEv2-based approach (2nd)	25
3.2.3	Two Frames Is All You Need (3rd)	25
3.2.4	Multimodal approach (4th)	26
3.2.5	Positioning of this work	26
4	Method	27
4.1	Preprocessing	27
4.1.1	Clip generation strategy	27
4.2	Data loading and augmentation	28
4.2.1	Spatial augmentation	28
4.2.2	Temporal augmentation	28
4.2.3	Validation and test mode	29
4.3	Model	29
4.3.1	Backbone	29
4.3.2	Classification head	30
4.3.3	Temperature scaling	30
4.3.4	Label smoothing	30
4.4	Training objective	31
4.5	Training setup	31
4.6	Evaluation module	32
4.6.1	Test-time protocol	32
4.6.2	Model ensemble	32
4.7	Hyperparameter optimization	33
5	Experiments and results	35
5.1	Experimental setup	35
5.1.1	Hardware and software	35
5.1.2	Validation protocols	36
5.2	Preprocessing strategies	36
5.2.1	Strategy 1: sampling within the alert interval	37
5.2.2	Strategy 2: non-overlapping sliding windows	37
5.2.3	Strategy 3: test-aligned clip sampling	37
5.2.4	Strategy 4: uniform frame subsampling	38
5.3	Video level	38
5.3.1	ViViT results	38
5.3.2	VideoMAE results	46
5.3.3	Hybrid approach and 3D-CNN	52
5.4	Frame level	54
5.4.1	ViViT results	54
5.5	Ensemble learning	58

5.5.1	Aggregation strategy	58
5.5.2	Results	59
5.5.3	Analysis	59
6	Future work	62
6.1	Frame-level approach and hybrid architectures	62
6.2	Object-aware feature extraction	63
6.3	Vision-language models	63
6.4	Reducing the false positive rate	63
7	Conclusion	64
	Bibliography	65

Chapter 1

Introduction

The automobile is an essential mode of transportation whose use continues to grow worldwide, with the global fleet now estimated at approximately 1.6 billion vehicles. According to the World Health Organization, around 1.19 million people lose their lives each year in traffic accidents [1]. Although this figure is stabilizing, road safety remains a major global crisis.

Authorities and manufacturers have committed to addressing this phenomenon. In particular, the European Union General Safety Regulation II, effective July 7, 2024, imposes the mandatory installation of advanced driver-assistance systems (ADAS) on all vehicles sold in Europe. ADAS represent a major step toward fully autonomous driving and have demonstrably reduced traffic accidents [2]–[4]. This progress is enabled by the widespread adoption of onboard cameras, lidar and radar, which cover a wider and more varied field of view than human vision. Early detection of imminent traffic accidents provides considerable added value to such systems, which continuously analyze dashboard camera footage to anticipate potential collisions involving the vehicle or occurring nearby.

This is the context of the Nexar Kaggle competition, which aims to develop models that predict traffic accidents from in-vehicle camera footage. This thesis presents our approach to the challenge. Three major families of architectures are examined: hybrid CNN-LSTM models, 3D CNNs and transformer-based models (ViViT and VideoMAE). Two distinct supervision strategies are also considered: a video-level approach, in which the target is assigned to the entire clip and a frame-level approach, in which a target is assigned to each individual frame.

1.1 Objectives of the Thesis

The objectives of this thesis are to analyze the problem of accident prediction from video data and to develop a solution based on deep learning techniques. Rather than

aiming solely for a top leaderboard position, the thesis conducts an experimental study of computer vision and deep learning methods for crash prediction, focusing on architectural choices that compare video Transformers with hybrid approaches and 3D CNNs. A complete training and evaluation pipeline is developed, building on established models from the literature and adapting their components to the specificities of accident prediction. The final approach is evaluated on the public and private test sets of the Nexar Dashcam Crash Prediction Challenge and additional experiments analyze its behavior across different time-to-event intervals.

1.2 Structure of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 provides the necessary background, first reviewing the deep learning architectures used as building blocks, then presenting the Nexar dataset and the evaluation framework. Chapter 3 reviews the state of the art in accident prediction. Chapter 4 describes the final methodology. Chapter 5 details the learning strategies, the experimental setup and analyzes the results. Chapter 6 discusses directions for future work and Chapter 7 concludes.

Chapter 2

Background

Video sequence analysis for event prediction requires architectures capable of capturing both spatial features and temporal dependencies. This chapter first reviews the deep learning building blocks used in this work, from convolutional neural networks to sequential models and vision Transformers, then introduces the Nexar dataset and the evaluation framework that define the problem.

2.1 Convolutional Neural Networks

Convolutional neural networks have long dominated image processing thanks to their ability to extract visual features. Although Transformers have outperformed them on many tasks, CNNs remain widely used, particularly for their computational efficiency.

2.1.1 Architecture

A traditional CNN consists of several types of layers applied successively.

Convolutional layers These form the core of CNNs. They perform a convolution operation, which slides a small kernel across the input image. Each filter produces a feature map that represents the presence of a specific pattern in different regions of the image.

Pooling layers Aggregation operations reduce the spatial dimension by combining information from local areas. This helps reduce computational complexity while preserving the most distinctive features.

Fully connected layers A fully connected layer processes a flattened input in which each neuron is connected to all neurons in the previous layer. They are typically placed at the end of the network, where they combine the features extracted by the convolutional layers to perform the final task.

Limitations of CNNs for video processing Despite their effectiveness on static images, CNNs face two main limitations for accident prediction from video sequences. First, their receptive field is local: convolutions mainly capture nearby patterns, which makes long-range dependencies difficult to model. Second, 2D CNNs do not capture the temporal dimension of videos.

2.2 Sequential Modeling for Video

Beyond the spatial structure captured by convolutional neural networks (CNNs), predicting events in a video requires modeling how visual content evolves over time. This section briefly reviews the different categories of approaches used in this work to handle the temporal dimension: recurrent architectures, hybrid CNN-LSTM pipelines, and 3D convolutional networks.

2.2.1 RNN and LSTM

Recurrent neural networks (RNNs) deal with sequential data by preserving a hidden state that is updated at each time step according to the current input and the previous hidden state. This recursive formulation allows the network to remember past observations and capture dependencies over time. Traditional RNNs, however, suffer from vanishing and exploding gradients, which prevents them from effectively learning long-term relationships.

The Long Short-Term Memory network [5] was introduced to address this limitation. An LSTM cell replaces the simple recurrence of an RNN with a gated mechanism composed of three gates (input, forget and output) that regulate the flow of information into and out of a dedicated memory cell. This explicit memory mechanism enables LSTMs to selectively preserve relevant information across many time steps, which makes them well-suited for modeling temporal patterns in video sequences.

2.2.2 Hybrid CNN-LSTM Architectures

This family of models separates spatial and temporal modeling into two distinct components. In a hybrid CNN-LSTM architecture, each frame of the video is first processed by a CNN, which produces a feature vector representing the frame's

content. The resulting sequence of feature vectors is then passed to an LSTM that models the temporal dependencies between frames and returns a representation of the entire clip.

This two-stage pipeline has been widely used in early work on video processing. Its main advantage is that it leverages the strong spatial inductive bias of CNNs while delegating temporal understanding to a dedicated recurrent network. The main drawback is that spatial and temporal features are learned independently. The LSTM also processes frames one at a time, which makes parallelization difficult and long-range dependencies remain difficult to model despite the gating mechanism. These limitations motivate the alternative architectures discussed below.

2.2.3 3D Convolutional Networks

A different strategy to capture spatio-temporal patterns is to extend the convolution operation itself to the time dimension. While a 2D convolution applies a kernel of size $k_h \times k_w$ over the spatial dimensions of a single image, a 3D convolution uses a kernel of size $k_t \times k_h \times k_w$ that operates jointly over a small temporal window and the two spatial dimensions of the input clip. As a result, a single 3D convolutional layer can detect motion patterns, such as a moving edge or an approaching object, directly from raw pixels, without requiring an explicit recurrent module.

Architectures such as C3D [6] and I3D [7] have shown that 3D CNNs can significantly outperform 2D CNN-LSTM pipelines on large-scale video benchmarks, particularly when pre-trained on large datasets such as Kinetics [8]. Their main drawback is computational cost: the additional temporal dimension substantially increases the number of parameters and the amount of computation per forward pass, which constrains both training and inference. This has motivated subsequent work on factorizing 3D convolutions into separate spatial and temporal operations, as well as the attention-based architectures presented in the next section.

2.3 Vision Transformers

Transformer models have been the leading architecture in natural language processing (NLP) since the publication of *Attention Is All You Need* [9] in 2017. Originally designed for sequence modeling, they have since been successfully adapted to image processing. Introduced by Dosovitskiy et al. [10] in 2020, Vision Transformers (ViTs) showed that a transformer-only architecture (Figure 2.1), without convolutional layers, can achieve cutting-edge performance in computer vision.

2.3.1 Fundamental principle

A ViT processes an image as a sequence of patches, treated as tokens analogous to those used in NLP. Thanks to global self-attention, ViTs can capture long-range dependencies across the entire image from the very first layer, unlike CNNs. While Transformers for NLP consist of both an encoder and a decoder, ViT uses the encoder alone, followed by a classification head that produces the final prediction.

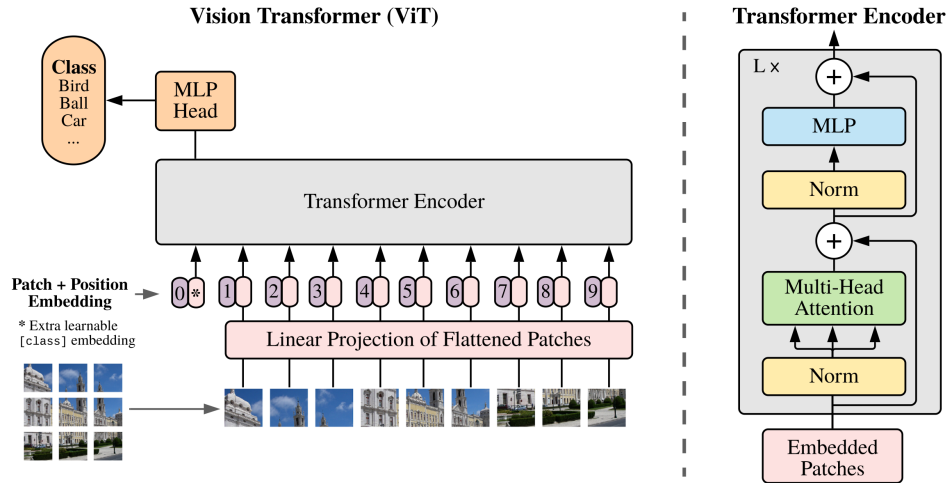


Figure 2.1: Vision Transformer architecture (Source: [10])

2.3.2 Architecture

Patch embedding The input image of size $H \times W \times C$ is divided into N non-overlapping patches of size $P \times P$ (Figure 2.2), where $N = \frac{H \times W}{P^2}$. Each patch is flattened into a $P^2 \times C$ vector and linearly projected onto a D -dimensional embedding via a learned matrix.



Figure 2.2: Splitting the image into patches (Source: [10])

Classification token (CLS) A learnable token is prepended to the sequence of patch embeddings. Through attention, it aggregates information from all patches and serves as a global representation of the image used for the final classification.

Positional encoding Patch embeddings carry no spatial information by themselves. Positional encodings are therefore added to each embedding (Figure 2.3) so that the model can recover the spatial location of each patch.

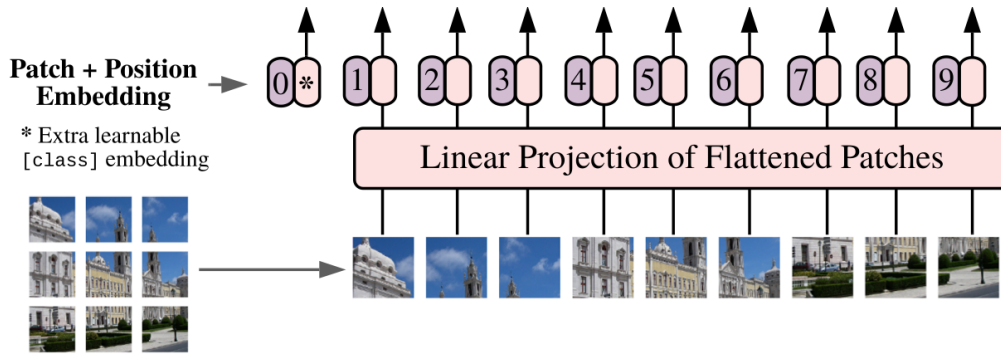


Figure 2.3: Positional encoding (Source: [10])

Transformer encoder The sequence passes through L identical Transformer encoder blocks, where L is a hyperparameter (Figure 2.1). Each block contains four components: layer normalization, which stabilizes training; multi-head self-attention, which models relationships between tokens; a multi-layer perceptron (MLP), which transforms and enriches the resulting representations; and residual

connections, which preserve the original signal and mitigate the vanishing-gradient problem.

The self-attention mechanism, introduced by Vaswani et al. [9] calculates pairwise relationships between tokens enabling the model to capture dependencies regardless of the distance. For each token, three vectors are calculated:

- **Query** (Q): what the token is looking for in others.
- **Key** (K): how the token describes itself.
- **Value** (V): the content the token contributes.

Attention is computed as:

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

where $Q = XW_Q$, $K = XW_K$ and $V = XW_V$ are the projections of the input X and W_Q , W_K and W_V are matrices learned during training. The scaling factor $\sqrt{d_k}$ helps limit the magnitude of the dot products and stabilize the gradients during backpropagation. The attention weights obtained in this way are then used to compute a weighted sum of the value vectors, which produces an enriched representation of the context for each token.

Multi-head attention applies this operation in parallel with h heads, allowing the model to attend to different types of relationships simultaneously and concatenates the outputs.

Classification head After the final encoder block, the CLS token is passed to an MLP that produces classification logits (the raw and unnormalized outputs of the model before normalization). A softmax or sigmoid function is then applied to convert these logits into probabilities.

2.3.3 Benefits for vision

- **Global attention from the first layer:** Unlike CNNs, each patch can interact immediately with all the others.
- **Scalability:** Performance improves as the size of the dataset and the model increases.
- **Flexibility:** In the absence of strong spatial inductive bias, the model learns relevant structures directly from the data.

2.4 Video Transformers

2.4.1 Temporal challenges

Extending the Vision Transformer to video introduces two new difficulties that did not arise in the image case. The first is the explosion of the number of tokens. A standard ViT divides a 224×224 image into 16×16 patches, which yields 196 tokens per frame. Applying the same tokenization to a clip of T frames produces $T \times 196$ tokens and since self-attention has quadratic complexity in the number of tokens, the resulting cost quickly becomes prohibitive. For a clip of $T = 16$ frames, the attention matrix already contains roughly ten million entries, compared to less than forty thousand for a single image.

The second challenge concerns the temporal information itself. A video is not merely a sequence of independent frames: relevant information often lies in how the content evolves from one frame to the next. An effective video architecture must therefore capture both what appears in each frame and how it evolves over time, without the computational cost becoming unmanageable. The architectures discussed in the following sections propose several strategies for solving this dilemma.

2.4.2 Video Vision Transformer

The Video Vision Transformer (ViViT), introduced by Arnab et al. [11], was one of the first works to adapt the ViT framework to video classification. It contributes on two complementary axes: how a clip is converted into a sequence of tokens and how attention is structured over these tokens (Figure 2.5).

Tokenization ViViT proposes two strategies to tokenize an input clip of T frames of size $H \times W$. The first, called uniform frame sampling, treats each frame independently: it applies a 2D patch embedding to each of the T frames in the same way as the original ViT, producing $T \times N$ tokens, where $N = (H \times W)/P^2$ and P is the spatial patch size. Temporal information is therefore not captured during tokenization itself and must be modeled entirely by the attention layers.

The second strategy, tubelet embedding, generalizes the notion of a patch to the time dimension. A tubelet is a non-overlapping 3D volume of size $t \times P \times P$ that spans several consecutive frames as well as a spatial region. Each tubelet is linearly projected into a token through a 3D convolution. With this approach, the number of tokens is reduced from $T \times N$ to $(T/t) \times N$ and each token already encodes a local spatio-temporal pattern before attention is applied. Tubelet embedding has become the standard choice in subsequent work, as it produces more compact and temporally grounded token sequences than uniform frame sampling. It is also the

strategy used in the ViViT experiments of this thesis, as well as in VideoMAE (Section 2.4.3), which inherits the same tokenization scheme.

Four variants of factorized attention Once the tokens are obtained, ViViT explores four ways of structuring attention over the spatial and temporal dimensions. The full architecture is illustrated in Figure 2.4.

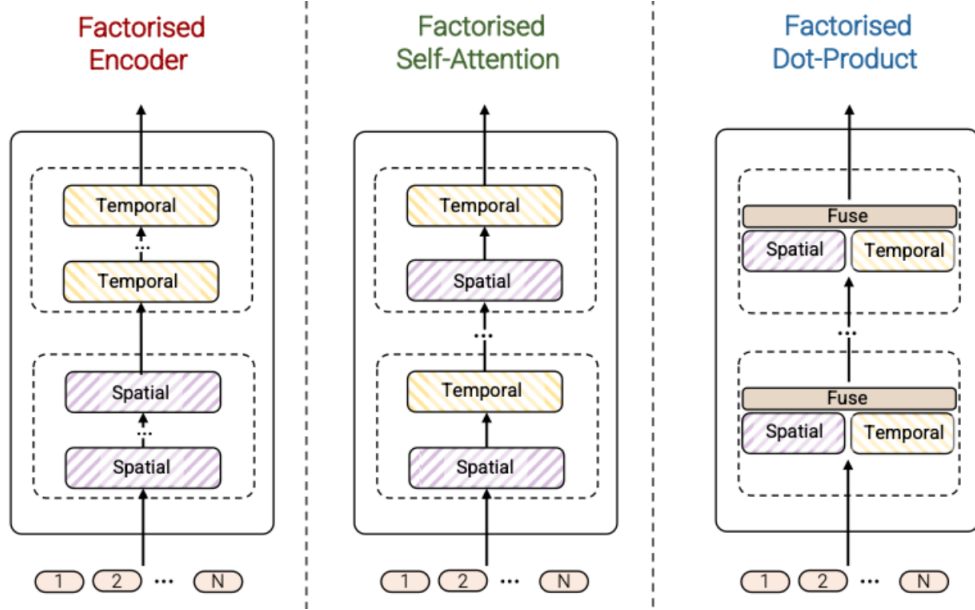


Figure 2.4: Variants of factorized attention

1. **Joint attention:** patches from all frames attend to all other patches in a single attention operation. This is the most expressive variant but also the most computationally expensive.
2. **Factorized encoder:** the architecture consists of two sequential encoders. A spatial encoder first processes each image individually to capture spatial dependencies. A temporal encoder then processes the resulting sequence to capture temporal dependencies between frames. This reduces computational load but limits interactions between frames.
3. **Factorized self-attention:** inside each Transformer block, attention is decomposed into two consecutive sub-blocks. The first is a spatial attention that operates on tokens belonging to the same frame. The second is a temporal attention that operates on tokens at the same spatial location across frames.

4. **Factorized dot-product attention:** spatial and temporal attention are computed in parallel within the same attention head, by splitting the heads into two groups: one attending over space and the other over time. The outputs are then concatenated.

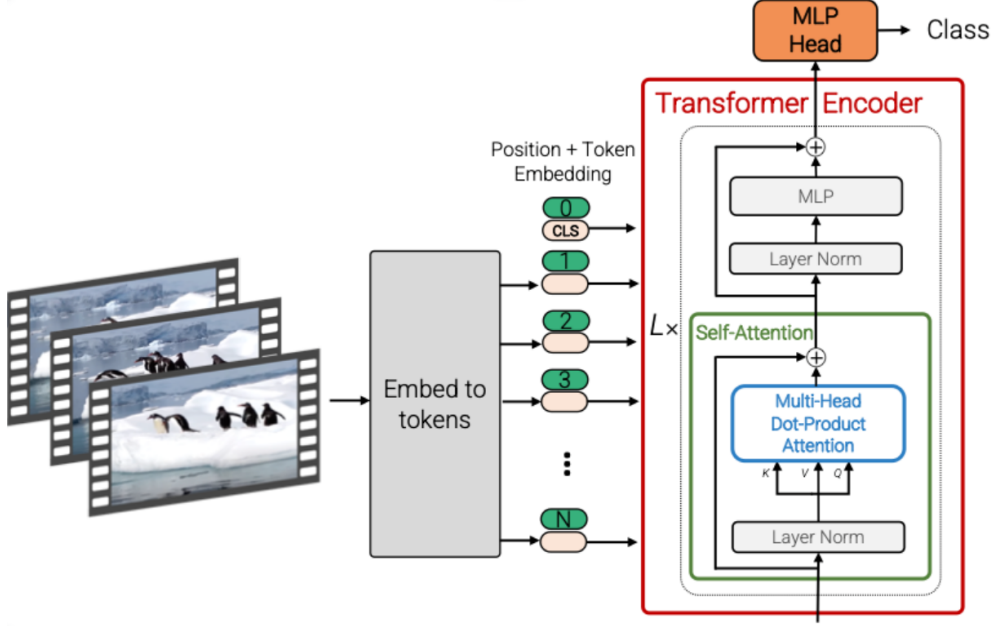


Figure 2.5: Video Vision Transformer (Source: [11])

2.4.3 VideoMAE

The Video Masked Autoencoder (VideoMAE), introduced by Tong et al. [12], is the architecture used in the final solution of this thesis. Unlike ViViT, which is trained in a supervised manner, VideoMAE relies on self-supervised pre-training: the model learns useful video representations by reconstructing artificially masked content, without requiring any labels. The full pipeline is illustrated in Figure 2.6.

Masked autoencoders The masked autoencoder paradigm was popularized in vision by the Masked Autoencoder (MAE) for images [13], itself inspired by masked language modeling in BERT [14]. The general principle is the following: a large fraction of the input tokens is removed and the model is trained to reconstruct the missing content from the remaining visible tokens. The intuition is that, to fill in the missing patches, the network must learn semantically meaningful representations rather than relying on low-level texture statistics.



Figure 2.6: Video Masked Autoencoders (Source: [12])

Tube masking versus random masking A naive extension of image MAE to video would mask patches independently in each frame, an approach known as random masking. This proves problematic, because adjacent frames are highly redundant: a patch masked in frame t can typically be reconstructed by copying the corresponding patch from frame $t - 1$ or $t + 1$, where it is likely still visible. The reconstruction task therefore becomes trivial and the model fails to learn useful temporal representations.

VideoMAE introduces tubular masking to address this issue. Instead of masking areas independently for each frame, a tubular mask removes the same spatial position across all frames in the clip, so that the masked region forms a vertical tube in the spatiotemporal volume. This forces the model to reconstruct the missing content from other spatial locations rather than from neighboring frames, which encourages the encoder to capture true spatiotemporal context rather than exploiting redundancy across frames.

Self-supervised pre-training on Kinetics-400 In this work, the VideoMAE model is initialized with weights pre-trained on the Kinetics-400 dataset [8], a large-scale action recognition benchmark containing approximately 240,000 training clips spanning 400 human action classes. Self-supervised pre-training on Kinetics-400 provides the encoder with a generic understanding of how objects move and interact in real-world video, without requiring the costly collection of labels. This pre-trained backbone is then fine-tuned on the comparatively small Nexar dataset, which would be too small to train a video Transformer from scratch. This transfer learning strategy is one of the main reasons VideoMAE is particularly well suited to the accident anticipation task considered in this thesis.

2.5 The Nexar Dataset and Evaluation Framework

2.5.1 Background of the Competition

The Nexar Dashcam Crash Prediction Challenge was hosted on Kaggle, a competition platform for data science and machine learning. The challenge was launched by Nexar Inc., a company specializing in computer vision for autonomous driving and vehicle safety, on February 10, 2025 and concluded on May 4, 2025. The competition consists of a single task: predicting whether a collision is about to occur within a given time frame, based on a short video clip from an onboard camera. The result is a probability between 0 and 1, with higher values indicating a greater likelihood of an imminent collision.

2.5.2 Dataset

The videos are high-definition clips (1280×720) lasting approximately 40 seconds at an average of 30 frames per second (FPS), providing rich temporal context. They span a wide variety of weather, scenes, lighting conditions, obstructions and unexpected traffic events (Figure 2.7). A team of reviewers categorized hundreds of thousands of videos into three groups:

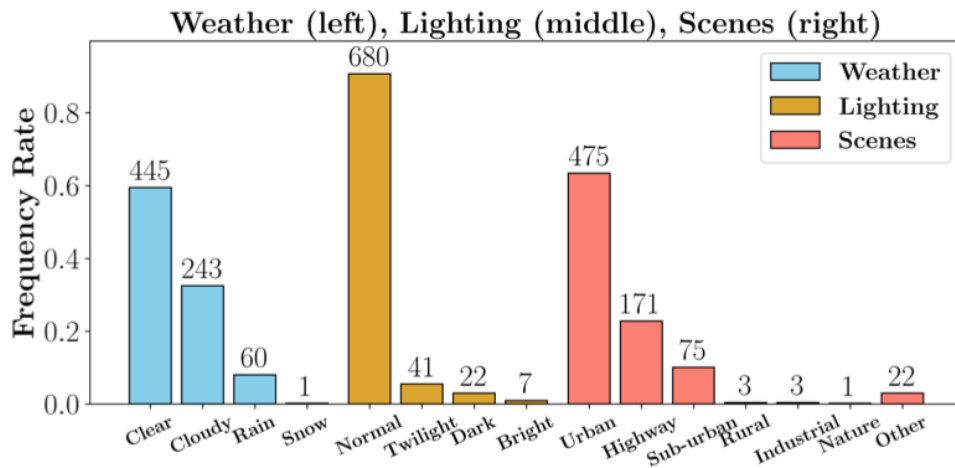


Figure 2.7: Variations in conditions in videos (Source: [15])

- **Collision:** an accident involving the ego vehicle and a third party.

- **Near-collision (or near-miss):** a critical situation between the vehicle in question and a third party, without an actual collision.
- **Normal driving:** neither a collision nor a near-miss.

Collisions and near-misses are considered critical events and are grouped as positive examples, while videos of normal driving serve as negative examples.

Training Set

The training set contains 1,500 videos: 750 negative examples and 750 positive examples (400 collisions and 350 near-collisions), forming a balanced dataset. Accidents generally occur around the middle of the video, and the average time between the moment of the collision and the first moment the accident can be detected is 1.5 seconds (Figure 2.8). Each video is associated with a binary label, and positive samples are accompanied by two additional timestamps:

- **Event time:** the moment when the collision or near-miss occurs.
- **Alert time:** the earliest moment at which an attentive person can recognize that a dangerous situation is about to occur. To reduce subjectivity, this annotation is derived from ten independent annotators.

These two timestamps divide each positive video into three intervals (Figure 2.9):

1. **Before-accident interval:** from the start of the video to the alert time. This is the earliest moment at which a driver should intervene to prevent a potential accident.
2. **Alert interval:** from the alert time to the moment of the accident. This is the last window during which anticipation is meaningful.
3. **After-accident interval:** from the moment of the accident to the end of the video.

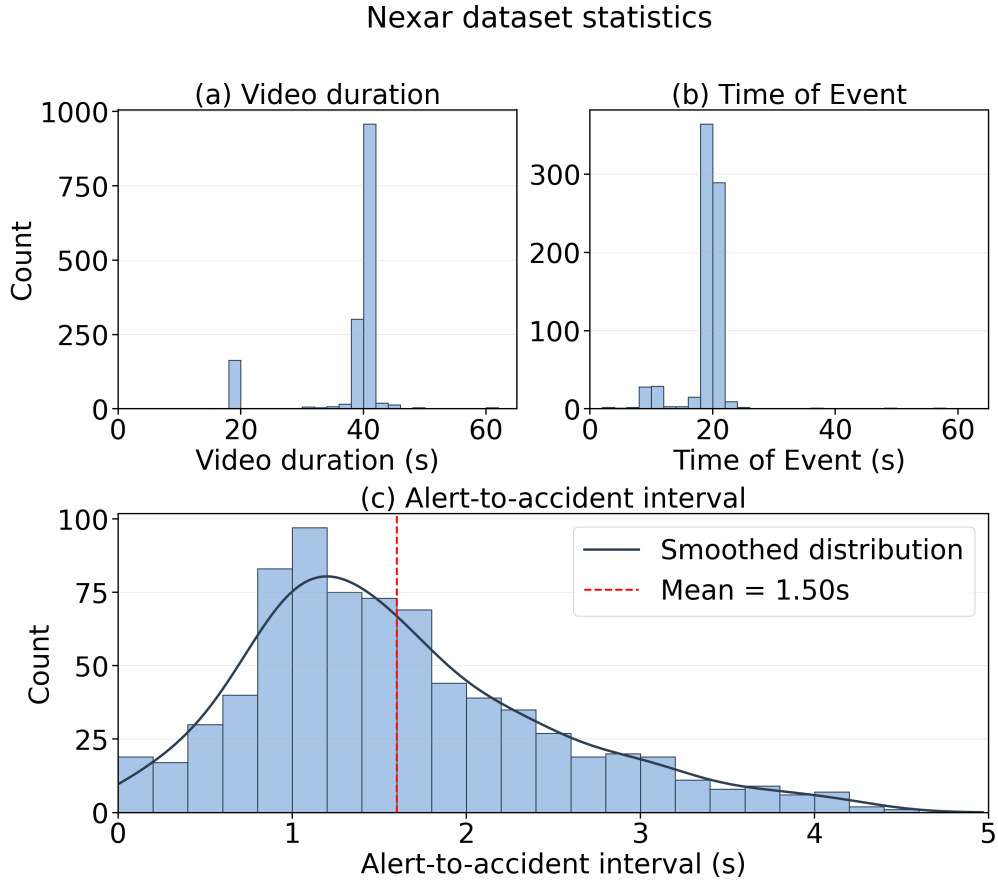


Figure 2.8: Dataset statistics (Source: [15])



Figure 2.9: Temporal structure of an accident video (Source: [15])

Test set

The test set consists of 1 344 balanced videos, each approximately 10 seconds long, truncated at a predefined point before the accident. These videos were generated

from a pool of 568 source videos: for each positive video, up to three test videos are generated by truncating the original so that it ends at $t_{\text{event}} - \Delta_{\text{TTE}}$, where $\Delta_{\text{TTE}} \in \{0.5, 1, 1.5\}$ s is the time-to-event. For negative examples, fake event times are generated by adding Gaussian noise to half of the video duration and fake alert times are produced to match the distribution of the positive cases.

The test set is divided into two balanced subsets of equal size: the public subset and the private subset. The public subset is used to evaluate submissions during the competition. The private test set is used at the end of the competition to determine the final rankings. Submissions are evaluated on the complete test set, without participants knowing which subset a given video belongs to. The partially labeled test set has been published since the contest closed (Table 2.1).

TTE	Number of videos		Total
	Public Set	Private Set	
500 ms	284	284	568
1000 ms	231	233	464
1500 ms	152	160	312
Total	667	677	1344
Class distribution			
Class	Public Set	Private Set	Total
Non-accident (0)	333	339	672
Accident (1)	334	338	672

Table 2.1: Breakdown of videos by TTE and class distribution.

Evaluation metric The challenge rewards solutions that maximize precision, recall and anticipation time simultaneously. An accident correctly detected too late is of little operational value, so anticipation matters as much as predictive accuracy. The official metric is the mean Average Precision (mAP), computed at three times-to-event: 0.5, 1 and 1.5 s. For each TTE, an Average Precision (AP) is computed from the precision-recall curve produced by varying the decision threshold on the confidence scores submitted for that TTE group. The final mAP is the average of the three resulting APs:

$$\text{mAP} = \frac{\text{AP}_{0.5} + \text{AP}_1 + \text{AP}_{1.5}}{3}, \quad \text{AP} = \sum_n (R_n - R_{n-1}) P_n,$$

where P_n and R_n are the precision and recall at the n -th threshold.

In addition to the competition metric, several complementary measures are monitored to provide a more detailed evaluation. The training loss is tracked to assess convergence and detect overfitting. The confusion matrix yields standard

metrics such as precision, recall, accuracy and F1-score. Finally, the ROC curve and its area (AUC-ROC) measure the discriminatory power of the model independently of any specific threshold.

2.5.3 Challenges of the dataset

The Nexar dataset presents several technical challenges that reflect the difficulty of predicting accidents under real-world conditions:

- **The online nature of the prediction:** unlike accident detection, which operates on complete video sequences, prediction must occur in real time, before the event takes place. An incorrect decision may either fail to prevent an accident (false negative) or trigger unnecessary preventive actions (false positive) such as sudden braking, which itself may introduce new risks.
- **The nature of the negative examples:** video recording is triggered by the inertial measurement unit (IMU) built into the camera as soon as sudden braking or sudden acceleration is detected. The excerpts of normal driving are therefore not random sequences but situations that likewise involved a strong physical reaction without ending in a collision or near-collision. The trigger rests on a physical criterion while the label reflects a semantic criterion assessed by the annotators. This is what makes the negative examples harder to filter out, since they share pronounced braking dynamics with the positive examples.
- **The subjectivity of time markers:** determining the exact moment at which an accident becomes predictable is inherently subjective. The dataset mitigates this by deriving the alert time from annotations provided by ten independent annotators.
- **Extremely short alert intervals:** a significant fraction of videos contain only a few frames between the alert and the event. This interval is much shorter than the typical human reaction time (around 0.7 seconds), making anticipation particularly difficult for computer vision models.
- **Diversity and environmental factors:** the dataset spans a wide variety of lighting and weather conditions, road types and includes camera artifacts such as reflections, motion blur and lens flare.
- **Precision-anticipation trade-off:** the model must maximize precision, recall and anticipation time simultaneously. This balance is calculated by the MAP indicator, which is designed for use across multiple TTE.

Chapter 3

Related Work

3.1 Solutions in the literature

3.1.1 Foundational Work

In 2016, Chan et al. [16] published the first paper on Traffic Accident Anticipation (TAA). They defined the task, introduced standard evaluation metrics (mAP and Average Time-to-Collision, ATTC) and proposed the first large-scale dashcam-based dataset, the Dashcam Accident Dataset (DAD). Their architecture combines an RNN with a Dynamic Spatial Attention (DSA) mechanism applied to objects detected frame by frame; their exponential loss, which penalizes late predictions more heavily, has since become a benchmark in the field. Suzuki et al. [17] extended this line of work in 2018 with AdaLEA, an adaptive loss inspired by curriculum learning that dynamically adjusts the late-prediction penalty over the course of training. Architecturally, they replaced LSTMs with quasi-recurrent neural networks (QRNNs), which are faster and better suited for extracting motion cues thanks to their temporal convolutions.

3.1.2 Relational Approaches

A paradigm shift occurred with the introduction of graph neural networks (GNNs) in TAA. Bao et al. [18] published UString, which explicitly models the relationships between agents using a graph convolutional network (GCN) whose adjacency matrix is based on the spatial distances between detected objects. An RNN then tracks the temporal evolution of this relational graph. The most significant contribution, however, remains the integration of Bayesian neural networks, which allow the model to estimate its own uncertainty and gradually gain confidence as it observes more frames. Malawade et al. [19] take this approach further with spatio-temporal encodings of scene graphs that explicitly encode semantic relationships

between vehicles, while Wang et al. [20] introduce GSC (Graph and Spatio-temporal Continuity), which combines graph representations with continuity constraints to produce more stable predictions over time.

3.1.3 Reinforcement Learning and Human Cognition

Bao et al. [21] propose DRIVE, which reformulates TAA as a Markov decision process solved using deep reinforcement learning. The agent simultaneously predicts a crash score and the next gaze point by dynamically merging top-down and bottom-up attention. This visual attention mechanism gives the model intrinsic explainability, since attention is not a post hoc justification but the direct cause of the decision. In a similar vein, Fang et al. [22] introduce CAP (Cognitive Accident Prediction), which fuses BERT-encoded textual descriptions with visual features via cross-attention, along with driver attention maps, during training. This transfers semantic knowledge to the visual path, which can then be used alone at inference. The model also relies on a GCN applied to a 64-token semantic graph constructed from the text-vision fusion.

3.1.4 Transformers

With the rise of Vision Transformers, several recent studies have moved away from recurrent architectures in favor of self-attention. Al-Thani et al. [23] proposed the Traffic Transformer, the first direct adaptation of the Transformer architecture to TAA. Patera et al. [24] introduced a spatio-temporally dilated neighborhood attention mechanism that extends the receptive field at limited computational cost. Kumamoto et al. [25] presented AAT-DA, a fully Transformer-based architecture combining a spatial Transformer that processes each frame and its detected objects, a causal temporal Transformer that considers only past and present frames to support proactive anticipation and a spatial attention module that fuses predicted driver attention with the relative proximity between objects; the system reaches state-of-the-art performance on DAD and CCD. Finally, Song et al. [26] combine UString’s relational approach with attention-based mechanisms in a graph network augmented by dynamic attention.

3.2 Solutions from the Challenge

3.2.1 Data-centric approach (winner)

The solution that won first place in the Nexar Dashcam Crash Prediction Challenge, proposed by Paul Endresen [27], adopts a strictly data-centric approach, retaining

an off-the-shelf multiscale vision transformer and focusing on iteratively improving the training data. This strategy reached a private score of 0.872 mAP, without any modification of the architecture or hyperparameter tuning. The pipeline relies on three principles. First, preprocessing: the videos are resized to 256×256 , all frames following the event are excluded from the training set and the model is fed a window of 16 consecutive frames (with step size 4) covering approximately 2 seconds at 30 FPS. Second, model errors are systematically inspected using the 3LC framework: predictions on the training set are visualized in embedding space, which reveals clusters of errors concentrated around the alert time, suggesting that the model struggles to distinguish frames immediately before and after the onset of the alert window. Third, targeted modifications were applied to the training set: removal of frames immediately preceding the impact, rebalancing of frames within the alert window to compensate for their under-representation and doubling the weight of frames located outside the alert window but taken from videos containing an event, in order to force the model to learn subtle distinctions within a single video.

3.2.2 VideoMAEv2-based approach (2nd)

This solution [28], ranked second on the private leaderboard with 0.864 mAP, formalizes the task as a binary classification problem at the video clip level and compares several architectures. CNN-LSTM and CNN-Attention baselines reach an AP between 0.7 and 0.8 and are outperformed by VideoMAEv2-giant. The author does not use the human-annotated alert time but instead labels as positive any sliding window ending within 1.5 seconds of the event. This threshold is chosen because it encompasses the three TTE values of the official metric; a retrospective experiment strictly following the official annotations yielded slightly lower results. The classification head is reduced to a simple linear layer followed by dropout and temperature scaling is applied to the logits during training to smooth the predicted probability distributions. At inference, only the last two seconds of each test video are fed to the model.

3.2.3 Two Frames Is All You Need (3rd)

This solution [29], ranked third with a private score of 0.855 and a public score of 0.853, reformulates the problem as an image classification task using only two consecutive frames. YOLO-v8 generates a mask isolating the cars and Farneback optical flow encodes the direction and magnitude of motion, forming a three-channel composite image. The architecture combines two ResNet-18 networks pre-trained on ImageNet: one for the mask and the optical flow, the other for the masked

RGB image. At inference, the predictions on the last three frames are weighted (approximately 50/30/20%) and averaged across an ensemble of two models.

3.2.4 Multimodal approach (4th)

This solution [30], ranked fourth with 0.853 on the private set (0.766 on the public set), takes a different approach: rather than letting a pre-trained backbone implicitly learn spatio-temporal cues, it explicitly injects them into the model’s inputs. The author hypothesizes that a collision is fundamentally defined by three cognitively accessible elements: the proximity between vehicles, the three-dimensional structure of the scene and relative lateral motion. These three aspects are translated into corresponding input modalities for a CNN-RNN architecture based on ResNet-18 used as a per-frame feature extractor. Depth is estimated using Depth Anything, motion using a dense optical flow computed with OpenCV and vehicle proximity using YOLO segmentation. These signals are stacked into a composite representation that replaces the standard RGB input of the network.

3.2.5 Positioning of this work

Two elements of the present work are directly inspired by the existing solutions. The use of temperature scaling is adapted from [28] and proved beneficial in our experiments, increasing the mAP by approximately +0.02 compared to an equivalent model without temperature scaling. The idea of sizing the preprocessing window to cover all three TTE values of the evaluation metric is inspired by [28], whose effective window can extend up to 2.5 seconds. The exact window size of 3 seconds adopted here is slightly larger to provide a safety margin against the FPS variability of the dataset, which ranges from 23 to 31 FPS.

Our approach differs in one key respect: we apply the same fixed-window preprocessing at training and at evaluation time, whereas previous work uses different temporal conditions for each phase. Aligning the two reduces the train/test distribution shift and is, to our knowledge, the most distinctive choice of this work. During training we add temporal augmentation, jittering the sampling window by a few frames for robustness to small temporal shifts; at inference we apply test-time augmentation (TTA), averaging predictions over several window lengths. Finally, the system is an ensemble of three VideoMAE models pre-trained on Kinetics-400, combined by a weighted average of their logits. Together with temperature scaling, this consistent preprocessing, the ensemble and the temporal augmentation form the core of the proposed pipeline. Two further directions explored by other participants (explicit object-aware feature extraction [29] and the injection of optical-flow magnitude as a motion cue [30]) were not investigated here but are discussed as future directions in Chapter 6.

Chapter 4

Method

This chapter describes the proposed solution for accident anticipation from dashcam video in the Nexar Dashcam Crash Prediction Challenge. The system is organized as a sequential pipeline composed of four modules: preprocessing, data loading, modeling and evaluation. The preprocessing module generates fixed-length video clips aligned with the TTE convention used at test time. The data-loading module handles frame extraction and applies temporal and spatial augmentation during training. The modeling module uses a VideoMAE-Large backbone with a custom binary classification head, regularized by temperature scaling and label smoothing. The evaluation module computes classification metrics and produces the associated visualizations. The optimal hyperparameters for this pipeline are found via Bayesian optimization. The values retained for the final ensemble are reported in Chapter 5.

4.1 Preprocessing

4.1.1 Clip generation strategy

The competition test set provides videos that are truncated at fixed TTE values before the event. To align the training distribution with the evaluation conditions, each training video is truncated at three TTE values:

$$t_{\text{cut}} = t_{\text{event}} - \Delta_{\text{TTE}}, \quad \Delta_{\text{TTE}} \in \{0.5, 1.0, 1.5\} \text{ seconds.}$$

For negative videos, the same procedure is applied using a fake event time as reference.

Images are sampled from a three-second window preceding t_{cut} for each truncated video. The duration of this range is determined by the fluctuation in frame rate across the dataset. Although the dataset paper reports an average of 30 frames per second, the frame rate per video measured using OpenCV ranges from 23 to

31 frames per second. Rounding up to three seconds guarantees that the window covers enough temporal context even on the fastest videos (Chapter 5). Within this window, 16 frames are selected at regular intervals. The value of 16 is imposed by the VideoMAE backbone which expects a fixed number of frames per clip.

Each clip inherits the label of its source video. The full procedure yields a balanced training set of approximately 4 500 clips. For each clip, the selected frame indices and the associated label are stored in a JSON metadata file which avoids recomputing the sampling indices at every training iteration.

4.2 Data loading and augmentation

Data augmentation aims to improve the robustness and generalization of the model without requiring additional data. However, augmentations that are too aggressive can erase the subtle visual cues that the model needs to detect. The augmentation pipeline used in this final model is therefore deliberately conservative and combines a spatial component and a temporal component. Before any augmentation, each frame is resized to 224×224 pixels and normalized using the ImageNet mean and standard deviation, matching the statistics on which the backbone was pre-trained.

4.2.1 Spatial augmentation

Spatial augmentation tends to slightly reduce recall but improves precision and overall accuracy, producing a more conservative classifier. It is applied only to a subset of the ensemble models (Section 4.6.2). Two operations are used during training:

- **Color jittering:** the brightness and the contrast of each frame are randomly modified within $\pm 20\%$. This exposes the model to a variety of lighting conditions which is relevant for dashcam footage captured at different times of the day.
- **Random horizontal flip:** Each sequence is rotated horizontally with a 50% probability which simulates driving on the opposite side of the road.

4.2.2 Temporal augmentation

In addition to spatial transformations, the pipeline applies an augmentation specifically designed for video data: temporal jittering. For every frame index selected by the preprocessing module, a random offset drawn uniformly in $\{-3, \dots, +3\}$ frames is added before reading the frame. This perturbation has three intended effects:

- it makes the model robust to small variations in the effective sampling rate which is relevant given the FPS variability of the dataset.
- this discourages the model from relying on exact temporal alignment of the input data. Instead, it favors representations that capture movement patterns rather than the absolute positions of the images.
- it implicitly increases the diversity of training sequences without requiring additional data.

Temporal jittering is applied to all ensemble models and proved to be the most reliable form of augmentation in our experiments (Chapter 5).

4.2.3 Validation and test mode

At validation and test time, the augmentation pipeline is reduced to the strict minimum: resize to 224×224 , tensor conversion and ImageNet normalization. No color jittering, horizontal flip, or temporal jittering is applied. This guarantees that the reported metrics reflect a deterministic and reproducible evaluation of the model.

4.3 Model

The final model is built on the VideoMAE backbone, fine-tuned on the Nexar dataset with a custom binary classification head, cross-entropy loss and two complementary regularization mechanisms: temperature scaling and label smoothing.

4.3.1 Backbone

The backbone is the VideoMAE-Large model pre-trained for 1 600 epochs in a self-supervised manner and subsequently fine-tuned on Kinetics-400 for action recognition (MCG-NJU/videomae-large-finetuned-kinetics [12]). VideoMAE extends the masked-autoencoder paradigm to the temporal dimension which makes it data-efficient on video tasks. It expects exactly 16 frames per clip which the preprocessing module enforces. The choice of the Large variant rather than Base or Huge is justified empirically in Chapter 5: scaling from Base to Large yields a clear improvement but Huge offers only a marginal additional gain at significantly higher computational cost.

4.3.2 Classification head

The original action-recognition head of VideoMAE is replaced by a custom binary classifier composed of two layers applied in sequence:

1. a dropout layer with a configurable rate which regularizes the network by randomly zeroing a fraction of the encoder features at each training step.
2. a linear projection from the encoder’s hidden dimension to a two-dimensional output (accident vs. no accident).

The dropout layer mitigates overfitting given the limited size of the dataset while the linear projection produces the logits used by the loss function.

4.3.3 Temperature scaling

A temperature-scaling mechanism is applied to the logits during training. Given the raw logit vector $z \in \mathbb{R}^2$ and a temperature $T > 1$, the scaled logits are obtained by computing z/T before the softmax function. This reduces the magnitude of the differences between class scores and produces a less irregular probability distribution during training.

We emphasize that this approach differs from the traditional, post-training use of temperature tuning for calibration [31], where a single temperature is tuned on a validation set and applied at the time of inference. Here, the temperature is instead treated as a training hyperparameter: dividing the logits by T rescales the gradient on the output logits by a factor of $1/T$, thereby mitigating the rate at which the network diverges in its class scores. This acts as a regulator against excessive growth of the logits.

During inference and evaluation, the logits are used as-is, so that the reported probabilities reflect the model’s natural behavior. The value of T is included in Optuna’s search space, and the values selected for each ensemble model are presented in Section 5.

4.3.4 Label smoothing

Label smoothing complements temperature scaling on the target side. Given a one-hot target y and a smoothing factor $\varepsilon \in [0, 1]$, the smoothed target is defined as

$$\tilde{y} = (1 - \varepsilon) y + \frac{\varepsilon}{C},$$

where $C = 2$ is the number of classes. With $\varepsilon = 0.1$ for instance, the correct class receives a target probability of 0.95 and the other class 0.05.

Without smoothing, cross-entropy encourages the model to push the logit of the correct class toward $+\infty$ and the others toward $-\infty$ which is a recipe for overfitting on a small dataset. Label smoothing prevents this by injecting a small amount of uncertainty in the targets and acts as an additional regularizer that complements the calibration effect of temperature scaling.

Label smoothing and dropout are two complementary regularization techniques. Although they operate at different levels of the network, they share the same goal: reducing overconfidence and limiting overfitting. Excessive cumulative regularization from both can degrade performance through underfitting. Our Optuna study reveals two distinct high-performing regimes: one with a high label smoothing ($\varepsilon \approx 0.30$) and a low dropout (≈ 0.05) and another with a moderate label smoothing ($\varepsilon \approx 0.15$) and a slightly higher learning rate. Both regimes are exploited in the final ensemble.

4.4 Training objective

The model is trained with the standard cross-entropy (CE) loss combined with the temperature scaling and label smoothing mechanisms introduced above. For a clip with smoothed target $\tilde{y}$ and raw logits z , the standard cross-entropy is defined as

$$\mathcal{L}_{\text{CE}}(y, \hat{y}) = - \sum_{c=1}^C y_c \log(\hat{y}_c),$$

and the actual training loss, with smoothed targets and temperature scaling, is

$$\mathcal{L}_{\text{train}} = \mathcal{L}_{\text{CE}}\left(\tilde{y}, \text{softmax}\left(\frac{z}{T}\right)\right).$$

Cross-entropy was compared with three alternative loss functions (Table 5.9). Although the focal loss yields a better AP, the difference with cross-entropy is within the noise level of our setup. Cross-entropy is therefore retained for its simplicity and stability across configurations (Chapter 5).

4.5 Training setup

All models are trained for 3 epochs with the AdamW optimizer. The learning rate follows a cosine decay schedule with a linear warmup over the first 5-20% of training steps. The number of epochs is kept deliberately low to limit overfitting on a comparatively small training set (1500 videos yielding ≈ 4500 clips at three TTE values) when fine-tuning a large-capacity backbone (305 million parameters for VideoMAE-Large).

Training is performed on a single NVIDIA RTX 6000 Ada GPU (48 GB) with mixed-precision (FP16) and a batch size of 4 clips per device, combined with gradient accumulation over 6 steps, yielding an effective batch size of 24 clips. Each model checkpoint is saved at the end of each epoch and only the last one is retained. No validation is performed during the final training run, as the hyperparameters have been selected beforehand by Optuna on a held-out validation split (Section 4.7). Each individual training run takes approximately 3 hours on the target hardware.

4.6 Evaluation module

4.6.1 Test-time protocol

At inference time, the same preprocessing and frame-reading procedure is applied as during training, with two differences: no augmentation is performed and temperature scaling is disabled. The model returns a two-dimensional logit vector per clip which is converted into a probability over the two classes through a standard softmax.

The test set provided by the competition contains videos that are already truncated at the three TTE values. Each test video is therefore directly compatible with the preprocessing pipeline. A single 16-frame clip is extracted from the last three seconds of each video using the same regular sampling strategy as in training without any augmentation. The model then produces a probability of accident for each clip which is submitted as the final prediction.

4.6.2 Model ensemble

The final submission is based not on a single model but on an ensemble of three independently trained models. The three models share the same VideoMAE-Large backbone and the same preprocessing pipeline but differ in:

- their random seeds (42, 343, 643).
- their data augmentation strategies: temporal jittering only for two models, temporal jittering combined with color jittering and horizontal flip for the third.
- part of their hyperparameters: in particular learning rate (9.0×10^{-6} and 1.86×10^{-5}), label smoothing (0.30 and 0.15) and temperature (2.5 and 1.5).

These differences are deliberate: the goal is to obtain predictors that are individually robust but sufficiently decorrelated, so that their errors do not systematically

overlap. The final prediction is obtained by combining the three models in logit space rather than directly averaging their probabilities. Each model outputs a probability $p_i \in (0, 1)$ which is first mapped back to its logit $z_i = \log \frac{p_i}{1-p_i}$, i.e. the model’s raw, unbounded score. The three logits are then combined as a weighted sum and the result is passed through the sigmoid function to recover an ensemble probability:

$$p_{\text{ens}} = \sigma \left(\sum_{i=1}^3 w_i z_i \right), \quad \sigma(x) = \frac{1}{1 + e^{-x}}, \quad \sum_{i=1}^3 w_i = 1.$$

Averaging in logit space, rather than averaging probabilities, preserves the confidence of the individual models: a probability average pulls confident predictions toward the centre, whereas the logit space is unbounded and symmetric around zero, so strong predictions retain their weight. The exact configuration of each model and the contribution of the ensemble to the final mAP are reported in Chapter 5. The ensemble relies on three instances of the same model, each trained with a distinct random seed (42, 343 and 643) to encourage prediction diversity. The three configurations share the same training length (3 epochs) and the same dropout rate (0.1) but differ in their augmentation regime and hyperparameters. Models 42 and 643 apply temporal augmentation only and use identical hyperparameters (learning rate 9.03×10^{-6} , label smoothing 0.30, weight decay 2.90×10^{-4} , 10% warmup), so their diversity stems solely from random initialisation and sampling order. Model 343, by contrast, combines the full set of augmentations (ColorJitter, horizontal flip and temporal augmentation) and adopts more pronounced settings (learning rate 1.86×10^{-5} , label smoothing 0.15, weight decay 3.39×10^{-4} , 20% warmup), making it the most complementary model of the ensemble. This controlled diversity, with two configurations that differ only by their seed and a deliberately distinct third model, is precisely what makes the aggregation worthwhile, since an ensemble only improves when its models make partially uncorrelated errors.

Test-time augmentation (TTA) applies a variation of the temporal window used to sample each clip at inference time. Rather than relying on a single fixed window of 90 frames the clip is also evaluated with a shorter window of 75 frames and a longer one of 105 frames and the resulting predictions are averaged. Averaging over these slightly different temporal views produces a more stable score than a single prediction would at the cost of an inference time multiplied by the number of windows.

4.7 Hyperparameter optimization

The hyperparameters of the model are not tuned by hand. Instead, the search is delegated to Optuna [32], a hyperparameter-optimization framework based on

Bayesian optimization. The optimized hyperparameters are the learning rate, weight decay, warmup ratio, dropout rate, label-smoothing factor and temperature. The Optuna study splits the training data 80/20 into training and validation (1 200/300 videos, 3 600/900 clips, stratified on the label). The validation Average Precision (AP) is the objective maximized across trials.

The search proceeds in two stages. A first study explores a broad search space over 20 trials with 3 training epochs per trial. A second study refines the most promising regions over 15 trials with 5 epochs per trial, using a Tree-structured Parzen Estimator sampler combined with a Median pruner that early-stops trials whose intermediate AP falls below the median of completed trials. The two best configurations from the first study are enqueued as warm-start points in the second study which both accelerates convergence and guarantees that the second study cannot perform worse than the first. This two-stage protocol improves the AP of the model by approximately +0.05 to +0.08 compared to manual tuning.

Chapter 5

Experiments and results

This chapter presents the experimental work that led to the proposed method. The exploration follows a top-down logic: it first compares preprocessing strategies, then selects an architecture among the candidates considered, then refines the chosen architecture through dedicated ablations on the backbone size, the loss function and the data augmentation pipeline. The chapter closes with the final performance of the proposed solution.

5.1 Experimental setup

5.1.1 Hardware and software

All experiments were conducted on the Consortium des Équipements de Calcul Intensif (CÉCI), a federated Tier-2 HPC infrastructure shared by UCLouvain, ULB, ULiège, UMONS and UNAMUR, supported by the F.R.S-FNRS and the Walloon Region. The Lyra cluster (ULB) was the primary resource used for training and fine-tuning, since each node is equipped with an AMD EPYC-Genoa processor and an NVIDIA RTX 6000 Ada GPU with 48 GB of video memory, which is well suited to fine-tuning a large video Transformer. Queueing times on Lyra were often long; Dragon2 (UMONS) was therefore used as a fallback for long-running GPU jobs and Lemaitre4 (UCLouvain) as a CPU-only fallback for preprocessing tasks.

The training procedure was implemented in Python with PyTorch2.1.2, CUDA 12.1.1 and OpenCV 4.9.0.80. Pre-trained Transformers were trained with the Hugging Face Trainer API and the hyperparameters of the final model were tuned with Optuna.

5.1.2 Validation protocols

Two validation protocols are used throughout the chapter, depending on the experiment:

- a 5-fold stratified cross-validation on a balanced subset of 402 videos drawn from the training set, used to estimate the variance of each preprocessing configuration. The subset size was chosen as the largest one that allowed all configurations to be evaluated within the available GPU budget.
- 80/20 division between training and validation on all the training data for the final configuration selected.

When the official Kaggle test set is also evaluated, public and private scores are listed separately, since only the public score was available during the competition itself.

5.2 Preprocessing strategies

This section addresses what turned out to be the most critical aspect of the work. Model selection is relatively straightforward thanks to available benchmarks but performance can deteriorate significantly under inadequate preprocessing. The fundamental question is therefore: how should the relevant frames be selected for training?

A naive approach would be to use all available frames. This is not feasible in practice: a video lasts on average 40 seconds at 30 FPS, that is about 1 200 frames per video and the 1 500 videos of the training set amount to roughly 1.8 million frames per epoch. This figure does not even account for the multiplicative cost of cross-validation folds, multiple epochs and data augmentation. Furthermore, the Transformer-based architectures used in this work impose fixed clip lengths: 16 frames for VideoMAE and 32 frames for ViViT, whereas the CNN-LSTM and 3D-CNN baselines accept variable-length clips. Frames must therefore be selected carefully, since not all of them carry the same predictive weight.

Four preprocessing strategies are evaluated. Each is adjusted to the input size required by a specific architecture: strategies 1 and 2 produce clips of 32 frames for ViViT, strategy 3 produces clips of 16 frames for VideoMAE and strategy 4 produces clips of variable length for the CNN-LSTM and 3D-CNN baselines. Since each strategy is tied to a specific architecture, the experimental protocol does not constitute a full factorial comparison of architectures and strategies. Instead, each strategy is evaluated on the architecture for which it was designed

5.2.1 Strategy 1: sampling within the alert interval

For each accident video, 32 frames are selected within the alert interval, which contains on average 45 frames (about 1.5 seconds, Figure 2.8). A fraction of videos contain only a handful of frames in this interval; in such cases, frames are added before or after to reach the required count. For non-accident videos, the 32 frames are uniformly sampled around the middle of the video.

Two variants of this strategy are tested.

- **V1:** The subjective alert time is substituted by the deterministic indicator $t_{\text{event}} - 2.5$ s. It is used as the beginning of the sampling window. The 32 frames are then sampled over the window $[t_{\text{event}} - 2.5 \text{ s}, t_{\text{event}}]$. This ensures a coherent time range for all videos and eliminates any subjectivity in alert annotation.
- **V2:** Images from videos that do not show accidents are selected at random throughout the video rather than evenly distributed around the middle.

V1 and V2 are independent and can be combined; the configuration V1 + V2 applies both modifications simultaneously.

5.2.2 Strategy 2: non-overlapping sliding windows

A sliding window of 32 frames without overlap is applied to the entire video. A clip is labeled 1 if at least one of its frames falls within the alert interval and 0 otherwise. To balance the dataset, all positive clips are kept and an equal number of negative clips is sampled at random.

V1: As in Strategy 1, the subjective alert time is replaced by $t_{\text{event}} - 2.5$ s. The value 2.5 s is chosen as the maximum TTE of 1.5 s plus a 1-second clip duration, so that any single frame within the new interval is sufficient to label the clip as positive.

5.2.3 Strategy 3: test-aligned clip sampling

Each accident video is truncated into three sub-videos ending at $t_{\text{event}} - \Delta_{\text{TTE}}$ for $\Delta_{\text{TTE}} \in \{0.5, 1.0, 1.5\}$ s, to replicate the conditions of the official test set. From each truncated video, a 16-frame clip is uniformly extracted from a window of 90 frames preceding the cutoff. This window is defined as a fixed number of frames at a reference rate of 30 FPS (3 s) and the same definition is used at training and at evaluation time to keep both conditions consistent. The choice of 3 s rather than 2.5 s provides a safety margin: since the real frame rate varies between 23 and 31 FPS, a fixed count of 90 frames still covers at least ≈ 2.9 s of context

even on the fastest videos, whereas a strict 2.5-second window would have been unreliable. Non-accident videos follow the same procedure using a fake event time as reference. Each clip inherits the label of its source video. Because Strategy 3 produces 16-frame clips, it is specifically designed for the VideoMAE architecture.

5.2.4 Strategy 4: uniform frame subsampling

Whereas the earlier strategies were tailored to Transformer architectures operating on clips of fixed length, this fourth strategy instead addresses the CNN-LSTM and 3D-CNN models, which enforce no hard constraint on the number of frames a clip may contain. Here, the selection spans the full duration of the video: rather than being confined to the alert interval, one frame in every x is retained throughout the sequence. A range of values was explored for this interval, namely $x \in \{10, 16, 20, 32, 40, 50, 70, 90\}$. The label of each generated clip is inherited directly from the video it originates from.

5.3 Video level

5.3.1 ViViT results

Experimental protocol

The ViViT architecture is initialized with the pre-trained weights from google/vivit-b-16x2-kinetics400. Training is performed using 5-fold stratified cross-validation to preserve class balance across the folds. Each clip consists of 32 frames and the model is trained for 10 epochs with a batch size of 3 and a learning rate of 10^{-5} . The training dataset is a balanced subset of 402 videos drawn from the training set, consisting of 201 accident videos and 201 non-accident videos. This subset is used so that each cross-validation fold can be trained within the available GPU budget. For each configuration, predictions are generated on the official Kaggle test set; the reported mAP values therefore correspond exactly to the metric used for ranking. The Kaggle test scores reported throughout this section correspond to a single submission per configuration, obtained from the fold achieving the highest validation AP. This configuration serves as a controlled baseline that allows the effect of different clip-selection strategies to be isolated before introducing data augmentation in subsequent experiments.

Evaluation strategies

To evaluate the trained models on the Kaggle test set, each test video is analyzed using three distinct sampling strategies. All three strategies operate on the last

frames of the test clip, since the official test videos are cropped to end at $t_{\text{event}} - \Delta_{\text{TTE}}$ with $\Delta_{\text{TTE}} \in \{0.5, 1, 1.5\}$ s, meaning that the most informative frames for predicting an imminent collision are located at the end of each video.

- **Stride 1.** evaluation is carried out over the final 32 consecutive frames of the test video. At 30 FPS, this spans roughly 1 second leading directly up to the cropping point and thus captures the visual context that sits closest to the potential event.
- **Stride 2.** the model receives 32 frames drawn from the tail of the video but sampled every other frame. The temporal span therefore widens to around 64 frames (close to 2 seconds) without altering the input size. The trade-off is a broader view obtained at a coarser temporal resolution.
- **Sliding window.** over the last 64 frames, a 32-frame window advances with a step of 8, which yields five clips that overlap one another. Each clip is processed separately and the video-level prediction comes from averaging their scores. By no longer depending on one fixed temporal alignment, this approach yields a more robust estimate.

Results without data augmentation

On both the public and the private test sets, Strategy 1 and its variants stay ahead of Strategy 2 (Figure 5.1). This difference most likely comes from the kind of segments each method creates. The segments used by Strategy 2 are often only partly useful. A clip is already given the label 1 as soon as a single one of its frames falls inside the alert interval, so many clips become positive even though the alert covers just 1 or 2 frames out of 32. Strategy 1, on the other hand, forces the model to find the key moment inside every positive clip.

Furthermore, the cross-validation analysis of Strategy 2 cannot be fairly compared with that of Strategy 1, because its clip-generation pipeline suffers from a video-level data leakage. Strategy 2 pre-computes all the clips of every video into a shared cache, without filtering by the train/validation split of the current fold. As a consequence, clips originating from videos belonging to the validation fold are inadvertently included in the training set, producing artificially high validation metrics ($\text{AP} \approx 0.99 \pm 0.02$, $\text{recall} = 1.000$ across all folds). The Kaggle test scores remain valid measurements, they bypass the internal split entirely and are reported in Figure 5.1. They correctly position Strategy 2 below Strategy 1 in terms of generalization performance. Fixing this implementation issue, by restricting the clip cache to the videos of the current training fold via the video id field, is left as a straightforward future improvement. Strategy 2 is not retained for further cross-validation analysis in the present chapter.

The best private score (0.782, without data augmentation) is achieved by Strategy 1-V2 evaluated with a stride of 1, while the best public score (0.739) corresponds to Strategy 1-V1+V2 with a stride of 1. This discrepancy is consistent with the general public/private behavior discussed later, where private scores are systematically higher than public ones across most configurations.

Random sampling of negatives Under Strategy 1, drawing frames at random from non-accident videos proves more effective than centring the selection on their middle. This can likely be traced to a positional bias within accident footage. Since collisions tend to take place around the midpoint of the video, sampling the negative examples from that same region as well would let the model lean on temporal position rather than on genuinely discriminative visual cues. Sampling at random dissolves this spurious correlation.

Deterministic alert marker Using a fixed marker $t_{\text{event}} - 2.5$ s instead of the human-annotated alert time (variant V1) does not yield a significant improvement on the Kaggle test set, although it reduces inter-fold variability in cross-validation.

Evaluation method The strongest results come from evaluation with stride 1. The final 32 consecutive frames are precisely those just before the cutoff, and so they hold the greatest amount of information useful for prediction. Widening to a stride 2 window brings in a longer temporal context (around 2 seconds), yet the coarser resolution translates into a modest loss of performance. Lowest of all are the scores from the sliding-window scheme. Averaging predictions over every window waters down the decisive signal carried by the closing frames with earlier context that carries less information.

It should be noted that the training and evaluation conditions differ in this competition. In principle, a model is evaluated under conditions as close as possible to those under which it was trained, so that the distribution of inputs encountered at test time matches that of training. Here, however, the test set consists of videos of approximately 10 seconds trimmed at $t_{\text{event}} - \Delta_{\text{TTE}}$, while the training videos are kept at their full average duration of ≈ 40 seconds without prior segmentation. This train/test mismatch is precisely what Strategy 3 is designed to address.

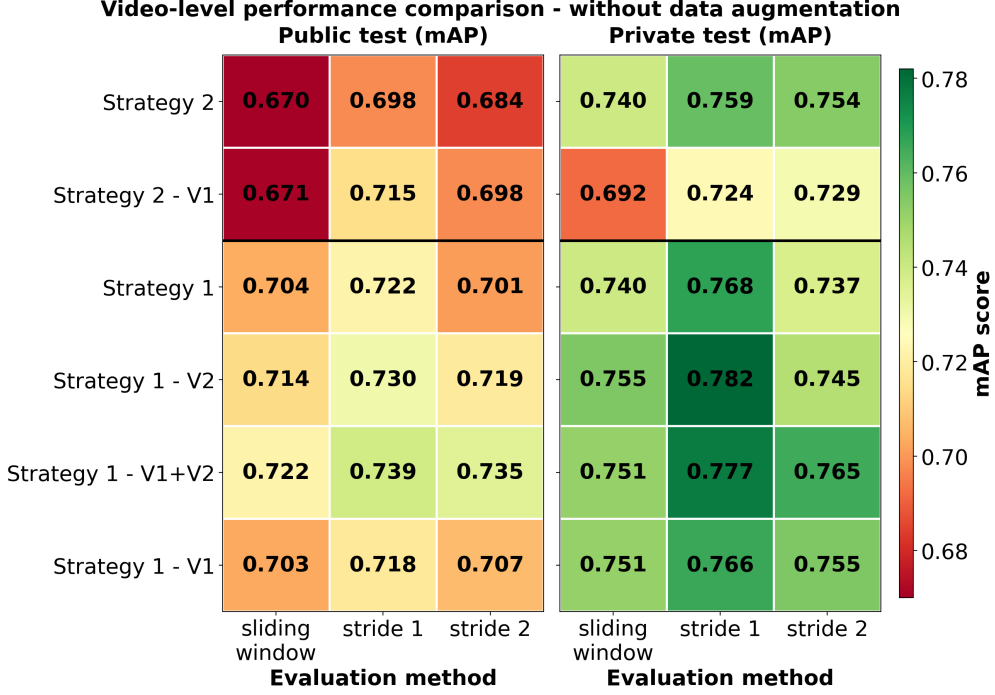


Figure 5.1: Video-level performance of ViViT without data augmentation, on the public (left) and private (right) Kaggle test sets.

Results with data augmentation

Data augmentation is intended to simulate the diversity of capture conditions present in the dataset and to improve robustness. Its intensity reflects an attempt to test whether strong regularization could compensate for the small size of the training subset. The augmentation pipeline used here is the following:

- **ColorJitter:** simulates different lighting and weather conditions.
- **GaussianBlur:** simulates rain and fog.
- **RandomRotation:** simulates camera shake.
- **RandomResizedCrop:** simulates different viewing angles.

Data augmentation yields varied results (Figure 5.2). For Strategy 1-V2, augmentation slightly enhances the private mAP from 0.782 (without DA) to 0.786 (with DA) at stride 1 but lowers the public score from 0.730 to 0.709. The reverse pattern is seen for Strategy 1-V1+V2: the public score improves from 0.739 to 0.746, while the private score decreases slightly from 0.777 to 0.769. Two factors

likely explain the modest impact of augmentation. First, the ViViT backbone pre-trained on Kinetics-400 is already strongly regularized leaving little room for additional gain. Second, augmentations such as RandomRotation may destroy spatial cues that are critical in the dashcam setting, notably the position of the horizon and the alignment of vehicles.

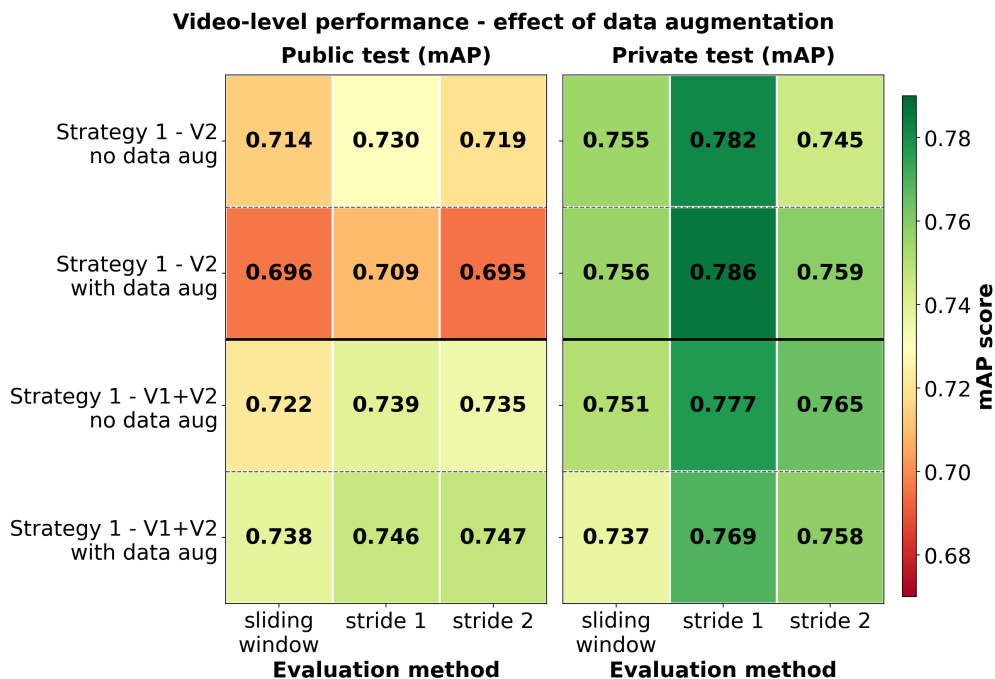


Figure 5.2: Video-level performance of ViViT with data augmentation.

The configurations evaluated on the public dataset show that Strategy 1-V1+V2 with data augmentation obtains the highest public mAP (0.746). The alert marker $t_{\text{event}} - 2.5$ s defined by V1 is retained and combined with random sampling of negative clips (V2). This setup is used as the basis for the future tests on comparing architectures and backbone. The decision to keep Strategy 1-V1+V2 + DA was made during the competition under the constraint that only the public score was visible.

Cross-validation variance

To characterize the stability of each approach independently of the single-submission selection, the variance of the validation metrics across the 5 folds is reported in Table 5.1 for Strategy 1 and its variants, with and without data augmentation.

Metric	Strategy 1	Strategy 1 - V1	Strategy 1 - V2	Strategy 1 - V1+V2
<i>Without data augmentation</i>				
Accuracy	0.652 ± 0.054	0.649 ± 0.058	0.654 ± 0.073	0.669 ± 0.041
Precision	0.617 ± 0.034	0.607 ± 0.046	0.614 ± 0.057	0.626 ± 0.038
Recall	0.786 ± 0.146	0.871 ± 0.027	0.836 ± 0.072	0.846 ± 0.034
F1-score	0.688 ± 0.078	0.714 ± 0.034	0.708 ± 0.062	0.720 ± 0.030
AUC-ROC	0.709 ± 0.066	0.761 ± 0.046	0.730 ± 0.061	0.771 ± 0.060
AP	0.665 ± 0.057	0.735 ± 0.033	0.678 ± 0.060	0.754 ± 0.067
<i>With data augmentation</i>				
Accuracy	-	-	0.634 ± 0.041	0.729 ± 0.080
Precision	-	-	0.588 ± 0.028	0.676 ± 0.072
Recall	-	-	0.911 ± 0.097	0.895 ± 0.073
F1-score	-	-	0.713 ± 0.034	0.769 ± 0.063
AUC-ROC	-	-	0.767 ± 0.075	0.807 ± 0.077
AP	-	-	0.745 ± 0.078	0.784 ± 0.084

Table 5.1: Validation metrics across the 5 cross-validation folds for Strategy 1 and its variants, with and without data augmentation.

Strategy 1-V1 is the most stable variant ($\sigma_{AP} = 0.033$) among the configurations with augmentation. V1+V2 achieves the highest mean AP (0.784) just as it does without augmentation.

Effect of data augmentation on variance A counter-intuitive observation is that data augmentation, while designed to act as a regularizer, increases the inter-fold variance of the AP rather than reducing it. For V2, the AP rises from 0.678 ± 0.060 to 0.745 ± 0.078 ; for V1+V2, from 0.754 ± 0.067 to 0.784 ± 0.084 . The standard deviation increases by approximately 0.017 and 0.018 respectively. This is most likely due to the strong geometric augmentations (RandomRotation, RandomResizedCrop), which produce inputs whose difficulty varies considerably from one sample to another: some folds may by chance contain more difficult augmented clips than others.

Inversion between validation and test rankings Comparing the validation results to the official Kaggle test set (Table 5.2) reveals an important phenomenon.

The ranking of configurations differs between validation and the Kaggle test set. The Strategy 1-V1+V2 + DA configuration achieved the best validation AP (0.784) but ranks fifth out of six on the private set (0.769). Conversely, V2 + DA is third in validation (0.745) but achieves the highest private mAP (0.786). No single configuration can therefore be claimed to be definitively superior based on a single submission and the choice of V1+V2 + DA as the basis for subsequent experiments must be understood as a decision taken under the constraint that only the public score was available.

Configuration	Val AP (mean $\pm$ std)	Public mAP	Private mAP
Strategy 1	0.665 $\pm$ 0.057	0.722	0.768
Strategy 1 - V1	0.735 $\pm$ 0.033	0.718	0.767
Strategy 1 - V2	0.678 $\pm$ 0.060	0.730	0.782
Strategy 1 - V1+V2	0.754 $\pm$ 0.067	0.739	0.777
Strategy 1 - V2 + DA	0.745 $\pm$ 0.078	0.709	0.786
Strategy 1 - V1+V2 + DA	0.784 $\pm$ 0.084	0.746	0.769

Table 5.2: Comparison between internal 5-fold validation AP and official Kaggle test mAP (stride 1 evaluation).

Public versus private test set discrepancy

A consistent observation across all configurations is that the private scores are systematically higher than the public ones (Table 5.2, gap of ≈ 0.03 to 0.08). The same trend is observed in the overall competition leaderboard, where most teams achieve a better private than public score, which suggests an asymmetry between the two subsets rather than a specific flaw of our models.

Since the test annotations are not publicly released in full, the sources of this asymmetry can only be conjectured. The Nexar dataset spans a wide range of capture conditions (weather, scene, lighting, obstructions) as well as camera-related artifacts (motion blur, lens flare, fog) (Figure 2.7) and there is no guarantee that the proportions of difficult cases, of true collisions versus near-misses, or of short alert-to-accident intervals are identical in the public and private subsets. Any of these unobservable factors could account for the gap.

Per-TTE analysis

The mean Average Precision is by construction the average of three AP values measured at $\Delta_{\text{TTE}} \in \{0.5, 1, 1.5\}$ s. This single number hides the fact that the difficulty of the task varies with the prediction horizon: at 0.5 s the cause of the event is generally visible in the last frames of the clip, whereas at 1.5 ms the model must rely on more distant precursor cues. The per-TTE breakdown is therefore needed to understand the actual anticipation profile of each configuration.

Table 5.3 shows the performance (AP) at each TTE for the six baseline configurations, evaluated with a step size of 1 on the private test set.

Configuration	AP@500ms	AP@1000ms	AP@1500ms	mAP
Strategy 2	0.821	0.734	0.724	0.759
Strategy 1	0.810	0.733	0.761	0.767
Strategy 1 - V2	0.823	0.759	0.764	0.782
Strategy 1 - V1+V2	0.819	0.751	0.762	0.777
Strategy 1 - V1+V2 + DA	0.791	0.773	0.745	0.769
Strategy 1 - V2 + DA	0.819	0.742	0.798	0.786

Table 5.3: Average Precision per time-to-event on the private test set (stride 1 evaluation).

The anticipation profile depends strongly on the strategy The drop $\Delta = \text{AP@500} - \text{AP@1500}$ measures how much a model degrades as the prediction horizon increases. A small Δ indicates that the model remains accurate at long horizon. A large Δ indicates a model that is essentially reactive (Table 5.4).

Configuration	$\Delta = \text{AP@500} - \text{AP@1500}$
Strategy 2	+0.097
Strategy 1	+0.049
Strategy 1 - V2	+0.059
Strategy 1 - V1+V2	+0.057
Strategy 1 - V1+V2 + DA	+0.046
Strategy 1 - V2 + DA	+0.021

Table 5.4: Anticipation gap (private test set, stride 1).

The two configurations with augmentation display the lowest drop, and Strategy 1-V2 + DA shows a drop of only 0.021 making it the most balanced option overall.

The best configuration depends on the target horizon No single configuration dominates at every TTE. Strategy 1-V2 has the highest AP@500ms (0.823), Strategy 1-V1+V2 + DA has the highest AP@1000ms (0.773) and Strategy 1-V2 + DA has the highest AP@1500ms (0.798). The overall mAP ranking therefore reflects an averaging over three quite different operating points. For an ADAS application that prioritizes early anticipation, Strategy 1-V2 + DA is the most appropriate choice; for a late-warning application, Strategy 1-V2 without augmentation is preferable.

Effect of the evaluation method on the per-TTE profile The shape of the per-TTE profile also depends on the evaluation sampling method. Table 5.5 compares stride 1, stride 2 and sliding window for the two most relevant configurations.

Configuration / Eval	AP@500ms	AP@1000ms	AP@1500ms	mAP
<i>Strategy 1 - V2 + DA</i>				
stride 1	0.819	0.742	0.798	0.786
stride 2	0.772	0.735	0.771	0.759
sliding window	0.761	0.733	0.774	0.756
<i>Strategy 1 - V1+V2 + DA</i>				
stride 1	0.791	0.773	0.745	0.769
stride 2	0.786	0.751	0.737	0.758
sliding window	0.765	0.725	0.720	0.737

Table 5.5: Effect of the evaluation sampling method on the per-TTE profile (private test set).

With Strategy 1-V2 + DA, stride 1 stays ahead of stride 2 at every TTE, though the lead shrinks from +0.05 at AP@500ms down to +0.03 at AP@1500ms. Stride 1 thus proves most valuable for short-horizon predictions, where fine-grained visual evidence weighs heaviest. Over longer horizons, the wider temporal context offered by stride 2 and the sliding window makes up in part for their coarser resolution and the gap closes.

A further point concerns the sliding-window evaluation, which smooths out the per-TTE profile. On the private set for V2 + DA, AP@500ms and AP@1500ms differ by a mere -0.012 under the sliding window, against $+0.021$ at stride 1. Because averaging over overlapping clips dilutes the strong signal of the final frames, the short-horizon AP suffers more than the long-horizon one. This makes the sliding window the more conservative of the evaluators.

In four configurations out of six (all except V1+V2 + DA and strategy 2), AP@1000ms is lower than both AP@500ms and AP@1500ms. The effect is typically 0.02 to 0.06 and inconsistent across configurations so no firm conclusion is drawn here.

Ranking on the Kaggle leaderboard At its best, this ViViT baseline reaches a private mAP of 0.786 (Strategy 1-V2 + DA), which places it around 24th to 25th out of the 237 competing teams. The VideoMAE-based solution later improves on this exploratory baseline by a wide margin, climbing to a private mAP of 0.844 and a final 5th place.

5.3.2 VideoMAE results

The transition to VideoMAE represents the next step in this exploration. VideoMAE shares the same transformer backbone as ViViT but introduces a self-supervised pre-training method based on masked autoencoding, which has proven particularly effective for video understanding tasks. A notable architectural dif-

ference is the number of frames per clip: VideoMAE uses 16 frames instead of 32 for ViViT, which partially motivated the introduction of Strategy 3, as the shorter clip length requires a wider temporal window to capture the same amount of pre-accident context.

Initial attempt

As a first experiment, VideoMAE was trained using the same configuration as ViViT, keeping all other hyperparameters unchanged. The objective was to test whether the architectural change alone would be sufficient to improve over ViViT.

The results are reported in Table 5.7. Under this configuration, VideoMAE collapses to a degenerate prediction regime. Across all 5 folds, the recall reaches 1.000 and the precision stagnates at 0.513, indicating that the model predicts the positive class on virtually all examples. The mean confusion matrix in Table 5.6 confirms the failure: 78 out of 80 examples per fold are predicted positive regardless of their true label. This collapse is a overfitting signature on a small balanced dataset. It shows that VideoMAE, whose capacity significantly exceeds that of ViViT, cannot be trained under the same regime given the limited amount of available data.

Model	Actual	Predicted	
		Negative	Positive
ViViT	Negative	19.80	20.40
	Positive	6.20	34.00
VideoMAE	Negative	2.20	38.00
	Positive	0.20	40.00

Table 5.6: Mean confusion matrices across the 5 folds on the validation set.

Modified training configuration

To prevent this overfitting three changes are introduced:

- Preprocessing changes from strategy 1 to strategy 3, which generates three training segments per accident video (one per TTE value) rather than just one. This triples the actual size of the training set while mitigating the class imbalance caused by the increase in the number of positive clips.
- Cross-validation is replaced by a simple 80/20 split between the training set and the validation set, which reduces the computational cost per experiment and allows for a more in-depth exploration of hyperparameters.

- Reducing the number of epochs from 10 to 3 and increasing the batch size from 3 to 8.

The VideoMAE backbone is initialised from the publicly available weights MCG-NJU/videomae-large-finetuned-kinetics.

The direct comparison between ViViT and VideoMAE is therefore limited by the change of training regime, which is acknowledged explicitly. The remainder of this section focuses on ablations within the modified VideoMAE configuration: backbone size and pre-training dataset, loss function and augmentation strategies.

Model	Fold	Accuracy	Precision	Recall	F1	AUC-ROC	AP
ViViT	1	0.617	0.576	0.850	0.687	0.693	0.677
	2	0.654	0.628	0.781	0.696	0.742	0.736
	3	0.713	0.660	0.875	0.753	0.812	0.747
	4	0.713	0.660	0.875	0.753	0.858	0.859
	5	0.650	0.607	0.850	0.708	0.750	0.750
	Mean	0.669	0.626	0.846	0.719	0.771	0.754
	Std	± 0.042	± 0.036	± 0.039	± 0.032	± 0.064	± 0.066
VideoMAE	1	0.506	0.500	1.000	0.667	0.702	0.707
	2	0.519	0.513	1.000	0.678	0.659	0.655
	3	0.525	0.513	1.000	0.678	0.752	0.756
	4	0.500	0.500	0.975	0.661	0.551	0.610
	5	0.575	0.541	1.000	0.702	0.722	0.714
	Mean	0.525	0.513	0.995	0.677	0.677	0.688
	Std	± 0.030	± 0.017	± 0.011	± 0.016	± 0.078	± 0.057

Table 5.7: Per-fold cross-validation results for ViViT and VideoMAE on the validation set.

Backbone size and pre-training dataset

Table 5.8 compares four pre-trained models of different sizes and pre-training datasets.

The Large and Huge variants of VideoMAE, pre-trained on Kinetics-400, turn in broadly comparable performance, though the Huge variant holds a clear edge in AP (+0.033). The Base variant falls behind with an AP of 0.783, while the version pre-trained on Something-Something-V2 (SSv2) records the weakest results (AP of 0.736).

Scaling up the model yields a clear improvement between the Base and Large variants but this gain diminishes between Large and Huge. Large actually outperforms Huge on 4 out of 6 metrics (AUC-ROC, Accuracy, Recall and F1). This

saturation can be explained by the limited size of the dataset, which does not allow for full utilisation of the increased capacity of the Huge variant.

The choice of pre-training dataset also appears critical. Kinetics-400, which consists of web videos covering a wide range of human actions and natural scenes [7], offers significantly more effective transfer than SSv2 [33], which focuses on object manipulation. This result is consistent with the nature of the task: dashcam collision prediction benefits from a pre-training distribution that includes dynamic, scene-level visual content rather than fine-grained hand-object interactions.

The Large variant offers the best overall compromise. It achieves performance levels nearly equivalent to Huge on AP, outperforms it on most other metrics and requires fewer computational resources. It is therefore retained for the subsequent ablations.

Finally, it should be noted that the Huge and Large variants exhibit significantly different classification behaviors. The Huge variant achieves high precision (0.771) but lower recall (0.680), reflecting a conservative model that flags hazards less frequently but with greater reliability when it does. Large offers a better balance between the two (precision 0.766, recall 0.749). The choice between the two variants may therefore depend on the deployment objective: minimizing false positives (Huge) or maximizing overall AP (Large).

Pre-training dataset impact				
Metric	Finetuned on Kinetics-400			Finetuned on SSv2
	Base	Large	Huge	Base
Parameters	87M	305M	633M	87M
AP	0.7833	0.7847	0.8176	0.7363
AUC-ROC	0.7677	0.8132	0.8127	0.7491
Accuracy	0.6911	0.7600	0.7389	0.6922
Precision	0.6838	0.7659	0.7708	0.6901
Recall	0.7111	0.7489	0.6800	0.6978
F1	0.6972	0.7573	0.7226	0.6939

Table 5.8: Impact of backbone size and pre-training dataset on validation performance.

Loss function

Three loss functions are compared using the Large backbone pre-trained on Kinetics-400 (Table 5.9). Cross-entropy serves as the standard baseline: it penalises each example in proportion to the negative log-probability assigned to the true class.

Focal loss extends cross-entropy by multiplying it by a factor $\alpha \times (1 - p_t)^\gamma$, which downweights correctly classified examples and focuses learning on hard cases; the term α also helps rebalance the classes. Finally, exponential loss is inspired by the DSA loss introduced by Chan et al. [16] for accident prediction: it exponentially penalises poor predictions based on a margin and applies an increased weight to positive examples to favour their detection.

On five of the six metrics, focal loss comes out ahead. Cross-entropy trails it by a narrow margin, whereas exponential loss falls behind even though it records the highest recall. The exponential penalty, combined with the weight placed on positive examples, drives the model toward minimising false negatives, which gives rise to the characteristic high-recall, low-precision pattern that Chan et al. [16] already documented (80% recall, 56% precision with their DSA loss). The imbalance therefore appears to be not an artefact of the implementation but something intrinsic to exponential loss functions when applied to accident prediction. Cross-entropy stays a sturdy and straightforward baseline, yet focal loss is the better choice whenever maximising AP is the chief aim.

Cross-entropy remains a robust and simple baseline but focal loss is preferable when the primary objective is to maximise AP.

Loss function	Loss function impact					
	AP	AUC-ROC	Accuracy	Precision	Recall	F1
Focal loss	0.8348	0.8319	0.7622	0.7658	0.7556	0.7606
Exponential loss	0.7897	0.8025	0.7056	0.6703	0.8089	0.7331
Cross-entropy	0.8293	0.8245	0.7600	0.7588	0.7622	0.7605

Table 5.9: Impact of loss function on validation performance. TA and temperature scaling are disabled; all other hyperparameters are fixed.

Augmentation strategies

The ViViT experiments revealed an ambiguous effect of spatial data augmentation: improvements on some configurations were offset by degradations on others, suggesting that the augmentation pipeline was too aggressive for this task. Based on this observation, the VideoMAE experiments use a lighter spatial pipeline and introduce two new mechanisms specifically adapted to the video domain:

- **Spatial data augmentation (DA):** colour jitter on brightness and contrast and random horizontal flip. This pipeline is lighter than the one used for ViViT.

- **Temporal augmentation (TA)**: random shifting of the sampled frame indices by ± 3 frames, exposing the model to slightly different temporal positions within the 3-second window.
- **Temperature scaling (Temp.)**: dividing the logits by a factor $T > 1$ during training to smooth the output distribution and reduce overconfidence.

The results are reported in Table 5.10. The optimal configuration combines TA and temperature scaling without spatial augmentation, achieving an AP of 0.834, an AUC-ROC of 0.840, an accuracy of 0.763 and an F1 score of 0.763. The baseline without any augmentation already reaches an AP of 0.806.

Against intuition, layering in spatial data augmentation consistently hurts performance. DA + TA + Temp. falls to an AP of 0.785 and DA + TA without temperature to 0.790. The roughly 0.05-point AP gap separating TA + Temp. from DA + TA + Temp. is far from trivial. The most plausible reason is that several standard augmentations sit awkwardly with the dashcam domain. A horizontal flip reverses the flow of traffic and so clashes with driving conventions, while colour jitter, if poorly calibrated, can distort the look of traffic lights and brake lights, which are precisely the cues that matter most for anticipating sudden braking. Temperature scaling stands out as the single most effective factor, accounting for the largest portion of the AP gain over the no-augmentation baseline. Temporal augmentation adds a further lift in recall and pairing the two produces the strongest configuration overall. What matters here is that the gain does not come from trading precision against recall. Both improve together, which points to TA and temperature scaling acting as complementary mechanisms rather than ones that merely offset each other.

Data augmentation impact					
Metric	None	Temp. only	TA + Temp.	DA + TA + Temp.	DA + TA
AP	0.8057	0.8284	0.8336	0.7847	0.7896
AUC-ROC	0.8314	0.8132	0.8403	0.8132	0.8142
Accuracy	0.7244	0.7456	0.7633	0.7600	0.7500
Precision	0.7405	0.7676	0.7639	0.7659	0.7805
Recall	0.6911	0.7044	0.7622	0.7489	0.6956
F1	0.7149	0.7346	0.7631	0.7573	0.7356

Table 5.10: Impact of augmentation strategies on validation performance. DA = spatial data augmentation (colour jitter + horizontal flip); TA = temporal augmentation (± 3 frame jitter); Temp. = temperature scaling.

5.3.3 Hybrid approach and 3D-CNN

The CNN-LSTM and 3D-CNN architectures were evaluated with Strategy 4. The CNN-LSTM uses a ResNet-18 pre-trained on ImageNet as the per-frame feature extractor, followed by an LSTM that aggregates the resulting sequence of feature vectors into a clip-level representation. The 3D-CNN is also built on a ResNet-18 backbone whose 2D convolutions have been inflated to 3D. The training/validation split (201 accident videos and 201 non-accident videos) is the same as for the ViViT baseline (Section 5.3.1). Several subsampling rates $x \in \{10, 16, 20, 32, 40, 50, 70, 90\}$ were tested; the final mAP values vary by less than 0.02 across this range, so only the best configuration of each model is reported here.

On the private test set, the best configuration of each model achieves a mAP of 0.692 for CNN-LSTM and 0.681 for 3D-CNN. Public scores are consistently lower at 0.625 and 0.597 respectively (Table 5.11). The Private mAP column reports the average across the three TTEs.

Model	TTE (s)	Accuracy	AP	Private mAP
CNN-LSTM	0.5	65.14%	0.722	0.692
	1.0	63.95%	0.695	
	1.5	62.50%	0.659	
3D-CNN	0.5	66.55%	0.727	0.681
	1.0	62.66%	0.673	
	1.5	63.75%	0.645	

Table 5.11: Per-TTE performance of the CNN-LSTM and 3D-CNN models on the private test set.

Beyond the mAP values themselves, what proves most informative is the structure of the confusion matrices (Figure 5.3). Both models build up a large count of true negatives but they also produce a substantial number of false negatives. Recall on the collision class drops especially low at the longest horizon reaching 49% for CNN-LSTM and 49% for 3D-CNN at $\Delta_{\text{TTE}} = 1.5$ s. More than half of the real collisions are therefore missed once the model has to lean on distant precursor cues.

This behaviour traces back in large part to the preprocessing strategy. Keeping one frame in every x across the whole video offers no guarantee that the frames holding the actual accident evidence end up in the clip. A clip drawn from an accident video may thus contain only pre-accident or post-accident frames with no visible sign of the event, yet still inherit the positive video-level label. Training is consequently fed a fair amount of label noise. The problem grows most acute for

videos whose alert intervals are very short, where only a handful of frames carry the accident signal.

A second symptom of this misalignment is the way the predicted confidence evolves as the prediction horizon shortens. One would expect both models to grow more confident as Δ_{TTE} falls, since the accident draws visually closer within the clip. What happens is the reverse. The average predicted probability on collision videos drops from 0.59 to 0.51 for CNN-LSTM and from 0.50 to 0.46 for 3D-CNN as Δ_{TTE} contracts from 1500 ms to 500 ms. This counter-intuitive trend signals that the models have not learned to tie the visual nearness of an event to a higher collision probability. One direct consequence of training on full-video clips is that post-event frames are never observed at test time.

At a more fundamental level, Strategy 4 lends itself to accident detection rather than to accident anticipation. Throughout training, the model sees frames sampled from across the whole video, among them frames that follow the event (and occasionally the collision itself), none of which are ever available at test time, where the videos are cut off before the event. This discrepancy between the training and evaluation conditions makes the task especially difficult and accounts for the conservative behaviour of both models at long horizon, where they miss many collisions, as the high false-negative rate in the confusion matrices makes clear.

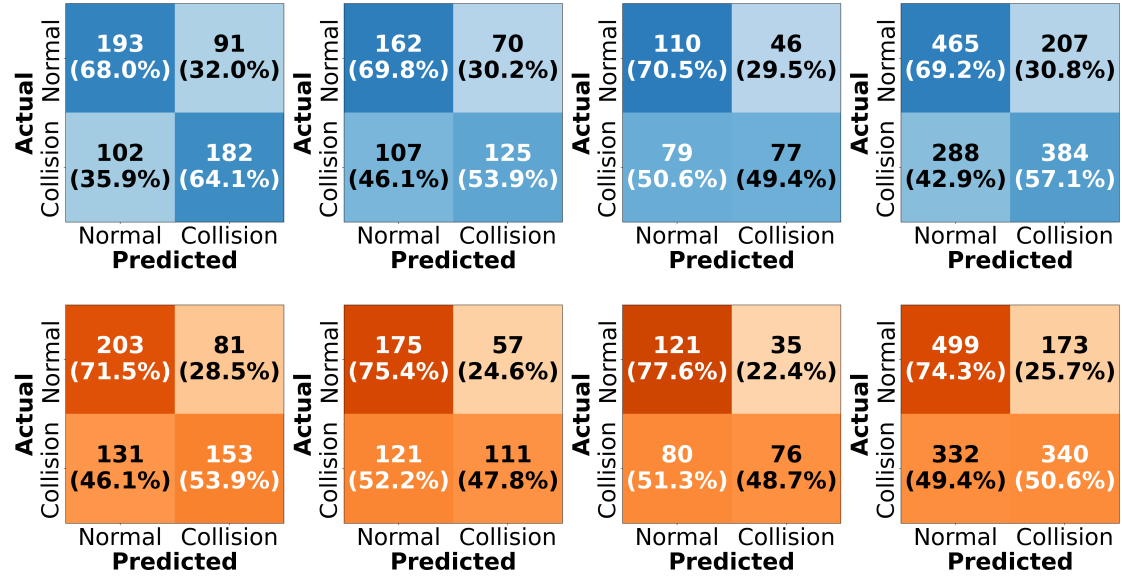


Figure 5.3: Confusion matrices for CNN-LSTM (top, blue) and 3D-CNN (bottom, orange), shown per time-to-event (0.5 ,1.0 and 1.5s) and aggregated (rightmost column).

5.4 Frame level

A frame-level formulation is appealing because it is the most natural fit for real-time processing. As frames arrive, the model emits a prediction about a potential accident on a per-frame basis, which matches the streaming nature of an ADAS setting. Unlike the video-level formulation, the frame-level approach also makes it possible to identify the exact moment at which the first signs of an accident appear, as well as the image regions the model relies on, through attention maps produced by Grad-CAM. These maps allow the model’s reasoning to be inspected: do its predictions rely on the right visual cues, or on spurious factors such as weather and sunlight? The per-frame outputs can also be traced along the video timeline, revealing where the model fails and where it remains uncertain.

The main difficulty of this formulation is the choice of the final video-level label, since the challenge requires one prediction per video. Several aggregation rules are possible: the probability of the last frame, a thresholding rule on the maximum probability, the mean, the minimum and so on, computed either over the whole video or over a sub-part of it. The most principled approach is the sliding-window scheme but it raises the same question: how is the final video-level score derived from the per-window scores? The aggregation strategy used in this work is the maximum score among the probability for each frame.

5.4.1 ViViT results

The frame-level scores are systematically lower than the corresponding video-level scores, with the gap depending strongly on the preprocessing strategy. This section compares Strategy 1 and Strategy 2 at the frame level, both without and with data augmentation, using the same evaluation protocols as at the video level: stride 1, stride 2 and sliding window.

Strategy 2 at the frame level

Strategy 2 collapses dramatically at the frame level. Without data augmentation and with stride-1 evaluation, the model reaches a mAP of only 0.596 (public: 0.564, private: 0.636), compared to 0.759 at the video level (Figure 5.4). The per-class breakdown reveals the nature of the failure: the recall on the collision class drops to 17% at $\Delta_{\text{TTE}} = 500$ ms, 11% at 1000 ms and 9% at 1500 ms (Table 5.12). The model essentially predicts no accident on the overwhelming majority of test videos. 94% of normal videos are correctly classified but more than 90% of genuine collisions are missed.

The cause of this collapse lies in the supervision signal itself. Strategy 2 assigns the video-level label (accident = 1) to any clip whose 32 frames overlap the

alert interval, even partially. Visually similar pre-accident frames may therefore receive inconsistent labels across clips, providing a noisy supervision signal that is particularly harmful for a frame-based model. Combined with the strong class imbalance inherent to frame-level evaluation (only a small fraction of frames in each accident video actually carry the event), this noise pushes the model toward a degenerate solution in which it predicts no accident with high confidence almost everywhere.

The distribution of predictions across the videos makes this clear. For both Strategy 2 and its variant V1, the majority of predictions are concentrated around 0, with very few intermediate scores. While part of this concentration reflects the natural imbalance between classes in the test set, the relative rarity of intermediate scores suggests a decision boundary that is overly confident rather than a model that is genuinely uncertain when faced with difficult cases. The average predicted probability on collision videos is only 0.19 at $\Delta_{\text{TTE}} = 500$ ms and decreases even further at longer time horizons.

TTE	Accuracy	AP	Recall (collision)	Precision (collision)
500 ms	55.11%	0.643	16.90%	71.64%
1000 ms	52.16%	0.593	11.21%	61.90%
1500 ms	51.92%	0.553	8.97%	63.64%
mAP			0.596 (public 0.564, private 0.636)	

Table 5.12: Frame-level performance of Strategy 2 (stride 1, no data augmentation) in the entire test set (private set and public set).

The contrast with the video-level evaluation is informative. At the video level, Strategy 2 reaches a public/private mAP of 0.670/0.759; at the frame level, the same model drops to 0.564/0.636. The reason lies in the asymmetric effect of max-aggregation under low recall: at the video level, a single correctly detected frame is enough to flag the entire video as an accident, so the aggregation partially compensates for the model’s tendency to miss most accident frames. Frame-level evaluation offers no such forgiveness, since every missed frame counts, which exposes the underlying weakness introduced by Strategy 2’s noisy labelling.

Strategy 1 at the frame level

At the frame level, Strategy 1 performs markedly better than Strategy 2, in line with how it was designed. Forcing every positive clip to span the critical moment yields a cleaner supervision signal, one that carries over more effectively to per-frame predictions. Without data augmentation and under stride-1 evaluation, Strategy 1 attains a mAP of 0.729 (public 0.702, private 0.762), against 0.596 for Strategy 2, an improvement of +0.133 on the private set (Figure 5.4). Under

stride-2 evaluation, the private mAP comes to 0.731 (public 0.675) and under sliding-window evaluation to 0.737 (public 0.681). Across all three protocols, Strategy 1 clearly outdoes Strategy 2, which confirms that the cleaner supervision signal feeds directly into better per-frame predictions.

The same configuration that proved optimal at the video level (Strategy 1 - V1+V2 with data augmentation) was also evaluated at the frame level. With stride-1 evaluation, it reaches a mAP of 0.729 (public: 0.706, private: 0.757), substantially above Strategy 2 and only marginally below the corresponding video-level score of 0.769 on the private set. A detailed breakdown by TTE is reported in Table 5.13.

Eval / TTE	AP@500ms	AP@1000ms	AP@1500ms	mAP
<i>Public test set</i>				
stride 1	0.723	0.707	0.689	0.706
sliding window	0.693	0.664	0.681	0.679
<i>Private test set</i>				
stride 1	0.792	0.718	0.761	0.757
sliding window	0.770	0.735	0.776	0.760

Table 5.13: Frame-level performance of Strategy 1 - V1+V2 with data augmentation broken down by TTE and evaluation method.

Two observations stand out. First, the collision recall stays high across every TTE, between 87% and 91% on the private set under stride-1 evaluation, the very opposite of what Strategy 2 showed. Second, the sliding-window evaluation lands very close to stride 1 on the private set (mAP of 0.760 against 0.757), which suggests that this model’s predictions hold steady from one frame to the next. That is a welcome property for an ADAS deployment, where the final decision usually draws on several recent frames rather than on a single instant. The gap that remains between the video-level mAP (0.769) and the frame-level mAP (0.757) of the same configuration is far narrower than its counterpart for Strategy 2 (0.759 against 0.636). Strategy 1 thus not only reaches higher absolute scores but also carries over more gracefully between the two granularities.

Effect of data augmentation

For Strategy 1 - V1+V2, data augmentation brings a small yet consistent gain on the private set under sliding-window evaluation, with the mAP climbing to 0.760, a sign of better generalisation on unseen data (Figure 5.5). Under stride-1 evaluation, augmentation nudges the private mAP up from 0.762 (without DA) to 0.763 (with DA, Strategy 1 baseline against V1+V2+DA), a marginal but stable gain. Under stride-2 evaluation, the private mAP rises from 0.731 (without DA) to 0.741 (with DA), a gain of +0.010.

For Strategy 1 - V1+V2, data augmentation produces a small but consistent gain on the private set under sliding-window evaluation (mAP rising to 0.760), suggesting better generalisation on unseen data. The stride-1 private score (0.757) is essentially equivalent to the same configuration without augmentation at the video level, indicating that the augmentation pipeline preserves the temporal structure of the clips while expanding the visual variability seen during training.

Despite this stability, the frame-level configuration of ViViT does not match the public-set performance of its video-level counterpart (0.706 vs. 0.746 at stride 1). Since submissions during the competition are ranked on the public leaderboard, the frame-level approach was therefore not retained for the final solution, even though its private mAP remains competitive. The interpretability advantages of frame-level predictions (per-frame timeline, Grad-CAM attribution) and their suitability for real-time deployment remain attractive properties for an industrial context and are discussed as future work in Chapter 6.

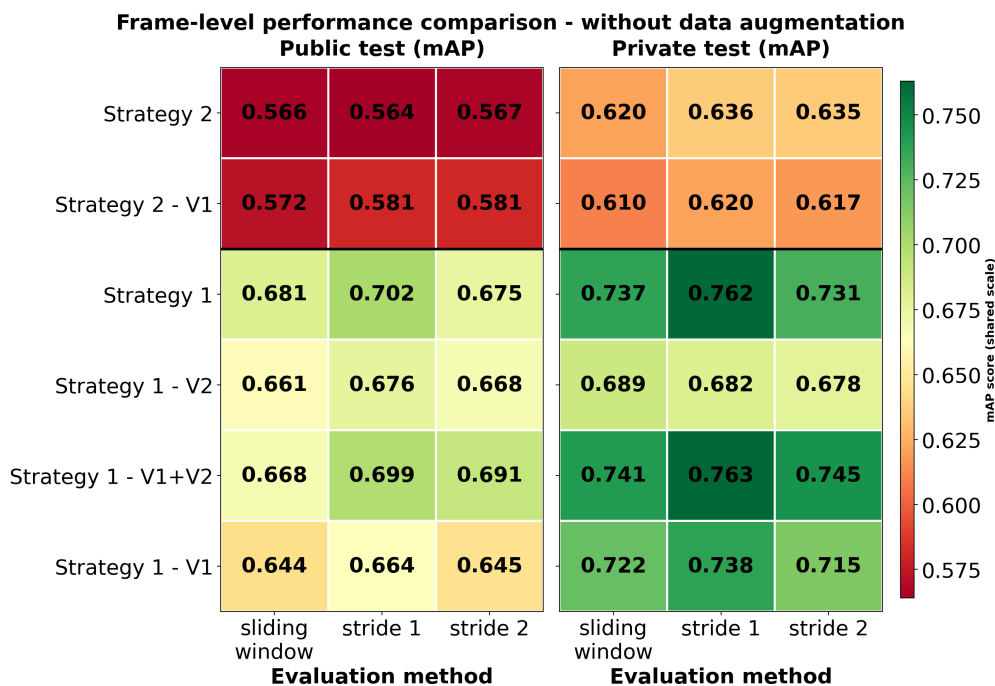


Figure 5.4: Frame-level performance without data augmentation on the public (left) and private (right) Kaggle test sets.

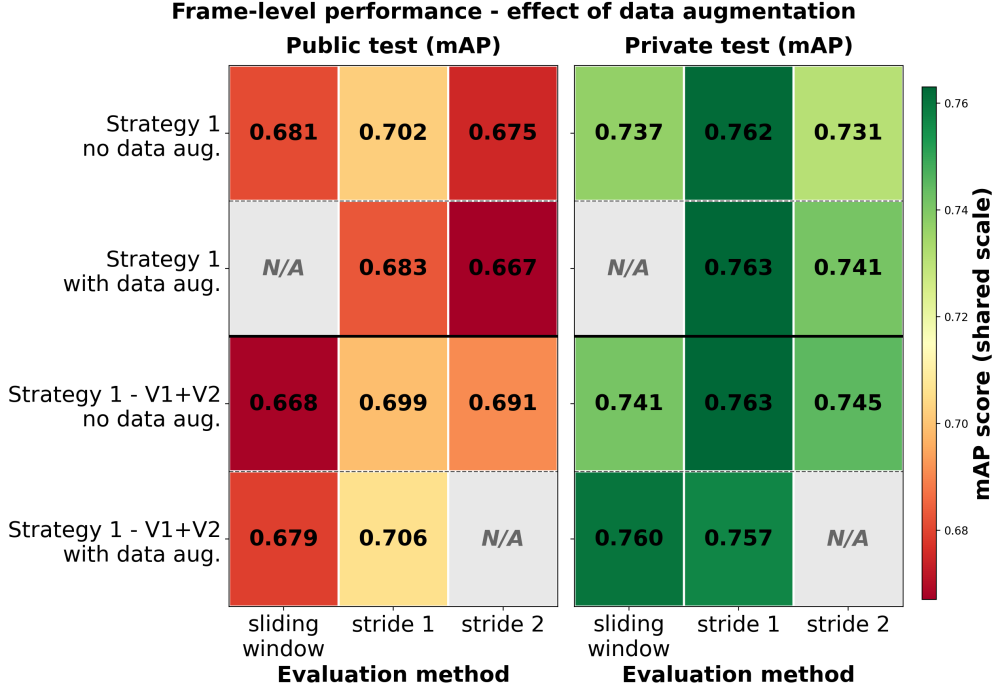


Figure 5.5: Frame-level performance with data augmentation on the public (left) and private (right) Kaggle test sets.

5.5 Ensemble learning

5.5.1 Aggregation strategy

Three instances of the model, trained with distinct seeds (42, 343 and 643), were combined in order to reduce the variance due to initialisation and sampling order. Several equal-weight probability aggregation rules were first compared: arithmetic mean, geometric mean, mean of logits, mean of ranks, maximum and minimum. Test-time augmentation (TTA) was then added. Finally, a weighted average was optimised in logit space: the probabilities are first transformed into logits $z_i = \log \frac{p_i}{1-p_i}$, combined linearly and then mapped back to a probability through the sigmoid function, $p = \sigma(w_{42} z_{42} + w_{343} z_{343} + w_{643} z_{643})$. A comprehensive study of the weights led to the selection of the weights $w_{42} = 0.65$, $w_{343} = 0.25$, $w_{643} = 0.10$ as the best combination. The primary metric is the mAP, complemented by accuracy at three horizons before the event (Table 5.14).

Configuration	mAP			Accuracy (%)		
	Global	Public	Private	@500 ms	@1000 ms	@1500 ms
Individual models						
Seed 42	0.8301	0.8203	0.8447	75.18	75.00	76.92
Seed 343	0.8239	0.8178	0.8347	77.29	77.59	77.88
Seed 643	0.8035	0.7888	0.8230	74.30	74.78	75.32
Equal-weight ensembles (3 seeds)						
Mean	0.8311	0.8230	0.8456	76.23	76.94	76.28
Geometric mean	0.8305	0.8225	0.8448	77.64	79.09	77.88
Mean of logits	0.8275	0.8177	0.8456	76.41	77.16	76.60
Mean of ranks	0.8271	0.8169	0.8450	78.70	80.17	75.96
Maximum	0.8203	0.8098	0.8369	73.06	73.71	75.96
Minimum	0.8243	0.8175	0.8364	77.46	77.80	77.24
Mean + TTA	0.8318	0.8260	0.8406	76.41	77.37	77.24
Retained model						
Weighted logit + TTA	0.8340	0.8275	0.8495	75.70	76.72	78.21

Table 5.14: Comparison of the individual models, the equal-weight ensemble strategies and the retained model.

5.5.2 Results

The retained system is a weighted average in logit space complemented by TTA. It achieves the best scores on all three mAP figures (global 0.8340, public 0.8275 and private 0.8495) (Figure 5.7) and outperforms every equal-weight strategy. The simple arithmetic mean remains the best unweighted rule whereas the extreme rules are consistently behind.

5.5.3 Analysis

Contribution of the weighting The search over 120 weight sets shows that a non-uniform weighting outperforms the equal-weight mean (+0.003 in global mAP and +0.004 in private mAP relative to the simple mean). Notably, the optimal weights follow exactly the ordering of the individual performances: the strongest seed receives the highest weight ($w_{42} = 0.65$), followed by seed 343 (0.25), while the weakest seed is reduced to its minimum weight ($w_{643} = 0.10$). The gain nevertheless remains modest: the ensemble mainly improves stability and robustness, by neutralising the effect of the weak seed (seed 643, clearly behind at 0.8035) and by smoothing the variations due to initialisation. Aggregation in logit space and weight optimisation are therefore a useful refinement but one whose

effect is limited compared with the strongest base model (seed 42, 0.8447 private score).

A strong imbalance between precision and recall The model detects almost all collisions (recall of 96.5% at 0.5 s and 97.4% at 1.0 s) but at the cost of a low precision, ranging from 0.68 to 0.72 depending on the horizon. At the shortest horizon, 128 normal videos out of 284 are wrongly classified as collisions, that is a false-positive rate close to 45%. Accuracy on the normal class therefore plateaus around 55%, against more than 96% on the collision class.

A systematic bias toward the positive class This asymmetry shows that at the 0.5 decision threshold the model over-predicts collisions. It favours not missing anything over avoiding false alarms. In a road-warning context this behaviour is partly desirable but the volume of false positives would limit the real-world usability of the system. This issue is directly linked to the type of negative event. Since recording is triggered by the camera’s built-in inertial measurement unit during sudden braking or acceleration, normal driving sequences correspond to situations that also involve strong braking dynamics without resulting in a collision. The model therefore tends to confuse abrupt braking with an imminent accident. The main avenues for improvement are threshold recalibration, class weighting and cost-sensitive learning. Figure 5.6 illustrates this phenomenon. At the default threshold, recall is very high while precision remains low, and the F1 score is only maximised at a markedly higher threshold (≈ 0.71 for the private set).

A trade-off that shifts with anticipation As the time to impact increases, the model becomes more conservative. At the most distant horizon (1.5 s), precision rises back to 0.72 and false positives decrease but recall drops from 97.4% to 92.3%, meaning that more collisions are missed. The progression analysis confirms that the mean confidence score decreases as one moves away from impact, reflecting the model’s difficulty in exploiting early visual cues.

The retained ensemble offers near-exhaustive collision detection and the best overall performance but it suffers from an excess of false positives at the standard threshold and from a precision/recall trade-off whose recall degrades as one attempts to anticipate the event earlier.

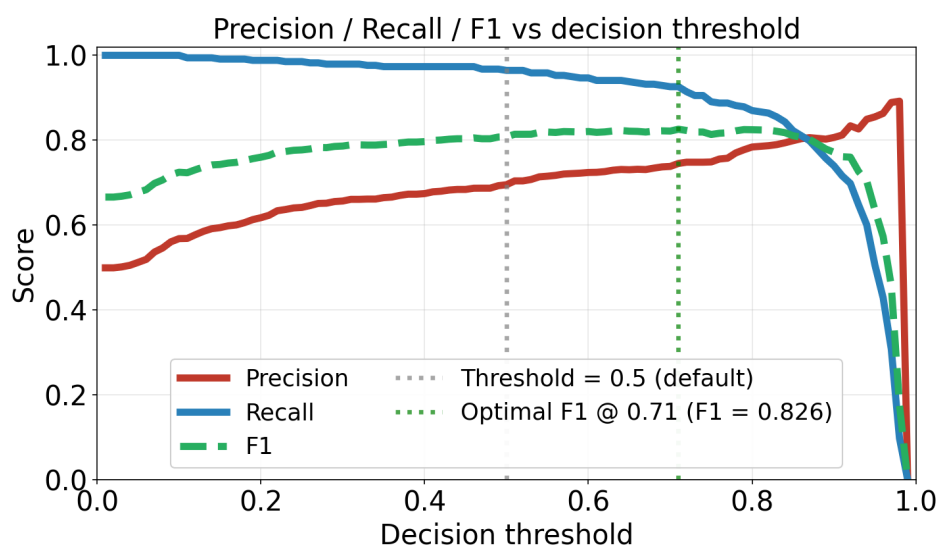


Figure 5.6: Precision, recall and F1 score of the retained model as a function of the decision threshold.

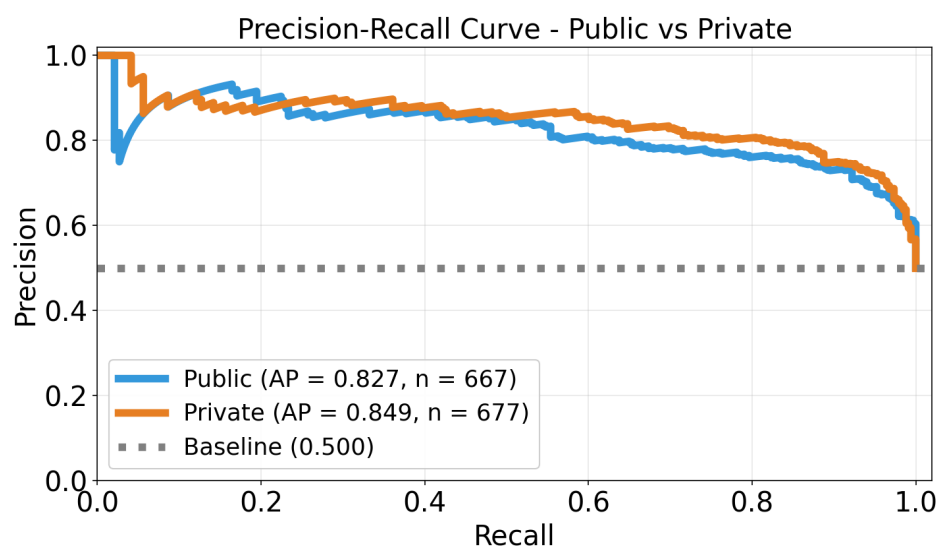


Figure 5.7: Precision-Recall curve from the best model

Chapter 6

Future work

Several avenues remain to be investigated. Numerous directions are presented below that could advance research in the field of traffic accident prediction using dashcam footage.

6.1 Frame-level approach and hybrid architectures

A frame-level approach was explored but did not match the public-set performance of the video-level pipeline. The main difficulty is the aggregation step: the Kaggle metric expects a single score per video, whereas a frame-level model produces one score per frame. Naive aggregations such as the maximum or the mean can lose information or dilute the critical signal, and a more rigorous aggregation strategy would be needed to fully exploit the per-frame predictions.

The frame-level and video-level approaches offer complementary advantages: the video-level model captures temporal patterns across the entire clip but is limited to a fixed input length, whereas a frame-level model is temporally flexible but does not account for long-range temporal context. A dual-head architecture sharing a common video encoder, producing both a video-level decision and a frame-level decision, would combine these strengths. The video-level head would provide a stable global score for the Kaggle metric, while the frame-level head would expose the per-frame attention maps and per-frame predictions useful for an ADAS deployment.

6.2 Object-aware feature extraction

Current models implicitly learn accident-related patterns from raw pixels. Incorporating explicit object-level information would allow the model to focus on what matters most: surrounding vehicles, pedestrians and cyclists. Two solutions from the Nexar challenge already explored this idea with promising results: [29] used YOLO-v8 to produce a mask isolating the cars and combined it with Farneback optical flow, while [30] stacked depth, motion and proximity signals as explicit input modalities. Building on these ideas, a three-stream architecture combining raw video, object tracking (e.g., YOLO bounding boxes and classes per frame), and optical-flow magnitude (encoding direction and velocity) appears as a natural extension of the present work, and could yield further improvements without requiring additional training data.

6.3 Vision-language models

Vision-language models (VLM) open a different direction. Beyond predicting a scalar confidence score, a VLM can produce a textual description of the scene that justifies the prediction. The main difficulty in the context of this competition is that the evaluation metric expects a single probability per video, whereas a VLM natively produces text. Bridging the two would require either a post-processing step that maps the description to a score, or a dual-output setup where the model produces both a description and a probability. Such an approach would also provide an interpretability layer that the current probability-only output lacks, which is a desirable property for safety-critical ADAS applications.

6.4 Reducing the false positive rate

Giving more weight to the non-accident clips in training would force the model to better understand these tricky negative examples, and tweaking the decision threshold would allow the balance between precision and recall to be adjusted. A sliding-window evaluation could also help, by smoothing the confidence score over time and requiring a sustained response rather than reacting to the isolated score peaks. Regularization methods could limit the model’s tendency to learn false correlations. Focal loss is a reasonable solution. It has been tested and has yielded good performances. However, cross-entropy was selected for the final model. Finally, evaluating the model on a completely different dataset and using metrics beyond mAP would provide a clearer assessment of its generalization.

Chapter 7

Conclusion

This thesis addressed dashcam collision prediction in the context of the Nexar Kaggle challenge, where a model must produce a single confidence score from a short video clip, evaluated through the mean Average Precision at three times-to-event. Several preprocessing strategies were compared, ranging from clips centred on the critical moment to uniform frame subsampling over the whole video, alongside a set of architectures spanning CNN-LSTM and 3D-CNN baselines and video Transformers such as ViViT and VideoMAE. The final system relied on a VideoMAE backbone combined with temporal data augmentation, a weighted average of the model outputs in logit space and test-time augmentation. On Kaggle’s official test set, the best configuration achieved a private mAP of 0.844, placing 5th out of 237 participating teams. The per-TTE breakdown ($\text{AP@500ms} = 0.8482$, $\text{AP@1000ms} = 0.8538$, $\text{AP@1500ms} = 0.8464$) shows comparable accuracy across all three horizons, a desirable property for early-warning applications. These results show that modern video Transformers outperform the hybrid CNN-LSTM and 3D-CNN baselines on this task. The central observation of this work is that in a compute and data bounded setting preprocessing has a stronger effect on performance than the choice of architecture. Careful design of the data pipeline appears more rewarding than aggressive model exploration, since a clean supervision signal proved decisive whereas poorly aligned clips injected label noise that no architecture managed to fully overcome. One limitation tempers these findings. The strong bias toward the positive class at the default decision threshold leaves a high volume of false positives that would hold back any real-world deployment. In the end, even with this limitation, the model attains promising performance on a difficult task and the path toward further gains stays wide open.

Bibliography

- [1] World Health Organization. “Road traffic injuries,” World Health Organization. (2023), [Online]. Available: <https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries> (visited on 05/11/2026).
- [2] M. Aleksa, A. Schaub, I. Erdelean, S. Wittmann, A. Soteropoulos, and A. Fördös, “Impact analysis of advanced driver assistance systems (adas) regarding road safety-computing reduction potentials,” *European Transport Research Review*, vol. 16, no. 1, p. 39, 2024.
- [3] L. Masello, G. Castignani, B. Sheehan, F. Murphy, and K. McDonnell, “On the road safety benefits of advanced driver assistance systems in different driving contexts,” *Transportation research interdisciplinary perspectives*, vol. 15, p. 100 670, 2022.
- [4] J. B. Cicchino, “Effectiveness of forward collision warning and autonomous emergency braking systems in reducing front-to-rear crash rates,” *Accident Analysis & Prevention*, vol. 99, pp. 142–152, 2017.
- [5] S. Hochreiter, “Long short-term memory,” *Neural Computation MIT-Press*, 1997.
- [6] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3d convolutional networks,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497.
- [7] J. Carreira and A. Zisserman, “Quo vadis, action recognition? a new model and the kinetics dataset,” in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6299–6308.
- [8] W. Kay, J. Carreira, K. Simonyan, *et al.*, “The kinetics human action video dataset,” *arXiv preprint arXiv:1705.06950*, 2017.
- [9] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020.

- [11] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, “Vivit: A video vision transformer,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 6836–6846.
- [12] Z. Tong, Y. Song, J. Wang, and L. Wang, “Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training,” *Advances in neural information processing systems*, vol. 35, pp. 10 078–10 093, 2022.
- [13] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, “Masked autoencoders are scalable vision learners,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 000–16 009.
- [14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 2019, pp. 4171–4186.
- [15] D. C. Moura, S. Zhu, and O. Zvitia, *Nexar dashcam collision prediction dataset and challenge*, 2025. arXiv: 2503.03848 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2503.03848>.
- [16] F.-H. Chan, Y.-T. Chen, Y. Xiang, and M. Sun, “Anticipating accidents in dashcam videos,” in *Asian conference on computer vision*, Springer, 2016, pp. 136–153.
- [17] T. Suzuki, H. Kataoka, Y. Aoki, and Y. Satoh, “Anticipating traffic accidents with adaptive loss and large-scale incident db,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3521–3529.
- [18] W. Bao, Q. Yu, and Y. Kong, “Uncertainty-based traffic accident anticipation with spatio-temporal relational learning,” in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 2682–2690.
- [19] A. V. Malawade, S.-Y. Yu, B. Hsu, D. Muthirayan, P. P. Khargonekar, and M. A. Al Faruque, “Spatiotemporal scene-graph embedding for autonomous vehicle collision prediction,” *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9379–9388, 2022.
- [20] T. Wang, K. Chen, G. Chen, *et al.*, “Gsc: A graph and spatio-temporal continuity based framework for accident anticipation,” *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 2249–2261, 2023.
- [21] W. Bao, Q. Yu, and Y. Kong, “Drive: Deep reinforced accident anticipation with visual explanation,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 7619–7628.

- [22] J. Fang, L.-L. Li, K. Yang, Z. Zheng, J. Xue, and T.-S. Chua, “Cognitive accident prediction in driving scenes: A multimodality benchmark,” *arXiv preprint arXiv:2212.09381*, 2022.
- [23] M. G. Al-Thani, Z. Sheng, Y. Cao, and Y. Yang, “Traffic transformer: Transformer-based framework for temporal traffic accident prediction,” 2024.
- [24] P. Patera, Y.-T. Chen, and W.-H. Fang, “Spatio-temporal adaptation with dilated neighbourhood attention for accident anticipation,” in *2024 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2024, pp. 2452–2458.
- [25] Y. Kumamoto, K. Ohtani, D. Suzuki, M. Yamataka, and K. Takeda, “Aat-da: Accident anticipation transformer with driver attention,” in *Proceedings of the Winter Conference on Applications of Computer Vision*, 2025, pp. 1142–1151.
- [26] W. Song, S. Li, T. Chang, K. Xie, A. Hao, and H. Qin, “Dynamic attention augmented graph network for video accident anticipation,” *Pattern Recognition*, vol. 147, p. 110 071, 2024.
- [27] P. Endresen. “Nexar dashcam kaggle challenge,” Kaggle. (2025), [Online]. Available: <https://www.linkedin.com/pulse/nexar-dashcam-kaggle-challenge-paul-endresen-ayd2c> (visited on 05/11/2026).
- [28] siuuuuuuu. “2nd place solution (public & private lb): Videomaev2-based vehicle collision prediction,” Kaggle. (2025), [Online]. Available: <https://www.kaggle.com/competitions/nexar-collision-prediction/writeups/siuuuuuuu-2nd-place-solution-public-private-lb-vid> (visited on 05/11/2026).
- [29] [MZWAGER]. “3rd place solution: Two frames is all you need,” Kaggle. (2025), [Online]. Available: <https://www.kaggle.com/competitions/nexar-collision-prediction/discussion/578712> (visited on 05/11/2026).
- [30] Hi-f. “[private 0.853] my solution,” Kaggle. (2025), [Online]. Available: <https://www.kaggle.com/competitions/nexar-collision-prediction/writeups/hi-f-private-0-853-my-solution> (visited on 05/11/2026).
- [31] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, “On calibration of modern neural networks,” in *International conference on machine learning*, PMLR, 2017, pp. 1321–1330.
- [32] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.

- [33] R. Goyal, S. Ebrahimi Kahou, V. Michalski, *et al.*, "The "something something" video database for learning and evaluating visual common sense," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5842–5850.
- [34] S. Hasan. "Recurrent neural network and it's variants." (2020), [Online]. Available: <https://medium.com/@hasanshujaat4/recurrent-neural-network-and-its-variants-de75f9ee063> (visited on 05/11/2026).
- [35] Croner-i. "Advanced driver assistance and safety systems: In-depth." (2024), [Online]. Available: <https://app.croneri.co.uk/topics/advanced-driver-assistance-and-safety-systems/indepth> (visited on 05/11/2026).

UNIVERSITÉ CATHOLIQUE DE LOUVAIN

École polytechnique de Louvain

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | www.uclouvain.be/epl