Activity Progress Prediction

F. de Boer

Is there progress in video progress prediction methods?

TU Delft

Activity Progress Prediction

by

F. de Boer

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday July 18, 2023 at 10:30 AM.

Student number:	5661439	
Project duration:	November 13, 2022 – July 18, 2023	
Thesis committee:	Dr. J. van Gemert,	TU Delft, supervisor
	Dr. S. L. Pintea,	TU Delft, supervisor
	Dr. J. W. Böhmer,	TU Delft

TU Delft

Preface

This report describes the work done for my thesis for the Master Computer Science at the Delft University of Technology.

First of all, I would like to thank my supervisor Silvia. Our weekly meetings kept me on track, and her feedback always gave me new ideas to work on. Second, I'd like to thank Jan, who always gave helpful ideas at just the right moment. Third, I'd like to thank my friends. Working together made bad days better, and good days great. Finally, I'd like to thank my family for their endless support during this thesis, and for always being there to listen to me.

F. de Boer
Delft, July 2023

Contents

1 Introduction

We perform activity progress prediction throughout our day-to-day lives: *e.g.* during cooking, we predict how long each cooking step takes and when will the food be ready; when travelling we predict how long it will take to reach our destination; etc. Being able to reliably predict activity progress could impact society. For example, in assisted systems for elderly care, progress prediction could be used to detect unintentional actions [4]. Another example is in healthcare, where estimating a surgery duration [16] could improve the management of resources (*e.g.* staff, operation rooms) and decrease patient waiting times.

The objective of activity progress prediction is relating the visual information to the activity progress. Current progress prediction methods report great results on complicated real-world datasets, that are challenging even for humans. In this work, we set forth to answer two main questions: (i) *On what information do progress prediction methods base their predictions?* And (ii) *Is it at all possible to predict progress from visual data only?*

The rest of this report aims to answer these questions and is structured as follows. First, Section 2 contains the scientific article describing the research. Second, the Appendix gives extra information about the topics discussed in the scientific article. Here, Section A discusses the datasets. Section B discusses the methods. Section C gives extra information about the *average-index* baseline. Finally, Section D concludes with a further discussion and more result visualizations.

2

Scientific Article

Is there progress in activity progress prediction methods?

Frans de Boer
Computer Vision Lab,
Delft University of Technology

Silvia L. Pintea
Division of Image Processing (LKEB),
Leiden University Medical Center

Jouke Dijkstra
Division of Image Processing (LKEB),
Leiden University Medical Center

Jan C. van Gemert
Computer Vision Lab,
Delft University of Technology

Abstract

Activity progress prediction methods report great results on complicated and realistic video datasets. This is impressive because the videos in these datasets drastically vary in length and appearance. In this work, we examine the results obtained by existing progress prediction methods, to determine what information these methods use to make their predictions. We find that current progress prediction methods do not use the visual information of the videos. Instead, they rely on simple frame counting and averaging to obtain their results. Additionally, we design a synthetic dataset for the progress prediction task, and show that the considered methods can make use of the visual information in the videos, when this directly relates to the progress prediction. These results point to a flaw in the benchmarks used in the context of progress prediction.

1. Introduction

Activity progress prediction is vital to our day-to-day lives: *e.g.* during cooking, we predict how fast will the food be ready; in healthcare, estimating how long a surgery will take allows for better resource allocation and shorter waiting times. Here, we define activity progress prediction as the task of predicting the percentage of completion of an activity. For our purpose, each video contains a single activity, which covers the complete duration of the video and may consist of multiple phases. However, we assume there are no phase annotations available, as is generally the case in real-world scenarios. The main challenge for activity progress prediction is extracting meaningful information from the visual data, that relates to the specific phases of the activity and, thus, enables predicting the progress.

To address this challenge, current methods rely on convolutional embeddings, such as *VGG-16* [23], *ResNet152* [9], *YOLOv2* [22], or *I3D* [4] to extract visual information. Furthermore, to remember information over time, current progress prediction methods [3, 26] rely on memory blocks and recurrent connections [12]. While these recurrent connections are useful for keeping track of the activity progress over time, they also allow for a trivial solution: *i.e.* counting video frames and predicting the average progress. Here, we aim to analyze if such undesirable learning strategies are employed by the current progress prediction methods.

To this end, we consider the state-of-the-art progress prediction methods [3, 17, 26], as well as two more simple learning-based methods: a $2D$-only *ResNet*, and a *ResNet* model augmented with recurrent connections. We evaluate all these learning methods across three video datasets used for progress prediction: *UCF101-24* [24], *Breakfast* [15, 16], and *Cholec80* [25]. Additionally, we compare the learning-based methods with naive non-learning methods. Firstly, we examine the learning methods when they are presented with the full video sequences during training, and thus making frame counting possible. Secondly, we inspect the results when frame counting is made impossible by training the networks on subsampled video segments. In the latter case, the networks cannot count frames, and therefore they have to rely on visual information for activity progress prediction. If the methods should fail to extract useful information from the visual data, they would perform on par with non-learning methods based on dataset statistics. Finally, we design a synthetic progress prediction dataset, *Progress-bar*, on which the visual information is directly related to the progress. And we investigate the learning methods, when trained on this synthetic data. We expect all learning methods should be able to make use of the visual information.

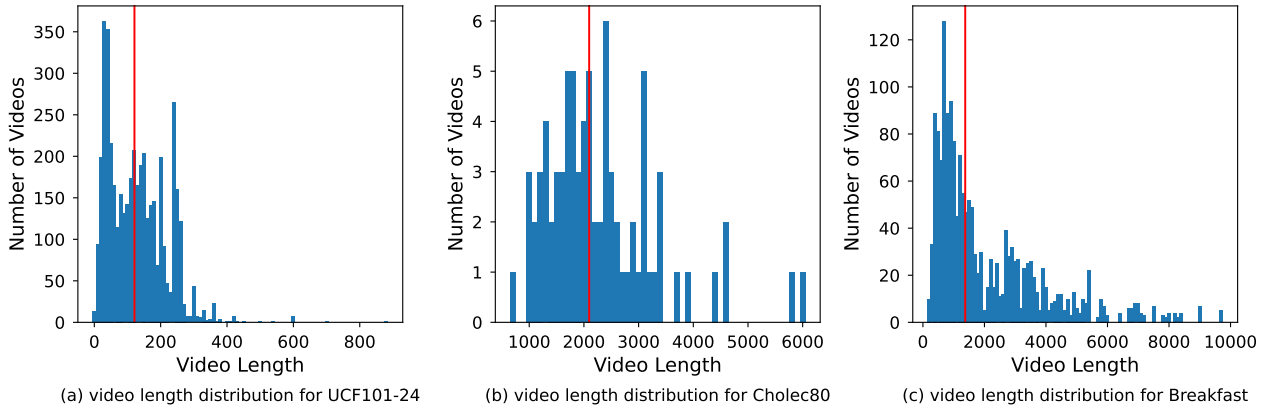


Figure 1. Length distributions for *UCF101-24*, *Cholec80*, and *Breakfast*. *UCF101-24* are grouped into bins of size 10, for *Cholec80* and *Breakfast* the bins are of size 100. Most notable is the long-tail distribution of the video lenghts in the *Breakfast* dataset, which makes progress prediction difficult. The vertical red line depicts the mean of each dataset.

1.1. Difficulties in current progress prediction

Progress prediction methods [3, 17, 26] report impressive results on complicated and realistic datasets such as *UCF101-24* [24], *Breakfast* [15, 16], and *Cholec80* [25]. The appearance and activity length can drastically vary between videos in these datasets, as shown Fig. 1. Also, *UCF101-24* and *Breakfast* follow a long-tail distribution, with few videos containing long activities. Moreover, some of the activities in these datasets do not have a clearly defined endpoint: *e.g.* 'skiing', 'walking the dog', etc. Predicting progress on these activities would be difficult even for a human. Therefore, we arrive at two main questions we aim to address here: (i) *On what information do progress prediction methods base their predictions?* And (ii) *Is it at all possible to predict progress from visual data only?*

2. Related work

Activity progress Prediction. The task of progress prediction was formally introduced in [3]. Because the progress of an activity is an easy-to-obtain self-supervision signal, it is often used as an auxiliary signal in a multi-task prediction problem, as in [13] to improve the performance of spatio-temporal action localisation networks. Progress prediction is also used as a pretext task for phase detection [18], or to create embeddings to perform unsupervised learning of action classes [17, 28]. The progress prediction problem can also be modelled as a classification problem, choosing from n bins each of size $1/n$ as is done in [7]. Based on the literature surveyed, of works done on progress prediction, only [3, 21] have progress prediction as their primary task. This work is also on the topic of progress prediction, but we do not propose our own progress prediction method. Instead, we consider the methods from [3, 17] in our analysis and show how their performance can drastically vary when changing how the training data is presented to the model.

Remaining Duration. A topic closely related to progress prediction is Remaining Duration (RD) prediction. While the progress prediction task models the course of the activity as a percentage value in $[0, 100\%]$, RD prediction models it as a remaining time t in minutes or seconds. Previous work that researches the RD problem often does this in a surgical setting [1, 20, 26, 30] and thus refers to it as the Remaining Surgery Duration (RSD) problem. Early methods work by pretraining a *ResNet-152* model to predict either the surgical phase [1] or the surgery progress [26], and then using the frame embeddings created from the ResNet-152 model in an LSTM block to perform RSD prediction. Building on top of this is the observation in [20] that predicting extra information such as surgeon skill, may be beneficial to do RSD prediction. Finally, RSD can also be modelled in a way closer to progress. By dividing all RSD values by the highest possible RSD, the RSD can be predicted as a value between 0 and 1 [29]. Unlike these methods that model the passage of time as a decreasing remaining duration, we model it as an increasing progress value. We use *RSDNet* [26] in our analysis, as it performs both RSD and progress prediction.

Phase prediction. For a linear activity, an activity in which the phases always follow the same order and are not repeated, the current phase gives a good approximation of the progress. Previous work jointly performs phase-based progress prediction and surgical gesture recognition [27]. Other methods try to improve the phase prediction by jointly predicting the phase and the surgery tools [25], or by pretraining a CNN to predict RSD and using the embeddings in an LSTM to predict the surgical phase online [31]. More recent work applies transformers to perform surgical

phase recognition [14, 19]. In this work, we do not consider phase-prediction methods as they are an inaccurate proxy for progress. Furthermore, when activities are non-linear, phase prediction is no longer a good indicator of activity progress. Knowing which phase is happening may be useful as an extra signal, however we do not consider this, as it requires additional annotations.

Activity Completion. The progress for each frame can be calculated using linear interpolation if the current activity time, t, the starting activity time, t_{start}, and the ending activity time, t_{end}, are available. Early work on this topic only predicts if an activity has been completed or not using an SVM [5]. Follow-up work of Heidarivincheh *et al*. [10] uses a CNN-LSTM architecture to predict the exact frame at which the activity is completed, *i.e*. the activity completion moment. The detection of the activity completion moment is done in a supervised setting [10], where the exact frame at which the activity ends is annotated. Alternatively, activity completion can be done in a weakly supervised setting where the only available annotation is if the activity has been completed or not [11]. Although related to progress prediction, activity completion only aims at predicting the completion moment. In contrast, we aim at predicting the more fine-grained targets of activity progression at every frame.

3. Video progress prediction

We formulate video progress prediction as the task of predicting a progress value $p_n^i \in [0, 100]\%$ at frame i in a video indexed by n, where

$$p_n^i = \frac{i}{l_n}, \tag{1}$$

l_n is the total number of frames for video n. Each video consists of a single activity which starts at frame 0 and ends at frame l_n. The activity may consist of multiple phases, but we do not use any phase annotation.

We predict progress percentages at every frame in the test videos. During training, the videos can be presented to the methods in two different ways: *full-videos* and *video-sequences*. We start by using complete videos during training – *full-videos*, where each video frame represents a data sample. Subsequently, we make the problem more realistic by applying two sampling augmentations, as done in [3]: (a) for every video, we sample a segment by randomly selecting a start and end point; (b) we randomly subsample every such segment to vary its speed. We call the video sampling strategy using (a)-(b): *video-segments*. On *video-segments* the methods cannot rely on frame indices, and needs to process the visual information for predicting progress.

3.1. Progress prediction methods

We consider 3 progress prediction methods from previous work: *ProgressNet* [3], *RSDNet* [26], and *UTE* [17]. We select these methods as they are the only methods in the surveyed literature that report results on the progress prediction task. Furthermore, these methods are the only methods in surveyed literature that do not require additional annotations, such as body joints [21].

ProgressNet [3]: A spatio-temporal network which uses a VGG-16 [23] backbone to embed video frames and extracts further features using spatial pyramid pooling (SPP) [8] and region of interest (ROI) pooling [6]. Additionally, the model uses 2 LSTM layers to incorporate temporal information. Becattini *et al*. also introduce a Boundary Observant (BO) loss. This loss enables the network to be more accurate around areas of phase transitions. In our work, we do not use the BO loss because it requires annotating the phase boundaries. *ProgressNet* uses ROI pooling and requires per-frame bounding box annotations. We use the complete frame as the bounding box on datasets where we do not have bounding box annotations.

RSDNet [26]: It uses a ResNet-152 [9] backbone, followed by an LSTM layer with 512 nodes, and two additional single-node linear layers to jointly predict RSD and video progress. The trained ResNet model creates embeddings from all the frames, which are concatenated with the elapsed time in minutes. *RSDNet* jointly trains on RSD and progress prediction but evaluates only on RSD prediction. Here, we evaluate only the progress prediction head and train with both the RSD and progress loss.

UTE [17]: This is a simple 3-layer MLP (Multilayer Perceptron) which takes as input features extracted from RGB video frames such as dense trajectories [30] or I3D network embeddings [4]. Both dense trajectories and I3D embed frames over a sliding window which encodes temporal information into the features. This gives the *UTE* network access to temporal information. Here, we use $3D$ convolutional embeddings from the I3D backbone of dimension 1024 and an embedding window of size 16 on all datasets. We use precisely the same network design as in [17].

3.2. Learning based baselines

Next to the published methods above, specifically designed for progress prediction, we also consider two more baselines. The first is a spatial only *ResNet-2D* model, and the second is a spatio-temporal *ResNet-LSTM* model. We use *ResNet-LSTM* as it is a progress-only variation of *RSDNet*. Furthermore, the $2D$ variant *ResNet-2D* can give us insights into the spatial-only information contained in the datasets, for progress prediction. We do not consider other architectures, such as a Video Transformer [2], because they do not share the same architecture structure as

the progress prediction methods we consider in Section 3.1, so they would not display similar behaviors during training.

ResNet-2D. A spatial $2D$ *ResNet* [9] architecture that can only make use of $2D$ visual data present in individual video frames, without access to any temporal information. The last layer of the *ResNet* predicts the progress at each frame via a linear layer, followed by a *sigmoid* activation.

ResNet-LSTM. Additionally, we extend the above *ResNet-2D* with an LSTM block with 512 nodes, and a final progress-prediction linear layer using a *sigmoid* activation. The LSTM block adds temporal information, which allows us to test the added value of the memory blocks for activity progress prediction.

3.3. Naive baselines

Next to the learning-based baselines, we consider a set of naive non-learning baselines. These non-learning baselines represent upper-bounds on the errors we expect the learning-based methods to make.

Static-0.5. This is the most obvious non-learning baseline, which always predicts 50% completion at every frame. This is the best guess without any prior information.

Random. Additionally, we consider a *random* baseline that predicts a random value in $[0, 100]\%$ at every frame. This represents the worst progress prediction a model can make, indicating that it failed to learn anything.

Average-index. Finally, we consider a non-learning baseline which computes training-set statistics. It makes a per-frame average progress prediction. For frame i in video n this baseline predicts a progress value equal to the average training-progress at frame i of all training videos indexed by $m \in \{1, ..., N_i\}$:

$$\hat{p}_n^i = \frac{1}{N_i} \sum_{m=1}^{N_i} p_m^i, \tag{2}$$

where N_i is the count of all the training videos with a length of at least i frames.

4. Empirical analysis

4.1. Datasets description

Each of the considered progress prediction methods evaluates on a different dataset: *RSDNet* on *Cholec80* [25], *ProgressNet* on *UCF101-24* [24], and *UTE* on *Breakfast* [15, 16]. To analyze these methods, we use all 3 datasets for all methods.

***Cholec80* [25]**: Consists of 80 videos of endoscopic cholecystectomy surgery. Note that [26] uses an extended version of this dataset, *Cholec120*, containing 40 additional surgery videos. However, *Cholec120* is not publicly available, so we used *Cholec80* to report our results. We randomly create four folds of the data, and follow the same train/test dataset split sizes as in [26]. This dataset has limited visual variability both across training and test splits. Moreover, the presence of different medical tools in the frames informs of the different surgery phases, which could aid the progress prediction task.

***UCF101-24* [24]:** Consists of a subset of *UCF101* containing 24 activities, each provided with a spatio-temporal action tube annotation.[1] Becattini *et al*. [3] split the dataset into 2 categories: *telic* and *atelic* activities. *Telic* activities are those with a clear endpoint, such as 'cliff diving', while *atelic* activities, such as 'walking the dog', do not have a clearly defined endpoint. Predicting progress for *atelic* activities is more difficult than for *telic* ones. The original implementation first trains on *telic* activities, and then finetunes on all activities. We did not observe a difference when using this training procedure, and instead train all methods on the full dataset.

***Breakfast* [15, 16]:** Contains 10 cooking activities: *e.g*. 'making coffee', 'baking pancakes', or 'making tea', etc., performed by 52 individuals in 18 different kitchens. We use the default splits and train each model across all cooking activities. Because the tasks are performed by different individuals in different kitchens, the video appearance varies even within the same task, making this dataset extra challenging for progress prediction.

UCF101-24 contains training videos of up to 599 frames, while *Cholec80* and *Breakfast* contain videos with thousands of frames. When training on *full-video* sequences, we could not train the *ProgressNet* model on the original *Cholec80* and *Breakfast* datasets, because of the long videos leading to memory problems. Thus, for the experiments using *full-video* sequences, we use a subsampled version of *Cholec80* from 1 fps to 0.1 fps (the original fps is 25; [26] subsamples this down to 1fps); and we subsample *Breakfast* dataset from 15 fps down to 1 fps. For our experiments on *video-segments* we use the original datasets.

4.2. Experimental setup

For the considered progress prediction methods only the code for *UTE* is published.[2] For the other methods, we follow the papers for implementation details and training procedures. We train *RSDNet* in a 2-step procedure following [26], however for training the LSTM blocks we found that using the Adam optimizer with a learning rate of 10^{-4} and no weight decay, for 30k iterations works best. For *ProgressNet* not all training details are mentioned in the paper, so we use Adam with a learning rate of 10^{-4} for 50k iterations, and we keep the VGG-16 backbone frozen during

[1] Following [3] we use the revised annotations available at https://github.com/gurkirt/corrected-UCF101-Annots

[2] https://github.com/Annusha/unsup_temp_embed

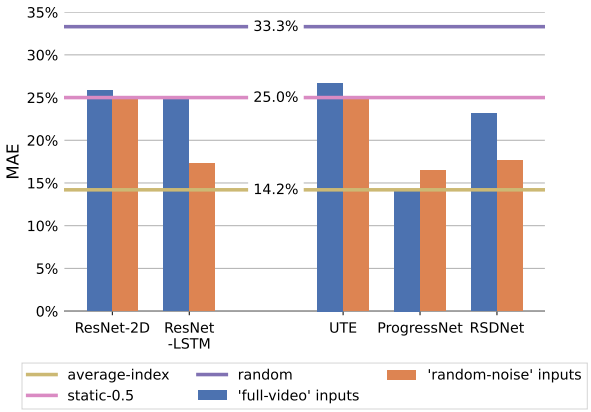


Figure 2. **UCF101-24 training on *full-videos***. MAE in percentages for all learning methods when inputting both *full-video* data and *random-noise*. For all methods except *ProgressNet* inputting *random-noise* performs on par or better than inputting *full-videos*.

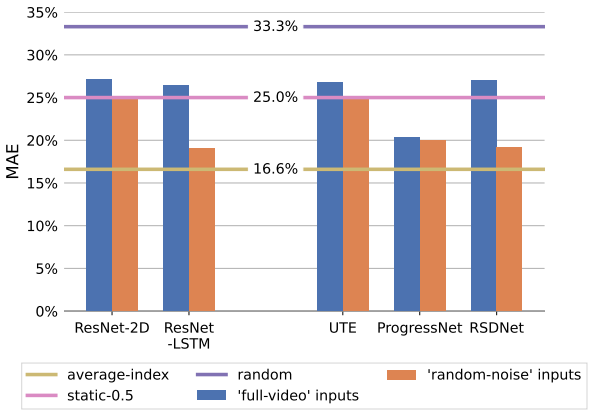


Figure 3. **Breakfast training on *full-videos***. MAE in percentages for all learning methods when inputting *full-video* data and *random-noise*. When using *random-noise* as input to the recurrent methods, they perform on par or better than when inputting *full-videos*, indicating that the methods learn to count video frames.

training. For all experiments we report the MAE (Mean Absolute Error) in percentages.

4.3. *(i) On what information do progress prediction methods base their predictions?*

(i.1) Progress predictions on *full-videos*. Here we want to test what information the learning-based models use to predict progress when trained on *full-videos*. For this we evaluate all learning methods when using *full-video* data as input: *i.e*. either frames or frame embeddings. We compare this with using *random-noise* as input. If the models learn to extract useful appearance information, their MAE scores should be considerably higher than when inputting *random-noise*. Additionally, we compare the learning-based methods with the naive baselines: *static-0.5*, *random*, and *average-index*.

Fig. 2 shows that when training on *full-videos* of *UCF101-24* both the *ResNet-2D* model and *UTE* models perform on par with the *static-0.5* baseline. This is because these spatial-only networks do not have any way of integrating temporal information and they fail to extract progress information from the visual data alone. When provided with *random-noise* as inputs, they always predict 0.5 and score on par with the *static-0.5* baseline. The results are similar for the recurrent models, *ResNet-LSTM* and *RSDNet* who are both close to the *static-0.5* baseline. We observe that the recurrent models overfit on the embedded features and fail to generalise to the test set. When these recurrent networks are provided with *random-noise* they learn to count frames, and thus reach the *average-index* baseline. *ProgressNet* is the only outlier here: when given *full-video* data it performs better than the *average-index* baseline, and when given *random-noise* as inputs, it performs slightly

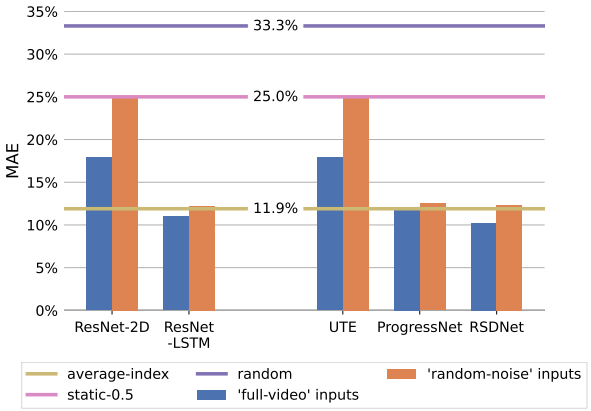


Figure 4. **Cholec80 training on *full-videos***. MAE in percentages for all considered methods when inputting *full-videos* and *random-noise*. On this dataset, all methods are able to learn from the visual data. When provided with *random-noise*, the recurrent methods perform on par with the *average-index* baseline, estimating dataset statistics.

worse. Since the visual information is not sufficiently informative to predict progress, as seen for the *ResNet-LSTM* and *RSDNet*, *ProgressNet* learns to count frames to aid in its progress predictions.

For *Breakfast* in Fig. 3 the results look very similar to those on *UCF101-24*. Both the *ResNet-2D* and *UTE* models cannot learn from visual information alone. *ResNet-LSTM* and *RSDNet* both perform worse than the *static-0.5* baseline, indicating that they are overfitting on the training data. When provided with *random-noise* as input, they

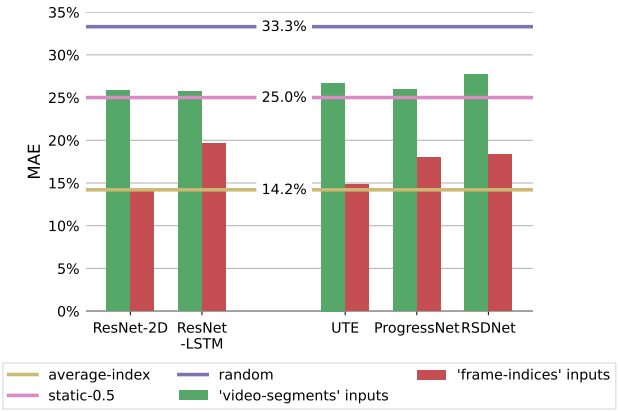


Figure 5. ***UCF101-24* training on *video-segments***. MAE in percentages for all considered methods when inputting both *video-segments* and *frame-indices*. For all methods inputting *frame-indices* is better than inputting *video-segments*. *ResNet-2D* and *UTE* get the biggest boost in performance because they can learn the one-to-one mapping from index to progress during training.

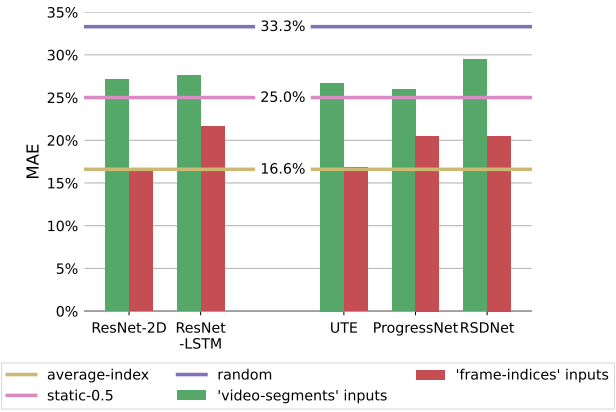


Figure 6. ***Breakfast* training on *video-segments***. MAE in percentages for all considered methods when inputting both *video-segments* and *frame-indices*. All methods perform better when using *frame-indices* as input. Also here *ResNet-2D* and *UTE* obtain the lowest error.

rely on frame counting. *ProgressNet* obtains similar results when given both *full-videos* and *random-noise*, indicating that in both cases it is performing frame counting.

Cholec80 in Fig. 4 is the only dataset where the spatial-only networks *ResNet-2D* and *UTE* perform better than the *static-0.5* baseline. This hints to the visual information present in this dataset being indicative of the activity progress. When inputting *random-noise* the methods again perform on par with the *static-0.5* baseline, as expected. Here, *ResNet-LSTM*, *RSDNet*, and *ProgressNet*, can make use of both the visual information and frame counting and perform slightly better than the *average-index* baseline.

Observation: *The progress prediction methods and the learning baselines can fail to extract useful information from video data. When this happens, memory-based methods will rely on frame counting when presented with full-videos as inputs.*

(i.2): Progress predictions on *video-segments*. We test what information learning methods use when trained to predict progress from *video-segments*. Using *video-segments* should encourage the methods to focus more on the visual information and less so on the temporal position of the frames. We compare this with inputting *frame-indices* – absolute frame indices replicated as images. Ideally, we would expect all methods to solve the progress prediction task by relying on visual information, and therefore surpassing the scores obtained when inputting *frame-indices*. Again, we also compare with the naive baselines: *static-0.5*, *random* and *average-index*.

Fig. 5 shows that when trained on *video-segments* of

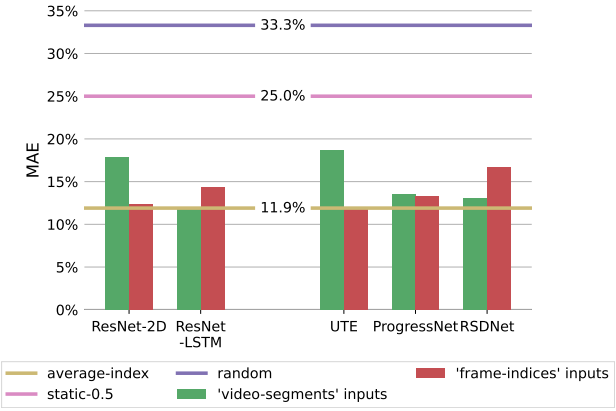


Figure 7. ***Cholec80* training on *video-segments***. MAE in percentages for all considered methods when inputting both *video-segments* and *frame-indices*. For *RSDNet* inputting *frame-indices* is slightly worse. This could be due to suboptimal hyperparameter settings.

UCF101-24 all methods perform on par with the *static-0.5* baseline. Thus, the models cannot learn to predict progress from the visual video data. Interestingly, *ProgressNet* using *full-videos* in Fig. 2 is better than the *average-index* baseline, however, here it fails to learn when trained on *video-segments*. This suggests that on the *full-videos* the network was not relying on the visual information, but just on the temporal progression of frames – frame counts. When provided with *frame-indices* as input, all methods improve. The improvement is most visible for *ResNet-2D* and *UTE*, which do not use recurrent blocks. This is because the non-

recurrent methods can learn the one-to-one mapping from index to progress during training and apply it at test-time.

The results on *Breakfast* in Fig. 6 are similar to those of *UCF101-24* in Fig. 5. None of the networks can extract useful information from the *video-segments*. And, as expected, *ProgressNet* performs worse than when trained on *full-videos*. All methods improve when trained on *frame-indices*. The improvement is again more obvious for *ResNet-2D* and *UTE*.

On *Cholec80* in Fig. 7 we see a similar trend. *ResNet-2D* and *UTE* improve when provided with *frame-indices* as input. For *ResNet-LSTM* and *ProgressNet* the performance is on par with the *average-index* baseline for both *video-sequences* and *frame-indices* as inputs, indicating that on the *Cholec80* these methods can learn progress prediction from visual information, however, they still cannot perform better than the *average-index* baseline. *RSDNet* performs worst when given *frame-indices* as inputs; we hypothesise that this is due suboptimal hyperparameter settings.

Observation: *The progress prediction methods cannot extract sufficient information from visual data for solving the progress prediction, on the currently used datasets. And when not provided with the complete videos, these methods are outperformed by native non-learning baselines relying on data statistics.*

4.4. *(ii) Is it at all possible to predict activity progress from visual data only?*

We observe that existing learning-based methods often fail to extract progress information from visual data alone. Our goal here is to construct a synthetic dataset in such a way that the learning-based methods perform optimal using visual information, and outperform the naive baselines.

We construct a synthetic *Progress-bar* dataset, as shown in Fig. 8. The dataset contains a progress bar (similar, for example, to a download bar) that slowly fills up from left to right. Each frame has a resolution of 32×32px. We generate 900 videos for the training split, and 100 for the test split. Each bar has its own rate of progression, but there is a variance per notch causing some uncertainty. This is why in the first image the progression appears to be slightly beyond 25%, but because the video may slow down after this section it is actually at 22.2%. Due to the large variance in video length, ranging from 19 to 145 frames, the *average-index* baseline, and thus frame-counting strategies, will give worse results than relying on visual information. Also, because of the different progress rates per video, the learning methods cannot just rely on visual information alone but also have to use memory to perform well on this progress prediction task.

Due to the reduced frame resolution and data complexity of our synthetic dataset, we scale down the *ResNet* backbone, for these experiments. Specifically, to avoid over-

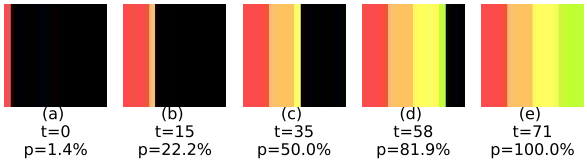


Figure 8. Visualisation of a progress bar from our synthetic *Progress-bar* dataset at timesteps t=0, t=15, t=35, t=58, and t=71. Each coloured section indicates visually a 25% section, but due to variance in the speed, the actual video progress may differ at these points.

fitting, we use *ResNet-18* as a backbone for *ResNet-2D*, *ResNet-LSTM*, and *RSDNet*. *ProgressNet* and *UTE* remain unchanged.

Fig. 9 shows the results of all the learning-based methods when predicting progress from both *full-videos* and *video-segments*. For this dataset the *average-index* baseline has an MAE of 12.9%, which is outperformed by all learning-based methods. *UTE* performs the best out of all the networks, even though it does not have memory. This is because *UTE* uses frame embeddings with a temporal-window size of 16 frames. This temporal-window gives the method information about 7 future frames, which is sufficient on this simple dataset. For the LSTM-based methods inputting *full-videos* still performs slightly better than inputting *video-segments*. At a closer look, this is because the *video-segments* sampling method has a bias towards frames in the middle of a video. The earlier frames are less likely to get sampled, thus the progress prediction methods will have a higher error here.

Observation: *It is feasible for the progress prediction methods to make effective use of the visual data present in the videos and outperform the average-index baseline, when the visual data is a clear indicator of the video progression.*

5. Discussion and limitations of our analysis

Discussion. This paper demonstrates empirically that progress prediction methods may not learn from visual information, and instead rely on frame counting. However, our results also show that some methods can learn from the visual information on the *Cholec80* and generalize well to the test set. Fig. 10, Fig. 11, and Fig. 12 show examples of predictions on the *Cholec80* dataset. Fig. 10 shows the first moment a new medical tool is present in the video, and the progress prediction methods adjust their predictions to this new visual information. Similarly, in Fig. 11 we show the moment the collection bag is present which signals the end of the procedure. For Fig. 12 when no visual land-

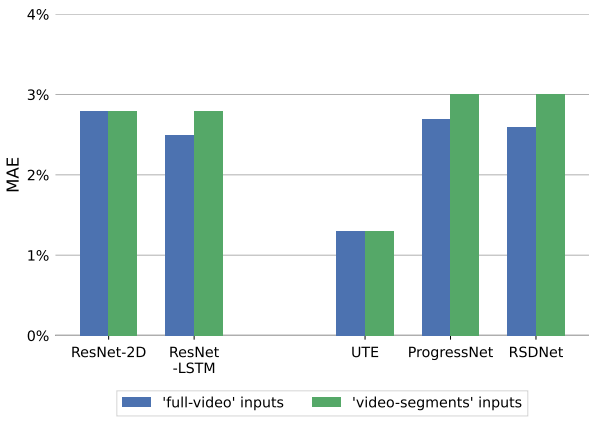


Figure 9. **MAE scores on our synthetic *Progress-bar* dataset, when training on *full-videos* and *video-segments***. The *average-index* baseline has an MAE of 12.9%, while the *static* baseline is at 25% and the *random* baseline at 33.3%. We see that all methods outperform the *average-index* baseline. *UTE* obtains the best result due to its 15-frame temporal window, which allows it to see 7 frames into the future. We conclude that the progress prediction methods are able to learn progress from visual information, if it is clearly present in the videos.

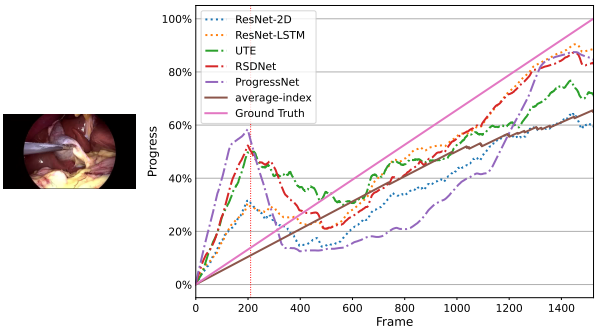


Figure 10. Progress prediction example on Video-04 of *Cholec80* at timestamp t=210. The methods recognize the medical tool, and correct their progress downwards to signal the start of the medical procedure.

mark is introduced, the progress increases linearly. Fig. 13 and Fig. 14 show two example predictions on our synthetic *Progress-bar* dataset. Here, the networks almost perfectly follow the ground truth progression. These results illustrate that for progress prediction is essential to have clearly recognizable visual transition points, that consistently correspond to a certain progress prediction percentage. This is related to the idea of Becattini *et al*. [3] who use phase annotations to increase the loss around the phase boundaries.

Limitations. The first limitation of our research is that we could only find 3 progress prediction methods to ana-

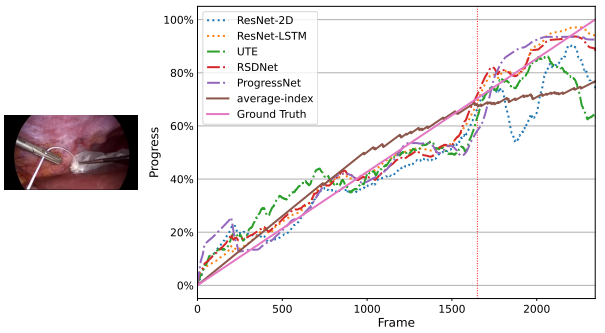


Figure 11. Progress prediction example on Video-05 of *Cholec80* at timestamp t=1650. The methods recognize the collection bag and correct their progress to signal the end of the procedure.

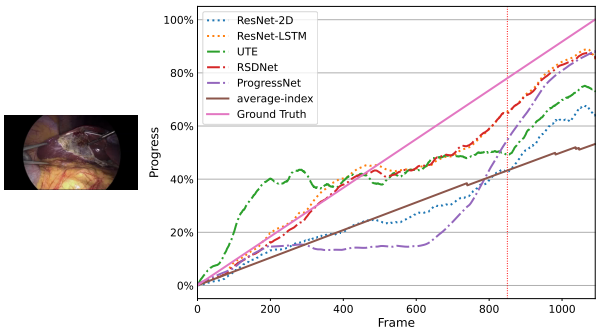


Figure 12. Progress prediction example on Video-12 of *Cholec80* at timestamp t=850. When no new visual landmarks are present the progress increases linearly.

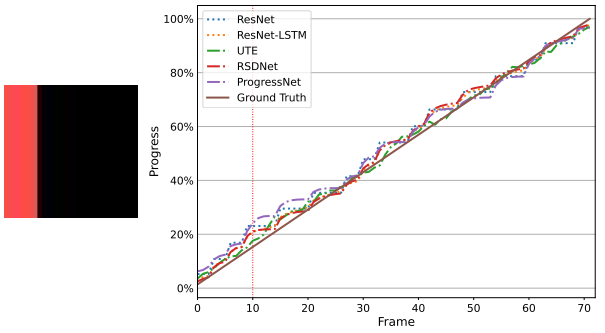


Figure 13. Progress prediction example on Video-00004 of our synthetic *Progress-bar* dataset at timestamp t=10. All methods are able to almost perfectly follow the ground truth.

lyze, on 3 datasets. Additionally, we do not consider here other video-architectures such as a Video Transformer [2], as these are not directly related to the progress prediction methods we analyze. However, we do consider 2D (ResNet) and 3D (I3D) convolutional networks, as well as recurrent

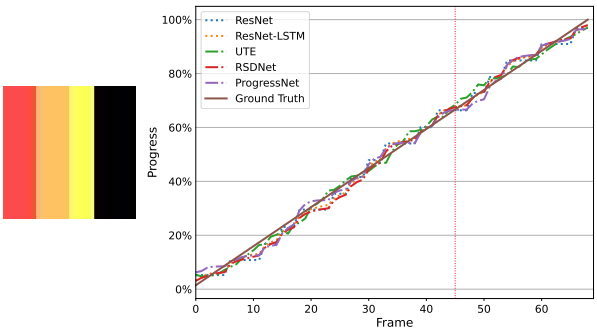


Figure 14. Progress prediction example on Video-00015 of our synthetic *Progress-bar* dataset at timestamp t=25. Also here, the learning methods can almost perfectly follow the ground truth.

networks (with LSTM blocks). Thirdly, we were unable to match the results of *ProgressNet* exactly as reported in [3]: when trained on *video-segments*, the authors report an MSE of 0.052 (MAE of approximately 22.8%), while we obtain an MAE of 25.9%. Nonetheless, the *average-index* outperforms the result reported in [3], which still validates our conclusions. Finally, we observed that on both *UCF101-24* and *Breakfast* the methods have a tendency to overfit. Maybe better strategies to overcome this overfitting phenomenon could improve the results.

6. Conclusion

In this paper, we investigate the behaviour of current progress prediction methods on the currently used benchmark datasets. We show that the progress prediction methods can fail to extract useful information from visual data on these datasets. Moreover, when the methods fail to extract visual information, memory-based methods adopt a frame-counting strategy when presented with *full-video* data as input. Additionally, we evaluate all the methods on a synthetic dataset we specifically designed for the progress prediction task. On our synthetic dataset the results show that all the methods can make use of the visual information and outperform the native, non-learning baselines. We conclude that in its current form the task of progress prediction is ill-posed. The learning methods tend to fail to extract useful information from the visual data and instead rely purely on frame counting.

References

[1] Ivan Aksamentov, Andru Putra Twinanda, Didier Mutter, Jacques Marescaux, and Nicolas Padoy. Deep neural networks predict remaining surgery duration from cholecystectomy videos. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2017. 2

[2] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer, 2021. 3, 8

[3] Federico Becattini, Tiberio Uricchio, Lorenzo Seidenari, Lamberto Ballan, and Alberto Del Bimbo. Am i done? predicting action progress in videos, 2017. 1, 2, 3, 4, 8, 9

[4] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset, 2018. 1, 3

[5] Majid Mirmehdi Farnoosh Heidarivincheh and Dima Damen. Beyond action recognition: Action completion in rgb-d data. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 142.1–142.11. BMVA Press, September 2016. 3

[6] Ross Girshick. Fast r-cnn, 2015. 3

[7] Tengda Han, Jue Wang, Anoop Cherian, and Stephen Gould. Human action forecasting by learning task grammars, 2017. 2

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Computer Vision – ECCV 2014*, pages 346–361. Springer International Publishing, 2014. 3

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 1, 3, 4

[10] Farnoosh Heidarivincheh, Majid Mirmehdi, and Dima Damen. Action completion: A temporal model for moment detection, 2018. 3

[11] Farnoosh Heidarivincheh, Majid Mirmehdi, and Dima Damen. Weakly-supervised completion moment detection using temporal attention. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 1188–1196, 2019. 3

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 1

[13] Bo Hu, Jianfei Cai, Tat-Jen Cham, and Junsong Yuan. Progress regression rnn for online spatial-temporal action localization in unconstrained videos, 2019. 2

[14] Muhammad Abdullah Jamal and Omid Mohareri. Surgmae: Masked autoencoders for long surgical video analysis, 2023. 3

[15] Hilde Kuehne, A. B. Arslan, and T. Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proceedings of Computer Vision and Pattern Recognition Conference (CVPR)*, 2014. 1, 2, 4

[16] Hilde Kuehne, Juergen Gall, and Thomas Serre. An end-to-end generative framework for video segmentation and recognition. In *Proc. IEEE Winter Applications of Computer Vision Conference (WACV 16)*, Lake Placid, Mar 2016. 1, 2, 4

[17] Anna Kukleva, Hilde Kuehne, Fadime Sener, and Juergen Gall. Unsupervised learning of action classes with continuous temporal embedding, 2019. 1, 2, 3

[18] Xinyu Li, Yanyi Zhang, Jianyu Zhang, Yueyang Chen, Shuhong Chen, Yue Gu, Moliang Zhou, Richard A. Farneth, Ivan Marsic, and Randall S. Burd. Progress estimation and phase detection for sequential processes, 2017. 2

[19] Yang Liu, Maxence Boels, Luis C. Garcia-Peraza-Herrera, Tom Vercauteren, Prokar Dasgupta, Alejandro Granados, and Sebastien Ourselin. Lovit: Long video transformer for surgical phase recognition, 2023. 3

[20] Andrés Marafioti, Michel Hayoz, Mathias Gallardo, Pablo Márquez Neila, Sebastian Wolf, Martin Zinkernagel, and Raphael Sznitman. Catanet: Predicting remaining cataract surgery duration, 2021. 2

[21] Davide Pucci, Federico Becattini, and Alberto Del Bimbo. Joint-based action progress prediction. *Sensors*, 23(1), 2023. 2, 3

[22] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger, 2016. 1

[23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 1, 3

[24] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild, 2012. 1, 2, 4

[25] Andru Putra Twinanda, Sherif Shehata, Didier Mutter, Jacques Marescaux, Michel de Mathelin, and Nicolas Padoy. Endonet: A deep architecture for recognition tasks on laparoscopic videos, 2016. 1, 2, 4

[26] Andru Putra Twinanda, Gaurav Yengera, Didier Mutter, Jacques Marescaux, and Nicolas Padoy. RSDNet: Learning to predict remaining surgery duration from laparoscopic videos without manual annotations. *IEEE Transactions on Medical Imaging*, 38(4):1069–1078, apr 2019. 1, 2, 3, 4

[27] Beatrice van Amsterdam, Matthew J. Clarkson, and Danail Stoyanov. Multi-task recurrent neural network for surgical gesture recognition and progress prediction, 2020. 2

[28] Rosaura G. VidalMata, Walter J. Scheirer, Anna Kukleva, David Cox, and Hilde Kuehne. Joint visual-temporal embedding for unsupervised learning of actions in untrimmed sequences, 2020. 2

[29] Bowen Wang, Liangzhi Li, Yuta Nakashima, Ryo Kawasaki, and Hajime Nagahara. Real-time estimation of the remaining surgery duration for cataract surgery using deep convolutional neural networks and long short-term memory, 2023. 2

[30] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *2013 IEEE International Conference on Computer Vision*, pages 3551–3558, 2013. 2, 3

[31] Gaurav Yengera, Didier Mutter, Jacques Marescaux, and Nicolas Padoy. Less is more: Surgical phase recognition with less annotations through self-supervised pre-training of cnn-lstm networks, 2018. 2

A

Datasets

A.1. UCF101-24

UCF101 [14] is a realistic human activity dataset collected from user-uploaded YouTube videos. The dataset consists of many actors performing various activities such as 'Cliff Diving', 'Push ups', etc. In total, the dataset has 13k videos and 101 different activities. Due to the diverse sources of the videos, they all vary in visual information, such as camera motion, subject appearance, and background. Each video in the dataset is downsampled to a resolution of 320×240 pixels and a framerate of 25 fps. This dataset has two problems for the task of progress prediction: **(1)** an activity may start and end at any point in the video, and **(2)** there may be multiple activities happening at the same time. As a solution to this *UCF101-24* was introduced.

UCF101-24 [14] is a subset of *UCF101* of just 24 activity categories. Each video in this subset is annotated with activity tubes, which can be seen as spatiotemporal bounding boxes. Each frame in the tube is annotated with a bounding box, and each bounding box is linked to the one before and after it. This encloses the subject throughout the video. There can be multiple activity tubes per video, and even multiple activity tubes going through the same frame. By using the activity tubes instead of the videos we can predict progress on multiple activities throughout the video, even when they are happening concurrently.

UCF101-24 consists of 3200 videos, of which 2290 are in the train split and 910 are in the test split. In total, there are 4456 annotated activity tubes, of which 3143 are in the train split and 1315 are in the test split. Tab. A.1(a) shows the number of activity tubes and the average activity length for each activity category. Fig. A.1 shows 10 videos per activity category, each at approximately 50% completion. The dataset has many activities, each with its own visual information, which makes the dataset challenging for the task of progress prediction.

Becattini *et al.* [2] split *UCF101-24* into two categories, *telic* activities and *atelic* activities. *Telic* activities have a clearly defined goal, such as 'Cliff Diving', this activity ends the moment the actor hits the water. *Atelic* activities do not have a clearly defined goal, such as 'Walking With Dog'. Prediction progress on *atelic* activities is more difficult, as there is no difference between 'Walking With Dog' at the start of the video and at the end. Since over half of the activities in *UCF101-24* are *atelic* the dataset can be challenging for the task of progress prediction.

A.2. Breakfast

Breakfast [10, 11] is a cooking activity dataset created with the goal of capturing real-world scenarios. The dataset consists of 1712 videos which are all downsampled to a resolution

Activity Class	Number of action tubes	Average activity length
Basketball	171	38
BasketballDunk	132	38
Biking	202	181
CliffDiving	143	63
CricketBowling	142	39
Diving	149	114
Fencing	240	123
FloorGymnastics	125	151
GolfSwing	149	119
HorseRiding	172	187
IceDancing	322	226
LongJump	130	130
PoleVault	164	141
RopeClimbing	119	164
SalsaSpin	616	86
SkateBoarding	120	136
Skiing	135	216
Skijet	105	217
SoccerJuggling	149	276
Surfing	196	114
TennisSwing	260	40
TrampolineJumping	254	143
VolleyballSpiking	124	31
WalkingWithDog	136	198

(a) Activity classes in *UCF101-24*

Activity Class	Number of videos	Average activity length
coffee	167	596
cereals	184	706
tea	184	719
milk	187	951
juice	162	1493
sandwich	169	1547
scrambledegg	166	3122
friedegg	173	3126
salat	163	3431
pancake	157	5979

(b) Activity classes in *Breakfast*.

Table A.1: (a) Activity classes in *UCF101-24* and the number of annotated action tubes per activity class. The *telic* activities are in **bold**. The large number of activities, each with its own visual information, and the inclusion of *atelic* activities makes predicting progress on this dataset challenging. (b) Activity classes in *Breakfast* and the number of videos per activity class. The activities are well-balanced. However, the large difference in average activity length makes predicting progress on this dataset challenging.

of 320×240 pixels and a framerate of 15 fps. The dataset is split into 4 groups for training and testing, these are in Tab. A.2. The 1712 videos are split across 10 different cooking activities, such as 'making coffee' and 'baking pancakes'. The activities are chosen in such a way that some of them share a lot of phases, while others are completely different. For example 'scrambled egg' and 'fried egg' are very similar, and so are 'juice' and 'milk'. Other activities, such as 'coffee' and 'pancake' share almost no phases between them. This is one of the factors which makes this a challenging dataset for progress prediction.

S_1	S_2	S_3	S_4
P_{01} - P_{15}	P_{16} - P_{28}	P_{29} - P_{41}	P_{42} - P_{54}

Table A.2: Breakfast dataset splits for testing where the actor IDs are given as $P_i, i \in \{01, .., 54\}$. The remaining actor IDs are used for training.

To further increase the difficulty of the dataset several additional steps were taken. The activities are performed by 52 actors so the videos have different subjects. Because every person cooks slightly differently, this adds extra variation to each activity. The activities are performed in 18 different kitchens, this changes the background information and tools used for the activity. Finally, each activity is recorded by 3 to 5 different cameras, so there are different viewpoints and different types of motion relative to the camera. All of this means that the learners cannot simply remember what certain objects and activities look like. Instead, they have to understand which task is happening and which phase is happening, and combine this to make a progress prediction.

Tab. A.1(b) shows the number of videos and the video length for each activity category. The number of videos per activity is quite well-balanced. The average video length differs a lot

per activity category, making pancakes takes on average 10 times longer than making coffee. In other words, making pancakes progresses 10 times slower than making coffee. This large variety in video lengths disadvantages counting-based methods because they cannot adjust their progress prediction based on what activity is happening. Fig. A.2 shows 10 videos per activity category, each at approximately 50% completion. There is a large variety of visual information, a lot of which is background information that learning methods cannot use to make predictions. For example, the first column shows the same kitchen multiple times but each time a different recipe is being made. These factors make the dataset challenging for the task of progress prediction.

A.3. Cholec80 & Cholec120

Cholec80 [15] is a dataset consisting of 80 endoscopic cholecystectomy surgery videos – gallbladder removal surgeries which are recorded from inside the body. Each video is captured with a framerate of 25 fps. For each video, the surgical phase and tool presence are labelled by a senior surgeon. While we do not use these annotations in our research, for surgery, where the phases often proceed in a linear fashion, this information could be useful for progress prediction. Fig. A.4 shows frames from 6 videos, each at approximately 50% completion. Each video in the dataset shares similar visual information. Additionally, different medical tools used at each phase give a clear visual indication of the current phase. Because the visual information is such an informative signal of the current progress, this dataset is easier for the task of progress prediction.

Unlike *UCF101-24* and *Breakfast*, *Cholec80* does not have standardised train/test/eval splits. The lack of standardised data splits can cause problems when comparing results from various papers [5]. Because we compare to [16], we follow their data strategy as close as possible.

Cholec120 [1] adds an additional 40 videos to the *Cholec80* dataset. Very little information is given about these 40 extra videos, and they are not publicly available. We can see from the duration distributions in Fig. A.3 that the 40 additional videos follow roughly the same distribution as *Cholec80*. Most of them are in the range of 30 to 40 minutes.

Figure A.1: 10 videos per activity category for *UCF101-24*, each at approximately 50% completion. The large difference in visual information between the videos makes this a challenging dataset for the progress prediction task.

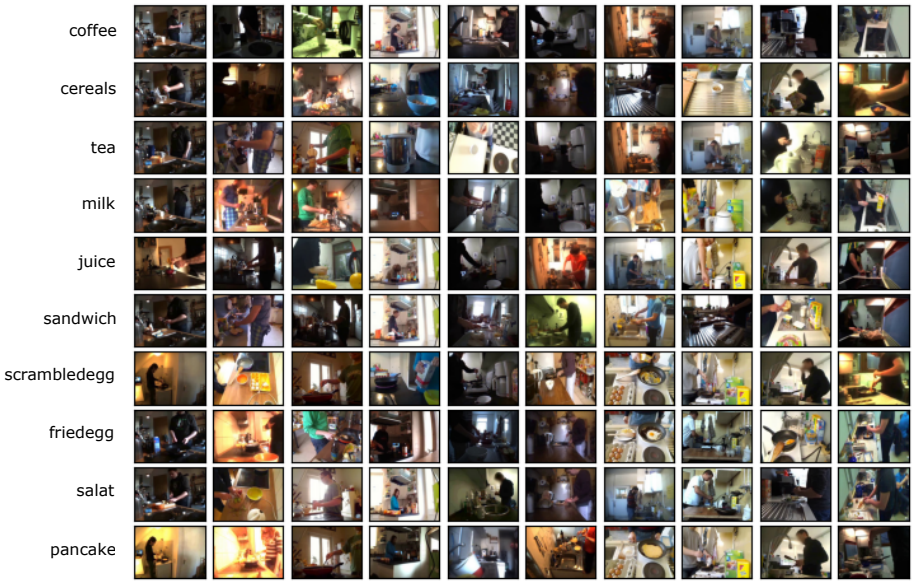


Figure A.2: 10 videos per activity category for *Breakfast*, each at approximately 50% completion. The large difference in visual information between the videos makes this a challenging dataset for the progress prediction task.

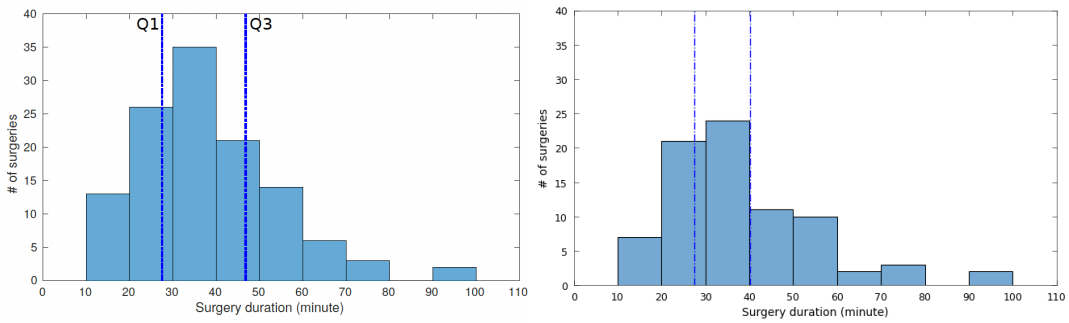


Figure A.3: The surgery durations for *Cholec120* [1] and *Cholec80* [15]. *Cholec120* figure taken from the original paper [16]. Most of the 40 additional videos added in *Cholec120* are in the range of 30 to 40 minutes, and they roughly follow the same distribution as *Cholec80*.

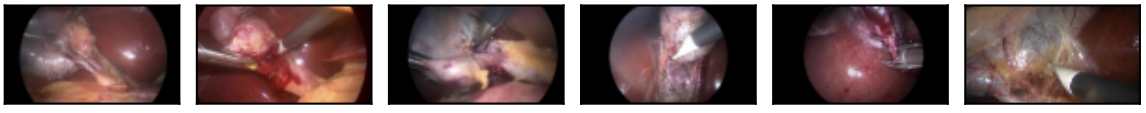


Figure A.4: Visualisations for 6 videos from the *Cholec80* dataset, each at approximately 50% completion. All videos share similar visual information, making it easier to learn progress on this dataset compared to *UCF101-24* and *Breakfast*.

B

Methods

B.1. ProgressNet

ProgressNet [2] is the first network designed specifically for the task of progress prediction. The *ProgressNet* architecture can be seen in Fig. B.1. *ProgressNet* consists of a *VGG-16* [6] backbone which extracts visual features from frames. From these features the network creates bounding boxes, which are linked through time via a *box linking* algorithm to create activity tubes. The main contribution of *ProgressNet* is the progress prediction head. This further extracts features using spatial pyramid pooling (SPP) [8] and region of interest (ROI) pooling [6]. These features are concatenated and forwarded through a linear layer, two LSTM layers with 64 and 32 hidden nodes, and a final linear layer to predict progress. The network predicts both bounding boxes and progress in a fully online manner. Not much information is given about the training procedure used for *ProgressNet*. For our implementation, we followed the paper as close as possible.

ProgressNet is evaluated on *UCF101-24* [14] and *J-HMDB* [9], here we focus on the results for *UCF101-24*. The loss for *ProgressNet* is calculated as the mean squared error (MSE) of predicted progress and actual progress between 0 and 1. Instead, we use the mean absolute error (MAE) of predicted progress and actual progress between 0 and 100. To compare the results we convert *ProgressNets* MSE loss to an approximate MAE loss. The results can be seen in Tab. B.1. We see that with and without the BO loss *ProgressNet* as implemented in [2] gets better results compared to our implementation of *ProgressNet*. The $2D$ only variant of *ProgressNet* is unable to learn from just visual information and performs worse than predicting

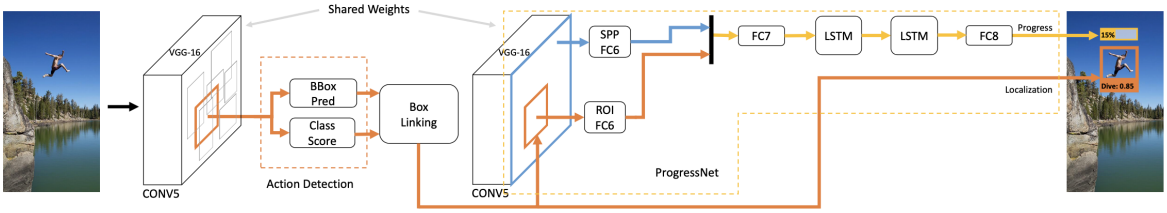


Figure B.1: *ProgressNet* [2] architecture. The network consists of a *VGG-16* [13] backbone which extracts visual features from the frames. From this network first **(1)** bounding boxes are created, which are linked through time via a *box linking* algorithm to create activity tubes. Second, **(2)** Spatial Pyramid Pooling (SPP) [8] and Region of Interest (ROI) pooling [6] layers extract further features. These features are then forwarded through a linear layer, an LSTM layer, and a final linear layer to predict progress. Figure from the original paper [2].

50%. From these results, we are unsure how the original *ProgressNet* implementation is able to get an improved performance. The network is unable to predict progress from visual information alone, yet when given *video-segments* the performance improves. There might be an implementation detail we are missing, but from just following the paper these are the best results we could get.

Progressnet	*ProgressNet BO*	*ProgressNet 2D*	*Ours*
22.8	21.4	28.1	25.9

Table B.1: Results of *ProgressNet* with and without the BO loss as implemented in [2] compared to the results of *ProgressNet* as implemented in this research. *ProgressNet* as implemented in [2] gets an improved score compared to ours.

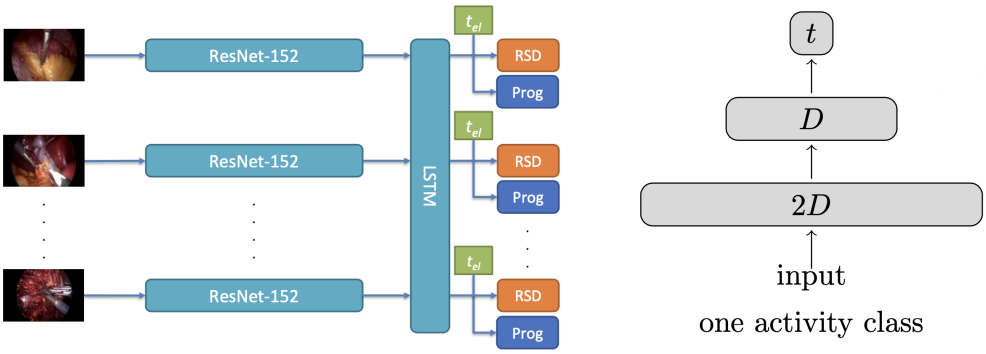
B.2. RSDNet

RSDNet [16] jointly predicts remaining surgery duration (RSD) and progress. The network architecture can be seen in Fig. B.2(a). *RSDNet* predicts RSD and progress for every frame in a sequence and is trained on *full-video* sequences. One problem with training on *full-video* sequences is that if the sequences are too large it can become difficult, if not impossible, to train the network on a GPU. To overcome this problem the network is split into two smaller networks which are trained separately: The *CNN* network and the *LSTM* network. To train the *CNN* network the video sequences are split into individual frames. The *CNN* network, which consists of a modified *ResNet-152* network, is then trained on $2D$ images to predict progress. After this, the *LSTM* network is trained on sequences of frame embeddings created by the *CNN* network

Both networks are trained using the SGD optimizer with a momentum of 0.9 and a smooth L1 loss. First, the *ResNet-152* model is modified by replacing the last layer of the model with a single-node linear layer with a sigmoid activation. The *ResNet-152* backbone is trained with an initial learning rate of 10^{-3}. The learning rate is reduced by a factor of 10 every 20k iterations. Finally, the weight decay is set to 5×10^{-4} and the model is trained for 50k iterations with a batch size of 48. Second, the LSTM network is trained. This network is trained for 30k iterations with an initial learning rate of 10^{-3}. The learning rate is reduced by a factor of 10 every 10k iterations. Finally, the weight decay is set to 10^{-2}.

To train and validate on *Cholec120*, Aksamentov *et al.* create 4 folds, which are optimized using a genetic algorithm. This genetic algorithm optimised the folds such that the mean of the sequence lengths is similar between all the subsets, and the diversity of the lengths is as large as possible within each subset. This ensures that all folds are similar while maximising diversity within the folds. Each fold is then split into 4 segments: $T1$, used to train the CNN, and $T2$, used to train the CNN-LSTM, are both 40 videos. V, the test split, is 10 videos. Finally, E, the validation split, is 30 videos. This split enables training both the *CNN* and *CNN-LSTM* on different videos to avoid overfitting. We follow the same strategy, but since *Cholec80* is missing 40 videos our splits are smaller. $T1$ and $T2$ are both 27 videos, V is 6 videos, and finally E is 20 videos.

Twinanda *et al.* [16] report a Mean Average Error (MAE) for the RSD predictions of 8.1 ± 5.4. We get an MAE on the RSD predictions of 10.5 ± 0.8, which is slightly higher and has a much lower standard deviation. We still think our results are comparable for the following reasons: First, *RSDNet* is trained on *Cholec120*, which has 40 extra videos compared to *Cholec80*. These 40 videos add both extra training data for the network to learn on, and extra validation data to get a more accurate estimate of the network loss. Second, the folds created in [1] were



(a) *RSDNet* [16] architecture. (b) *UTE* [12] architecture.

Figure B.2: (a) *RSDNet* [16] architecture. The network consists of a *ResNet-152* [7] backbone which extracts visual features from the frames. The features get passed through an LSTM layer after which they get concatenated with the elapsed time in minutes. Finally, two single-node linear layers jointly predict the RSD and Progress. Figure taken from the original paper [16]. (b) *UTE* [12] architecture. This simple network consists of 3 linear layers. The second-to-last layer is used in [12] to create embeddings. Figure taken from the original paper [12].

optimized using a genetic algorithm, these folds however are not publicly available. Instead, we opted to randomly generate our folds, which may have resulted in a worse distribution of sequence lengths.

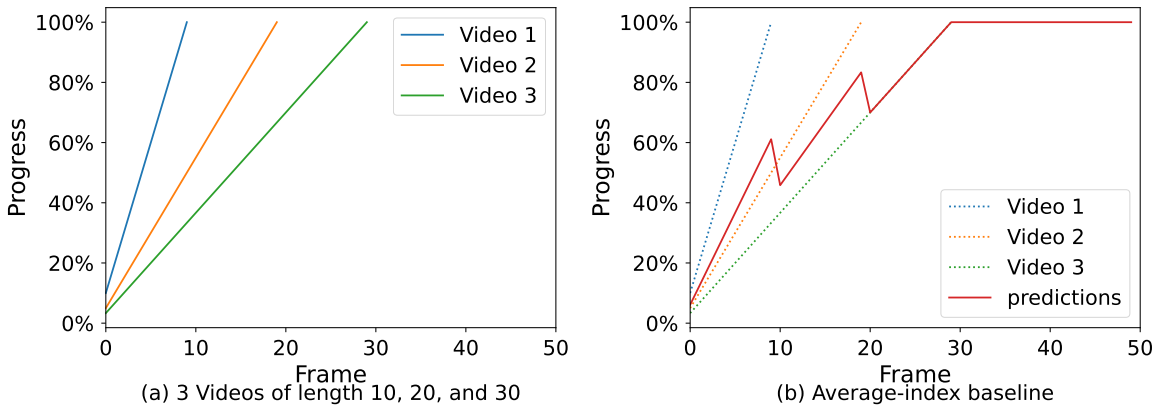
B.3. UTE

UTE [12] addresses the problem of detecting and segmenting phases in activity videos. Obtaining a phase label for each frame by human annotation is very expensive, *UTE* makes this process easier by labelling phases in an unsupervised manner. The network architecture used is a simple 3-layer MLP and can be seen in Fig. B.2(b). As input the network takes frame embeddings, such as dense trajectories [17] or I3D embeddings [3]. By predicting progress from frame embeddings they obtain a new temporal embedding, which can be used to temporally segment the activity video into phases. The main idea behind this strategy is similar to that of using phase prediction as a proxy for progress. For many activities, the phases follow a logical order, *i.e.* when making coffee the actor will get a cup early in the activity.

Kukleva *et al.* [12] train UTE on the Breakfast dataset in two ways. First, they train on each activity class individually. Second, they train on the dataset as a whole. We opt to go for the second case, training our methods on the full Breakfast dataset. This does make the problem more difficult, as the networks now have to learn to distinguish the visual information of 10 activity classes.

C

Average-Index Baseline



(a) 3 Videos of length 10, 20, and 30

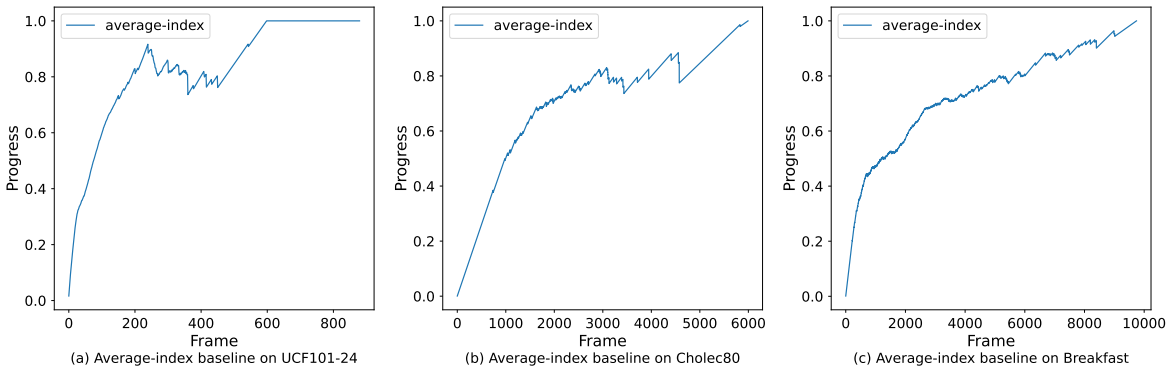
(b) Average-index baseline

Figure C.1: *Average-index* baseline on videos of length 20, 20, and 30 *a* and the resulting predictions *b*. The *average-index* baseline follows the average trajectory of the videos. This trajectory changes as videos end, causing a non-monotonically decreasing prediction. The *average-index* baseline always predicts 100% for videos that are longer than the longest training video. The *average-index* baseline minimises its error at each timestep, but except for the final video never gets the correct progress prediction.

Fig. C.1 shows an example calculation of the baseline on three videos of length 10, 20, and 30 *(a)*, and the resulting predictions *(b)*. For the first 10 frames the predictions are the average of videos 1, 2, and 3. For frames $10-20$, the predictions are the average of videos 2 and 3. For frames $20-30$ the prediction follows video 3 exactly because this is the only video left. Finally, after frame 30 the *average-index* baseline has no training data, so the predictions are pinned at 100%. Note that the predictions from the *average-index* baseline are non-monotonically increasing. This means that, unlike real activity progress, the predictions may go down. Fig. C.2 shows the *average-index* baseline predictions on the *UCF101-24*, *Cholec80*, and *Breakfast* datasets. The predicted progress increases rapidly at first. After this rapid increases it increases much slower and flatter till 100%.

The *average-index* baseline gets a good MAE score for the progress prediction task. By predicting the average progress at each frame it is able to minimise its error at each timestep. However, the *average-index* baseline is not very useful in real-world scenarios. The *average-index* baseline cannot exactly match the progress of any single video. The progress tends to increase rapidly at first, and then flatten out. Furthermore, unless the tested video is longer

than the longest video in the training set, the *average-index* baseline will never predict 100%. For all these same reasons a network that learns to predict progress simply by counting is equally problematic.



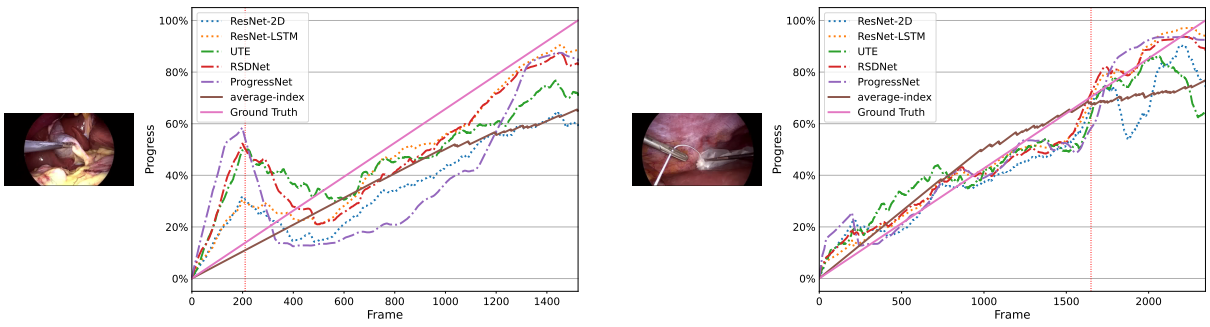
(a) Average-index baseline on UCF101-24 (b) Average-index baseline on Cholec80 (c) Average-index baseline on Breakfast

Figure C.2: *Average-index* baseline on *UCF101-24*, *Cholec80*, and *Breakfast*. The predicted progress increases rapidly at first due to the many short videos. After this, the predictions increase much slower to 100%. On *UCF101-24* the longest train video is 599 frames, while the longest test video is 880 frames. For the last 291 frames of the longest test video, the *average-index* baseline always predicts 100%. The *average-index* baseline is able to minimise its per-frame error on each dataset, but in doing so makes progress predictions which are not useful for real-world applications.

D

Results

In this section, we show visualisations of predictions the networks make on various videos of the datasets. Fig. D.1(a) shows the networks making predictions on Video-04 of the *Cholec80* dataset. The networks recognize the start of the surgical procedure by the first appearance of the medical tool. Because of this, the networks correct their prediction downwards. Fig. D.1(b) shows the networks making predictions on Video-05 of the *Cholec80* dataset. The networks again recognize a medical tool, the collection bag, and recognize this as the end of the procedure. All of them adjust their progress upwards towards 100%. This shows that when the visual information is clearly present in the datasets the methods are able to make use of this.



(a) (b)

Figure D.1: (a) Progress prediction example on Video-04 of *Cholec80* at timestamp $t{=}210$. (b) Progress prediction example on Video-05 of *Cholec80* at timestamp $t{=}1650$. The methods recognize the collection bag and correct their progress to signal the end of the procedure.

For the next results on *UCF101-24* we compare the predictions made by *ProgressNet* when given *full-video* sequences and *video-segments*. No networks, except for *ProgressNet* on *full-video* sequences, were able to learn on this dataset. They always predict a progress value of approximately 50% and are not interesting to look at. Fig. D.2(a) and Fig. D.2(b) show predictions made by *ProgressNet* on videos from the 'Biking' activity and the 'Fencing' activity on both *full-video* sequences and *video-segments*. When given *video-segments ProgressNet* is unable to learn and predicts a static value of approximately 60%. When given *full-sequences* we see that it approximately follows the ground truth values. If *ProgressNet* is predicting based on visual information we expect it to adjust its progress predictions based on recognizing phase transitions, like in Fig. D.1(a) and Fig. D.1(b). Instead, the progress appears to be increasing at a linear rate. Furthermore, if *ProgressNet* is using visual information

the predictions when given *video-segments* should also be able to recognize the same visual information, but instead this always predicts 60%. This shows that *ProgressNet* is simply counting frames and predicting an average progress. Fig. D.2(c) and Fig. D.2(d) show predictions made by *ProgressNet* on two videos from the 'GolfSwing' activity on both *full-video* sequences and *video-segments*. When given *video-segments ProgressNet* is again unable to learn and predicts a static value of approximately 60%. For Fig. D.2(c) it appears the network is responding to the visual information of the golfer swinging to update its prediction. However, when we look at Fig. D.2(d) the network does not respond to any visual information, and instead just predicts a linearly increasing progress value again. We conclude that *ProgressNet* is unable to learn from the visual information in *UCF101-24*.

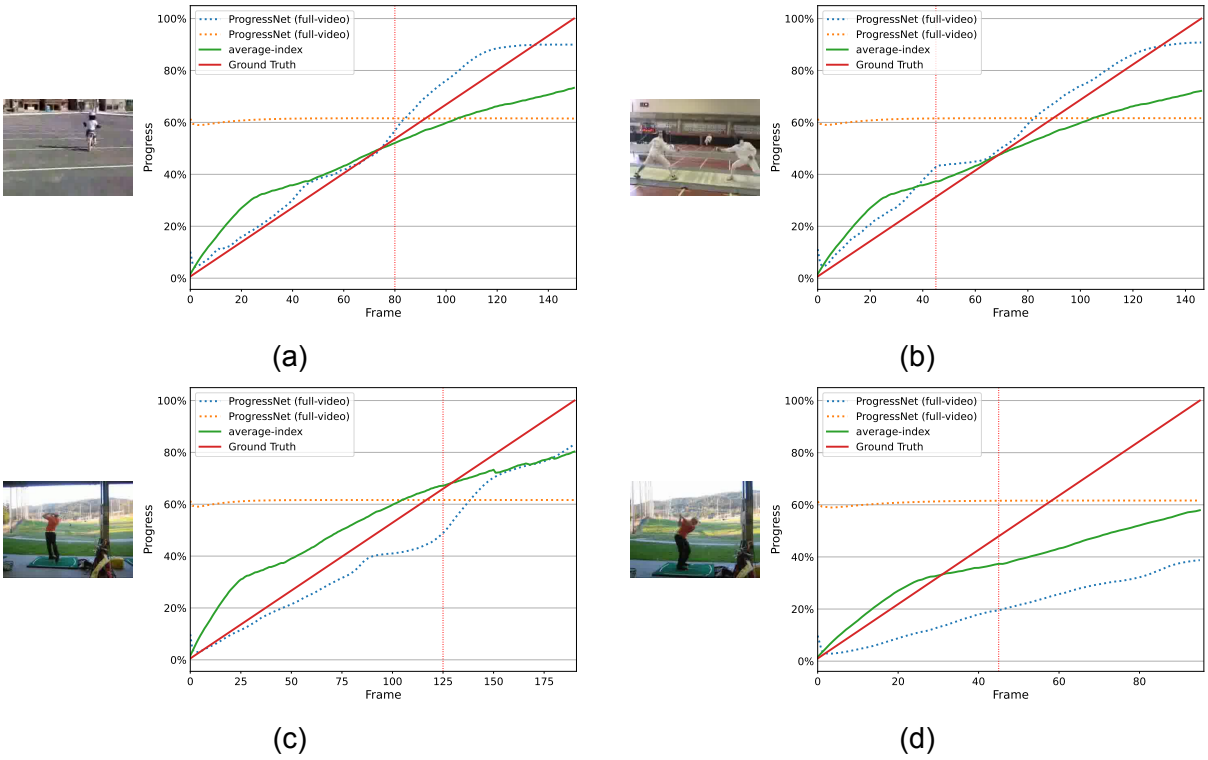
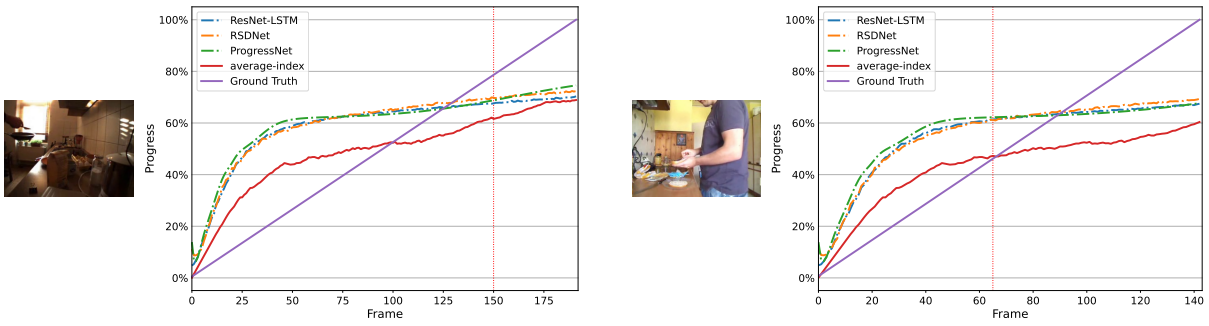


Figure D.2: (a) *ProgressNet* predictions on a video from the 'Biking' activity for both *full-video* sequences and *video-segments* at timestamp $t{=}80$. (b) *ProgressNet* predictions on a video from the 'Fencing' activity for both *full-video* sequences and *video-segments* at timestamp $t{=}45$. (c) *ProgressNet* predictions on a video from the 'Fencing' activity for both *full-video* sequences and *video-segments* at timestamp $t{=}125$. (d) *ProgressNet* predictions on a video from the 'Fencing' activity for both *full-video* sequences and *video-segments* at timestamp $t{=}45$. When given *video-segments* the predictions stay static around 60%. When given *full-video* sequences the predictions tend to increase in a linear fashion without responding to the change in visual information.

Finally, Fig. D.3(a) and Fig. D.3(b) show predictions made by all the LSTM-based networks when given *full-video* sequences of *random-noise* inputs. The networks all learn to count and make approximately the same predictions on both the videos. Note that because neither of these videos are the longest video in the dataset the predictions never reach 100%.



(a) (b)

Figure D.3: (a) Network predictions on a video from the 'pancake' cooking activity for all LSTM-based methods when given *full-video* sequences of *random-noise* at timestamp $t{=}150$. (b) Network predictions on a video from the 'sandwich' cooking activity for all LSTM-based methods when given *full-video* sequences of *random-noise* at timestamp $t{=}65$. All networks make approximately the same predictions and follow the same trajectory as the *average-index* baseline. The networks all learn to count.

Bibliography

[1] Ivan Aksamentov et al. “Deep Neural Networks Predict Remaining Surgery Duration from Cholecystectomy Videos”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2017.

[2] Federico Becattini et al. *Am I Done? Predicting Action Progress in Videos*. 2017. DOI: 10.48550/ARXIV.1705.01781. URL: https://arxiv.org/abs/1705.01781.

[3] Joao Carreira and Andrew Zisserman. *Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset*. 2018. arXiv: 1705.07750 [cs.CV].

[4] Dave Epstein, Boyuan Chen, and Carl Vondrick. *Oops! Predicting Unintentional Action in Video*. 2019. arXiv: 1911.11206 [cs.CV].

[5] Isabel Funke, Dominik Rivoir, and Stefanie Speidel. *Metrics Matter in Surgical Phase Recognition*. 2023. arXiv: 2305.13961 [cs.CV].

[6] Ross Girshick. *Fast R-CNN*. 2015. arXiv: 1504.08083 [cs.CV].

[7] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[8] Kaiming He et al. “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition”. In: *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 346–361. DOI: 10.1007/978-3-319-10578-9_23. URL: https://doi.org/10.1007%2F978-3-319-10578-9_23.

[9] Hueihan Jhuang et al. “Towards Understanding Action Recognition”. In: *2013 IEEE International Conference on Computer Vision*. 2013, pp. 3192–3199. DOI: 10.1109/ICCV.2013.396.

[10] Hilde Kuehne, A. B. Arslan, and T. Serre. “The Language of Actions: Recovering the Syntax and Semantics of Goal-Directed Human Activities”. In: *Proceedings of Computer Vision and Pattern Recognition Conference (CVPR)*. 2014.

[11] Hilde Kuehne, Juergen Gall, and Thomas Serre. “An end-to-end generative framework for video segmentation and recognition”. In: *Proc. IEEE Winter Applications of Computer Vision Conference (WACV 16)*. Lake Placid, Mar. 2016.

[12] Anna Kukleva et al. *Unsupervised learning of action classes with continuous temporal embedding*. 2019. arXiv: 1904.04189 [cs.CV].

[13] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[14] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. *UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild*. 2012. arXiv: 1212.0402 [cs.CV].

[15] Andru Putra Twinanda et al. *EndoNet: A Deep Architecture for Recognition Tasks on Laparoscopic Videos*. 2016. arXiv: 1602.03012 [cs.CV].

[16] Andru Putra Twinanda et al. “RSDNet: Learning to Predict Remaining Surgery Duration from Laparoscopic Videos Without Manual Annotations”. In: *IEEE Transactions on Medical Imaging* 38.4 (Apr. 2019), pp. 1069–1078. DOI: 10.1109/tmi.2018.2878055. URL: https://doi.org/10.1109%2Ftmi.2018.2878055.

[17] Heng Wang and Cordelia Schmid. “Action Recognition with Improved Trajectories”. In: *2013 IEEE International Conference on Computer Vision*. 2013, pp. 3551–3558. DOI: 10.1109/ICCV.2013.441.