Utrecht University

Bridging the gap: Threefold knowledge distillation for language-enabled action recognition models operating in real-time scenarios

Graduate school of natural sciences

Master thesis, May 2024

Student name:

Frans Huntink
f.g.huntink@students.uu.nl

Thesis Supervisor:

Ronald Poppe
r.w.poppe@uu.nl

Secondary Supervisor:

Heysem Kaya
h.kaya@uu.nl

Daily Supervisor:

Wikke Heesterbeek
wikkejoris@hotmail.com

Thesis completed in collaboration with Oddity.ai

Abstract

In this work, we propose a set of knowledge distillation techniques that aim to transfer the benefits of large and computationally slow language-enabled action recognition (AR) models to smaller, faster student models that can operate in real-time scenarios. This means that important benefits of these models such as unprecedented predictive performance, flexible natural language interaction and "zero-shot" predictions can be used in real-time scenarios. We study existing language-based AR models and find that transformer models using the Contrastive Language-Image Pretraining (CLIP) model as an encoder backbone perform best among the set of existing language-enabled AR models. We determine that the CLIP-based AR model called ActionCLIP is most suitable for our distillation experiments by comparing it to other CLIP-based models in terms of predictive performance and inference time using a dense frame sampling strategy. We then propose three distillation techniques that each distill a specific portion of the knowledge contained in the ActionCLIP model into a smaller, faster student model. First, we propose a way to replace the CLIP encoder backbone of the ActionCLIP model with a model from the distilled TinyCLIP family. In doing so, we find a steep decrease in inference time but also find a significant drop in predictive performance. Next, we propose a method to distill the spatial knowledge contained in the CLIP model itself. We do this by creating a multi-task learning problem for our ActionCLIP model in which the model has to predict both the ground truth human actions label and a set of additional spatial objectives for a given AR dataset. We generate these additional objectives by designing a CLIP-based spatial prediction framework. We find that spatial distillation improves the predictive performance of our ActionCLIP model as compared to its original single-objective implementation. Finally, we adapt a distillation technique called the data efficient image transformer distillation (DeiT) approach to be able to distill video transformers instead of image transformers. We then apply the DeiT strategy by using a large pretrained ActionCLIP model as a teacher to the smaller, faster ActionCLIP student. We find significant improvement in predictive performance over the original ActionCLIP training strategy but also find that this is caused by our so-called two-headed training strategy we introduced to accommodate for DeiT distillation, not because of the teacher supervision. This two-headed training strategy implies that not one but two perspectives on a video sample are learned: a token-based perspective and a frame-based perspective. We find that its success is likely because the model is supervised to simultaneously learn these two perspectives of a video. We then explore ways to combine the distillation techniques and obtain an ActionCLIP student model that reaches a Top-1 validation score of 66.5 on the HMDB51 dataset, which is only slightly lower than the 68.8 Top-1 validation score obtained by the original ActionCLIP model at half the backbone parameters and more than 1.73× the inference speed. We conclude by showing that this allows our model to be applied to the domain of real-time AR.

Table of content

Contents

List of Tables

List of Figures

1 Introduction

In the field of computer vision, action recognition (AR) is the area that specialises in the analysis and classification of images or a set of video frames in order to identify specific human actions or behaviours. The introduction of large pre-trained neural networks, particularly transformer models, have provided significant advancements in the field of image and video understanding. These models are known for their exceptional robustness to outliers and varying input representations thanks to their extensive training on diverse, very large datasets [21]. However, the biggest advantage that some of these large pretrained models bring is the fact that they can perform language-enabled image or video recognition. For a model to be language-enabled means that it can flexibly match image or video content with natural language prompts. Intuitively, it means that it can match a piece of text to a corresponding photo or video and vice versa. It also means that for image or video classification, one can dynamically supply a set of so-called prompts, which means the possible labels to distinguish between, in a natural language form. Given a sufficiently large pretraining, these language-enabled models can then perform classification in a "zero-shot" or "few-shot" context. This means that they can perform predictions on a dataset they have never seen before during training without requiring any additional fine-tuning on examples of that specific dataset. This is a key advantage that gives these models unparalleled flexibility in the domain of visual recognition. One such language-enabled transformer network is CLIP. CLIP stands for Contrastive Language-Image Pretraining and is a relatively novel transformer-based approach for image recognition [21]. During training, CLIP sees large amounts of paired text and image content and uses this to create a shared embedding space between natural language and image content. Given its pretraining on a very large dataset, CLIP can then be applied as a flexible image recognition model in a wide range of tasks with little to no finetuning on data of the specific domain it is applied to [21]. Work has been done to transfer these impressive results to the domain of video AR. Models like ActionCLIP [30], Seeing in Flowing [31] and STAN [16] are language-enabled AR models that combine the CLIP model as an encoder backbone with additional components that aim to capture the temporal characteristics of video content. Unsurprisingly, given the semantically rich encoding provided by CLIP, these models perform well on video AR benchmarks like Kinetics, HMDB51 and UCF101 and they can typically be inferred in the same "zero-shot" configuration as their CLIP backbone. However, the computationally costly CLIP backbone and the additional (often transformer-based) pipeline components make these AR models suffer from higher run-time complexity than most traditional CNN-based AR architectures. This makes these models unfit to perform real-time language enabled action recognition.

1.1 Problem statement

With the current paradigm of increasing compute power and designing larger models to obtain favourable predictive performance, the gap between academic advances in the field of computer vision and its applications in real-time scenarios continues to widen. At the same time, improving model performance through sheer amounts of data and increasing the model size to squeeze out an extra few percent of predictive performance is something that suits the (vision) transformer architecture particularly well. Indeed, the transformer architecture excels at fitting to large amounts of data, more so than the traditional convolutional neural networks (CNN) that used to dominate the field of computer vision [21]. The direct implication is that most advances in the field of vision transformers go hand-in-hand with slow and impractical models that are ill-suited to operate in real-time scenarios. While AR can be applied in both real-time and non-real-time contexts, the real-time variant stands out for its ability to offer immediate interpretation of streamed video footage. These real-time interpretations can be used in a wide range of fields such as safety and surveillance, robotics or that of autonomous vehicles. These fields share one key characteristic, namely that speed is of the essence. Whether the model is applied in an autonomously driving car or to perform surveillance in a public space, a quick classification can mean the difference between a useful and a useless model. At the same time, sufficient accuracy is equally important. A small and fast model that has significantly lower accuracy than its larger counterpart can be equally useless. Indeed, for AR models operating in real-time, a suitable speed-accuracy tradeoff is exceptionally important. Given their typical size, this is a factor that large, pretrained vision transformer models struggle with. For example, an AR model that requires one NVIDIA A100 GPU to monitor a single video stream with a five second delay is not very likely to be implemented in an autonomous car or security camera context because it is too impractical and expensive. Indeed, many AR models applied in real-time scenarios do not yet fully benefit from important advancements in this field such as the unprecedented predictive performance, the natural language interaction and "zero-shot" capabilities found in novel language-enabled vision models. Smaller, faster versions of these models are needed with minimal sacrifice in predictive performance in order to bring these advantages to the broad and relevant domain of real-time AR.

1.2 Research scope

An increasing number of solutions exist that aim to bridge the gap between the aforementioned academic advances in computer vision and their application in real-time scenarios. In this work, we consider a model to be suited to operate in a real-time scenario if it takes less (or an equal amount of) time to process a set of frames than the time it takes to receive a set of new frames. Intuitively, this means that there is no processing "backlog" and that we are therefore operating in real-time. It follows that part of the processing delay is the result of waiting for the n frames to perform inference on. For example, for a camera running at 32 frames per second (fps), waiting for 32 frames would add an instant one-second delay. At the same time, we will want to process multiple concurrent camera streams on a single GPU because of the point we made earlier: requiring a single GPU for only two or three simultaneous streams is not a viable option in real-time scenarios because it is too costly and comes with impractical hardware overhead.
To this end, different forms of knowledge distillation (KD) have been deployed to condense the information contained in large pretrained teacher models into smaller, faster student networks [7]. Because of the CLIP model's recent success in both the image and action recognition domain, a logical first step would be to apply KD to CLIP and its AR counterparts in order to obtain a language-enabled AR model that operates in real-time. However, what makes matters complicated is the fact that these language-enabled models are cross-modal. A cross-modal model creates both a textual and visual embedding over an image or a set of video frames, and it aligns the two in a shared embedding space using a contrastive learning strategy [21]. In the context of AR, the cross-modal embedding space is a space shared between language and video. This particular characteristic makes applying KD more challenging because it has to be applied symmetrically to both the language and vision components of the cross-modal model so as not to skew this shared embedding space. What is more, the largest CLIP variant took 18 days to train on 592 V-100 GPUs [21]. We suspect that applying classical KD techniques using the same (very large) dataset that CLIP was trained on would be as computationally demanding as training the CLIP model itself. This places using standard KD techniques to distill the CLIP model out of reach of the average training hardware. Given these considerations, we intend to research the ways in which KD techniques can be applied towards distilling CLIP and the corresponding set CLIP-based AR models in a way that respects the cross-modal space that these models operate in and that can be distilled without the need for unrealistic hardware budgets. By doing so, we hope to distill a flexible language-enabled AR model that can operate in a real-time context at minimal cost to the predictive performance.

1.3 Thesis outline

We start by studying existing literature on four relevant topics: transformers for image recognition, transitioning from image recognition to video recognition, existing language-enabled AR models, and finally the field of Knowledge Distillation (KD) for transformers. We study the first two topics to gain an understanding of the underlying mechanisms of the image and video transformers, which will help us pinpoint the most important contributors to the existing speed-accuracy trade-off of these architectures. We are interested in the existing set of language-enabled AR models and KD strategies because these can help in transferring the benefits of these AR models to the domain of real-time use. We aim to identify a set of effective KD strategies that respect the cross-modal space in which the language-enabled AR models operate, and apply them to the existing set of language-enabled AR models. Based on the results of this literature study, we formulate a more informed research scope and an accompanying set of research questions. We then investigate these research questions by constructing and evaluating relevant experiments for each. We combine these experimental results in the discussion section to answer our set of research questions. We conclude by discussing the limitations of our proposed methods and a section on possible future work, after which we present an overarching conclusion.

2 Literature study

In this literature study we start by providing a brief overview of existing transformers for image recognition tasks in Section 2.1. We discuss important models like the vision transformer (ViT) and CLIP. We also compare these transformer models to CNN models that can perform similar tasks and look at differences in performance and generalisation. We then move on to the process of transforming image understanding to video understanding in section 2.2. In this section, we are particularly interested in the way in which temporal information is modelled in existing video classification models. We discuss important works like optical flow fields, two-stream networks, frame differences and the more sophisticated ViVit approach that models temporal information through its ViT backbone.
Next, in Section 2.3 we provide an overview of available language inclusive AR models like ActionCLIP, Seeing in Flowing, STAN and VideoBERT. We compare the models and focus on how each model integrates temporal information, spatial information and how natural language is connected to the spatio-temporal representation of a video clip. Finally, we look at a selection of transformer distillation techniques that are potentially relevant in distilling large language-enabled vision transformers in Section 2.4.

2.1 Transformers for image recognition

Transformers have caused a paradigm shift in the field of deep learning. The success of the transformer architecture in the field of natural-language processing has led to the development of similar models that can operate in the field of computer vision. Most notably, the vision transformer (ViT), a transformer architecture designed to perform image recognition, outperforms many CNN-based architectures on various benchmarks. [27]. Given that the image transformer lies at the heart of the best-performing language-enabled vision models such as CLIP, we provide a brief overview of the most important models in that field.

2.1.1 Vision Transformer (ViT)

The driving factor behind the success of transformers in the fields of natural language processing (NLP) and computer vision is the concept of attention [27]. Originally designed for NLP encoder-decoder systems, attention mechanisms allow modelling of dependencies without regard to their distance in the input or output sequences [28]. The rise of transformer architectures in computer vision is mainly due to Dosovitskiy et al. who successfully designed the vision transformer (ViT) suited for classification tasks. In their paper, they propose a way to tokenise images so that these can be used the same way as tokens (e.g. words, parts of words, punctuation marks) in an NLP application. They do this by dividing an image into patches. These patches are linearly embedded and a positional token is added to each patch to indicate its intra-frame position. Linear projection is a standard procedure in transformers. By multiplying the input by a learned projection matrix it compresses the frame input size and it learns to extract valuable information. The added positional encodings are equally important in transformer architectures. Where convolution-based models sequentially process a frame and therefore do not require positional encodings, transformers do not possess this characteristic and will therefore disregard the order of the tokens if they are not explicitly included in the representation [28]. Similar to NLP transformers, a learnable classification token is prepended to the image representation that serves as the final image embedding used for classification. The ViT architecture is depicted in Figure 1. In this example, the input image is divided into 9 patches that are linearly projected to form a sequence of tokens. These are then passed through the Transformer Encoder module for a predefined number of layers. Finally, an MLP takes the classification token as input and produces a class label.

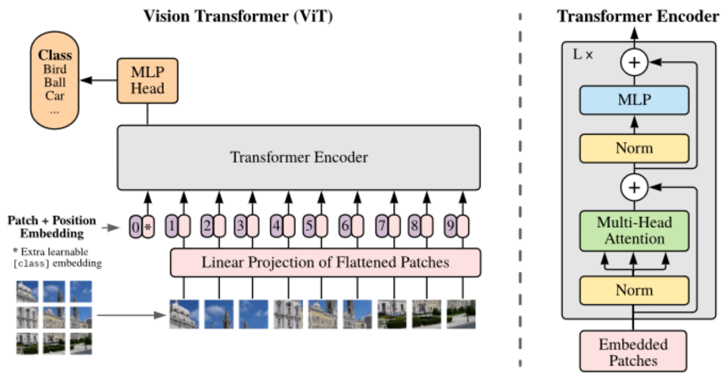


Figure 1: **The Vision Transformer (ViT) model** Figure by Dosovitskiy et al.

In the original ViT paper, Dosovitskiy et al. show that their ViT-based model outperforms the older generations of CNN-based ResNets [5]. More ViT-based architectures have been proposed since then. In many cases, these transformers perform better than older CNN-based architectures like ResNets on popular image recognition benchmarks like ImageNet and Cifar-100.

2.1.2 Contrastive Language-Image Pretraining (CLIP)

Another model is CLIP, a model proposed by Radford et al. CLIP leverages a very broad source of supervision. More specifically, it is trained on a dataset consisting of over 400 million image-text pairs, after which it is finetuned on more specific tasks [21]. Notably, the text labels describe the images in natural text similar to how the image and text would appear together on the internet. Figure 2 outlines the general concept of CLIP. An image encoder and text encoder are jointly trained using contrastive loss with a dataset of images and text that appear together in a batch. This means that matched image-text pairs are either rewarded or penalised depending on whether they appear together in that dataset. For example, the training process would stimulate the embeddings for the words "a dog playing with a ball" and a picture of a dog playing with a ball to be close together and would simultaneously discourage similarity to unrelated pictures. This way, text is essentially connected to image and vice versa. This allows the model to be dynamically prompted with interchangeable labels rather than being confined to a fixed set of class labels. That is, given a large enough pretraining, CLIP should have a general idea of how natural language provided in a prompt relates to an input image and vice versa, even for data it has not explicitly seen. As for the encoders, the authors experiment with both a CNN-based and ViT-based backbone for the vision encoder and find that the ViT performs best. This means that the ViT is trained using the contrastive loss depicted in Figure 2 so that it learns to produce semantically rich visual embeddings that align with the embeddings produced with the text encoder (and vice versa). Four different CLIP backbone variants are introduced: CLIP-ViT-B/32, CLIP-ViT-B/16, CLIP-ViT-L/14 and CLIP-ViT-L14-336px. Each backbone variant is pretrained on the same data, and the B, and L stand for Base and Large, respectively. These indicators describe the depth of the underlying ViT transformer that performs the encoding of the input image. The numbers indicate the size of the patches. For example, the CLIP-ViT-B/32 variant uses the Base ViT model and uses a 32×32 pixels patch size, meaning that an input image is divided into patches of $32\times$ pixels. A patch size of 16×16 (ViT-B/16) therefore means more patches and thus granular information, but comes at a higher computational complexity because more patch-based tokens have to processed in the model. The text encoder is a transformer as introduced by Vaswani et al. with some minor modifications to the layer normalisation method and weight scaling [20].

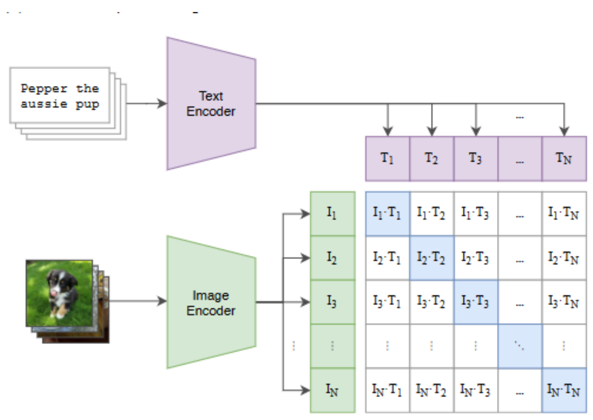

Pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 I_2 I_3 ⋮ I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$

Figure 2: **CLIP's contrastive learning approach.** Figure by Radford et al.

2.1.3 Differences with CNN-based architectures

There are significant architectural differences between CNNs and transformers. One such difference is that transformers apply global attention to an image as opposed to the local attention of a CNN. That is, a CNN slides multiple $K \times K$-dimensional convolutional filters (where K is in pixels) over each image, and by doing so it looks for patterns within that (local) filter window. This stands in contrast to the transformer model, whose attention heads each focus on specific patterns within the confines of the entire image. Transformers outperform CNNs on transfer learning tasks in which the transformer is pre-trained on a very large dataset and then finetuned on data for a more specific task. At the same time, a CNN-based architecture is typically better able to fit to limited data because of the inductive bias present in the convolutional approach. The fact that transformers are better able to fit large amounts of data is demonstrated in the previously mentioned CLIP paper by Radford et al. As previously mentioned, they experiment with both a (CNN-based) ResNet backbone and a transformer backbone for producing visual embeddings of a given image. They find that the transformer backbone outperforms the ResNet backbone in all cases. This shows that the transformer architecture is better able to fit to very large amounts of data. The demonstrated flexibility of transformer finetuning allows the same pretrained model to be applied to different downstream tasks by finetuning the model on a dataset for the task at hand. The CLIP model provides an excellent example of this ability.

2.2 From image recognition to video recognition

An important consideration in applying these powerful image transformers to AR is the translation of image representation to video representation. The key difference between the two is that video recognition requires both a strong spatial understanding and a strong temporal understanding [27] [22] [29]. That is, one cannot reliably infer if and how something moves by just looking at a fixed spatial representation. Therefore, a model needs not only a spatial dimension, but also a temporal dimension. The difference is that the spatial representation tells us what we are looking at (e.g. a person, a bike, a landscape) whereas the temporal layer tells how its movements change over time. Various ways of capturing temporal dimensions have been proposed.
One way to capture the spatial dimension is by calculating so-called optical flow fields as originally proposed by Horn et al. Optical flow fields give an indication of both the magnitude and direction of movement. In Figure 3 the optical flow of a rotating cylinder (left) and sphere (right) are depicted. As can be seen, we can infer both speed and direction by looking at the length and angle of each so-called flow line depicted in the Figure, respectively. A combination of optical flow fields and a strong spatial representation can also be an effective way to perform AR [22] [29]. This is demonstrated by Simonyan et al., in which a two-stream CNN combines both spatial information and dense optical flow information to perform AR. In their approach, one stream performs convolutions on RGB frames and one

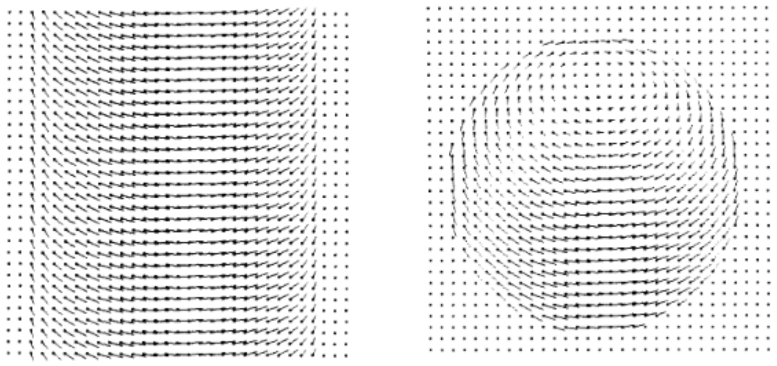
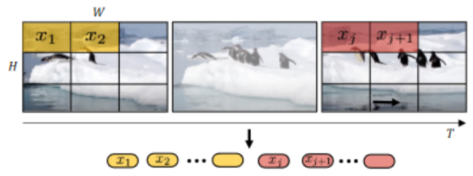
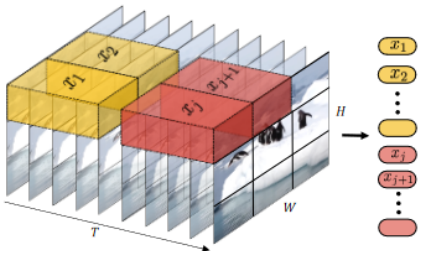


Figure 3: **Demonstration of Optical flow.** Figure by Horn et al.

stream performs convolutions on dense optical flows and the results are concatenated together. However, calculating optical flow over a set of frames is computationally expensive [8] and this makes this temporal representation ill-suited for performing real-time computer vision tasks.
A computationally cheaper but less accurate method is to calculate frame representation differences over time, capturing temporal information by subtracting representations at different time intervals. A frame representation does not necessarily have to be in simple RGB format, there are existing approaches that calculate differences over the spatial embeddings of frames [31]. These frame differences mainly highlight areas of change, without capturing the direction or magnitude of the motion. Calculating frame difference is computationally efficient and can typically be performed in real-time at negligible extra cost.
A more specific, transformer-oriented approach is the Video Vision Transformer (ViVit) as proposed by Arnab et al., which is a pure transformer approach for video classification based on the ViT transformer. First, the authors describe the way in which a single image is mapped to a sequence of tokens. This is similar to the way that the vanilla ViT model does so: by linearly projecting the patches of an image and by adding positional embeddings to each patch so that they retain positional information. Finally, the learnable classification token is prepended to the sequence, which serves as the final representation used by the classification layer. Given the way that an input image is embedded, the authors propose two ways to map a video to a sequence of tokens. The first way is to uniformly sample frames from a video. Each sampled frame is mapped to a sequence of tokens (as previously described) and these tokens are then concatenated to form the final video embedding. This can be seen in Figure 4-a. A second way in which



(a) ViVit Uniform sample video embedding



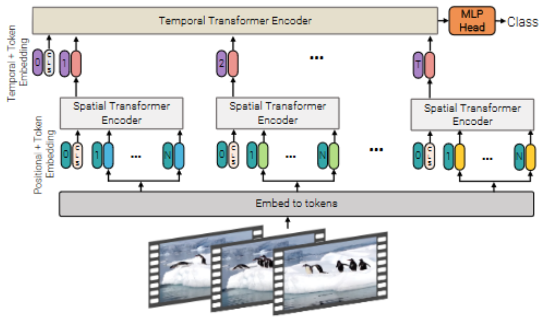
(b) ViVit Tubelet video embedding

Figure 4: **ViVit movie embedding approaches**. Figures by Arnab et al.

the ViVit transformer can embed a video is depicted in Figure 4-b. Non-overlapping, spatio-temporal "tubes" are extracted from the input volume, and these are then linearly projected to an embedding similar to that of the uniform frame sampling method. The authors then propose multiple ways to use this spatial-temporal embedding of a video for the task of classification. The simplest way is to simply forward all spatio-temporal tokens through the transformer encoder, called Spatio-Temporal Attention

(STA). Each transformer layer models all pairwise interactions between all spatio-temporal tokens and it

Figure 5: **ViVit factorised encoder.** Figure by Arnab et al.



Temporal Transformer Encoder
MLP Head
Class
Temporal + Token Embedding
Spatial Transformer Encoder
Positional + Token Embedding
Embed to tokens

thus models long-range interactions in the video. However, they state that the approach has quadratic complexity with respect to the number of tokens. This is because all pairwise interactions are modelled in each attention head. Therefore, a second method is proposed called the Factorised Encoder (FE), as depicted in Figure 5. This module consists of two transformer encoders: the spatial encoder processes tokens that are derived from the same temporal indices, therefore creating a spatial representation of each frame. The second encoder is dedicated to modelling interaction between each time step. The input to the second encoder are the concatenated classification (CLS) tokens of each frame. This effectively results in a "late fusion" approach, where spatial and temporal data are integrated at later stage of the analysis. The authors also present two additional modelling approaches, the Factorised Self-Attention Model and the Factorised Dot-Product Model, but report inferior performance compared to the Factorised Encoder. The authors observe that the Spatio-Temporal Attention outperforms the Factorised Encoder approach in terms of Top-1 accuracy but is significantly slower.

2.3 Language inclusive AR models

2.3.1 The vanilla CLIP model

Although the CLIP model as proposed by Radford et al. is designed to perform image classification, it is flexible enough to apply to a range of AR tasks. Typically, AR models integrate both temporal and spatial information to perform classification. In contrast, the vanilla CLIP model exclusively looks at spatial information and disregards temporal information entirely. For example, CLIP has no reliable way to discern a rotating screwdriver from a stationary screwdriver from a single frame alone. This is something that would be recognised by other AR models using methods such as optical flow fields or by looking at spatial changes over time. As shown in the paper, CLIP is evaluated on two industry-standard AR benchmarks: UCF101 and Kinetics700. The authors state for each video in that dataset, the middle frame is sampled and classified. The ViT-L/14 configuration as described in Section 2.1.2 obtains a Top-1 accuracy scores in linear probe and zero-shot evaluations that are comparable to some lower-end AR ResNet architectures on the Kinetics 700 dataset, with the linear probe accuracy being slightly higher than the zero-shot accuracy. By linear probe the authors mean the performance of a linear regression model that uses the produced CLIP embeddings as its input features. More specifically, it means that a (linear) fully connected layer is attached to the image embedding and serves as a decision head that provides logits for the possible output classes of a dataset. The evaluated performance therefore shows that CLIP embeddings are semantically rich and it also shows that they can be used for AR tasks by sampling and analysing only the middle frame of each video clip. Especially interesting is the fact that this single-frame AR approach is able to compete favourably with some older AR frameworks that perform classification using anywhere from 8 up to 64 frames [21].

2.3.2 VideoBERT

VideoBERT, as proposed by Sun et al. is a language-enabled video comprehension model. In this model, the authors have extended the BERT model to learn bidirectional joint distributions across sequences of both visual and linguistic tokens. These sequences are obtained from video data through a process of vector quantisation and from the outputs of ready-made speech recognition systems, respectively. The training data used in the paper includes datasets like "YouCook 2", that contains cooking videos and corresponding spoken instructions. For each input video, 30-frame clips that are non-overlapping (i.e. frame windows) are sampled from the video for all frames in the video. These clips are then fed into a pretrained CNN network (trained on Kinetics-700). The representation used for each clip are the feature activations before the final classification layer of the CNN and they apply average pooling to obtain a 1024-dimensional feature vector. Additionally, the authors extract speech from each video using an off-the-shelf automatic speech recognition (ASR) system. ASR provides a string transcript of speech in a video, and this transcript is tokenised using the standard BERT preprocessing steps. As such, the authors have both a visual and textual set of tokens that describe a video. The model is then trained

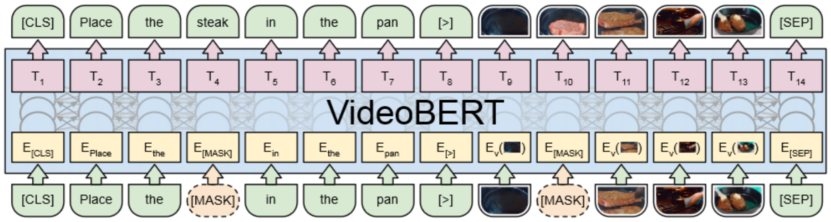

[CLS]
Place
the
steak
in
the
pan
[>]
[SEP]
T_1 T_2 T_3 T_4 T_5 T_6 T_7 T_8 T_9 T_{10} T_{11} T_{12} T_{13} T_{14}
VideoBERT
$E_{[CLS]}$ E_{Place} E_{the} $E_{[MASK]}$ E_{in} E_{the} E_{pan} $E_{[>]}$ E_v $E_{[MASK]}$ E_v E_v E_v $E_{[SEP]}$
[CLS]
Place
the
[MASK]
in
the
pan
[>]
[MASK]
[SEP]

Figure 6: **VideoBERT.** In this figure, VideoBERT performs masked token prediction, figure by Sun et al.

by calculating loss over three objectives: loss over textual masked objective, loss over the visual masked objective and loss over its the correspondence between visual and textual tokens. An example of this cross-modal masked objective is depicted in Figure 6. A sequence containing two $[MASK]$ indicators is passed into the model. Notice that this sequence contains both visual tokens and textual tokens, separated by the ">" token. The output is then presented in the top half of Figure 6 and we see that the model has predicted both a textual token and a visual token for the masked objectives. After training, the model can be applied to a variety of downstream tasks. The authors state two main use cases:

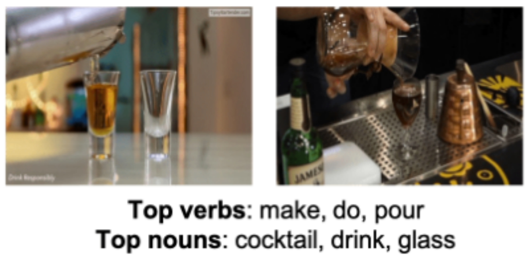

Top verbs: make, do, pour
Top nouns: cocktail, drink, glass

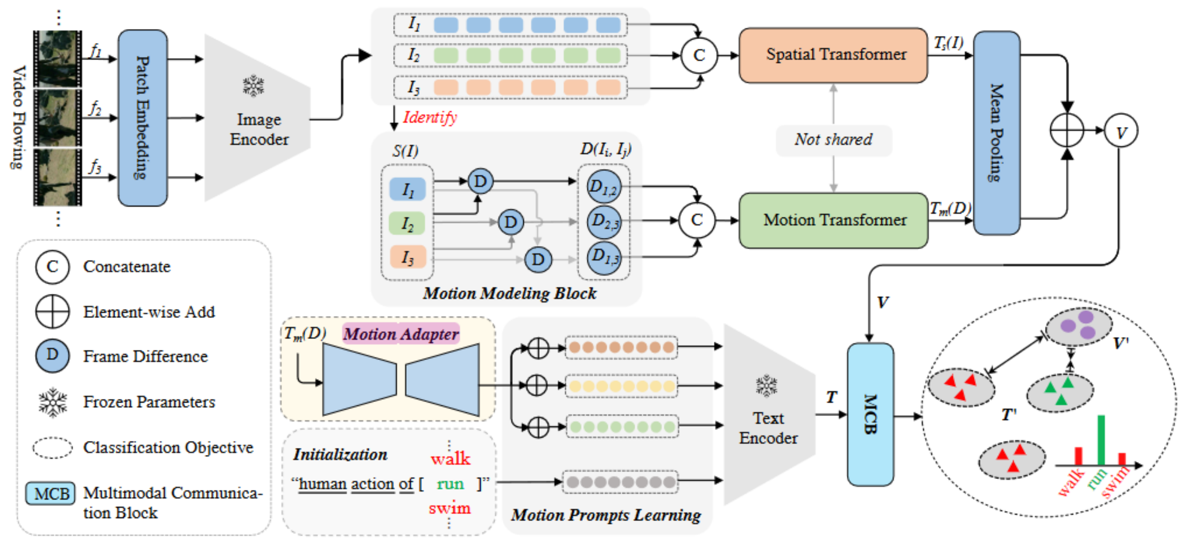
Figure 7: **Pretrained VideoBERT model performing masked prediction.** Figure by Sun et al.

predicting masked symbols and using the output $[CLS]$ token as a feature representation of the entire input for supervised classification. For the first case, the authors present a basic *cloze* prompt for which the model has to predict the masked objective. For example, the prompt "now let me show you how to $[MASK]$ the $[MASK]$, along with an input video from the YouCook II dataset is presented to the pretrained videoBERT model. The model is then asked to predict the masked objective. By doing so, the most likely verb and noun are the predictions for the first and second masked objective, respectively. An example of this capability can be seen in Figure 7. One way videoBERT could be applied to AR tasks is to pretrain it on an AR dataset like Kinetics-700 and use its produced $[CLS]$ token for classification. However, as the Kinetics dataset (and most other recognised AR datasets) does not contain explicit narration, the model would have to be trained visual-only and would therefore likely not function as well as with cross-modal training data. VideoBERT could also be used for AR tasks in a "zero-shot" capacity on diverse datasets without additional fine-tuning. However, its effectiveness may be constrained by the specificity and diversity of its pre-training data. If the pre-training data, such as the "YouCook 2" dataset, is very specialised, the model may not generalise well to completely different types of videos or actions that were not represented in the training set. The model's performance on a new dataset will depend on the similarity between the new data and the data VideoBERT was originally trained on.

2.3.3 Seeing in Flowing

An AR extension to the CLIP model is presented in the paper 'Seeing in Flowing: Adapting CLIP for AR with Motion Prompts Learning' by Wang et al. The paper builds on the original CLIP framework in order to fit it to AR tasks. This network utilises a temporal transformer and a spatial transformer, and this is depicted in Figure 8. Given a set of frames, the motion transformer of the network looks at differences between frame representations (embeddings) in the Motion Modeling Block given a hyper-parameter that sets the maximum temporal interval in-between frames. This information is then fed into a Motion Transformer that produces a motion embedding. This information is then combined with the output of the spatial transformer using a mean pooling operation. At the same time, the Motion Modeling Block presents its output to a so-called Motion Adapter that learns to fine-tune the prompt passed into the frozen CLIP text encoder. As a result, the model learns not only to capture spatial and temporal information, but it also learns to adapt its prompts based on the motion that is observed to improve accuracy. Therefore, there is a heavy focus on spatial information, spatial differences over time and suitable prompts to match. The authors claim that this is particularly beneficial given the importance of informative prompts in the CLIP model. They verify the importance of the novel Motion Adapter by performing an ablation study. From this study it becomes clear that the model performs significantly better when utilising dynamic learning of prompts using the Motion Adapter. This underscores the critical role of semantically-rich prompts for the network [31].

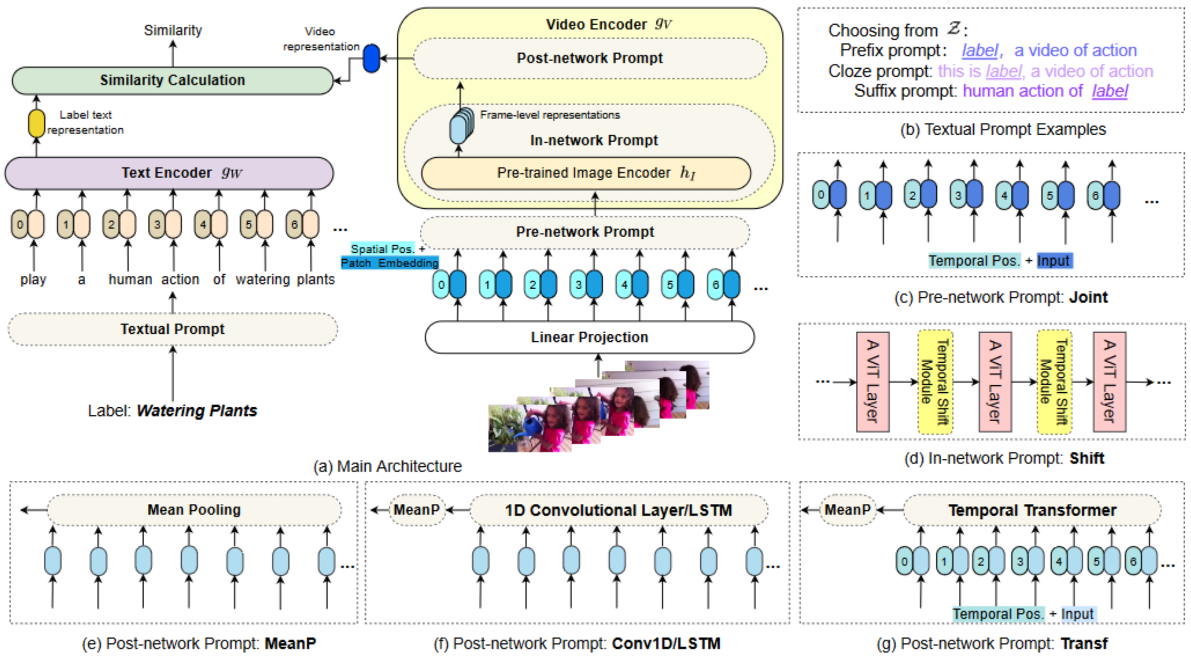
Figure 8: **Seeing In Flowing model architecture**. Figure by Wang et al.



2.3.4 ActionCLIP

Another implementation is that of ActionCLIP. ActionCLIP is an AR model proposed by Wang et al. that performs AR in videos by applying both a spatial and temporal encoder, as previously proposed by Bertasius et al. The spatial encoder, which is a pre-trained CLIP model, encodes each frame of a video separately. The encodings for each frame are concatenated and fed into a temporal encoder that models interactions between tokens from different temporal indices (that is, frames). ActionCLIP also employs Temporal Shift modules as introduced by Lin et al. in order to exchange information among neighbouring input frames. It does this by shifting the input channels of certain layers in the network along the temporal dimension. This allows for an exchange of information across frames in a video sequence. ActionCLIP differs from the Seeing In Flowing model in a few important ways. Firstly, actionCLIP performs a late fusion of spatial and temporal information. That is, ActionCLIP produces spatial embeddings for each frame, then utilises a Post-network prompt to extract temporal information from this sequence of spatial frame embeddings. Seeing in Flowing uses a two-stream approach instead. It first produces spatial embeddings for a sequence of frames, then feeds this information into a spatial stream and a temporal stream. The results of the two streams are then combined by a Mean Pooling module. Another difference is that ActionCLIP directly feeds its spatial embeddings into its Post-network prompt. Seeing In Flowing extracts frame representation differences between a set of frames in the Motion Modelling Block and passes these into the Motion Transformer instead.
Interestingly, ActionCLIP offers four interchangeable Post-network architecture choices that convert frame-level representations from the Video Encoder (g_v) to a video representation: 'Transf', 'Conv1D', 'LSTM' and 'MeanP'. The entire architecture overview is depicted in Figure 9. Given its relevance to

Figure 9: **The ActionCLIP architecture.** Figure by Wang et al.

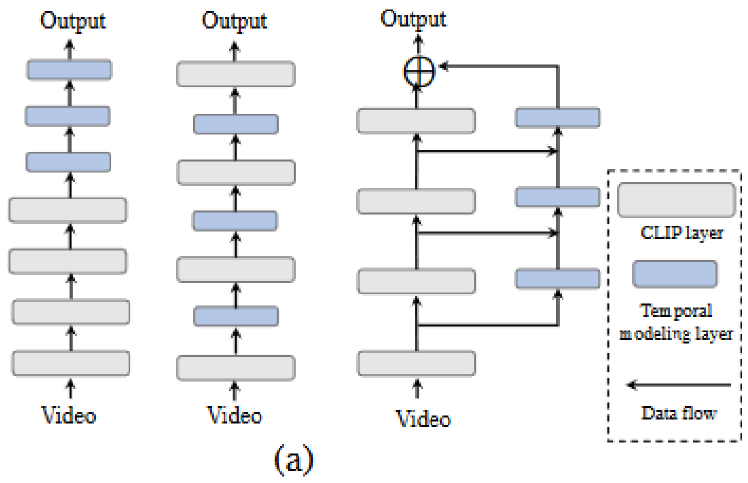

Similarity
Video representation
Similarity Calculation
Label text representation
Text Encoder g_W
play a human action of watering plants
Textual Prompt
Label: Watering Plants
Video Encoder g_V
Post-network Prompt
Frame-level representations
In-network Prompt
Pre-trained Image Encoder h_I
Pre-network Prompt
Spatial Pos. + Patch Embedding
Linear Projection
(a) Main Architecture
Choosing from $\mathcal{Z}$:
Prefix prompt: label, a video of action
Cloze prompt: this is label, a video of action
Suffix prompt: human action of label
(b) Textual Prompt Examples
Temporal Pos. + Input
(c) Pre-network Prompt: Joint
A ViT Layer
Temporal Shift Module
(d) In-network Prompt: Shift
Mean Pooling
(e) Post-network Prompt: MeanP
MeanP
1D Convolutional Layer/LSTM
(f) Post-network Prompt: Conv1D/LSTM
Temporal Transformer
(g) Post-network Prompt: Transf

our literature study, we will briefly discuss each method. the 'MeanP' method performs mean pooling on the temporal dimension. More specifically, it takes the average of all frame embeddings. The idea behind mean pooling is that its output more or less represents an average representation of all frames in the video. 'Conv1D' is a 1D convolutional layer applied on the temporal dimension. Frames are sequentially concatenated side-by-side and 1D convolutional filters slide over this representation. The filter has a certain length, defined by the kernel size of the convolution. For example, a kernel size of 3 means the filter operates on 3 consecutive frames at a time. Its height is equal to the amount of features in each frame's representation. The advantage over performing mean pooling is that it captures patterns in between sequential frames. This way, less information is discarded than when simply averaging over them. The 'LSTM' method is simply a recurrent neural network, which is a network designed to remember long-term dependencies in sequential data. It processes the frame embeddings one-by-one, updating its internal state based on the current frame and its memory of previous frames. In other words, its output depends not only on the input frame but also on the previous output(s). This allows it to capture patterns like human actions that span multiple frames by combining observations from previous frames with the current frame. Finally, the 'Transf' method uses a temporal vision transformer. Each frame embedding is treated as a patch similar to how images are tokenised in CLIP. These tokens are then passed through the transformer encoder layers. The self-attention mechanism allows the transformer to weigh the importance of different frames relative to each other, finding patterns between them. Output comes in the form of a single [CLS] token similar to how CLIP assigns an embedding to an image. This token therefore aggregates information from all frames. Each of these four methods essentially combines the set of frame-level representations into a single video embedding. As a result, ActionCLIP models the patterns in the absolute sequence of spatial frame representations (and not over the differences between them). According to their Post-network prompt evaluation, the most effective way to do so is the temporal transformer that applies global attention over the set of spatial embeddings. The authors show that this temporal transformer outperforms the other three Post-Network modules on the Kinetics-400 benchmark. However, the authors make no mention of the change in runtime performance between these post-network modules. For example, transformer layers are much more computationally expensive than a simple mean pooling module. Therefore, although it is shown that the transformer-based Post-network prompt obtains higher Top-1 accuracy, we do not know its impact on the runtime performance of the network.

2.3.5 Spatio-Temporal Auxiliary Network (STAN)

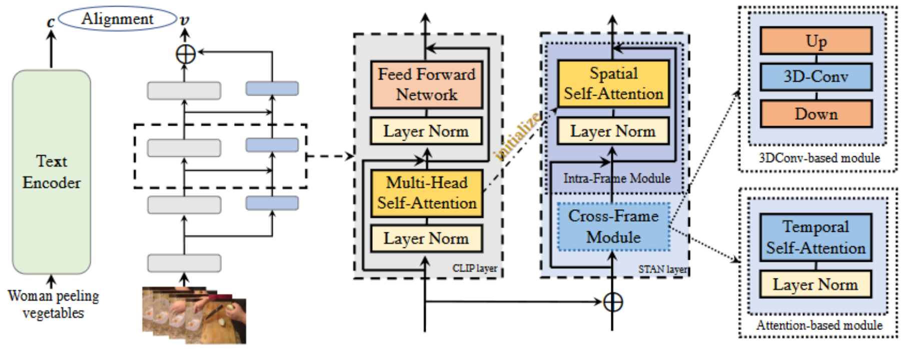
This study, proposed by Liu et al., separates the spatial and temporal dimensions of a video. The authors state that thanks to the branch structure, STAN augments the features of video frames with spatiotemporal contexts at increasingly deep CLIP transformer layers without affecting the forward-propagating of CLIP itself. The structures in Figure 10 are posterior (left), intermediate (middle) and the STAN structure (right), respectively. The specific architectural choices can be seen in Figure 11. In the posterior structure, a sequence of frames is first passed into the complete set of CLIP layers in order to create a complete set of spatial CLIP embeddings for a video. This set of frame embeddings is then passed into a set of temporal modelling layers to extract temporal information. In the intermediate structure (middle) , the model uses interchanging temporal modelling layers and CLIP layers. This means that in each layer of the network, spatial embeddings are calculated for each frame (or frame embeddings from a previous layer) and this set of new embeddings is then passed into a temporal modelling layer. Lastly, the STAN structure (right) causes separation of the spatial and temporal representations. More specifically, the spatial embeddings affect the temporal representation at every layer, but the temporal embedding never interferes with the pretrained CLIP model. The authors claim that this is a big advantage over the intermediate model structure (left) because CLIP's capacity to produce semantically rich spatial embeddings is not reduced or distorted by a combination of temporal and spatial information [16]. They also claim that it is an advantage over the posterior structure (left) because as opposed to the posterior structure, this architecture allows the temporal modelling layer to learn not only from the final spatial embeddings but also from the intermediate spatial embeddings produced by the CLIP model. ActionCLIP and Seeing in Flowing both mostly follow the posterior structure (left) because they pass input into multiple visual CLIP layers before passing them into their temporal modelling module. For example, the first step in the Seeing in Flowing model is to pass the input frames through the entire CLIP (visual) encoder. In contrast, the STAN architecture provides spatial information from every intermediate CLIP layer to its temporal modelling layer.
For our purpose, the Cross-Frame module in the STAN-layer is most relevant because it models the

Figure 10: **Different CLIP-based AR frameworks.** Figure by Liu et al.



temporal information across frames. The authors propose two separate ways to do so: by using either 3D convolutions or Temporal Self-Attention. For Temporal Self-Attention, it processes a set of frames by stacking each patch embedding at the same spatial location and by then passing that stack into the Temporal Self-Attention. By doing so, the model captures temporal information at a given spatial location and each patch is contextualized with temporal information at that same location. The second approach, using 3D convolutions, stacks the patch embeddings to form a 3D 'feature cube' on which convolutions are performed. The authors provide extensive experimentation on both of these cross-frame modelling apparoaches. In their results, they describe that the Temporal Self-Attention module outperforms the 3D CNN approach by a large margin and performs slightly fewer GFLOPs (floating point operations) to do so. Thus, the authors conclude that the Temporal Self-Attention module is the best choice in modelling Cross-Frame information.

Figure 11: **The STAN model architecture.** Figure by Liu et al.



2.3.6 Concluding observations on language-enabled models

The language-enabled models we researched in this study each present a unique way to perform language-enabled AR. We have seen that the VideoBERT model is a language-enabled, cross-modal model that can be applied to the downstream task of AR. However, it is an older model (2019) that does not report performance on AR benchmarks such as Kinetics or HMDB51 directly. What is more, the model is trained to predict sentences with an open vocabulary, not to distinguish between a finite set of prompts. This has its purposes (like captioning video content), but makes evaluation on datasets with fixed ground-truth labels such as HMDB51 and Kinetics more challenging. The authors do not report Top-1 and Top-5 performance on datasets like Kinetics-400 or HMDB51, but we suspect the predictive performance to be much lower than the other (much more recent) CLIP-based AR models that we studied. This is in part because it is not the main purpose of the model, and also because assessing whether VideoBERT's open-vocabulary predictions match the ground truth label is a open problem in itself, as mentioned by the authors. ActionCLIP, Seeing in Flowing and STAN are CLIP-based AR models that combine the semantically rich CLIP embeddings with temporal information in order to perform AR tasks. We outline a concise overview of key findings in our area of interest. More specifically, we look at the ways in which the models differ in how they incorporate CLIP's visual embeddings in their architecture in order to perform AR tasks. Given its relevance to our research scope, we also look at the computational complexity of each model and how each component contributes to that. The approach proposed in Seeing in Flowing essentially employs a two-stream architecture in which motion and spatial information are fused together by applying mean pooling. This means that temporal information and spatial information are processed in parallel. In contrast, the ActionCLIP model first spatially encodes each frame and then looks for changes in this spatial representation over time. Additionally, the Seeing in Flowing model uses the output of the Motion Modelling Block to shape the prompts presented to the model. ActionCLIP and STAN both use a fixed template in which the word describing the action to be recognised is passed in and do nothing to further shape these prompts later on in the model. STAN differs from both Seeing in Flowing and ActionCLIP in that it does not pass a set of frames through all visual CLIP (transformer) layers in the model at once. It feeds a set of frames into the first visual CLIP layer, obtains their embeddings and passes them to the temporal modelling layer. STAN then moves on to the next layer. It does this for every visual CLIP layer in the model, therefore obtaining more granular information than just the final visual CLIP embedding. Its architecture bears some resemblance to the Seeing in Flowing architecture in that they both model spatial and temporal information in a parallel, two-stream way. However, whereas Seeing in Flowing performs a late fusion of the two streams, STAN exchanges information from the spatial layer to the temporal layer in between each visual CLIP layer. It should be noted that this information exchange strictly moves from the spatial stream to the temporal stream and not vice versa. Another way in which all three CLIP-based models differ is the way in which temporal information is extracted. Both STAN and ActionCLIP note that temporal self-attention is the best performing way to model temporal information, although each model applies this component in its own way. Seeing in Flowing uses mean pooling over the output of its Spatial Transformer and Motion Transformer components instead, and provides no ablation study that compares its performance to a temporal self-attention module. Runtime performance wise, STAN uses 1×3 views in its reported benchmarks whereas ActionCLIP only uses a single view. This means that STAN essentially performs three separate inference passes using three randomly selected crops and averages those results. In assessing the runtime performance of the two models, we take this into account. On average, we have that STAN sampling 16 video frames performs $\frac{1187}{3} = 395.67$ GFLOPs per view whereas ActionCLIP sampling 16 video frames performs 281 GFLOPs. Unfortunately, there is no clear mention of GFLOPs and view amount in the Seeing In Flowing paper. This makes comparing runtime complexity to the other models challenging. However, spatial and temporal information is fused using a mean pooling operation instead of a (typically very costly) temporal self-attention operation over a set of frames. We therefore speculate that the Seeing in Flowing model would likely be slightly faster than the STAN model and the Seeing in Flowing model, provided that all three models use the same number of (visual) CLIP layers, the same patch size and an equal number of views. In this estimation, we assume that the additional Motion Modelling Adapter of the Seeing in Flowing model does not significantly impact its performance. Finally, we also recognise the fact that all three CLIP-based models are evaluated by sampling video frames of the validation set evenly distributed over an entire video clip. The validation set performance achieved this way is not representative of the validation set performance that would be achieved in a real-time context. Currently, 8, 16 or 32 frames are sampled uniformly over an entire video clip that can range anywhere from 40 to 600 frames. This means that the window of sampled frames is much larger than in a real-time scenario, and this improves performance because a larger action window is taken into account. In contrast, sampling a similarly sized action window in real-time would mean waiting for those 600 frames to come in before performing inference, adding a $\frac{600}{32}$=18.75 second delay for a camera running at 32 frames per second. Obviously, this is not a viable option. We therefore keep in mind that Top-1 and Top-5 validation performance in a real-time context will likely be lower than reported in each model's paper.

2.4 Transformer distillation

2.4.1 Vanilla knowledge distillation

The classical use case of knowledge distillation (KD) is to create a secondary, smaller version of a model that learns to mimic the way in which the original model makes its decisions by training the student model to predict the output distribution (the so-called soft targets) of the teacher model. This stands in contrast to predicting the hard targets (or hard labels) of the teacher, which are the actual labels assigned to a sample by the teacher model. The goal is to find an appropriate speed-accuracy trade-off that suits the use-case of the student model. The reason that the student model predicts the soft target and not just the correct label is because more knowledge is contained in an entire output distribution than in simply the *argmax* of that distribution (i.e. its final output) [6]. KD can be an effective way to downscale transformers, as shown by Habib et al. In this paper, KD is compared to other complexity reduction techniques like weight quantisation, pruning and weight multiplexing, which typically involve complex pipelines for performing the compression. The paper concludes that in many cases, KD is a simple yet effective model compression technique that allows a relatively simple model to perform tasks almost as accurately as a complex model. Another advantage is that the complexity of the student model can be determined beforehand. That is, the student model can be initialised as large or as small as the task at hand requires to find an optimal speed-accuracy trade-off for the student model. Figure 12 demonstrates the relation between the student and teacher in applying vanilla knowledge distillation.

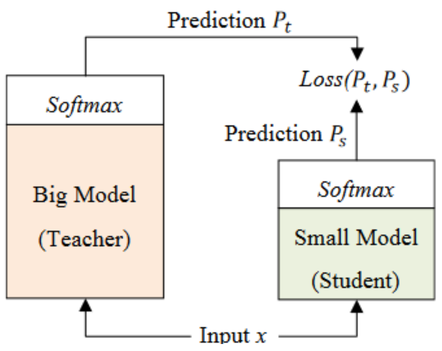


Figure 12: **Vanilla knowledge distillation.** As can be seen in this figure, the larger teacher model provides a prediction on an input sample x to serve as a soft label for the student model. By doing so, the student learns to mimic the teacher model. Figure by [2].

2.4.2 Patient knowledge distillation

An interesting per-layer approach for knowledge distillation between a teacher and a student transformer is described in "Patient Knowledge Distillation for BERT Model Compression" by Sun et al. As opposed to the previously described distillation approaches, which use only the last layers of the teacher and student model, their model learns from multiple intermediate layers of the teacher model for incremental knowledge extraction. The authors propose two PKD (Patient Knowledge Distillation) strategies: PKD-Last: learning from the last k layers and PKD-Skip: learning from every k layers. These are illustrated in Figure 13. In PKD-Last (13-b), the student learns from the last k layers of the teacher, whereas in PKD-Skip (13-a) the student learns from every k layers. Equation 1 represents the distillation loss,

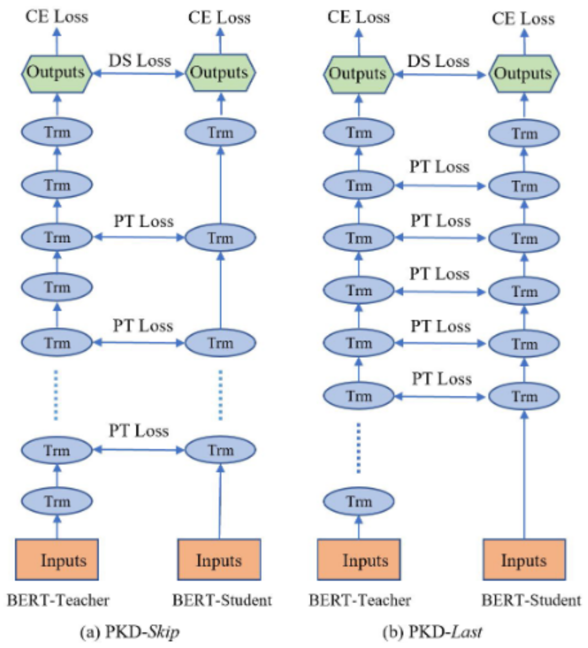


Figure 13: **Overview of the PKD strategies.** Figure by Sun et al.

where we sum over all data samples and every possible prediction class, as indicated by $\sum_{i\in[N]}$ and $\sum_{c\in C}$. Next, $p^t(y_i = c|x_i;\theta^t)$ and $\log p^s(y_i = c|x_i;\theta^s)$ provide the student and teacher prediction probabilities for a specific sample and class. This allows the authors to minimise the difference between the teacher's predicted probability distribution P_t and the student's predicted probability distribution P_s for all inputs and class labels. This is achieved by using the cross-entropy loss between the teacher's predictions (soft labels) and the student's predictions. This loss function therefore encourages the student to imitate the soft labels of the teacher model.

$$L_{DS} = -\sum_{i\in[N]}\sum_{c\in C} p^t(y_i = c|x_i;\theta^t)\log p^s(y_i = c|x_i;\theta^s) \tag{1}$$

The authors also choose to fine-tune the model on the ground truth labels using cross-entropy loss as depicted in Equation 2. This is similar to Equation (1) except for the fact that the model predictions are compared to ground-truth labels instead of the teacher's soft labels. That is, $1[y_i = c]$ will only be '1' if the true label of sample i is c. Loss is then added for the difference between the ground truth label and the probability assigned to that label by the student.

$$L^s_{CE} = -\sum_{i\in[N]}\sum_{c\in C} 1[y_i = c]\cdot\log P^s(y_i = c|x_i;\theta^s) \tag{2}$$

Equations 1 and 2 are common distillation loss terms that exclusively consider the soft label and ground truth differences, respectively. Interestingly, PKD introduces an additional loss term that focuses on student-teacher convergence throughout multiple layers in the network. Similar to the vision transformer used in the CLIP model, the BERT model prediction is performed by using only the $[CLS]$ classification token of the output layer, in which the entire sequence is encoded [24]. In order to improve distillation performance, the authors propose an additional loss term in the form of the mean squared error between the $[CLS]$ token embedding of teacher and student over different layers. Intuitively, this means that the difference between intermediate embeddings over time are compared between teacher and student. In contrast to soft-target or ground truth loss, this loss is not calculated by comparing the final layers of the teacher and student networks. Instead, it is calculated throughout different layers of the two networks as the embedding develops in each model when a training sample is passed in. This is shown in Equation 3. M denotes the number of layers in the student network, N is the number of training samples, and the superscripts s, t indicate student and teacher network respectively. Note that each j-value of ht is mapped through ${}_{pt(j)}$ for a given training sample. This is to account for different layer counts between teacher and student because the teacher model is typically much larger than the student.

$$LP_T = \sum_{i=1}^{N} \sum_{j=1}^{M} \left(\frac{hs_{i,j}}{\|hs_{i,j}\|_2} - \frac{ht_{i,Ipt(j)}}{\|ht_{i,Ipt(j)}\|_2} \right)^2 \tag{3}$$

Finally, we arrive at the total loss function LP_{KD}, in which the three loss equations 1, 2 and 3 are weighted by hyper-parameters α and β and then summed to obtain the total loss between the teacher and the student, as depicted in Equation 4.

$$LP_{KD} = (1 - \alpha)Ls_{CE} + \alpha LDS + \beta LP_T \tag{4}$$

Clearly, this proposed distillation technique provides a more comprehensive teacher-student mapping than simply calculating loss over the soft-targets of the teacher. It forces the student not only to mimic the teacher's output, but also penalises it for deviating from the teacher classification token as it develops in the student model [24]. From the results presented in the paper it becomes clear that PKD (using Skip) outperforms KD (vanilla knowledge distillation) and FT (no distillation but with finetuning) in almost all cases. In relation to our research goal, it should be noted that the proposed method can only be applied to transformer-to-transformer distillation because it assumes that each layer in both teacher and student contains a classification embedding for a given input 4, which prohibits using PKD for distilling a transformer into a CNN-based architecture.

2.4.3 TinyCLIP distillation

TinyCLIP: CLIP Distillation via Affinity Mimicking and Weight Inheritance, written by Wu et al. proposes a cross-modal language-image distillation method called TinyCLIP. The TinyCLIP method uses two core techniques: affinity mimicking and weight inheritance. Affinity mimicking explores specific interactions between teacher and student with regards to the cross-modal feature alignment. Weight inheritance deals with inheriting pre-trained weights from the teacher model to the student model to improve distillation efficiency. Finally, they describe a multi-stage progressive distillation scheme to mitiage the loss of informative weights during transmission from teacher to a (much) smaller student. In this strategy overview, we pay particular attention to the Affinity Mimicking mechanism because it is most relevant to our study because Affinity Mimicking deals with transmitting information from teacher to student. As we have seen in Section 2.1.2, CLIP consists of both an image encoder and text encoder. The visual and textual representations are projected into a cross-modal embedding space by these encoders by minimising a contrastive loss between the two representations. To better capture the teacher's embedding space, TinyCLIP introduces two new loss functions: image-to-language loss and language-to-image loss. The former, $\mathcal{L}_{I2T}$ compares the affinity between teacher and student of the image-to-language affinity and the latter, $\mathcal{L}_{T2I}$ the language-to-image affinity. The terms A_{T2I} and A_{I2T} for both teacher (t) and student (s) are described in Equations 5 and 6, respectively.

$$A_{T2l}(i, j) = \frac{\exp(I_i \cdot T_j/\tau)}{\sum_{k \in \mathcal{B}} \exp(I_i \cdot T_k/\tau)}, \tag{5}$$

$$A_{I2T}(i, j) = \frac{\exp(I_i \cdot T_j/\tau)}{\sum_{k \in \mathcal{B}} \exp(I_k \cdot T_j/\tau)}. \tag{6}$$

The matrix A_{I2T} depicts the similarity of each sample k in batch $\mathcal{B}$ to each text embedding in that same batch. This matrix is obtained by multiplying the image embedding matrix by the text embedding matrix. The softmax operation is then performed on the values in the matrix. This means that each image embedding gets a similarity score for every text embedding in the batch and that the similarity scores for each image embedding sum to one. Matrix A_{T2I} is exactly the opposite matrix. That is, each text embedding gets a similarity score for every image embedding in the batch and the scores for each text embedding also sum to one. This can also be seen in Figure 14. In this figure, each yellow row depicts

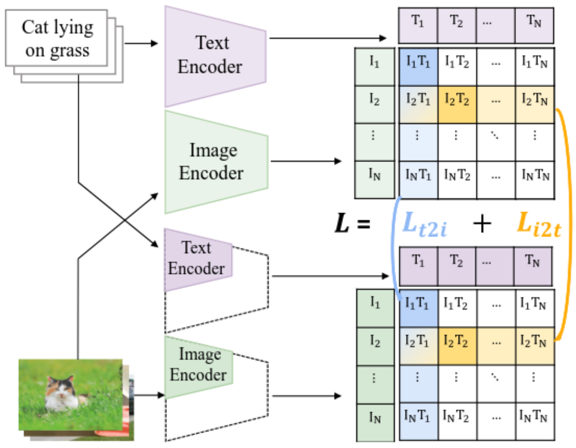


Figure 14: **TinyCLIP affinity mimicking.** This schematic illustrates the affinity mimicking strategy. Cross-entropy (CE) is calculated between both text-to-image and image-to-text logits of the teacher and student. Figure by Wu et al.

an image-to-text similarity and each blue column a text-to-image similarity. In training a student model, the student is tasked to mimic the soft labels of the teacher for both the image-to-text and text-to-image task. This way, it learns to form the same affinity (i.e. its tendency to match specific image-text pairs together) between image-text and text-image pairs. This is shown in Equation 7.

$$\mathcal{L}_{\text{distill}} = \mathcal{L}_{12T} + \mathcal{L}_{T2l}, = CE(A^s_{12T}, A^t_{12T}) + CE(A^s_{T2l}, A^t_{T2l}). \tag{7}$$

The total Affinity Mimicking loss is quantified as the sum of the cross-entropy loss between the A_{T2I} (text-to-image) affinity plus the A_{I2T} (image-to-text) affinity similarity between teacher and student. Intuitively, this means that the student model is punished for having different image-to-text and text-to-image similarity scores than the teacher and is therefore supervised to mimic these as closely as possible. The authors also provide a strategy to inherit pretrained weights between teacher and student, therefore speeding up the distillation process because not all student model weights have to be trained from scratch. They propose two ways to do this: manual inheritance and automatic inheritance. For the manual inheritance, the authors analyze weight redundancy of existing CLIP pre-trained models. They do so by calculating the cosine similarity between the input and output embedding of each layer. They then uniformly select k layers of the text and image branch of the original model to serve as the initialisation for the smaller student models. A second way to do this is to perform automatic weight inheritance. This removes the need for manual inspection of the model weights before inheritance, thus making it more practical for a diverse set of models. They do this by applying a learnable mask on both the vision and text encoder of the model. This mask learns to prune certain parts of the teacher's weights by operating under a size constraint, guiding it to keep only the most important weights. This subset of weights is then transferred to the student model as its initial weights. The authors show that automatic inheritance of weights beat manual inheritance and that both inheritance methods significantly reduce the time it takes to distill a large teacher model. They also propose a strategy called Progressive Multi-Stage

Method	Image Encoder	Text Encoder depth	Text Encoder width	#Params (M) Image+Text	MACs (G)	Throughput (pairs/s)	Training datasets	IN-1K top1 acc (%)	Flickr30k I → T@1	Flickr30k T → I@1	MSCOCO I → T@1	MSCOCO T → I@1
Training data: 15M / 20M												
RILS	ViT-B/16	12	512	86 + 38	20.5	818	LAION-20M	45.0	45.1	34.9	32.2	25.5
MaskCLIP	ViT-B/16	12	512	86 + 38	20.5	818	LAION-20M	46.6	64.9	48.1	38.5	24.8
SLIP	ViT-B/16	12	512	86 + 38	20.5	818	YFCC-15M	42.8	57.6	40.1	31.1	20.3
CLIP	ViT-B/16	12	512	86 + 38	20.5	818	YFCC-15M	37.6	51.6	32.2	26.5	17.1
TinyCLIP (**Ours**)	ViT-39M/16	6	512	39+19(2.1×)	9.5(2.2×)	1,469(1.8×)	YFCC-15M	63.5	84.4	66.7	54.9	38.9
TinyCLIP (**Ours**)	ViT-8M/16	3	256	8+3(11.3×)	2.0(10.3×)	4,150(5.1×)	YFCC-15M	41.1	62.3	42.3	36.2	21.5
Training data: 400M												
Florence [37]	DaViT-5M	6	256	5+5	1.1	2,980	LAION-400M	45.0	61.2	41.5	36.2	20.9
TinyCLIP (**Ours**)	DaViT-5M	6	256	5+5	1.1	2,980	LAION-400M	50.0	66.2	48.5	40.7	24.5
CLIP [21]	ResNet-101	12	512	56 + 38	12.8	1,161	WIT-400M	62.2	78.1	59.2	49.3	30.7
CLIP [21]	ResNet-50	12	512	38 + 38	9.1	1,549	WIT-400M	59.6	81.2	58.2	50.8	28.3
TinyCLIP (**Ours**)	ResNet-30M	9	512	30+29(1.3×)	6.9(1.3×)	1,811(1.2×)	LAION-400M	59.1	81.1	61.2	52.7	33.9
TinyCLIP (**Ours**)	ResNet-19M	6	512	19+19(2.0×)	4.4(2.1×)	3,024(2.0×)	LAION-400M	56.4	76.2	58.3	48.9	30.9
OpenCLIP [9]	ViT-B/32	12	512	88 + 38	7.4	2,452	LAION-2B	65.7	84.7	66.8	56.9	39.3
CLIP [21]	ViT-B/32	12	512	88 + 38	7.4	2,452	WIT-400M	63.2	80.1	59.8	51.2	30.6
OpenCLIP [9]	ViT-B/32	12	512	88 + 38	7.4	2,452	LAION-400M	62.9	79.3	62.0	53.3	35.4
TinyCLIP (**Ours**)	ViT-61M/32	9	512	61+29(1.4×)	5.3(1.4×)	3,191(1.3×)	LAION-400M	62.1	78.6	63.3	53.9	35.9
TinyCLIP (**Ours**)	ViT-40M/32	6	512	40+19(2.1×)	3.5(2.1×)	4,641(1.9×)	LAION-400M	59.7	77.3	58.9	49.8	33.1
TinyCLIP (**Ours**)	ViT-63M/32	auto	auto	63+31(1.3×)	5.6(1.3×)	2,905(1.2×)	LAION-400M	63.9	83.2	64.4	55.5	37.6
TinyCLIP (**Ours**)	ViT-45M/32	auto	auto	45+18(2.0×)	3.7(2.0×)	3,682(1.5×)	LAION-400M	61.4	80.9	62.2	52.8	34.7
TinyCLIP (**Ours**)	ViT-22M/32	auto	auto	22+10(3.9×)	1.9(3.9×)	5,504(2.2×)	LAION-400M	53.7	71.3	52.0	44.4	28.3
TinyCLIP (**Ours**)	ViT-63M/32	auto	auto	63+31(1.3×)	5.6(1.3×)	2,909(1.2×)	LAION+YFCC-400M	64.5	84.9	66.0	56.9	38.5
TinyCLIP (**Ours**)	ViT-45M/32	auto	auto	45+18(2.0×)	3.7(2.0×)	3,685(1.5×)	LAION+YFCC-400M	62.7	80.3	63.9	54.0	36.7

Table 1: **Comparison with the state-of-the-art methods.** ”auto” denotes automatic weight inheritance, and the other models use manual weight inheritance. Table by Wu et al.

Distillation that decreases the amount of weights in the teacher step-by-step rather than directly. That is, they compress the teacher's weights by a modest amount in each step for multiple steps (e.g. 25%) rather than directly discarding the majority of the weights. They do this to allow for distilling very large models into much smaller student models. If the weights would be inherited without this step-by-step approach, almost all weights would immediately be discarded and the student models often failed to converge. The resulting architecture variants are shown in Table 1. From this table it becomes clear that the distillation techniques proposed in the TinyCLIP paper are a surprisingly effective way to efficiently distill a large CLIP model into a smaller, well-performing student variant.

2.4.4 Data-efficient image transformer distillation

The data-efficient image transformer distillation (DeiT) introduced by Touvron et al. introduces a teacher-student strategy specifically designed for transformers. It relies on a distillation token that ensures that the student learns from the teacher through attention. The way this works is that aside from the class token and positional embeddings that are normally added to the patches of an image before being passed into a ViT, the authors also add a so-called distillation token. This is illustrated in Figure 15. The token interacts with the global attention layers the same way as the class token, but the two have different loss calculations. The class token calculates loss over the ground truth label of the image, whereas the distillation token calculates loss over the hard label provided by the teacher. The authors state that they have also tried to calculate the distillation token loss over the soft labels provided by the teacher but they found this to be less effective. The results of the DeiT approach are presented in Table 2. In many cases, the number of parameters present in a model is halved at no cost in performance. One explanation the authors provide for the effectiveness of this method is because the teacher model can compensate for image crops that clash with the ground truth label. For example, in a picture that is labelled as “a flower', a random crop could consist of only the stem of that flower. Then, the ground truth label would be a deceptive supervisory signal, but the teacher model would be able to provide the correct label. It should be noted that, to the best of our knowledge, this distillation strategy has not yet been applied to the distillation of video transformers. This means that its applicability to these models is not known. There is also no mention of its application on cross-modal vision transformers such as CLIP. However, there is no conceptual difference between adding such a distillation token to a ViT or a video transformer because the former operates on patch-based tokens and the latter on frame-based tokens. That is to say, we suspect that a distillation token can be added to the AR transformers in much the same way as it is described in this paper to allow for experimentation with the DeiT distillation strategy on video transformers. We therefore think of this approach as a suitable candidate for further experimentation.

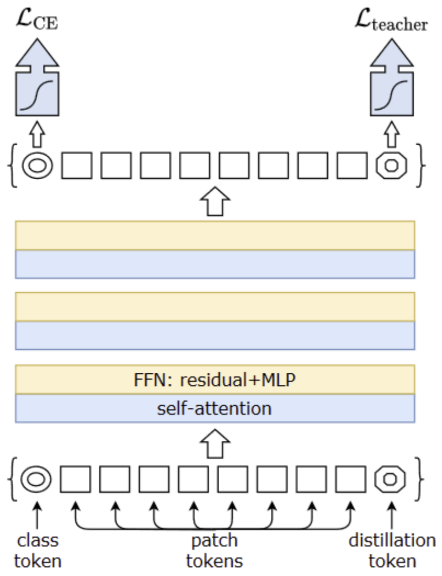


Figure 15: **Illustration of the added distillation token.** The rest of the model functions as normal. Cross-entropy loss between the prepended class token and ground truth labels are calculated. At the same time, cross-entropy (CE) loss between the appended distillation token and the hard teacher label is also calculated. Loss is split equally in a 50/50 fashion. Figure by Touvron et al.

Network	#param.	image size	throughput (image/s)	ImNet top-1	Real top-1	V2 top-1
		Transformers				
ViT-B/16 [5]	86M	384^2	85.9	77.9	83.6	-
ViT-L/16 [5]	307M	384^2	27.3	76.5	82.2	-
DeiT-Ti	5M	224^2	2536.5	72.2	80.1	60.4
DeiT-S	22M	224^2	940.4	79.8	85.7	68.5
DeiT-B	86M	224^2	292.3	81.8	86.7	71.5
DeiT-B↑384	86M	384^2	85.9	83.1	87.7	72.4
DeiT-Ti⚗	6M	224^2	2529.5	74.5	82.1	62.9
DeiT-S⚗	22M	224^2	936.2	81.2	86.8	70.0
DeiT-B⚗	87M	224^2	290.9	83.4	88.3	73.2
DeiT-Ti⚗/ 1000 epochs	6M	224^2	2529.5	76.6	83.9	65.4
DeiT-S⚗/ 1000 epochs	22M	224^2	936.2	82.6	87.8	71.7
DeiT-B⚗/ 1000 epochs	87M	224^2	290.9	84.2	88.7	73.9
DeiT-B⚗↑384	87M	384^2	85.8	84.5	89.0	74.8
DeiT-B⚗↑384 / 1000 epochs	87M	384^2	85.8	85.2	89.3	75.2

Table 2: **Performance comparison of the DeiT strategy.** In this figure, results between distilled DeiT students and two original ViT implementations are shown. For example, the regular ViT-B/16 network with 86M parameters obtains an ImageNet 1K Top-1 score of 77.9. At the same time, a distilled DeiT-S student with 22M parameters obtains a Top-1 score of 79.8, showing the performance of the DeiT distillation strategy. Transformer subset of results table by Touvron et al.

3 Methodology

3.1 Research scope

In this paper, we attempt to close the gap between academic advances in the field of language-enabled AR and the real-time application of these models. More specifically, we want to transform the promising language-enabled AR models that we have seen in Section 2.3 to models that are suited for use in real-time scenarios rather than being applied in academic setting. We choose to focus entirely on the CLIP-based ActionCLIP and STAN models because they have the best speed-accuracy trade-off out of the four language-based AR models we investigated, and they also feature an openly available code implementation. As discussed in Section 2.3.6, we recognise two key issues that prevent these models from operating in real-time: they are computationally slow and they sample frames in a way that is not possible in real-time scenarios. The latter is relatively easy to solve, namely by changing the way in which frames are sampled during validation time to better represent the real-time predictive performance of the models. Next, to achieve real-time performance with (one of) the existing language-enabled AR models, we propose three distillation approaches that each aim to condense a specific portion of the knowledge contained in the existing CLIP model and its AR counterparts.

First, we explore the possibility of distilling the CLIP backbone that performs the spatial and textual encoding of the input in the existing AR models. In Section 2.4.3, we have seen that an effective way to do this is by using the TinyCLIP distillation strategy proposed by Wu et al. In their work, the authors propose a novel distillation method that effectively distills the knowledge of the CLIP model into a smaller student model that has similar predictive performance. However, this paper focuses on TinyCLIP's perfomance on image recognition tasks, not on its performance as an encoder backbone in an AR pipeline such as ActionCLIP or STAN. We will therefore evaluate the effect that using a pretrained TinyCLIP backbone has on these models. We do so by replacing the vanilla CLIP backbone with the TinyCLIP model family in existing CLIP-based AR pipelines to investigate how the performance-accuracy trade-off changes.

Next, we aim to distill the spatial knowledge of the original CLIP model in what we will call spatial distillation. Here, we define spatial information as information that can be determined without looking at frames over time. For example, a car's colour, make and size are spatial details that can be determined by looking at a single frame, but the direction it is moving and its speed are not. We perform spatial distillation by creating a multi-task learning objective for the student model in which it not only has to learn to predict the ground truth human actions in an AR dataset, but also a set of additional spatial objectives. Annotating these spatial objectives manually is very time consuming, and we therefore aim to leverage CLIP's broad spatial knowledge to generate these objectives. That is, we design a framework that can assign spatial labels to videos for a predefined set of additional objectives by using the CLIP model and its large pretraining. These spatial objectives can range from characteristics of the person performing the action like gender, pose or age to information about the surroundings in which the action is performed like the time of day, the location or the weather. By doing so, we aim to improve the supervisory signal and therefore the predictive performance of our student model on predicting human actions on datasets such as HMDB51 and UCF101.

Finally, we explore ways to distill the entire CLIP-based AR model in what we will call global distillation. What we mean by global distillation is that we do not distill a specific component of the existing AR models such as the backbone. Instead, we distill the entire model agnostically, by focussing exclusively on guiding the student to match the output of the teacher. By doing so, we entirely ignore the components within that model. Concretely, this means that we take a large CLIP-based AR model like ActionCLIP or STAN and experiment with distilling it into a smaller, more efficient student model by using the DeiT distillation strategy outlined in Section 2.4. We choose this strategy in particular because it creates the best-performing students in terms of speed and accuracy out of all strategies investigated. Given that this approach is strictly designed to distill into a transformer architecture, we feel like it is worth experimenting whether this strategy can be applied to effectively distill cross-modal video transformers.

Given these three proposed distillation approaches, we aim to optimally leverage the knowledge of large CLIP variants and its AR counterparts to create a language-enabled AR model that can be inferred in real-time at comparable performance to its teachers, thus making it suitable for application in real-time scenarios instead of merely an academic settings.

3.2 Research approach

We distinguish one central research question `(Main)` and four sub-questions `(RQ)` that each answer part of our central question. We combine our findings in the discussion section in which we answer and discuss these research questions in detail. An illustration of the topics covered per research question can be seen in Figure 16.

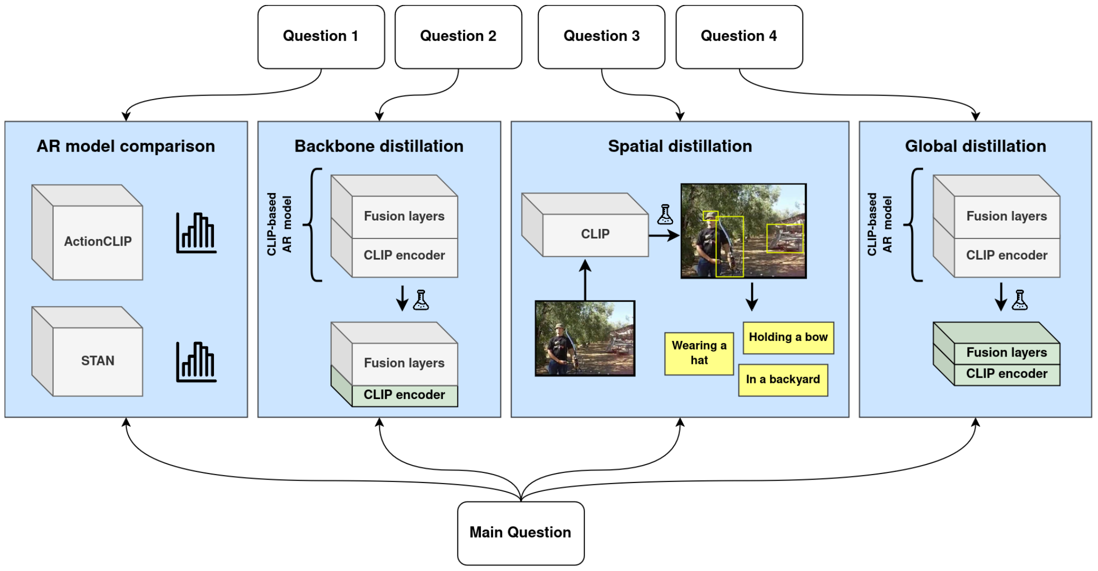


Figure 16: **Research approach overview.** This visual illustrates the three different distillation approaches that will be covered in this work as well as the model comparison between STAN and ActionCLIP that serves as a preliminary to further experiments.

`(Main)` Can a CLIP-based AR model be designed that can operate in real-time scenarios at comparable predictive performance to larger CLIP-based AR counterparts by applying global distillation, spatial distillation and backbone distillation or a combination thereof?

`(Q1)` Between the STAN and ActionCLIP models, which model has the best speed-accuracy ratio on HMDB51 and UCF101 when evaluated using a dense frame sampling strategy?

We have previously determined that ActionCLIP and STAN are most suitable for further experimentation. In Section 4, we compare the speed-accuracy ratios of these two models to determine the most suitable model to perform our threefold distillation experiments on. This serves as a preliminary investigation that helps us determine the best model to conduct further experimentation on between the two models. As previously mentioned, the reported performance of these models on AR datasets like HMDB51 and Kinetics is not entirely representative of their AR performance in real-time scenarios. We therefore compare performance of the two models using a dense frame sampling strategy.

`(Q2)` How does replacing the backbone of the CLIP-based AR model of our choice with a TinyCLIP variant influence its speed-accuracy trade-off on the HMDB51 dataset?

As we have seen, the TinyCLIP distillation strategy transforms models into smaller, faster students at minimal performance loss. The proposed distillation strategy produces the TinyClip family, a set of smaller CLIP models that are much faster than the original CLIP models at minimal cost in performance [33]. In Section 5, we assess the changes in inference time and performance when the backbone of the CLIP-based AR model of our choice is replaced by a member of the TinyCLIP family.

`(Q3)` **Can we distill CLIP's spatial knowledge by predicting additional spatial classification objectives for the CLIP-based AR model of our choice to train on by applying CLIP's zero-shot predictions on video frame data, and does training on these additional objectives improve predictive performance of the language-enabled AR model of our choice?**

In Section 6, we investigate whether a CLIP model can be used to predict spatial labels in video frames that in turn form additional objectives that a language-enabled AR model like ActionCLIP or STAN can be trained on. During training, the student model is then not only asked to predict the (ground-truth) action label, but also a set of spatial pseudo labels. We are particularly interested in whether these additional objectives improve the AR performance of the model by increasing the strength of the supervisory signal. We keep in mind that CLIP may not be able to determine (all) spatial characteristics with certainty, and thus have to determine the best way to deal with that uncertainty in generating spatial prompts.

`(Q3.1)` **What is the optimal way to present additional spatial objectives to the CLIP-based AR model: a single, combined natural language prompt or a multi-objective classification task?**

Most CLIP-based AR models are trained on a fixed set of labels that are placed into a prompt template. For example, from a ground truth label "smoking", a simple prompt in the form of "a person smoking" would be generated. The model then trains on associating that prompt with the embedding of a video in which a person smokes. The important question is in which form the additional spatial information should be presented to the AR model, and how loss should be weighted and calculated over these objectives. There are multiple ways to do this. One way to do this is to combine the additional spatial pseudo labels into a single prompt as a replacement for the more generic prompt of "a person label". Another way is to create multiple separate classification objectives and weight them according to importance in the loss function. This research question explores both possibilities.

`(Q3.2)` **In which way should the CLIP-based AR model's additional objectives be predicted, and should cross-entropy or contrastive loss be preferred to train the model on those objectives?**

Equally important is the way in which the AR model predicts these additional objectives. This can be done using an extra fully connected or cosine prediction head for each additional objective, and can be supervised contrastively or not. Here it is key to find a way to add prediction heads to the AR model of our choice that respects the previously mentioned cross-modal embedding space it operates in. We evaluate the effectiveness of multiple approaches by measuring whether predictive performance of the AR model is increased.

`(Q4)` **Can the DeiT image transformer distillation approach be adapted to suit the distillation between two transformer-based, language-enabled AR models?**

The DeiT distillation approach studied in Section 2.4 has demonstrated that large vision transformer models can be distilled in a faster student model with an increase in predictive performance. They achieve this by adding a distillation token that is similar in form to the class token normally used in a ViT. The teacher model is then used to predict hard labels for a set of input data. The student does its own prediction on that input data and two losses are combined using the DeiT strategy. In their paper it is shown that this significantly improves the distillation results compared to more traditional distillation approaches. We therefore investigate whether this distillation strategy is also applicable to transformer-based AR models. We deal with this question in Section 7.

3.3 General experimental set-up

Unless indicated otherwise, all models are evaluated by densely sampling 8 frames per video using a fixed frame interval of 4 because this better reflects the availability of video frames in real-world scenarios. This means that the total action window size while validating the model is 32 frames. These frames are sampled from the centre of the video. Sampling frames during training has no such contraints, frames are uniformly sampled over all frames in a video similar to the regular ActionCLIP sampling implementation. All experiments are run on a single NVIDIA RTX 4000 Ada generation with 20475MB of VRAM. For both the ActionCLIP and the STAN model we follow the implemented learning strategy from the respective papers. This means that both models are trained using a cosine learning schedule with a predefined amount of warmup steps. We use the *adamw* optimiser for both models. Both models use either a CLIP-ViT-B/32 or CLIP-ViT-B/16 pretrained backbone identical to the original CLIP implementation. The backbone weights for both the vision and text encoder of the CLIP model are learnable, albeit at a lower rate than the other AR model weights. Unless indicated otherwise, experiments are conducted on the Human Motion Database (HMDB51) dataset. We choose this dataset because it is small enough to conveniently experiment on while being complex enough to be a good measure of the quality of our methods. Given the breadth of our distillation experiments, we provide an extensive overview of all experimental results in our ablation study in Section A.

4 Selecting an existing CLIP-based AR pipeline

4.1 Motivation

An important first step in testing the effectiveness of our proposed threefold distillation CLIP-based AR models is to choose an appropriate CLIP-based AR model to experiment on. As we have seen in Section 2.3, multiple approaches exist. In this section, we briefly evaluate the two models that have an openly accessible implementation: the STAN model [18] and the ActionCLIP model [30].

4.2 The dense frame sampling strategy

The way in which frames are sampled can have a significant impact on the performance of a model. In our case, because our goal is to bridge the gap between academic advances in language-video transformers and real-time usage, we are only interested in sampling strategies suitable for use in real-time. Commonly used validation strategies that sample frames evenly distributed over a clip with n frames (e.g. 1, 50, 100, 150) increase the window in which actions are detected, but require the model to wait for those frames to come in before performing inference on them, thus introducing an artificial delay in its predictions. What is more, the starting and ending point of a video in a real-time scenario is not known, as opposed to video samples in the HMDB51 or Kinetics-400 dataset that are readily centred around a human action. Uniformly sampling frames from the entire action window is therefore not possible. As previously mentioned, this makes uniformly distributed frame-sampling a poor representation of a model's predictive performance in real-time scenarios. Instead, we choose to integrate a simple dense frame sampling strategy for model validation. This means that a frame interval is specified and that frames are sampled in this interval around the middle frame of that video. For example, for a frame interval of 4 and a frame sample size of 8, we cover an action window of (8×4) 32 frames in total. For a camera running at 32 FPS, this means the instant delay added by waiting for these frames to come in is approximately one second. It follows that increasing this window adds more delay to the model, because it will have to wait for these frames to come in from the camera stream. This is the reason that we choose to evaluate each model by sampling only 8 frames per video. Increasing this amount to 16 while at the same time respecting a maximum instant delay of one second to satisfy the real-time constraint means that 16 frames could be sampled with a frame interval of 2, leading to a similarly sized frame window of (16×2) 32. This does not make much sense because a 4 frame interval is already quite small and it is unlikely that predictive performance is increased by making this interval even smaller at the cost of a significant increase in inference time introduced by doubling the amount of frame tokens that need to be embedded and processed in the model. Sampling frames during training is not bound by the aforementioned constraints. There exist multiple options to do so, such as sampling random frames, sampling frames uniformly, sampling frames densely around the middle of the video or sampling frames from a random starting frame. We find that the default uniform frame sampling strategy used in both the original ActionCLIP and STAN implementation works best. This means that it is trained on n uniformly sampled frames between a random starting point and end point. This sampling strategy is applied during training because it increases the variety of the frames that are sampled.

4.3 Experimental set-up

In order to determine the most suitable AR architecture for our proposed distillation strategies, we evaluate both models on the HMDB51 and UCF101 datasets with their default backbone(s). Furthermore, we replace existing frame sampling strategies with a dense sampling strategy to suit our intention to apply the model in real-time scenarios. Finally, for all runs the averaged inference time over a validation set batch of size 32 is recorded to assert the efficiency of each model. We record this time at CUDA-level, meaning that the asynchronous nature of GPU processing is taken into account in measuring the inference time of the model.

4.4 Experimental results

The predictive validation scores and the inference time of both architectures are presented in Table 3. Clearly, STAN benefits more from a smaller CLIP backbone, with the CLIP-ViT-B/16 being more than four times slower than the CLIP-ViT-B/32 configuration. That said, we can see that the ActionCLIP model is faster for both the CLIP-ViT-B/32 and the CLIP-ViT-B/16 backbone. The STAN model outperforms ActionCLIP on Top-1 score using the CLIP-ViT-B/16 backbone, but only by 0.42 and with an batch inference time increase of approximately 73%. Between the CLIP-ViT-B/32 backbones, ActionCLIP outperforms STAN in terms of both Top-1 score and inference time. We therefore decide to perform further distillation experiments on the ActionCLIP model.

Arch	Backbone	UCF T1	UCF T5	HMDB51 T1	HMDB51 T5	Inference (ms)
STAN	CLIP-ViT-B/16	91.60	98.90	72.09	93.53	861.66
	CLIP-ViT-B/32	90.10	98.10	67.45	92.88	200.13
ActionCLIP	CLIP-ViT-B/16	93.8	99.33	71.67	94.62	495.75
	CLIP-ViT-B/32	**90.40**	**98.40**	**68.80**	**93.54**	**140.24**

Table 3: **Performance comparison of the STAN and ActionCLIP architectures on HMDB51 and UCF101.** Experiment run according to the experimental setup described in Section 3.3. Inference time is reported in milliseconds over the average time it takes to process a batch of 32 video samples in evaluation mode. That is, without tracking loss gradients that are present during training. No further optimisations were performed.

5 CLIP backbone distillation and finetuning

5.1 Motivation

As we have seen, the TinyCLIP approach proposed by Wu et al. is an effective way to distill the CLIP model. The TinyCLIP family offers multiple small, efficient CLIP models that aim to improve the trade-off between complexity and performance. However, the paper does not research the effect that the TinyCLIP family has on CLIP-based AR models such as ActionCLIP. As we have seen, this model uses the CLIP model as its encoder backbone. All frames are encoded by CLIP's visual encoder before being passed into the frame-level global attention layer of the model. Similarly, the text prompts are encoded by CLIP's text encoder in the same way. It therefore plays an integral role in the model. A logical step is therefore to investigate how the speed-accuracy trade-off changes when we replace this backbone. We also look at whether performance of this backbone can further be improved by finetuning the backbone. That is, to include a loss term that is calculated over the backbone's ability to match each frame's embedding with the correct text (label) embedding and vice versa as if it were an AR model rather than an encoder backbone.

5.2 Experimental set-up

In order to assess the impact of the TinyCLIP family on ActionCLIP, we extend the available CLIP backbones (CLIP-ViT-B/32, CLIP-ViT-B/16) with a distilled TinyCLIP-ViT-40M/32-19M model that has a patch size of 32 and 59 million parameters, less than half of the 16 million parameters in the CLIP-ViT-B/32 model. We choose this TinyCLIP variant because it effectively more than halves the number of parameters in the network. In our evaluation, we measure both the Top-1 and Top-5 scores on HMDB51 and UCF101 to assess ActionCLIP's predictive performance for each backbone. We also measure the average inference time in milliseconds per batch of 32 videos in the validation set using the NVIDIA CUDA timing toolkit. For measuring inference time we ensure that they are set to evaluation mode so that gradients are not tracked, thus better reflecting model speed.
Integration of the TinyCLIP-ViT-40M/32-19M model into the ActionCLIP framework is relatively straightforward because the TinyCLIP backbone is a ViT, similar to that of the original CLIP model. With some minor adjustments to the ActionCLIP architecture, we can choose to use either a pretrained CLIP or TinyCLIP model as the backbone of our AR model.

5.3 Experimental results

Replacing the CLIP backbone

All three backbones were evaluated on both the HMDB51 and the UCF101 dataset. The results are illustrated in Table 4. Clearly, a larger backbone positively impacts the Top-1 and Top-5 scores on both HMDB51 and UCF101. That said, the CLIP-ViT-B/32 takes approximately 73% more time to complete inference on a batch than the much faster TinyCLIP-ViT-40M/32-19M, albeit at a significant Top-1 score increase of 9.55%. What is also interesting is the significant jump in inference time between the CLIP-ViT-B/32 and CLIP-ViT-B/16 models. In moving from the model with patch size 32 to the model with patch size 16, a Top-1 score improvement of 4.17% is obtained at the cost of an 253.5% increase in inference time.

Backbone	**UCF T1**	**UCF T5**	**HMDB51 T1**	**HMDB51 T5**	**Inference (ms)**
CLIP-ViT-B/16	93.80	99.33	71.67	94.62	495.75
CLIP-ViT-B/32	90.40	98.40	68.80	93.54	140.24
TinyCLIP40M/32-19M	89.80	98.86	62.80	90.10	**80.87**

Table 4: **Comparison of the ActionCLIP backbone architectures on HMDB51 and UCF101**. Compared on the UCF101 and HMDB51 datasets. Batch size is set to 32 for all three models using a dense frame sampling strategy. The inference column reports the average time it takes the model to process a batch of 32 videos in milliseconds over the entire validation set.

Actively finetuning the backbone

First and foremost, ActionCLIP does not freeze the image or text encoder weights of its CLIP backbone because the authors show that allowing CLIP to improve its weights to better serve the AR model improves its predictive performance [30]. This in itself can be considered backbone finetuning, and this form of finetuning is present in all experiments because as per ActionCLIP's original implementation, the CLIP encoder weights are never frozen.

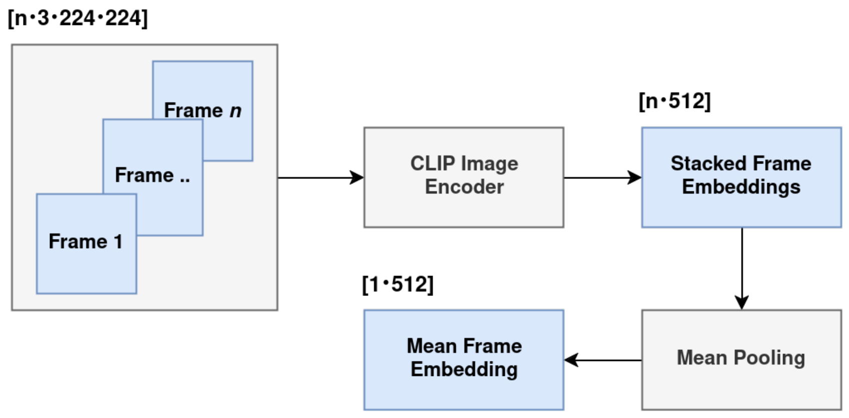


Figure 17: **Finetuning architecture**. The video representation of the backbone is obtained by mean pooling the frame-level embeddings. In each step, the dimensionality of the original input is reported in the top left corner. In this figure, n represents the numbers of frames (densely) sampled from each video. The "Mean Frame Embedding" of each video in the batch is then contrastively matched to all text embeddings in the batch and KLL loss is calculated over the result. This way, the backbone is directly supervised in predicting the ground truth action label.

This means that part of the weights of ActionCLIP that are updated during training are those of the encoder backbone. This further underlines the previously stated fact that the cross-modal space that language-enabled models operate in has to be respected. Failure to do so could potentially disrupt CLIP's pretrained weights and therefore its ability to produce semantically rich embeddings for the ActionCLIP model. That said, it is also possible to actively finetune the CLIP encoder backbone on specific tasks while training the CLIP-based AR model in order to improve performance of the AR model. Finetuning CLIP is a strategy outlined in the original CLIP paper by Radford et al. and is more or less similar to the way that CLIP itself is pretrained, albeit at a lower learning rate [21]. That is, CLIP can be provided with image-text pairs from a dataset and it can then be tasked to contrastively match these to improve its performance on that specific dataset. This means that when ActionCLIP trains on a dataset such as HMDB51, we can also guide its encoder to individually match image-text pairs from that dataset to actively finetune it. Results indicate that this increases the performance of the CLIP model compared to a zero-shot configuration. However, finetuning a CLIP model that serves as an encoder backbone raises some potential issues. For example, CLIP as an encoder component of an AR model like ActionCLIP will have to be finetuned on video data rather than on the image data that CLIP is finetuned on in its original paper. More specifically, it does not receive a single frame from which it can be tasked to predict the appropriate action label, but receives a set of n sampled frames instead. There are multiple ways to deal with this problem. One way is to let CLIP predict an action category based on a majority vote over the n sampled video frames. For example, if the majority of the n frames sampled from a video matches most closely with the embedded label of "a person waving", that label is taken as the backbone's action prediction and loss can be calculated accordingly. Likewise, each text embedding votes for its most likely video embedding counterpart in exactly the same way. Another way is to transform the n separate frame embeddings into a single averaged frame embedding by applying mean pooling over the embeddings of the n sampled frames. Alternatively, it is also possible to sample a random subset of the n frames that each get a vote in predicting an action label. We choose to apply mean pooling over the frame embeddings of the n sampled frames. The way the mean pooled representation is obtained is depicted in Figure 17. We do this because a dense frame sampling strategy is used and the sampled frames can therefore not be so far apart such that they depict very different actions. This means that a new part of the total loss term in training the ActionCLIP model is the ability of its backbone to match mean pooled frame embeddings to text embeddings and vice versa, in which the mean pooled frame embeddings are taken over the n frame embeddings sampled from a video. This is different than the loss calculated over video-text pairs in the final layer of the ActionCLIP model because the frame embeddings have not gone through ActionCLIP's previously discussed fusion module in which global attention is applied over the n frame embeddings in order to create a video embedding. The results of this finetuning approach are depicted in Table 18 and its corresponding graph. Again, it is important to note that by "no finetune" in the graph legend we mean that no *active* backbone finetuning it performed, but CLIP's weights are not frozen. The "no finetune" variant (blue) follows ActionCLIP's original implementation whereas the finetuned variant (green) finetunes CLIP during training of the AR model. We conclude that actively finetuning the CLIP backbone in a contrastive way using our proposed mean pooled frame representation approach improves the predictive performance of the ActionCLIP model.

HMDB51 Experimental results

Backbone	Backbone finetuning	Dataset	Loss	Top-1 score	Top-5 score
TinyCLIP-ViT-40	Yes	HMDB51	vanilla	**63.90**	**91.10**
TinyCLIP-ViT-40	No	HMDB51	vanilla	62.80	90.10

Table 5: **Experimental results of backbone finetuning on HMDB51.** The table shows that backbone finetuning slightly improves the Top-1 and Top-5 scores of the ActionCLIP model as compared to the vanilla model.

HMDB51 Top-1 validation graph

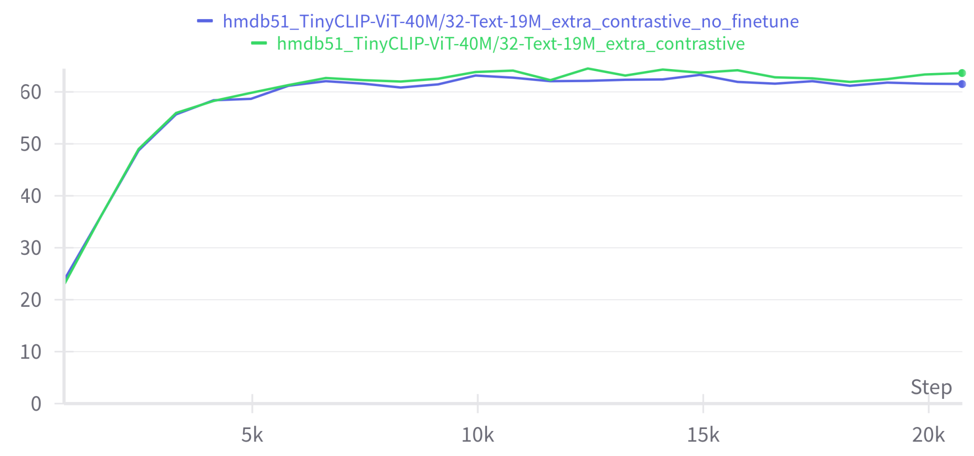


Figure 18: **Graph depicting backbone finetune results on HMDB51 Top-1 score.** ActionCLIP performance on HMDB51 with backbone finetuning (green) with a peak Top-1 score of 63.90 and without backbone finetuning (blue) with a peak Top-1 score of 62.80. The figure illustrates the effectiveness of actively finetuning the backbone using our proposed mean-pooling approach and a contrastive loss function.

6 Spatial CLIP distillation

6.1 Motivation

A key strength of the CLIP model is its flexibility in reasoning about spatial details in images. Examples include details of what someone is wearing, what someone is holding, the pose of their body, the weather situation or whether a picture was taken indoors or outdoors. In this section, we describe the ways in which we use this knowledge to supply an AR model with spatial context for video samples. More specifically, we aim to generate a set of pseudo-objectives for each video sample in a dataset using a large CLIP-ViT-B/14 model that operates on frames sampled from a video. The key characteristic of a pseudo-objective is that while it represents a learning target for the student model, there is no guarantee that it is correct when matched with a ground-truth label. This is because the label is not manually annotated but predicted by another model instead. We extract this spatial information in order to aid in the supervisory signal of the AR model being trained. By doing so, we hope to indirectly improve the AR model's recognition performance because we have more control over what the model pays attention to. We hypothesise that penalising or rewarding an AR model based on its prediction of spatial details that are potentially relevant to the action being performed improves its performance to recognise those actions.

6.2 CLIP limitations and dealing with uncertainty

Although impressive, CLIP's predictive performance is not without flaws. Aside from its obvious inability to interpret motion, we present a number of cases where CLIP is not suited to perform classification even though it is faced with a relatively straightforward task. These limitations are relevant because they make using CLIP for the generating of spatial pseudo-objectives challenging. In this section, we discuss some relevant limitations in the CLIP model and then explore the best way to deal with them.

Logical reasoning

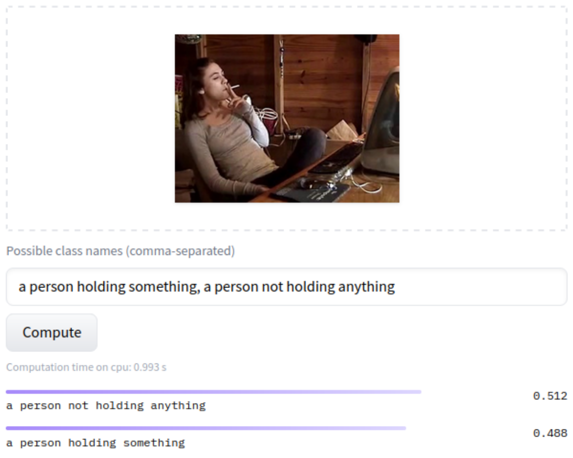
One important limitation is that CLIP struggles with prompts containing logical statements, negations in particular. This is illustrated in figure 19a. When CLIP is presented with a prompt on whether someone is holding something in their hands or not, it is not certain. Likely because both text embeddings are strongly aligned with the embedding of the video frame, the only difference being a "not". Similarly, we found that if the model is prompted with "a person holding a cigarette" and "a person holding nothing", it will predict the former by a large margin. We suspect that CLIP makes this decision not because it reasons about the negation of "nothing", but because "cigarette" is much more strongly aligned with the frame embedding of a person smoking. This example shows that CLIP does not argue in a way that a Large Language Model such as ChatGPT does. It naively aligns text-image pairs in a joint embedding space, but does comprehend logical statements from text prompts. We therefore note that constructing prompts that are sufficiently specific is important.

Overfitted knowledge

At the same time, the visual details that CLIP pays attention to are not always clear. More specifically, it sometimes seems as if CLIP has overfitted its knowledge on certain topics, like looking at a person's pose to determine what someone is holding. This can also cause unexpected behaviour, as illustrated in Figure 19b. CLIP hallucinates a cigarette, likely by looking at human pose rather than the object that a person is holding in their hands. Using CLIP to generate spatial pseudo-objectives, it quickly becomes clear that CLIP does well on certain objectives like the *location* objective and the *gender* objective, but it struggles greatly with the more complex *pose* and *holding* objectives. This makes sense, because a setting such as a park or backyard is very different from being in a bar or restaurant, whereas the difference between holding a baseball bat and a golf club is much more subtle.

Overconfidence

Another potential pitfall lies in the many ways in which CLIP can be prompted. We previously mentioned that CLIP should always be offered a wildcard way out of incorrect prompts. Figure 20 illustrates why: CLIP can be surprisingly confident when presented with exclusively wrong options to pick from, and techniques like certainty thresholding on the softmax logits will therefore not be helpful to filter out wrongful predictions. The wildcard option in the form of "a woman" or "a person" will typically be preferred over the wrong labels. This compensates for CLIP's overconfidence given that each frame contains at least one person at all times.

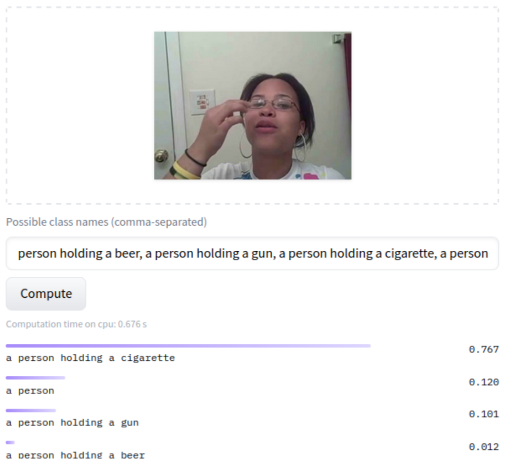

Possible class names (comma-separated)

a person holding something, a person not holding anything

Computation time on cpu: 0.993 s

```
a person not holding anything        0.512
a person holding something           0.488
```

(a)


Possible class names (comma-separated)

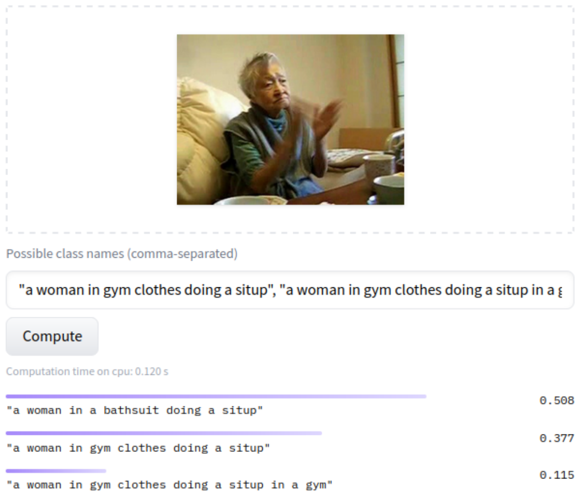
person holding a beer, a person holding a gun, a person holding a cigarette, a person

Computation time on cpu: 0.676 s

```
a person holding a cigarette        0.767
a person                            0.120
a person holding a gun              0.101
a person holding a beer             0.012
```

(b)

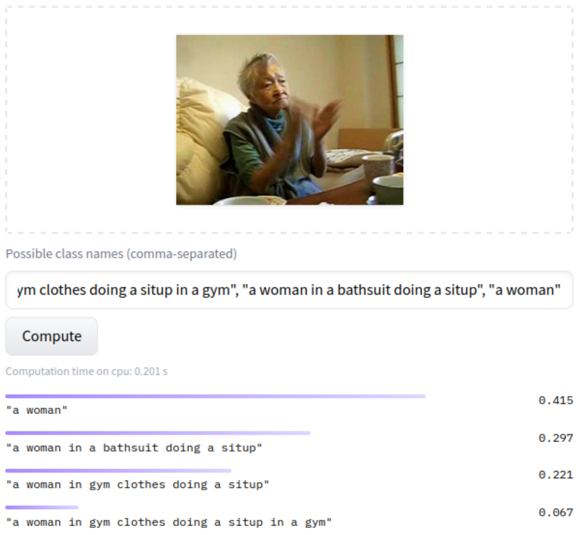
Figure 19: **Illustration of two CLIP limitations.** In these two figures, two key limitations are illustrated. In (a), we show that CLIP ignores logical reasoning, assigning equal probabilities to two similar but contradicting prompts. In (b), we see that CLIP has predicts that someone is holding a cigarett even though no cigarette is present in the picture. The most likely explanation is that CLIP has overfitted on the pose of people holding cigarettes rather than looking at the cigarette itself.


Possible class names (comma-separated)

"a woman in gym clothes doing a situp", "a woman in gym clothes doing a situp in a g

Computation time on cpu: 0.120 s

```
"a woman in a bathsuit doing a situp"                0.508
"a woman in gym clothes doing a situp"               0.377
"a woman in gym clothes doing a situp in a gym"      0.115
```


Possible class names (comma-separated)

ym clothes doing a situp in a gym", "a woman in a bathsuit doing a situp", "a woman"

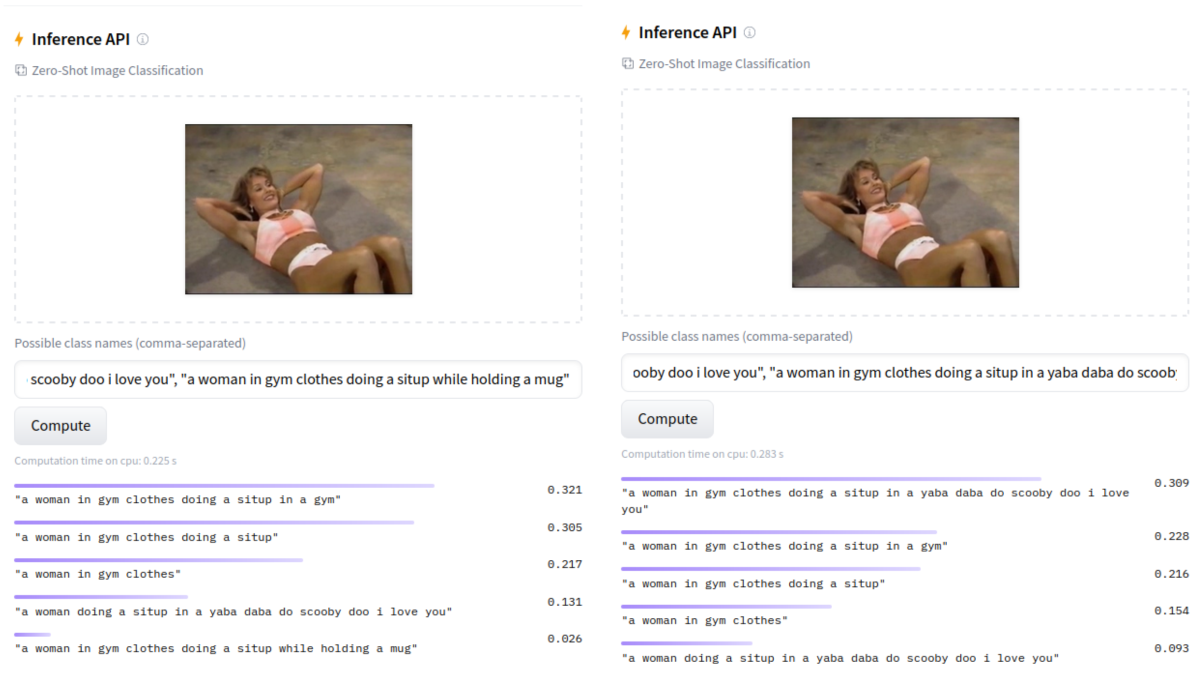
Computation time on cpu: 0.201 s

```
"a woman"                                            0.415
"a woman in a bathsuit doing a situp"                0.297
"a woman in gym clothes doing a situp"               0.221
"a woman in gym clothes doing a situp in a gym"      0.067
```

Figure 20: **Prediction wildcard example**. This figure shows CLIP ViT-B/14 logits for two different sets of text prompts. The importance of the wildcard prompt is illustrated in a situation where there are no correct prompts to choose from. On the left, CLIP shows surprising confidence in selecting a plain wrong prompt. On the right, the wildcard prevents this mistake by providing a more likely option as a way out for CLIP.

Information loss on extended prompts

A more complex CLIP limitation that is less obvious is the way in which CLIP deals with text that it has never seen before. This is illustrated in Figure 21. We present the model with two similar images. On the left we see a valid prediction, in which CLIP option that contains the longest correct prompt possible. On the right, that changes when we introduce a label that mimics the longest correct label, but that adds a string of random gibberish. Suddenly, CLIP prefers the longer prompt, despite the fact that it contains a combination of words that it has probably never come across before. We speculate that this occurs because CLIP is trained contrastively. This means that it is rewarded or penalised based on whether it aligns image-text pairs correctly, but does not know how to deal with image-text pairs that it has never come across during training. That is, if the added string of gibberish “in a yaba daba do scooby do I love you” would add incorrect information such as “in a bath suit”, CLIP would have given it a much lower probability. However, because it simply adds information that CLIP cannot make sense of, it seems to give the prompt the benefit of the doubt. This problem seems insignificant at first, but we find that this unreliable behaviour limits the length of the prompts that can be presented to CLIP.


Inference API
Zero-Shot Image Classification
Possible class names (comma-separated)
Compute

scooby doo i love you", "a woman in gym clothes doing a situp while holding a mug"

Computation time on cpu: 0.225 s

```
"a woman in gym clothes doing a situp in a gym"                              0.321
"a woman in gym clothes doing a situp"                                       0.305
"a woman in gym clothes"                                                     0.217
"a woman doing a situp in a yaba daba do scooby doo i love you"             0.131
"a woman in gym clothes doing a situp while holding a mug"                  0.026
```

ooby doo i love you", "a woman in gym clothes doing a situp in a yaba daba do scoob

Computation time on cpu: 0.283 s

```
"a woman in gym clothes doing a situp in a yaba daba do scooby doo i love    0.309
you"
"a woman in gym clothes doing a situp in a gym"                              0.228
"a woman in gym clothes doing a situp"                                       0.216
"a woman in gym clothes"                                                     0.154
"a woman doing a situp in a yaba daba do scooby doo i love you"             0.093
```

Figure 21: **CLIP ViT-B/14 logits for two different sets of prompts.** In this figure, the unreliable behaviour of the CLIP model can be seen when presented with long prompts that contain information unknown to CLIP. On the left, CLIP correctly picks the most fitting prompt. On the right, we see that CLIP prefers a prompt that starts similarly but appends a set of nonsensical words that CLIP has likely never seen before.

Dealing with CLIP’s limitations

Given this information, we distinguish three core predictive limitations of the CLIP model and a way to address each of them: the inability to interpret logical negations contained in prompts, the need for an appropriate wildcard and its tendency to produce volatile predictions when presented with multiple long prompts. The wildcard limitation stems from the fact that CLIP is surprisingly confident when picking between classes that are all incorrect. The impact of this tendency can be minimised by presenting an appropriate wildcard. In the HMDB51 AR dataset that we are working with, we can safely assume that a person can typically be seen in any given frame. It follows that we can append the prompt of “a person” to the labels that CLIP can pick from for each objective to serve as a wildcard. However, it is easy to see how this would cause issues for more varied datasets in which an intuitive wildcard is not available. Datasets that are not as readily processed as HMDB51 or Kinetics often have large portions of neutral footage in which no particular action occurs and in which no person is present. In these situations, a more advanced set of wildcard prompts would have to be designed.

As for the inability to interpret logic, we found that a good workaround is to be very specific. Ideally, the list of possible classes in a category is complete. This means that all possible classes for a given objective should be provided to CLIP. For example, if the CLIP model is presented with the set *holding* prompts "a baseball bat", "something to eat" and "a ball" and the object that the person is actually holding is a golf club, it is likely that CLIP will confidently pick the option "a baseball bat". It will do this for every occurrence where the object that the person is holding is remotely similar to one of the available (albeit incorrect) labels. As we have seen, if no remotely similar prompt presented to the model is available, CLIP will sometimes even confidently pick a seemingly random prompt. However, by providing a list of possible labels for each objective that is as complete as possible and by using the previously described wildcard prompt to prevent random answers is an effective way to work around these limitations. Another workaround we developed is a two-step inference process that uses simple logical rules to transform a multi-class classification problem into a binary pseudo-label. In the first step, CLIP is asked to choose between a set of prompts of all things a person could possibly hold in that dataset and the prompt wildcard. Because CLIP struggles with fine-grained details, it will likely make many mistakes in this task. However, we can then transform its prediction to a binary class of "a person holding not holding anything" and "a person holding something" by assigning the former label if the wildcard was preferred over all the possible things a person could hold and by assigning the latter otherwise. However, by doing so, granular information on what a person is holding is lost.

6.3 Choosing spatial objectives

A key aspect of multi-objective learning is to choose suitable additional objectives, as described in the paper Multi-task Learning by Caruana. Because our goal is to increase the AR model's predictive performance, we want to choose objectives that hold some relation to the actions that need to be predicted. However, choosing the correct objectives can be tricky. In our case we want to obtain a set of objectives that are both related to the action categories *and* that can be predicted reliably by the CLIP model. For example, if CLIP struggles to consistently assign a person's supposed gender correctly, it is unlikely to provide a consistent supervisory signal to the ActionCLIP model during training. Ideally, we want to use a set of ground truth labels for each of these objectives in a particular dataset so that CLIP's predictive performance on that objective can be measured. However, labelling (part of) a dataset with additional objectives is very time consuming, especially as spatial objectives become more complex. It also entirely wastes the potential of using CLIP's zero-shot capabilities to generate pseudo-objectives for any dataset you want to train a model on. As such, we evaluate CLIP's predictions manually. Spotting mistakes is relatively straightforward. Given all objective labels, one can concatenate them into a single string. For example, given the ground truth label "shooting a bow" and the pseudo-labels "a boy", "wearing a suit" and "in a bar or restaurant", it is safe to say that one of the pseudo-labels is wrong. However, it says nothing about which particular pseudo-label is wrong, and manual inspection of the video frames is required to determine the faulty label. A specific label that CLIP cannot predict correctly in a consistent manner will likely stand out in this list. A second, more objective, way to assess whether a spatial pseudo-label can be predicted consistently by the CLIP model is by measuring the training and validation set accuracy of the AR model for each additional objective. This gives us some indication as to whether there is a consistent pattern shown in the pseudo-labels generated by the CLIP model that ActionCLIP can fit to. Here it is important to note that the pseudo-objective accuracy of each objective measures its ability to mimic the CLIP model and therefore reflects the CLIP's consistency in predicting the pseudo-objective. More specifically, if the AR model can fit to the pseudo-objectives, then at the very least there is a consistent signal that it can fit to, whether or not those pseudo-objectives are always correct. The latter cannot be verified other than by the previously mentioned manual inspection, because no ground truth labels for the additional objectives exist. Keeping that in mind, we select five preliminary pseudo-objectives: *gender*, *location*, *clothing*, *holding* and *pose* to experiment with. We choose these preliminary objectives based on the video content of the HMDB51. For example, the *holding* objective matches all the objects that can be interacted with according to the action labels of the dataset. At the same time, we try to minimise overlap between labels for an objective as much as possible so that they easier to distinguish.

Genders: a man, a woman, a boy, a girl
Locations: in a park or backyard, in a cafe or restaurant, in an office building, in a gym, on the streets, *a person*
Pose: sitting, standing upright, laying down, *a person*
Clothes: wearing a gym outfit, wearing a suit, wearing a dress, wearing winter clothes, wearing a t-shirt, *a person*
Holding: holding a brush, holding something to eat, holding a drink, holding a sword, holding a golf club, holding a ball, holding a bow, holding a gun, holding a baseball bat, holding a bag, holding a bar, *a person*

The pseudo-label prediction framework

Given the set of initial objectives, we construct a prediction framework that samples a set of n frames evenly spaced in the middle of the video from our training set. The prediction pipeline for a single frame is presented in Figure 22. The available set of labels for an objective plus the wildcard are preprocessed and embedded by the pretrained CLIP model. Similarly, the video frame is also preprocessed using the default visual CLIP preprocessing pipeline and then encoded to a frame embedding. Then, cosine similarity is measured to determine the most likely objective label for that frame. This process is repeated for each of the frames sampled from the video. More specifically, each frame gets a vote on the most likely class for a given objective. For example, if CLIP predicts that the person performing a certain action is "a girl" in 10 out of 16 frames, the label "a girl" is assigned to that video sample. The wildcard strategy described earlier is applied to the framework by adding the "a person" prompt to the set of possible labels for each objective. For example, if CLIP is asked to choose to pick the most likely location label for a given frame, it will choose between labels such as "at home", "in a park" *and* the label "a person". If the label "a person" is preferred over the other options, that label is masked in loss function of the training loop later on because it means that CLIP could not predict that objective reliably. Additionally, we average the objective logits over the n sampled frames and mask any label that falls below a set threshold. We determined that a threshold of 0.4 is the optimal value to filter out the faulty predictions while maintaining the correct predictions. Having some faulty pseudo-labels is to be expected, but this two-fold uncertainty handling mechanism helps to filter out the majority of the incorrect predictions.

6.4 Experimental set-up and criteria

In order to determine whether spatial distillation improves the predictive performance of a CLIP-based AR model, we conduct a number of experiments. Pseudo-labels are generated for each training sample in HMDB51 and these are combined with the ground truth labels of that dataset. We choose to evaluate the effectiveness of the additional objectives on the ActionCLIP model with a configuration as described in Section 3.3. As for the prediction objectives, we choose to combine the ground truth HMDB51 action labels with the three most reliable additional objectives predicted by our proposed prediction framework: *location, clothes and gender*. The other two objectives, *holding* and *pose*, are discarded. This is described in detail in appendix Section B, in which we provide an in-detail analysis of the generated spatial data. Because the effectiveness of contrastive learning can be influenced by batch size, we set a fixed batch size of 94 for all experiments. We choose this number because it is the largest possible batch size with which all training experiments can be run on our hardware. Finally, to assert whether spatial distillation improves the predictive performance of the ActionCLIP model, we compare Top-1 and Top-5 scores on the validation set of HMDB51. It should be noted that during evaluation of the model, the prediction heads for the additional objectives are disregarded because their sole purpose is to serve as an extra supervisory signal during training.

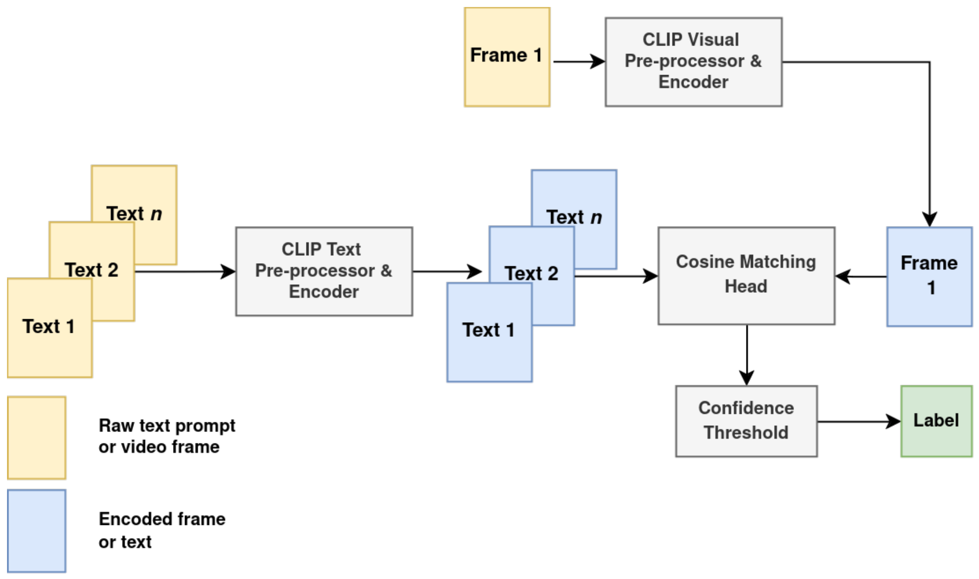


Figure 22: **Spatial pseudo-objective generation framework.** Illustrates the way in which a pseudo-label is assigned to a frame for the *location* objective through cosine matching. Raw text prompts for all n possible labels of an objective are passed into the CLIP pre-processor and text encoder. At the same time, a raw frame is passed into the visual pre-processor and encoder. Then, given a textual embedding for each prompt and a visual embedding of a frame, the most likely text is matched to the frame embedding. In case the wildcard text prompt was preferred *or* the confidence threshold is not met, the label is discarded. Else, the label is saved as part of the video that the frame was extracted from as a pseudo-label. In case a label is discarded, it is marked as such for that specific pseudo-objective and its subsequently ignored in the loss calculation during training.

6.5 Presenting the objectives to the AR pipeline

After generating the pseudo-labels for each video with the provided framework, we are left with a ground truth label and a label for each pseudo-objective for every sample in the training set. In our case, this means that every training sample has a label for the performed action, but also for the *gender*, *location* and *clothes* objectives that describe the person performing the action. The next step is to present this information to our model. This is not a straightforward process for a cross-modal architecture. Typical CNNs have one or multiple prediction heads in the form of a fully connected layer that transforms the final flattened layer of the CNN to the desired amount of output classes. In the cross-modal ActionCLIP architecture, no such fully connected layer is present. Instead, as per its original implementation, the model matches video-to-text and text-to-video using a cosine similarity matching head. Then, during validation, prediction logits are obtained by matching any video embedding to its most likely text embedding (label). Given a ground truth label for a video such as “shooting a bow” and its corresponding pseudo-labels “a man”, “in a backyard or park” and “wearing a gym outfit”, we distinguish two ways to present the set of labels. The first way is by fusing the pseudo-labels into a single prompt such as “a man wearing a gym outfit shooting a bow in a backyard or park”. This way, each video label is transformed into a small, descriptive sentence. Loss can then be calculated over its ability to match the video embedding with the right sentence prompt embedding for each video. We call this the **descriptive multi-objective approach**. This means that despite the fact that multiple objectives are included in the label, the model is still only asked to perform a single matching task. The second way is to perform cosine similarity matching between objectives separately. This means that the video embedding uses cosine similarity matching with the right text embeddings for each objective. In line with the original ActionCLIP implementation, we add cosine matching prediction head for each additional objective so as to respect the model’s cross-modality. We call this the **separated multi-objective approach**. We experiment with both objective representations.

Descriptive multi-objective approach

We start by using the approach that seems most similar to the contrastive image-text alignment that is used in training CLIP, namely to create a single, descriptive prompt out of the ground truth label and the additional pseudo-labels for a sample and perform contrastive alignment on the resulting video-text pairs. This is illustrated in Figure 23. The descriptive label adds spatial detail to this prompt in order to increase the supervisory signal. For example, the vanilla (single objective) approach would learn to associate a certain video embedding with “a person shooting a gun” whereas the multi-objective strategy would associate that same video embedding with “a woman shooting a gun in a backyard or park” instead.

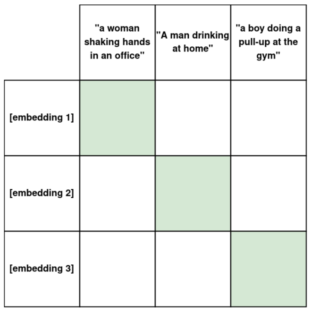

"a woman shaking hands in an office"
"A man drinking at home"
"a boy doing a pull-up at the gym"
[embedding 1]
[embedding 2]
[embedding 3]

Figure 23: **Illustration of the descriptive multi-objective approach.** Video embeddings are matched to the correct text embeddings and vice versa. This approach is similar to the vanilla ActionCLIP approach, other than that the ground truth action label is combined with information of the additional objectives in a single, descriptive sentence. Intuitively, the correct labels are represented by an identity matrix. That is, the correct video-text pairs are aligned diagonally.

Descriptive multi-objective experimental results

Figure 24 shows a comparison between the descriptive multi-objective approach (green) and the default single-objective prompt with no additional spatial information (blue). For example, one such descriptive label could be “a woman riding a bike on the street” whereas a default prompt would simply be “riding a bike”. In this comparison, the ActionCLIP variant using the simple labels obtains a Top-1 accuracy of 62.8 whereas the descriptive multi-objective approach obtains a Top-1 accuracy of less than 60. We therefore conclude that merging the objectives into a single label does not improve the predictive performance of the model.

HMDB51 Experimental results

Backbone	**Backbone finetune**	**Objectives**	**Label type**	**Loss mode**	**Top-1**	**Top-5**
TinyCLIP-ViT-40	No	Multi	Spatially descriptive	Contrastive	59.40	89.40
TinyCLIP-ViT-40	No	Single	Vanilla	Contrastive	**62.80**	90.10

Table 6: **Experimental results comparing spatially descriptive labels versus the vanilla model.** The vanilla single objective strategy uses a strict template of “a person [label]” where “[label]” is replaced by the ground truth label of that video sample. We find that the descriptive sentence representation of the additional spatial objectives does not improve the Top-1 or Top-5 performance on the HMDB51 validation set compared to the vanilla approach.

HMDB51 Top-1 validation graph

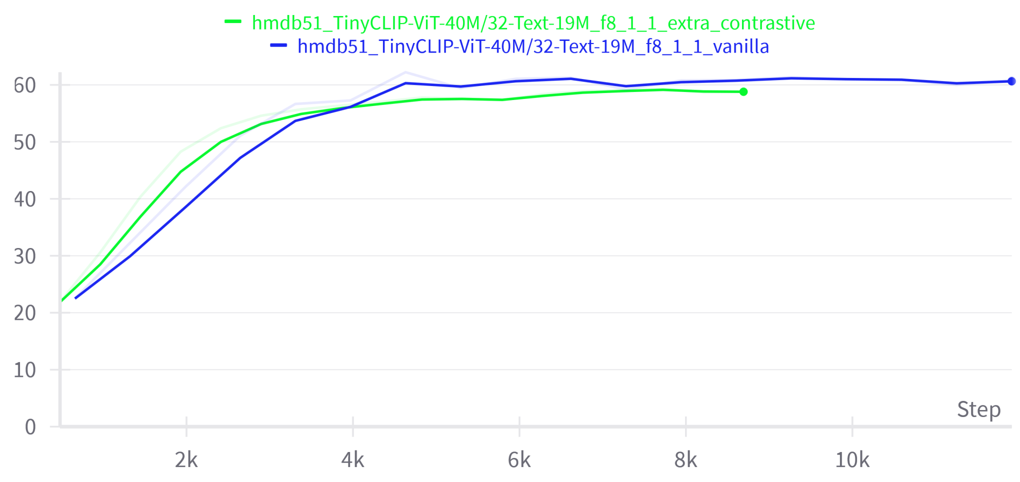


Figure 24: **Descriptive multi-objective strategy Top-1 graph on HMDB51**. Top-1 performance on the HMDB51 dataset for the “vanilla” model (blue) with a peak Top-1 score of 62.80 and the “extra contrastive” model (green) using descriptive sentence prompts composed out of additional (spatial) pseudo-objective labels with a peak Top-1 score of 59.42.

Separated multi-objective approach

In the case of presenting the additional labels separately, we have two options: inter-batch matching or intra-batch matching. By inter-batch matching we mean that we match video embeddings to all possible text embeddings (i.e. embedded labels) for a given objective. Intra-batch matching works by matching video embeddings to the right text embeddings and vice versa *within* a batch instead. Intra-batch matching is therefore similar to the original ActionCLIP training strategy. The traditional way to perform separate auxiliary objective predictions in a multi-task learning strategy is to add a decision head for each additional objective in the form of a fully connected layer [4]. As we have seen in Section 2.3.4, ActionCLIP does not use a fully connected head for its predictions but uses a cosine matching head instead. Thus, we add a cosine matching head for every additional objective instead of a fully connected layer. For example, for the *gender* objective, we obtain an embedding of a video that contains a woman performing an action and use cosine matching to measure similarity to either all possible gender labels in the dataset (inter-batch) or to all gender labels contained in that particular batch (intra-batch) in the dataset. We then select the most similar label as the decision head's prediction. Inter-batch matching is illustrated in Figure 25. We perform cosine matching between video embeddings and all possible labels of a certain pseudo-objective rather than matching video and text embeddings in that specific batch. It follows that the loss calculated over the logits of each objective is no longer contrastive. Cross-entropy (CE) is then calculated over the video-to-text logits of the cosine matching heads. We also include a variant of this cross-entropy approach that also computes text-to-video cosine similarity between all possible labels of an objective and the videos in a batch. We do this to re-align the text-to-video head with the updated video-to-text head, thus better preserving the cross-modal embedding space.

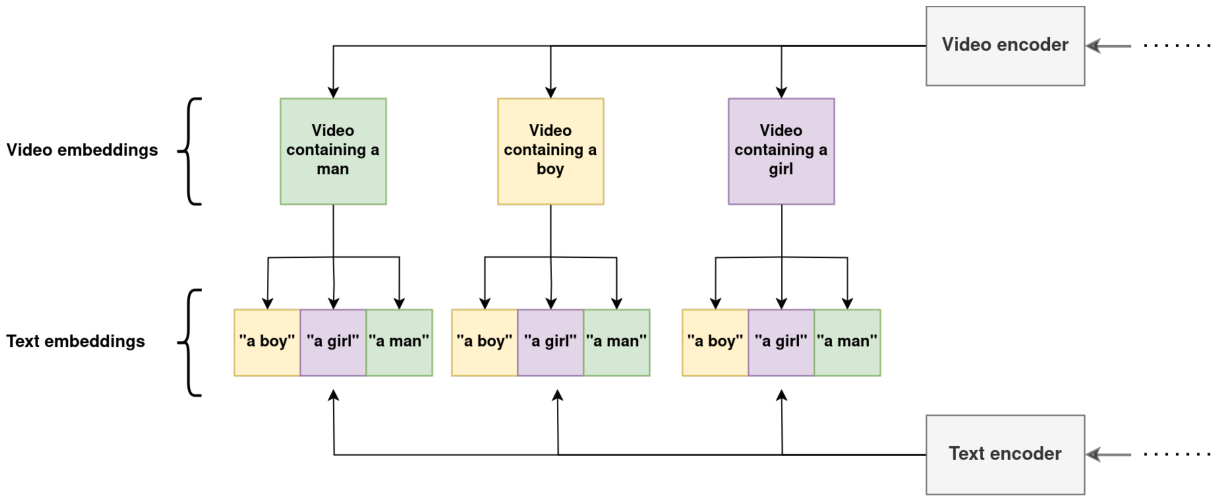


Figure 25: **Inter-batch: Cross-entropy cosine matching**. Shows the matching process for the *gender* objective. Logits are calculated between every video embeddings in a batch and every possible label of the *gender* objective (we choose to show only three of the possible genders to reduce illustration complexity). CE loss is calculated over the logits to quantify the loss of the inter-batch prediction.

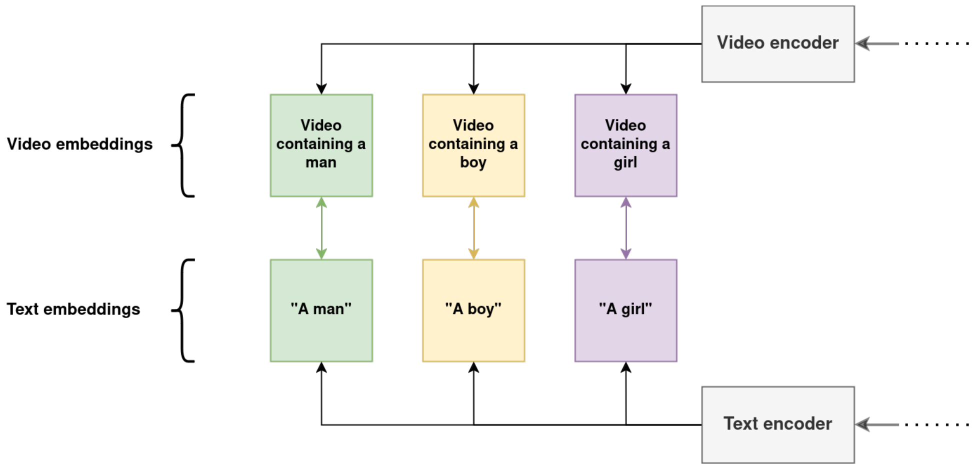


Figure 26: **Intra-batch: Contrastive cosine matching.** Shows the matching process for the *gender* objective. Each video embedding in a batch is matched to each text embedding in the batch, and vice versa. This is similar to the original ActionCLIP training strategy, but repeated multiple times for each additional pseudo-objective. KLL loss is calculated over the logits to quantify the loss of the intra-batch prediction.

Despite the variant that performs text-to-video re-alignment, the inter-batch cross-entropy approach is not a method that directly fits the cross-modal language-text embedding space of the model. As we mentioned in Section 2.3.4, ActionCLIP makes test-set predictions by generating a video embedding, and the model then calculates the cosine similarity between that embedding and the text embeddings of each action label (e.g. “doing a handstand”, “waving”..). It follows that in order to match the correct video embedding to the correct text embedding, we require both a solid video-to-text alignment *and* a solid text-to-video alignment. For that reason, we provide a second way to present the separate objectives to the ActionCLIP model in a way that respects the cross-modal space that it operates in. We do this by training the model to recognise the additional objectives in exactly the way that the main action objective is trained, which is contra/stively. We employ one contrastive matching task between video-to-text and text-to-video logits within a batch for each objective. This is therefore an intra-batch matching strategy. Kullbach Leibler divergence loss is then calculated over the matched pairs. This creates a cross-modal embedding space in which corresponding video-text pairs are rewarded or penalised depending on whether they were matched correctly. The approach is illustrated in Figure 26.

HMDB51 Experimental results

Backbone	Backbone finetune	Objectives	Loss mode	Top-1	Top-5
TinyCLIP-ViT-40	No	Multi	Contrastive (intra-batch)	**63.40**	90.00
TinyCLIP-ViT-40	No	Multi	CE one-way (inter-batch)	60.80	90.20
TinyCLIP-ViT-40	No	Multi	CE two-way (inter-batch)	61.50	90.30

Table 7: **Experimental results of comparing intra-batch and inter-batch learning strategies.** It is shown that the intra-batch learning strategy outperforms the inter-batch approach. CE one-way indicates that cross-entropy loss is calculated over the video-to-text logits for each of the additional objectives. CE two-way is similar, but also calculates a CE loss over the text-to-video logits.

HMDB51 Top-1 validation graph

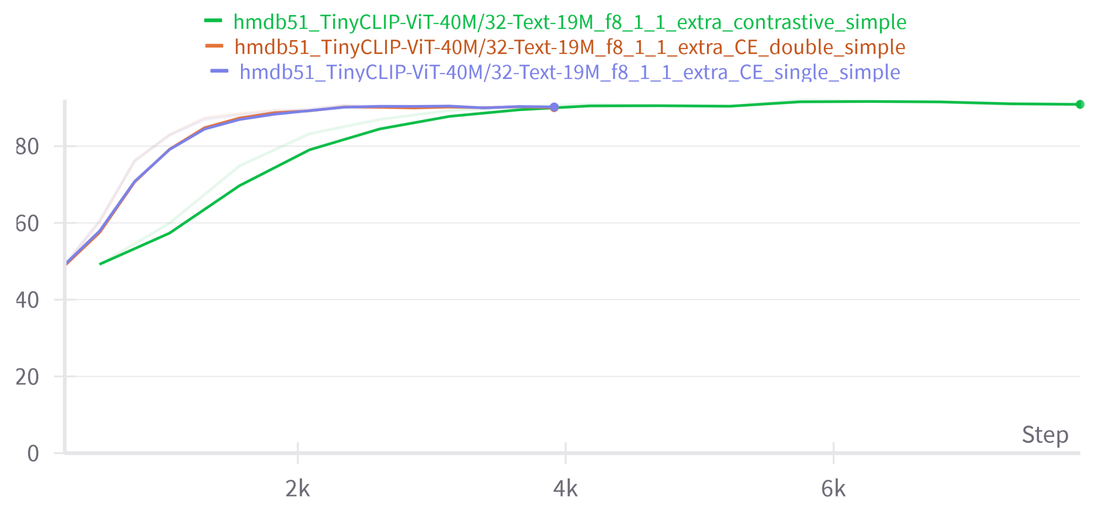
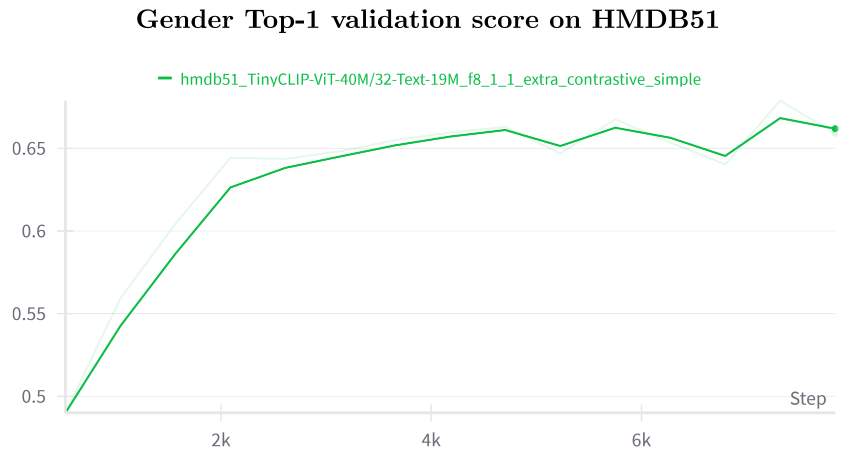


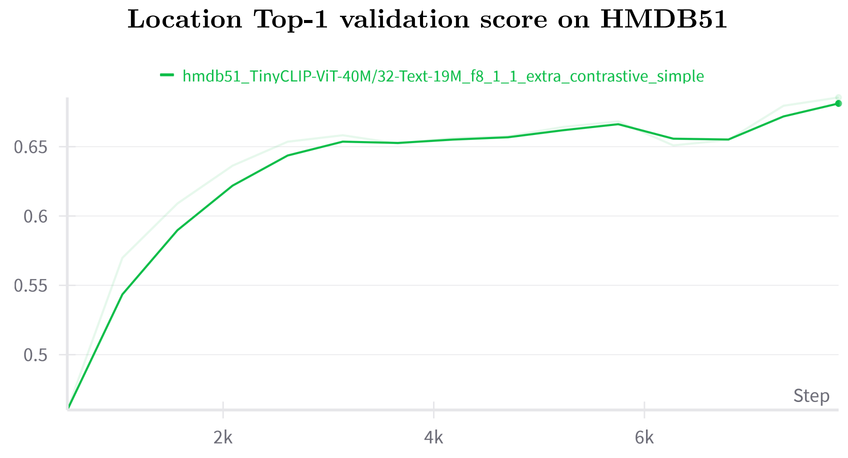
Figure 27: **Multi-task objective representation results graph on HMDB51.** Top-1 performance on HMDB51 with multiple objectives using video-to-text CE (blue), video-to-text and text-to-video CE (orange) and the contrastive approach (green). The contrastive single label results serve as the (vanilla) baseline performance. Peak top-1 of the CE-based strategy that calculates both video-to-text and text-to-video loss (orange) is 61.57, slightly outperforming the one-way CE approach that only includes video-to-text logits. The Top-1 performance of the contrastive approach (green) surpasses both with a score of 63.43.

We test both the inter-batch CE strategy and the contrastive intra-batch strategy. In our experiment, we sum the original contrastive loss over the ground truth action label and the additional objectives. We choose a 50/50 distribution in which the original contrastive loss over the ground truth label counts for

50% and the additional three objectives count for 16.66% each.



(a)



(b)

Figure 28: **Top-1 accuracy on additional objectives in the validation set.**The graph shows ActionCLIP's predictive performance on two additional objectives in the validation set, namely the *gender* category (a) and the *location* category (b). The model was trained using a contrastive (separate) multi-objective learning strastegy. The figure shows that the model obtains a Top-1 score of more than 60.0 on both additional objectives.

Separated multi-objective experimental results

From Figure 27 it becomes clear that the inter-batch cross-entropy approaches "CE double" (orange) and "CE single" (blue) converge faster than the contrastive intra-batch approach (green), but peak Top-1 performance of the contrastive approach is much higher. That is, presenting the additional objectives as contrastive matching tasks easily outperforms presenting the additional objectives as an inter-batch classification problem. To gain more insight into ActionCLIP's capability to learn the spatial details presented in the set of additional pseudo-objectives, we also generate pseudo-labels for the validation set and evaluate ActionCLIP on its ability to correctly predict the spatial objectives on that set. We choose to evaluate the contrastive learning approach because it is more effective than the cross-entropy learning approach. Figure 28 shows the Top-1 score of ActionCLIP predicting two of the three spatial objectives in the validation set. The model learns to predict the three pseudo-objectives with a Top-1 score between 0.60 to 0.66. In this Top-1 score, wildcard labels are disregarded entirely. From this, we conclude that the model has learned to distinguish the objectives. It also shows that multi-objective contrastive learning is possible and that guiding the same video embedding to multiple, different textual embeddings does not interfere with the existing video-text embedding space. For example, learning the model to associate a certain video embedding with the "in a park or backyard" pseudo-objective does not interfere with the other video-text associations like "shooting a bow" or "wearing a sweater". If that were the case, we would expect one validation Top-1 score on a certain objective to improve at the cost of another objective.

HMDB51 Experimental results

Backbone	**Backbone finetuning**	**Loss**	**Objectives**	**Top-1**	**Top-5**
TinyCLIP-ViT-40	Yes	Contrastive	Multi	**64.50**	91.00
TinyCLIP-ViT-40	Yes	Contrastive	Single	63.40	90.00

Table 8: **Experimental results of applying multi-objective learning.** In this table, the multi-objective approach is compared to the vanilla single-objective strategy.

HMDB51 Top-1 validation graph

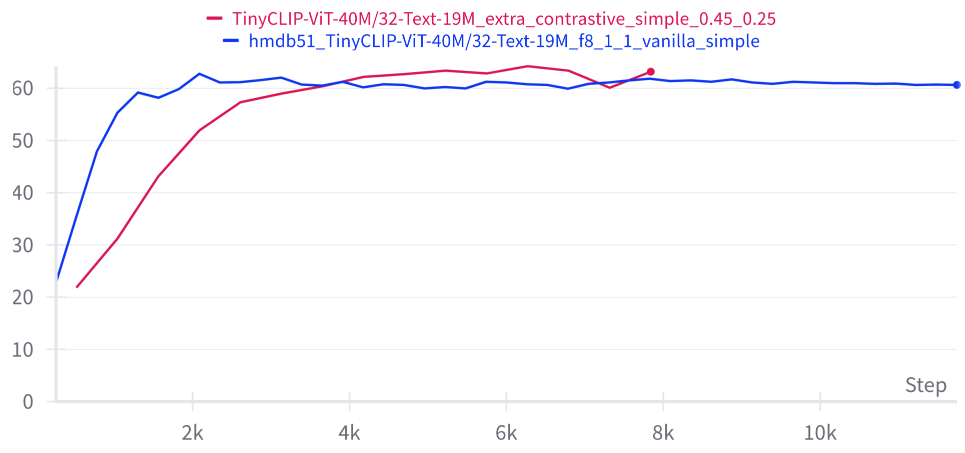


Figure 29: **Spatial distillation graph of Top-1 on HMDB51**. This figure shows the impact of spatial distillation through predicting additional spatial objectives on the HMDB51 dataset. We can see that the multi-objective training strategy (red) with a Top-1 of 64.50 outperforms the baseline vanilla training strategy (blue) with a Top-1 of 63.40. For both models, active backbone finetuning was used.

Spatial distillation compared to the vanilla ActionCLIP implementation

Comparison results between our spatial distillation training strategy and the original ActionCLIP implementation can be seen in Figure 29. We compare the best performing **separate multi-objective** objective representation using a (contrastive) intra-batch training strategy (red). The additional spatial objectives that are used are the *gender*, *location* and *clothing* objectives and each additional objective is matched separately using its own cosine matching head. These heads are added to the original matching head for the ground truth label. Loss is distributed in a 50/50 split between the ground truth prediction and the additional pseudo-objectives. The configuration is compared to the vanilla single-objective training strategy (blue). Both configurations use active backbone finetuning to give us an idea of the synergy between active backbone finetuning and multi-objective learning. As with all experiments, we provide more detailed results in the ablation study in Section A. As becomes clear, the multi-objective training strategy is slower to converge but results in a higher Top-1 validation score on the HMDB51 validation set. We further discuss these results in the discussion section.

7 Global CLIP-based AR model distillation

7.1 Motivation

Rather than distilling the separate sources of knowledge contained in the ActionCLIP model such as its encoder backbone or the spatial knowledge contained in the CLIP model, we can also choose to distill a model into a smaller student model globally. This means that the model's separate components are ignored and the only goal is to make the student model match the teacher model's predictions as closely as possible. In Section 2.4 we found that multiple relevant distillation techniques exist that do so. In this chapter, we adapt the DeiT distillation technique as proposed by Touvron et al. to distill a cross-modal AR transformer model. We do this because to the best of our knowledge, the DeiT approach has not been applied to (cross-modal) AR transformer models despite its success in distilling image transformers. The DeiT strategy focuses exclusively on distilling into a transformer architecture, making it particularly suited for our purpose.

7.2 Experimental set-up

In order to assess the effectiveness of the DeiT approach, we experiment with DeiT distillation on the ActionCLIP model. We choose an ActionCLIP model with a pretrained TinyCLIP-ViT-40M/32-19M backbone as the student model. Originally, the number of parameters in the ActionCLIP model using a CLIP-ViT-B/32 backbone is 144.1M. By replacing the backbone with this TinyCLIP variant, we reduce the number of backbone parameters from 126M to 59M, thereby lowering the total model parameters from 144.1M to 77.1M. Because the backbone accounts for more than 85% of the parameters in the model, we choose not to further remove parameters in the (post-CLIP) fusion layers of the ActionCLIP model. As for the teacher model, we use an ActionCLIP model that is pretrained on the Kinetics400 dataset and that uses a CLIP-ViT-B/32 backbone. We choose the CLIP-ViT-B/32 backbone over the more powerful CLIP-ViT-B/16 backbone because of memory constraints. These memory constraints stem from the fact that the number of patches per image increases quadratically moving from CLIP-ViT-B/32 to CLIP-ViT-B/16 backbone. For example, one colour channel of 224×224 pixels contains 49 patches when divided in patches of 32×32 but contains 196 patches when the patch size is 16 instead, drastically increasing the memory requirements of the model. Both models are run in parallel and follow the configuration described in Section 3.3.

7.3 Applying the DeiT approach to the ActionCLIP model

Applying the DeiT approach to ActionCLIP is relatively straightforward. ActionCLIP transforms the n frame tokens produced by its CLIP encoder backbone into a video embedding by applying global attention over these tokens. After doing so, it applies mean pooling on the n tokens to obtain the final video embedding. Alternatively, the authors offer a more conventional $[CLS]$ token approach in which a $[CLS]$ token is prepended to the separate frame tokens before being passed through the global attention layers. The token then interacts with all frames through the subsequent global attention layers and the $[CLS]$ token at the final layer represents the video embedding [30]. Both strategies are implemented in the original ActionCLIP paper, but the authors find better performance by using the mean pooling approach. We propose a hybrid integration between these two output head strategies. The heads are illustrated in Figure 30. As can be seen, the existing mean pooling output head of the ActionCLIP model is combined with a distillation token approach indicated by *"Dist Token"* in the diagram. The distillation token functions similarly to the standard $[CLS]$ token approach of the transformer architecture, except that it is appended behind the last token instead of being prepended before the first frame token. This token interacts with all frames throughout the global attention layers of the ActionCLIP model.

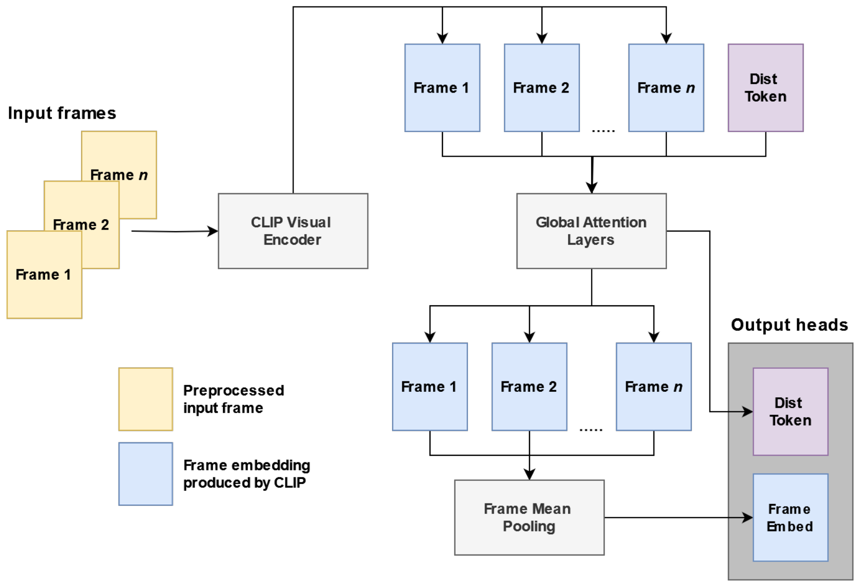


Figure 30: **Two-headed model architecture**. Combining the mean pooling and token-based output heads in the ActionCLIP model to allow for distillation through the DeiT approach. The embeddings of frames 1 to n produced by the (Tiny)CLIP Visual Encoder are passed into the model along with an appended distillation token. These tokens are then passed through the model's global attention layers. This means that the distillation token interacts with all initial frame tokens through these layers. After the global attention layers, the resulting frame tokens are mean pooled to form the mean embedding head, the distillation token itself forms the distillation embedding head.

Given the two separate output embeddings of the model, the two loss terms described in the DeiT approach are calculated: between the mean pooled embedding and the ground truth action labels and between the distillation token and the hard teacher labels. We use the Kullback-Leibler loss (KLL) function for both heads, similar to the original ActionCLIP implementation. Both heads produce prediction logits by calculating the cosine similarity between video embeddings and text embeddings in a batch (video-to-text logits) and vice versa (text-to-video logits). KLL is then calculated over these logits for both the mean pooled head and the distillation token head separately. The total loss is represented as a 50/50 split between the loss calculated over the mean pooling head and the loss calculated over the distillation token head.
As a result of this dual loss training strategy, the distillation token head is supervised to correctly calculate video-to-text and text-to-video logits between the token-based video embeddings batch and text embeddings of the hard labels provided by the teacher, whereas the mean pooling head is supervised to correctly calculate video-to-text and text-to-video logits between mean pooled video embeddings in that same batch and text embeddings of the ground truth labels of the dataset.

7.4 Experimental results

The results of applying the DeiT distillation strategy on an ActionCLIP model with a TinyCLIP-ViT-40M/32-19M backbone can be seen in Figure 31. A maximum Top-1 score of 64.30 is achieved by the student model. The DeiT distillation approach outperforms training on the HMDB51 dataset without the DeiT distillation strategy.

HMDB51 Experimental results

Backbone	DeiT distillation	Teacher backbone	Teacher T1	Top-1	Top-5
TinyCLIP-ViT-40	No	-	-	62.80	90.10
TinyCLIP-ViT-40	Yes	ViT-B-32	71.60	**64.30**	90.90

Table 9: **Experimental results of the DeiT distillation strategy performed on the HMDB51 dataset.** A minor improvement over the vanilla model's Top-1 score can be seen when DeiT distillation is applied.

HMDB51 Top-1 validation graph

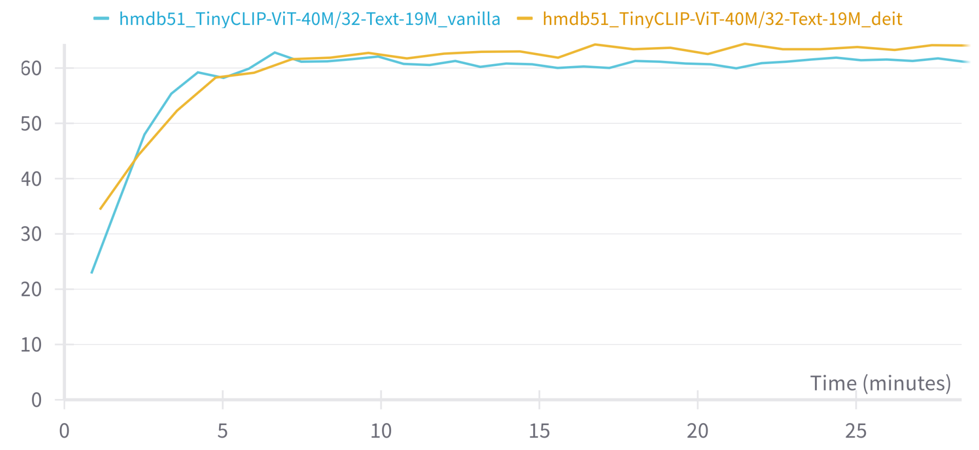


Figure 31: **Graph depicting DeiT distillation result on HMDB51**. Results of applying the modified DeiT distillation technique on the HMDB51 dataset. Batch size is set to 32. The "deit" variant (yellow) uses the previously described DeiT distillation technique. The "vanilla" (blue) baseline is configured according to Section 3.3 and uses the original ActionCLIP training strategy without applying DeiT KD.

Additional results

Surprisingly, the results of a follow-up experiment show that the increase in Top-1 score is not caused by the teacher-student interaction, but because videos are represented by both the mean embedding head and the distillation embedding head. That is, when both heads calculate contrastive loss over the ground truth labels instead of only the mean embedding head, performance is improved over using the hard teacher labels. In other words, the hard labels predicted by the teacher are discarded and loss is exclusively calculated by contrastively matching each of the two heads' embeddings with the set of ground truth labels for that batch. The total loss is a 50/50 split between the mean pooling and the distillation token heads. For validation-time inference, we find that using the embedding produced by the mean pooling head as the video representation works best, and the distillation token is disregarded entirely. We also experimented with a combination of the two heads as final video embedding by using mean pooling but find inferior performance. We call the added token the "distillation" token because that was its original purpose in the model, even though no distillation is performed in this additional experiment. The results can be seen in Figure 32. The two-headed models easily outperform the vanilla model and repeated validation runs shows that the two-headed approach (with backbone finetuning) consistently score in the 65.50 – 66.50 Top-1 range on the HMDB51 dataset using a TinyCLIP backbone and contrastive backbone finetuning. We conclude our additional results with our two-headed training strategy applied to an ActionCLIP model pretrained on Kinetics-400 that uses a CLIP-ViT-B/32 encoder

HMDB51 Experimental results

Backbone	Backbone finetune	Objectives	Loss mode	Top-1	Top-5
TinyCLIP-ViT-40	No	Single	Vanilla	62.80	90.10
TinyCLIP-ViT-40	Yes	Single	Vanilla	63.90	91.10
TinyCLIP-ViT-40	No	Single	Two-head	65.80	91.80
TinyCLIP-ViT-40	Yes	Single	Two-head	**66.50**	91.90

Table 10: **Experimental results of the two-headed strategy.** Maximum performance is obtained by using a two-headed loss strategy and backbone finetuning. The vanilla model reaches Top-1 scores of 62.80 and 63.90 with and without backbone finetuning respectively.

HMDB51 Top-1 validation graph

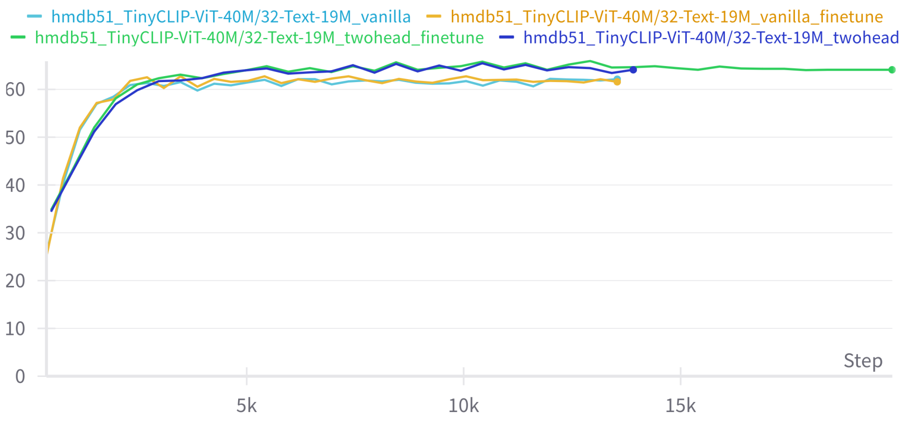


Figure 32: **Two-headed train strategy graph of the HMDB51 dataset.** This graph shows the ActionCLIP performance on HMDB51 using the two-headed distillation token and mean pooling approach (dark blue), that same model but with finetuning (green), and two vanilla models without the two-headed approach with and without finetuning (yellow and and cyan). Peak Top-1 is achieved by the two-headed variant using backbone finetuning with a Top-1 score of 66.50 whereas the vanilla model reaches Top-1 score of 62.80 and 63.90 with and without backbone finetuning respectively.

backbone. As can be seen in Table 11, the maximum Top-1 score obtained on HMDB51 by an ActionCLIP CLIP-ViT-B/32 model that is pretrained on Kinetics-400 is 72.03 whereas finetuning that same model with the two-headed approach obtains a Top-1 score of 71.00. Therefore, the two-headed training strategy likely interferes with the weights that were pretrained on Kinetics-400 using the original implementation's training strategy. That is, we suspect that the two-headed training strategy that the model is finetuned with potentially disrupts the weights the way the weights were fitted during its single-head pretraining on Kinetics-400.

Backbone	**Loss mode**	**Objectives**	**BB finetune**	**K400 pretrain**	**Top-1**	**Top-5**
CLIP-ViT-B/32	Vanilla	Single	Yes	Yes	**72.30**	94.20
CLIP-ViT-B/32	Two-head	Single	Yes	Yes	71.30	94.80

Table 11: **Two-head ablation experiment**. This table shows the Top-1 and Top-5 validation scores on the HMDB51 dataset for a model pre-trained on Kinetics-400. The pretrained weights are provided by the ActionCLIP authors. Both models sample 8 frames per video and used a batch size of 32. In this figure, we show that finetuning a model that was pre-trained using the vanilla training strategy using our two-headed approach does not cause improvement in predictive performance.

8 Discussion

In this discussion we start by answering and discussing each of the sub-questions formulated in our research approach. Next, we combine this information to answer our main research question. We then look at the limitations that come with the methods proposed in this paper. Finally, we explore avenues for future work in which we identify opportunities for extending the research and methodologies discussed in this paper.

8.1 Research question

In Section 3.2 we formulated our main research question and a set of sub-questions. In this section, we start by answering the sub-questions and then combine those findings to answer our main research question.

(Q1) Between the STAN and ActionCLIP model, which model has the best speed-accuracy ratio on HMDB51 and UCF101 when evaluated using a dense frame sampling strategy?

As becomes clear from our experimental results in Section 4, STAN and ActionCLIP perform roughly similarly in terms of predictive performance, but ActionCLIP is significantly faster for both the CLIP-ViT-B/32 and CLIP-ViT-B/16 backbone. Both models adapted fairly well to switching from a uniform frame sampling method to a dense frame sampling method for evaluating validation set performance, with both models reporting only a small decrease in Top-1 scores on the HMDB51 and UCF101 datasets. We therefore conclude that the ActionCLIP model has a slightly better speed-accuracy trade-off than the STAN model when evaluated using a dense frame sampling strategy.

(Q2) How does replacing the backbone of the CLIP-based AR model of our choice with a TinyCLIP variant influence its speed-accuracy trade-off on the HMDB51 dataset?

We have seen that the TinyCLIP model significantly speeds up the inference time of the model in Table 4, albeit at a substantial loss in Top-1 performance. More specifically, we find that moving from a CLIP-ViT-B/32 backbone to a TinyCLIP-40M/32-19M backbone causes an 11.5% drop in Top-1 performance on the HMDB51 dataset, but reduces the average inference time in milliseconds on a batch of 32 video samples by as much as 73.4%. The inference time improvement that we measured is in line with our expectations that the impact of a more condensed model such as TinyCLIP is more pronounced when used in an AR pipeline. This is because TinyCLIP serves as the image and text encoder, and is called 8, 16 or even 32 times per video sample for encoding frames alone depending on the number of frames that are sampled. The speed at which the backbone operates is therefore a substantial part of ActionCLIP's total inference time over a batch and CLIP also accounts for over 85% of its parameters. We conclude that replacing the backbone significantly reduces the inference time and size of the ActionCLIP model, albeit at a significant drop in Top-1 performance on HMDB51. This drop in predictive performance conflicts with our research criterion that the student model should have a comparable predictive performance to its teacher model. We have also demonstrated that our proposed strategy for actively finetuning the CLIP backbone proves to be an powerful way to improve the predictive performance of the backbone.

(Q3) Can we distill CLIP's spatial knowledge by predicting additional spatial classification objectives for the CLIP-based AR model of our choice to train on by applying CLIP's zero-shot predictions on video frames data, and does training on these additional objectives improve predictive performance of the language-enabled AR model of our choice?

This question consisted of two sub-questions, one that dealt with the optimal way to present the additional spatial objectives to the AR model and one that dealt with whether the loss calculated over predicting these objectives should be contrastive or based on a cross-entropy loss function. As outlined in Section 6, we conclude that between presenting the additional labels as either a combined single (descriptive) label or presenting the objectives as separate tasks, the latter is significantly more effective. In our best-performing implementation, the loss distribution is roughly a 50/50 split between the loss over the ground truth label objective and the three additional objectives of *gender*, *holding* and *clothing* combined. We hypothesise that presenting additional objectives as separate matching tasks compensates for the batch size and the lack of label variety. To see this, consider a video where the descriptive label is *"a man wearing gym clothes shooting a bow in a gym"*. If our hypothetical batch size is 51 and we train on the HMDB51 dataset that has 51 individual classes, then from a probabilistic standpoint we would expect to see exactly one occurrence of each action class if the label distribution is exactly equal in the dataset. That is, we would expect no other descriptive label that contain the words *"shooting a bow"* in our batch. The matching process can therefore be performed by looking solely at this part of the descriptive label, potentially disregarding the additional information. If we present the prompt as separate matching objectives such as *"a man"*, *"wearing gym clothes"*, *"shooting a bow"* and *"in a gym"* instead, this task becomes much harder because the model can no longer rely on a strictly unique part of the label that provides a strong clue about the correct embedding to match. The results agree with this hypothesis because the training strategy incorporating separate objective tasks easily outperforms the single-label approach described in Section 6.5.
As for the loss strategy, we find that the (contrastive) intra-batch cosine matching strategy easily outperforms the CE-based inter-batch matching strategy, as was shown in Figure 27. The figure also shows that the inter-batch cross-entropy approach performs worse than the baseline implementation. In other words, training the model to match video-to-text and text-to-video contrastively works much better. This makes sense, as both the CLIP model and the original ActionCLIP model also perform intra-batch cosine matching during training. It is therefore likely that presenting the separate additional objective tasks in this same way works best because by doing so, the video-text alignment for both the ground-truth labels and the additional labels are performed in the same way. This potentially encourages a more uniform alignment over the objectives.
The question that remains is whether this spatial distillation training strategy improves the predictive performance of the ActionCLIP model on the validation set compared to the original (vanilla) ActionCLIP training strategy. As became clear from Table 29, our best-performing spatial distillation strategy improves the predictive performance of the ActionCLIP model in terms of Top-1 and Top-5 validation scores. This provides evidence that performing spatial distillation by generating spatial pseudo-objectives using the previously discussed (optimal) strategy to do so strengthens the supervisory signal presented to the model compared to training only on the ground truth action label.

(Q4) Can the DeiT image transformer distillation approach be adapted to allow for distillation between two transformer-based teacher-student AR models?

In Section 7 we describe the ways in which we apply the DeiT technique to the ActionCLIP model and the corresponding results. We show that by adding a distillation token and an extra cosine matching head to the ActionCLIP architecture, we can perform the DeiT distillation strategy on a video transformer instead of an image transformer. Although initial experimental results indicate that the DeiT distillation approach improves performance over the vanilla training strategy, we find that this is not caused by a teacher-student interaction. In fact, we find that the increase in performance is caused by the two-headed architecture that was originally intended to accommodate the DeiT distillation strategy. Initially, we speculated that this increase was caused by the fact that an extra token was added to the frame tokens of

each sampled frame. This means that parameters are added to the model that can potentially be fitted to the data. However, even though the ActionCLIP paper uses a mean pooling approach to fuse frame-level embeddings together after global attention, the authors also experimented with a more conventional $[CLS]$ token head for their model. They report that using this classification token as the video embedding slightly degrades the performance of the model [30]. In other words, adding a classification token to the frame tokens does not in itself improve the performance of the model. It follows that the success of this two-headed approach is most likely due to the fact that the loss calculation is evenly distributed between the mean pooled frame embeddings and the token-based output head that was originally meant as the DeiT distillation token. We hypothesise that this could encourage the model to learn two perspectives. Indeed, the loss calculated over the mean pooling head potentially ensures that each frame is a good representation of part of the action performed in the video sample, whereas the loss calculated over the added single token pushes the model to also learn a global, singular understanding of the performed action. This in turn could make the model less reliant on a single representation of a video sample, therefore possibly improving its performance.
What is also interesting is that the two-headed training strategy degrades performance when used to finetune an ActionCLIP model that is pre-trained with a single (vanilla) output head. For example, the maximum Top-1 score obtained on HMDB51 by an ActionCLIP CLIP-ViT-B/32 model that is pretrained on Kinetics-400 is 72.3, whereas finetuning that same model with our two-headed approach obtains a Top-1 score of 71.0. It is therefore likely that the multiple perspectives introduced by the two-headed approach conflicts with the original (single-head) perspective that the model learned during its Kinetics-400 pre-training. This can be seen in Table 11. We conclude that one the one hand, we ran into a number of hardware related issues when applying the DeiT distillation strategy on our ActionCLIP model. On the other hand, in applying the DeiT strategy we found that the two-headed training strategy is a powerful way to increase the supervisory signal when training our model.

`(Main)` Can a CLIP-based AR model be designed that can operate in real-time scenarios at comparable predictive performance to larger CLIP-based AR counterparts by applying global distillation, spatial distillation and backbone distillation or a combination thereof?

Given the results of each sub-question presented in this work, we arrive at the main research question. In this work, we have experimented with three distillation strategies. By replacing the vanilla CLIP-ViT-B/32 or CLIP-ViT-B/16 backbone with a TinyCLIP-ViT-40M/32-19M backbone, we were able to reduce the inference time on a batch of 32 video samples with 8 sampled frames per video from 140.24 to 80.87 milliseconds, which is a 40% speed-up. However, this came at a sharp reduction in predictive performance. On the HMDB51 dataset, Top-1 performance on the validation set dropped from 68.80 to around 62.80. Backbone distillation by itself does therefore not satisfy the criterion that the distilled model should perform roughly similarly to its larger counterpart. However, in Section 6, we show that combining our multi-objective spatial distillation strategy with a TinyCLIP backbone and the contrastive backbone finetuning approach introduced in Section 5 increases HMDB51 Top-1 performance from 62.80 to 64.50. The best-performing configuration is described in Section 7.4, reaching a HMDB51 Top-1 score of 66.50 by using the TinyCLIP-ViT-40M/32-19M backbone with a two-headed training strategy and active backbone finetuning. Surprisingly, the two-headed training strategy has little to no synergy with the spatial distillation strategy. Therefore, the spatial distillation strategy is not part of our best-performing model configuration. We discuss this limitation further in Section 8.2.
In order to answer whether this language-enabled model would be able to operate in real-time scenarios at comparable performance to its larger counterparts, we compare this model to the default CLIP-ViT-B/32 and CLIP-ViT-B/16 variants in Table 8.1. From the results it becomes clear that our best-performing model is significantly faster than the fastest original ActionCLIP backbone, the CLIP-ViT-B/32. It easily meets the requirements set out in our research question in that it can be used in real-time scenarios and that inference can be applied to multiple concurrent camera streams at the same time on a single GPU. In terms of predictive performance there is a 3.46% decrease in Top-1 when moving from the CLIP-ViT-B/32 backbone to the much smaller and faster TinyCLIP backbone when evaluated on HMDB51. This means that we have succeeded in applying our threefold distillation strategies in order to obtain a model that

Experimental results on HMDB51

Backbone	Inference time(ms)	Frames	Raw video p/sec	Top-1	Parameters
CLIP-ViT-B/16	495.80	8	64.50	71.70	144M
CLIP-ViT-B/32	140.20	8	228.20	68.80	145M
TinyCLIP-ViT-40*	80.90	8	395.60	66.50	77.1M

Table 12: **Speed-accuracy comparison of ActionCLIP on the HMDB51 dataset.** The average inference time in milliseconds as measured by the CUDA timing library over a batch of 32 video samples. Models are set to evaluation mode but no further model compression was performed. The inference time includes all input frame processing such as normalisation and cropping but does not include encoding label prompts because this is a one-off calculation. The experiment was conducted using the experimental set-up described in Section 3.3. The raw videos per second metric represents the total number of videos the model can theoretically process in one second when 8 frames per video are sampled. Note that in calculating this metric, we disregard memory constraints as these are outside the scope of this study. It serves purely as a theoretical benchmark. ***:** Trained using the two-headed training strategy and contrastive backbone finetuning, as described in Section 7.4

is able to operate in real-time scenarios at comparable predictive performance to its teacher. We further illustrate this comparison in Figure 33 in which we plot the average inference time in milliseconds against its validation Top-1 on the HMDBB51 dataset. All experimental results are reported in more detail in the appendix Section A. We do note that the inference performance reported in the table is not completely representative. That is, on the one hand we could reduce the inference time of all models (including the vanilla CLIP-ViT-B/32 and ViT-B/16 backbones) faster by using further model compression techniques used in production scenarios. In a real-world context without model compression, the performance would be slightly slower than the reported videos per second in Table 8.1. This is because in our test scenario, the dataloader readily serves a full batch of 32 video samples, which is faster than performing 32 separate inference passes, one for each camera stream. This could be mitigated by synchronising the hypothetical incoming camera streams so that they deliver their 8 frames per second at exactly the same time, after which they could be batched together and dispatched to the model in a single inference pass. However, evaluating the viability of such a strategy is outside of the scope of this work.

8.2 Limitations

Spatial distillation limitations

The increased supervisory signal provided by the additional labels does come with some limitations. In our research, we observed that a larger batch size generally leads to better performance. This makes sense, as the task of contrastively matching videos embeddings and text embeddings becomes harder and therefore provides a stronger supervisory signal. However, the extra contrastive objectives and in particular their similarity matrices take up a significant portion of extra VRAM. This means that the regular training strategy can in fact beat the multiple-objective strategy simply because it can be run with a higher batch size. More specifically, the default training strategy with a batch size of 128 goes toe-to-toe with our multi-objective training strategy using a batch size of 94. That said, we expect the multi-objective strategy to beat the default training strategy if the available amount of VRAM would allow a batch size of 128 so that they could be compared on equal footing. A second limitation of this strategy is the fact that it consistently improves the Top-1 scores of the vanilla ActionCLIP model on HMDB51, but not the performance of the two-headed ActionCLIP variant described in Section 7.4. This is a limitation because the two-headed training strategy causes a substantial jump in Top-1 performance in the ActionCLIP model for both the vanilla CLIP and the TinyCLIP backbone. The are many possible reasons as to why these models do not work well together. For one, it is unclear whether these additional objectives should be predicted by the mean pooling head, the token head or by both heads. Also, the way in which these objectives should interact with the additional output head is not clear. All three

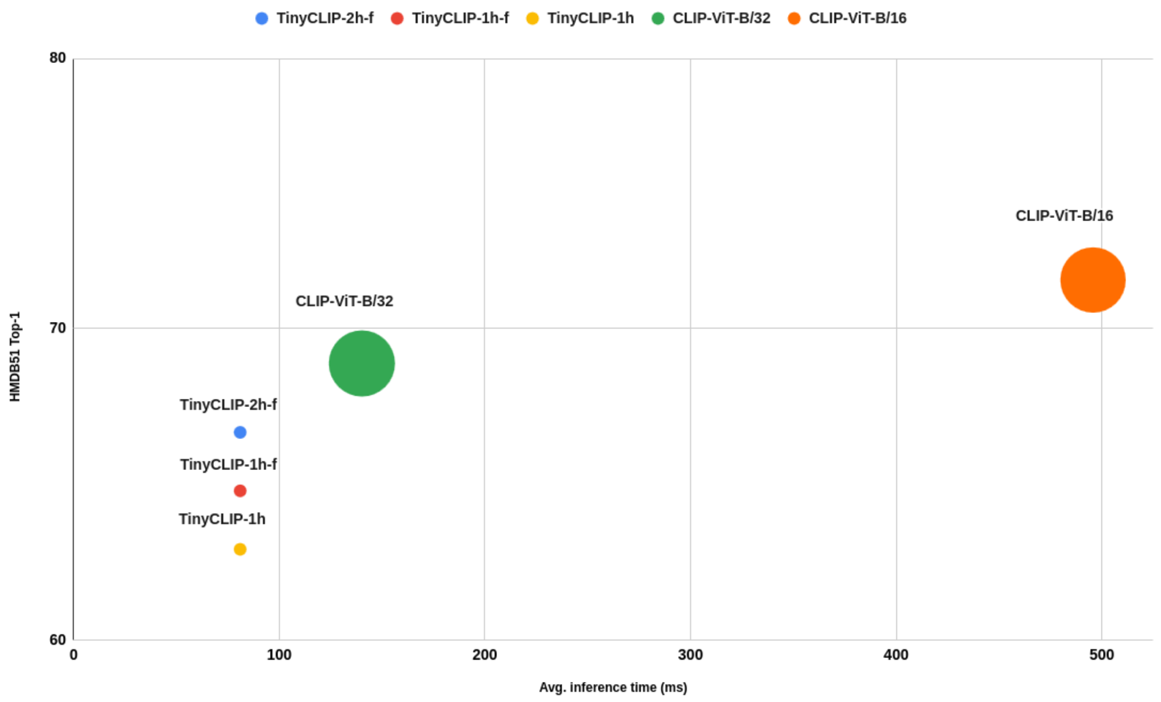


Figure 33: **ActionCLIP architecture comparison on HMDB51**. This bubble diagram shows a comparison between different ActionCLIP configurations. The names indicate the backbone. "2h" indicates that the two-headed training strategy was used whereas "1h" indicates a single-headed training strategy. An appended "f" indicates that active backbone finetuning was applied during training. The CLIP-ViT-B/32 and CLIP-ViT-B/16 configurations serve as a baseline and are trained using the original ActionCLIP training strategy (i.e. with a single-headed training strategy and no active backbone finetuning). Size indicates the number of parameters in the model. The average inference time is the average inference time over all batches of 32 video samples in the validation set.

prediction combinations were tried using different loss distributions, but we found no improvement over the two-headed, single-objective training strategy. It is therefore possible that a different interaction between the additional objectives and the two-heads is required or that the two strategies are simply not compatible.

Global distillation limitations

The most fundamental problem we encountered when applying DeiT distillation was hardware related. Running a student model alongside a teacher model with a large backbone like the CLIP-ViT-B/16 or even the CLIP-ViT-B/32 requires a lot of memory. Ideally, we would like the teacher to be as large as possible, but this constrains the maximum batch size that can be reached to train the student model. This meant that in our DeiT training strategy, we could fit a maximum of 16 samples in a batch, far less than the maximum batch sizes of 64, 84 or 94 that we were able to train our best performing models on. We believe that this small batch size is sub-optimal for a model that learns by contrastively matching video-text pairs in a batch because it makes the matching process easier, thus lowering the strength of the supervisory signal.
The original CLIP training strategy uses a very large batch size for that same reason [21]. We provide evidence for this suspicion in Figure 34. From the follow-up experiment it becomes clear that the Top-1 performance of the ActionCLIP model is significantly impacted by a change in batch size. We

HMDB51 validation Top-1 table result

Backbone	Batch size	Backbone finetune	Objectives	Loss mode	Top-1	Top-5
TinyCLIP-ViT-40	84	Yes	Single	Two-head	**66.50**	91.90
TinyCLIP-ViT-40	32	Yes	Single	Two-head	65.00	90.10
TinyCLIP-ViT-40	16	Yes	Single	Two-head	64.20	91.70

Table 13: **Experimental results on decrementing batch size on HMDB51.** These results provide evidence that the batch size plays an important factor in the contrastive learning process. Runs are identical other than their batch size. The last number in the name of the run indicates the batch size. Peak Top-1 is obtained by the run with a batch size of 84 with a score of 66.50. In contrast, the run with a batch size of 16 results in a maximum Top-1 score of 64.20.

HMDB51 validation Top-1 graph

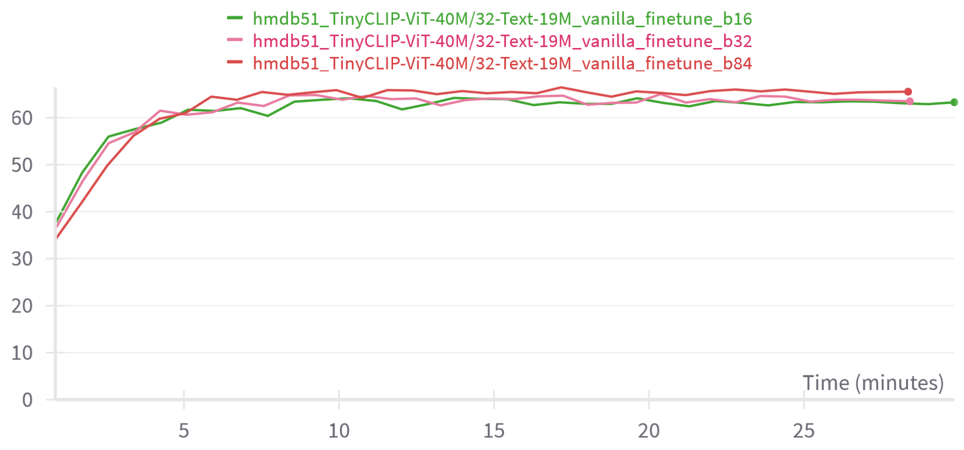


Figure 34: **Graph showing performance changes on decrementing batch size.** This graph shows the Top-1 convergence for runs of different batch size. The graph shows that decreasing the batch size consistently lowers the Top-1 scores obtained by each model.

therefore conclude that the results we obtained with our modified DeiT training strategy are not entirely representative of the effectiveness of the DeiT approach in distilling AR models due to the previously mentioned set of hardware limitations. Another limitation that we came across is the limited amount of class alternatives in the HMDB51 dataset. According to the authors, one reason why DeiT distillation is so effective is because it maintains a supervisory signal even though augmentations can cause dissonance between the ground truth label and what is shown in an image [25]. For example, consider the sample of the ImageNet-1K dataset labelled as “tow truck” in Figure 35. A random crop on this specific image as part of an augmentation pipeline could focus on a tree, street sign, traffic light, car mirror or car wheel, all of which are valid ImageNet-1K categories. What is more, there are many ImageNet-1K categories that a random crop could focus on in similar pictures labelled as “tow truck” such as a school bus, house, ambulance, restaurant or a sports car. In these cases, the teacher can provide the correct label even though the ground truth label is incorrect for that particular crop. However, the HMDB51 dataset has only 51 categories. This means that category mix-ups due to unlucky crops are not as likely to occur and if they do, the teacher does not have a lot of alternative action labels to choose from.
Neither the hardware and dataset limitations are reason to conclude that the DeiT approach is not suitable for distilling cross-modal AR models like ActionCLIP. More powerful hardware and larger datasets with more class variation exist. This way, teacher and student could run in parallel with a suitable

batch size and the teacher would add more value to the supervisory signal by being able to disagree with the ground truth label and offer an alternative label. We speculate that applying the DeiT distillation strategy on ActionCLIP to distill on the much larger Kinetics-400 or Kinetics-710 dataset instead of HMDB51 and by using multiple GPUs for a suitable batch size would provide much better distillation results compared to the experimental results ran on our limited hardware. The potential solutions to this are further discussed in Section 8.3.

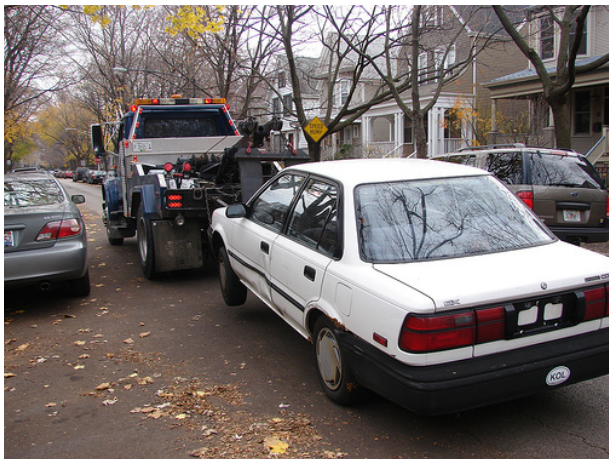


Figure 35: **A sample of the ImageNet-1K dataset labelled as "tow truck".** This sample illustrates how a random crop can cause a dissonance between the ground truth label and the information contained in a crop.

Backbone distillation limitations

We have previously discussed the fact that TinyCLIP variants exist that have a slightly better speed-accuracy trade-off. For example, the TinyCLIP-ViT-45M/32-18M performs similarly on the ImageNet 1K dataset to the CLIP-ViTB/32 at exactly half the number of parameters. However, it performs automatic weight inheritance of the teacher model and therefore its embedding dimensions can differ from the manually composed TinyCLIP family members. This makes the existing language-enabled AR model of or choice, ActionCLIP, incompatible with this particular backbone. In order to integrate this possibly faster backbone, more work is needed to adapt the ActionCLIP architecture to the different embedding dimensions. We therefore limit ourselves to the ViT-40M/32-19M variant but recognise that the ViT-45M/32-18M variant would likely perform slightly better.

8.3 Future work

Dealing with hardware constraints

As extensively discussed and evaluated in Section 8.2, we ran into hardware limitation problems that prevented us from properly evaluating the DeiT distillation strategy on the ActionCLIP model. We think that a worthwhile area of investigation is simply to repeat these experiments on more powerful hardware. We hypothesise that this would alleviate the two main limitations of our DeiT experiments: the fact that we were only able to train on the relatively small HMDB51 dataset (with only 51 possible classes) and the small batch size of 16 that we were forced to use when training our student model. For example, training a student model with an ActionCLIP ViT-B/16 teacher variant on Kinetics400 with a batch size of 32 or 64 would likely provide better results because the contrastive task becomes harder and the teacher model has more room to correct the previously discussed dissonance between the ground truth label and what is actually depicted within the video frames. Another potential solution to the hardware constraints involves separating inference of the teacher and student models. That is, the teacher model

can perform inference on an entire dataset, generating and storing hard predictions for each sample. Subsequently, the student model can be trained exclusively on these stored hard labels. This approach eliminates the need for parallel processing, allowing for the models to be loaded into system memory sequentially rather than simultaneously, thereby mitigating the hardware constraints. An approach like this called TinyViT is proposed by Wu et al., in which they provide a framework to distill ViT models on limited hardware resources.
Although not nearly as affected as the DeiT strategy, a hardware update could also be used to further experiment with the multi-objective spatial distillation strategies discussed in Section 6. As we have seen in Section 8.2, this experiment was also affected by batch-size limitations when comparing it to the vanilla training strategy, although to a much lesser degree than the global distillation DeiT experiments. However, it is unlikely to benefit from the sequential TinyVit distillation strategy because all additional objective gradients have to be loaded into memory at the same time during training.

Extending the spatial distillation strategy

Results show that the spatial distillation strategy proposed in Section 6 is promising. By extensively studying CLIP's limitations and how to deal with them, we were able to provide a clear outline of which pseudo–objectives can and cannot be reliably predicted by the model. However, the problem at the core of this training strategy is to find an overlap between objectives that CLIP can reliably predict and objectives that improve the predictive performance of an AR model. This can prove to be a challenge. A logical next step would therefore be to use even more powerful language-enabled vision models to generate these additional objectives. In our work, we used a CLIP-ViT-B/14 model to generate spatial pseudo-labels for video frames, but more powerful language-enabled models exist. For example, one could also employ a model like OpenAI's ChatGPT4.0 to generate these objectives for specific video frames. This would come at both a computational and monetary cost, but would likely unlock the generation of many objectives previously impossible with CLIP. For example, in Figure 19a we illustrated that a limitation of the CLIP model is its inability to deal with logical statements contained within prompts. We suspect that, given that the ChatGPT model family are consumer-facing models, these models would be more capable to deal with this and the other limitations that we found. For example, to the best of our knowledge the unreliable behaviour of the CLIP model in providing long prompts is not present in the ChatGPT model family. What is more, we suspect that the information that can be extracted using the ChatGPT4 family can be tuned more granularly given its ability to deal with more complex, detailed prompts. It would also allow for a more controlled response by asking the model to respond in a certain format fit for the ActionCLIP model to train on. That is, we could instruct the model to present spatial pseudo-labels exactly in the template that we want to present the objectives in (e.g. with a preset maximum number of words or tone of voice that works best for the model). For example, one could ask the model

```
“Briefly describe the following characteristics for this picture.  Please be concise,
use no more than one sentence per objective.  Write the sentence as if you are
certain.  If you do not know an objective, mark it with an [UNCERTAIN] tag and proceed
with the other characteristics, do not guess.  The objectives are the gender, location
and setting, object(s) the person is interacting with and what they are wearing.”
```

By supplying this prompt, we specifically ask the model to be concise and to use an uncertainty tag when a specific spatial detail cannot be determined. These tagged labels can then be ignored during training. When this prompt is presented to the ChatGPT 4.0 model to generate spatial objectives for Image 36, the response is as follows:

- **Gender:** "Male."
- **Location and setting:** "The individual is in a room with posters on the wall."
- **Objects the person is interacting with:** "He is holding a cigarette."
- **What they are wearing:** "A white hoodie with green lettering and sunglasses."

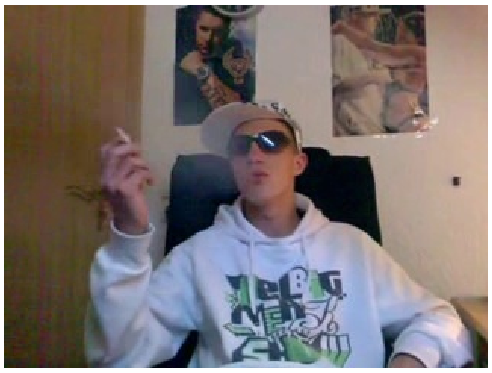


Figure 36: **A frame sampled from the HMDB51 dataset labelled as "smoking".**

Already the switch from the CLIP model to a more advanced model shows that more detail is preserved. It does not just predict a single label about what someone is wearing, but describes a combination of details about this person's clothes instead. In this case the label describes that this person is wearing a white hoodie with green lettering and sunglasses. What is more, the specific details do not have to be prompted explicitly. That is, the CLIP model will only pick up the text "A white hoodie with green lettering and sunglasses" if it is a label provided to the model. ChatGPT 4.0 is able to generate its own prompts instead, and this could prove to be very useful in providing descriptive (additional) objectives to a language-enabled AR model. More experimentation in the field of spatial distillation using ChatGPT instead of CLIP is therefore a promising avenue for future work in the field of spatial knowledge distillation. Future would should aim to answer important questions like whether the additional detailed picked up by the ChatGPT 4.0 model provides a benefit over the more coarse information contained in the spatial labels generated by CLIP, and how reliably ChatGPT can gauge its own (un)certainty on a given classification task.

Temporal pseudo-labels

Given our progress in distilling spatial information from a large teacher model, a promising next step could be to extend these methods to extract temporal information. A key challenge will be identifying a suitable teacher model. Ideally we would utilise a counterpart to CLIP that possesses a breadth of knowledge comparable to CLIP but that is tailored to video analysis rather than image analysis. To the best of our knowledge, no such model exists. This is in part due to the fact that the significant increase in size of moving from an image to a video representation makes training a model on the scale of CLIP challenging [30]. However, an ActionCLIP model pretrained on a large dataset such as Kinetics400 or Kinetics600, or the pretrained VideoCLIP model proposed by Xu et al. that is capable of video-to-text reasoning could both provide good starting points. Given a suitable teacher model, effective temporal objectives have to be determined that meet the following two criteria: they can be predicted reliably by the teacher model and they improve the supervisory signal sent to the student model tasked with predicting these temporal pseudo-objectives. At the same time, these objectives have to present new information that is not present in the ground-truth action label. We hypothesise that objectives such as step patterns (i.e. walking, running) jump patterns, head movements and rotational movements could all potentially present valuable temporal knowledge to the model. Finding a teacher model that is capable of detecting these basic temporal patterns could however prove challenging and this capability will entirely depend on the data it is pretrained on. That said, by implementing a multi-task training strategy that incorporates temporal pseudo-objectives, the predictive performance of a language-enabled AR model could potentially be improved similar to how spatial objectives improved the predictive performance of the ActionCLIP model.

Two-headed experimentation

The two-headed training strategy proposed in Section 7.4 is equally promising. It significantly outperforms its vanilla (mean pooled, single head) counterpart in every configuration that we experimented with. The dual representation that the model learns clearly helps to create semantically rich video embeddings out of video frames that in turn aid its predictions at test time. An interesting avenue of research would be to further experiment with the proposed two-headed approach in the context of vision transformers. One could even experiment with more than the proposed two heads, that each guide the AR model to represent the video in a specific perspective during training only. This could potentially intersect with our work on generating spatial objectives. For example, a "spatial" token could be added to the existing set of tokens of a video sample. This token could then be tasked to contrastively match with the correct embedded pseudo-labels for each additional spatial objective through a cosine matching head. One could even go one step further and create an additional token for each spatial objective. However, it should be kept in mind that adding tokens to the model scales the model complexity quadratically due to the global attention characteristic of the transformer. At the same time, increasing the number of trainable parameters of the model could potentially increase the required amount of data to train the multi-headed model. We therefore suspect that this approach is more likely to succeed when the model is trained on larger datasets such as Kinetics400 rather than the HMDB51 dataset.

Generalisation to larger datasets

A question that goes unanswered in this work is the performance that can be obtained if our best-performing training configuration would be pretrained on a large dataset and finetuned on the HMDB51 dataset. For example, we could pretrain our ActionCLIP model with the two-headed training strategy, TinyCLIP 40M/32-19M backbone and contrastive backbone finetuning on a very large dataset such as Kinetics-400 or even Kinetics-710. The peak performance obtained by the vanilla ActionCLIP model with a CLIP-ViT-B/32 backbone pretrained on Kinetics-400 and finetuned on HMDB51 is a Top-1 score of around 71.0. It would be interesting to see how well our best-performing variant would do on this task. It would also provide us with an indication of whether our novel training strategies generalise to much larger, more complex datasets.

Implementing other TinyCLIP variants

We only implemented a single TinyCLIP variant, namely the model with 40 million ViT parameters, 19 million text parameters and a patch size of 32 because it is roughly half the size of the CLIP-ViT-B/32 variant. As briefly mentioned in Section 8.2, TinyCLIP family members exist that perform slightly better than the TinyCLIP-ViT-40M/32-19M in terms of speed-accuracy, especially the set of models that use automatic weight inheritance as described in Section 2.4.3. It would therefore be interesting to measure the change in speed-accuracy upon integrating these models as additional backbone options for ActionCLIP. This can be done by adapting the ActionCLIP architecture to allow a variable embedding size produced by its CLIP-based encoder backbone. This in turn makes it possible to use these TinyCLIP architectures to function as ActionCLIP's encoder similar to the TinyCLIP architectures that use an embedding size similar to the vanilla CLIP model(s).

9 Conclusion

We have proposed three different distillation techniques that each aimed to distill specific portions of the knowledge contained in the CLIP model and its AR counterparts like ActionCLIP. These three strategies all served a single goal, namely to optimally utilise the vast amount of knowledge contained in large, language-enabled vision models in order to transfer their benefits to smaller, faster student models capable of operating in real-time scenarios at roughly similar predictive performance. Apart from achieving faster inference speeds, we stated that is critical to ensure that these distillation strategies

could be implemented within the constraints of conventional hardware setups, avoiding the need for extensive arrays of cutting-edge GPUs traditionally employed in (pre)training very large models like CLIP. To this end, we saw that replacing the CLIP backbone of the ActionCLIP model by a readily distilled TinyCLIP family member with half the parameters significantly sped up the inference time. However, this came at a significant drop in predictive performance. We then proposed a strategy to contrastively finetune the backbone while training the ActionCLIP model and found a consistent increase in predictive performance. Furthermore, our experiments showed that spatial distillation through generating spatial pseudo-objectives using the CLIP model further improved the predictive performance of the ActionCLIP model. Finally, we looked at ways to distill the ActionCLIP model as a whole through a modified DeiT distillation technique. We found a significant increase in predictive performance, but a follow-up experiment showed that this was caused by the two-headed training strategy described in Section 7.4 and not by influence of the teacher model. Instead, we found that this performance increase was likely caused by moving from a single-perspective video representation to a dual-perspective video representation in which both a frame-based and token-based representation is taught to the ActionCLIP model. That said, we found no synergy between this strategy and the multi-objective spatial distillation strategy.
The best performing model in terms of speed-accuracy ratio was shown to be an ActionCLIP model with a TinyCLIP-ViT-40M/32-19M backbone using our proposed two-headed training strategy and our active backbone finetuning approach. This model reached a Top-1 validation score of 66.5 on the HMDB51 validation set, reasonably close to the 68.8 Top-1 validation score of CLIP-ViT-B/32 backbone at 1.73× the speed and less than half the backbone parameters. All together, we recognise the following contributions made to the field of distilling cross-modal, language enabled AR models:

- Proposed a new strategy to distill spatial knowledge contained in large, pretrained CLIP models. This strategy involves using CLIP as part of a novel spatial label generation framework that generates spatial pseudo-labels for video samples.

- Proposed a set of new multi-task learning strategies for training language-enabled AR models on multiple objectives in a way that respects the cross-modal embedding space of these models and improves their predictive performance. We show an increase in predictive performance when the best performing strategy is used in combination with the spatial objectives generated by our prediction framework.

- Investigated the impact of backbone distillation on an existing CLIP-based AR model by replacing the ActionCLIP encoder backbone with a smaller, pretrained TinyCLIP variant.

- Proposed a backbone finetuning strategy with which the CLIP model serving as an encoder backbone for a video transformer is contrastively finetuned on video data despite being an image transformer. This is a method of which we show that it increases predictive performance of the AR model across all conducted experiments in this work.

- Adapted the existing DeiT distillation strategy so that it is able to distill cross-modal AR transformers instead of a uni-modal image recognition transformers.

- Proposed a new two-headed training strategy for cross-modal AR transformer models that guides a model to form a dual representation of a video sample. This is a training strategy that is shown to significantly improve the predictive performance of the ActionCLIP model over its vanilla implementation.

Aside from these contributions, we have provided an extensive set of promising avenues for future work in this field. We therefore conclude that this work has helped in the creation of smaller, faster student models that operates at near comparable performance to its much larger and slower counterparts. At the same time, we have made these improvements while being mindful of two important pitfalls in translating academic advances to models that operate in real-time scenarios: the realistic validation of a model's performance operating in real-time and to the ability to perform our proposed distillation techniques on a realistic hardware budget. We have therefore provided the groundwork for distilling real-time language-enabled AR models that can be applied in real-time scenarios and have laid the foundation for future work in this area.

Acknowledgements

I would like to thank Dr. Ir. R. W. Poppe, whose guidance was invaluable throughout the process of writing this thesis. I would also like to thank everyone at Oddity and especially Wikke Heesterbeek for his exceptional guidance and for his critical yet constructive feedback throughout the writing of this thesis. His dedication to rigorous research and attention to detail has been crucial in writing this thesis. I want to thank Dr. H. Kaya for his feedback as a second supervisor, and I want to thank my family for spotting mistakes that had escaped my notice. Finally, I extend a special thanks to Sophie and my cat, Flint, who have patiently endured my endless ramblings about language-enabled action recognition models.

References

[1] Anurag Arnab et al. “ViViT: A Video Vision Transformer”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 6836–6846.

[2] Gedas Bertasius et al. “Is space-time attention all you need for video understanding?” In: *ICML*. Vol. 2. 3. 2021, p. 4.

[3] Lucas Beyer et al. *Better plain ViT baselines for ImageNet-1k*. 2022. arXiv: 2205.01580 [cs.CV].

[4] Rich Caruana. “Multitask Learning”. In: *Machine Learning* 28 (1997), pp. 41–75. URL: https://api.semanticscholar.org/CorpusID:45998148.

[5] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV].

[6] Gousia Habib et al. *Knowledge Distillation in Vision Transformers: A Critical Review*. 2023. arXiv: 2302.02108 [cs.CV].

[7] Geoffrey Hinton et al. “Distilling the knowledge in a neural network”. In: *arXiv preprint arXiv:1503.02531* (2015).

[8] Berthold K.P. Horn et al. “Determining optical flow”. In: *Artificial Intelligence* 17.1 (1981), pp. 185–203. ISSN: 0004-3702. DOI: https://doi.org/10.1016/0004-3702(81)90024-2. URL: https://www.sciencedirect.com/science/article/pii/0004370281900242.

[9] Gabriel Ilharco et al. *OpenCLIP*. July 2021.

[10] Mingi Ji et al. “Show, Attend and Distill: Knowledge Distillation via Attention-based Feature Matching”. In: *CoRR* abs/2102.02973 (2021). arXiv: 2102.02973. URL: https://arxiv.org/abs/2102.02973.

[11] Chen Ju et al. “Prompting visual-language models for efficient video understanding”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 105–124.

[12] Georgios Kapidis et al. *Multitask Learning to Improve Egocentric Action Recognition*. 2019. arXiv: 1909.06761 [cs.CV].

[13] Ji Lin et al. *TSM: Temporal Shift Module for Efficient Video Understanding*. 2019. arXiv: 1811.08383 [cs.CV].

[14] Sihao Lin et al. “Knowledge Distillation via the Target-Aware Transformer”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 10915–10924.

[15] Sihao Lin et al. “Knowledge distillation via the target-aware transformer”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10915–10924.

[16] Ruyang Liu et al. “Revisiting Temporal Modeling for CLIP-Based Image-to-Video Knowledge Transferring”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2023, pp. 6555–6564.

[17] Ze Liu et al. “Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 10012–10022.

[18] Ze Liu et al. *Video Swin Transformer*. 2021. arXiv: 2106.13230 [cs.CV].

[19] Zhuang Liu et al. *A ConvNet for the 2020s*. 2022. arXiv: 2201.03545 [cs.CV].

[20] Alec Radford et al. “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8 (2019), p. 9.

[21] Alec Radford et al. “Learning transferable visual models from natural language supervision”. In: *International conference on machine learning*. PMLR. 2021, pp. 8748–8763.

[22] Karen Simonyan et al. *Two-Stream Convolutional Networks for Action Recognition in Videos*. 2014. arXiv: 1406.2199 [cs.CV].

[23] Chen Sun et al. *VideoBERT: A Joint Model for Video and Language Representation Learning*. 2019. arXiv: 1904.01766 [cs.CV].

[24] Siqi Sun et al. *Patient Knowledge Distillation for BERT Model Compression*. 2019. arXiv: 1908.09355 [cs.CL].

[25] Hugo Touvron et al. *Training data-efficient image transformers distillation through attention*. 2021. arXiv: 2012.12877 [cs.CV].

[26] Hugo Touvron et al. "Training data-efficient image transformers amp; distillation through attention". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila et al. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 10347–10357. URL: https://proceedings.mlr.press/v139/touvron21a.html.

[27] Anwaar Ulhaq et al. *Vision Transformers for Action Recognition: A Survey*. 2022. arXiv: 2209.05700 [cs.CV].

[28] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[29] Limin Wang et al. "Temporal segment networks: Towards good practices for deep action recognition". In: *European conference on computer vision*. Springer. 2016, pp. 20–36.

[30] Mengmeng Wang et al. "Actionclip: A new paradigm for video action recognition". In: *arXiv preprint arXiv:2109.08472* (2021).

[31] Qiang Wang et al. *Seeing in Flowing: Adapting CLIP for Action Recognition with Motion Prompts Learning*. 2023. arXiv: 2308.04828 [cs.CV].

[32] Wenhui Wang et al. *MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers*. 2020. arXiv: 2002.10957 [cs.CL].

[33] Kan Wu et al. "TinyCLIP: CLIP Distillation via Affinity Mimicking and Weight Inheritance". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2023, pp. 21970–21980.

[34] Kan Wu et al. *TinyViT: Fast Pretraining Distillation for Small Vision Transformers*. 2022. arXiv: 2207.10666 [cs.CV].

[35] Zhenyu Wu et al. *Privacy-Preserving Deep Action Recognition: An Adversarial Learning Framework and A New Dataset*. 2021. arXiv: 1906.05675 [cs.CV].

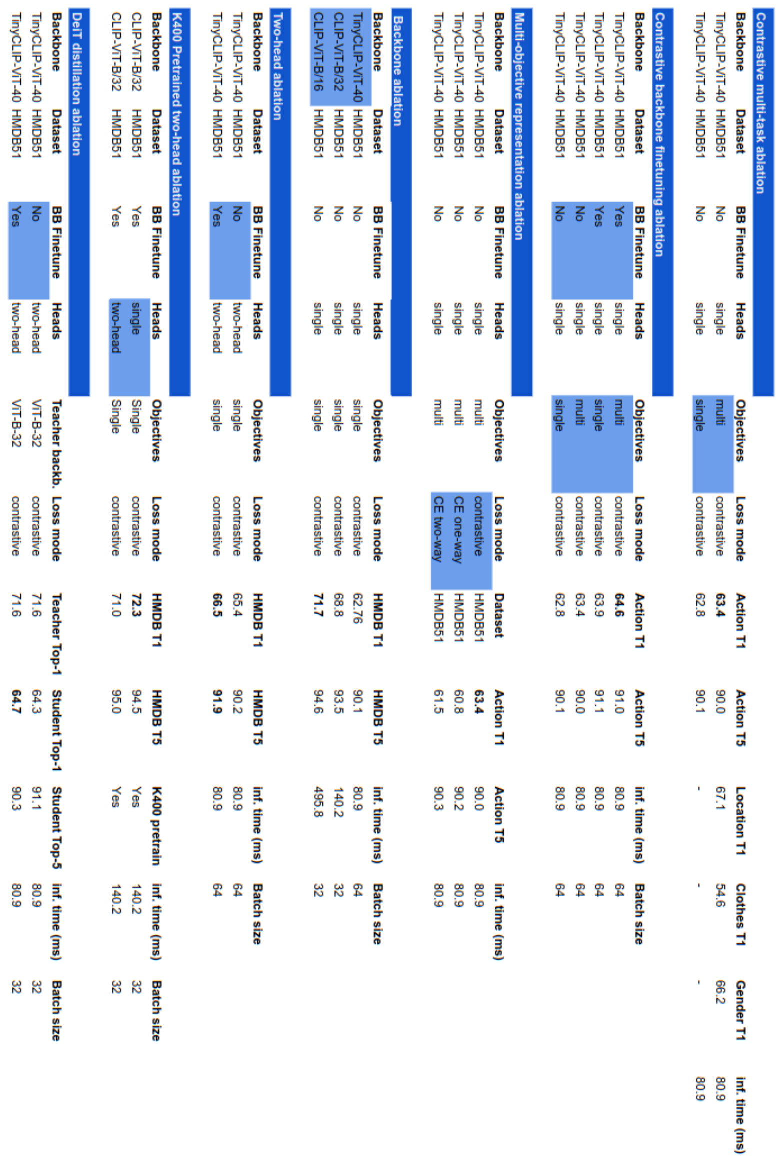
[36] Hu Xu et al. *VideoCLIP: Contrastive Pre-training for Zero-shot Video-Text Understanding*. 2021. arXiv: 2109.14084 [cs.CV].

[37] Lu Yuan et al. "Florence: A new foundation model for computer vision". In: *arXiv preprint arXiv:2111.11432* (2021).

Appendices

A Ablation study

Columns in blue mark the target of the ablation study. In all cases, performance of the ActionCLIP model is reported and frame sampling was performed as described in Section 3.3.



Contrastive multi-task ablation

Backbone	Dataset	BB Finetune	Heads	Objectives	Loss mode	Action T1	Action T5	Location T1	Clothes T1	Gender T1	inf. time (ms)
TinyCLIP-ViT-40	HMDB51	No	single	multi	contrastive	**63.4**	90.0	67.1	54.6	66.2	80.9
TinyCLIP-ViT-40	HMDB51	No	single	single	contrastive	62.8	90.1	-	-	-	80.9

Contrastive backbone finetuning ablation

Backbone	Dataset	BB Finetune	Heads	Objectives	Loss mode	Action T1	Action T5	inf. time (ms)	Batch size
TinyCLIP-ViT-40	HMDB51	Yes	single	multi	contrastive	**64.6**	91.0	80.9	64
TinyCLIP-ViT-40	HMDB51	Yes	single	single	contrastive	63.9	91.1	80.9	64
TinyCLIP-ViT-40	HMDB51	No	single	multi	contrastive	63.4	90.0	80.9	64
TinyCLIP-ViT-40	HMDB51	No	single	single	contrastive	62.8	90.1	80.9	64

Multi-objective representation ablation

Backbone	Dataset	BB Finetune	Heads	Objectives	Loss mode	Dataset	Action T1	Action T5	inf. time (ms)
TinyCLIP-ViT-40	HMDB51	No	single	multi	contrastive	HMDB51	**63.4**	90.0	80.9
TinyCLIP-ViT-40	HMDB51	No	single	multi	CE one-way	HMDB51	60.8	90.2	80.9
TinyCLIP-ViT-40	HMDB51	No	single	multi	CE two-way	HMDB51	61.5	90.3	80.9

Backbone ablation

Backbone	Dataset	BB Finetune	Heads	Objectives	Loss mode	HMDB T1	HMDB T5	inf. time (ms)	Batch size
TinyCLIP-ViT-40	HMDB51	No	single	single	contrastive	62.76	90.1	80.9	64
CLIP-ViT-B/32	HMDB51	No	single	single	contrastive	68.8	93.5	140.2	32
CLIP-ViT-B/16	HMDB51	No	single	single	contrastive	**71.7**	94.6	495.8	32

Two-head ablation

Backbone	Dataset	BB Finetune	Heads	Objectives	Loss mode	HMDB T1	HMDB T5	inf. time (ms)	Batch size
TinyCLIP-ViT-40	HMDB51	No	two-head	single	contrastive	65.4	90.2	80.9	64
TinyCLIP-ViT-40	HMDB51	Yes	two-head	single	contrastive	**66.5**	**91.9**	80.9	64

K400 Pretrained two-head ablation

Backbone	Dataset	BB Finetune	Heads	Objectives	Loss mode	HMDB T1	HMDB T5	K400 pretrain	inf. time (ms)	Batch size
CLIP-ViT-B/32	HMDB51	Yes	single	Single	contrastive	**72.3**	94.5	Yes	140.2	32
CLIP-ViT-B/32	HMDB51	Yes	two-head	Single	contrastive	71.0	95.0	Yes	140.2	32

DeiT distillation ablation

Backbone	Dataset	BB Finetune	Heads	Teacher backb.	Loss mode	Teacher Top-1	Student Top-1	Student Top-5	inf. time (ms)	Batch size
TinyCLIP-ViT-40	HMDB51	No	two-head	ViT-B-32	contrastive	71.6	64.3	91.1	80.9	32
TinyCLIP-ViT-40	HMDB51	Yes	two-head	ViT-B-32	contrastive	71.6	**64.7**	90.3	80.9	32

- **Backbone**: The ActionCLIP backbone that serves as both a spatial and textual encoder.
- **BB Finetune**: Stands for backbone finetune. This indicates whether *active* backbone finetuning is used as described in Section 5.3.
- **Heads**: Number of output heads. Can be the vanilla single-headed approach or our proposed two-head training strategy as described in Section 7.4.
- **Objectives**: By default the model trains on a single ground-truth action objective. The multi-objective indicates that the model is trained as described in Section 6.
- **Loss mode**: Can either be contrastively or using cross-entropy. Both approaches are described in Section 6.5.
- **Inference time (ms)**: Average time it takes to perform inference on a batch of 32 videos sampling 8 frames per video.

B Spatial objectives

In this section we provide additional descriptive analysis of the spatial objectives generated by our prediction framework that were used to perform spatial distillation of the CLIP model. By doing so, we aim to illustrate why certain objectives are more effective when presented in a multi-task problem than others. As we have seen in Section 6.3 in which we presented our pseudo-generation framework, the best performing objectives are the *gender*, *location* and *clothing* objectives. On the other hand, the *holding* and *pose* objectives were determined to be too inconsistent to serve as worthwhile additional tasks during training. For clarity, we deal with the set of successful and unsuccessful set of additional objectives separately and conclude by providing an outline of the criteria that have to be met for an objective to be a worthy addition towards training the ActionCLIP model using a multi-task strategy.

B.1 Successful additional objectives

In Figure 37, we see the label distributions of the *clothing* and *location* objectives. As immediately becomes clear, the wildcard label is the most occurring label. As previously described, this is the label we assign in two cases: if the "a person" prompt is preferred over all other spatial prompts such as 'a person wearing a T-shirt" or if the softmaxed probability of a label falls below the certainty threshold. At first, the fact that roughly half of the dataset does not get a label for the *clothing* objective may seem problematic. However, by masking the loss calculated over the model's prediction of this label we ignore these samples entirely. What is more, the labels are essentially "free" in that they are automatically generated by CLIP using our prediction framework and are not obtained by manually labelling the entire dataset. Our prediction framework may therefore not cover the entire dataset, but it is far more time efficient than manual labelling and even assigning half the dataset with a *clothing* is therefore effectively "free" additional information that is provided to the ActionCLIP model. We also see that the other labels are not equally distributed. For example, a T-shirt is assigned to 14% of all video samples whereas winter clothes only make up of 1.6%. What is also interesting is the fact that the gym outfit is assigned very often. We find that this is because CLIP confidently assigns this label to people wearing a wide range

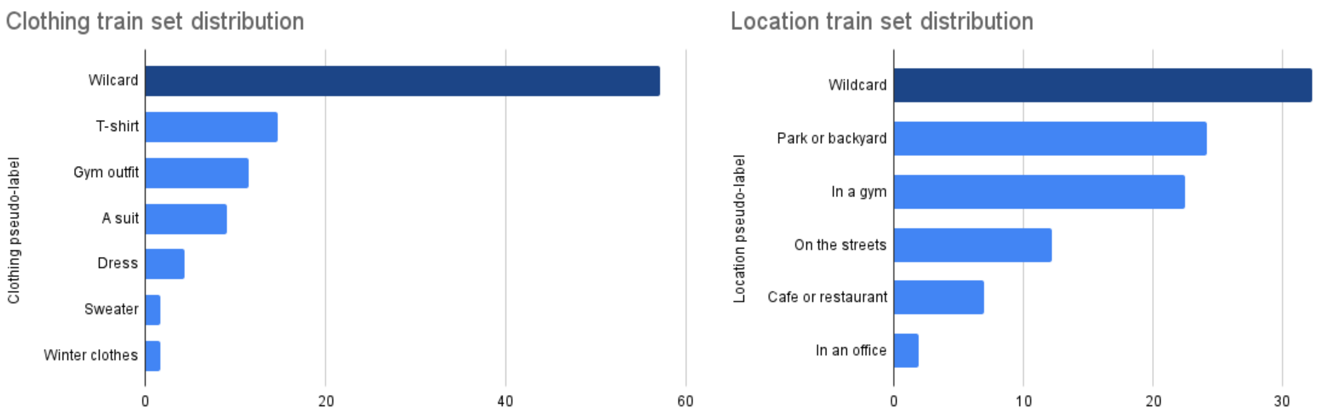


Figure 37: **Label distribution for the *clothing* and *location* spatial objectives on HMDB51.**

of clothes such as swimwear, shorts without a shirt and people wearing small tops. As for the location objective we find that (aside from the wildcard label) the gym and the park or backyard are the locations that are most often assigned to video samples. The label distributions for both the *clothing* and the *location* objectives make sense given the categories present in the HMDB51 dataset. More specifically, the HMDB51 dataset contains many categories related to sports, many of which almost exclusively take place outside. It follows that spatial labels related to outside sports are more abundant than spatial labels that are not. Figure 38 shows the distribution of the *gender* objective. For this objective, we chose

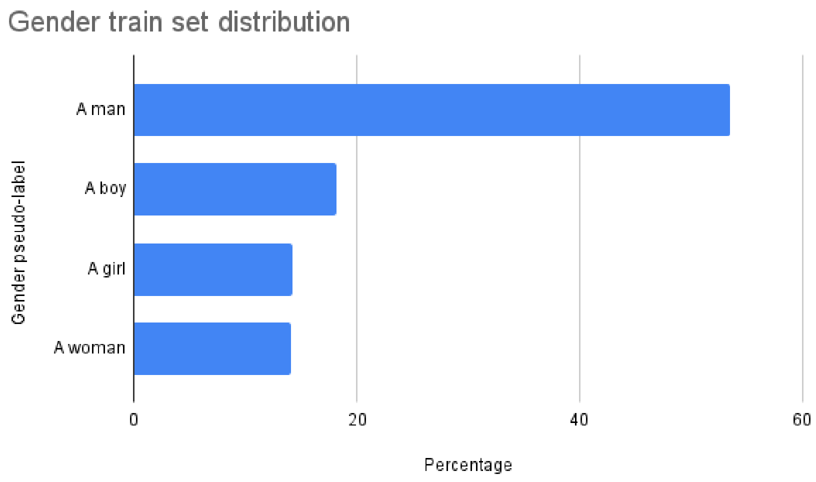


Figure 38: **Label distribution for the *gender* objective on HDMB51.**

not to apply the "a person" wildcard used in the other objectives. The reason why is that one out of the four gender labels must be true for any given clip. For example, it is possible that a person in a video sample from the HMDB51 dataset performs their action in a location different from the ones listed in the set of possible location labels, but it is not possible that they do not fit one of the four gender labels [1]. What makes matters worse is that the wildcard label in the form of the "a person" prompt could steer

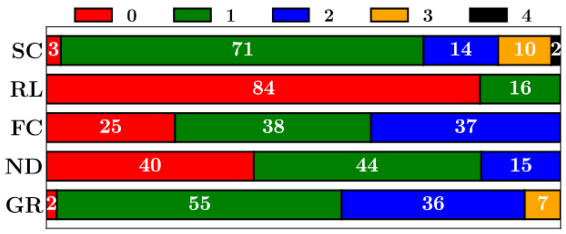


Figure 39: **Distribution of privacy-oriented labels in the PA-HMDB51 dataset.** Figure by Wu et al. "GR" stands for "gender" and that row in the graph shows the distribution between the annotated genders where 0 (red) means that the gender is unidentifiable, 1 (green) means male, 2 (blue) means female and 3 (orange) means different genders coexist in the video sample

the prediction framework away from the correct label because both the "a person" label and a specific gender label *must* be true at the same time. For this objective, we choose to add the additional "a boy" and "a girl" labels to make the classification task of the ActionCLIP model predicting this additional objective slightly harder than simply a binary classification problem. What is interesting about the label distribution of the *gender* objective is that the male gender (i.e. either a man or a boy) is assigned to people in the dataset in more than 71.6% of all cases. For this, there are two possible explanations: either CLIP is biased towards classifying people as men or the HMDB51 dataset contains more videos with men than women. In a work by Wu et al. on training privacy-preserving AR models, a privacy-annotated (PA) HMDB51 dataset is presented called PA-HMDB51. In this dataset, the gender of a person is among the set of 5 characteristics that are annotated for each video sample. As can be seen in the "GR" (gender) bar in Figure 39, a man (green) is present in 55% of all video samples across the HMDB51 dataset as opposed to 36% for the female counterpart (blue). We therefore conclude that gender is not equally represented in

[1]In stating this, the authors acknowledge that people can identify as neither male nor female, but this makes the classification task too unpredictable and subjective to provide a consistent supervisory signal

the HMDB51 dataset, and this can in part explain the distribution of the objective in Figure 38. At the same time, Figure 39 shows us that multiple genders coexist in a sample for 7% of the samples. In these cases, the frames that are sampled from a video determine the gender that is assigned by our prediction framework. In case two genders coexist in that sample, CLIP will likely choose the gender that is most clearly in view. The samples in which a male is (possibly) present add up to about 62% of the video samples according to the PA-HMDB51 dataset, this is almost 10% less than the distribution found in the data produced by our prediction framework. We therefore suspect that CLIP is also slightly more likely to predict that someone is male rather than female. At the same time, the predicted distribution is not too far off from the PA-HMDB51 ground-truth distribution in which the male gender also dominates the distribution. This means that the gender labels presented by our prediction framework will likely be able to strengthen the supervisory signal sent to the ActionCLIP model.

B.2 Unsuccessful additional objectives

We previously stated that the *holding* and *pose* objectives were not suitable as additional spatial objectives because they did not increase the predictive performance of the ActionCLIP model when presented as additional objectives. In this subsection, we briefly describe the generated data by our prediction framework for both objectives.

Lack of new language-video associations

Figure 40 shows the label distribution of the *holding* objective, which does not look fundamentally different from the previously discussed *clothing*, *location* and *gender* objective distributions. However, in

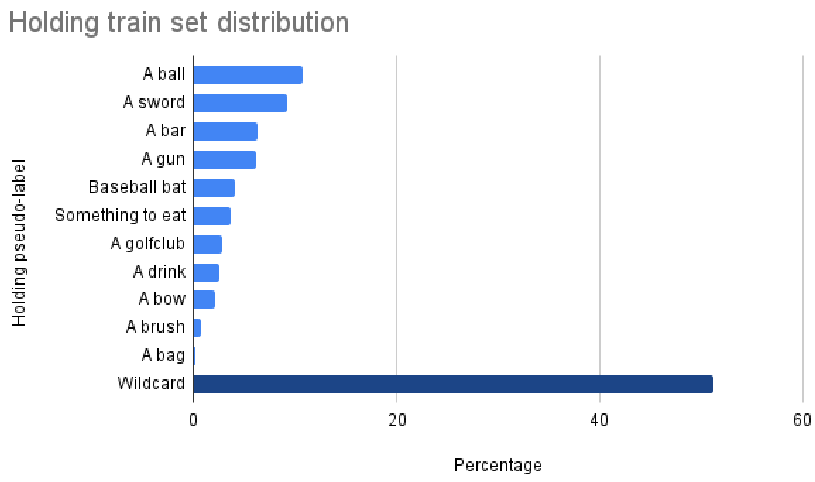


Figure 40: **Label distribution for the *holding* objective on HDMB51.**

analysing the generated data for this objective, it becomes clear why this objective does not increase the predictive performance of the ActionCLIP model. We have previously illustrated that ActionCLIP learns to align language and video. By default, the ActionCLIP takes the ground-truth action label and passes it into a template to describe the video. For example, the ground-truth label "shooting a gun" would be transformed into a descriptive text such as "a person shooting a gun". By ActionCLIP's original implementation, the model then trains by contrastively matching that text's embedding with the video embedding of a person shooting a gun and vice versa. Now, if we were to add the *holding* match task as per Section 6.5, the model would also have to match the "a person holding a gun" to the video embedding and vice versa, an almost identical language-video association to the ground-truth label "a person shooting a gun". We therefore hypothesise that the *holding* objective does not provide enough new information for the ActionCLIP model to improve its predictive performance. The same goes for many other action categories in HMDB51 such as “holding a sword” or “holding a ball”: they do not teach sufficiently new language-video associations to the ActionCLIP model because they strongly overlap with the text associated with the ground truth label. Intuitively, it means that combining this objective with the ground-truth objective is roughly equal to doubling the loss calculated over the ground-truth label for each video sample, which does not make sense.

Improper label distributions

Figure 41 shows the label distribution of the *pose* objective. As can be seen, the “standing upright” label is assigned in more than 50% of the video samples. This label easily dominates the other two labels, “sitting” and “laying down”. As opposed to the dominance of the wildcard label in other objectives, the dominance of the “Standing upright” label is problematic. Whereas the wildcard label is ignored in training the model, this label is not. This means that the model can safely assign the “standing upright”

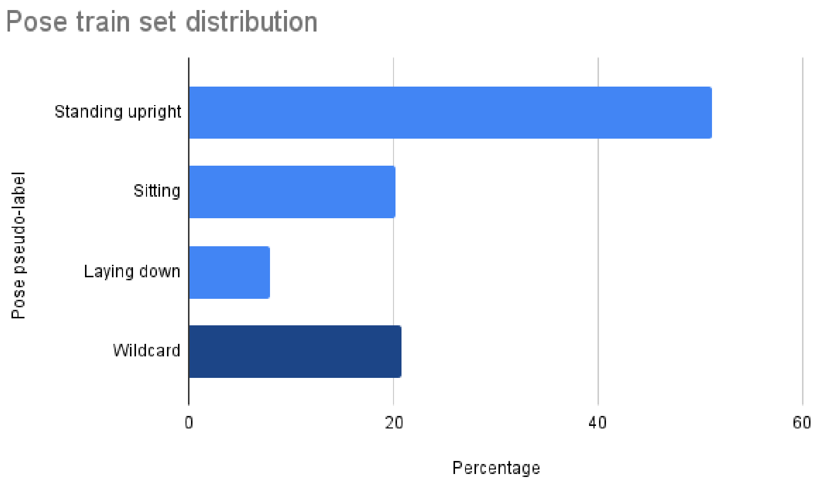


Figure 41: **Label distribution for the *pose* objective on HDMB51.**

label and be correct 50% of the time and its faults get ignored for another 20% of all samples given the percentage of wildcard labels for this objective. We do not suspect a fault in CLIP’s prediction’s on this objective. Rather, we think that this observation is in line with the point made earlier that CLIP contains many action categories that are almost exclusively performed outdoors. It therefore makes sense that most people would be standing upright while performing these actions. Based on these results, we initially also experimented with more granular *pose* labels such as “a person kneeling”, “a person hanging” or “a person laying on their back” but found CLIP to inconsistent to reliably predict these. At the same time, defining the pose of a person as a spatial characteristic proved to be challenging. That is, a person’s clothes, location and gender do not change over a couple of frames but a person’s pose does, especially while performing an action. Whether the *pose* objective can be considered a purely spatial objective is therefore questionable. We therefore suspect that this is the most likely reason why the *holding* objective does not improve the predictive performance of the ActionCLIP model.

B.3 Criteria for selecting suitable spatial objectives

In this section, we have combined the empirical results of ActionCLIP’s predictive performance when trained on different additional objectives with the descriptive analysis of these objectives. By doing so, we have gained more insight into the reason why certain additional objectives work better than others.

We have seen that in order to be a suitable spatial objective, an objective should meet a number of criteria. First and foremost, the spatial generation framework should be consistently able to predict the objective in question. In selecting our first five spatial objectives, this was something we kept in mind by designing clear, discriminate labels that are not overly descriptive so as not to confuse the label generation framework. Secondly, the label distribution over an objective should be not be skewed towards a single label that takes up a disproportionate amount of space in the set of labels. We saw this problem in the *pose* objective for which more than 50% of the labels "standing upright" was assigned and another 20% was ignored because they were assigned the wildcard label. This leaves only 30% of the label distribution for anything other than standing upright. Finally, we have also seen that each objective should add relevant information to the model. We find that this is the hardest criterion to meet, because there is no clear way to determine whether specific information can strengthen the supervisory signal in a multi-task learning strategy. In particular, the *holding* objective seemed promising because in theory it would guide the model to focus on the object that a person is holding and therefore providing potential clues about the action that is being performed. However, in this analysis we have seen that this information strongly overlaps with the information contained in the ground truth label and therefore does not provide meaningful additional information to the ActionCLIP model.
We conclude this section by proposing the three key criteria that an additional objective has to meet in order to provide benefit to the language-enabled AR model:

- 1. The CLIP-based generation framework has to be consistently able to predict the labels for a given objective. Ambiguous and overly complex labels are therefore problematic and should be replaced by discriminate labels that are as short as possible.
- 2. A label distribution of an objective is a good indication of its added value when presented as an additional objective. In particular, objectives containing a single label that dominates the label distribution should be avoided as much as possible.
- 3. Each objective should provide relevant *new* information to the model that does not overlap with existing objectives. In the context of language-enabled AR models, this implies that the text labels for an objective should not overlap with the text labels of other objectives.