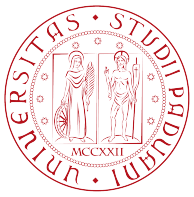



**University of Padova**

Department of Mathematics

Ph.D. School in Brain, Mind and Computer Science

Computer Science and Innovation for Societal Challenges

XXXV Series


# Prediction of Activities and Visual Concepts Under Complex and Changing Conditions


*Ph.D. Candidate*
Guglielmo Camporese

*Supervisor*
Lamberto Ballan

*Co-Supervisor*
Gianluca Campana

# Abstract

In the last years, computer vision dramatically changed because of the deep learning systems that have overtaken in most of the tasks the performances of previous models establishing a new way of thinking about vision problems. Despite the success on traditional computer vision tasks, our systems are still a long way from the general visual intelligence of people. In this dissertation, I will discuss my findings on different problems related to the visual prediction of activities and visual concepts under complex and changing conditions. A core problem of visual intelligence is the capability of anticipating future events on videos given the current knowledge state and, to achieve such predictive capabilities, specific vision systems have to be developed for encoding current representations and creating hypotheses of future scenarios. In this dissertation, I will discuss different directions I proposed for reaching predictive capabilities of vision systems based on semantic label smoothing of future actions, representing videos with slow and fast temporal scales, predicting latent goals, and prototyping future action representations. Another challenge of visual intelligence is related to recognizing unknown visual concepts that are not previously seen by the visual system. In this thesis, I will discuss my work on open set recognition where the vision model has to detect unknown classes not seen during training, maintaining the recognition capability on previously seen categories. Another core task related to the prediction of visual concepts is the representation learning problem where the model has to learn good representations from visual input, without any supervision. In this context, in this thesis, I will discuss my proposal on how vision transformers can learn efficiently good representations on small datasets by designing self-supervised tasks based on spatial relations of input patches.

# Acknowledgments

I would like to thank enormously my advisor Lamberto Ballan for guiding me as a researcher during the last three years in my PhD. Thank you to my coauthors, all of whom helped shape my research: Pasquale Coscia, Antonino Furnari, Giovanni Maria Farinella, Yunrui Guo, Alessandro Sperduti, Valentin Mendelev, Tina Raissi, Manuel Giollo, Nada Osman, Elena Izzo, Alessandro Bergamo, Xunyu Lin, Joe Tighe, and Davide Modolo. I am thankful to all members of the Visual Intelligence and Machine Perception (VIMP) group, past and present, for all the great discussions and for all the precious time spent together: Tommaso Campari, Enrico Cancelli, Luigi Filippo Chiara, and Sourav Das. During my Ph.D., I had wonderful internships at Amazon Alexa AI in Turin with the European team, and at Amazon Web Services AI Labs with the Seattle team; I am thankful to both teams for growing me as a scientist. I am enormously thankful to my parents Igino, and Antonella for giving me a good head and for teaching me the great thinking, and to my brother Pierluigi and my sister Maria Giulia for supporting my passions. Finally, I would like to thank my source of inspiration, my girlfriend Elena for constantly believing in me.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Background

In the last years, the capabilities of computer vision systems dramatically improved given the fast development of deep learning. For this reason, we moved from old low-level computer vision applications toward more challenging, complex, and interesting problems such as autonomous driving, video generation from language, 3D mesh reconstruction, and many more. One interesting and important problem that recently has gained attention is the prediction of future activities on videos. This problem is at the core of several real-case scenarios and can be beneficial for lots of applications such as autonomous driving and human-robots interaction. Another recent important problem is the ability of a recognition system not only to accurately generalize to unseen samples of known classes but also to detect if a new sample belongs to a new unknown class. This is the open set recognition problem and it's important as lots of real applications involving classification models need to handle new unseen classes. So far, we introduced all recent problems that usually required annotations in order to train working models. However, another important field of investigation that we worked on called Self-Supervised Learning (SSL), focuses on training models without annotations. These SSL methods proved to train strong model representations, however, the amount of data required for having benefits is huge.

In this dissertation, we investigated in-depth different problems related to the prediction of activities and the prediction of visual concepts under complex and changing conditions. We started with the prediction of activities problem where a model has to forecast the future action in a video from the past and current video observation. We first investigated this task in an ego-centric scenario where the model has to forecast future human complex activities. Subsequently, we focused on the scenario of pedestrian intent prediction in an urban environment. Finally, we worked on the early action recognition of different natures: sports videos, videos with human-object interaction, and human-human interaction. From the activity prediction problem we investigated the problem of the open set recognition task where a model trained for classification, not only has to recognize known classes but also has to detect unknown unseen classes. Afterward, we worked on improving the training of vision transformers on small datasets by designing a set of self-supervised tasks based on spatial relations between input patches extracted from the model.

## 1.2 Contributions

In the following chapters we discuss all the contributions done in the different works investigated and here we reported a summary of each contribution:

**Action Anticipation with Semantic Labels Smoothing.** In Chapter 2 we focused on the action anticipation problem on videos, developing a new label smoothing method that inject the future action semantics in the loss function. Since multiple actions may equally occur in the future, we treat action anticipation as a multi-label problem with missing labels extending the concept of label smoothing. This idea resembles the knowledge distillation process since useful information is injected into the model during training. We implement a multi-modal framework based on long short-term memory (LSTM) networks to summarize past observations and make predictions at different time steps. We perform extensive experiments on EPIC-Kitchens and EGTEA Gaze+ datasets including more than 2500 and 100 action classes, respectively. The experiments show that label smoothing systematically improves performance of state-of-the-art models for action anticipation.

**Action Anticipation with Slow and Fast Temporal Scales.** Action anticipation in egocentric videos is a difficult task due to the inherently multi-modal nature of human actions. Additionally, some actions happen faster or slower than others depending on the actor or surrounding context which could vary each time and lead to different predictions. Based on this idea, in Chapter 3 we discuss our proposed model where we build upon RULSTM architecture, which is specifically designed for anticipating human actions, and propose a novel attention-based technique to evaluate, simultaneously, slow and fast features extracted from three different modalities, namely RGB, optical flow and extracted objects. Two branches process information at different time scales, i.e., frame-rates, and several fusion schemes are considered to improve prediction accuracy. We perform extensive experiments on EpicKitchens-55 and EGTEA Gaze+ datasets, and demonstrate that our technique systematically improves the results of RULSTM architecture for Top-5 accuracy metric at different anticipation times.

**Early Intent Prediction with Latent Goal Estimation.** Anticipating human motion is an essential requirement for autonomous vehicles and robots in order to primary guarantee people's safety. In urban scenarios, they interact with humans, the surrounding environment, and other vehicles relying on several cues to forecast crossing or not crossing intentions. For these reasons, this challenging task is often tackled using both visual and non-visual features to anticipate future actions from 2 $s$ to 1 $s$ earlier the event. Our work, discussed in Chapter 4, primarily aims to revise this standard evaluation protocol to forecast crossing events as early as possible. To this end, we conceive a solution upon an extensively used model for egocentric action anticipation (RU-LSTM), proposing to envision future features, or modalities, that can better infer human intentions using a properly attention-based fusion mechanism. We validate our model against JAAD and PIE datasets and demonstrate that an intent prediction model can benefit from these additional clues for anticipating pedestrians crossing events.

**Early Action Recognition with Action Prototypes.** Anticipating the future activities in a visual input is an important and challenging problem as it enables the model to predict a future event before it ends or even it starts. Early recognition

of actions in videos is challenging as it requires to predict an activity defined on the entire video, starting from a partially observed one. In Chapter 5 we propose to learn a prototypical representation of the full action realization for each action class, and to use them as source of regularization for the model during the training phase. Specifically, we learn the prototypes and at the same time we let the model to predict them from all the partial observations of the video. In this way, the model not only tries to recognize the action from a partial observation but tries to predict the overall general action prototypical representation. We conducted several experiments and thanks to our method we achieved state-of-the-art results on different video datasets of different nature.

**Open Set Recognition.** In open set recognition, a classifier has to detect unknown classes that are not known at training time. In order to recognize new categories, the classifier has to project the input samples of known classes in very compact and separated regions of the features space for discriminating samples of unknown classes. Recently proposed Capsule Networks have shown to outperform alternatives in many fields, particularly in image recognition, however they have not been fully applied yet to open-set recognition. In capsule networks, scalar neurons are replaced by capsule vectors or matrices, whose entries represent different properties of objects. In our proposal in Chapter 6, during training, capsules features of the same known class are encouraged to match a pre-defined gaussian, one for each class. To this end, we use the variational autoencoder framework, with a set of gaussian priors as the approximation for the posterior distribution. In this way, we are able to control the compactness of the features of the same class around the center of the gaussians, thus controlling the ability of the classifier in detecting samples from unknown classes. We conducted several experiments and ablation of our model, obtaining state of the art results on different datasets in the open set recognition and unknown detection tasks.

**Self-Supervised Learning Through Spatial Relations.** Vision Transformers (ViTs) enabled the use of the transformer architecture on vision tasks showing impressive performances when trained on big datasets. However, on relatively small datasets, ViTs are less accurate given their lack of inductive bias. To this end, in Chapter 7 we propose a simple but still effective self-supervised learning (SSL)

strategy to train ViTs, that without any external annotation or external data, can significantly improve the results. Specifically, we define a set of SSL tasks based on relations of image patches that the model has to solve before or jointly the supervised task. Differently from ViT, our RelViT model optimizes all the output tokens of the transformer encoder that are related to the image patches, thus exploiting more training signals at each training step. We investigated our methods on several image benchmarks finding that RelViT improves the SSL state-of-the-art methods by a large margin, especially on small datasets.

**Speech Recognition for Speech Disfluencies.** Automatic Speech Recognition (ASR) based on Recurrent Neural Network Transducers (RNN-T) is gaining interest in the speech community. In Chapter 8, we investigate data selection and preparation choices aiming for improved robustness of RNN-T ASR to speech disfluencies with a focus on partial words. For evaluation we use clean data, data with disfluencies and a separate dataset with speech affected by stuttering. We show that after including a small amount of data with disfluencies in the training set the recognition accuracy on the tests with disfluencies and stuttering improves. Increasing the amount of training data with disfluencies gives additional gains without degradation on the clean data. We also show that replacing partial words with a dedicated token helps to get even better accuracy on utterances with disfluencies and stutter. The evaluation of our best model shows 22.5% and 16.4% relative WER reduction on those two evaluation sets.

# Chapter 2

# Action Anticipation with Semantic Labels Smoothing

## 2.1 Introduction

[1]Human action analysis is a central task in computer vision that has a enormous impact on many applications, such as, video content analysis [7, 100], video surveillance [96, 190], and intelligent transportation [11, 139]. Systems interacting with humans also need the capability to promptly react to the context changes, and plan their actions accordingly. Most previous works focus on the tasks of action recognition [117, 183, 55] or early-action recognition [172, 15, 34], i.e., recognition of an action *after* its observation (happened in the past) or recognition of an *on-going* action from its partial observation (only part of the current action is available). A more challenging task is to predict near future, i.e., to forecast actions that will be performed ahead in time. Predicting future actions before observing the corresponding frames [30, 64] is demanded by many applications which need to anticipate human behaviour. For example, intelligent surveillance systems may support human operators to avoid hazards or assistive robotics may help non-self-sufficient people. Such task requires to

[1]**G. Camporese**, P. Coscia, A. Furnari, G. M. Farinella, L. Ballan. "Knowledge Distillation for Action Anticipation via Label Smoothing", *Proc. of International Conference on Pattern Recognition (ICPR 2020).*

analyze significant spatio-temporal variations among actions performed by different people. For this reason, multiple modalities (e.g., appearance and motion) are typically considered to improve the identification of similar actions. Egocentric scenarios provide useful settings to study early-action recognition or action anticipation tasks. Indeed, wearable cameras offer an explicit point-of-view to capture human motion and object interaction.

In this work, we address the problem of anticipating egocentric human actions in an indoor scenario at several time steps. More specifically, we aim to anticipate an action by leveraging its previous video segments. We employ the encoding-decoding scheme of [64]. During the first stage, the model summarizes the video content extracting relevant information to infer the future while, in the second stage, predicts the next action at multiple time-steps (e.g., $1.5s - 1.0s - ... - 0.25s$ before the action to be predicted). Fig. 2.1 depicts such procedure. We exploit an LSTM-based network to capture temporal correlations among video frames and three different modalities as input representation: *appearance* (RGB), *motion* (optical flow) and *object-based* features.

An important aspect to consider when dealing with human actions is the future's uncertainty since multiple actions may equally occur. For example, the actions "*sprinkle over pumpkin seeds*" and "*sprinkle over sunflower seeds*" may be equally performed when preparing a recipe. Nevertheless, assigning zero-probability to uncorrect yet semantically similar actions may lead predictions to not capture data structure randomly selecting from a set of plausible targets.

For this reason, to deal with the uncertainty of future predictions, we propose several label smoothing techniques which broaden the set of possible futures and reduce the uncertainty caused by one-hot encoded labels. Label smoothing is introduced in [193] as a form of regularization for classification models since it introduces a positive constant value into wrong classes components of one-hot encoded targets. Compared to the typical knowledge distillation process where soft-labels are provided by an external network (the teacher) to train a smaller network (the student), our model distills knowledge extracted from prior of action labels (e.g., verb-noun). We extensively test our architecture on EPIC-Kitchens and EGTEA Gaze+ datasets

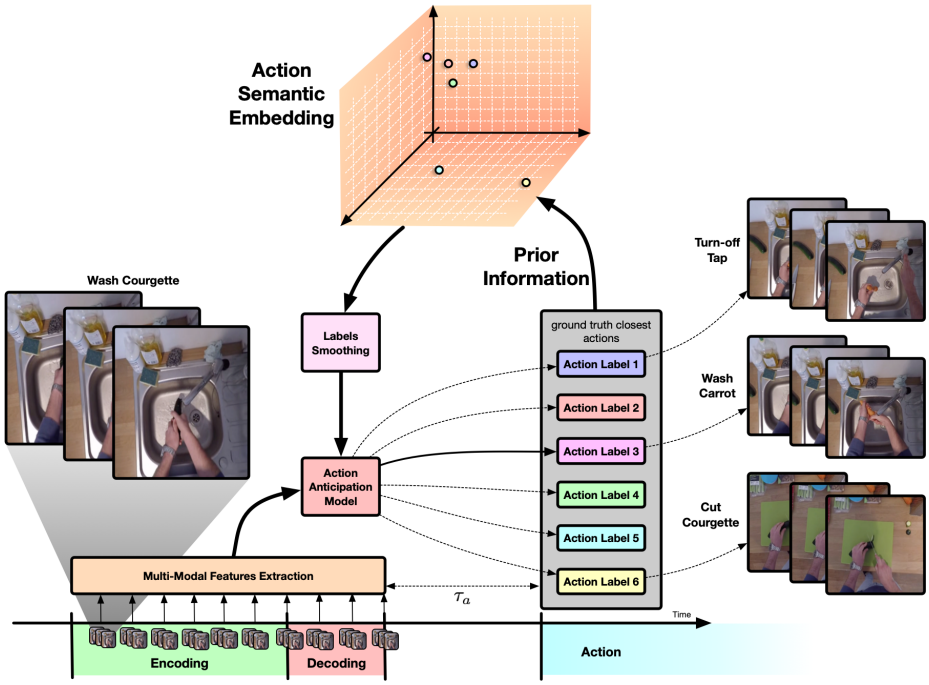



Figure 2.1: To anticipate next actions in egocentric scenarios, our framework processes input videos summarizing useful information during the encoding stage. In the decoding stage, our model predicts the next action at different timesteps. Label smoothing techniques are employed to reduce the uncertainty on future predictions.

which record egocentric scenarios where different people are involved to complete a recipe performing numerous actions and involving different objects within a kitchen. Our results show that label smoothing increases the performance of state-of-the-art models and yields better generalization on test data.

The main contributions of our work are as follows: 1) we generalize the label smoothing idea extrapolating semantic priors from the action labels to capture the multi-modal future component of the action anticipation problem. We show that label smoothing, in this context, can be seen as a knowledge distillation process where the teacher gives semantic prior information for the action anticipation model; 2) we perform extensive experiments on egocentric videos proving that our label smoothing techniques systematically improve results of action anticipation of state-of-the-art models.

## 2.2 Related Work

Action anticipation requires the interpretation of the current activity using a number of observations in order to foresee the most likely actions. For this reason, we briefly review three related research areas: *action recognition, early-action recognition* and *action anticipation*.

***Action recognition.*** Action recognition is the task of recognizing the action contained in an observed trimmed video. Classic approaches to action recognition have leveraged hand-designed features, coupled with machine learning algorithms to recognize actions from videos [117, 209, 210]. More recent works have investigated the use of deep learning to obtain representations suitable for action recognition directly from videos in an end-to-end fashion. Among these approaches, a line of works has investigated ways to exploit standard 2D CNNs for action recognition, often relying on optical flow as a mean to represent motion [183, 54, 55, 213, 237, 126]. Other works have focused on the extension of 2D CNNs to 3D CNNs able to process spatio-temporal volumes [100, 197, 199]. Some approaches have used recurrent networks to model the temporal relationships between per-frame observations [42, 189, 64]. All of these works investigate deeply how to represent and leverage the input video but

less or even no importance is given to the representation of the action labels.

***Early-action recognition.*** Early action recognition consists in recognizing ongoing actions from partial observations of streaming video [34]. Classic works have addressed the task using integral histograms of spatio-temporal features [172], sparse-coding [15], Structured Output SVMs [89], and Sequential Max-Margin event detectors [92]. Another line of research has leveraged the use of LSTMs [3, 8, 35, 64] to account for the sequential nature of the task.

***Action anticipation.*** Action anticipation deals with forecasting actions that will happen in the future. Previous studies have investigated different approaches such as hierarchical representations [115], auto-regressive HMMs [94], regressing future representations [207], encoder-decoder LSTMs [67], and inverse reinforcement learning [230]. Some other approaches proposed to perform long-term predictions only focusing on appearance features [137, 49]. However, differently from this work, very little or even no attention has been payed either to the knowledge distillation of the action semantics or label smoothing for action anticipation.

***Knowledge distillation and label smoothing*** Knowledge distillation [87] is the procedure of transferring the information extracted by a teacher network (with high learning capacity) to a student network (with low learning capacity) in order to allow the latter to reach similar performance. This is usually obtained by training the student via a distillation loss which takes into account both the ground truth and the prediction of the pre-trained teacher. Since this procedure can distill useful information to low learning capacity networks, we perform semantic distillation via label smoothing without employing an external network that supervises the process.

Label smoothing, introduced in [193], is the procedure of softening the distribution of the target labels, reducing the most confident value of the one-hot vector and considering a uniform value for all the zero vector components. Although such procedure improves results for classification problems reducing overfitting, no previous works investigate other design approaches except for the uniform smoothing. In our work, we both generalize this idea and show a systematically improvement to state-of-the-art models.

## 2.3 Proposed Approach

Anticipating human actions is essential for developing intelligent systems able to avoid accidents or guide people to correctly perform their actions. Nevertheless, training models on one-hot encoded labels in egocentric videos may led to poor performance when dealing with similar actions. To this end, in the next sections we briefly describe our architecture and then study different label smoothing techniques to address this issue and improve generalization performance.

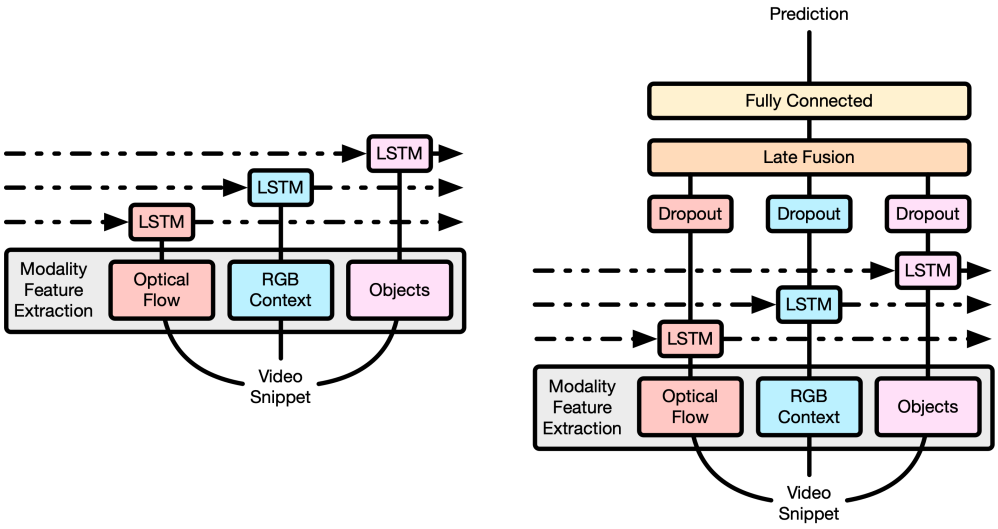



Figure 2.2: Encoding (left) and decoding (right) branches of our architecture. The encoding branch processes, using LSTM networks, video snippets of three modalities (optical flow, RGB and objects). The decoding branch simultaneously processes input video snippets and predicts the next action through a fully connected layer with softmax activation.

### 2.3.1 Action Anticipation Architecture

For our experiments, we consider a learning architecture based on recurrent neural networks (RNNs). We process the input frames preceding the action that we want to anticipate grouping them into video snippet of length 5. Each video snippet is collected every $\alpha = 0.25$ seconds and processed considering three different modalities:

*RGB* features computed using a Batch Normalized Inception CNN [93] trained for action recognition, *objects* features computed using Fast-R CNN [76] and *optical flow* computed as in [213], processed through a Batch Normalized Inception CNN trained for action recognition. Our multi-modal architecture processes the above inputs and encompasses two building blocks: an encoder which recognizes and summarises past observations and a decoder which predicts future actions at different anticipation time steps. Fig. 2.2 depicts the two stages. Firstly, each modality is independently processed by an LSTM layer; then, such streams are merged with late fusion and fed to a fully connected layer to predict the next action.

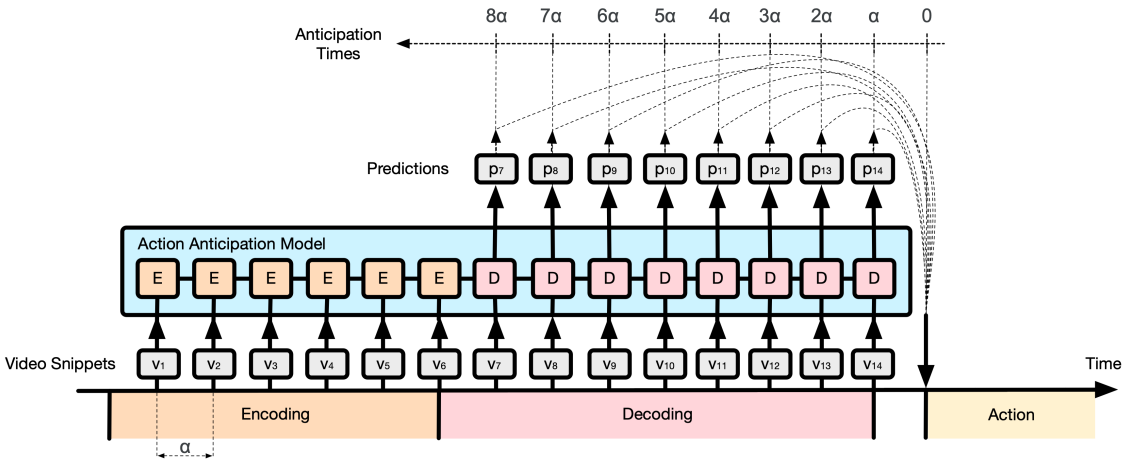



Figure 2.3: Action anticipation protocol based on the encoding and decoding stages. We summarize past observations by processing video snippets sampled every $\alpha = 0.25$ seconds in the encoding stage. After 6 steps, we start making predictions every $\alpha$ seconds for 8 steps to anticipate the future action.

### 2.3.2 Label Smoothing

Egocentric videos exhibit an inherent uncertainty when dealing with predicting future actions [62]. In fact, given the current state observation of an action there can be multiple, but still plausible, future scenarios that can occur. For this reason, the problem can be reformulated as a multi-label task with missing labels where, from a set of valid future realizations, only one is sampled.

All previous models designed for action anticipation are trained with cross-entropy using one-hot labels, leveraging only one of the possible future scenarios as ground truth. A major drawback of using hard labels is to favour logits of correct classes weakening the importance of other plausible ones. In fact, given the one-hot encoded vector $y^{(k)}$ for a class $k$, the prediction of the model $p$ and the logits of the model $z$ such that $p(i) = e^{z(i)} / \sum_j e^{z(j)}$, the cross entropy is minimized only if $z(k) \gg z(i)\ \forall i \neq k$. This fact encourages the model to be over-confident about its predictions since during training it tries to focus all the energy on one single logit leading to overfitting and scarce adaptivity [193]. To this purpose, we smooth the target distribution enabling the chance of negative (yet still plausible) classes to be selected. The most common form of label smoothing procedure relies on a uniform positive component added to a set of similar classes without capturing the difference between such actions. To overcome this issue, we design and compare several label smoothing techniques that exploit both semantic and temporal relations among actions. In the following, we firstly model the general idea of label smoothing and then propose multiple techniques to define the corresponding components.

As a form of regularization, [193] introduces the idea of smoothing hard labels by averaging one-hot encoded targets with constant vectors as follows:

$$y^{soft}(i) = (1 - \alpha)y(i) + \alpha/K, \tag{2.1}$$

where $y(i)$ is the one-hot encoding of the $i^{th}$ example, $\alpha$ is the smoothing factor ($0 \leq \alpha \leq 1$) and $K$ represents the number of classes. Since cross entropy is linear w.r.t. its first argument, it can be written as follows:

$$\begin{aligned} CE[y^{soft}, p] &= \sum_i -y^{soft}(i) \log(p(i)) \\ &= (1 - \alpha)CE[y, p] + \alpha CE[1/K, p]. \end{aligned} \tag{2.2}$$

The optimization based on the above loss can be seen as a distillation knowledge procedure [87] where the teacher randomly predicts the output, i.e., $p^{teacher}(i) = 1/K,\ \forall i$. Hence, the connection with the distillation loss proves that the second

term in Eq. (2.1) can be seen as a prior knowledge, given by an agnostic teacher, for the target $y$. Taking this into account, we extend the idea of smoothing labels by modeling the second term of Eq. (2.1), i.e., the prior knowledge of the targets, as follows:

$$y^{soft}(i) = (1-\alpha)y(i) + \alpha\pi(i), \tag{2.3}$$

where $\pi \in \mathbb{R}^K$ is the prior vector such that $\sum_i \pi(i) = 1$ and $\pi(i) \geq 0 \ \forall i$.

Therefore, the resulting cross entropy with soft labels is written as follows:

$$CE[y^{soft}, p] = (1-\alpha)CE[y, p] + \alpha CE[\pi, p] \tag{2.4}$$

This loss not only penalizes errors related to the correct class but also errors related to the positive entries of the prior. Starting from this formulation, we introduce *Verb-Noun*, *GloVe* and *Temporal* priors for smoothing labels in the knowledge distillation procedure.

***Verb-Noun label smoothing.*** EPIC-KITCHENS [30] contains action labels structured as verbs-noun pairs, like "*cut onion*" or "*dry spoon*". More formally, if we define $\mathcal{A}$ the set of actions, $\mathcal{V}$ the set of verbs, and $\mathcal{N}$ the set of nouns, then an action is represented by a tuple $a = (v, n)$ where $v \in \mathcal{V}$ and $n \in \mathcal{N}$. Let $\mathcal{A}_v(\bar{v})$ the set of actions sharing the same verb $\bar{v}$ and $\mathcal{A}_n(\bar{n})$ the set of actions sharing the same noun $\bar{n}$, defined as follows:

$$\mathcal{A}_v(\bar{v}) = \{(\bar{v}, n) \in \mathcal{A} \ \forall \ n \in \mathcal{N}\}, \tag{2.5}$$

$$\mathcal{A}_n(\bar{n}) = \{(v, \bar{n}) \in \mathcal{A} \ \forall \ v \in \mathcal{V}\}, \tag{2.6}$$

where $\bar{n} \in \mathcal{N}$ and $\bar{v} \in \mathcal{V}$.

We define the prior of the $k^{th}$ ground-truth action class as

$$\pi^{(k)}_{VN}(i) = \mathbb{1}\left[a^{(i)} \in \mathcal{A}_v(v^{(k)}) \cup \mathcal{A}_n(n^{(k)})\right] \frac{1}{C_k}, \tag{2.7}$$

where $a^{(i)}$ is the $i^{th}$ action, $v^{(k)}$ and $n^{(k)}$ are the verb and the noun of the $k^{th}$ action, $\mathbb{1}[\cdot]$ is the indicator function, and $C_k = |\mathcal{A}_v(v^{(k)})| + |\mathcal{A}_n(n^{(k)})| - 1$ is a normalization term. Using such encoding rule, the cross entropy not only penalizes the error related to the correct class but also the errors with respect to all the other *similar* actions with either the same verb or noun[2].

***GloVe based label smoothing.*** An important aspect to consider when dealing with actions represented by verbs and/or nouns is their semantic meaning. In the *Verb-Noun* label smoothing, we define the prior considering a rough yet still meaningful semantic where actions that share either the same verb or noun are considered similar. To extend this idea, we extrapolate the prior from the word embedding of the actions. One of the most important properties of word embeddings is to put closer words with similar semantic meanings and to move dissimilar ones far away, as opposed to hard labels that cannot capture at all similarity between classes

Using such action representation, we enable the distillation of useful information into the model during training since the cross entropy not only penalizes the error related to the correct class but also the error related to all other similar actions. In order to compute the word embeddings of the actions we use the GloVe model [163] pretrained on the Wikipedia 2014 and Gigaword 5 datasets. We use the GloVe model since it does not only rely on local statistics of words, but also incorporates global statistics to obtain word vectors. Since the model takes as input only single words, we encode the action as follows:

$$\phi^{(k)} = Concat\left[GloVe(v^{(k)}), GloVe(n^{(k)})\right] \tag{2.8}$$

where $\phi$ is the obtained action representation of $a^{(k)} = (v^{(k)}, n^{(k)})$ and $GloVe(\cdot) \in \mathbb{R}^{300}$ is the output of the GloVe model. We finally compute the prior probability for smoothing the labels as the similarity between two action representations, which is computed as follows:

[2]It can be proved that in terms of scalar product two different classes having the same noun or verb and encoded with Verb-Noun label smoothing are closer with respect to classes encoded with hard labels

| | Temporal | Uniform | Verb-Noun | GloVe | GLoVe+Verb-Noun |
|---|---|---|---|---|---|
| $\alpha^*$ | 0.6 | 0.1 | 0.45 | 0.6 | 0.5 |

Table 2.1: Results of a grid search procedure to detect the best smooth factor $\alpha$ for proposed label smoothing techniques.

$$\pi_{GL}^{(k)}(i) = \frac{|\phi^{(k)T}\phi^{(i)}|}{\sum_j |\phi^{(k)T}\phi^{(j)}|}. \tag{2.9}$$

Hence, $\pi_{GL}^{(k)}(i)$ in Eq. (2.9) represents the similarity between the $k^{th}$ and the $i^{th}$ action.

***Temporal label smoothing.*** Some pairs of actions are more likely to occur than other ones. Furthermore, only specific action sequences may be considered valid. For this reason, it could be reasonable to focus on most frequent action sequences since they may reveal possible valid paths in the actions space. In this case, we build the prior probability of their observations by considering two subsequent actions in order to estimate the transition probability from the $i^{th}$ to the $k^{th}$ action as follows:

$$\pi_{TE}^{(k)}(i) = \frac{Occ\left[a^{(i)} \rightarrow a^{(k)}\right]}{\sum_j Occ\left[a^{(j)} \rightarrow a^{(k)}\right]} \tag{2.10}$$

where $Occ\left[a^{(i)} \rightarrow a^{(k)}\right]$ is the number of times that the $i^{th}$ action is followed by the $k^{th}$ action. Using such representation, we reward both the correct class and most frequent actions that precede the correct class.

## 2.4 Experiments

### 2.4.1 Dataset, Experimental Protocol and Evaluation Measures

***Dataset.*** Our experiments are performed on EPIC-Kitchens-55 [30] and EGTEA Gaze+ [122] datasets. The former, is a large-scale collection of egocentric videos that contains $39,596$ action annotations divided into 2513 unique actions, 125 verbs, and

352 nouns. We use the same split as [64] producing 23,493 segments for training and 4,979 segments for validation. The latter contains 10,325 action annotations, 19 verbs, 51 nouns and 106 unique actions. In EGTEA Gaze+, methods are evaluated reporting the average performance across the three splits provided by the authors of the dataset [122].

***Experimental Protocol.*** Following [64], our approach uses the protocol depicted in Fig. 2.3 for anticipating future actions based on encoding and decoding stages. Given the action to predict, we consider the previous 14 timesteps spaced out by 0.25 $s$. In the encoding phase, our network processes input video snippets of 5 frames to extract relevant information about the past (6 timesteps). During the decoding phase, it predicts the next action at 8 successive timesteps simultaneously processing the corresponding video snippets. This protocol lets us to measure the performance of the network at different anticipation times.

***Evaluation Measures.*** To asses the quality of predictions and compare all methods, we use the Top-k accuracy, i.e. we assume the prediction correct if the label falls into the best top-k predictions. As reported in [62, 108], such measure is one of the most appropriate given the uncertainty of future predictions. More specifically, we use the Top-5 accuracy for methods comparison. For the test set, we also use the Top-1 accuracy, the Macro Average Class Precision and the Macro Average Class Recall. The last two metrics are computed only on many-shot nouns, verbs and actions as explained in [30].

### 2.4.2 Models and Baselines

In the comparative analysis, we exploit the architecture proposed in Sec. 2.3.1 employing the different label smoothing techniques defined in Sec. 2.3.2. In our experiments we consider models trained using one-hot vectors (One Hot), uniform smoothing (Smooth Uniform), temporal soft labels (Smooth TE), Verb-Noun soft labels (Smooth VN), GloVe based soft labels (Smooth GL) and GloVe + Verb-Noun soft labels (Smooth GL+VN). We design the last method (Smooth GL+VN) by smoothing hard labels with the average of the two related priors.

| Model | Smoothing | Top-5 Acc. (%) @ $\tau_a[s]$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 2 s | 1.75 s | 1.5 s | 1.25 s | 1 s | 0.75 s | 0.5 s | 0.25 s |
| LSTM | - | 27.71 ±0.33 | 28.69 ±0.34 | 29.84 ±0.24 | 30.90 ±0.48 | 31.93 ±0.45 | 33.14 ±0.36 | 34.10 ±0.44 | 35.16 ±0.35 |
| LSTM | TE | 27.94 ±0.24 | 28.90 ±0.27 | 30.06 ±0.24 | 31.13 ±0.19 | 32.19 ±0.28 | 33.21 ±0.36 | 34.17 ±0.37 | 35.10 ±0.25 |
| LSTM | Uniform | 28.16 ±0.27 | 29.06 ±0.26 | 30.23 ±0.24 | 31.25 ±0.27 | 32.41 ±0.28 | 33.64 ±0.27 | 34.69 ±0.19 | 35.75 ±0.13 |
| LSTM | VN | 28.43 ±0.30 | 29.41 ±0.31 | 30.68 ±0.28 | 31.85 ±0.21 | 33.08 ±0.18 | 34.35 ±0.19 | 35.38 ±0.34 | 36.46 ±0.26 |
| LSTM | GL | 28.61 ±0.26 | 29.87 ±0.25 | 30.97 ±0.34 | 31.94 ±0.34 | 33.12 ±0.36 | 34.40 ±0.37 | 35.51 ±0.37 | 36.87 ±0.25 |
| **LSTM** | **GL+VN** | **28.88** ±0.20 | **29.94** ±0.19 | **31.23** ±0.32 | **32.54** ±0.31 | **33.56** ±0.28 | **34.92** ±0.25 | **36.06** ±0.33 | **37.29** ±0.30 |
| **Improv.** | | **+1.17** | **+1.25** | **+1.39** | **+1.64** | **+1.63** | **+1.78** | **+1.96** | **+2.13** |
| RU-LSTM [63] | - | 29.44 | 30.73 | 32.24 | 33.41 | 35.32 | 36.34 | 37.37 | 38.39 |
| **RU-LSTM [63]** | **GL+VN** | **30.37** | **31.64** | **33.17** | **34.86** | **35.90** | **37.07** | **38.96** | **39.74** |
| **Improv.** | | **+0.93** | **+0.91** | **+0.93** | **+1.45** | **+0.58** | **+0.73** | **+1.59** | **+1.35** |

Table 2.2: Top-5 action accuracy for action anticipation task at different anticipation time steps on EPIC-KITCHENS validation set. We report results of several label smoothing techniques and show a systematic performance improvement compared to one-hot encoding. These results are confirmed for a simple LSTM-based model and also for the state-of-the-art RU-LSTM [64] model.

| Top-5 Acc. (%) @ 1 $[s]$ | | | | | | |
|---|---|---|---|---|---|---|
| **Model** | **Modality** | **Baseline** | **TE Smoothing** | **Uniform Smoothing** | **VN Smoothing** | **GL Smoothing** |
| LSTM | RGB | 29.54 | 29.60 | 29.76 | 29.83 | **29.85** |
| LSTM | Flow | 20.43 | 20.53 | 20.65 | 20.69 | **20.77** |
| LSTM | OBJ | 29.5 | 29.64 | 29.69 | 29.79 | **29.82** |
| RU-LSTM [63] | RGB | 30.83 | 30.95 | 31.00 | 31.05 | **31.19** |
| RU-LSTM [63] | Flow | 21.42 | 21.51 | 21.51 | 21.63 | **21.73** |
| RU-LSTM [63] | OBJ | 29.89 | 29.99 | 30.07 | 30.04 | **30.19** |

Table 2.3: Top-5 action accuracy for action anticipation task at $\tau_a = 1$ second on the EPIK-KITCHENS validation set for each model branch (i.e. RGB, Flow and OBJ). We report results of several label smoothing techniques and show a systematic performance improvement compared to one-hot encoded labels.

We also evaluate the effect of the proposed label smoothing techniques on the state-of-the-art approach RU-LSTM [64]. In this case, we choose *GL+VN* as main label smoothing technique.

To implement our architecture, we used TensorFlow 2. Each model is trained for 100 epochs with batch size of 256 using Adam optimizer with learning rate of 0.001. The best model is selected using early stopping by monitoring the Top-5 accuracy at anticipation time $\tau_a = 1$ on the validation set.

***Results.*** For each label smoothing method we select the best smooth factor $\alpha^*$

| Model | Smoothing | Top-5 Acc. % @ $\tau_a[s]$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 2 s | 1.75 s | 1.5 s | 1.25 s | 1 s | 0.75 s | 0.5 s | 0.25 s |
| LSTM | - | 55.94 | 58.75 | 60.94 | 63.02 | 65.78 | 68.04 | 71.55 | 73.94 |
| LSTM | TE | 56.13 | 58.93 | 61.17 | 63.24 | 66.00 | 68.20 | 71.75 | 74.20 |
| LSTM | Uniform | 56.35 | 59.20 | 61.37 | 63.36 | 66.12 | 68.41 | 71.95 | 74.35 |
| LSTM | VN | 56.85 | 59.67 | 61.64 | 64.03 | 66.81 | 68.94 | 72.56 | 75.44 |
| **LSTM** | **GL** | **57.34** | **60.11** | **62.25** | **64.42** | **67.21** | **69.56** | **73.02** | **75.83** |
| **Improv.** | | **+1.4** | **+1.36** | **+1.31** | **+1.4** | **+1.43** | **+1.52** | **+1.47** | **+1.89** |
| RU-LSTM [63] | - | 56.82 | 59.13 | 61.42 | 63.53 | 66.40 | 68.41 | 71.84 | 74.28 |
| **RU-LSTM [63]** | **GL** | **59.99** | **62.02** | **63.95** | **66.47** | **68.74** | **72.16** | **75.21** | **78.11** |
| **Improv.** | | **+3.17** | **+2.89** | **+2.53** | **+2.94** | **+2.34** | **+3.75** | **+3.37** | **+3.83** |

Table 2.4: Top-5 action accuracy for action anticipation task at different anticipation time steps on the EGTEA GAZE+ dataset.

with a grid search procedure between 0 and 1 and step size $\Delta\alpha = 0.05$. The smooth factors used for the results discussed in this section are shown in Table 2.1.

We notice that our label smoothing procedures such as Verb-Noun, GloVe and Temporal perform well with higher smooth factors ($\alpha^* \simeq 0.5$) compared to Uniform label smoothing ($\alpha^* = 0.1$). This suggests that, when soft labels encodes semantic information, prior information becomes more relevant assuming the same importance of the ground-truth due to the similar smooth factors ($\alpha \simeq 1 - \alpha$). During training, we fix $\alpha = \alpha^*$ for each method and, in order to have a robust performance estimation among trials, we iterate ten times all the experiments.

Table 2.2 reports our results on the EPIC-KITCHENS validation set. We notice that all our proposed label smoothing methods improve model performance as compared with training using one-hot encoded labels. Soft labels based on GloVe + Verb-Noun attain best performance improving the Top-5 accuracy from +1.17% to +2.13% compared to hard labels. To validate our results, we trained RU-LSTM [64], the state-of-the-art model for action anticipation, considering label smoothing using EPIC-KITCHENS validation and test sets. Similar results are obtained with the RU-LSTM architecture using both one-hot encoding (our baseline) and smoothed labels. We select *GloVe + Verb-Noun* soft labels since they show a higher performance

increase. As shown by the experimental results (see Table 2.2), label smoothing improves the Top-5 accuracy for all the anticipation times of a margin from +%0.58 to +1.59%. Such behavior highlights the systematic effect of smoothed labels on the model performance. In Table 2.3 we also report the performance of our label smoothing procedures for each model branch. The results show a systematic increase of the accuracy using soft labels for each branch mainly for the GloVe-based method.

Table 2.4 reports our results on the EGTEA Gaze+ test set. All our label smoothing methods improve the baseline trained with one-hot labels. GloVe softening procedure shows the highest accuracy increase ranging from +1.31% to +1.89%. We also report the performance of RU-LSTM baseline trained using GloVe label smoothing and, in this case, for all the anticipation times, the accuracy is improved by a significant margin, from +2.34% to +3.83. Compared to the EPIC-KITCHENS dataset, the larger improvement could be owed to the smaller dataset showing that label smoothing helps bridge the lack of data in the training set.

In Table 2.5, we report the results obtained considering the test set of EPIC-KITCHENS. Different approaches are considered for the comparison. It is wort noting that the method which consider label smoothing together with RU-LSTM improves the performances obtaining best Top-1 and Top-5 accuracy for anticipating *verb*, *noun* and *action*. Label smoothing helps also to improve Precision for *verb* and obtains comparable results in anticipating *action* and *noun*. Results in terms of Recall point out that label smoothing helps for *noun* anticipation maintaining comparable results for the anticipation of *verb* and *action*.

Finally, Fig. 2.4 shows some qualitative results obtained with our framework.

## 2.5 Discussion

This study proposed a knowledge distillation procedure which accounts for multi-modality of future predictions in the action anticipation task. We implemented knowledge distillation through label smoothing by relying on priors capturing inter-dependencies between verb and noun labels, past and future actions, as well as semantic relevance between different actions.

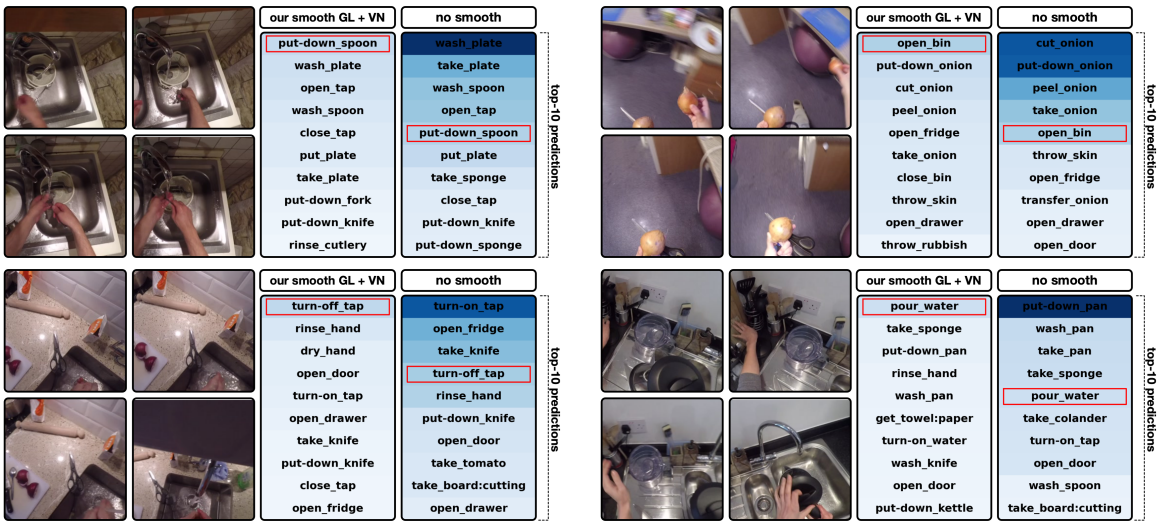



Figure 2.4: Qualitative results of our smoothing labels procedure. We show the comparison between the top-10 predictions of our model at $\tau_a = 1$ second trained either with smoothed labels or with one-hot vectors. These examples are from the validation set where both the models predict correctly, under the top-5 accuracy, the upcoming action. The model trained on one-hot labels is more confident and tries to concentrates all the prediction energy on a very restricted set of actions, without capturing the uncertainty of the future. By contrast, smoothing labels shape the prediction distribution considering similar actions to the ground truth.

Experimental results corroborate out findings compared to state-of-the-art models highlighting that label smoothing systematically improves performance when dealing with future uncertainty.

| | Model | Top-1 Accuracy (%) | | | Top-5 Accuracy (%) | | |
|---|---|---|---|---|---|---|---|
| | | **Verb** | **Noun** | **Action** | **Verb** | **Noun** | **Action** |
| *S1* | DMR [207] | 26.53 | 10.43 | 01.27 | 73.30 | 28.86 | 07.17 |
| | 2SCNN [30] | 29.76 | 15.15 | 04.32 | 76.03 | 38.56 | 15.21 |
| | ATSN [30] | 31.81 | 16.22 | 06.00 | 76.56 | 42.15 | 28.21 |
| | MCE [62] | 27.92 | 16.09 | 10.76 | 73.59 | 39.32 | 25.28 |
| | ED [67] | 29.35 | 16.07 | 08.08 | 74.49 | 38.83 | 18.19 |
| | Miech et al. [141] | 30.74 | 16.47 | 09.74 | 76.21 | 42.72 | 25.44 |
| | RU-LSTM [64] | 33.04 | 22.78 | 14.39 | 79.55 | 50.95 | 33.73 |
| | **RU-LSTM GL+VN** | **35.04** | **23.03** | **14.43** | **79.56** | **52.90** | **34.99** |
| *S2* | DMR [207] | 24.79 | 08.12 | 00.55 | 64.76 | 20.19 | 04.39 |
| | 2SCNN [30] | 25.23 | 09.97 | 02.29 | 68.66 | 27.38 | 09.35 |
| | ATSN [30] | 25.30 | 10.41 | 02.39 | 68.32 | 29.50 | 06.63 |
| | MCE [62] | 21.27 | 09.90 | 05.57 | 63.33 | 25.50 | 15.71 |
| | ED [67] | 22.52 | 07.81 | 02.65 | 62.65 | 21.42 | 07.57 |
| | Miech et al. [141] | 28.37 | 12.43 | 07.24 | 69.96 | 32.20 | 19.29 |
| | RU-LSTM [64] | 27.01 | 15.19 | 08.16 | 69.55 | 34.38 | 21.10 |
| | **RU-LSTM GL+VN** | **29.29** | **15.33** | **08.81** | **70.71** | **36.63** | **21.34** |

| | Model | Avg Class Precision (%) | | | Avg Class Recall (%) | | |
|---|---|---|---|---|---|---|---|
| | | **Verb** | **Noun** | **Action** | **Verb** | **Noun** | **Action** |
| *S1* | DMR [207] | 06.13 | 04.67 | 00.33 | 05.22 | 05.59 | 00.47 |
| | 2SCNN [30] | 13.76 | 17.19 | 02.48 | 07.32 | 10.72 | 01.81 |
| | ATSN [30] | 23.91 | 19.13 | 03.13 | 09.33 | 11.93 | 02.39 |
| | MCE [62] | 23.43 | 17.53 | 06.05 | 14.79 | 11.65 | 05.11 |
| | ED [67] | 18.08 | 16.37 | 05.69 | 13.58 | 14.62 | 04.33 |
| | Miech et al. [141] | 12.42 | 16.67 | 03.67 | 08.80 | 12.66 | 03.85 |
| | RU-LSTM [64] | 25.50 | 24.12 | **07.37** | **15.73** | 19.81 | **07.66** |
| | **RU-LSTM GL+VN** | **30.65** | **24.29** | 06.64 | 14.85 | **20.91** | 07.61 |
| *S2* | DMR [207] | 09.18 | 01.15 | 00.55 | 05.39 | 04.03 | 00.20 |
| | 2SCNN [30] | 16.37 | 06.98 | 00.85 | 05.80 | 06.37 | 01.14 |
| | ATSN [30] | 07.63 | 08.79 | 00.80 | 06.06 | 06.74 | 01.07 |
| | MCE [62] | 10.02 | 06.88 | 01.99 | 07.68 | 06.61 | 02.39 |
| | ED [67] | 07.91 | 05.77 | 01.35 | 06.67 | 05.63 | 01.38 |
| | Miech et al. [141] | 11.62 | 08.36 | 02.20 | 07.80 | 09.94 | 03.36 |
| | RU-LSTM [64] | 13.69 | **09.87** | 03.64 | **09.21** | 11.97 | **04.83** |
| | **RU-LSTM GL+VN** | **14.66** | 09.86 | **04.48** | 08.95 | **12.36** | 04.78 |

Table 2.5: Results of action anticipation on the test sets of EPIC-Kitchens-55. The test set is divided into kitchens already seen $S1$ or unseen $S2$ from the model. The table confirms also on the test set that our smoothing labels procedure can improve performances of the state-of-the-art model RU-LSTM.

# Chapter 3

# Action Anticipation with Slow and Fast Temporal Scales

## 3.1 Introduction

[1]Human action anticipation [28, 60, 208] is a popular research topic in computer vision due to a wide range of involved applications. For example, assistive robotic platforms [109, 176] need to anticipate human movements to correctly perform their tasks when multiple people are present in the same environment. Similarly, advanced video-surveillance systems [125] require to anticipate human motion to promptly provide timely assistance. In this context, egocentric videos have provided a considerable amount of information to be used for training action anticipation models thanks to low-cost wearable devices which offer different streams to be used [143, 186], *e.g.*, RGB videos, audio or depth data.

State-of-the-art approaches [177, 215] are mainly based on attention mechanisms to efficiently extract relationships across subsequent frames at a specific frame rate. Nevertheless, action speed may differ based on the actor, surrounding environment and action itself.

[1]**G. Camporese**, P. Coscia, L. Ballan. "SlowFast Rolling-Unrolling LSTMs for Action Anticipation in Egocentric Videos", *International Conference on Computer Vision, Virtual, 2021 (ICCVW 2021)*.

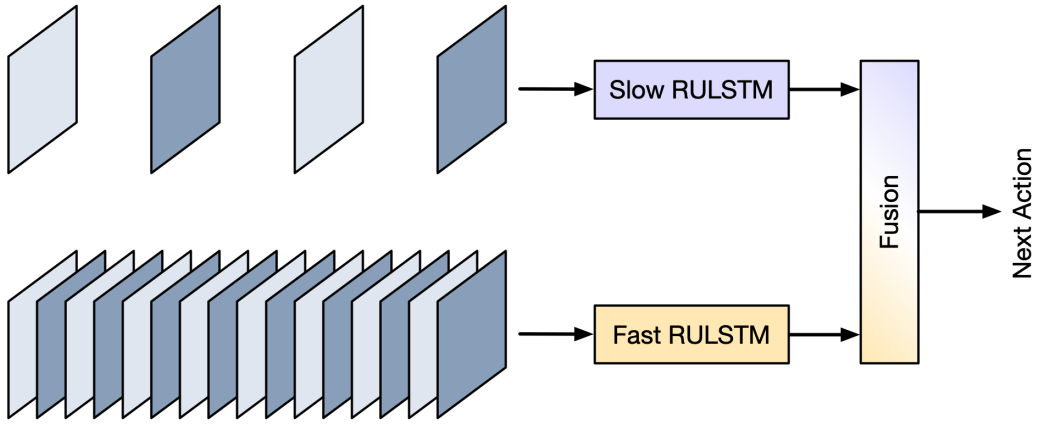



Figure 3.1: Human actions happen at different speeds requiring a multi-scale approach for better predicting future behaviors. We propose a Slow-Fast RU-LSTM model containing two branches, namely *slow* and *fast* branch, which learn independently from input videos features at different time scales.

To anticipate future actions, two main factors should be taken into account: window size (*i.e.*, number of current and past actions to be considered) and processing frame rate (*i.e.*, quantity of information to be extracted from each action). While the former is typically fixed for a fair results comparison, the latter can be arbitrary selected. In this case, a different choice of this parameter may lead to completely different results. We demonstrate that, if multiple streams of the same modality is provided to an action anticipation model, it is able to appropriately select which stream to focus on and improve its predictive capabilities leading to a better generalization.

Based on this idea, we propose to consider multiple branches for each input modality which process the corresponding stream at different frame rates. We focus on two popular egocentric datasets, namely EPIC-Kitchens-55 and EGTEA GAZE+. Based on RU-LSTM [63] model, we propose a slow-fast architecture that learns from input videos at two different scales, as shown in Figure 3.1. A slow branch processes input videos with a small frame rate while another branch uses a higher frame rate.

In this way, redundant information is discarded for actions that evolve slowly while retained for faster actions. In order to efficiently combine these two branches, we use an attentive-based mechanism which efficiently weights their output scores and provides only one result which is subsequently decoded to extract future actions. We show that our model systematically outperforms state-of-the-art models at different anticipation times.

The main contributions of our work can be summarized as follows:

(i) We propose a multi-scale learning technique that benefits from a slow and fast branch to augment performance of RU-LSTM model;

(ii) We perform extensive ablation experiments in order to select the most appropriate frame rates and window sizes;

(iii) We conduct multiple evaluation experiments on popular action anticipation benchmarks and also compare different model architectures and slow-fast fusion mechanisms.

## 3.2 Related Work

### 3.2.1 Action Recognition

Action recognition consists of predicting a labelled action category assigned to an input video. Learning from videos requires capturing both spatial and temporal information, and several approaches have been proposed to solve this task. A simple modelling strategy is based on extracting spatial features from observed video frames with a 2D Convolutional Neural Network (CNN) and their aggregation at temporal level [101, 148], or with Long-Short Term Memory (LSTM) networks [41, 148]. Another popular approach exploits 3D CNNs where spatio-temporal information is gradually fused, leading to a better video representation and more accurate results [20, 200, 195, 198]. Another successful idea uses two-stream networks where

RGB frames and optical flow features are processed providing a more detailed motion information contained in input videos [184, 56, 20, 148].

Recognizing an observed action is the first step for solving more complex tasks, such as early action recognition, where a future action is predicted using only a partial observation of the input video, and action anticipation, where an action category is predicted using only past observed frames.

### 3.2.2 Action Anticipation

Action anticipation requires to predict future actions relying only on past video frames [68]. Previous works proposed different models for activity anticipation in third-person videos [50, 57, 68, 116, 138, 229] and first-person videos [46, 61, 170, 233, 13]. In our work, we adopt the formulation presented in [63], where an action to be predicted is computed at fixed anticipation times before it starts. This is a challenging task since it involves learning both spatial and temporal relationships among past and future frames. To this end, [63] proposes an encoder-decoder LSTM-based architecture where past information is firstly summarized, and future actions are then computed leveraging features extracted from past information.

### 3.2.3 Multi-Scale Modelling in Vision

Multi-scale modelling is a powerful design paradigm that empowers a hidden input representation to be more robust to scale changes with respect to a single-scale modelling approach. This technique can be adopted in both spatial [144, 152] and temporal [52, 177] domains. Slow-Fast networks [52] for video recognition builds upon this idea and show to benefit from processing video sequences at slow and fast frame rates with two separate branches that capture patterns at different time resolutions. In our work, we take advantage of this idea aiming at capturing slow and fast features for anticipating future actions.

## 3.3 Proposed Method

Action anticipation consists of predicting future actions using only visual information extracted from current and past frames. As proposed in [63], a future action is predicted at the anticipation time of 1 $s$ before it occurs, and only video information before this anticipation time can be used for its prediction.

More specifically, the evaluation protocol requires to anticipate the future action at subsequent time steps in order to evaluate the performance of the model approaching the target action.

In the following, we briefly summarize our baseline, which constitutes the backbone used at different time granularities, and then present our slow-fast fusion technique. Finally, we describe how our slow-fast approach can be both used for one input modality and multiple modalities using modality attention [63].

### 3.3.1 Rolling-Unrolling LSTM

Our technique is built upon RU-LSTM [63] model, which processes sequences of feature vectors computed from input video frames. This model defines an encoding stage of $S_{enc}$ steps and an anticipation stage of $S_{ant}$ steps for a total of $\alpha \cdot (S_{enc} + S_{ant})$ seconds, where $\alpha$ is the time interval between two subsequent frames. This model is based on an LSTM-based encoder, named *rolling* LSTM (R-LSTM), and an LSTM-based decoder, named *unrolling* LSTM (U-LSTM). The former summarizes, during the encoding and anticipation stages, past information extracted from input videos and provides to the latter a useful context for predicting the future action. The decoder, in the anticipation stage, receives the representation from the encoder and, using the last observation, computes a plausible distribution over future action classes. The encoding-decoding process is performed for each time step in the anticipation stage, and the network is trained for predicting the actual action label using a cross-entropy loss. To exploit more context and create a more informative hidden representation, RU-LSTM processes multi-modal features which are combined using a mixture-of-experts-based method named Modality Attention (MATT). Since this model shows remarkable performance on predicting future actions from multi-modal input streams,

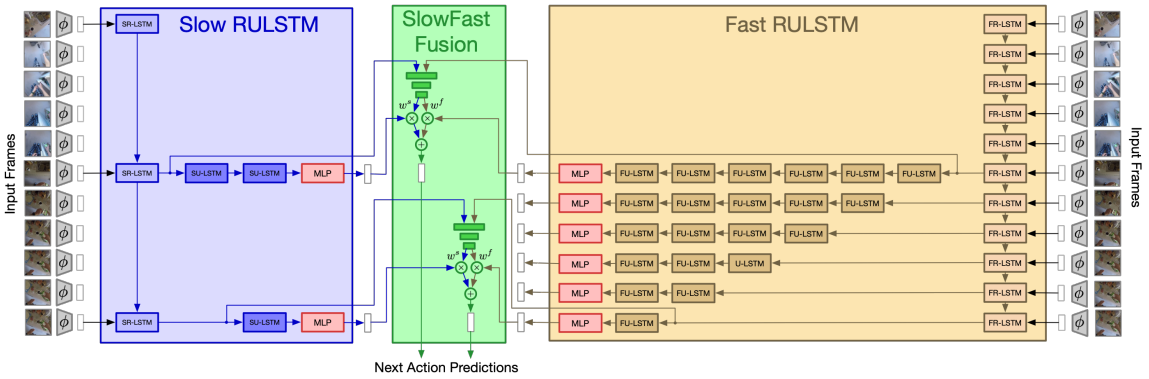



Figure 3.2: Our SlowFast RULSTM model. Input videos are firstly processed by a CNN feature extractor and then sequence representations are fed to two branches processing information at two different frame rates. Our slow and fast branches are based on RU-LSTM architecture that encodes past information and then decodes future actions. To better capture the correlations in past observed frames, we design a slow-fast fusion mechanism that merges the predictions of these two branches leading to a better accuracy.

we extend its predictive capability by explicitly designing a multi-scale fusion mechanism able to capture slow and fast features from observed video sequences.

### 3.3.2 SlowFast RULSTM

As depicted in Figure 3.1, our SlowFast RULSTM model consists of two branches: a slow branch, that processes input videos using a low frame rate (one frame every $\alpha_s$ seconds), and a fast branch, which uses a high frame rate (one frame every $\alpha_f$ seconds). Our idea is to process input features at different time resolutions in order to capture slow and fast relations between past and future frames.

Let $\boldsymbol{x} \in \mathbb{R}^{T \times C \times H \times W}$ be the input video to be processed and $\boldsymbol{z} \in \mathbb{R}^{T \times D}$ the corresponding representation computed at each time step. Given a single input frame $\boldsymbol{x}_t$ at time t, $\boldsymbol{z}_t = \phi(\boldsymbol{x}_t)$ is its related representation where $\phi$ is a CNN feature extractor, and $T = S_{enc} + S_{ant}$ the total sequence length. Our slow branch processes input video frames at $1/\alpha_s$ frame rate while our fast branch at $1/\alpha_f = R/\alpha_s$ with $R = \alpha_s/\alpha_f$ being the ratio between fast and slow frame rates, respectively. Given an internal representation $\boldsymbol{z}_t$, the encoder in the fast branch produces feature representations

used by the decoder as follows:

$$\boldsymbol{r}_t^f = FR\text{-}LSTM\left(\boldsymbol{z}_t, \boldsymbol{r}_{t-1}^f\right) \tag{3.1}$$

where $t \in \{1, 2, \ldots, T\}$ and $\boldsymbol{r}_t^f = (\boldsymbol{h}_t^f, \boldsymbol{c}_t^f)$ is the state that contains hidden and context vectors of FR-LSTM with $\boldsymbol{h}_t^f, \boldsymbol{c}_t^f \in \mathbb{R}^d$. Our slow branch is similarly defined:

$$\boldsymbol{r}_t^s = SR\text{-}LSTM\left(\boldsymbol{z}_t, \boldsymbol{r}_{t-1}^s\right) \tag{3.2}$$

where $t = kR+1$ with $k \in \{0, 1, \ldots, \lfloor T/R \rfloor\}$, and $\boldsymbol{r}_t^s = (\boldsymbol{h}_t^s, \boldsymbol{c}_t^s)$ is the state containing hidden and context vectors of slow R-LSTM with $\boldsymbol{h}_t^s, \boldsymbol{c}_t^s \in \mathbb{R}^d$. The decoder in the fast branch receives the representations given by the fast encoder and produces the prediction by unrolling the Fast U-LSTM for $T - t + 1$ steps as follows:

$$\boldsymbol{u}_{t,q}^f = FU\text{-}LSTM\left(\boldsymbol{z}_t, \boldsymbol{u}_{t,q-1}^f\right), \tag{3.3}$$

$$\boldsymbol{u}_{t,t-1}^f = \boldsymbol{r}_t^f, \qquad \boldsymbol{u}_t^f = \boldsymbol{u}_{t,T}^f, \tag{3.4}$$

where $q \in \{t, \ldots, T\}$. Then, a fast prediction score over all action classes is computed from the output of the decoder with a Multi-Layer Perceptron (MLP) at each time step as $\boldsymbol{l}_t^f = MLP(\boldsymbol{u}_t^f)$, where hidden and context vectors in $\boldsymbol{u}_t^f$ are concatenated. Similarly, the slow decoder receives the slow encoded features $\boldsymbol{r}_t^s$ and produces $\boldsymbol{u}_t^s$ by unrolling the Slow U-LSTM and then slow logits $\boldsymbol{l}_t^s$ are computed with a MLP. The formulation related to the slow decoding step is as follows:

$$\boldsymbol{u}_{t,q}^s = SU\text{-}LSTM\left(\boldsymbol{z}_t, \boldsymbol{u}_{t,q-1}^s\right), \tag{3.5}$$

$$\boldsymbol{u}_{t,t-1}^s = \boldsymbol{r}_t^s, \qquad \boldsymbol{u}_t^s = \boldsymbol{u}_{t,T}^s, \tag{3.6}$$

$$\boldsymbol{l}_t^s = MLP\left(\boldsymbol{u}_t^s\right). \tag{3.7}$$

After slow and fast logits scores computation, our model fuses the obtained predictions with an attention mechanism. Specifically, given both slow and fast scores ($\boldsymbol{l}_t^s$

and $\boldsymbol{l}_t^f$), we compute our final merged logits as $\boldsymbol{l}_t = w_t^s \cdot \boldsymbol{l}_t^s + w_t^f \cdot \boldsymbol{l}_t^f$, where $w_s$ and $w_f$ represents slow and fast multipliers that weight slow and fast predictions computed as follows:

$$\left[\lambda_t^s, \lambda_t^f\right] = \text{MLP}\left(\left[\boldsymbol{r}_t^s, \boldsymbol{r}_t^f\right]\right), \tag{3.8}$$

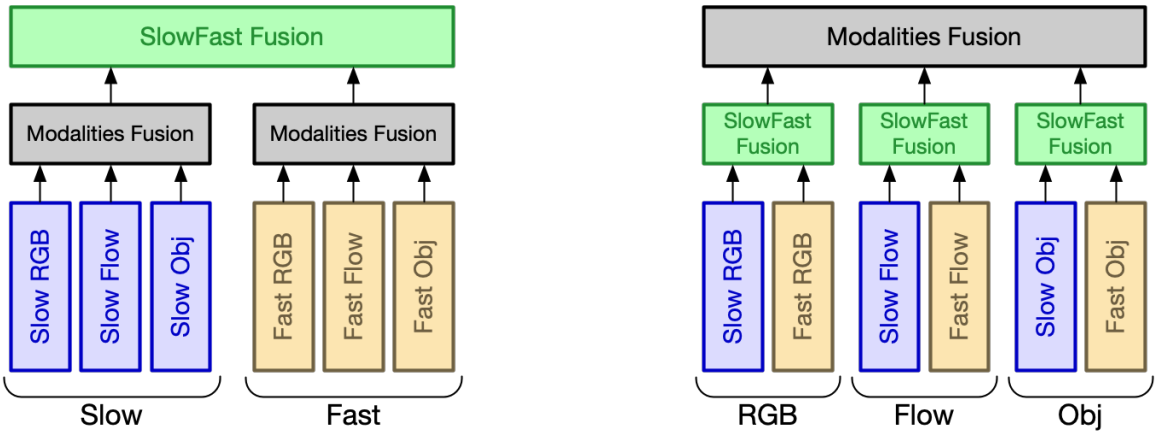
$$w_t^s = \frac{e^{\lambda_t^s}}{e^{\lambda_t^s} + e^{\lambda_t^f}}, \qquad w_t^f = \frac{e^{\lambda_t^f}}{e^{\lambda_t^s} + e^{\lambda_t^f}}, \tag{3.9}$$

where $\left[\,\cdot\,\right]$ stands for the concatenation operator.

### 3.3.3 SlowFast and Modalities Fusion Strategies

As proposed in [63], anticipating future actions can take advantage of multi-modal input representations. For this reason, RU-LSTM proposes an attention mechanism (MATT module) that properly weights each input modality. In our work, we exploit the multi-modal video representation and investigate two different techniques to embed both multi-modal and multi-scale inputs. As shown in Figure 3.3, we could either merge our modalities with a MATT module and then fuse both slow and fast branches (Fig. 3.3a), or firstly fuse slow and fast branches for each modality, and then merge with a MATT module the multi-modal representations (Fig. 3.3b). More specifically, Figure 3.3a depicts an architecture that fuses two RU-LSTMs trained on two different time scales with our slow-fast attention scheme. The input of the attention network is the concatenation of the time scale branches, where each branch is represented by the weighted internal representation $\boldsymbol{r}_t$ of the R-LSTM encoders for all the modalities, using the pre-trained modalities attention weights.

As discussed in Sec 3.3.2, in Figure 3.3b each modality is trained with a slow and fast branch, fused with the slow and fast module and then each modality is merged with the same MATT used in RU-LSTM.




(a) Mod-SF Fusion

(b) SF-Mod Fusion

Figure 3.3: SlowFast and Modalities fusion schemes. (a) Modalities fusion is applied at slow and fast frame rates and then SlowFast fusion is applied to the fused modalities. (b) SlowFast fusion is firstly applied to each modality separately, and then the modalities fusion is applied to the fused time scales.

## 3.4 Experimental Results

We conduct several experiments on two popular datasets used for action anticipation in order to investigate our SlowFast RULSTM model. Furthermore, we study two architectures that embed different fusion mechanisms dealing with multi-modal and multi-scale inputs. In the following, we describe our datasets, metrics and performed experiments in order to show the impact of our slow and fast modelling approach.

**Datasets** We experiment on two popular egocentric datasets: EpicKitchens-55 [29] and EGTEA Gaze+ [123]. EpicKitchens-55 collects 55 hours of recorded videos and $39,596$ annotations of 32 participants involved in their daily kitchen activities. The annotations contain 125 verb and 352 noun classes. All unique (*verb*, *noun*) pairs are considered for a total of $2,513$ unique action labels. EGTEA Gaze+ contains 28 hours of video clips showing hand-object interaction actions performed by 32 participants. It contains 19 verbs, 51 nouns and 106 unique actions. The average across three splits reported by the authors of the dataset is considered.

**Evaluation Metric** For both datasets, we evaluate our proposed SlowFast RULSTM model using Top-5 accuracy metric at different anticipation times.

**Implementation Details** We use PyTorch [160] for our implementation and use pre-extracted features provided by [63] for training our method. We found beneficial to train each branch separately and then fine-tuning at the fusion stages. Specifically, for Mod-SF Fusion approach (see 3.3a), we train RU-LSTM at different frame rates using its standard training pipeline and then fine-tune slow and fast branches at the final stage. For SF-Mod Fusion approach, we apply a similar training strategy.

### 3.4.1 Quantitative Results

**Evaluation Results on EpicKitchens-55** Table 3.1 reports our results for SlowFast RULSTM and RU-LSTM models on EpicKitchens-55 dataset. Our method outperforms RU-LSTM considering both each modality separately and their fusion. The RGB branch shows an improvement of 1.22% at 1 $s$. Additionally, almost 1% of improvement is achieved for both FLOW and OBJ modalities. Our model, combining all modalities, achieves a 36.09% anticipation accuracy at 1 $s$, with an improvement of approximately 0.8% over RU-LSTM baseline. Our model also shows a remarkable gain at 2 $s$ of 1.14% validating our idea to use a multi-scale approach for capturing more information at the early stages of action anticipation. Our results prove that processing egocentric videos at different frame rates improves the prediction accuracy.

Table 3.2 reports a comparison between SlowFast RULSTM and Temporal Aggregation Block (TAB) models, as proposed in [177], which is a current state-of-the-art multi-scale approach for action anticipation. We report results at anticipation accuracy of 1 $s$, as authors do not provide anticipation accuracy at different anticipation times. TAB performance is obtained by using the same configuration reported in [177]. Our results show an accuracy improvement for both RGB and FLOW modalities of +3.8% and +2.76%, respectively. In this case, our improvement for both OBJ modality and complete model is less marked, yet our slow-fast fusion model still outperforms TAB model.

| Modality | Model | Top-5 Acc. (%) @ $\tau_a[s]$ | | | |
|---|---|---|---|---|---|
| | | 2.0 s | 1.5 s | 1.0 s | 0.5 s |
| RGB | RULSTM [63] | 25.44 | 28.32 | 30.83 | 33.31 |
| | **SF-RULSTM (ours)** | **26.78** | **29.25** | **32.05** | **34.34** |
| | **Improv.** | **+1.34** | **+0.93** | **+1.22** | **+1.03** |
| Flow | RULSTM [63] | 17.38 | 18.91 | 21.42 | 23.49 |
| | **SF-RULSTM (ours)** | **18.01** | **19.82** | **22.36** | **24.15** |
| | **Improv.** | **+0.63** | **+0.91** | **+0.94** | **+0.66** |
| Obj | RULSTM [63] | 24.56 | 26.61 | 29.89 | 31.82 |
| | **SF-RULSTM (ours)** | **25.61** | **27.64** | **30.8** | **32.15** |
| | **Improv.** | **+1.05** | **+1.03** | **+0.91** | **+0.33** |
| Fusion | RULSTM [63] | 29.44 | 32.24 | 35.32 | 37.37 |
| | **SF-RULSTM (ours)** | **30.58** | **32.83** | **36.09** | **37.87** |
| | **Improv.** | **+1.14** | **+0.59** | **+0.77** | **+0.5** |

Table 3.1: Top-5 accuracy at different anticipation times for RU-LSTM and our SF-RULSTM model.

| Model | Top-5 Acc. (%) @ 1 s | | | |
|---|---|---|---|---|
| | RGB | Flow | Obj | Fusion |
| TAB [177] | 28.25 | 19.60 | 30.09 | 35.73 |
| **SF-RULSTM (ours)** | **32.05** | **22.36** | **30.8** | **36.09** |
| **Improv.** | **+3.8** | **+2.76** | **+0.71** | **+0.36** |

Table 3.2: Comparison of action anticipation Top-5 accuracy at 1 $s$ between SF-RULSTM and TAB [177] model.

**Evaluation Results on EGTEA Gaze+** Table 3.3 compares our proposed Slow-Fast RULSTM model to RU-LSTM model on EGTEA Gaze+ dataset. For this dataset, only RGB and optical flow features are available, and we train RU-LSTM model to obtain results for both modalities. By contrast, RU-LSTM fusion results are reported from [63]. The table shows a maximum improvement for the FLOW modality of approximately +3.5% at 1 $s$. Furthermore, our complete model improves the anticipation accuracy at 1 $s$ by 1.2%, which can be considered a relevant gain due to the reduced number of classes of this dataset compared to EPIC-Kitchens-55 (106 instead of 2513 classes).

| Modality | Model | Top-5 Acc. (%) @ $\tau_a[s]$ | | | |
|---|---|---|---|---|---|
| | | 2.0 s | 1.5 s | 1.0 s | 0.5 s |
| RGB | RULSTM [63] | 56.41 | 60.68 | 66.76 | 72.04 |
| | **SF-RULSTM (ours)** | **57.84** | **62.36** | **67.21** | **72.32** |
| | **Improv.** | **+1.43** | **+1.68** | **+0.45** | **+0.28** |
| Flow | RULSTM [63] | 33.92 | 35.83 | 39.51 | 42.62 |
| | **SF-RULSTM (ours)** | **36.93** | **39.29** | **42.84** | **45.94** |
| | **Improv.** | **+3.01** | **+3.46** | **+3.33** | **+3.32** |
| Fusion | RULSTM [63] | 56.82 | **61.42** | 66.4 | 71.84 |
| | **SF-RULSTM (ours)** | **57.48** | 61.37 | **67.6** | **72.22** |
| | **Improv.** | **+0.66** | **-0.05** | **+1.2** | **+0.38** |

Table 3.3: Top-5 accuracy at different anticipation times for EGTEA Gaze+ dataset.

### 3.4.2 Ablation Experiments on EPIC-Kitchens-55

To assess the performance of each part of our model, we conduct a set of ablative experiments. In this case, we focus on EpicKitchens-55 dataset. Additionally, all single modality-related experiments use only RGB features, as they can be assumed to be more inclusive features than both optical flow and object-based features.

**Selection of Time Step Value** The main element of our model is represented by the choice of slow and fast time steps to be fused. Table 3.4 illustrates our anticipation accuracy using different time steps ($\alpha \in \{0.1, 0.2, 0.25, 0.5, 1.0\}$) for RGB features. As shown, the best results (at 1 $s$) are obtained selecting $\alpha = 0.125$ $s$ and $\alpha = 0.5$ $s$. For this reason, we use these two values for our fast and slow branches, respectively.

Additionally, Figure 3.4 compares Top-5 accuracy results, for each modality, using three different time steps: 0.125 and 0.5, as obtained by our previous experiments for the RGB modality, and 0.25, which represents the default time step value used in [200]. As shown, our selected time steps improve Top-5 accuracy for each modality.

**Sequence Length Encoding** Extracting relevant features from a video sequence may not only depend on the selected frame rate but also on the length of observed sequences. To this end, we test the impact of different buffer lengths on the anticipation task for the RGB features. Two buffer lengths are considered: $\tau_e = 1.5$ $s$, as

| Time Step [s] | Top-5 Acc. (%) @ $\tau_a[s]$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 2.0 s | 1.75 s | 1.5 s | 1.25 s | 1.0 s | 0.75 s | 0.5 s | 0.25 s |
| $\alpha = 0.1$ | 25.13 | - | 27.26 | - | 30.44 | - | 33.27 | - |
| $\alpha = 0.125$ | 24.53 | 25.63 | 27.3 | **28.97** | **30.96** | **32.23** | **33.49** | **35.02** |
| $\alpha = 0.2$ | 25.16 | - | - | - | 30.71 | - | - | - |
| $\alpha = 0.25$ | 25.2 | **25.84** | 27.78 | 28.84 | 30.55 | 31.92 | 33.19 | 34.43 |
| $\alpha = 0.5$ | **26.39** | - | **28.4** | - | **30.94** | - | 32.87 | - |
| $\alpha = 1.0$ | 25.56 | - | - | - | 30.13 | - | - | - |

Table 3.4: Top-5 action accuracy at different time steps ($\alpha$) for a single modality (RGB). At 1 $s$ the best performance is achieved considering two frame rates: 0.125 and 0.5.

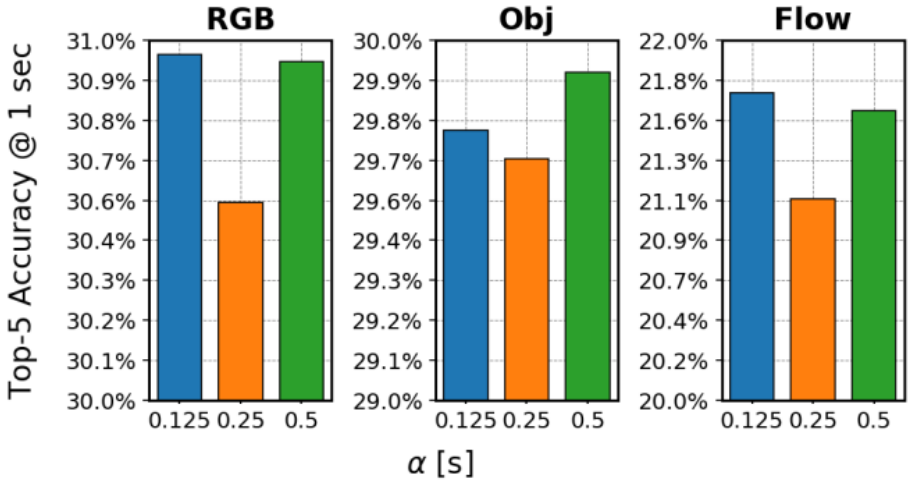



Figure 3.4: Top-5 accuracy varying the time step $\alpha$ for different input modalities. We select $\alpha \in \{0.125, 0.5\}$ for our SlowFast architecture as each branch appears more accurate with respect to selecting $\alpha = 0.25$, as used in [63].

proposed in [63], and $\tau_e = 3.0$ $s$. As shown in Table 3.5, increasing the buffer length provides a noticeable improvement for the slow model ($\alpha = 0.5$ $s$), while the opposite arises for the fast model ($\alpha = 0.125$ $s$). Since the slow model processes a smaller number of video frames, it seems to be able to store more past frames. By contrast, increasing the buffer of the fast model increases its complexity, requiring a smaller window size to achieve better results.

| Top-5 Acc. (%) @ 1 s | | |
|---|---|---|
| $\tau_e$ \ $\alpha$ | 0.125 s | 0.5 s |
| 1.5 s | **30.96** | 30.94 |
| 3.0 s | 30.66 | **31.44** |

Table 3.5: Action anticipation results at 1 $s$ for two different lengths of encoding time ($\tau_e$) for RGB modality.

**SlowFast Fusion** Table 3.6 reports our results for different slow-fast fusion schemes considering the RGB modality. The first three rows shows different fusion methodologies using two scale-branches: slow (with $\alpha = 0.5s$) and fast (with $\alpha = 0.125s$). We consider two additional fusion techniques other than the proposed attention-based fusion:

- *Concat*: prediction obtained directly from the concatenation of the internal representations of the slow and fast branches;
- *Ensemble*: average of the predictions of the slow and fast branches.

As shown, the best fusion scheme at 1 $s$ is represented by an attention-based approach, which appears to better discriminate which branch should be used more for predicting future actions. The last row reports our results for the attention-based model considering an additional scale-branch ($\alpha = 0.25$ $s$). These results confirm that anticipating future human actions requires different time scales for obtaining better performance. Among the proposed models, best results are achieved using two scale branches (slow and fast), while adding another branch does not provide any improvement.

**Modalities Fusion** To assess the performance of the proposed modalities fusion mechanism, shown in Figure 3.3a (Mod-SF Fusion), an alternative fusion architecture (SF-Mod Fusion) is tested (see Figure 3.3b). Table 3.7 provides Top-5 accuracy for both Mod-SF Fusion and SF-Mod Fusion approaches. Additionally, we change the input of the slow-fast attention to be the concatenation of the internal representations

| Fusion Scheme | $\alpha[s]$ | Top-5 Acc. (%) @ $\tau_a[s]$ | | | |
|---|---|---|---|---|---|
| | | 2.0 s | 1.5 s | 1.0 s | 0.5 s |
| Concat | {0.125, 0.5} | 24.59 | 26.9 | 30.04 | 32.73 |
| Ensemble (AVG) | {0.125, 0.5} | 26.98 | 29.59 | 31.71 | 34.2 |
| **Attention** | **{0.125, 0.5}** | 26.78 | 29.25 | **32.05** | 34.34 |
| Attention | {0.125, 0.25, 0.5} | 26.84 | 29.51 | 31.91 | 33.96 |

Table 3.6: Top-5 action accuracy at different anticipation times for different slow-fast fusion schemes (RGB modality).

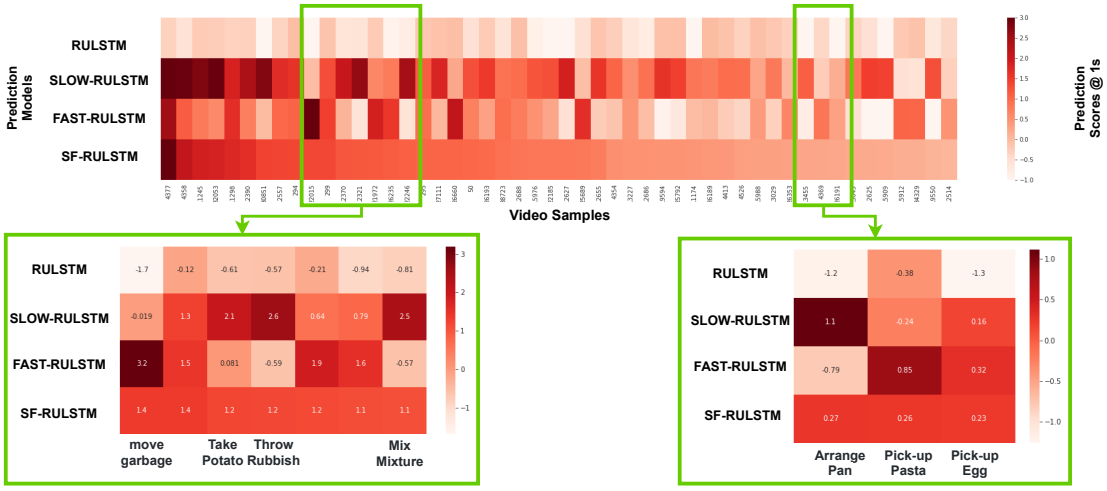



Figure 3.5: Predictions scores of different video samples from our validation set, where our model provides higher prediction scores than RU-LSTM model. For many actions (*e.g.*, *move garbage, arrange pan)* at least one slow/fast branch has a higher prediction score, and so our complete slow-fast model compared to the selected baseline.

of all R-LSTM branches, instead of the weighting mechanism, as discussed in 3.3.3. As shown, Mod-SF Fusion approach appears the best configuration, since it is easier, compared to the other models, to combine different modalities and allow them to aid each other. Using SF-Mod Fusion, the combination of multi-modal predictions is more complex, and reduces model performance. The approach based on the concatenation, provides the lowest accuracy, which can be due to the huge input size to the attention network.

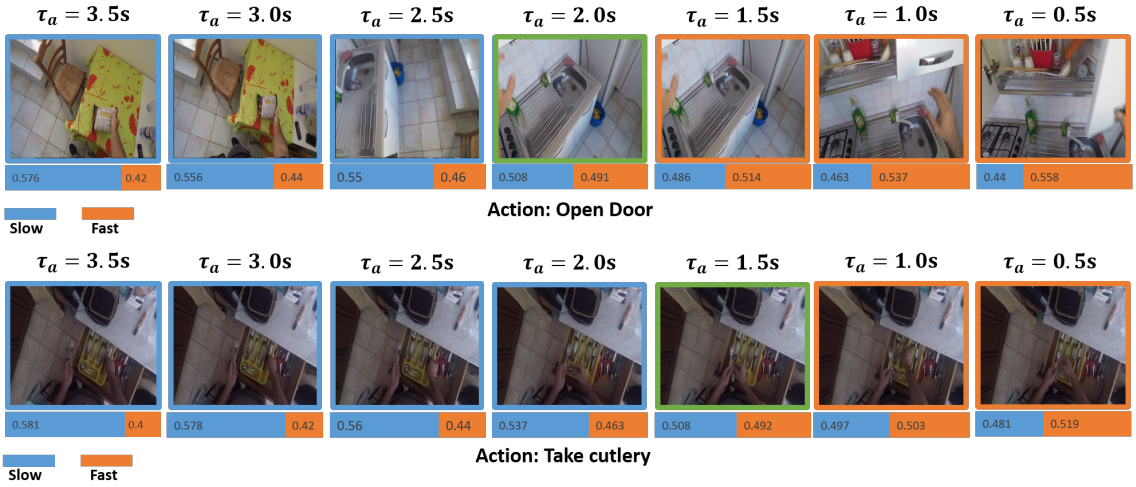



Figure 3.6: Two examples of actions where slow-fast attention weights change over time. The actions start with no significative changes in the input frames, so the attention mechanism weights more the slow branch. When the action rapidly evolve, more attention is instead provided to the fast branch.

| **Fusion Mechanism** | **Top-5 Acc. (%)** |
|---|---|
| Concatenation | 31.92 |
| **Mod-SF Fusion** | **36.09** |
| SF-Mod Fusion | 35.28 |

Table 3.7: Top-5 action accuracy at 1 $s$ for different variations of modalities fusion.

### 3.4.3 Qualitative Results

We qualitatively evaluate the behaviour of our proposed SF-RULSTM in Figure 3.5 and Figure 3.6. Figure 3.5 shows the prediction scores of our SF-RULSTM model (last row) against RU-LSTM model scores (first row) considering a subset of validation samples, *i.e.*, the ones where RU-LSTM assigns low scores. By contrast, our model benefits from either slow (second row) or fast branch (third row) and results in a higher score.

Finally, Figure 3.6 shows how the slow-fast attention model adapts to different action speeds. Our model is able to select the most appropriate branch for the current action speed, *i.e.*, the slow one, when limited changes in the RGB video stream occur,

or the fast branch for actions that evolve more rapidly.

## 3.5 Conclusion

This work proposes a multi-scale attention-based approach to fuse information extracted at different time scales for anticipating human actions in egocentric videos. Two branches process input videos to capture slow and fast features and better discriminate among different actions (or same action performed by different actors). We design several fusion techniques for combining multiple input modalities and demonstrate that an anticipation model can benefit from fusing input modalities before combining different time scales.

We outperform a state-of-the-art model on two popular benchmarks, *e.g.*, EpicKitchens-55 and EGTEA GAZE+ and show better results compared to a multi-scale model on EpicKitchens-55 dataset. Our future work will focus on considering more branches and investigating new techniques to better combine several multi-scale branches.

# Chapter 4

# Early Intent Prediction with Latent Goal Estimation

## 4.1 Introduction

[1]Self-driving cars and autonomous robots have recently demonstrated to be capable of performing multiple tasks (*e.g.*, planning, control, manipulation, perception). Nevertheless, in crowded scenarios, such as streets, parks or airports, they must face critical decisions to avoid accidents and perform a smooth navigation. For example, assistive or delivery robots need to anticipate human motion to better plan their motion and be social compliant. In this regard, urban contexts represent relevant scenarios where predicting human intentions relies on both fast and correct scene perception. Furthermore, predicting future intentions as early as possible helps in better plan their interaction with the environment. Multiple cues are typically involved in this process, *e.g.*, relative speed, pedestrian pose and road signs [169, 146, 110, 127].

In our work, we primary focus on predicting pedestrians intentions to cross roads as early as possible given a fixed history of frames. To this end, we build on top of an action anticipation model an architecture that takes multiple input features and

[1]N. Osman, E. Cancelli, **G. Camporese**, P. Coscia, L. Ballan. "Early Pedestrian Intent Prediction via Features Estimation", *IEEE International Conference on Image Processing, Bordeaux, France, 2022 (ICIP 2022).*

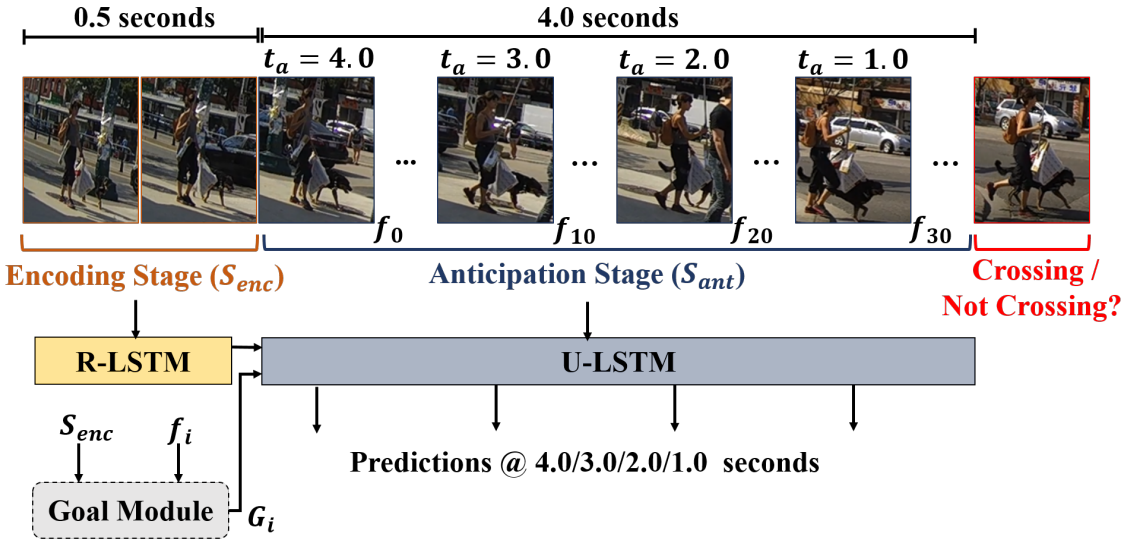



Figure 4.1: Our model detects crossing events in two stages: an encoding stage (R-LSTM), processing the initial part of the sequence (0.5 $s$), and a decoding stage (U-LSTM), predicting the event at multiple anticipation times. We estimate future visual and non visual features, with an attention-based (goal) module, that are provided to the decoding stage. We consider a set of 4 anticipation times: $4.0, 3.0, 2.0$, and 1.0 seconds.

provides the probability that each monitored pedestrian is either crossing or not in the future. Using past motion along with the surrounding context is extremely useful when a pedestrian is walking on a sidewalk prior to crossing or standing due to low visibility in rainy or foggy days, for example. To improve our prediction accuracy, we conceive a *goal* module, whose aim is to predict future features to be fused to motion history (see Fig. 4.1).

Previous works have mainly addressed human action anticipation tasks in both third-person [50, 57, 68, 116, 229] and first-person videos [46, 63, 170, 13, 155]. In this regard, predicting if a pedestrian will cross or not is still a challenging problem. Prior works on intent crossing prediction only focus on a time window of 0.3-0.5 $s$ before the event to extract context features and classify the event [169]. More recent works extend this anticipation time to different values with different observation lengths, in addition to developing more advanced models with multiple input features [146, 110, 127]. In [111], an evaluation benchmark is proposed to tackle this problem, also

adopted in [224, 134, 71, 72]. This benchmark focuses on predicting future intentions between 1.0 to 2.0 $s$ earlier the event and uses overlapping windows of 0.5 $s$ as motion history. An averaging operation is then employed to obtain the final prediction. By contrast, our work aims to extend this anticipation time from 1.0 $s$ to 4.0 $s$ and use an adaptive observation window. To extract predictions at fixed anticipation times, we do not employ an averaging operator yet use pedestrian records as single samples. We mainly focus on two largely adopted benchmarks, namely Joint Attention for Autonomous Driving (JAAD) [169] and Pedestrian Intention Estimation (PIE) [168], which include a large set of annotations.

Our contributions can be summarized as follows: (1) We extend the standard evaluation protocol to predict pedestrian crossing intentions as early as possible. (2) We build on top of a state-of-the-art action anticipation model by introducing a *goal* module to envision future motion in multiple feature spaces. We fuse these features with the motion history using an attention-based mechanism to predict crossing events. (3) We demonstrate, experimenting on multiple benchmarks, that pedestrian intents can be foreseen several seconds in advance, thus improving human safety and social awareness.

## 4.2 Related Work

### 4.2.1 Action Anticipation

With the current increase of artificial intelligence applications that interacts with humans, anticipating future actions of the human actor receives more interest from the research community. The computer vision literature has a wide range of previous works proposing different model architectures to anticipate different types of human activities, in both third-person videos [50, 57, 68, 116, 138, 229] and first-person videos [46, 170, 233, 13, 63].

### 4.2.2 Pedestrian Intent Prediction

One critical sub-problem in action anticipation is pedestrian action prediction. A self-driving car is required to predict the future action _specifically, the crossing action_ of the pedestrians in urban scenarios, based on observed motions of the pedestrians and the visual context of the scene. Early work in crossing prediction observed only 0.3-0.5 seconds before the crossing event to extract context features using CNN and then applying an SVM classifier to predict the crossing action in the proceeding frame [169]. More recent works extended the anticipation time to different values with different observation lengths, in addition to applying more advanced models using different types of features [146, 110, 127]. In [111], an evaluation benchmark is proposed for the crossing prediction problem, is followed by many works, including [224, 134, 71, 72]. This benchmark aims to predict the crossing action in a time range between 1.0 to 2.0 seconds before the event and uses an overlapped 0.5 seconds of observation. In contrast, our work aims to extend the anticipation time to 4.0 seconds, use an adaptive, not fixed, observation length, and dispense the overlapping to have an explicit performance at different anticipation times.

## 4.3 Method

Pedestrian intent prediction relies on past motion to detect whether a pedestrian will cross or not the street, and it can be treated as a binary classification task at different anticipation times. More specifically, let $r$ be a set of RGB frames of a pedestrian starting from $t_0$ to the crossing event at time $T$, and let $T - t_a$ be the anticipation time, where $t_a$ represents the remaining time from the current observation until the event. Our task is to observe $r$ from $t_0$ until $T - t_a$ and predict the crossing/not crossing action at $T$. We aim to predict crossing intentions at different anticipation times, from very early ($t_a \uparrow$) to very close the event ($t_a \downarrow$).

**RU-LSTM.** As backbone, we consider RU-LSTM [63], which processes the observed video frames in two stages: an encoding stage of $S_{enc}$ steps and an anticipation stage of $S_{ant}$ steps, with $\alpha$ as time interval between subsequent time steps.

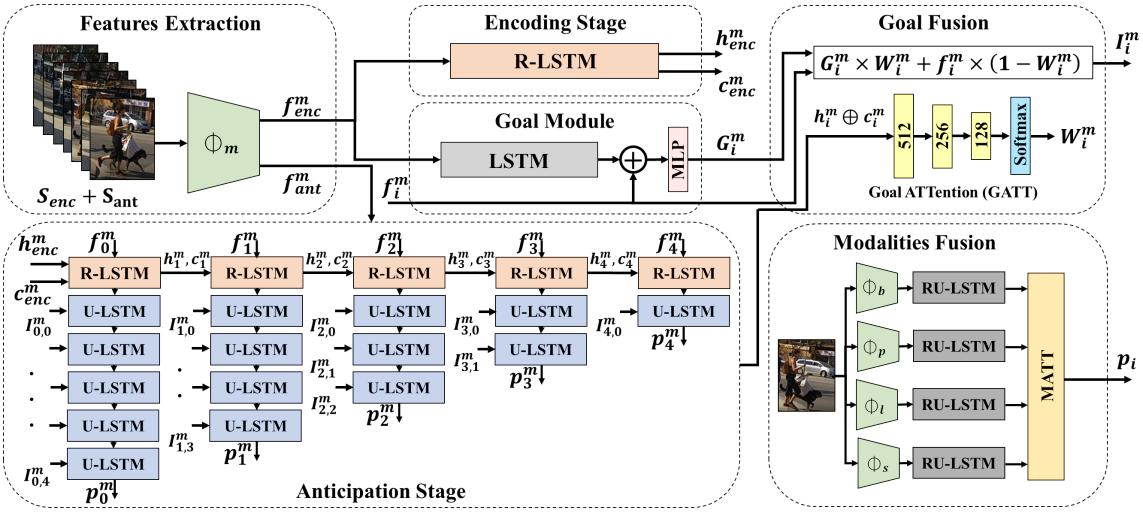



Figure 4.2: Our architecture is composed of six sequential steps: **1) Features Extraction**: where a function $\Phi_m$ extracts different types of features from raw images; **2) Encoding stage**: encodes the motion history with length $S_{enc}$; **3) Goal Module**: takes the input features of the encoding part, along with the features at the $i^{th}$ anticipation step ($f^m_i$) and predicts goal features associated to the modality $m$ ($G^m_i$); **4) Goal Fusion**: fuses predicted goal features $G^m_i$ and extracted frame features $f^m_i$, and produces a new input representation for the next step; **5) Anticipation Stage**: takes $I^m_i$ as input and predicts the crossing probability at each time step ($p^m_i$); **6) Modalities Fusion**: represents our final stage taking predictions provided by each modality and applies an attention-based fusion mechanism to produce the final prediction.

This model uses an LSTM-based encoder-decoder, referred as to *rolling*-LSTM (R-LSTM) and *unrolling*-LSTM (U-LSTM), respectively. During the decoding stage, it simultaneously produces predictions at various anticipation times. RU-LSTM uses multi-modal features, where different modalities are fused using an attention-based mechanism[2](MATT).

**Goal Estimation.** Since RU-LSTM cannot use any information after $T - t_a$ during the evaluation phase, it repeats the observed frame at $T - t_a$ for each step in the unrolling stage. Nevertheless, this model may benefit from having a glimpse into the future. For this reason, we define a *goal* module that predicts features that might

[2]We refer the reader to [63] for a full description of this model.

be extracted at $T$ and fuse this information with the features at $T - t_a$ to enhance its unrolling capability. Let $i$ be the index of the frame at time $T - t_a$; then, for each modality $m$, we use both encoding sequence ($f^m_{enc}$) and frame features ($f^m_i$) to predict future features at $T$, as follows:

$$G^m_i = D^m_{goal}(LSTM^m_{goal}(f^m_{enc}) \oplus f^m_i), \tag{4.1}$$

where $\oplus$ denotes the concatenation operation, $LSTM^m_{goal}$ represents the encoding process related to the goal prediction of modality $m$, and $D^m_{goal}$ is a feed-forward neural network.

**Goal Fusion.** Using an MLP-based network, we define an attention mechanism (see Fig. 4.2) to predict weights to be assigned to *goal* features at each unrolling stage, denoted by $W^m_i$ in Eq. 4.2. $W^m_i$ inherits its length $l_i$ from the corresponding unrolling stage, where $l_i = \frac{t_a}{\alpha}$ is equal to the number of time steps in $t_a$ seconds, with an interval of $\alpha$ seconds; for example, in Fig. 4.2, $l_0 = 5$, $l_1 = 4$, and the length keeps decreasing reaching the event, where $l_4 = 1$ at $t_a = \alpha$ and $t = T - \alpha$. Concretely,

$$W^m_i = Softmax(D^m_{GATT}(h^m_i \oplus c^m_i)) \tag{4.2}$$

where $D^m_{GATT}$ is a feed-forward neural network representing our goal attention mechanism (GATT) which uses hidden ($h^m_i$) and cell ($c^m_i$) vectors of R-LSTM at time $t_a$. Our new input to the U-LSTM is then a weighted average of $G^m_i$ and $f^m_i$, defined as $I^m_i = W^m_i \times G^m_i + (1 - W^m_i) \times f^m_i$.

Finally, predictions from each modality are combined with the modality attention (MATT) mechanism, as proposed in [63], to provide the final prediction.

## 4.4 Experiments

We rely on two popular datasets for our evaluation, namely JAAD and PIE. JAAD dataset contains 2786 pedestrian samples, annotated with bounding boxes and is split into two subsets ($\text{JAAD}_{beh}$ and $\text{JAAD}_{all}$). PIE dataset contains 1842 annotated

| | 4 $s$ | | | 3 $s$ | | | 2 $s$ | | | 1 $s$ | | | [2-1] $s$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acc | AUC | F1 | Acc | AUC | F1 | Acc | AUC | F1 | Acc | AUC | F1 | Acc | AUC | F1 |
| **JAAD$_{beh}$** | | | | | | | | | | | | | | | |
| PCPA [111] | - | - | - | - | - | - | - | - | - | - | - | - | 0.58 | 0.50 | 0.71 |
| PCPA$^{\dagger}$ [111] | 0.54 | 0.51 | 0.62 | 0.47 | 0.46 | 0.54 | 0.49 | 0.45 | 0.61 | 0.45 | 0.52 | 0.63 | 0.56 | 0.50 | 0.67 |
| R-LSTM [63] | 0.67 | 0.64 | 0.74 | 0.70 | 0.64 | 0.78 | 0.66 | **0.62** | 0.75 | 0.65 | 0.60 | 0.75 | 0.65 | 0.59 | 0.74 |
| RU-LSTM [63] | **0.72** | **0.67** | **0.79** | **0.72** | **0.64** | **0.81** | **0.69** | **0.62** | **0.78** | **0.70** | **0.63** | **0.79** | **0.69** | **0.62** | **0.78** |
| **JAAD$_{all}$** | | | | | | | | | | | | | | | |
| PCPA [111] | - | - | - | - | - | - | - | - | - | - | - | - | 0.85 | **0.86** | **0.68** |
| PCPA$^{\dagger}$ [111] | 0.75 | 0.75 | 0.50 | 0.74 | 0.76 | 0.53 | 0.72 | 0.75 | 0.52 | 0.76 | **0.79** | 0.55 | 0.80 | 0.79 | 0.57 |
| R-LSTM [63] | 0.83 | 0.74 | 0.54 | 0.86 | 0.73 | 0.58 | **0.85** | 0.73 | 0.57 | **0.87** | 0.77 | **0.62** | **0.86** | 0.76 | 0.60 |
| RU-LSTM [63] | **0.84** | **0.76** | **0.57** | **0.87** | **0.78** | **0.64** | **0.85** | **0.76** | **0.59** | 0.86 | 0.78 | **0.62** | **0.86** | 0.78 | 0.62 |
| **PIE** | | | | | | | | | | | | | | | |
| PCPA [111] | - | - | - | - | - | - | - | - | - | - | - | - | **0.87** | **0.86** | **0.77** |
| PCPA$^{\dagger}$ [111] | 0.76 | 0.75 | 0.62 | 0.77 | 0.76 | 0.63 | 0.83 | **0.84** | 0.73 | 0.86 | **0.85** | 0.77 | 0.86 | **0.86** | **0.77** |
| R-LSTM [63] | 0.75 | 0.64 | 0.48 | 0.75 | 0.66 | 0.50 | 0.76 | 0.66 | 0.51 | 0.76 | 0.67 | 0.52 | 0.76 | 0.67 | 0.52 |
| RU-LSTM [63] | **0.77** | **0.76** | **0.63** | **0.80** | **0.79** | **0.68** | **0.85** | 0.82 | **0.74** | **0.88** | **0.85** | **0.79** | **0.87** | 0.84 | **0.77** |

Table 4.1: Comparison among mutiple intent prediction models for both JAAD and PIE datasets. We consider four anticipation times and the standard evaluation protocol averaging predictions within the range [2-1] $s$. PCPA$^{\dagger}$ [111] denotes our retrained PCPA model (using the original code and configurations provided in the official GitHub repository).

pedestrian samples with bounding boxes and behavioral tags, in addition to speed, GPS coordinates, and heading direction of the ego-vehicle.

**Features.** Three main modalities for both datasets are employed: bounding box coordinates, pose features and visual local context features. The PIE dataset also includes ego-vehicle speed as an additional modality.

**Baselines.** To evaluate the performance of the proposed method for early predictions, we develop an LSTM-based model using RU-LSTM without the unrolling stage (R-LSTM). Furthermore, we compare our model, at different anticipation times, against PCPA [111] which uses fixed and overlapped observations of 0.5 $s$ in the range $[2-1]$ s earlier the event. For our purposes, this model is modified to output earlier predictions in the range $[4-1]$ s. We also consider CAPformer [134] and TrouSPI-Net [72] models.

**Evaluation Metrics.** As classification metrics we use accuracy, AUC and F1 score.

**Implementation Details.** Our LSTMs have 512 hidden layers, with $S_{enc} = 6$

| JAAD$_{beh}$ | | | | | |
|---|---|---|---|---|---|
| | 4 $s$ | 3 $s$ | 2 $s$ | 1 $s$ | [2-1] $s$ |
| **Bounding Box** | | | | | |
| Without Goal | 0.55 | 0.59 | 0.63 | <u>0.64</u> | 0.58 |
| Interpolation | **0.61** | 0.56 | 0.61 | **0.65** | **0.64** |
| Concatenation | **0.61** | **0.64** | <u>0.63</u> | 0.63 | <u>0.63</u> |
| Attention | **0.61** | <u>0.63</u> | **0.65** | **0.65** | **0.64** |
| **Pose** | | | | | |
| Without Goal | 0.51 | 0.60 | <u>0.65</u> | <u>0.65</u> | <u>0.65</u> |
| Interpolation | 0.59 | <u>0.64</u> | 0.63 | 0.64 | 0.63 |
| Concatenation | <u>0.60</u> | **0.67** | **0.66** | **0.66** | **0.66** |
| Attention | **0.62** | **0.67** | 0.57 | <u>0.65</u> | <u>0.65</u> |
| **Local Context** | | | | | |
| Without Goal | 0.68 | 0.66 | 0.65 | 0.67 | <u>0.66</u> |
| Interpolation | 0.73 | <u>0.67</u> | 0.63 | 0.66 | 0.61 |
| Concatenation | 0.65 | **0.70** | **0.68** | <u>0.68</u> | **0.68** |
| Attention | **0.69** | <u>0.67</u> | <u>0.66</u> | **0.69** | <u>0.66</u> |

Table 4.2: Ablation study reporting the accuracy metric using different fusion methods for each modality on JAAD$_{beh}$. Underlined numbers refer to the $2^{nd}$ best performing model.

and $S_{ant} = 40$. The time interval is 3 frames ($\alpha = 0.1$ $s$), while our models output 40 predictions from $t_a = 4$ $s$ until $t_a = 0.1$ $s$. For the sake of clarity, we consider only predictions at $t_a \in \{4, 3, 2, 1\}$ $s$. For each anticipation time ($t_a$), we discard videos that do not satisfy the minimum length requirement, *i.e.*, videos whose length is less than $t_a$. Therefore, we consider 4 $s$ as the earliest anticipation time, where longer predictions would lead to discarding too many examples (more than 50% of data from the JAAD dataset). For a fair comparison, in the range [2-1] $s$, we use the same evaluation protocol proposed in [111], averaging the predictions with a step of 0.1 $s$ for JAAD and 0.2 $s$ for PIE.

**Results.** Firstly, we evaluate our models on different anticipation times and, eventually, measure the impact of the *goal* module. Table 4.1 reports our results for different anticipation times (from 4 $s$ to 1 $s$). Depending on the dataset, two main trends emerge when approaching the event. On JAAD$_{beh}$ dataset, all metrics remain stable approaching the event confirming our intuition to forecast features in

| Model | $\mathbf{JAAD}_{beh}$ | | | $\mathbf{JAAD}_{all}$ | | |
|---|---|---|---|---|---|---|
| | Acc | AUC | F1 | Acc | AUC | F1 |
| PCPA [111] | 0.58 | 0.50 | 0.71 | 0.85 | 0.86 | 0.68 |
| CAPformer [134] | - | 0.55 | 0.76 | - | 0.82 | 0.63 |
| TrouSPI-Net [72] | 0.64 | 0.56 | 0.76 | 0.82 | 0.77 | 0.58 |
| RU-LSTM [63] | 0.69 | 0.62 | 0.78 | **0.86** | 0.78 | 0.62 |
| G-RULSTM (ours) | **0.72** | **0.65** | **0.80** | **0.86** | **0.80** | **0.63** |
| **Improv.** | **3%** | **3%** | **2%** | - | **2%** | **1%** |

Table 4.3: Comparison between our architectures and state-of-the-art models within the $[2-1]$ $s$ range on JAAD dataset.

advance. This behavior may also be related to the limited samples of this dataset which is quite unbalanced yet on $\text{JAAD}_{all}$ this trend is confirmed since no remarkable drops in performance can be observed. Among the used models, RU-LSTM performs the best in both cases. By contrast, on PIE dataset our metrics increase when the anticipation time gets close to the event. It is worth noting that RU-LSTM systematically improves performance metrics compared to the other considered models for the different anticipation times. In the $[2-1]$ $s$ range, RU-LSTM outperforms the considered models on $\text{JAAD}_{beh}$ dataset, while is on par with PCPA [111] on $\text{JAAD}_{all}$ and PIE datasets. It is also worth mentioning that the standard protocol used in [111] increases the number of training samples considering overlapping windows. By contrast, our proposed protocol dumps this overlapping technique leading to a more realistic evaluation of this task yet largely reducing the number of training samples. This could also explain the limited performance of RU-LSTM in the $[2-1]$ $s$ range, where PCPA uses overlapping windows, while it outperforms at fixed anticipation times all the models using the same number of training samples.

Table 4.2 reports an ablation study considering three different fusion techniques for estimating future features: **1) Interpolation**, where $I_i^m$ is obtained as a linear combination of $f_i^m$ and $G_i^m$, based on the time distance $d_j$ from the current step to the frame corresponding to the event, *i.e.*, $I_i^m = \frac{(1-d_j)}{l_i} \times G_i^m + \frac{d_j}{l_i} \times f_i^m$;

**2) Concatenation** followed by an MLP layer; **3) Attention**, as defined in Sec. 4.3. We observe that our goal module, with any considered fusion technique, improves prediction metrics over the baseline, for all modalities and anticipation

times. Furthermore, depending on the considered modality, a different fusion techniques could be considered. For example, for bounding box coordinates, representing pedestrian's motion within the image, a linear relationship over time is reported. By contrast, both pose and local context features show a different trend. In this case, concatenation and attention perform better. We select the attention-based technique for all modalities to adapt to both their linear and non-linear relationships.

In Table 4.3, we compare our model to state-of-the-art architectures. RU-LSTM does not contain the goal module, while G-RULSM uses future features estimation. We observe that G-RULSTM outperforms state-the-art-models for $\text{JAAD}_{beh}$, in addition to a noticeable improvement over RU-LSTM. For $\text{JAAD}_{all}$, goal-boosted G-RULSTM outperforms our baseline for AUC and F1 metrics, which are more robust metrics for unbalanced datasets. Our proposed model suffers from a hard reduction of training samples, compared to [111] which limits its performance on larger datasets, *e.g.*, $\text{JAAD}_{all}$. Nevertheless, our model is on par with CAPformer [134] and outperforms TrouSPI-Net [72] for both subsets.

## 4.5 Conclusion

In this work, we revise the standard evaluation protocol used to measure the performance of pedestrian intent prediction models. We demonstrate that crossing events can be predicted several seconds in advance with no (or negligible) impact on the performance. To validate our intuition, we build upon an action anticipation model a *goal* module to forecast future features and improve its prediction metrics. This information can increase prediction accuracy up to 3% compared to models that do not envision future features.

# Chapter 5

# Early Action Recognition with Action Prototypes

## 5.1 Introduction

[1]Our brain has the innate capability of anticipating events in our daily lives. One of the fundamental ideas of the Predictive Coding (PC) theory [142] in neuroscience is that the brain is constantly engaged in predicting its upcoming states and recent works show that the physical neurons may involve the prediction of its future activity in their overall learning dynamics [136, 59]. Nowadays, deep learning vision systems have achieved strong learning capabilities on understanding the content of videos especially in the tasks of recognition, action localization, summarization, and representation learning. However, the understanding and the investigation of models designed for predicting the future video content have been partially covered, and only recently gained more attention [65, 217]. Action anticipation and early action recognition on videos are computer vision problems that are important for various scenarios such as pedestrian anticipation in autonomous driving, early recognition of activities in video surveillance, and anticipation of possible dangerous situations in ego-centric

[1]**G. Camporese**, X. Lin, A. Bergamo, J. Tighe, D. Modolo. "Early Action Recognition with Action Prototypes" *Proc. of IEEE/CVF Computer Vision and Pattern Recognition (CVPR 2023)* [Under submission]. Work done while interning at AWS AI Labs.

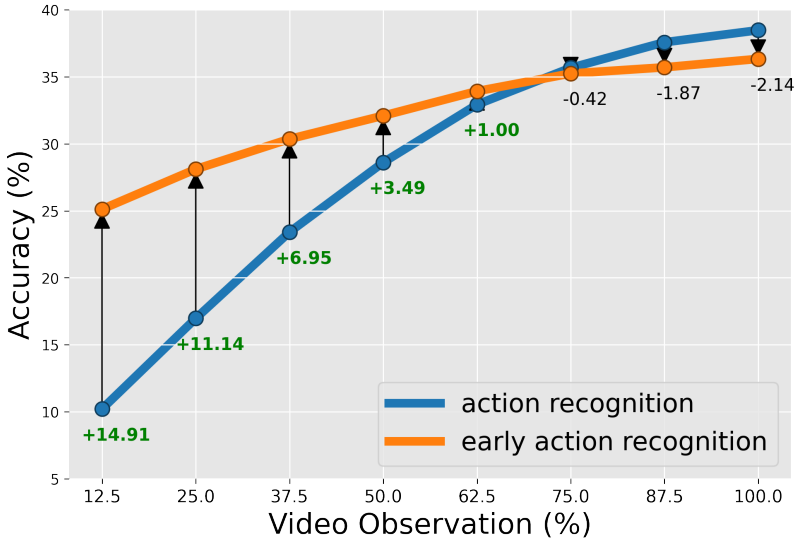



Figure 5.1: Standard action recognition models lack in accuracy when the action is partially observed, whereas our model designed for early action recognition highly improves the results at early predictions, slightly reducing the performances when the video is fully observed.

cameras worn by workers. Specifically, in the early action recognition problem the model predicts the action contained in the entire video by looking only at the initial portion of it. This task is challenging and harder than action recognition since the model not only has to learn a good representation of the video, but also has to relate current encoded representation with possible upcoming scenarios dealing with the uncertainty of the future. Our work focuses on modeling the future representation as a set of possible scenarios one for each possible action class and to use these future representations during the learning process as an additional predictive task in parallel to the early action recognition objective. We can summarize the main contributions of this work in the following:

- we propose a simple yet effective method for early action recognition by learning action prototypes and by using them as a source of regularization for the model during the training process. The model is trained end-to-end and the prototypes are both learned and used as regularizers during the training phase through the stop-gradient operator;

- we propose a novel dynamic loss function that involves the transition between two different temporal losses during the training process. With this new loss we can decide how to balance the trade-off between the accuracy at early and late predictions, and we can improve the overall AUC metric of the model;

- we conducted several experiments on several video datasets proving that our method is effective, obtaining state-of-the-art results on all the benchmarks.

## 5.2 Related Work

**Video Understanding Modeling.** Over the years, a lot of effort have been spent on the modeling the spatio-temporal representations of videos. Approaches that are worth to mention are hand-crafted video representations [192, 39, 45, 105, 118, 162], recurrent neural networks based architectures [40, 97, 121, 124, 149], 2D CNN [211, 212, 218], and 3D CNN [19, 51, 53, 74, 124, 165]. Recently, transformers have been investigated for spatio-temporal modeling [47, 4, 74] improving the state-of-the art on many video related tasks.

**Vision Transformers.** In last years, the transformer [204] has been proposed as a general architecture for dealing with language problems and proved to achieve incredible performances [204, 37, 12] surpassing previous models based on recurrent and convolutional neural networks. Not only for their proven possibilities, the transformer gains a lot of attention also for its general architecture that can adapted to a variegate number of problems, since it can process input tokens of different nature. Recently, transformers have been successfully adapted for vision tasks such as image classification [43, 130, 196], video recognition [47, 4, 74], and object detection [16, 27, 133, 238, 23]. The Vision Transformer [196] (ViT) architecture follows a standard transformer encoder design, whereas more specialized versions [130, 47] add more inductive bias in the architecture exploiting hierarchical and multi-scale processing schemes.

**Action Recognition.** The general objective of the action recognition problem is to predict the activity that is contained in the input videos. Usually, the datasets involved in this task are human centric, meaning that the action labels that has to be predicted are related to activities performed by humans [102, 77, 187, 31, 32].

**Early Action Recognition and Anticipation.** In the early action recognition problem the model has to recognize the action observing only the initial crop of the video. It is similar to the action recognition problem, but the information of the spatio-temporal pattern to be captured by the model can be corrupted (partially observable), or missing depending on the percentage of observation allowed. Previous works on early recognition investigated several approaches [216, 236, 66, 1]. In the action anticipation task the model can observe only the part of the video related to the past, before the action occurs. Previous approaches are RU-LSTM [65], Temporal Aggregated Blocks [178], Anticipative Vision Transformers (AVT) [75], Memory Agumented Multi-Scale Vision Transofmers (MeMViT) [217], and others [156, 14].

**Prototype Learning.** Prototypical networks [185] learn to estimate the average representations of the classes of a supervised dataset with centroids called protoypes. Their application is mostly on few-shot learning where the prototypes are used as references for predicting new inputs of unseen classes, and also for representing fine-grained spatio-temporal patterns [140]. After training, prototypes provide a representation of all the classes of the dataset.

## 5.3 Our Model

**Early Recognition Setting.** The early recognition problem aims at recognizing actions starting from partially observed videos. Following previous works [1, 66, 236, 216, 188], we define the partial observation ratio $\rho \in (0, 1]$ as the percentage of video that the model observes. More specifically, we partition a raw video of $F$ number of frames into $T$ non-overlapping clips, each having $F_c$ frames, obtaining the observation ratio defined as $\rho = t \cdot F_c/F$ with $t \in \{1, \dots, T\}$.

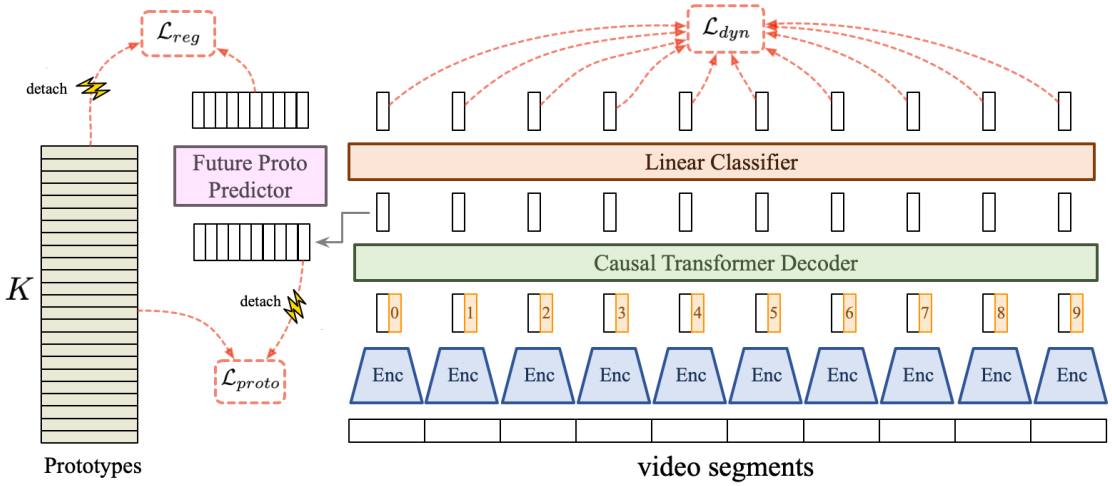



Figure 5.2: Overall architecture of our proposed model.

| Module | Instance | Size | Depth | D | In Shape | Out Shape |
|---|---|---|---|---|---|---|
| Encoder | MViT [47] | Base | 16 | 768 | $[C, F_{seg}, H, W]$ | $[D]$ |
| Decoder | T-Dec [204] | Base | 6 | 768 | $[N_{seg}, D]$ | $[N_{seg}, D]$ |

Table 5.1: Modules details.

**Visual Encoder.** The visual backbone processes one clip at a time. Specifically, given the $t$-th video clip $\mathbf{x}(t) \in \mathbb{R}^{C \times F_c \times H \times W}$ with $F_c$ number of frames, the visual encoder produces a $D_{enc}$ dimensional feature representation that is subsequently projected with a linear layer into $\mathbf{z_{enc}}(t) \in \mathbb{R}^D$ in order to be compatible with the decoder. Our encoder design is general and does not require a specific architecture, however, in our investigation we adopted the recent MViT [47] since it is an efficient and strong transformer based model.

**Temporal Aggregator Decoder.** The decoder takes as input the output sequence of the encoder $\mathbf{z_{enc}} \in \mathbb{R}^{T \times D}$, it adds the learnable positional embeddigs, and it produces the decoder features $\mathbf{z} \in \mathbb{R}^{T \times D}$. In our investigation we use a Transformer Decoder [204], and to avoid the processing of future tokens during the computation, the decoder adopts a masked multi-head attention with causal masks. Subsequently, the final action prediction $\hat{y}$ is computed by a linear head $h(\cdot)$ that processes each

token representation of the decoder independently as $\hat{y}(t) = h(\mathbf{z}(t))$, optimizing the classification objective:

$$\mathcal{L}_{clf}(\mathbf{x}, y, t) = -\sum_{k=1}^{K} y(k) \log \hat{y}(t, k), \tag{5.1}$$

where $K$ is the number of action classes of the dataset.

**Learning Action Representations with Prototypes.** The prototypes are learnable embeddings $\mathbf{P} \in \mathbb{R}^{K \times D}$, that are learned during the training process, and their goal is to represent globally the action classes, one prototype for each class. In order to train the prototypes, we use the embeddings of the last segment $\mathbf{z}(T) \in \mathbb{R}^{D}$ extracted from the decoder as they encode the whole spatio-temporal discriminative information of the action. Specifically, the representation $\mathbf{z}(T)$ is normalized and compared with all the $K$ prototypes under the $\ell_2$ similarity as:

$$s(k) = -\left\| \mathbf{P}(k) - sg\big(\mathbf{z}(T)\big) \right\|, \tag{5.2}$$

where $s(k) \in \mathbb{R}$ and $P(k) \in \mathbb{R}^{D}$ are respectively the similarity score and the prototype of the $k$-th class, and $sg(\cdot)$ is the stop-gradient operator. The similarity scores are softmax-activated and converted to a class distribution that can be trained with the cross-entropy loss. In particular, we define the objective for learning the prototypes as:

$$\mathcal{L}_{proto}(\mathbf{x}, y) = -\sum_{k=1}^{K} y(k) \log \left( \frac{e^{s(k)}}{\sum_{j=1}^{K} e^{s(j)}} \right) \tag{5.3}$$

where the similarity distribution is encouraged to be similar to the one-hot representation of the action class. Specifically, with this loss we maximize the similarity between the decoder representation $\mathbf{z}(T)$ and the prototype of the ground truth label $\mathbf{P}(y)$, and at the same time we minimize the similarity between the feature representation $\mathbf{z}(T)$ and all the other prototypes. This constrastive objective is implicitly defined in the cross-entropy loss. In order to learn the prototypes, we detach the target signal $\mathbf{z}(T)$ on the computation of $\mathcal{L}_{proto}$ with the stop-gradient operation for

separating the prototype objective from the other ones and not experiencing collapses issues.

**Regularization Predicting Future Prototypes.** One way to mitigate the model training using the prototypes information is to regularize the architecture by constraining the output feature space of the decoder to the regions where it is possible to predict the action prototypes. More specifically, we defined a feature predictive module $f(\cdot)$ that tries to predict the correct prototype from decoder' embeddings $\mathbf{z}(t)$ at all the partial observation steps $t$. In particular, we compute the $\ell_2$ similarity $s_{reg}$ between the predicted feature $f(\mathbf{z}(t))$ and all the prototypes $\mathbf{P}$ as:

$$s_{reg}(t,k) = -\left\| sg\big(\mathbf{P}(k)\big) - f(\mathbf{z}(t)) \right\| \tag{5.4}$$

and we optimize the following objective:

$$\mathcal{L}_{reg}(\mathbf{x}, y, t) = -\sum_{k=1}^{K} y(k) \log \left( \frac{e^{s_{reg}(t,k)}}{\sum_{j=1}^{K} e^{s_{reg}(t,j)}} \right) \tag{5.5}$$

The feature predictive module is implemented as a multi-layer perceptron that processes the decoder' features at each partial observation independently. In order to train the feature predictive module we detach the prototype target signal $P$ with the stop-gradient operator for separating the regularization objective from the other ones and not experiencing collapses issues as done in the prototypes objective.

```
"""
encoder: model visual encoder
proj: encoder linear projector
decoder: model transformer decoder
head: classification head
xent_dyn: dynamic cross-entropy loss
P: prototypes [K, D]
"""
from torch import cdist

def train_step(x, y):

  # classification
  z_enc = proj(encoder(x)) # [B, T, D]
  z = decoder(z_e) # [B, T, D]
```

```
logits = head(z) # [B, K, T]
preds = logits.softmax(1) # [B, K, T]
loss_clf = xent_dyn(logits, y)

# prototypes learning
sim_P = - cdist(P, z[:, -1].detach()) # [B, K]
loss_proto = xent(sim_P, y)

# prototypes prediction
z_feat = f(z) # [B, T, D]
sim_feat = - cdist(z_feat, P.detach()) # [B, K, T]
loss_feat = xent(sim_feat, y)

# total loss
loss = loss_clf + loss_proto + loss_feat
return preds, loss
```

Algorithm 5.1: Single training step of our model.

**Temporal Losses.** In our investigation we noticed that our model, regarding the use of the prototypes, has a different behaviour depending on the temporal step the classification prediction is trained on. Specifically, when the model is trained for pure action classification, that is when only the last token is optimized, the model strongly performs on action recognition however it experiences a drastically low accuracy at the early temporal stages. By contrast, training the model on all the temporal steps leads to better performances at the early stages but degrades the accuracy in the final steps. To overcome the sensitivity of the model to the temporal shape of the loss function we introduce the dynamic loss that benefits from both temporal losses. More formally, we define the action recognition loss that optimizes only the last temporal step as:

$$\mathcal{L}_{ol}(\mathbf{x}, y) = \mathcal{L}_{clf}(\mathbf{x}, y, T) \tag{5.6}$$

and the loss that train all the temporal tokens at the same time as:

$$\mathcal{L}_{all}(\mathbf{x}, y) = \frac{1}{T} \sum_{t=1}^{T} \mathcal{L}_{clf}(\mathbf{x}, y, t). \tag{5.7}$$

To balance the trade-off of the two previously mentioned temporal objectives, we propose the dynamic loss that schedules the two losses in order to benefit from both the behaviours. More specifically, our loss is defined as:

$$\mathcal{L}_{dyn}(\mathbf{x}, y) = \begin{cases} \mathcal{L}_{ol}(\mathbf{x}, y) & \text{if } e \leq e^* \\ \mathcal{L}_{all}(\mathbf{x}, y) & \text{if } e > e^* \end{cases} \tag{5.8}$$

where $e$ is the training epoch and $e^*$ is the switch epoch.

The final total loss is the sum of the previously mentioned losses that are related to the early action recognition, the learning of the prototypes, and the training of the feature predictor:

$$\mathcal{L}_{tot} = \mathcal{L}_{dyn} + \mathcal{L}_{proto} + \mathcal{L}_{reg} \tag{5.9}$$

In Algorithm 5.1 we reported the PyTorch-like code for a single training step of our model.

## 5.4 Experiments

**Datasets.** We conducted experiments on several datasets: UCF-101 [187], EPIC-Kitchens-55 (EK55) [31], Something-Something v2 [77] (SSv2), a reduced version of Something Something (SSsub21) used in previous works [1, 107, 221, 220, 188]. UCF-101 [187] consists of 13k videos in total of 101 action classes. The dataset includes various types of human actions including human-object interactions, body-motions, human-human interactions, playing musical instruments and sports. As a common practice, we reported the results on the first split of the validation dataset. EPIC-Kitchens-55 [31] is an egocentric video dataset that contains 40k action segments of 2513 classes. Each action class is composed by a single verb and noun pair coming from a set of 125 unique verbs, and 352 unique nouns. We used the train and validation dataset split provided by [65]. The Something-Something v2 dataset [77] contains 169k training, and 25k validation videos. The videos show human-object interactions to be classified into 174 classes which is known as a 'temporal modeling'

task, and we report the accuracy on the validation set. Moreover, we used also a subset of the Something Something dataset containing 13k video samples of 21 classes for comparing with previous works.

### 5.4.1 Main Results

**Evaluation Metrics.** To measure the performance of our early action recognition model we follow previous works; at each partial observation of the input video the model is evaluated with the standard classification accuracy. Moreover, not only we are interested in maximizing the overall accuracy of the model, but we also focused on recognizing the action as early as possible. In order to capture both the performance aspects, thus taking into account different accuracies at different time steps, following previous works we measure the early recognition capability of the model with the Aurea Under the Curve (AUC) defined by the partial observation ratios and the accuracy curves. This metric aggregates all the accuracy information of different steps in a single scalar value.

**Early Action Recognition Results.** As reported in Table 5.2 our model achieves better results on UCF-101 w.r.t. previous methods on almost all partial observations. Specifically, we can improve by +3.55% at the observation ratio $\rho = 10\%$ and by +1.05% at $\rho = 70\%$ w.r.t. to TemPr [188] that uses MoViNet-A4 [106] that is a stronger encoder than MViT-B [47]. We obtain a similar improvement trend on the SSsub21 dataset as shown in Table 5.3 where we gain +3.68% at $\rho = 10\%$ with respect to TemPr [188] up to +13.64% at $\rho = 90\%$. On EK-55 we compare with RU-LSTM [65] that is a multi-modal encoder-decoder architecture. Even in this case we improve the results by +0.64% at $\rho = 12.5\%$ and by +1.25% of AUC. Moreover, it is worth to mention that our model uses the RGB input whereas RU-LSTM uses the RGB as well as the optical flow and an object pre-computed prediction. In Table 5.5 we report the results on SSv2. Although TemPr [188] uses V-Swin-B [132] that is a stronger visual backbone than our encoder, our method reaches higher accuracy on all the partial observations from +2.23% at $\rho = 10\%$ up to +11.44% at $\rho = 70\%$.

| Model | Encoder | Decoder | Inference Mode | Top-1 Acc | | | | | | | | | | AUC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% | 100% | |
| ProgTS [216] | ResNeXt-101 | LSTM | Online | 83.32 | 87.13 | 88.92 | 89.82 | 90.85 | 91.04 | 91.28 | 91.23 | 91.31 | 91.47 | 80.9 |
| RGN-KF [236] | TSN | RGN | Online | 83.1 | 85.16 | - | 90.78 | - | 92.03 | - | 93.19 | - | - | - |
| AA-GAN [66] | ResNet-50 | LSTM | Online | - | 84.2 | - | - | 85.6 | - | - | - | - | - | - |
| TemPr [188] | ResNet-18 | TemPr | Batched | 84.3 | 90.2 | 90.4 | 90.9 | 91.2 | 91.8 | 92.1 | 92.3 | 92.4 | - | - |
| TemPr [188] | ResNet-50 | TemPr | Batched | 84.8 | 90.5 | 91.2 | 91.8 | 91.9 | 92.2 | 92.3 | 92.4 | 92.6 | - | - |
| TemPr [188] | ResNeXt-101 | TemPr | Batched | 85.7 | 91.4 | 92.1 | 92.7 | 93.5 | 93.9 | 94.4 | 94.6 | 94.9 | - | - |
| TemPr [188] | X3D | TemPr | Batched | 87.9 | 93.4 | 94.5 | 94.8 | 95.1 | 95.2 | 95.6 | 96.4 | 96.3 | - | - |
| TemPr [188] | MoViNet-A4 | TemPr | Batched | 88.6 | 93.5 | 94.9 | 94.9 | 95.4 | 95.2 | 95.3 | **96.6** | 96.2 | - | - |
| Early-ViT (ours) | MViT-B | T-Dec-B | Online | **92.15** | **93.97** | **95.03** | **95.3** | **95.75** | **95.88** | **96.35** | 96.33 | **96.54** | **96.64** | **85.95** |

Table 5.2: UCF-101 Results.

| Model | Encoder | Decoder | Inference Mode | Top-1 Acc | | | | | | | | | | AUC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% | 100% | |
| MS-LSTM [1] | VGG-16 | MS-LSTM | Online | 16.9 | 16.6 | 16.8 | - | 16.7 | - | 16.9 | - | 17.1 | - | - |
| mem-LSTM [107] | ResNet-18 | mem-LSTM | Online | 14.9 | 17.2 | 18.1 | - | 20.4 | - | 23.2 | - | 24.5 | - | - |
| MSRNN [1] | VGG-16 | S-RNN | Online | 20.1 | 20.5 | 21.1 | - | 22.5 | - | 24.0 | - | 27.1 | - | - |
| GGN+LSTGCNN [221] | ResNet-50 | GNN | Online | 21.2 | 21.5 | 23.3 | - | 27.4 | - | 30.2 | - | 30.5 | - | - |
| IGGN+LSTGCNN [220] | ResNet-50 | GCN | Online | 22.6 | - | 25.0 | - | 28.3 | - | 32.2 | - | 34.1 | - | - |
| TemPr [188] | MoViNet-A4 | TemPr | Batched | 28.4 | 34.8 | 37.9 | - | 41.3 | - | 45.8 | - | 48.6 | - | - |
| Early-ViT (ours) | MViT-B | T-Dec-B | Online | **32.08** | **35.52** | **40.31** | **47.39** | **52.36** | **57.14** | **60.65** | **61.86** | **62.24** | **62.12** | **46.46** |

Table 5.3: SSsub21 Results.

| Model | Encoder | Decoder | Inference Mode | Top-1 Acc | | | | | | | | AUC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 12.5% | 25% | 37.5% | 50% | 62.5% | 75% | 87.5% | 100% | |
| RU-LSTM [65] (rgb) | TSN | RU-LSTM | Online | 20.09 | 22.97 | 24.38 | 25.54 | 26.75 | 27.49 | 27.84 | 28.32 | 22.40 |
| RU-LSTM [65] (rgb + obj + flow) | TSN | RU-LSTM | Online | 24.48 | 27.63 | 29.44 | 30.93 | 32.16 | 33.09 | 33.63 | 34.07 | 27.02 |
| Early-ViT (ours) | MViT-B | T-Dec-B | Online | **25.12** | **28.11** | **30.36** | **32.1** | **33.93** | **35.25** | **35.71** | **36.34** | **28.27** |

Table 5.4: EPIC-Kitchens-55 Results.

| Model | Encoder | Decoder | Inference Mode | Top-1 Acc | | | | | | | | | | AUC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% | 100% | |
| TemPr [188] | V-Swin-B | TemPr | Batched | 20.5 | - | 28.6 | - | 41.2 | - | 47.1 | - | - | - | - |
| Early-ViT (ours) | MViT-B | T-Dec-B | Online | **22.73** | **27.81** | **33.62** | **40.52** | **47.95** | **53.94** | **58.54** | **61.49** | **63.03** | **63.56** | **43.00** |

Table 5.5: Something Something v2 Results.

| Model | Encoder | Decoder | Inference Mode | Frames Per Clip | Top-1 Acc | | | | | | | | AUC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 12.5% | 25% | 37.5% | 50% | 62.5% | 75% | 87.5% | 100% | |
| MViT-B | MViT-B | - | Batched | 16 | 10.21 | 16.97 | 23.41 | 28.61 | 32.93 | **35.67** | **37.58** | **38.48** | 24.94 |
| Early-ViT (ours) | MViT-B | LinearAvgPool | Online | 2 | 18.04 | 19.12 | 20.41 | 21.69 | 22.37 | 22.72 | 22.98 | 23.00 | 18.72 |
| Early-ViT (ours) | MViT-B | T-Dec-B | Online | 2 | **25.12** | **28.11** | **30.36** | **32.1** | **33.93** | 35.25 | 35.71 | 36.34 | **28.27** |

Table 5.6: Early action recognition baselines of standard recognition models.

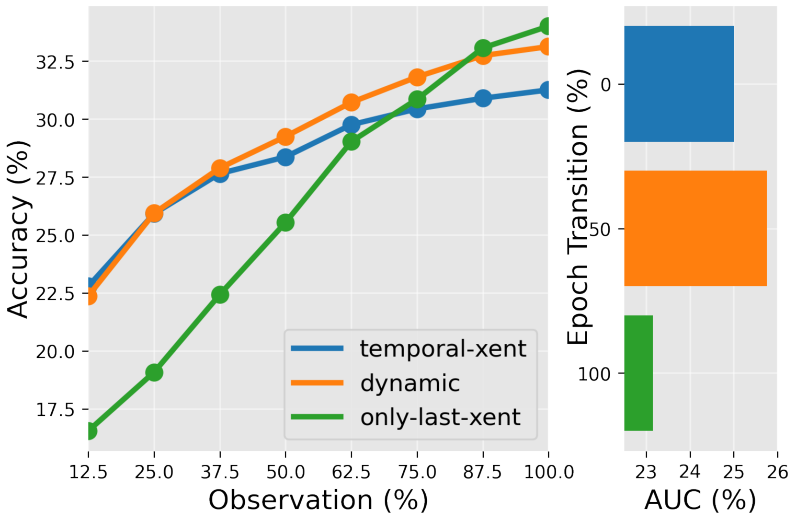



Figure 5.3: Role of the dynamic loss.

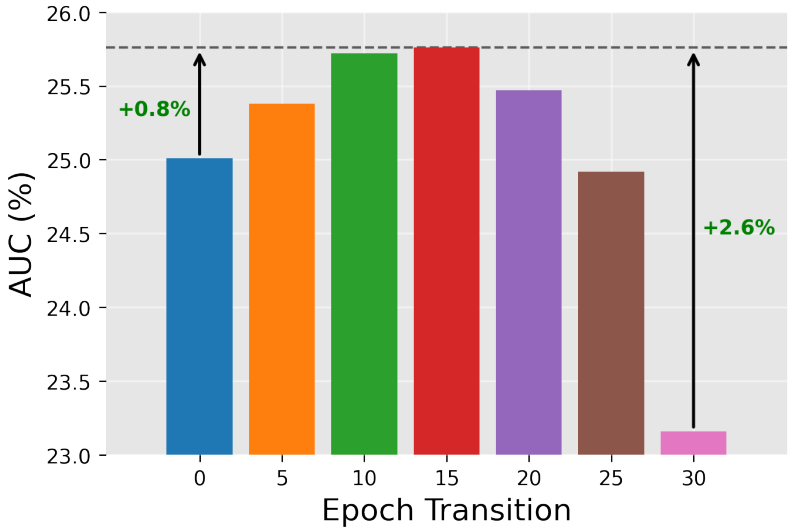



Figure 5.4: When to switch the losses in the dynamic loss scheduling.

| Model | Top-1 Acc | | | | | AUC |
|---|---|---|---|---|---|---|
| | 10% | 30% | 50% | 70% | 90% | |
| Early-ViT (no-proto) | 29.46 | 39.73 | 51.79 | 59.31 | 61.86 | 45.66 |
| Early-ViT (pred $z(t+1)$) | 30.48 | 39.99 | 52.23 | **61.29** | **62.24** | 46.24 |
| Early-ViT (pred $z(T)$) | 31.12 | 39.60 | 51.98 | 61.22 | 61.48 | 46.03 |
| Early-ViT | **32.08** | **40.31** | **52.36** | 60.65 | **62.24** | **46.46** |

(a) Model ablation on the source of regularization.

| | | | | | | |
|---|---|---|---|---|---|---|
| Early-ViT (R3D enc) | 27.74 | 38.07 | 48.15 | 54.53 | 56.76 | 42.39 |
| Early-ViT (MViT-B enc) | **32.08** | **40.31** | **52.36** | **60.65** | **62.24** | **46.46** |

(b) Our model trained using different encoder backbones.

| | | | | | | |
|---|---|---|---|---|---|---|
| Early-ViT ($F_c = 1$) | 27.61 | 31.95 | 37.56 | 43.75 | 46.05 | 34.74 |
| Early-ViT ($F_c = 2$) | 32.08 | 40.31 | 52.36 | 60.65 | 62.24 | 46.46 |
| Early-ViT ($F_c = 4$) | **32.91** | **44.01** | **57.14** | **66.84** | **69.45** | **50.8** |

(c) Varying the number of frames per clip.

| Model | Dataset | Prototypes | Top-1 Acc | | | | | | | | | | AUC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% | 100% | |
| Early-ViT | UCF-101 | ✗ | 91.65 | 93.74 | 94.45 | 95 | 95.59 | 95.72 | 96.01 | 96.14 | 96.14 | 96.03 | 85.66 |
| Early-ViT | UCF-101 | ✓ | **92.15** | **93.97** | **95.03** | **95.3** | **95.75** | **95.88** | **96.35** | **96.33** | **96.54** | **96.64** | **85.95** |
| Early-ViT | SSsub21 | ✗ | 29.46 | 33.23 | 39.73 | 46.68 | 51.79 | 56.76 | 59.31 | 61.86 | 61.86 | 61.22 | 45.66 |
| Early-ViT | SSsub21 | ✓ | **32.08** | **35.52** | **40.31** | **47.39** | **52.36** | **57.14** | **60.65** | **61.86** | **62.24** | **62.12** | **46.46** |
| Early-ViT | SSv2 | ✗ | 22.35 | 27.5 | 33.52 | 39.95 | 47.17 | 52.62 | 57.04 | 59.95 | 61.36 | 61.64 | 42.11 |
| Early-ViT | SSv2 | ✓ | **22.73** | **27.81** | **33.62** | **40.52** | **47.95** | **53.94** | **58.54** | **61.49** | **63.03** | **63.56** | **43.00** |

Table 5.8: How the use of the prototypes impacts the final results.

| Model | Dataset | Prototypes | Top-1 Acc | | | | | | | | AUC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 12.5% | 25% | 37.5% | 50% | 62.5% | 75% | 87.5% | 100% | |
| Early-ViT | EK-55 | ✗ | 24.28 | 27.48 | 30.13 | 31.55 | 32.94 | 34.08 | 34.81 | 35.57 | 27.61 |
| Early-ViT | EK-55 | ✓ | **25.12** | **28.11** | **30.36** | **32.1** | **33.93** | **35.25** | **35.71** | **36.34** | **28.27** |

Table 5.9: How the use of the prototypes impacts the final results.

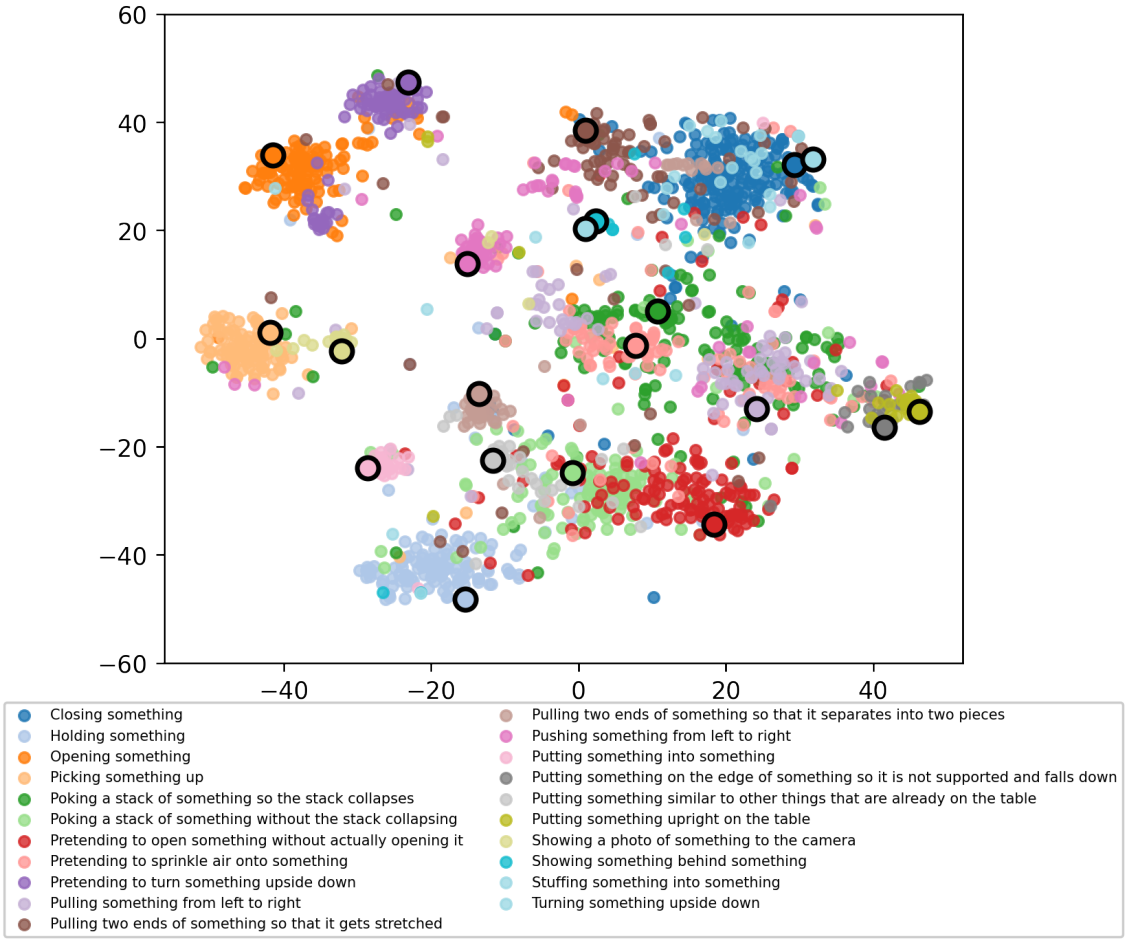



Figure 5.5: t-SNE visualization of the learned prototypes. Different colours are related to different classes. The prototypes are the edged dots while all the other points are the feature embeddings of the videos. At the bottom we reported the classes of the SSsub21 dataset.

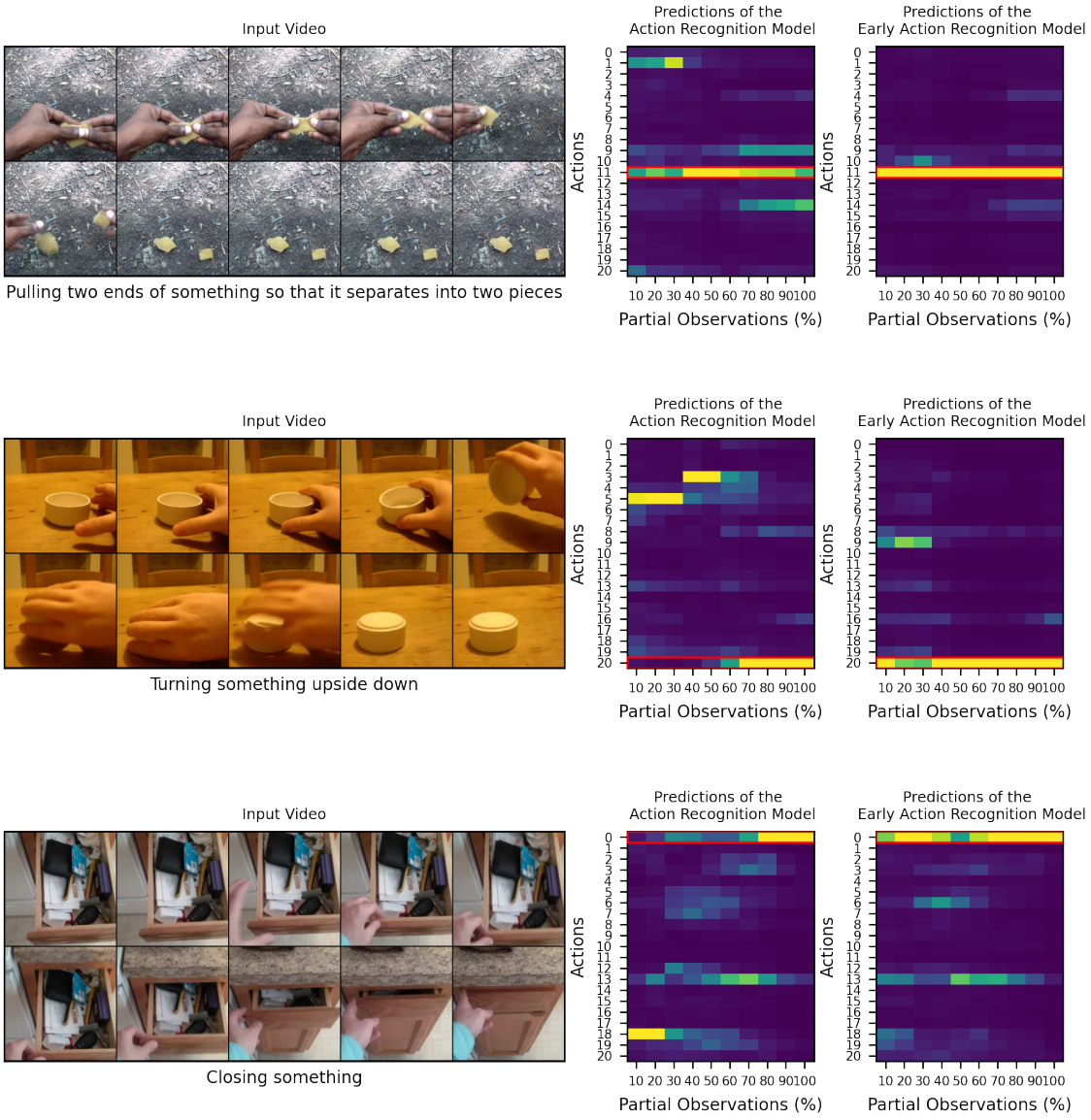



Figure 5.6: Qualitative comparison between the action recognition model and our early action recognition model. On the left we reported the sampled video with the corresponding action class on the bottom. On the right we reported the softmax predictions of the two models over different partial observation of the video ($x$-axis), for each class ($y$-axis). The row highlighted with the red box corresponds to the ground truth class.

# Chapter 6

# Open Set Recognition

## 6.1 Introduction

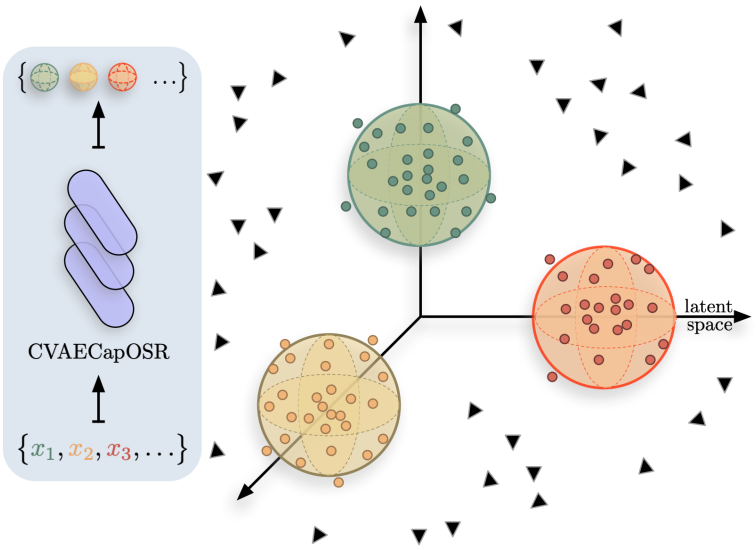



Figure 6.1: CVAECapOSR Model. Input samples are fed into the Capsule Network that produces distributions over the latent space. Each class has its own prior gaussian distribution in the feature space, that in the figure are represented as spheres. After training, the known samples (represented as small points) are clustered around the class target gaussians. The samples belonging to unknown classes are represented as black triangles, far from the target distributions.

[1]Over the past decade, deep learning has become the dominant approach in many computer vision problems, achieving spectacular results on many visual recognition tasks [36, 182, 81, 194]. However, most of these results have been obtained in a closed set scenario, where a critical assumption is that all samples should belong to at least one labeled category. When observing a sample from an unknown category, closed-set approaches are forced to choose a class label from one of the known classes, thus limiting their applicability in dynamic and ever-changing scenarios.

To overcome such a limitation, open set recognition has been introduced to enable a classification system to identify all of known categories, while simultaneously detecting unknown test samples [174, 9]. In the open set scenario, samples included/excluded in label space are referred to as knowns/unknowns. Therefore, open set classifiers need to use incomplete knowledge learned from a finite set of accessible categories to devise effective representations able to separate knowns from unknowns. Early works have identified this issue, thus proposing methods employing different thresholding strategies for rejection of unknowns [174, 9].

Deep neural networks, despite demonstrating strong capabilities in learning discriminative representations in closed scenarios, show accuracy degradation within open set settings [10]. As a naive strategy, modeling a threshold for Softmax outputs has been demonstrated to be a sub-optimal solution for deep neural networks to identify unknowns. Thus the Extreme Value Theory was introduced to better adapt these discriminative models, fully based on supervised learning, for open-set settings. The underpinning idea is to calibrate Softmax scores so to estimate the probability of unknowns [226, 10]. In addition to deep discriminative models, deep generative models focusing on learning efficient latent feature representations by unsupervised leaning, have been widely utilized in open-set recognition tasks, and have gained successes one after the other [145, 157, 164, 191]. In particular, The Variational Auto-Encoder (VAE) is a typical probabilistic generative model ideal for detecting unknowns, due to its ability in learning low-dimensional representations in latent space not only supporting input reconstruction but also approximating a specified prior distribution. On

[1]Y. Guo*, **G. Camporese***, W. Yang, A. Sperduti, L. Ballan. "Conditional Variational Capsule Network for Open Set Recognition", *Proc. of IEEE/CVF International Conference on Computer Vision (ICCV 2021)* (* means equal contribution).

the other hand, the VAE-based models may be not sufficiently effective for identifying known categories as all feature representations only follow one distribution. To this end, we employ a Conditional VAE (CVAE) that uses multiple prior distributions for modeling the known classes, and indirectly the unknown counterpart. Furthermore, we propose to represent the input samples with probabilistic capsules, given their already proved representation power capability [167, 173].

Capsule Networks (CapsNet) [173] were proposed as an alternative to Convolutional Neural Networks (CNNs). Unlike CNNs' scalar neurons, capsules ensemble a group of neurons to accept and output vectors. The vector of an activated capsule represents the various properties of a particular object, such as position, size, orientation, texture, etc. In essence, CapsNet can be viewed as an encoder encoding objects by distributed representations, which is exponentially more efficient than encoding them by activating a single neuron in a high-dimensional space. Besides, CapsNet has been successfully used to detect fake images and videos in a task setting similar to open set recognition [150]. This motivated us to design a novel capsule network architecture in combination with CVAE for the open set recognition problem, dubbed CVAECapOSR, that is depicted in Figure 6.1.

The contributions of this paper are three-fold: *i)* We present a novel open set recognition framework based on CapsNet and show its advantages for learning an efficient representation for known classes.

*ii)* We integrate CapsNet and conditional VAEs. In contrast to general VAEs that encourage the latent representation to approximate a single prior distribution, our model exploits multiple priors (i.e. one for each class), and it forces the latent representation to follow the gaussian prior selected by the class of the input sample.

*iii)* We conduct extensive experiments on all the standard datasets used for open set recognition, obtaining very competitive results, that in several cases outperform previous state of the art methods by a large margin.

## 6.2 Related Work

The open set recognition problem was introduced by [174] and was initially formalized as a constrained minimization problem based on Support Vector Machines (SVMs), whereas subsequent works focused on other more traditional approaches, such as Extreme Value Theory [95, 175], sparse representation [232], and Nearest Neighbors [99].

Following the success achieved by deep learning in many computer vision tasks, deep networks were first introduced for open set recognition in [10], in which it is proposed an Openmax function by calibrating the Softmax probability of each class with a Weibull distribution model. Subsequently, [69] extended Openmax to G-Openmax by introducing a generative adversarial network in which the generator produces synthetic samples of novel categories and the discriminator learns the explicit representation for unknown classes. A similar strategy has been adopted in [145], that presented a data augmentation technique based on generative adversarial networks, referred as counterfactual images generation. More recently, Yoshihashi *et al.* [226] analyzed and demonstrated the usefulness of training deep networks jointly for classification and reconstruction in the open set scenario. Specifically, the authors proposed to separate the knowns from the unknowns using the representations produced by the unsupervised training, while maintaining the discrimination capability of the model using the representations computed via the supervised learning process.

C2AE [157] introduced an architecture for open set recognition and unknown detection based on class conditioned VAEs by modelling the reconstruction error of the model based on the Extreme Value Theory. Sun *et al.* [191] have recently argued that one disadvantage of VAE-based architectures for open set recognition is the inadequate discriminative ability on instances of known classes. Therefore, the authors employed a conditional Gaussian distribution VAE model for learning conditional distributions of known classes and rejecting unknowns. A different approach is presented in [231], where normalizing flows are employed for density estimation of known samples. Specifically, the authors proposed an architecture that uses a CNN encoder and an invertible neural network that jointly learns the density of the input. However, a potential issue not discussed in the paper is that the CNN encoder has

no bijective property, that is crucial to employ the change-of-variables formula for density evaluation. Additionally, [21] introduced the concept of reciprocal points in prototype leaning to manage the open space. Although this work shows excellent performances in rejecting unknowns coming from a different dataset with respect to the known samples, the unknown detection capability degradates when the source of unknown samples is the same of the known counterpart.

## 6.3 Preliminaries

### 6.3.1 The Open Set Recognition Problem

In the open set recognition problem the model has to classify test samples that can belong to classes not seen during training. Given a classification dataset $D = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ such that $\boldsymbol{x}_i \in \mathcal{X}$ is an input sample, $y_i \in \{1, \ldots, K\}$ is the corresponding category label, the open set problem consists in the classification of test samples among $K + U$ classes, where the $U$ are the number of unknown classes. In the literature, the dataset used for training is called *closed* dataset, meanwhile the one used during evaluation, that contains samples from unseen classes, is called *open* dataset. In order to quantify the openness of a dataset during the evaluation, following [174, 145] we consider the *openness* measure as $O = 1 - \sqrt{\frac{K}{M}}$ where $K$ and $M = K + U$ are the number of classes observed during training and test, respectively.

### 6.3.2 Conditional VAE Formulation

The Conditional Variational Eutoencoder (CVAE) directly derives from the VAE model [103] and its objective is based on the estimation of the conditioned density $p(x|y)$ of the data $x$ given the label $y$. It is one of the most powerful probabilistic generative models for its theory elegancy, strong framework compatibility and efficient manifold representations. The CVAEs commonly consist on an encoder that maps the input $\boldsymbol{x}$ and class $y$ to a pre-fixed distribution over the latent variable $\boldsymbol{z}$, and on a decoder that, given a latent variable $\boldsymbol{z}$ and the class $y$ tries to reconstruct the input $\boldsymbol{x}$. During training the model is trained by minimizing the negative variational lower

bound of the conditional density of the data, defined as follows:

$$\mathcal{L}\big[\boldsymbol{x}, y; \theta, \phi\big] = D_{\mathrm{KL}}\Big[q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z}|y)\Big] - \mathbb{E}_{q_\phi(\boldsymbol{z}|x)}\Big[\log p_\theta(\boldsymbol{x}|\boldsymbol{z}, y)\Big]$$

where $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ denotes the posterior of the encoder, and $p_\theta(\boldsymbol{z}|y)$ indicates the prior distribution over the latent variable $\boldsymbol{z}$ conditioned on the class $y$. The first term in the loss function is a regularizer that enforces the approximate posterior distribution $q_\phi(\boldsymbol{x}|\boldsymbol{z})$ to be close to the prior distribution $p_\theta(\boldsymbol{z}|y)$, while the second term is the average reconstruction error of the chained encoding-decoding process. The original VAE [103], that uses an unconditioned prior distribution, presumes that $p_\theta(\boldsymbol{z})$ is an isotropic multivariate Gaussian $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is a general multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma^2})$. With these assumptions, the KL-divergence term given a $K$-dimensional $\boldsymbol{z}$, can be computed in closed form and expressed as:

$$D_{\mathrm{KL}}\Big[q_\phi(\boldsymbol{z}|\boldsymbol{x})\|p_\theta(\boldsymbol{z})\Big] = -\frac{1}{2}\sum_{i}^{K}(1 + \log(\boldsymbol{\sigma}_i^2) - \boldsymbol{\mu}_i^2 - \boldsymbol{\sigma}_i^2).$$

For the CVAE the KL-divergence term can be computed or estimated using only tractable latent prior distributions $p(\boldsymbol{z}|y)$ [158].

### 6.3.3 CapsNet Formulation

The capsule network, proposed by Hinton et al. [86], is a shallow architecture composed by two convolutional layers and two capsule layers. The first convolutional operation converts the pixel intensities of the input image $\boldsymbol{x}$ to primary local feature maps, while the second convolutional layer produces the primary capsules $\mathbf{u}_i$. Each capsule corresponds to a set of matrices that rotate the primary capsules for predicting the pose transformation $\hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij}\mathbf{u}_i$. Afterwards, the digit capsules $\mathbf{v}_j$, used for classification, are produced as the weighted sum of the primary capsules $\mathbf{v}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$, where the coefficients $c_{ij}$ are determined by the dynamic routing algorithm (DR), in which the primary capsules are compared to the digit capsules.

For $t$-th iteration of DR, the coefficients are updated by,

$$\mathbf{c}_i^{(t+1)} = \mathbf{Softmax}(\mathbf{b}_i^{(t+1)}), \quad b_{ij}^{(t+1)} = b_{ij}^{(t)} + \hat{\mathbf{u}}_{j|i} \cdot \mathbf{v}_j^{(t)}.$$

For all layers of capsules, a squash function is used to introduce non-linearity and shrunk the length of capsule vectors into $[0, 1]$,

$$\mathbf{Squash}(\mathbf{v}) = \frac{\|\mathbf{v}\|^2}{1 + \|\mathbf{v}\|^2} \frac{\mathbf{v}}{\|\mathbf{v}\|}.$$

In this way the norm of the capsule stands for the probability of a particular feature being present in the input image $\boldsymbol{x}$.

## 6.4 Proposed Method

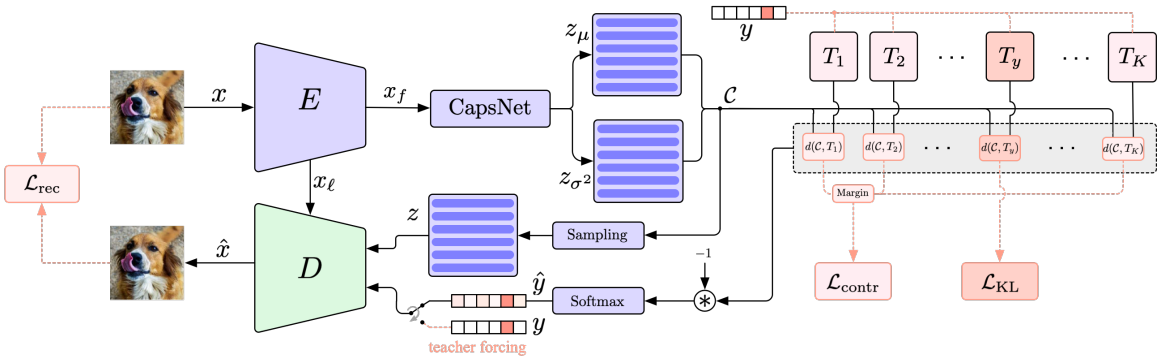



Figure 6.2: Outline of our CVAECapOSR model. The dashed orange lines stand for the computation of the model during training, whereas the solid black lines indicate the computation done by the model both during training and testing time.

### 6.4.1 Model Architecture

Our model, depicted in Figure 6.2, is based on a CVAE with $K$ different gaussian prior distributions, one for each known class. Given the input image $\boldsymbol{x}$ and its corresponding label $y$ the encoder processes $\boldsymbol{x}$ producing the feature representation $\boldsymbol{x}_f$. Afterward the capsule network computes the distribution $q(\boldsymbol{z}|\boldsymbol{x})$ that is pushed toward

the conditioned prior $p(\boldsymbol{z}|y) = \boldsymbol{T}_y$ during the learning process. Using the distance information between $q(\boldsymbol{z}|\boldsymbol{x})$ and all the targets we estimate the class $\hat{y}$, and using the reparametrization trick we sample $\boldsymbol{z}$ from $q(\boldsymbol{z}|\boldsymbol{x})$. Given $\hat{y}$ and $\boldsymbol{z}$ we compute the reconstruction $\hat{\boldsymbol{x}}$ through the decoder that is a convolutional neural network that uses transposed convolutions. After this general description of the computation of our model, we now present in deep the architecture step by step.

**Encoding Stage.** The blocks involved in the encoding stage are an encoder and a capsule network. The encoder is a convolutional neural network that processes the input image $\boldsymbol{x} \in \mathbb{R}^{C \times H \times W}$ producing the feature $\boldsymbol{x}_f \in \mathbb{R}^{d_c \times d_h \times d_w}$. Then, similar to [173], the capsule network processes the feature representation $\boldsymbol{x}_f$ by computing the primary capsules $\boldsymbol{x}_{\text{pc}} \in \mathbb{R}^{f_1 \times (\frac{d_c}{f_1} d_h d_w)}$ and then the digit capsules $\boldsymbol{x}_{\text{dc}} \in \mathbb{R}^{K \times f_2}$ using the dynamic routing algorithm. We indicate with $f_1$ the dimension of the primary capsules and with $f_2$ the dimension of the digit capsules. Given the capsules $\boldsymbol{x}_{\text{dc}}$ we compute the mean $\boldsymbol{z}_\mu \in \mathbb{R}^{K \times d}$ and variance $\boldsymbol{z}_{\sigma^2} \in \mathbb{R}^{K \times d}$ of the capsules distribution $\boldsymbol{\mathcal{C}} = q(\boldsymbol{z}|\boldsymbol{x})$ by applying a capsule-wise fully connected layer with $d$ output units. In this way the probabilistic capsule network produces $K$ mean capsules $\{\boldsymbol{z}_\mu^{(k)}\}_{k=1}^K$ and $K$ variance capsules $\{\boldsymbol{z}_{\sigma^2}^{(k)}\}_{k=1}^K$, each of size $d$.

**Contrastive Variational Stage.** We design each CVAE prior target $p(\boldsymbol{z}|y = k)$ to be a gaussian distribution $\boldsymbol{T}_k = \mathcal{N}\left(\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\Sigma}}_k\right)$ with learnable mean vector $\tilde{\boldsymbol{\mu}}_k \in \mathbb{R}^{Kd}$ and learnable diagonal covariance matrix $\tilde{\boldsymbol{\Sigma}}_k \in \mathbb{R}^{Kd \times Kd}$ with $1 \leq k \leq K$. In order to simplify the notation of our model framework we consider the targets $\boldsymbol{T}_k$ as gaussian distributions defined by $\boldsymbol{\mu}_k \in \mathbb{R}^{K \times d}$ being the reshaped version of the $\tilde{\boldsymbol{\mu}}_k$ and $\boldsymbol{\Sigma}_k \in \mathbb{R}^{K \times d}$ being the reshaped diagonal of $\tilde{\boldsymbol{\Sigma}}_k$. In order to map input instances from the same class into compacted and separated regions of the latent space, during the learning process, we let the probabilistic capsule $\boldsymbol{\mathcal{C}}$ to be attracted by the $y$-th target distribution $\boldsymbol{T}_y$ and at the same time we let all the other targets $\boldsymbol{T}_{\neq y}$ to be repulsed by $\boldsymbol{\mathcal{C}}$. Using this contrastive strategy, we encourage the encoded representation to belong to the correct region of the latent space while maintaining all the targets sufficiently

far apart to each other. We then estimate the class $\hat{y}$ of the input sample $\boldsymbol{x}$ as:

$$\hat{y}_k = p(y = k|\boldsymbol{x}) = \frac{e^{-\gamma d(\boldsymbol{\mathcal{C}}, \boldsymbol{T}_k)}}{\sum_{j=1}^{K} e^{-\gamma d(\boldsymbol{\mathcal{C}}, \boldsymbol{T}_j)}}, \quad 1 \leq k \leq K,$$

where

$$d(\boldsymbol{\mathcal{C}}, \boldsymbol{T}_i) = \frac{1}{K} \sum_{k=1}^{K} D_{\mathrm{KL}} \Big[ \boldsymbol{\mathcal{C}}^{(k)} || \boldsymbol{T}_i^{(k)} \Big],$$

is the distance between the probabilistic capsules $\boldsymbol{\mathcal{C}}$ and the target $\boldsymbol{T}_i$, and $\gamma$ is a coefficient parameter that controls the hardness of the probability assignment. In this way we estimate the probability of $\boldsymbol{x}$ being of class $k$ considering the whole configuration of capsules $\{\boldsymbol{\mathcal{C}}^{(k)}\}_{k=1}^{K}$ and not only the single most activated capsule as done in [173].

**Decoding Stage.** Given the class estimate $\hat{y} \in \mathbb{R}^K$ we compute its learnable embedding

$$\hat{y}_e = \mathrm{Embedding}(\arg\max_k(\hat{y}_k)) \in \mathbb{R}^d,$$

and given the sampled latent capsules $\boldsymbol{z} \in \mathbb{R}^{K \times d}$ we compute the reconstruction $\hat{\boldsymbol{x}}$ through the decoder starting from $\boldsymbol{z}_y \in \mathbb{R}^{K \times d}$ with $\boldsymbol{z}_y^{(k)} = \hat{y}_e + \boldsymbol{z}^{(k)}$. The decoder is a convolutional neural network with transposed convolutions that follows a symmetrical structure of the encoder. Similar to [171], we implement lateral connections $\boldsymbol{x}_\ell$ with $1 \leq \ell \leq 4$ from the internal features of the encoder to the decoder that during training are randomly dropout for making the decoder less dependent from the internal representations of the encoder.

### 6.4.2 Training

We train the model on the closed dataset, and during the learning process for a single input sample $(\boldsymbol{x}, y)$ we minimize the following loss function:

$$\mathcal{L}(\boldsymbol{x}, y) = \mathcal{L}_{\mathrm{KL}}(\boldsymbol{x}, y) + \alpha \mathcal{L}_{\mathrm{contr}}(\boldsymbol{x}, y) + \beta \mathcal{L}_{\mathrm{rec}}(\boldsymbol{x}), \tag{6.1}$$

where

$$\mathcal{L}_{\text{KL}}(\boldsymbol{x}, y) = d(\boldsymbol{\mathcal{C}}, \text{sg}[\boldsymbol{T}_y]), \tag{6.2}$$

$$\mathcal{L}_{\text{contr}}(\boldsymbol{x}, y) = \frac{1}{K-1} \sum_{k \neq y}^{K} \left[ m_k - d(\text{sg}[\boldsymbol{\mathcal{C}}], \boldsymbol{T}_k) \right]^+, \tag{6.3}$$

$$\mathcal{L}_{\text{rec}}(\boldsymbol{x}) = \left\| \hat{\boldsymbol{x}} - \boldsymbol{x} \right\|_2^2. \tag{6.4}$$

As already defined in [201], the function sg[·] in Eq. (6.2) and Eq. (6.3) stands for the stop-gradient operator that is defined as the identity at forward computation time and has zero partial derivatives, constraining its argument to be a non-updated constant. The loss term in Eq. (6.2) is responsible for pushing the probabilistic capsules $\boldsymbol{\mathcal{C}}$ toward the target $\boldsymbol{T}_y$, leading to the concentration of all the density of known samples in the targets region. On the other hand, the contrastive loss term in Eq. (6.3) pushes all the targets not related to $y$ far away from the distribution $\boldsymbol{\mathcal{C}}$ using a margin loss with margin $m_k$ where $[\cdot]^+$ is the function that returns the positive part of its argument. By considering $\boldsymbol{T}_{\neq y}$ to be the otherness of $\boldsymbol{T}_y$, therefore of $\boldsymbol{\mathcal{C}}$, the contrastive term not only avoid the collapse of the prior targets, but also encourages the separation between one class and *all the other* classes, potentially the unknown counterpart. Finally, the loss term in Eq. (6.4) is the mean squared error reconstruction between the input and the output of the model. We aggregate all loss terms in Eq. (6.1) and we control the strength of $\mathcal{L}_{\text{contr}}$ with a parameter $\alpha$ and the strength of $\mathcal{L}_{\text{rec}}$ with a parameter $\beta$. During training we found beneficial to use teacher forcing in the decoder i.e. we decided to feed $y$ instead of the estimate $\hat{y}$ to the decoder, while during validation and testing we feed only the estimated quantity, enabling in this way, the independence of our model respect to the label $y$ during inference.

### 6.4.3 Inference

We use the model's natural rejection rule based on the probabilistic distance between samples and targets to detect unknowns, and directly classify known samples with

the minimum probabilistic distance over than a given threshold.

Given a new sample $\boldsymbol{x}$ we decide if it is an outlier as follows:

$$\hat{y} = \begin{cases} K+1, & \text{if } \max_k \left\{ d(\boldsymbol{\mathcal{C}}, \boldsymbol{T}_k) \right\} = d^* < \tau \\ \arg\max_k \left\{ d(\boldsymbol{\mathcal{C}}, \boldsymbol{T}_k) \right\}, & \text{otherwise.} \end{cases}$$

where $\tau$ is found using cross validation, and $K+1$ is the new, unknown class not seen during training.

## 6.5 Experiments

Recent works in this area followed the protocol presented in [145]. In that work, an open set recognition scenario is obtained by randomly selecting $K$ classes from a specific dataset as known (see below for more details), while the remaining classes are considered to be open set classes. This procedure is applied to five random splits. However, as recently shown by [164], performance across different splits varies significantly (e.g. AUROC on CIFAR10 varied between 77% to 87% across different splits), and there are serious reproducibility issues. Moreover, not only the splits have a large influence on the results, but also the strategy used to select the samples belonging to the unknown classes. Therefore, starting from the splits used in [145] and following [164], we publicly release our code and data[2], as well as the implementation of other state-of-the-art methods, to foster a fair comparison on this task.

### 6.5.1 Datasets

We evaluate open set recognition performance on the standard datasets used in previous works, i.e. MNIST [120], SVHN [147], CIFAR10 [113], CIFAR+10, CIFAR+50 and TinyImageNet [119].

**MNIST, SVHN, CIFAR10.** All three datasets contain ten categories. MNIST consists of hand-written digit images, and it has $60,000$ $28 \times 28$ grayscale images for

[2]Code and data publicly available on https://github.com/guglielmocamporese/cvaecaposr.

| Method | MNIST | SVHN | CIFAR10 | CIFAR+10 | CIFAR+50 | TinyImageNet |
|---|---|---|---|---|---|---|
| Softmax † [191] | 0.978 | 0.886 | 0.677 | 0.816 | 0.805 | 0.577 |
| Openmax † [10] | 0.981 | 0.894 | 0.695 | 0.817 | 0.796 | 0.576 |
| G-Openmax † [69] | 0.984 | 0.896 | 0.675 | 0.827 | 0.819 | 0.580 |
| OSRCI † [145] | 0.988 ±0.004 | 0.91 ±0.01 | 0.699 ±0.038 | 0.838 | 0.827 | 0.586 |
| CROSR [226] | 0.991 ±0.004 | 0.899 ±0.018 | - | - | - | 0.589 |
| C2AE ‡ [157] | - | 0.892 ±0.013 | 0.711 ±0.008 | 0.810 ±0.005 | 0.803 ±0.000 | 0.581 ±0.019 |
| GFROR ‡ [164] | - | 0.955 ±0.018 | 0.831 ±0.039 | - | - | 0.657 ±0.012 |
| CGDL § [191] | 0.977 ±0.008 | 0.896 ±0.023 | 0.681 ±0.029 | 0.794 ±0.013 | 0.794 ±0.003 | 0.653 ±0.002 |
| RPL § [21] | 0.917 ±0.006 | 0.931 ±0.014 | 0.784 ±0.025 | 0.885 ±0.019 | 0.881 ±0.014 | 0.711 ±0.026 |
| **CVAECapOSR (ours)** | **0.992** ±0.004 | **0.956** ±0.012 | **0.835** ±0.023 | **0.888** ±0.019 | **0.889** ±0.017 | **0.715** ±0.018 |

Table 6.1: AUROC scores on the detection of known and unknown samples. Results are averaged over 5 different splits of known and unknown classes partitions. As discussed in Section 6.5.3, we report the results on the same data splits and, for the sake of clarity, we highlight the source of the results used to populate the table: † are provided by [145], ‡ is from [164] and § are the results that we obtained by running the code of the original paper.

| Method | Learn Target | MNIST | SVHN | CIFAR10 | CIFAR+10 | CIFAR+50 | TinyImageNet |
|---|---|---|---|---|---|---|---|
| CVAECapOSR | ✗ | **0.997** ±0.006 | 0.953 ±0.022 | 0.823 ±0.012 | 0.868 ±0.018 | 0.829 ±0.009 | 0.706 ±0.014 |
| CVAECapOSR | ✓ | 0.992 ±0.004 | **0.956** ±0.012 | **0.835** ±0.023 | **0.888** ±0.019 | **0.889** ±0.017 | **0.715** ±0.018 |

Table 6.2: AUROC scores on the detection of known and unknown samples comparing our model that uses fixed targets (first row) versus our model that learns the targets (second row) during the learning process. Results are averaged over 5 splits.

training and $10,000$ for testing. SVHN contains street view house numbers, consisting of ten digit classes each with between $9,981$ and $11,379$ $32 \times 32$ color images. Then we consider the CIFAR10 dataset, which has $50,000$ $32 \times 32$ color images for training and $10,000$ for testing. Following [145], in the unknown detection task each dataset is randomly partitioned into 6 known classes and 4 unknown classes. In this setting, the openness score is fixed to 22.54%.

**CIFAR+10, CIFAR+50.** To test our model in a setting of higher openness values, we perform CIFAR+$Q$ experiments using CIFAR10 and CIFAR100 [113]. To this end, 4 known classes are sampled from CIFAR10 and $Q$ unknown classes are drawn randomly from the more diverse and larger CIFAR100 dataset. Openness scores of CIFAR+10 and CIFAR+50 are 46.54% and 72.78%, respectively.

**TinyImageNet.** For TinyImagenet dataset, which is a subset of ImageNet that

contains 200 classes, we randomly sampled 20 classes as known and the remaining classes as unknown. In this setting, the openness score is 68.37%.

### 6.5.2 Metrics

Open set classification performance is usually measured using F-score and AUROC (Area Under ROC Curve) [70]. F-score is used to measure the in-distribution classification performance, while AUROC is commonly reported by both open set recognition and out-of-distribution detection literature. AUROC provides a calibration free measure and characterizes the performance for a given score by varying the discrimination threshold [33]. In our experiments, we use macro averaged F1-score on the open set recognition task, and the AUROC for the unknown detection task. For both metrics, higher values are better.

### 6.5.3 Experimental Results

Following [191], we conducted two major experiments in which our model has to solve the *unknown detection* task and the *open set recognition* task. For all the experiments we use ResNet34 [81] as the encoder backbone of our model.

**Unknown Detection.** In the unknown detection problem the model is trained on a subset of the dataset using $K$ classes, and the evaluation is done by measuring the model capability on detecting unknown classes, not seen during training. The evaluation is performed by considering the binary recognition task of the known *vs* unknown classes, and performances are reported in terms of AUROC scores. The results, shown in Table 6.1, are averaged over five random splits of known and unknown classes, provided by [145].

As already discussed (and recently shown in [70, 164]), performance across different splits varies significantly. For this reason we use the exact data splits provided by [145] that have been used also in other recent works [226, 164].

Nevertheless, not all the results reported in these works are directly comparable; although the splits are the same, [164] followed a particular strategy in selecting the open set classes in CIFAR+10 and CIFAR+50 experiments (i.e. they selected 10 and

50 samples from vehicle classes instead of purely random classes, which has gained a large impact on these results).

Therefore, following [164], we have run the code of [191] and [21] (whereas the results of [157] are provided by [164], since the code is no more available), and we compare all the results with the state of the art papers that have the same splits and use the same evaluation setting, and that can be reproduced.

As shown in Table 6.1, we obtain state of the art results, outperforming all previous methods, on all the datasets. Moreover, as also previously reported, we will release all data and code to guarantee reproducibility.

One important fact we observed during the training process is the boost we obtained by letting the targets distributions $\boldsymbol{T}_k$ to be learned and not to be used as fixed priors, as shown in Table 6.2. We initialized the learnable targets with $\boldsymbol{\mu}_k^{(i)} = \mathbb{1}_d \cdot \delta_{k=i}$ and $\boldsymbol{\Sigma}_k^{(i)} = \mathbb{1}_d$. We noticed that learning the targets without considering the contrastive term in the loss function ($\alpha = 0$) caused the collapse of the targets into one single distribution, leading to poor results. We thus consider the contrastive term, and we set $\alpha = 1.0$, $\beta = 0.05$ and $m_k = 10$.

**Open Set Recognition.** In the open set recognition problem, the model is trained on the closed dataset that contains $K$ classes, and it is evaluated on the open dataset considering $K+1$ classes. In this experimental setting, we evaluate the model using the macro F1-score on the $K+1$ classes. In the first experiment for open set recognition, we train on all the classes of the MNIST dataset and we then evaluate the performances by including new datasets in the open set. Similarly to [226], we used Omniglot, MNIST-Noise, and Noise that are datasets of gray-scale images. Each of this dataset contains $10,000$ test images, the same as MNIST. The Omniglot dataset contains hand-written characters from the alphabets of many languages, while the Noise dataset has images synthesized by randomly sampling each pixel value independently from a uniform distribution on $[0, 1]$. MNIST-Noise is also a synthesized set, constructed by superimposing MNIST's test images on Noise. The results of the open set recognition on these datasets are shown in Table 6.3. On each dataset, we outperform state of the art results by a large margin. On Omniglot, we improve the F1-score by $+0.121$, in the MNIST-Noise by $+0.095$, and in the Noise by $+0.123$.

| Method | Omniglot | MNIST-noise | Noise |
|---|---|---|---|
| Softmax [191] | 0.595 | 0.801 | 0.829 |
| Openmax [69] | 0.780 | 0.816 | 0.826 |
| CROSR [226] | 0.793 | 0.827 | 0.826 |
| CGDL [191] | 0.850 | 0.887 | 0.859 |
| **CVAECapOSR (ours)** | **0.971** | **0.982** | **0.982** |

Table 6.3: Results for the open set recognition on the MNIST dataset. We report the macro-averaged F1-score for 11 classes (10 from the test partition of the MNIST, and 1 from the test of another dataset).

(a) CVAECapOSR with Capsules

(b) CVAECapOSR with Standard Neurons

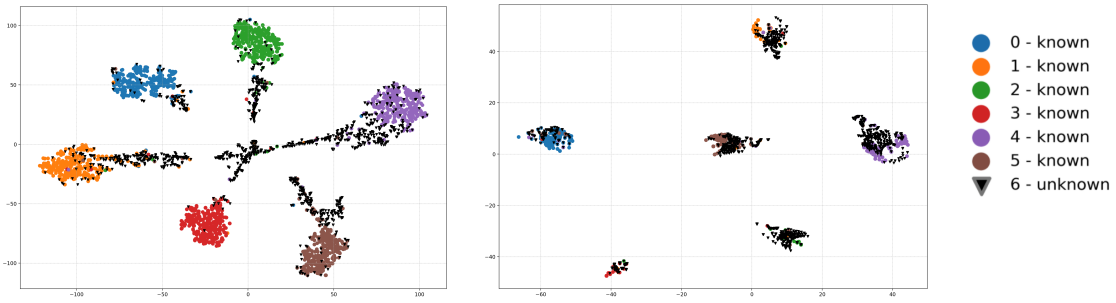



Figure 6.3: t-SNE latent space visualizations obtained with different components (i.e., (a) Capsules, (b) Standard Neurons) of the SVHN test set including the CIFAR+100 test set from which unknown samples are sampled. In particular, we use an openness value of $O = 35.67\%$. In both pictures, the unknown samples are represented by black triangles.

In the second experiment of open set recognition, following the same protocol used in [145], all samples from the 10 classes in CIFAR10 dataset are considered as known data, and samples from ImageNet and LSUN are selected as unknown samples. In order to have the same image size as known samples, we resized or croped the unknown samples, obtaining the following datasets: ImageNet-crop, ImageNet-resize, LSUN-crop, and LSUN-resize. For each dataset, we consider all their $10,000$ test samples as the unknown samples in the open set. The performance of the method is evaluated using the macro-averaged F1-scores in the 11 classes (10 known classes and 1 unknown), and the results are shown in Table 6.5. We can see that our method

| Model | AUROC scores w.r.t. different Openness Values ($O$) | | | |
|---|---|---|---|---|
| | $O = 0\%$ | $O = 15.98\%$ | $O = 30.72\%$ | $O = 39.69\%$ |
| Influence of the Feature Extractor (before CapsNet) | | | | |
| CapsNet | 0.971 | 0.753 | 0.767 | 0.781 |
| ResNet20 + CapsNet | **0.981** | **0.948** | **0.949** | **0.950** |
| **Improv.** | **+0.020** | **+0.195** | **+0.182** | **+0.169** |
| Influence of CapsNet | | | | |
| ResNet20 + FC | 0.975 | 0.581 | 0.595 | 0.606 |
| ResNet20 + CapsNet | **0.981** | **0.948** | **0.949** | **0.950** |
| **Improv.** | **+0.006** | **+0.367** | **+0.354** | **+0.344** |
| Influence of Dynamic Routing | | | | |
| ResNet20 + CapsNet | 0.981 | 0.948 | 0.949 | 0.950 |
| ResNet20 + CapsNet + DR | **0.982** | **0.952** | **0.954** | **0.955** |
| **Improv.** | **+0.001** | **+0.004** | **+0.005** | **+0.005** |

Table 6.4: Ablation study on the model architecture. We report results for the unknown detection task on SVHN dataset with outliers from CIFAR100. The performance is evaluated by AUROC with different openness values.

outperforms all previous methods under the F1-score on ImageNet-resize, ImageNet-crop, LSUN-crop, and LSUN-resize.

**Ablation Study on the Model Architecture.** In order to verify the contribution of each part of our model, we perform ablation on the relevance of the main model's components: the capsule network CapsNet, and the feature extractor. ResNet20 is selected as the feature extractor for getting shorter training times. We investigate also the impact of different components of CapsNet, in order to understand their importance. We consider four different variations of our model architecture: the model with CapsNet and dynamic routing that does not use the ResNet20 feature extractor, but just a single convolutional layer; ResNet20+CapsNet that includes the residual feature extractor before CapsNet and doesn't use dynamic routing; ResNet20+FC where a fully connected layer replaces CapsNet; and ResNet20+CapsNet+DR that implements dynamic routing in CapsNet. For all CapsNets that do not implement the dynamic routing, we process each capsule by a fully connected layer. For the ablation, we consider the entire SVHN dataset as the closed dataset, and we consider unknown samples from the CIFAR100 for the open dataset.

| Method | ImageNet-crop | ImageNet-resize | LSUN-crop | LSUN-resize |
|---|---|---|---|---|
| Softmax † [191] | 0.639 | 0.653 | 0.642 | 0.647 |
| Openmax † [10] | 0.660 | 0.684 | 0.657 | 0.668 |
| CROSR [226] | 0.721 | 0.735 | 0.720 | 0.749 |
| C2AE ‡ [157] | 0.837 | 0.826 | 0.783 | 0.801 |
| CGDL § [191] | 0.840 | 0.832 | 0.806 | 0.812 |
| RPL § [21] | 0.811 | 0.810 | 0.846 | 0.820 |
| **CVAECapOSR (ours)** | **0.857** | **0.834** | **0.868** | **0.882** |

Table 6.5: Open set recognition results on CIFAR-10 with various outliers added to the test set as unknowns. We evaluate the model using macro-averaged F1-scores on 11 classes (10 from the the test of the CIFAR10, and 1 from various test datasets). For the sake of clarity, we highlight the source of the results used to populate the table: † are provided by [145], ‡ is from [164] and § are the results that we obtained by running the code of the original paper

We then consider different numbers of unknown classes of CIFAR100, leading to different openness values $O$. The results of the ablation analysis are reported in Table 6.4. We can see that the residual network used as a feature extractor in the encoder helps, especially when the openness $O$ of the open set increases. This fact highlights the importance of having already pre-processed features for the CapsNet on the unknown detection problem when openness increases. Furthermore, another emerging fact is the higher representation capability of the CapsNet with respect to a FC layer: as the openness increases, the AUROC improvement increases up to +0.344. This result suggests that capsules are more capable in detecting unknown samples with respect to standard artificial neurons. This fact emerges also from the t-SNE [203] visualization of the latent space, reported in Figure 6.3, where the separation between known and unknown produced by the probabilistic capsules is more evident with respect to the one produced by a standard FC. Finally, from the experiments we see that dynamic routing achieves better performances with respect to not using it, and that the largest boost on using the capsule network is given by the pose transformation.

**Implementation details.** We also investigated the importance of the parameters $\alpha$, $\beta$, $m_k$ and $\gamma$. To this end, we conducted the unknown detection experiment with the ResNet20+CapsNet+DR architecture on SVHN, using CIFAR100 as the open dataset with openness $\mathcal{O} = 30.72\%$. As suggested by the results reported in Table 6.6,

| Contr. Params | $m_k = 5.0$ | $m_k = 10.0$ | $m_k = 20.0$ |
|---|---|---|---|
| $\alpha = 0.5$ | 0.527 | 0.564 | 0.947 |
| $\alpha = 1.0$ | 0.937 | 0.954 | 0.949 |
| $\alpha = 2.0$ | 0.944 | 0.951 | 0.945 |

Table 6.6: AUROC scores for different values of the parameters $\alpha$, $m_k$ in the loss function. Red cells indicate that targets during the learning process overlap at some point, leading to poor results. Green cells indicate no collapse of prior targets, suggesting good values for $\alpha$, and $m_k$.

we set $\alpha = 1.0$ and $m_k = 10.0$ and, finally, we set $\beta = 0.05$ and $\gamma = 1$ empirically.

## 6.6 Conclusion

In this paper, we introduced CVAECapOSR, a model for open set recognition based on CVAE that produces probabilistic capsules as latent representations through the capsule network. We extended the standard framework of CVAEs using multiple gaussian prior distributions rather than just one for all known classes in the closed dataset. Furthermore, targets are set to be learnable in order to cluster knowns inside their target regions. The contrastive term is used to model the otherness for known classes and to keep the target regions to be mutually separated. Experimental results, obtained on several datasets, show the effectiveness and the high performances on unknown detection and open set recognition tasks.

# Chapter 7

# Self-Supervised Learning Through Spatial Relations

## 7.1 Introduction

[1]Vision Transformer (ViT) [43] is a model that has been recently developed to address computer vision tasks, such as image classification and object detection. It builds on the Transformer architecture [205], the state of the art in natural language processing, considering patches as parts of the image such as words are parts of a sentence. Although ViT is a valid convolution neural networks (CNNs) competitor, showing comparable and even better results [166], a high-level performance is obtained only when training the model on huge amounts of data. To deal with this issue, aiming at better generalization levels, some recent works modified the attention backbone of ViT introducing hierarchical feature representation [131], progressively tokenization of the image [227] or a shrinking pyramid backbone [214]. Other works, inspired by BERT [37], used self-supervised learning (SSL) paradigm firstly pre-training ViT on a massive amount of unlabelled data and, subsequently, training a linear classifier over the frozen feature of the model or fine-tuning the pre-trained model to a downstream task [18, 25]. These variants outperform similar-size ResNet both trained on

[1]**G. Camporese**, E. Izzo, L. Ballan. "Where are my Neighbors? Exploiting Patches Relations in Self-Supervised Vision Transformer", *The British Machine Vision Conference (BMVC 2022)*.

ImageNet, however experiments carried out on smaller datasets are still limited.

In this paper, we investigate ViTs on small datasets introducing SSL tasks, easily integrated in the original architecture, that face the problem of learning spatial relations among couples of patches. In particular: *(i)* We propose and investigate various SSL tasks based on image patches, naturally used by ViTs, showing the improvement optimizing all input tokens, not only the classification one; *(ii)* We show that our SSL tasks are beneficial for the model under two different settings: when the model is pre-trained from scratch on our SSL tasks and subsequently fine-tuned on classification, and when the model is jointly trained from scratch for classification and SSL tasks; *(iii)* We show that on small datasets, both ViT and its variants jointly trained from scratch on our SSL tasks reach better performance with respect to similar state-of-the-art models and overcome the same architectures trained with supervision.

## 7.2 Related Work

Our work is related to recent efforts to move beyond fully supervised training of vision models by exploiting alternate self-supervised strategies and pre-training tasks. We focus our investigation on the recently proposed Vision Transformer and propose several self-supervised strategies that create useful internal representations for downstream tasks.

**Self-Supervised Learning in Vision.** Semi-supervised learning methods for image classification shared a common workflow that foresees the definition of a pretext task on unlabelled images and the subsequent evaluation of the learned features through the training of a linear classifier over the frozen representations or by the fine-tuning of the pre-trained model to a downstream task. Previous works on SSL investigate discriminative strategies, for instance, classification [22, 44, 82, 24, 222] where each image has its own different class, and the model is trained for discriminating them up to data perturbations. However, such modeling frameworks do not scale with

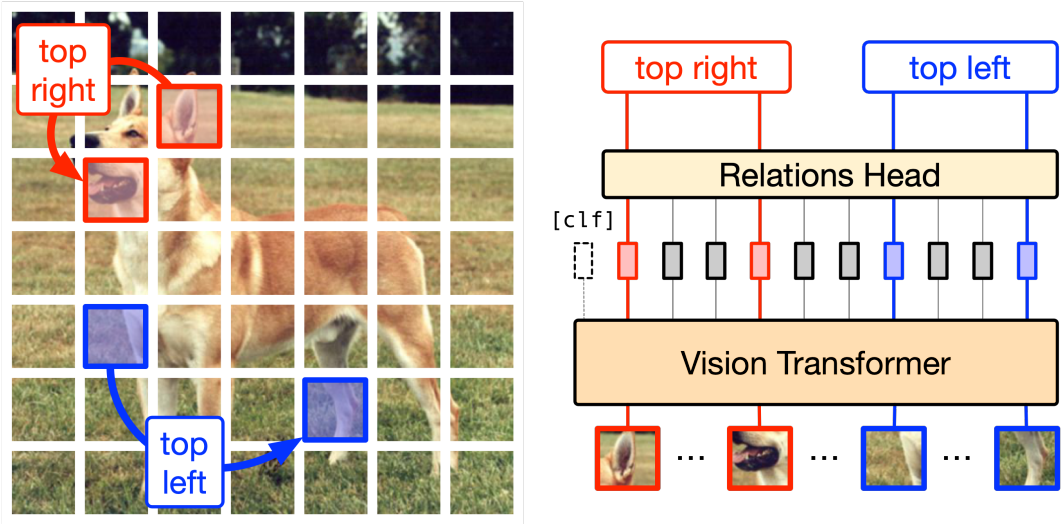



Figure 7.1: ViT splits the image into patches, and only the output classifier token that aggregates the global information of the image is directly trained with the label annotation. On the other hand, all the other tokens related to the input patches of the image are not used. Our work proposes several self-supervised tasks based on image patches used by ViT that can be beneficial for building strong internal features of the model. An example of such a task is the recognition of spatial relations among all the patches (in the picture above, only two relations examples are reported), whereas other tasks are described in Section 7.3.

the increasing number of input samples as the model is trained to discriminate between all the images. Other works on self-supervised learning focus on recognizing known perturbation functions on augmented images [44, 73], by using image patches for predicting their relative positions [38], learning perturbation functions over shuffled image patches [153], counting features [154], image colorization [234], denoising autoencoders, and context encoders [206, 161, 235]. Other recent ideas focus on contrastive methods [80] where the model is trained to attract positive sample pairs and at the same time to repulse negative sample pairs [222, 225, 202, 88, 84]. More recent methods explore siamese architectures [22, 18, 17, 25] where the representation of the student branch is compared and pushed towards the representation of the teacher that sees a different view of the input image.

**Vision Transformers.** Transformers [205] are architectures designed for processing sequential data and, through the attention layers, they enable global dependencies among the input tokens. Originally, the transformer architecture shined on NLP-related tasks raising the hypothesis that even on vision could have potentially been beneficial. Despite the substantial difference between text and images, a recent breakthrough model developed for image classification, dubbed Vision Transformer [43], adopted the transformers for images by treating the image as a sequence of patches like words are considered for phrases. This work highlights the high potential of ViTs when they are trained on large datasets. However, the high performances are not maintained when dealing with small datasets due to the lack of inductive bias of the architecture. As result, in the last few months many ViT variants emerged adapting the transformer architecture for the computer vision tasks. Some works replaced the rectangular backbone characterizing standard ViT with a pyramidal structure [131, 214, 227, 85, 219, 48] obtaining an improvement in accuracy at small/medium scale but increasing complexity and hyperparameters. Other works introduced effective design principals of CNNs into ViT [219, 223, 26] in order to improve performance and robustness.

## 7.3 Our Method

The ViT model takes an image $\boldsymbol{x} \in \mathbb{R}^{C \times H \times W}$ as input, divides it into $N$ patches $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$ of $C$ channels and size $(P, P)$ and makes the output prediction as follows:

$$\boldsymbol{z}_0 = \left[\boldsymbol{x}_{\text{class}}, \boldsymbol{x}_p^1 \boldsymbol{E}, \ldots, \boldsymbol{x}_p^N \boldsymbol{E}\right] + \boldsymbol{E}_{\text{pos}} \tag{7.1}$$

$$\boldsymbol{z}'_\ell = \text{MSA}(\text{LN}(\boldsymbol{z}_{\ell-1})) + \boldsymbol{z}_{\ell-1}, \quad \ell = 1 \ldots L \tag{7.2}$$

$$\boldsymbol{z}_\ell = \text{MSA}(\text{LN}(\boldsymbol{z}'_\ell)) + \boldsymbol{z}'_\ell, \quad \ell = 1 \ldots L \tag{7.3}$$

$$\boldsymbol{y} = \text{LN}(\boldsymbol{z}_L^0) \tag{7.4}$$

where $\boldsymbol{z}_\ell^i$ is the $i$-th token of the $\ell$-th layer, $\boldsymbol{E}$ is the projection matrix of the input patches, $\boldsymbol{E}_{\text{pos}} \in \mathbb{R}^{(N+1)\times D}$ is the learned positional embedding matrix, $\text{MSA}(\cdot)$ is the multi-head self-attention layer [205], and $\text{LN}(\cdot)$ is the layer normalization block [5]. As done in BERT [37], the model uses a randomly initialized classifier token $\boldsymbol{x}_{\text{class}}$ that, at the end, is taken as the reference for making the prediction $\boldsymbol{y}$.

### 7.3.1 Learning From All Tokens

We believe that during the learning process, extending the optimization of the classification token to all the output tokens of ViT could make the model more accurate and the training more efficient.

To this end, inspired by [38, 153, 128], we define a list of tasks over the output tokens $\boldsymbol{z}_L^i$ with $i = 1 \dots N$ of the model: *spatial relations* (**SpRel**), to face the problem of recognizing the spatial relation class of couples of image patches, *distances* (**Dist**) and *angles* (**Ang**), to learn a measure of distance and angle, respectively, between the input patches locations in the original image, and, finally, *absolute positions* (**AbsPos**) for recognizing the 2-d location of the input image patch. More formally, referring to $\boldsymbol{z}_i$ with $i = 1 \dots N$ as the $i$-th token of the $L$-th layer, given a couple of tokens $(\boldsymbol{z}_i, \boldsymbol{z}_j)$ where $\boldsymbol{z}_i$ is in position $\boldsymbol{p}_i = (x_i, y_i)$ and $\boldsymbol{z}_j$ is in position $\boldsymbol{p}_j = (x_j, y_j)$, we let the model to learn the spatial relation $r_{ij}$ defined as:

$$r_{ij} = (s_x, s_y), \quad \text{where} \quad \begin{matrix} s_x \in \{\text{L}, \text{C}, \text{R}\} \\ s_y \in \{\text{T}, \text{C}, \text{B}\} \end{matrix} \tag{7.5}$$

where the partial spatial relations {L, C, R, T, B} stand for {"left", "center", "right", "top", "bottom"}. In practice, we let the model to solve the relations classification task over the set of possible spatial relations $r \in \mathcal{R}$ with $|\mathcal{R}| = 9$.
We define the distance between the patch locations $d_{ij}$ and the relative angle $\alpha_{ij}$ as:

$$d_{ij} = \|\boldsymbol{p}_i - \boldsymbol{p}_j\| \tag{7.6}$$

$$\alpha_{ij} = \cos^{-1}\left(\frac{\boldsymbol{p}_i \cdot \boldsymbol{p}_j}{\|\boldsymbol{p}_i\|\|\boldsymbol{p}_j\| + \epsilon}\right) \tag{7.7}$$

To avoid numerical issues, in Eq. 7.7 we add a small positive constant $\epsilon$ in the denominator and shift $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ by a constant $\boldsymbol{s}$. Afterward, we normalize both $d_{ij}$

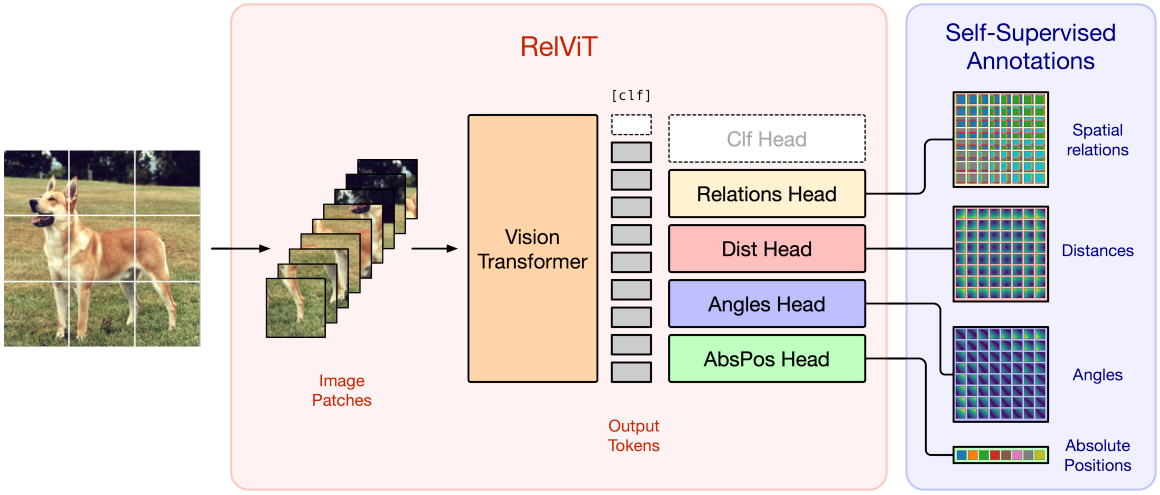



Figure 7.2: Our RelViT model architecture. The image is partitioned into non-overlapping patches and fed to the vision transformer that produces the output tokens used for the self-supervised tasks.

and $\alpha_{ij}$ in $[-1, 1]$ and train the model to learn the normalized values. Finally, we define the absolute position of each token $z_i$ simply as $i$ and we let the model solve the classification task over the absolute positions $\{1, \dots, N\}$.

### 7.3.2 The Relational Vision Transformer

Our model is depicted in Figure 7.2. We aim at training all the output tokens processed by the transformer backbone for all the tasks defined in 7.3.1. Our architecture, dubbed RelViT, uses the standard ViT backbone equipped with specialized heads designed for solving the different tasks which take the output tokens $\boldsymbol{z} \in \mathbb{R}^{N \times D}$ of the transformer encoder as inputs. Specifically, *Relations*, *Dist* and *Angles* heads, which process couple of tokens, are characterized by a MSA layer that produces the attention scores $\phi(\boldsymbol{z}, \boldsymbol{z}) \in \mathbb{R}^{N \times N \times K}$ described as follows:

$$\phi(\boldsymbol{z}, \boldsymbol{z}) = \big[\boldsymbol{a}_1, \dots, \boldsymbol{a}_K\big], \quad \boldsymbol{a}_i = \frac{\boldsymbol{z}\boldsymbol{W}_i\boldsymbol{z}^T}{\sqrt{D}} \tag{7.8}$$

where $\boldsymbol{a}_i \in \mathbb{R}^{N \times N}$ is the attention scores of the single attention head, $\boldsymbol{W}_i \in \mathbb{R}^{D \times D}$ is a learnable projection matrix, $K$ is the output dimension of the head, and $D$ is the

token dimension. Instead, the *AbsPos* head uses a fully connected (FC) layer that produces the logits $\phi(\boldsymbol{z}) \in \mathbb{R}^{N \times K}$ as follows:

$$\phi(\boldsymbol{z}) = \boldsymbol{z}\boldsymbol{W} + \boldsymbol{b} \tag{7.9}$$

where $\boldsymbol{W} \in \mathbb{R}^{D \times K}$ and $\boldsymbol{b} \in \mathbb{R}^{K}$ are respectively the weights and bias of the FC layer. Given the corresponding logits, the SSL losses are computed as in the following equations:

$$\mathcal{L}_{\text{sp-rel}}(\boldsymbol{z}) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \text{CE}\big[\phi_{\text{sp-rel}}(\boldsymbol{z}_i, \boldsymbol{z}_j), r_{ij}\big] \tag{7.10}$$

$$\mathcal{L}_{\text{abs-pos}}(\boldsymbol{z}) = \frac{1}{N} \sum_{i=1}^{N} \text{CE}\big[\phi_{\text{abs-pos}}(\boldsymbol{z}_i), i\big] \tag{7.11}$$

$$\mathcal{L}_{\text{dist}}(\boldsymbol{z}) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \big\|\phi_{\text{dist}}(\boldsymbol{z}_i, \boldsymbol{z}_j) - d_{ij}\big\|^2 \tag{7.12}$$

$$\mathcal{L}_{\text{angle}}(\boldsymbol{z}) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \big\|\phi_{\text{angle}}(\boldsymbol{z}_i, \boldsymbol{z}_j) - \alpha_{ij}\big\|^2 \tag{7.13}$$

where $\text{CE}(\cdot)$ is the cross-entropy with logits loss function. Finally, during training, we minimize the sum of all the losses of the tasks we want to solve. It is worth noticing that with this formulation, for each training step, we optimize all the combinations of couples of patches at the same time in parallel.

### Mega-Patches

Our SSL heads provide the spatial annotations thanks to a self-attention mechanism, where any output token of the ViT encoder interacts with itself and all the others in order to acquire knowledge about the spatial relations among couples of patches. Consequently, we believe that the increase of the number of tokens, obtained from the non-overlapping patches on which the input image is divided, could doubly impact on the global process. First of all, there is the well-know computational effort which

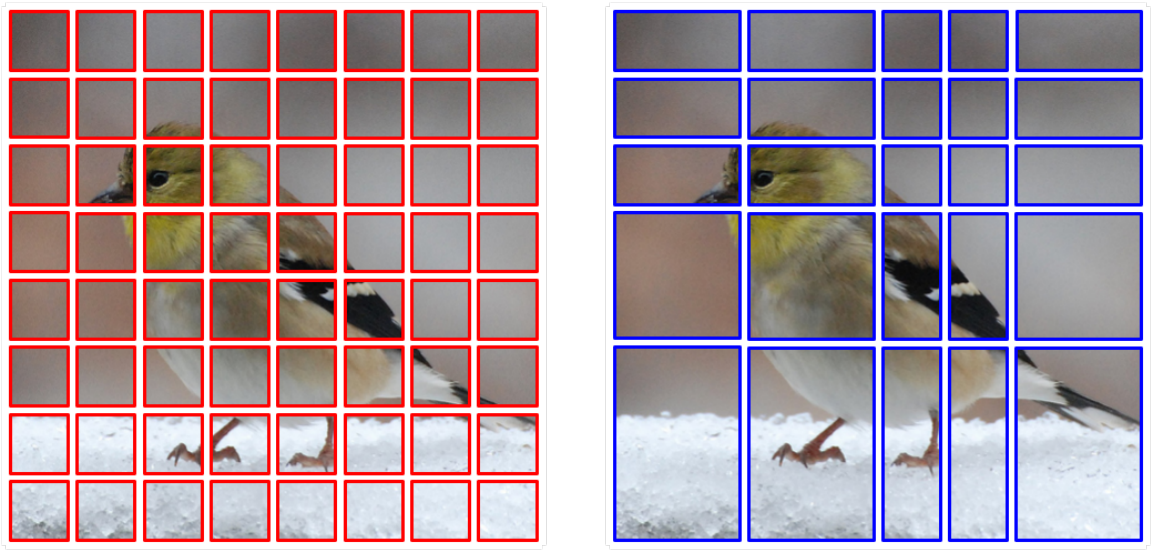


Figure 7.3: Comparison between using standard patches and our proposed mega-patches. On the left, the image is divided into 64 patches, on the right it is randomly divided into 25 mega-patches using $M = 5$.

has a quadratic complexity with respect to the number of tokens. Secondly, a small patch size could reduce the improvement obtained thanks to our SSL approach since the contextual information in a patch is reduced.

Thus, we propose to divide the input image into $M \times M$ mega-patches which are non-overlapping rectangles whose height and width are integer multiplies of the patch size $P$. More specifically, given an image $\boldsymbol{x} \in \mathbb{R}^{C \times H \times W}$, and the number of mega-patch per side $M$, we uniformly sample $M - 1$ random indices that cut the image horizontally and vertically, producing rectangular mega-patches of different dimensions that are resized to $(P, P)$ before feeding them to our RelViT model. A qualitative difference between patches and mega-patches of an image is illustrated in Figure 7.3.

## 7.4 Experiments

We aim at demonstrating the effectiveness of learning Visual Transformers via a number of self-supervised tasks whose benefits are shown on both ViT and two of its

| Model | Img Res. | Blocks | Heads | Dim | #Tokens | #Params |
|---|---|---|---|---|---|---|
| ViT-S/4 | (32, 32) | 12 | 6 | 384 | 65 | 21.3M |
| ViT-S/8 | (64, 64) | 12 | 6 | 384 | 65 | 21.4M |
| ViT-S/32 | (224, 224) | 12 | 6 | 384 | 50 | 22.5M |
| ViT-S/16 | (224,224) | 12 | 6 | 384 | 197 | 22.5M |
| ViT-S/14 | (224,224) | 12 | 6 | 384 | 257 | 22.5M |

Table 7.1: The used ViTs model configurations; *Img Res.* is the image resolution, *Blocks* and *Heads* are the number of layers in the backbone and of heads in the multi-head self attention layer, *Dim* is the dimension of the token in the transformer encoder, *#Tokens* is the number of patches coming from the image and *#Params* is the number of learnable parameters of the model.

variants: Swin [131] and T2T-ViT [227].

### 7.4.1 Datasets and Implementation Details

**Datasets and RelViT Configurations.** We conducted experiments on standard image benchmarks for classification focusing on relatively small datasets providing a simple and effective way of boosting ViTs performance. In particular, we tested RelViT on CIFAR-10 [113], SVHN [147], CIFAR-100 [113], Flower-102 [151], TinyImagenet [119] and ImageNet-100 (a subset of 100 labels from ImageNet), which have a number of train samples from 1k for Flower-102 to 130k for ImageNet-100. We used ViT-S backbone configuration [18] (model details in Table 7.1), while the experiments on ViT variants use Swin-T and T2T-ViT-14 backbones with the same model configurations as in [128].

**Training Details.** We developed our model in PyTorch and the code will be available upon acceptance. For all the experiments, we use the same configurations of the training parameters, except if differently mentioned. All the models are trained for 100 epochs both in the pre-training, fine-tuning and in the downstream-only experiments. Following [18], we use the AdamW [135] optimizer with a cosine learning rate scheduling, linear warm-up of 10 epochs, and $lr = 0.0005$ as the maximum value for the learning rate. We found that RelViT does not need regularization, so we avoid the use of dropout, drop-path, weight decay, and label smoothing. The batch size is set to

| Dataset | Backbone | ViT | RelViT | Improv. |
|---|---|---|---|---|
| CIFAR-10 | ViT-S/4 | 85.38 | **90.13** | **+4.75** |
| SVHN | ViT-S/4 | 96.07 | **97.15** | **+1.08** |
| CIFAR-100 | ViT-S/4 | 58.01 | **64.46** | **+6.45** |
| Flower-102 | ViT-S/32 | 41.67 | **45.49** | **+3.82** |
| TinyImagenet | ViT-S/8 | 42.09 | **52.10** | **+10.01** |
| ImageNet-100 | ViT-S/32 | 58.04 | **67.02** | **+8.98** |

Table 7.2: Comparison between our RelVit model and the supervised ViT baselines on several small datasets. RelVit is pre-trained on our SSL tasks and subsequently finetuned for classification, whereas the baselines are trained on classification.

256 for ViT backbone and 64 for its variants, and we use a single GPU for performing each experiment. The data augmentation for the classification problem follows the standard practice of resized crops and random horizontal flipping. Instead, when using ViT backbone and SSL tasks, each image patch is randomly resized-cropped, and following [25] a color shift with color-jittering is applied as well as random gray-scale perturbation.

### 7.4.2 Experimental Results

We investigated our RelViT model under two different scenarios: pre-training the model from scratch for our SSL tasks and subsequently fine-tuning on classification (from now on referred as to *pre-training* and *fine-tuning*, respectively) and training the model from scratch jointly on classification and SSL tasks (*downstream-only*). We used the spatial relations and the absolute positions tasks in both pre-training and downstream-only (more details in paragraph 7.4.3) and we removed the positional encodings in the pre-training, while they are used as aid in the classification task during both fine-tuning and downstream-only.

Table 7.2 shows RelViT accuracy after a supervised fine-tuning reporting *Supervised* method, obtained training the model only on classification under the same settings, as reference. Our approach leads to higher accuracy than the supervised one in all datasets, gaining +10.01% on Tiny-Imagenet, +6.45% on CIFAR-100 and +4.75%

| Backbone | Method | CIFAR-10 | SVHN | CIFAR-100 | Flower-102 |
|---|---|---|---|---|---|
| | | ViT-S/4 | ViT-S/4 | ViT-S/4 | ViT-S/32 |
| ViT [43] | Supervised | 85.18 | **96.02** | 58.11 | 40.59 |
| | **RelViT ours** | **88.98** | 95.86 | **62.04** | **44.61** |
| | **Improv.** | **+3.8** | **-0.16** | **+3.93** | **+4.02** |
| Swin [131] | Supervised [128] | 59.47 | 71.6 | 53.28 | 34.51 |
| | Swin+$\mathcal{L}_{drloc}$ [128] | 83.89 | 94.23 | 66.23 | 39.37 |
| | **RelSwin ours** | **92.17** | **96.55** | **69.97** | **59.22** |
| | **Improv.** | **+8.28** | **+2.32** | **+3.74** | **+19.85** |
| T2T-ViT [227] | Supervised [128] | 84.19 | 95.36 | 65.16 | 31.73 |
| | T2T-ViT+$\mathcal{L}_{drloc}$ [128] | 87.56 | 96.49 | **68.03** | 34.35 |
| | **RelT2T-ViT ours** | **91.08** | **96.52** | 66.27 | **50.59** |
| | **Improv.** | **+3.52** | **+0.03** | **-1.76** | **+16.24** |

Table 7.3: Comparison of our method with other works on multiple backbones. The results are obtained training the model from scratch jointly on our SSL tasks and classification.

on CIFAR-10. The first row in Table 7.3 shows RelViT results on classification with downstream-only (using ViT as backbone). Also this configuration improves the results on almost all the datasets up to +4.02% on Flower102, +3.93% on CIFAR-100, and +3.71% on CIFAR-10. The improvements obtained in both scenarios demonstrate the effectiveness of our self-supervised tasks for image classification. Finally, we investigated how our SSL tasks generalize across different backbones. We chose Swin [131] and T2T-ViT [227] to compare our method with [128] that shares a similar setting. As reported in Table 7.3, we outperform the supervised baselines and also results reported in [128] up to +19.85% and +16.24% on Flower-102, with Swin and T2T-ViT backbones, respectively.

### 7.4.3 Ablations

**SSL Multi-Task Ablation.** We defined multiple SSL tasks over the image tokens and evaluated each combination of tasks in order to highlight both the importance of each sub-problem and the relations among them.

As shown in Table 7.4, that displays our results, solving, individually, each SSL

| Method | AbsPos | Ang | Dist | SpRel | Finetuning |
|---|---|---|---|---|---|
| | Acc. ↑ | MSE ↓ | MSE ↓ | Acc. ↑ | Acc. ↑ |
| Supervised | ✗ | ✗ | ✗ | ✗ | 85.38 |
| RelViT | ✗ | ✗ | ✗ | 48.53 | 89.02 |
| RelViT | ✗ | ✗ | 10.54 | ✗ | 89.27 |
| RelViT | ✗ | 13.08 | ✗ | ✗ | 88.75 |
| RelViT | 15.24 | ✗ | ✗ | ✗ | 89.39 |
| RelViT | ✗ | ✗ | 10.29 | 48.91 | 89.24 |
| RelViT | ✗ | 12.55 | ✗ | 48.66 | 90.12 |
| RelViT | 15.73 | ✗ | ✗ | 48.53 | **90.13** |
| RelViT | ✗ | 12.58 | 10.32 | ✗ | 89.34 |
| RelViT | 16.03 | ✗ | 10.49 | ✗ | 89.75 |
| RelViT | 15.51 | 12.82 | ✗ | ✗ | 89.90 |
| RelViT | ✗ | 12.49 | 10.28 | 48.98 | 88.81 |
| RelViT | 15.90 | ✗ | 10.39 | 48.67 | 89.80 |
| RelViT | 16.03 | 12.52 | ✗ | 48.80 | 89.73 |
| RelViT | 16.06 | 12.65 | 10.50 | ✗ | 89.67 |
| RelViT | 16.07 | 12.54 | 10.39 | 48.74 | 90.06 |

Table 7.4: Results using our SSL tasks on CIFAR-10 with ViT-S/4 backbone. The columns from the 2$^{\text{nd}}$ to the 6$^{\text{th}}$ report the performance on the SSL tasks whereas the last column is the accuracy of the fine-tuned models.

task leads to similar fine-tuning accuracy, improving the performance w.r.t the supervised baseline of +3.00% on average. Among SSL tasks, the *SpRel* one is the most performing since if we rank the combinations by the fine-tuning accuracy of the fine-tuned model, it is present in the top-3 entries, suggesting its importance for the fine-tuning classification task. Moreover, we find that the optimal combination of tasks is obtained by learning the *SpRel* and the *AbsPos* of the patches.

**Positional Embeddings and Patches Permutations.** We investigated the shortcut learning problem in RelViT using the positional embeddings (PEs) during the self-supervised pre-training. Moreover, to avoid trivial solutions and break the dependency between PEs and SSL labels, we investigated the shuffling of the input patches before adding the PEs, permuting the rows and columns of the label matrix with the same permutation used for shuffling the input tokens. Since we aim at focusing on the role of the PEs and the patch shuffling, for simplicity we used only the *SpRel* learning on the pre-training. As shown in Figure 7.4, if the model uses the PEs without shuffling during the pre-training, it solves the task with 100% of *SpRel* accuracy but the corresponding fine-tuned model reaches a classification accuracy comparable to a randomly initialized network (red dashed line in Fig. 7.4). Even though the shuffling removes the trivial solution and improves the classification accuracy, we removed PEs from RelViT since, in this way, we obtained the best results. Whereupon, the shuffling is optional due to the permutation equivariance property of the model.

**Inspecting RelViT.** In Figure 7.5 and 7.6 we reported some visualizations related to the RelViT respectively after the pre-training and after the fine-tuning. Specifically, in Figure 7.5 are depicted the most important filters learned during the pre-training strategy by the linear projection layer that processes the input image patches. As you can see, the kernels learned interesting structures such as vertical and horizontal edges and some complicated patterns among color channels. Another interesting fact we observe after pre-training and fine-tuning the RelViT is in the visualization of the similarity of the PEs. As depicted in Figure 7.6, the PEs learned by the ViT baseline during the training for classification have some artifacts as the

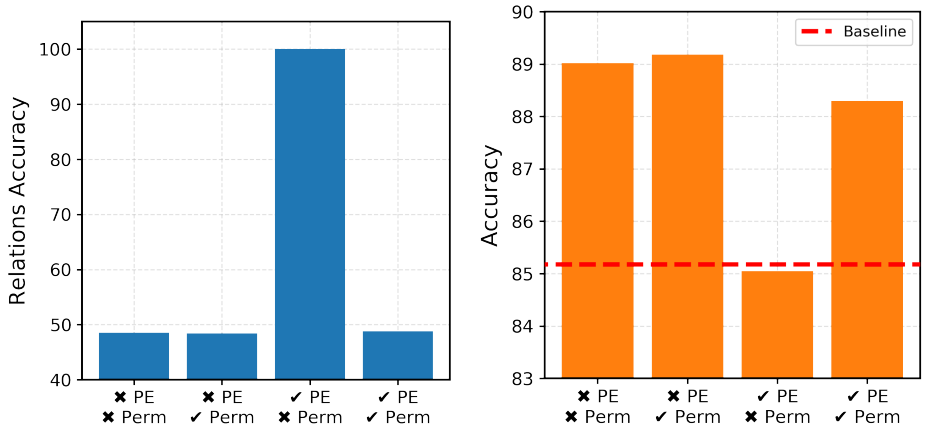



Figure 7.4: The role of the positional embedding (PE) and random permutations (Perm) of the input patches in RelViT during the SSL Pre-Training (left) and the Fine-Tuning (right).

encodings at the boundary of the opposite side of the image can be exchanged by the model. On the other hand, our pre-training and fine-tuning strategy lead to more regular and smooth PEs configurations as the similarity is shared only among neighbor encodings.

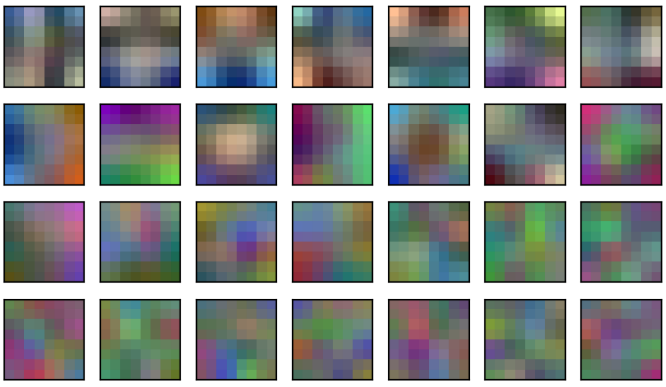


Figure 7.5: RGB embedding filters (first 28 principal components) of our RelViT trained using *SpRel* and *AbsPos* on the CIFAR-10 dataset.

**Mega-Patches Ablation.** We investigated our RelViT model using the mega-patches approach pre-training the model from scratch using mega-patches for the spatial relations and the absolute positions tasks and subsequently fine-tuning on

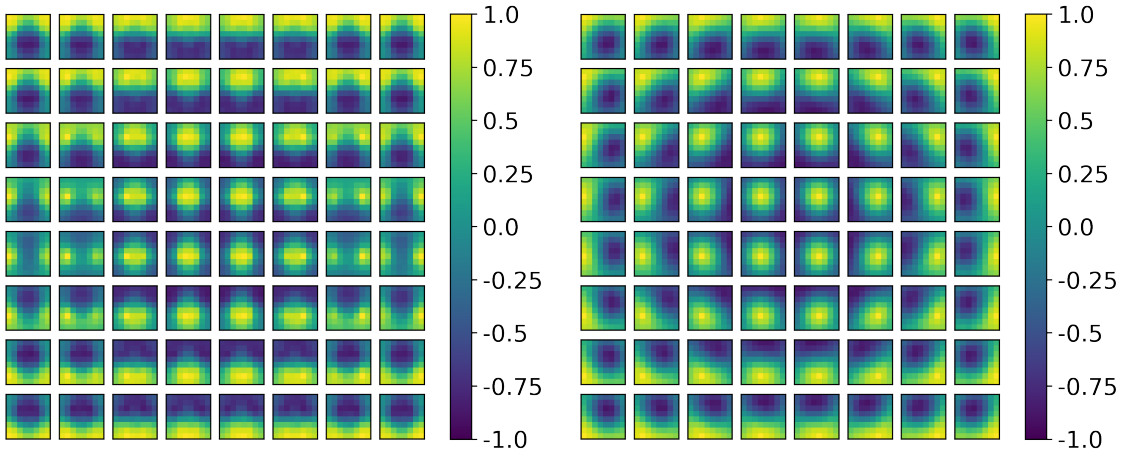



Figure 7.6: Comparison between cosine similarity of the positional encodings learned by ViT (on the left) and RelViT (on the right). ViT is both pre-trained and fine-tuned on classification, whereas RelViT is pretrained using *SpRel* and *AbsPos* and fine-tuned on classification.

classification using the standard patches. Table 7.5 shows RelViT accuracy after a supervised fine-tuning on ImageNet-100 and Flower-102 using three different backbones: ViT-S/32, ViT-S/16 and ViT-S/14 which differ in the patch size. *Supervised* and *No-Mega-Patches* are reported as references. *Supervised* is obtained training only on classification, whereas *No-Mega-Patches* is obtained pre-training for the spatial relations and the absolute positions without mega-patches and fine-tuning on classification. We tested several configurations by varying the parameter $M \in [3, 10]$. We observe that our RelViT model always outperforms the supervised baselines regardless of the value of the parameter $M$ underlining the effectiveness of our RelViT model. Moreover, comparing the results obtained with and without the use of the mega-patches, as expected, the mega-patches approach improves the accuracy levels as much as the patch size of the backbone is reduced suggesting that our SSL taks need more contextual information in each patch to express their potential.

| # Mega-Patches | ImageNet-100 | | | Flower-102 | | |
|---|---|---|---|---|---|---|
| | ViT-S/32 | ViT-S/16 | ViT-S/14 | ViT-S/32 | ViT-S/16 | ViT-S/14 |
| Supervised | 58.04 | 69.42 | 71.14 | 41.67 | 46.37 | 45.39 |
| No-Mega-Patches | 67.02 | 74.74 | 75.76 | **45.49** | 49.31 | 50.98 |
| $M = 10$ | - | 75.30 | 76.10 | - | 48.43 | 49.61 |
| $M = 8$ | - | 74.36 | 76.42 | - | 46.76 | **52.25** |
| $M = 6$ | **67.26** | **75.36** | **76.60** | 45.39 | 49.61 | 51.67 |
| $M = 5$ | 67.14 | 75.26 | 75.58 | 44.22 | 48.63 | 51.37 |
| $M = 4$ | 66.02 | 74.26 | 74.24 | 44.31 | 49.31 | 50.39 |
| $M = 3$ | 64.38 | 73.16 | 74.12 | 43.43 | **50.78** | 50.10 |

Table 7.5: Comparison between our RelViT model using our mega-patches approach and the supervised ViT baselines. The results are obtained pre-training the model from scratch for on SSL tasks using the mega-patches and then fine-tuning on the standard patches. $M$ is the number of mega-patches per side (height or width) used during the SSL pre-training.

## 7.5 Conclusion

In this work, we have proposed and investigated several self-supervised tasks defined over image patches that are naturally used by ViT. Our proposed methods are straightforward to implement and effective as the RelViT shows significant improvement on all the tested datasets and in both scenarios: pre-training on SSL tasks and then fine-tuning and training jointly on SSL and SL tasks. All the reported experiments focused on relatively small datasets, showing a simple way of making ViTs stronger even on the small dataset regime.

# Chapter 8

# Speech Recognition for Speech Disfluencies

## 8.1 Introduction

[1]Human speech typically contains disfluencies alongside the articulation of an intended word sequence. Speech from any speaker has filled pauses, partial words and repetitions, while certain speech disorders (e.g. stuttering) amplify these phenomena. And despite the general performance achievements in speech recognition using End-to-End (E2E) models, there is still not enough robustness to them. The main objective of this work is to investigate training data filtering and transcription processing choices for an ASR system based on the Recurrent Neural Network Transducer (RNN-T)[78], which may improve its robustness to speech disfluencies with a special focus on the partial words and repetitions.

Motivation for this work comes from discussions with our colleagues who reported that self corrections are responsible for significant share of entity resolution errors in several voice assistant use-cases. Also, we wanted to investigate if ASR robustness to disfluencies may be improved by overweighting the data with partial words in the

[1]V. Mendelev, T. Raissi, **G. Camporese**, M. Giollo. "Improved Robustness to Disfluencies in RNN-Transducer Based Speech Recognition", *International Conference on Acoustics, Speech and Signal Processing, Toronto, Canada, 2021. (ICASSP 2021)*. Work done while interning at Amazon Alexa AI.

training set and if this gives a higher ASR accuracy for speakers with stuttering, even though vast majority of those data came from fluent speakers.

Main contributions of this work are: a study of the effect of different ways to represent partial words in transcripts used to train RNN-T system, experimental results with different fractions of data with disfluencies in the training set and a view of the influence of the factors mentioned before on ASR performance for speakers with stuttering.

In the next session an overview of the prior work is presented, then the datasets are described in Sec. 8.3. The experimental setting and results are discussed in Sec. 8.4 which is followed by conclusions and future work.

## 8.2 Prior work

The initial attention of the research community towards speech disfluencies derives from the importance of the improvement of the ASR system accuracy not only for the speech signal which is recorded under controlled conditions but also for the spontaneous speech [181, 83]. The resulting task comprises the identification and the consecutive removal of the disfluency events in the recognizer output and is solved by using the *noisy channel* approach [98, 91]. Following the Bayesian statistical framework, this entails maximization of the a-posteriori probability of the word sequence with disfluencies, given the originally intended word sequence. Since disfluency events affect different phonetic aspects of speech [180] many researchers tried to take advantage of the possible combination of different sources of knowledge on both acoustic and language model sides [129, 58, 228, 2]. The disfluency detection task in all mentioned works is solved by using sequence labelling/tagging approaches which can rely either on a generative approach such as Hidden Markov Model or discriminative log-linear models, such as Maximum Entropy Markov Model or Conditional Random Fields as well as Bidirectional Long Short-Term Memory based networks in combination with an attention mechanism [6]. In most of the cases the overall system maintains its modular setting and therefore requires not only separate optimization criteria for

different components but also in some cases hand-labeled features for the annotation of the disfluencies to train the language model. A recent work brings the focus on the acoustic side and does not take into consideration any language-dependent information [112]. To the best of the authors' knowledge, with the exception of a work on personalized ASR for dysarthric speech [179], none of the published papers are aimed to improve speech recognition accuracy of an E2E ASR system by dealing with disfluencies without solving disfluency detection task itself.

## 8.3 Datasets

For our experiments we used subsets of the transcribed data pool available to train Alexa ASR models. The recordings comprising the data pool are anonimized voice assistant requests recorded with various far-field devices in compliance with terms of service. Each transcription, in addition to the spoken words, contains tags provided by the transcriber indicating additional information on the speech signal. The attribution of the described tags relies on the transcriber's perception and expertise and therefore can be source of possible inaccuracy for both tag and spoken word annotations. This aspect is especially valid when the utterance contains unintelligible or disfluent speech. Most disfluency events such as word or syllable prolongation are actually not marked in the transcriptions.

In this work we use three datasets, which we call *Ordinary*, *Disfluencies* and *Stutter*.

The *Ordinary* dataset contains ordinary utterances with intelligible device-directed speech but without partial words. Acoustic conditions may be challenging because of low signal-to-noise ratio, media speech or due to the presence of multiple speakers.

The *Disfluencies* dataset is derived by applying a set of filters, which operate on transcriptions level on the large pool of data. The filters aim to select challenging utterances with partial words, repetitions and hesitations. More specifically, an utterance is included into this dataset if its transcription contains a partial word and its subsequent completion (e.g. 'alarm on tw- on twelve') and at least one of the following conditions is true:

- there are no more than 4 words in the transcription;
- there is at least one other partial word (not necessarily with completion);
- there are hesitations;
- there are repetitions.

This set of filters was chosen after trying several alternatives and observing that without additional conditions the dataset contained a lot of utterances with a single partial word, which were considered not challenging enough. We have to note that after most of the experiments mentioned in this work were done we repeated some of them with the simplest filter, which accepted utterances with a single partial word in the transcript. We found that conclusions reported in the following sections were mostly valid for datasets derived with this simple filter as well.

The Ordinary and Disfluencies datasets include train, dev and test partitions, which do not have speaker overlap.

The *Stutter* dataset was recorded by a vendor and contains speech samples provided by 11 speakers with stuttering. The speakers were reading prompts containing possible requests to a voice assistant in a quiet acoustic environment. This dataset is used for the evaluation purpose only.

Size of the datasets is presented in Table 8.1. We restricted the amount of training data in the Ordinary dataset to have faster turnaround time. Also, the data for the Disfluencies dataset were selected from an order of magnitude bigger data pool in comparison to the Ordinary Train to enable experiments with increased relative amount of challenging data in the training sets.

| **Dataset** | **Train (hours)** | **Test (hours)** |
|---|---|---|
| Ordinary | $\sim 2300$ | $> 20$ |
| Disfluencies | 47 | 5 |
| Stutter | - | 2 |

Table 8.1: Size of the datasets used in this work in hours of sound.

### 8.3.1 Handling Partial Words in Transcriptions

Once data with partial words are included in the training set, there are several options how to mark such words. In this work we assume that our goal is to have ASR output free from disfluencies. This can be achieved if the system: (1) ignores them, (2) outputs a label instead of the partial word, (3) concatenates a partial label with the disfluency content which later can be removed via the post-processing (e.g. for the recording with the reference transcript 'p- play' the system may output (1) 'play', (2) '⟨pw⟩ play', (3) 'p⟨pw⟩ play'). We decided to test the 3 options mentioned plus the one where only the first letter of the partial word remains and is appended with ⟨pw⟩ on the right. The motivation behind the last two options is clear: ideally we would like to keep disfluency content in order to preserve more information for the downstream tasks. Still, the quality of the ASR output with disfluencies removed is considered as the main criterion in the current work.

## 8.4 Experimental Setup and Results

### 8.4.1 Experimental Setting

We train models suitable for on-line recognition. The model consist of a 5 layers deep encoder, a 2 layers deep prediction network, a joint network as in [79], and an output layer with a softmax nonlinearity. Each layer of the encoder and the prediction network comprises 1024 Long Short-Term Memory [90] units. The size of the joint network layer is 512 and the output layer size is 4001 corresponding to 4000 wordpeices and a *blank* symbol. The wordpiece model was trained on a large set of voice assistant requests using a unigram language model [114].

The model accepts 192 dimensional input feature vectors each comprising three 64 dimensional Log-Mel-Filterbanks extracted every 10 milliseconds and stacked together.

Training objective is minimization of RNN-T loss function [78, 79] with Adam optimizer [104], with total batch size of 1536 utterances and warmup-hold-decay learning rate schedule. We also use SpecAugment [159].

**REF:** **play devotional music hindu dev- devotional music**

CLEAN: play devotional music hindu devil devotional music

DISFL: play devotional music hindu devotional music

Figure 8.1: Example recognition results. *REF* denotes the reference transcription, *CLEAN* was produced by the baseline model, *DISFL* – by the model trained on Ordinary Train merged with Disfluencies Train and with partial words removed from training transcripts.

Evaluations were done using a beam decoding with the beam size 16. Model specific post-processing was applied on both the hypothesis and the reference in order to get transcripts free from partial words. For models trained with replacement of the full partial word with ⟨pw⟩ tag, those were removed. For models where a partial word or its first letter persisted in the training transcript, the tag was removed together with all letters before first space to the left of it.

### 8.4.2 Case Studies

After training the baseline model on the Ordinary Train, the experimental models on the Ordinary Train merged with the Disfluencies Train datasets and different handling of partial words in the transcripts, we looked into the decoding results of Disfluencies Test in order to ensure that the models behave as we expect. Indeed, we observed that the baseline model produces quite a lot of insertions, while the one trained with Disfluencies Train and partial words removed does not. Example transcripts are depicted in Fig. 8.1.

The additional examples are presented in Fig. 8.2 including those derived with the model trained with the replacement of partial words by ⟨pw⟩ in transcripts for training. As one would expect, the model trained with ⟨pw⟩ produces reasonable 'alignments' in some cases capturing the amount of partial words uttered as in Fig. 8.2b-c, while not so reasonable in the others (Fig. 8.2a). In some cases (Fig. 8.2d) this model produces a better result than the one with partial words removed without outputting

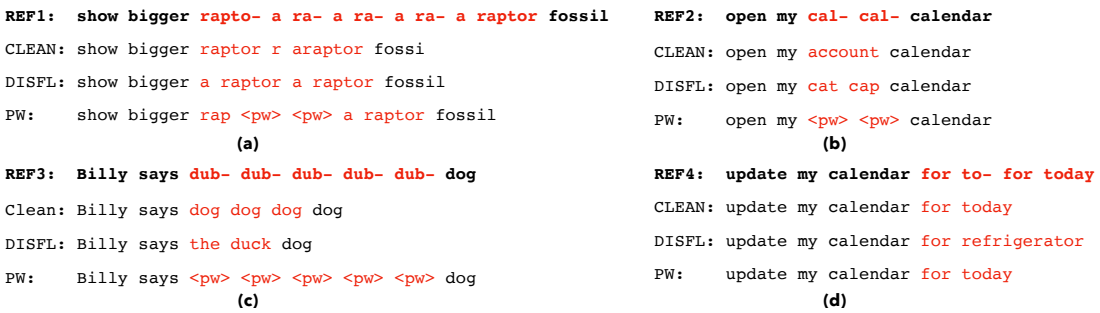



Figure 8.2: Example recognition results. *REF* denotes the reference transcription, *CLEAN* was produced by the baseline model, *DISFL* – by the model trained on Ordinary Train merged with Disfluencies Train and with partial words removed from training transcripts, *PW* – same as *DISFL* but with partial words replaced by a tag.

the ⟨pw⟩ tag.

One can speculate that mapping all partial words to a single tag allows the model to capture acoustic and lingustic patterns associated with partial word appearance and to preserve the integrity of the 'normal' speech patterns which would not happen if partial words were removed from the transcripts.

### 8.4.3 Word Error Rates

In Table 8.2 one can find word error rates for different models trained. The Ordinary Test results are provided to make sure that while improving on data with disfluencies we don't have degradation on this dataset. As expected, the error on the test sets with disfluencies is more than 2 times higher than on Ordinary Test and the baseline model produced the highest number of insertions on all three test sets. By adding Disfluencies Train with removed partial words we achieved 21% and 14% relative WER reduction on the the test sets with disfluencies in comparison to the baseline. If partial words are replaced by a tag, we see a modest additional reduction by 1.7% and 3.9%. When we try to preserve all or some characters of a disfluency, the reduction is much smaller (lines 4 and 5 in Table 8.2). It may seem surprising that WER on Stutter Test is significantly lower, than on Disfluencies Test. This is explained by the nature of the data: the former was recorded in a quiet room with a limited set of popular prompts, while the latter contains challenging field data.

| # | Ordinary Train and... | Partial words are ... | Ordinary Test | | | | | Disfluencies Test | | | | | Stutter Test | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | NWER | WERR | S | I | D | NWER | WERR | S | I | D | NWER | WERR | S | I | D |
| 1 | - | absent | 1 | 0.0 | 60 | 18 | 23 | 3.02 | 0.0 | 29 | 57 | 13 | 2.35 | 0.0 | 34 | 49 | 17 |
| 2 | | deleted | 1 | 0.2 | 59 | 18 | 23 | 2.38 | 21.0 | 33 | 47 | 20 | 2.02 | 14.0 | 40 | 39 | 21 |
| 3 | Disfluencies | replaced by ⟨pw⟩ | 1 | 0.5 | 59 | 17 | 24 | **2.34** | **22.4** | 33 | 45 | 22 | **1.93** | **17.9** | 39 | 34 | 29 |
| 4 | Train | appended by ⟨pw⟩ | 1 | 0.2 | 59 | 18 | 23 | 2.6 | 13.7 | 31 | 51 | 18 | 2.33 | 0.9 | 37 | 45 | 18 |
| 5 | | replaced by 1st letter &⟨pw⟩ | 1 | 0.4 | 59 | 17 | 24 | 2.49 | 17.4 | 31 | 49 | 20 | 2.1 | 10.5 | 36 | 40 | 23 |

Table 8.2: Evaluation results on different test sets depending on partial words handling in transcripts. Model-specific post-processing was applied before evaluation to get rid of partial words, or ⟨pw⟩ tag. Partial words were removed from reference transcriptions as well. *NWER* column contains the corresponding model WER divided by WER of the baseline model on Ordinary Test. *WERR* (%) is $100 * (y - x)/y$ where $x$ is the corresponding model WER and $y$ is WER of the baseline model on the same test set. *S, I, D* columns contain shares (%) of substitutions, insertions, deletions in the observed WER.

In order to verify reproducibility of the observed effects and to investigate how much data with disfluencies is actually needed to improve WER for speakers with stuttering, we conducted additional experiments. We used only the training sets where partial words were removed or replaced by ⟨pw⟩ because the corresponding models demonstrated much lower WER values than the others. Trainings with 3 different seeds were performed for each dataset configuration, then each model was evaluated and the WER numbers were averaged between the runs. The results are summarized in Table 8.3. The models corresponding to lines 4 and 5 give lower WER on the Disfluencies and Stutter datasets than the model number 6, which confirms benefits from using a tag to denote all partial words in transcripts. Another observation is that as the amount of data with disfluencies increased, there was a gradual decrease in WER on Disfluencies and Stutter datasets with saturation more pronounced for the latter one. Probably in order to further improve accuracy one needs to take into account additional aspects associated with stuttering which we don't pay attention to.

We emphasise that the natural frequency of partial words appearing in the dataset available to us is rather low due to the heavy head of the requests distribution, so even 1/4 of the Disfluencies Train should be considered as oversampling (it constitutes about 0.5% of Ordinary Train size).

## 8.5 Conclusions

In this work we showed that RNN-T based speech recognition models tend to produce insertions when presented with speech containing partial words if data with such words were not included in the training set. This contributes to low recognition accuracy for speakers with a stuttering disorder. Adding the data with partial words to the training set and increasing their relative share leads to significant WER reduction on the test sets with disfluencies without accuracy degradation on the average data. Replacing partial words in transcripts with a tag for training allows to reach even lower WER. Relative to the baseline the best model configuration allowed to achieve 22% reduction on the test with disfluencies and 16% on the test containing stuttering speech.

| # | **Ordinary Train and ...** | **Partial words are ...** | **Ordin. Test** | **Disfl. Test** | **Stutter Test** |
|---|---|---|---|---|---|
| 1 | - | absent | 0.0 | 0.0 | 0.0 |
| 2 | 1/10 Disfl. | | 0.3 | 8.7 | 5.8 |
| 3 | 1/4 Dislf. | replaced by | 0.2 | 13.9 | 6.6 |
| 4 | 1/2 Disfl. | ⟨pw⟩ | 0.0 | 18.3 | 14.5 |
| 5 | Full Disfl. | | 0.0 | **22.5** | **16.4** |
| 6 | Full Disfl. | deleted | -0.1 | 19.1 | 13.1 |

Table 8.3: WER reduction relative to the baseline model (%) depending on the fraction of the Disfluencies Train dataset used for training.

## 8.6 Future work

We see two directions for the future work which benefit each other. The first is increasing ASR robustness to disfluencies occurring in fluent speech by using data augmentation, semi-supervised learning approaches. The second is pushing the boundary of what an ASR system can do out-of-the-box for speakers with speech less fluent due to stutter, age or other factors.

# Chapter 9

# Conclusions

This dissertation makes a contribution to the fields of visual understanding in many directions: the activity prediction, the open set recognition, the self-supervised learning for vision transformers, and the speech recognition with speech disfluencies. The major contributions are summarized below:

- In Chapter 2 we focused on the action anticipation problem on videos, developing a new label smoothing method that inject the future action semantics in the loss function. This idea resembles the knowledge distillation process since useful information is injected into the model during training. We implement a multi-modal framework based on long short-term memory (LSTM) networks to summarize past observations and make predictions at different time steps. We perform extensive experiments on EPIC-Kitchens and EGTEA Gaze+ datasetsa and the experiments show that label smoothing systematically improves performance of state-of-the-art models for action anticipation.

- In Chapter 3 we proposes a novel attention-based technique to evaluate, simultaneously, slow and fast features extracted at different temporal scales, and several fusion schemes are considered to improve prediction accuracy. We perform extensive experiments on EpicKitchens-55 and EGTEA Gaze+ datasets,

and demonstrate that our technique systematically improves the results at different anticipation times.

- Our work in Chapter 4, primarily aims to revise this standard evaluation protocol to forecast crossing events as early as possible. To this end, we conceive a solution upon an extensively used model for egocentric action anticipation (RU-LSTM), proposing to envision future features, or modalities, that can better infer human intentions using a properly attention-based fusion mechanism. We validate our model against JAAD and PIE datasets and demonstrate that an intent prediction model can benefit from these additional clues for anticipating pedestrians crossing events.

- In Chapter 5 we propose to learn a prototypical representation of the full action realization for each action class, and to use them as source of regularization for the model during the training phase. In this way, the model not only tries to recognize the action from a partial observation but tries to predict the overall general action prototypical representation. We conducted several experiments and thanks to our method we achieved state-of-the-art results on different video datasets of different nature.

- In Chapter 6, we proposed a model based on capsule networks and based on variational auto-encoders for open set recognition. We conducted several experiments and ablation of our model, obtaining state of the art results on different datasets in the open set recognition and unknown detection tasks.

- In Chapter 7 we define a set of SSL tasks based on relations of image patches that the model has to solve before or jointly the supervised task. Differently from ViT, our RelViT model optimizes all the output tokens of the transformer encoder that are related to the image patches, thus exploiting more training signals at each training step. We investigated our methods on several image benchmarks finding that RelViT improves the SSL state-of-the-art methods by a large margin, especially on small datasets.

- In Chapter 8 we investigate data selection and preparation choices aiming for improved robustness of RNN-T ASR to speech disfluencies with a focus on partial words. For evaluation we use clean data, data with disfluencies and a separate dataset with speech affected by stuttering. We show that after including a small amount of data with disfluencies in the training set the recognition accuracy on the tests with disfluencies and stuttering improves.

# Appendix A

# Publications

This research activity has led to several publications in international conferences. These are summarized below[1].

## International Conferences and Workshops

1. **G. Camporese**, E. Izzo, L. Ballan. "Where are my Neighbors? Exploiting Patches Relations in Self-Supervised Vision Transformer", *The British Machine Vision Conference (BMVC 2022).*

2. N. Osman, E. Cancelli, **G. Camporese**, P. Coscia, L. Ballan. "Early Pedestrian Intent Prediction via Features Estimation", *IEEE International Conference on Image Processing, Bordeaux, France, 2022 (ICIP 2022).*

3. Y. Guo*, **G. Camporese***, W. Yang, A. Sperduti, L. Ballan. "Conditional Variational Capsule Network for Open Set Recognition", *Proc. of IEEE/CVF International Conference on Computer Vision (ICCV 2021).*

4. V. Mendelev, T. Raissi, **G. Camporese**, M. Giollo. "Improved Robustness to Disfluencies in RNN-Transducer Based Speech Recognition", *International*

[1]The author's bibliometric indices are the following: $H$-index = 4, total number of citations = 41 (source: Google Scholar on November 30, 2022).

* means equal contribution.

*Conference on Acoustics, Speech and Signal Processing, Toronto, Canada, 2021. (ICASSP 2021).*

5. **G. Camporese**, P. Coscia, A. Furnari, G. M. Farinella, L. Ballan. "Knowledge Distillation for Action Anticipation via Label Smoothing", *Proc. of International Conference on Pattern Recognition (ICPR 2020).*

6. N. Osman, **G. Camporese**, P. Coscia, L. Ballan. "SlowFast Rolling-Unrolling LSTMs for Action Anticipation in Egocentric Videos", *International Conference on Computer Vision, Virtual, 2021 (ICCVW 2021).*

# Preprint

1. **G. Camporese**, X. Lin, A. Bergamo, J. Tighe, D. Modolo. "Early Action Recognition with Action Prototypes". ArXiv, abs/2312.06598, 2023.

# Bibliography


[1] Mohammad Sadegh Ali Akbarian, Fatemeh Sadat Saleh, Mathieu Salzmann, Basura Fernando, Lars Petersson, and Lars Andersson. Encouraging lstms to anticipate actions very early. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 280–289, 2017.

[2] Sadeen Alharbi, Madina Hasan, Anthony JH Simons, Shelagh Brumfitt, and Phil Green. Sequence labeling to detect stuttering events in read speech. *Computer Speech & Language*, 62:101052, 2020.

[3] Mohammad Sadegh Aliakbarian, Fatemeh Sadat Saleh, Mathieu Salzmann, Basura Fernando, Lars Petersson, and Lars Andersson. Encouraging LSTMs to anticipate actions very early. In *IEEE Int. Conf. on Computer Vision*, 2017.

[4] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lucic, and Cordelia Schmid. Vivit: A video vision transformer. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6816–6826, 2021.

[5] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

[6] Nguyen Bach and Fei Huang. Noisy bilstm-based models for disfluency detection. In *Proc. Interspeech*, pages 4230–4234, 2019.

[7] Lamberto Ballan, Marco Bertini, Alberto Del Bimbo, Lorenzo Seidenari, and Giuseppe Serra. Event detection and recognition for semantic annotation of video. *Multimedia tools and applications*, 51(1):279–302, 2011.

[8] Federico Becattini, Tiberio Uricchio, Lorenzo Seidenari, Lamberto Ballan, and Alberto Del Bimbo. Am I done? predicting action progress in videos. *ACM Trans. on Multimedia Computing, Communications, and Applications (TOMM)*, in press, 2020.

[9] A. Bendale and T. E. Boult. Towards open world recognition. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1893–1902, 2015.

[10] A. Bendale and T. E. Boult. Towards open set deep networks. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[11] Apratim Bhattacharyya, Mario Fritz, and Bernt Schiele. Long-term on-board prediction of people in traffic scenes under uncertainty. In *The IEEE Conf. on Computer Vision and Pattern Recognition*, 2018.

[12] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[13] Guglielmo Camporese, Pasquale Coscia, Antonino Furnari, Giovanni Maria Farinella, and Lamberto Ballan. Knowledge distillation for action anticipation via label smoothing. *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 3312–3319, 2021.

[14] Guglielmo Camporese, Pasquale Coscia, Antonino Furnari, Giovanni Maria Farinella, and Lamberto Ballan. Knowledge distillation for action anticipation via label smoothing. *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 3312–3319, 2021.

[15] Yu Cao, Daniel Barrett, Andrei Barbu, Siddharth Narayanaswamy, Haonan Yu, Aaron Michaux, Yuewei Lin, Sven Dickinson, Jeffrey Mark Siskind, and Song Wang. Recognize human activities from partially observed videos. In *IEEE Computer Vision and Pattern Recognition*, 2013.

[16] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *ArXiv*, abs/2005.12872, 2020.

[17] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[18] Mathilde Caron, Hugo Touvron, Ishan Misra, Herve Jegou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[19] João Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about kinetics-600. *ArXiv*, abs/1808.01340, 2018.

[20] João Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, 2017.

[21] G. Chen, L. Qiao, Y. Shi, P. Peng, J. Li, T. Huang, S. Pu, and Y. Tian. Learning open set network with discriminative reciprocal points. In *Proc. of European Conference on Computer Vision (ECCV)*, pages 507–522, 2020.

[22] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *Proc. of the International Conference on Machine Learning (ICML)*, 2020.

[23] Ting Chen, Saurabh Saxena, Lala Li, David Fleet, and Geo rey E. Hinton. Pix2seq: A language modeling framework for object detection. *ArXiv*, abs/2109.10852, 2021.

[24] Xinlei Chen, Haoqi Fan, Ross B. Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *ArXiv*, abs/2003.04297, 2020.

[25] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[26] Zhengsu Chen, Lingxi Xie, Jianwei Niu, Xuefeng Liu, Longhui Wei, and Qi Tian. Visformer: The vision-friendly transformer. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[27] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Up-detr: Unsupervised pre-training for object detection with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1601–1610, June 2021.

[28] Dima Damen, Hazel Doughty, Giovanni Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. The epic-kitchens dataset: Collection, challenges and baselines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020.

[29] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736, 2018.

[30] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. The epic-kitchens dataset: Collection, challenges and baselines. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

[31] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. The epic-kitchens dataset: Collection, challenges and baselines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43:4125–4141, 2021.

[32] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Rescaling egocentric vision. *ArXiv*, abs/2006.13256, 2020.

[33] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *Proc. of International Conference on Machine Learning (ICML)*, 2006.

[34] Roeland De Geest, Efstratios Gavves, Amir Ghodrati, Zhenyang Li, Cees Snoek, and Tinne Tuytelaars. Online action detection. In *European Conf. on Computer Vision*, 2016.

[35] Roeland De Geest and Tinne Tuytelaars. Modeling temporal structure with lstm for online action detection. In *IEEE Winter Conf. on Applications in Computer Vision*, 2018.

[36] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2009.

[37] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.

[38] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2015.

[39] Piotr Dollár, Vincent Rabaud, Gary Cottrell, and Serge J. Belongie. Behavior recognition via sparse spatio-temporal features. *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 65–72, 2005.

[40] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Trevor Darrell, and Kate Saenko. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015.

[41] Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2625–2634, 2015.

[42] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2015.

[43] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929, 2021.

[44] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[45] Alexei A. Efros, Alexander C. Berg, Greg Mori, and Jitendra Malik. Recognizing action at a distance. *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 726–733 vol.2, 2003.

[46] Chenyou Fan, Jangwon Lee, and Michael S. Ryoo. Forecasting hands and objects in future frames. In *ECCV Workshops*, 2018.

[47] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6804–6815, 2021.

[48] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[49] Yazan Abu Farha, Alexander Richard, and Juergen Gall. When will you do what? anticipating temporal occurrences of activities. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5343–5352, 2018.

[50] Yazan Abu Farha, Alexander Richard, and Juergen Gall. When will you do what? anticipating temporal occurrences of activities. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5343–5352, 2018.

[51] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 200–210, 2020.

[52] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6201–6210, 2019.

[53] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6201–6210, 2019.

[54] Christoph Feichtenhofer, Axel Pinz, and Richard P Wildes. Spatiotemporal multiplier networks for video action recognition. In *IEEE Computer Vision and Pattern Recognition*, 2017.

[55] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *IEEE Computer Vision and Pattern Recognition*, 2016.

[56] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1933–1941, 2016.

[57] Panna Felsen, Pulkit Agrawal, and Jitendra Malik. What will happen next? forecasting player moves in sports videos. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3362–3371, 2017.

[58] James Ferguson, Greg Durrett, and Dan Klein. Disfluency detection with a semi-markov model and prosodic features. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 257–262, 2015.

[59] Linda Ficco, Lorenzo Mancuso, Jordi Manuello, Alessia Teneggi, Donato Liloia, Sergio Duca, Tommaso Costa, Gyula Zoltán Kovacs, and Franco Cauda. Disentangling predictive processing in the brain: a meta-analytic study in favour of a predictive network. *Scientific Reports*, 11(1):16258, Aug 2021.

[60] A. Furnari, S. Battiato, and G. M. Farinella. Leveraging uncertainty to rethink loss functions and evaluation measures for egocentric action anticipation. In *International Workshop on Egocentric Perception, Interaction and Computing (EPIC) in conjunction with ECCV*, 2018.

[61] Antonino Furnari, Sebastiano Battiato, Kristen Grauman, and Giovanni Farinella. Next-active-object prediction from egocentric videos. *Journal of Visual Communication and Image Representation*, 49, 10 2017.

[62] Antonino Furnari, Sebastiano Battiato, and Giovanni Maria Farinella. Leveraging uncertainty to rethink loss functions and evaluation measures for egocentric action anticipation. In *European Conf. on Computer Vision Workshops*, 2018.

[63] Antonino Furnari and Giovanni Farinella. Rolling-unrolling lstms for action anticipation from first-person video. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[64] Antonino Furnari and Giovanni Maria Farinella. Rolling-unrolling lstms for action anticipation from first-person video. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2020.

[65] Antonino Furnari and Giovanni Maria Farinella. Rolling-unrolling lstms for action anticipation from first-person video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43:4021–4036, 2021.

[66] Harshala Gammulle, Simon Denman, Sridha Sridharan, and Clinton Fookes. Predicting the future: A jointly learnt model for action anticipation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5561–5570, 2019.

[67] Jiyang Gao, Zhenheng Yang, and Ram Nevatia. RED: Reinforced encoder-decoder networks for action anticipation. *British Machine Vision Conf.*, 2017.

[68] Jiyang Gao, Zhenheng Yang, and Ram Nevatia. RED: reinforced encoder-decoder networks for action anticipation. In *British Machine Vision Conference 2017, BMVC 2017, London, UK, September 4-7, 2017*. BMVA Press, 2017.

[69] Z. Ge, S. Demyanov, and R. Garnavi. Generative openmax for multi-class open set classification. In *Proc. of British Machine Vision Conference (BMVC)*, 2017.

[70] C. Geng, S.-J. Huang, and S. Chen. Recent advances in open set recognition: A survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, in press, 2020.

[71] Joseph Gesnouin, Steve Pechberti, Bogdan Stanciulcscu, and Fabien Moutarde. Rnn-based pedestrian crossing prediction using activity and pose-related features. In *Proc. of IEEE International Conference on Automatic Face and Gesture Recognition*, 2021.

[72] Joseph Gesnouin, Steve Pechberti, Bogdan Stanciulcscu, and Fabien Moutarde. TrouSPI-Net: Spatio-temporal attention on parallel atrous convolutions and u-grus for skeletal pedestrian crossing prediction. In *Proc. of IEEE International Conference on Automatic Face and Gesture Recognition*, 2021.

[73] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2018.

[74] Rohit Girdhar, João Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 244–253, 2019.

[75] Rohit Girdhar and Kristen Grauman. Anticipative video transformer. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 13485–13495, 2021.

[76] Ross B. Girshick. Fast R-CNN. In *IEEE Int. Conf. on Computer Vision*, 2015.

[77] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fründ, Peter N. Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax,

and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5843–5851, 2017.

[78] Alex Graves. Sequence transduction with recurrent neural networks. *arXiv:1211.3711*, 2012.

[79] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

[80] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[81] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[82] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[83] Peter A Heeman and James Allen. Speech repains, intonational phrases, and discourse markers: modeling speakers' utterances in spoken dialogue. *Computational Linguistics*, 25(4):527–572, 1999.

[84] Olivier Henaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, S.M. Ali Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. In *Proc. of the International Conference on Machine Learning (ICML)*, 2020.

[85] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. In

*Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[86] G. E. Hinton, A. Krizhevsky, and S. D. Wang. Transforming auto-encoders. In *Proc. of Int'l Conference on Artificial Neural Networks (ICANN)*, 2011.

[87] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[88] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2018.

[89] Minh Hoai and Fernando De la Torre. Max-margin early event detectors. *Int. Journal of Computer Vision*, 107(2):191–202, 2014.

[90] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[91] Matthias Honal and Tanja Schultz. Correction of disfluencies in spontaneous speech using a noisy-channel approach. In *Eighth European Conference on Speech Communication and Technology*, 2003.

[92] Dong Huang, Shitong Yao, Yi Wang, and Fernando De La Torre. Sequential max-margin event detectors. In *European Conf. on Computer Vision*, 2014.

[93] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Int. Conf. on Machine Learning*, 2015.

[94] Ashesh Jain, Hema S. Koppula, Bharad Raghavan, Shane Soh, and Ashutosh Saxena. Car that knows before you do: Anticipating maneuvers via learning temporal driving models. In *IEEE Int. Conf. on Computer Vision*, 2015.

[95] L. P. Jain, W. J. Scheirer, and T. E. Boult. Multi-class open set recognition using probability of inclusion. In *Proc. of European Conference on Computer Vision (ECCV)*, 2014.

[96] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35(1):221–231, Jan 2013.

[97] Boyuan Jiang, Mengmeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. Stm: Spatiotemporal and motion encoding for action recognition. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2000–2009, 2019.

[98] Mark Johnson and Eugene Charniak. A tag-based noisy-channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 33–39, 2004.

[99] P. R. M. Junior, R. M. de Souza, R. O. Werneck, B. V. Stein, D. V. Pazinato, W. R. de Almeida, O. A. B. Penatti, R. S. Torres, and A. Rocha. Nearest neighbors distance ratio open-set classifier. *Machine Learning*, 106(3):359–386, 2017.

[100] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE Computer Vision and Pattern Recognition*, Jun 2014.

[101] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[102] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Apostol Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *ArXiv*, abs/1705.06950, 2017.

[103] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *Proc. of International Conference on Learning Representations (ICLR)*, 2014.

[104] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[105] Alexander Kläser, Marcin Marszalek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, 2008.

[106] D. I. Kondratyuk, Liangzhe Yuan, Yandong Li, Li Zhang, Mingxing Tan, Matthew A. Brown, and Boqing Gong. Movinets: Mobile video networks for efficient video recognition. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16015–16025, 2021.

[107] Yu Kong, Shangqian Gao, Bin Sun, and Yun Raymond Fu. Action prediction from videos via memorizing hard-to-predict samples. In *AAAI*, 2018.

[108] Hema S. Koppula and Ashutosh Saxena. Anticipating human activities using object affordances for reactive robotic response. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 38(1):14–29, 2016.

[109] Hema S. Koppula and Ashutosh Saxena. Anticipating human activities using object affordances for reactive robotic response. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):14–29, 2016.

[110] Iuliia Kotseruba, Amir Rasouli, and John K Tsotsos. Do they want to cross? understanding pedestrian intention for behavior prediction. In *Proc. of IEEE Intelligent Vehicles Symposium (IV)*, 2020.

[111] Iuliia Kotseruba, Amir Rasouli, and John K. Tsotsos. Benchmark for evaluating pedestrian action prediction. In *Proc. of IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2021.

[112] Tedd Kourkounakis, Amirhossein Hajavi, and Ali Etemad. Detecting multiple speech disfluencies using a deep residual network with bidirectional long short-term memory. In *Proc. IEEE Intern. Conf. on Acoustics, Speech and Signal Process. (ICASSP)*, pages 6089–6093, 2020.

[113] A. Krizhevsky. Learning multiple layers of features from tiny images. In *https://www.cs.toronto.edu/ kriz/learning-features-2009-TR.pdf*, 2009.

[114] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

[115] Tian Lan, Tsung-Chuan Chen, and Silvio Savarese. A hierarchical representation for future action prediction. In *European Conf. on Computer Vision*, 2014.

[116] Tian Lan, Tsung-Chuan Chen, and Silvio Savarese. A hierarchical representation for future action prediction. In *European Conference on Computer Vision*, 2014.

[117] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2008.

[118] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[119] Y. Le and X. Yang. Tiny imagenet visual recognition challenge. In *CS 231N*, 2015.

[120] Yann LeCun and Corinna Cortes. The mnist database of handwritten digits. In *http://yann.lecun.com/exdb/mnist/*, 2005.

[121] Dong Li, Zhaofan Qiu, Qi Dai, Ting Yao, and Tao Mei. Recurrent tubelet proposal and recognition networks for action detection. In *ECCV*, 2018.

[122] Yin Li, Miao Liu, and James M. Rehg. In the eye of beholder: Joint learning of gaze and actions in first person video. In *ECCV*, 2018.

[123] Yin Li, Miao Liu, and James M. Rehg. In the eye of beholder: Joint learning of gaze and actions in first person video. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 639–655, Cham, 2018. Springer International Publishing.

[124] Zhenyang Li, Kirill Gavrilyuk, Efstratios Gavves, Mihir Jain, and Cees G. M. Snoek. Videolstm convolves, attends and flows for action recognition. *ArXiv*, abs/1607.01794, 2018.

[125] Junwei Liang, Lu Jiang, Juan Carlos Niebles, Alexander G Hauptmann, and Li Fei-Fei. Peeking into the future: Predicting future person activities and locations in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5725–5734, 2019.

[126] Ji Lin, Chuang Gan, and Song Han. TSM: Temporal shift module for efficient video understanding. In *IEEE Int. Conf. on Computer Vision*, 2019.

[127] Bingbin Liu, Ehsan Adeli, Zhangjie Cao, Kuan-Hui Lee, Abhijeet Shenoi, Adrien Gaidon, and Juan Carlos Niebles. Spatiotemporal relationship reasoning for pedestrian intent prediction. *IEEE Robotics and Automation Letters*, 5(2):3485–3492, 2020.

[128] Yahui Liu, Enver Sangineto, Wei Bi, Nicu Sebe, Bruno Lepri, and Marco De Nadai. Efficient training of visual transformers with small-size datasets. In *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[129] Yang Liu, Elizabeth Shriberg, Andreas Stolcke, and Mary Harper. Comparing hmm, maximum entropy, and conditional random fields for disfluency detection. In *Ninth European Conference on Speech Communication and Technology*, 2005.

[130] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9992–10002, 2021.

[131] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[132] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. *ArXiv*, abs/2106.13230, 2021.

[133] Ze Liu, Zheng Zhang, Yue Cao, Han Hu, and Xin Tong. Group-free 3d object detection via transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2949–2958, October 2021.

[134] Javier Lorenzo, Ignacio Parra Alonso, Rubén Izquierdo, Augusto Luis Ballardini, Álvaro Hernández Saz, David Fernández Llorca, and Miguel Ángel Sotelo. CAPformer: pedestrian crossing action prediction using transformer. *Sensors*, 21(17), 2021.

[135] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2019.

[136] Artur Luczak, Bruce L. McNaughton, and Yoshimasa Kubo. Neurons learn by predicting future activity. *Nature Machine Intelligence*, 4(1):62–72, Jan 2022.

[137] Tahmida Mahmud, Mahmudul Hasan, and Amit K. Roy-Chowdhury. Joint prediction of activity labels and starting times in untrimmed videos. In *IEEE Int. Conf. on Computer Vision*, Oct 2017.

[138] Tahmida Mahmud, Mahmudul Hasan, and Amit K. Roy-Chowdhury. Joint prediction of activity labels and starting times in untrimmed videos. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[139] Ana I. Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso García, and Davide Scaramuzza. Event-based vision meets deep learning on steering prediction for self-driving cars. In *The IEEE Conf. on Computer Vision and Pattern Recognition*, June 2018.

[140] Brais Martínez, Davide Modolo, Yuanjun Xiong, and Joseph Tighe. Action recognition with spatial-temporal discriminative filter banks. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5481–5490, 2019.

[141] Antoine Miech, Ivan Laptev, Josef Sivic, Heng Wang, Lorenzo Torresani, and Du Tran. Leveraging the present to anticipate the future in videos. In *IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, 2019.

[142] Beren Millidge, Anil. K. Seth, and Christopher L. Buckley. Predictive coding: a theoretical and experimental review. *ArXiv*, abs/2107.12979, 2021.

[143] Jonathan Munro and Dima Damen. Multi-modal Domain Adaptation for Fine-grained Action Recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[144] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 257–265, 2017.

[145] L. Neal, M. Olson, X. Fern, W.-K. Wong, and F. Li. Open set learning with counterfactual images. In *Proc. of European Conference on Computer Vision (ECCV)*, 2018.

[146] Satyajit Neogi, Michael Hoy, Kang Dang, Hang Yu, and Justin Dauwels. Context model for pedestrian intention prediction using factored latent-dynamic conditional random fields. *IEEE Transactions on Intelligent Transportation Systems*, 22(11):6821–6832, 2020.

[147] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[148] Joe Yue-Hei Ng, Matthew J. Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4694–4702, 2015.

[149] Joe Yue-Hei Ng, Matthew J. Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4694–4702, 2015.

[150] H. H. Nguyen, J. Yamagishi, and I. Echizen. Capsule-forensics: Using capsule networks to detect forged images and videos. In *Proc. of IEEE Int'l Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[151] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. *Proc. of Indian Conference on Computer Vision, Graphics & Image Processing (ICVGIP)*, 2008.

[152] Yulei Niu, Zhiwu Lu, Ji-Rong Wen, Tao Xiang, and Shih-Fu Chang. Multi-modal multi-scale deep learning for large-scale image annotation. *IEEE Transactions on Image Processing*, 28:1720–1731, 04 2019.

[153] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2016.

[154] Mehdi Noroozi, Hamed Pirsiavash, and Paolo Favaro. Representation learning by learning to count. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

[155] Nada Emam Ahmed Osman, Guglielmo Camporese, Pasquale Coscia, and Lamberto Ballan. Slowfast rolling-unrolling lstms for action anticipation in egocentric videos. *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 3430–3438, 2021.

[156] Nada Emam Ahmed Osman, Guglielmo Camporese, Pasquale Coscia, and Lamberto Ballan. Slowfast rolling-unrolling lstms for action anticipation in egocentric videos. *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 3430–3438, 2021.

[157] P. Oza and V. M. Patel. C2AE: Class conditioned auto-encoder for open-set recognition. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2302–2311, 2019.

[158] A. Pagnoni, K. Liu, and S. Li. Conditional variational autoencoder for neural machine translation. *ArXiv*, abs/1812.04405, 2018.

[159] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*, 2019.

[160] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[161] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *Proc.*

*of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[162] Xiaojiang Peng, Changqing Zou, Yu Qiao, and Qiang Peng. Action recognition with stacked fisher vectors. In *ECCV*, 2014.

[163] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing*, 2014.

[164] P. Perera, V. I. Morariu, R. Jain, V. Manjunatha, C. Wigington, V. Ordonez, and V. M. Patel. Generative-discriminative feature representations for open-set recognition. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11811–11820, 2020.

[165] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5534–5542, 2017.

[166] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? In *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[167] J. Rajasegaran, V. Jayasundara, S. Jayasekara, H. Jayasekara, S. Seneviratne, and R. Rodrigo. Deepcaps: Going deeper with capsule networks. In *Proc. of IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[168] Amir Rasouli, Iuliia Kotseruba, Toni Kunic, and John K. Tsotsos. Pie: A large-scale dataset and models for pedestrian intention estimation and trajectory prediction. In *Proc. of IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[169] Amir Rasouli, Iuliia Kotseruba, and John K Tsotsos. Are they going to cross? a benchmark dataset and baseline for pedestrian crosswalk behavior. In *Proc. of IEEE/CVF International Conference on Computer Vision Workshops*, 2017.

[170] Nicholas Rhinehart and Kris M. Kitani. First-person activity forecasting with online inverse reinforcement learning. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3716–3725, 2017.

[171] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Proc. of International Conf. on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[172] Michael S Ryoo. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *IEEE Int. Conf. on Computer Vision*, 2011.

[173] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. In *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[174] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult. Toward open set recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(7):1757–1772, 2013.

[175] W. J. Scheirer, L. P. Jain, and T. E. Boult. Probability models for open set recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 36(11):2317–2324, 2014.

[176] Paul Schydlo, Mirko Rakovic, Lorenzo Jamone, and José Santos-Victor. Anticipation in human-robot cooperation: A recurrent neural network approach for multiple action sequences prediction. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pages 1–6. IEEE, 2018.

[177] Fadime Sener, Dipika Singhania, and Angela Yao. Temporal aggregate representations for long-range video understanding. In *European Conference on Computer Vision*, 2020.

[178] Fadime Sener, Dipika Singhania, and Angela Yao. Temporal aggregate representations for long term video understanding. *ArXiv*, abs/2006.00830, 2020.

[179] Joel Shor, Dotan Emanuel, Oran Lang, Omry Tuval, Michael Brenner, Julie Cattiau, Fernando Vieira, Maeve McNally, Taylor Charbonneau, Melissa Nollstadt, et al. Personalizing asr for dysarthric and accented speech with limited data. *arXiv:1907.13511*, 2019.

[180] Elizabeth E Shriberg. Phonetic consequences of speech disfluency. Technical report, SRI INTERNATIONAL MENLO PARK CA, 1999.

[181] Elizabeth Ellen Shriberg. *Preliminaries to a theory of speech disfluencies*. PhD thesis, Citeseer, 1994.

[182] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. of International Conference on Learning Representations (ICLR)*, pages 1–14, 2015.

[183] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[184] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *Advances in Neural Information Processing Systems*, 1, 06 2014.

[185] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *NIPS*, 2017.

[186] Sijie Song, Jiaying Liu, Yanghao Li, and Zongming Guo. Modality compensation network: Cross-modal adaptation for action recognition. *IEEE Transactions on Image Processing*, 29:3957–3969, 2020.

[187] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *ArXiv*, abs/1212.0402, 2012.

[188] Alexandros Stergiou and Dima Damen. Temporal progressive attention for early action prediction. *ArXiv*, abs/2204.13340, 2022.

[189] Swathikiran Sudhakaran, Sergio Escalera, and Oswald Lanz. LSTA: Long short-term attention for egocentric action recognition. In *IEEE Computer Vision and Pattern Recognition*, 2018.

[190] Waqas Sultani, Chen Chen, and Mubarak Shah. Real-world anomaly detection in surveillance videos. In *IEEE Conf. on Computer Vision and Pattern Recognition*, June 2018.

[191] X. Sun, Z. Yang, C. Zhang, K.-V. Ling, and G. Peng. Conditional gaussian distribution learning for open set recognition. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[192] Thattapon Surasak, Ito Takahiro, Cheng hsuan Cheng, Chi en Wang, and Pao you Sheng. Histogram of oriented gradients for human detection in video. *2018 5th International Conference on Business and Industrial Research (ICBIR)*, pages 172–176, 2018.

[193] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conf. on Computer Vision and Pattern Recognition*, June 2016.

[194] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1701–1708, 2014.

[195] Graham W. Taylor, Rob Fergus, Yann LeCun, and Christoph Bregler. Convolutional learning of spatio-temporal features. In *European Conference on Computer Vision*, 2010.

[196] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herv'e J'egou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021.

[197] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *IEEE Int. Conf. on Computer Vision*, 2015.

[198] Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015.

[199] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2018.

[200] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[201] A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. In *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[202] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018.

[203] L. Van der Maaten and G. E. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[204] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, L ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[205] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[206] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proc. of the International Conference on Machine Learning (ICML)*, 2008.

[207] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating visual representations from unlabeled video. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016.

[208] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating visual representations from unlabeled video. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 98–106, 2016.

[209] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. *Int. Journal of Computer Vision*, 103(1):60–79, 2013.

[210] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *IEEE Int. Conf. on Computer Vision*, 2013.

[211] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4305–4314, 2015.

[212] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. *ArXiv*, abs/1608.00859, 2016.

[213] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European Conf. on Computer Vision*, 2016.

[214] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[215] Xiaohan Wang, Yu Wu, Linchao Zhu, and Yi Yang. Symbiotic attention with privileged information for egocentric action recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):12249–12256, Apr. 2020.

[216] Xionghui Wang, Jianfang Hu, Jianhuang Lai, Jianguo Zhang, and Weishi Zheng. Progressive teacher-student learning for early action prediction. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3551–3560, 2019.

[217] Chao-Yuan Wu, Yanghao Li, Karttikeya Mangalam, Haoqi Fan, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Memvit: Memory-augmented multiscale vision transformer for efficient long-term video recognition, 2022.

[218] Chao-Yuan Wu, Manzil Zaheer, Hexiang Hu, R. Manmatha, Alex Smola, and Philipp Krähenbühl. Compressed video action recognition. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6026–6035, 2018.

[219] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. CvT: Introducing convolutions to vision transformers. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[220] Xinxiao Wu, Ruiqi Wang, Jingyi Hou, Hanxi Lin, and Jiebo Luo. Spatial-temporal relation reasoning for action prediction in videos. *Int. J. Comput. Vis.*, 129:1484–1505, 2021.

[221] Xinxiao Wu, Jianwei Zhao, and Ruiqi Wang. Anticipating future relations via graph growing for action prediction. In *AAAI*, 2021.

[222] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[223] Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollár, and Ross Girshick. Early convolutions help transformers see better. In *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[224] Dongfang Yang, Haolin Zhang, Ekim Yurtsever, Keith Redmill, and Ümit Özgüner. Predicting pedestrian crossing intention with feature fusion and spatio-temporal attention. *arXiv preprint arXiv:2104.05485*, 2021.

[225] Mang Ye, Xu Zhang, PongChi Yuen, and Shih-Fu Chang. Unsupervised embedding learning via invariant and spreading instance feature. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[226] R. Yoshihashi, W. Shao, R. Kawakami, S. You, M. Iida, and T. Naemura. Classification-reconstruction learning for open-set recognition. In *Proc. of IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[227] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[228] Vicky Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. Disfluency detection using a bidirectional lstm. *arXiv:1604.03209*, 2016.

[229] Kuo-Hao Zeng, Bokui (William) Shen, De-An Huang, Min Sun, and Juan Carlos Niebles. Visual forecasting by imitating dynamics in natural sequences. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3018–3027, 2017.

[230] Kuo-Hao Zeng, William B Shen, De-An Huang, Min Sun, and Juan Carlos Niebles. Visual forecasting by imitating dynamics in natural sequences. In *IEEE Int. Conf. on Computer Vision*, Oct 2017.

[231] H. Zhang, A. Li, J. Guo, and Y. Guo. Hybrid models for open set recognition. In *Proc. of European Conference on Computer Vision (ECCV)*, 2020.

[232] H. Zhang and V. M. Patel. Sparse representation-based open set recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(8):1690–1696, 2017.

[233] Mengmi Zhang, Keng Teck Ma, Joo Hwee Lim, Qi Zhao, and Jiashi Feng. Deep future gaze: Gaze anticipation on egocentric videos using adversarial networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3539–3548, 2017.

[234] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2016.

[235] Richard Zhang, Phillip Isola, and Alexei A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[236] He Zhao and Richard P. Wildes. Spatiotemporal feature residual propagation for action prediction. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7002–7011, 2019.

[237] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *European Conf. on Computer Vision*, 2018.

[238] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *ArXiv*, abs/2010.04159, 2021.