Temporal DINO: A Self-supervised Video Strategy to Enhance Action Prediction

Izzeddin Teeti\textsuperscript{1,}\textsuperscript{*}, Rongali Sai Bhargav\textsuperscript{2}, Vivek Singh\textsuperscript{1}, Andrew Bradley\textsuperscript{1}, Biplab Banerjee\textsuperscript{2}, Fabio Cuzzolin\textsuperscript{1}

\textsuperscript{1}VAIL, Oxford Brookes University, (iteeti, vsingh, abradley, fabio.cuzzolin)@brookes.ac.uk
\textsuperscript{2}Indian Institute of Technology, Bombay, (sat.bhargav, Biplab)@iitb.ac.in

Abstract

The emerging field of action prediction - the task of forecasting action in a video sequence - plays a vital role in various computer vision applications such as autonomous driving, activity analysis and human-computer interaction. Despite significant advancements, accurately predicting future actions remains a challenging problem due to high dimensionality, complex dynamics and uncertainties inherent in video data. Traditional supervised approaches require large amounts of labelled data, which is expensive and time-consuming to obtain. This paper introduces a novel self-supervised video strategy for enhancing action prediction inspired by DINO (self-distillation with no labels). The approach, named Temporal-DINO, employs two models; a ‘student’ processing past frames; and a ‘teacher’ processing both past and future frames, enabling a broader temporal context. During training, the teacher guides the student to learn future context by only observing past frames. The strategy is evaluated on ROAD dataset for the action prediction downstream task using 3D-ResNet, Transformer, and LSTM architectures. The experimental results showcase significant improvements in prediction performance across these architectures, with our method achieving an average enhancement of 9.9% Precision Points (PP), which highlights its effectiveness in enhancing the backbones' capabilities of capturing long-term dependencies. Furthermore, our approach demonstrates efficiency in terms of the pretraining dataset size and the number of epochs required. This method overcomes limitations present in other approaches, including the consideration of various backbone architectures, addressing multiple prediction horizons, reducing reliance on hand-crafted augmentations, and streamlining the pretraining process into a single stage. These findings highlight the potential of our approach in diverse video-based tasks such as activity recognition, motion planning, and scene understanding. Code can be found at \url{https://github.com/IzzeddinTeeti/ssl_pred}.

1. Introduction

Computer vision techniques have advanced to the point at which they are able to outperform humans at certain object recognition tasks \cite{challenge55}. However, for many computer vision applications, a higher-level understanding of the scene is required. For example, achieving human-level performance in autonomous vehicles remains a formidable challenge \cite{challenge47}. One of the key reasons for this gap is the inherent difficulty in understanding what may happen next. Thus, there is a growing recognition of the importance of prediction. Prediction plays a crucial role in enhancing the decision-making process of autonomous systems by anticipating the future behaviour of dynamic elements in the environment, e.g. other vehicles, pedestrians, and cyclists - thus ensuring safer operation. Moreover, prediction also facilitates the development of high-level understanding within autonomous systems, enabling more nuanced and contextually appropriate planning, leading to smoother interactions with other agents on the road \cite{challenge30}.

However, prediction poses its own set of challenges, encompassing spatial, temporal, social, and stochastic dimensions \cite{challenge50}. Modelling these dimensions requires complex models, such as \cite{challenge43, challenge45, challenge61, challenge37}, which require significant amounts of data - often scarce and costly to gather and annotate. To address this, leveraging the abundance of unlabelled data through self-supervised methods offers an enticing opportunity to enhance performance with minimal impact upon resources.

While existing self-supervised prediction methods, including \cite{challenge54, challenge63, challenge27}, have shown promise, they have limitations that hinder their effectiveness. Firstly, their predictive capability is limited to a very short-term horizon (typically one frame ahead) which is impractical for autonomous driving scenarios requiring longer-term predictions. Secondly, these methods often involve a two-stage process \cite{challenge54}, which is computationally expensive and time-consuming. Finally, they are typically designed for a specific architecture, lacking the ability to generalize across different architectures.

In this paper, we present a novel one-stage self-supervised representation learning strategy specifically de-
signed for videos. Our proposed approach draws inspiration from the image-based self-supervised DINO [7] model and extends its application to the temporal dimension. Leveraging a student-teacher framework, our method guides the student model to focus on the most informative (temporal) features, enabling accurate predictions of future events. The student model learns to attend to the relevant cues in the past and current moments, extracting valuable information that aids in forecasting forthcoming actions. Our approach addresses the aforementioned limitations by significantly extending the prediction range beyond one frame, enabling more practical and effective autonomous driving applications. Moreover, our proposed method eliminates the need for a two-stage process, reducing computational complexity and saving valuable time. Finally, our approach is not limited to a single architecture but is a wrapper strategy that can be used to improve prediction performance across various architectures, enhancing the generalisability and applicability of the method. The contributions of this work include:

- A novel one-stage self-supervised representation learning strategy for videos, addressing limitations in prediction range, computational complexity, and architectural generalisability.

- Proof-of-concept evaluation of the approach in both autonomous driving, and a more general task - with various deep-learning architectures (including 3D-CNN, Transformer, and LSTM), showcasing its effectiveness on real-world challenging data, and versatility to use on other models and domains.

- Identification of the optimal model architecture and loss function for capturing long-range dependencies in video data, shedding light on the most effective design choices for robust action prediction in autonomous driving scenarios.

In the following sections, we provide a comprehensive overview of related work in self-supervised learning for images and videos, and action prediction (Section 2), followed by a detailed description of our proposed method (Section 3). We then present the experimental setup, including datasets, evaluation metrics, and training details (Section 4), and discuss the results and analysis of our experiments (Section 5). Finally, we provide a thorough discussion of our findings, and highlight potential applications and future research directions (Section 6).

2. Related Work

2.1. Image-based Self-supervised Learning

Two types of self-supervised strategies are commonly employed in computer vision: pretext methods and contrastive learning. The former involves utilizing specific internal properties or tasks of the input data to learn useful representations. For instance, some models learn the contextual information of the entire scene by analyzing small image patches [10]. Other approaches focus on tasks like re-colourization of grayscale images [64], predicting transformations between different views of the same image [14], or determining the order of patches extracted from an image [38, 35]. With the introduction of Vision Transformer (ViT) [11], patch-based self-supervised methods have gained prominence in the literature [21, 3, 2]. Particularly, masked auto-encoders (MAE) [21] have emerged as a preferred mechanism for model pretraining due to their superior performance on downstream tasks.

Contrastive learning methods, on the other hand, focus on maximizing the dissimilarity between features extracted
from different samples to encourage discriminative representations [12, 5, 58]. Some discriminative self-supervised algorithms leverage instance-level discrimination to create distinct feature representations for different examples, leading to robust feature learning [22, 9, 17, 7]. Other approaches draw inspiration from clustering mechanisms [6, 1]. The DINO method [7], for example, employs self-distillation that trains a student model on local crops and a teacher model on the entire image, then find the loss between both representations. The teacher will push the student to learn global representations by seeing only the local ones. Whilst these techniques offer significant potential on images, their performance is limited on video-related tasks, such as action prediction. Thus, there is a need for advancements in video self-supervised learning methods.

2.2. Video-based Self-supervised Learning

Video-based self-supervised learning strategies, akin to their image counterparts, encompass both pretext and contrastive approaches [46]. However, videos introduce an additional dimension, namely the temporal dimension. The inherent order of frames within a video sequence offers intrinsic properties that can be leveraged to develop effective self-supervised learning mechanisms. Despite this potential, the research in this specific domain remains relatively limited [51, 4, 36, 57].

To address this gap, recent works have explored different strategies in video-based self-supervised learning. For instance, [20] employed complementary information from RGB and optical flow streams to co-train their model using contrastive loss. [18] utilized a 3D-CNN architecture with a video transformer and trained the model using contrastive loss on positive pairs of video sequences. CVRL [42] employed a discriminative learning approach, utilizing two augmented views of the same video as positive examples and views of other videos as negatives. Another method, VideoMoCo [39], extended the MoCo [22] framework to videos by randomly dropping frames from the video and learning the same representation for each random input.

In line with these advancements, our proposed model introduces a novel approach where the teacher model incorporates a longer temporal sequence than the student model. This design choice aims to provide the student with a wider temporal context, encompassing future frames that the student has not yet observed. By leveraging this extended temporal context within the student-teacher framework, our model aims to enhance the student’s ability to learn representations that capture long-term dependencies and improve its performance in video-based self-supervised learning.

2.3. Supervised Action Prediction

Solving the action prediction task requires modelling the different dimensions of the problem, including the spatial, temporal, stochastic, and social dimensions, since the driving environment is dynamic, uncertain, and multi-agent. To model the temporal dimension, [13, 29] used 3D-CNN, [43, 16] Recurrent Neural Networks, while [61] used Transformers. Regarding the stochastic dimension, GANS [28], and CVAE models [16, 61] were utilised. To model the multi-agent aspect of the problem, different types of Graph Neural Networks of different connectivity, sparsity and homogeneity were used [15, 45, 16], semantic segmentation cues [44, 34] and social pooling [40]. However, all of these methods are supervised; they require a huge amount of labelled data which is time-consuming and expensive.

2.4. Self-supervised Action Prediction

In recent studies, limited approaches have been explored in the field of self-supervised action prediction. Zatsarynna et al. [63] employed the contrastive loss of InfoNCE [19] to encourage proximity between temporally adjacent video clips in the embedding space. To preserve the order of the clips, they complemented the InfoNCE loss with an order loss in the form of cross-entropy. Their proposed model focused on using 3D-CNN as the backbone architecture, and its evaluation was limited to this specific backbone without examining the performance on other architectures.

Another approach by Kochakarn et al. [27] utilized graph contrastive learning with the SimCLR loss function [9] to learn more informative embeddings for the prediction task. They incorporated an attention mechanism to achieve explainable action prediction, albeit limited to predicting only the next one frame. In contrast, our proposed method extends the prediction horizon to include the next 3, 6, and 12 frames, offering a more comprehensive temporal context. Furthermore, their approach specifically focused on graphs, while our method is designed to work with 3D-CNN, Transformers, and LSTMs, providing broader applicability.

It is worth noting that both of these aforementioned approaches rely heavily on the use of contrastive loss, which requires careful crafting of augmentations. This dependency on specific augmentations can limit the generalisability and robustness of the learned representations.

In contrast, Tran et al. [54] adopted a knowledge distillation approach, where a teacher model trained on recognition tasks transfers its knowledge to a prediction model. However, this method follows a two-stage process involving training a teacher model for recognition and then distilling the knowledge into a student model for prediction. This two-stage approach introduces additional time and computational costs, which may impact the scalability and practicality of the method, particularly in real-time or resource-constrained scenarios. Furthermore, their evaluation was focused on the I3D architecture [8] without exploring the performance on other architectures.
These related works provide valuable insights into self-supervised action prediction approaches. However, they have certain limitations in terms of the backbone architectures considered, the prediction horizons addressed, the reliance on specific augmentations, and the two-stage training process. In contrast, our one-stage proposed method aims to overcome these limitations by leveraging a different loss formulation and considering multiple backbone architectures while extending the prediction horizon, thereby contributing to the advancement of self-supervised action prediction techniques.

3. Methodology

3.1. Representation Learning for Action Prediction

In this study, our objective is to learn a mapping function \( f \) in an unsupervised manner, which takes an unlabelled and untrimmed video clip consisting of \( t \in \mathbb{R}^+ \) frames as input. The goal is to map the 4D input clip \( x_{1:t} \in \mathbb{R}^{T \times C \times H \times W} \), to a feature vector \( g(x_{1:t}) \in \mathbb{R}^d \), such that the learned features effectively transfer to the downstream task of action prediction. To achieve this, we draw inspiration from the DINO approach and adopt a student-teacher setup, as depicted in Figure 1.

The student network \( S_\theta \) processes only the past frames \( x_{1:t} \) during both training and inference, without access to future frames. On the other hand, the teacher network \( T_\phi \) processes both the past and future frames \( x_{1:t+t_{pred}} \), where \( t_{pred} \) denotes the length of the future sequence. To ensure consistency in architecture, we downsample the sequence processed by the teacher network to match the number of frames processed by the student. The downsampling is performed by determining a sampling frequency, which is calculated as \( \frac{(t+t_{pred})}{t} \). For instance, if \( t = 12 \) frames and \( t_{pred} = 12 \) frames, the student processes the past (12) frames, while the teacher processes (24 frames) with a step of 2, resulting in 12 frames. Despite processing sequences of the same sequence length, the teacher network has access to a wider temporal context.

3.2. Future-past Distillation Loss

During training, we introduce a knowledge distillation loss between the final embeddings of the student and teacher networks. This loss guides the student to distil knowledge about the future from the teacher, despite not having direct access to future frames. The aim is to teach the student to focus on the most relevant features from the past frames that contribute to predicting the future. In contrast to DINO, our Future-Past Distillation (FPD) loss is defined in the Cosine Similarity form, instead of using cross-entropy. This formulation is motivated by the findings of our ablation analysis, which indicate that the Cosine-based FPD loss yields improved performance on downstream tasks. The learning objective for the pretraining stage of future-past distillation is expressed in Equation 1. The student network parameters \( \theta \) are updated using backpropagation optimized by stochastic gradient descent (SGD), while the teacher network parameters \( \phi \) are updated using an Exponential Moving Average (EMA) based on the student network, with a scheduled momentum variable \( m \) as shown in Equation 2.

\[
\begin{align*}
\theta^*, \phi^* &= \arg \min_{\theta, \phi} \mathcal{L}_{FPD}(S_\theta(x_{1:t}), T_\phi(x_{1:t+t_{pred}})) \\
\phi_{i+1} &= m_i \times \phi_i + (1 - m_i) \times \theta_i
\end{align*}
\]

3.3. Downstream Task Definition

The objective of pretraining is to enhance the performance of the model in the downstream task of predicting driver’s actions. Given a past (observed) clip of length \( t \in \mathbb{R}^+ \), the task is to predict the Ego-vehicle (driver) action in each frame in the next \( t_{pred} \) frames. Building upon the optimal pretrained student model \( S_{\theta^*} \), the prediction model \( f \) will use the student model as a backbone and add a classification head on top of it. Subsequently, it performs further optimization (fine-tuning) on either both the backbone and the head parameters or solely the latter (we conducted experiments on both scenarios) for the prediction task. The objective is to map the learned features \( S_{\theta^*}(X_{1:t}) \) to the future action labels \( y_{t+1:t+t_{pred}} \), formally, \( f_\psi : S_{\theta^*}(X_{1:t}) \rightarrow y_{t+1:t+t_{pred}} \). Given the nature of a classification task, cross-entropy (CE) loss is utilized to guide the optimization process and refine the learning objective for the downstream task, as illustrated in Equation 3.

\[
\begin{align*}
\theta^{**}, \psi^* &= \arg \min_{\theta^*, \psi} \mathcal{L}_{CE}(f_\psi(S_{\theta^*}(x_{1:t})), y_{t+1:t+t_{pred}})
\end{align*}
\]

4. Experiments

4.1. Datasets

We adhered to the convention in self-supervised learning, utilizing two distinct datasets for distinct purposes: a larger dataset for pretext task pretraining and a smaller dataset for downstream task fine-tuning. Specifically, we employed the Kinetics-400 dataset [25] and the ROad event Awareness Dataset (ROAD) [48].

Kinetics-400 It is designed for action recognition and comprises over 240,000 videos. On average, each video spans 10 seconds and is assigned a single label from a pool of 400 possible action classes. The dataset’s substantial video collection has facilitated its adoption in numerous
video self-supervised methods [51, 39]. For our purposes, we disregard the label information as it will not be used during pretraining.

**ROAD**
The ROad event Awareness Dataset (ROAD) [48] is built on a fraction of Oxford RobotCar Dataset [33], and it is extended with multi-label annotations for action recognition, localisation, and prediction tasks within the context of autonomous driving. It comprises 22 videos from an egocentric view as shown in Figure 2, each with an 8-minute duration and a frame rate of 12 frames per second (fps). Importantly, it contains labels indicating the actions performed by the ego vehicle (driver). Notably, the dataset encompasses seven distinct ego-vehicle actions: Move, Stop, Turn Left, Turn Right, Overtake, Move Left, and Move Right. These labels serve as training data for fine-tuning our models to predict the driver’s actions in future frames during the prediction task.

### 4.2. Models

We conducted a series of experiments utilizing four diverse deep-learning architectures.

**R3D** [52]
This architecture employs a 3D convolutional neural network (3D-CNN) backbone for processing video data. It is based on the ResNet-18 [23] architecture, which has proven to be successful in image recognition tasks.

**Swin** [31]
This architecture utilizes a Transformer-based 3D backbone for video processing. It is based on the Transformer architecture, originally introduced in the context of image recognition as Vision Transformer (ViT) [11].

**ResNet-LSTM**
This architecture combines a convolutional neural network (CNN)-based 2D backbone using the ResNet-50 architecture for image processing, along with a Long Short-Term Memory (LSTM) layer for capturing temporal dependencies.

**ViT-LSTM**
Same as the previous but replacing the ResNet with a Transformer-based 2D backbone.

The first two models use spatio-temporal backbones, which extract spatial and temporal features simultaneously, while the last two use a spatial backbone and connect it using a temporal one. These models exhibit differences in their depth and working mechanisms, offering a diverse range of approaches to our experiments. By leveraging these varied architectures, we can comprehensively investigate our proposed representation learning strategy, examine their performance, and compare their effectiveness according to the experiments outlined in the subsequent section.

### 4.3. Experimental Protocol

In accordance with the conventions of self-supervised learning experiments, we evaluated the performance of our models on the downstream task of action prediction under three distinct protocols:

1) **Full-Supervised**: This protocol represents the results obtained from the model without employing the pretraining strategy. The model was trained solely on the labelled data of the prediction task.

2) **Linear Probing**: In this protocol, the proposed strategy was applied, and the model was fine-tuned by solely updating the parameters of the prediction head. The backbone of the model was kept frozen during this fine-tuning process.

3) **Fine-tuning**: This protocol involved applying the proposed strategy and performing full fine-tuning of the entire network, including both the backbone and the prediction head. All parameters of the model were updated during the fine-tuning process.

By examining the model’s performance across these three protocols, we can gain insights into the effectiveness and impact of the pretraining strategy on action prediction.

### 4.4. Implementation Details

We used Pytorch framework [41] to implement the models, and the Precision (P) metric was used to evaluate their performance on the action prediction task. All experiments were conducted on NVIDIA A30 graphics cards. Below are the details of the pretraining and fine-tuning.

**Pretraining**
For pretraining, we utilized either the full dataset of Kinetics-400 or the training split of ROAD (depending on the experiment). The pretraining was optimised for 1000 (50) epochs using SGD optimiser with a learning rate of 0.005 (0.001) and a batch size of 64 (32) for Kinetics-400 and ROAD, respectively. Additionally, a cosine scheduler was employed for updating the momentum variable of the exponential moving average (EMA). Concerning the models, the LSTM architecture had a hidden dimension of 512, and the small structure of ViT (ViT-s) was used.

**Fine-tuning**
Fine-tuning was performed on the ROAD dataset, with a data split of 60% for training, 20% for validation, and 20% for testing. Cross-entropy loss was employed as the objective function for fine-tuning, and it was optimised for 10 epochs using the SGD optimizer with a learning rate of 0.001, and a batch size of 32.

### 4.5. Ablations

In this section, we present different ablations to study the effects of different architectural components on the model.
Backbone and Temporal Length  The choice of model backbone plays a critical role in the model’s ability to learn effective features. Similarly, the length of the input sequence contributes to the model’s capability to capture temporal dynamics and visual changes within the scene. Longer sequence lengths generally provide more visual data, but they may also introduce challenges in modelling long-term dependencies and potentially lead to performance degradation, as shown in Table 1.

In this specific ablation study, We conducted experiments with the four backbones mentioned in Section 4.2, and we varied the temporal depth by using sequence lengths of 3, 6, and 12 frames for each backbone. We trained these models using the three experimental protocols outlined in Section 4.3. The prediction performance of the four models with different input lengths under the three protocols is summarized in Table 1.

Loss Function  Existing literature demonstrates that the choice of learning objective in self-supervised algorithms significantly influences the performance of models on downstream tasks [46]. In our experiments, we employed three widely used loss functions: Cross-entropy, Cosine Similarity, and Mean Squared Error (MSE) loss. The evaluation of these loss functions was performed on R3D and Swin backbones, and the corresponding results are summarized in Table 2.

Action Recognition  In addition, we performed supplementary experiments to assess the impact of our proposed strategy on a different video-based downstream task, namely Action Recognition. For this purpose, the models underwent pretraining on the Kinetics-400 dataset and subsequent fine-tuning on the UCF101 dataset [49], shown in Figure 2. Table 4 presents the state-of-the-art (SOTA) performance for action recognition.

5. Results  Observing Table 1, it is evident that our proposed strategy yielded notable enhancements in the prediction performance of all backbone architectures, albeit to varying extents. Notably, the impact of the strategy is particularly pronounced on the 2D-based backbones, which initially exhibited comparatively lower results in the fully-supervised setup. This observation aligns with expectations since these backbones lack inherent spatio-temporal modelling capabilities. However, when trained with T-DINO, the ResNet-LSTM and ViT-LSTM backbones witnessed substantial improvements, with average increases of 22.5 and 10.1 in terms of precision points (PP), respectively. Additionally, the video backbones, R3D and Swin, experienced PP gains of 5.2 and 1.8, respectively. Table 3 shows the comparison of our strategy to other SOTA methods, showing that it surpasses the performance of both supervised and self-supervised approaches on action prediction on ROAD. Within the same table, the column ‘Supervised’ indicates a fully supervised method, the opposite denotes a self-supervised approach, and ‘Frozen’ means the fine-tuning process exclusively updates the classification head while leaving the pre-trained backbone untouched, the opposite indicates that the entire model was updated during the fine-tuning.
Table 1: The precision of different backbones with varying input lengths under the three protocols mentioned in Section 4.3.

<table>
<thead>
<tr>
<th>Backbone</th>
<th>Pretrained on</th>
<th>Interval (frames)</th>
<th>Linear Probe</th>
<th>Fine-tuning</th>
<th>Supervised</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>R3D ROAD</td>
<td>3</td>
<td>36.6</td>
<td>77.7</td>
<td>74.1</td>
<td>4.3</td>
<td>3.6</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>29.7</td>
<td>69.2</td>
<td>64.9</td>
<td>4.3</td>
<td>3.6</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>36.2</td>
<td>53.3</td>
<td>45.7</td>
<td>7.6</td>
<td>7.6</td>
</tr>
<tr>
<td>Swin ROAD</td>
<td>3</td>
<td>84.7</td>
<td>87.2</td>
<td>86.4</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>75.8</td>
<td>82.6</td>
<td>81.7</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>60.1</td>
<td>60.9</td>
<td>57.1</td>
<td>3.8</td>
<td>3.8</td>
</tr>
<tr>
<td>ResNet+LSTM Kinetics-400</td>
<td>3</td>
<td>77.6</td>
<td>84.6</td>
<td>62.9</td>
<td>21.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>70.3</td>
<td>81.8</td>
<td>58.3</td>
<td>23.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>58.3</td>
<td>76.7</td>
<td>54.3</td>
<td>22.4</td>
<td></td>
</tr>
<tr>
<td>ViT+LSTM Kinetics-400</td>
<td>3</td>
<td>73.5</td>
<td>77.7</td>
<td>69.3</td>
<td>8.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>70.2</td>
<td>76.7</td>
<td>65.5</td>
<td>11.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>64.21</td>
<td>66.2</td>
<td>55.5</td>
<td>10.7</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Evaluation of three common loss functions on R3D and Swin backbones.

<table>
<thead>
<tr>
<th>Backbone</th>
<th>Loss</th>
<th>Linear Probe</th>
<th>Fine-tuning</th>
<th>Supervised</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>R3D</td>
<td>MSE</td>
<td>32.0</td>
<td>37.3</td>
<td>45.7</td>
<td>-8.4</td>
</tr>
<tr>
<td></td>
<td>Cosine</td>
<td>36.2</td>
<td><strong>53.3</strong></td>
<td>45.7</td>
<td>7.6</td>
</tr>
<tr>
<td></td>
<td>Cross-entropy</td>
<td>30.9</td>
<td>50.4</td>
<td>45.7</td>
<td>4.7</td>
</tr>
<tr>
<td>Swin</td>
<td>MSE</td>
<td>57.2</td>
<td>57.4</td>
<td>57.1</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>Cosine</td>
<td>60.1</td>
<td><strong>60.9</strong></td>
<td>57.1</td>
<td>3.8</td>
</tr>
<tr>
<td></td>
<td>Cross-entropy</td>
<td>49.6</td>
<td>58.4</td>
<td>57.1</td>
<td>1.3</td>
</tr>
</tbody>
</table>

To have a better understanding of the generalisation capability of the proposed model, we compare the performance of T-DINO with SOTA methods on another task, human action recognition (on UCF101), summarised in Table 4. The results highlight the effectiveness of the enhanced temporal modelling offered by T-DINO when applied to the R3D backbone.

Furthermore, examining the results for varying input sequence intervals across each backbone, it becomes apparent that greater improvements are observed at longer input sequences. This suggests that pretraining with T-DINO equips the backbones with enhanced abilities to capture and model long-term dependencies in the data. Notably, the Swin-transformer-based models demonstrate higher accuracy, attributed to the superior representation capabilities offered by Transformers. We observed that transformer-based spatial feature extractor (ViT) combined with LSTM-based temporal sequence modelling results in a huge improvement in the downstream task. Analyzing the results obtained from the longest input configuration consisting of 12 frames, it is evident that models pretrained on the larger dataset, Kinetics-400, exhibit superior performance compared to those pretrained on the ROAD dataset. Moreover, the models that incorporate separate modelling of spatial and temporal relationships, such as ViT+LSTM and ResNet+LSTM, outperform the models that jointly model these relationships, namely R3D and Swin, by a margin of 16.6 and 5.7 percentage points, respectively.

In regard to the selection of the most optimal loss function, the findings presented in Table 2 indicate that T-DINO pretrained using the Cosine Similarity loss surpasses the performance of models trained with MSE or Cross-entropy loss.

Of significant importance, self-supervised models typically require extensive datasets and a high number of pretraining epochs to achieve satisfactory generalization on downstream tasks. However, our proposed strategy, pretrained on the ROAD dataset, exhibits a relatively comparable level of performance to models pretrained on the larger Kinetics-400 dataset. Notably, the ROAD dataset possessed a substantially smaller size and was pretrained with a lower number of epochs. These findings demonstrate T-DINO’s resource and time efficiency in achieving desirable results.
Table 3: Comparison of the proposed methods with SOTA on ROAD for the action prediction task when the input length is 12 frames

<table>
<thead>
<tr>
<th>Method</th>
<th>Pretrained on</th>
<th>Backbone</th>
<th>Supervised</th>
<th>Frozen</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>R3D [53]</td>
<td>Road</td>
<td>R3D</td>
<td>✓</td>
<td>✗</td>
<td>45.7</td>
</tr>
<tr>
<td>Swin [32]</td>
<td>Road</td>
<td>Swin</td>
<td>✓</td>
<td>✗</td>
<td>57.1</td>
</tr>
<tr>
<td>Resnet+LSTM [62]</td>
<td>Road</td>
<td>ResNet50</td>
<td>✓</td>
<td>✗</td>
<td>54.3</td>
</tr>
<tr>
<td>ViT+LSTM</td>
<td>Road</td>
<td>ViT</td>
<td>✓</td>
<td>✗</td>
<td>55.5</td>
</tr>
<tr>
<td>VideoMAE [51]</td>
<td>Kinetics</td>
<td>ViT</td>
<td>✗</td>
<td>✗</td>
<td>75.1</td>
</tr>
<tr>
<td>T-DINO (ours)</td>
<td>Road</td>
<td>R3D</td>
<td>✗</td>
<td>✓</td>
<td>53.3</td>
</tr>
<tr>
<td>T-DINO (ours)</td>
<td>Kinetics</td>
<td>ResNet+LSTM</td>
<td>✗</td>
<td>✓</td>
<td>58.3</td>
</tr>
<tr>
<td>T-DINO (ours)</td>
<td>Kinetics</td>
<td>ResNet+LSTM</td>
<td>✗</td>
<td>✓</td>
<td>76.7</td>
</tr>
<tr>
<td>T-DINO (ours)</td>
<td>Kinetics</td>
<td>ViT+LSTM</td>
<td>✗</td>
<td>✓</td>
<td>64.2</td>
</tr>
<tr>
<td>T-DINO (ours)</td>
<td>Kinetics</td>
<td>ViT+LSTM</td>
<td>✗</td>
<td>✓</td>
<td>66.2</td>
</tr>
</tbody>
</table>

Table 4: Comparison of the proposed methods with SOTA on UCF101 for action recognition task.

<table>
<thead>
<tr>
<th>Method</th>
<th>Year</th>
<th>Pretrained on</th>
<th>Arch.</th>
<th>Supervised</th>
<th>Frozen</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClipOrder [59]</td>
<td>2019</td>
<td>UCF101</td>
<td>R3D</td>
<td>✗</td>
<td>✗</td>
<td>72.40</td>
</tr>
<tr>
<td>3D ST-puzzle [26]</td>
<td>2019</td>
<td>Kinetics</td>
<td>C3D</td>
<td>✗</td>
<td>✗</td>
<td>65.80</td>
</tr>
<tr>
<td>Wang et al. [56]</td>
<td>2019</td>
<td>UCF101</td>
<td>C3D</td>
<td>✗</td>
<td>✗</td>
<td>61.20</td>
</tr>
<tr>
<td>PRP [60]</td>
<td>2020</td>
<td>Kinetics</td>
<td>R3D</td>
<td>✗</td>
<td>✗</td>
<td>72.10</td>
</tr>
<tr>
<td>VideoMoCo [39]</td>
<td>2021</td>
<td>Kinetics</td>
<td>R(2+1)D</td>
<td>✗</td>
<td>✗</td>
<td>78.7</td>
</tr>
<tr>
<td>CACL [18]</td>
<td>2022</td>
<td>UCF101</td>
<td>R(2+1)D</td>
<td>✗</td>
<td>✗</td>
<td>82.5</td>
</tr>
<tr>
<td>VideoMAE [51]</td>
<td>2022</td>
<td>Kinetics</td>
<td>ViT</td>
<td>✗</td>
<td>✗</td>
<td>96.1</td>
</tr>
<tr>
<td>T-DINO (ours)</td>
<td>2023</td>
<td>Kinetics</td>
<td>ResNet+LSTM</td>
<td>✗</td>
<td>✗</td>
<td>80.26</td>
</tr>
<tr>
<td>T-DINO (ours)</td>
<td>2023</td>
<td>Kinetics</td>
<td>R3D</td>
<td>✗</td>
<td>✗</td>
<td>83.9</td>
</tr>
<tr>
<td>T-DINO (ours)</td>
<td>2023</td>
<td>Kinetics</td>
<td>ViT+LSTM</td>
<td>✗</td>
<td>✗</td>
<td>85.86</td>
</tr>
</tbody>
</table>

6. Conclusion

This study represents the first attempt to leverage future information in a ‘past training’ model, and the promising results indicate that this teacher-student approach could provide a significant performance improvement in various prediction tasks across a number of ubiquitous model architectures in a variety of different domains - without the requirement for any additional training data. Our proposed strategy, called Temporal-DINO, leverages a teacher-student self-distillation architecture to guide the student model to learn future temporal context by observing the past only. Unlike other approaches that involve a two-stage process or rely on hand-crafted augmentations with limited prediction horizons, our one-stage strategy overcomes these limitations. Additionally, ablations highlight the strategy’s generalisability, efficiency, and feasibility in hardware-constrained applications.

In terms of future directions, several avenues can be pursued to further enhance and expand our proposed approach. Firstly, the inclusion of Graph Neural Networks (GNNs) as additional architectural variations could be explored to examine the strategy’s ability to enhance the social dimension modelling. Secondly, expanding the evaluation of our approach to encompass a broader range of datasets from diverse domains.

References


