Is there progress in activity progress prediction?

de Boer, Frans ; van Gemert, Jan C.; Dijkstra, Jouke; Pintea, Silvia L.

DOI
10.1109/ICCVW60793.2023.00318

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the 2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)

2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW) | 979-8-3503-0744-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCVW60793.2023.00318

Is there progress in activity progress prediction?

Frans de Boer[1] Jan C. van Gemert[1] Jouke Dijkstra[2] Silvia L. Pintea[1,2]

[1] Computer Vision Lab, Delft University of Technology
[2] Division of Image Processing (LKEB), Leiden University Medical Center

Abstract

Activity progress prediction aims to estimate what percentage of an activity has been completed. Currently this is done with machine learning approaches, trained and evaluated on complicated and realistic video datasets. The videos in these datasets vary drastically in length and appearance. And some of the activities have unanticipated developments, making activity progression difficult to estimate. In this work, we examine the results obtained by existing progress prediction methods on these datasets. We find that current progress prediction methods seem not to extract useful visual information for the progress prediction task. Therefore, these methods fail to exceed simple frame-counting baselines. We design a precisely controlled dataset for activity progress prediction and on this synthetic dataset we show that the considered methods can make use of the visual information, when this directly relates to the progress prediction. We conclude that the progress prediction task is ill-posed on the currently used real-world datasets. Moreover, to fairly measure activity progression we advise to consider a, simple but effective, frame-counting baseline.

1. Introduction

Visual activity progress prediction is vital to our day-to-day lives: *e.g*. in cooking, we predict how fast the food is ready; in healthcare, estimating how long a surgery will take allows for better resource allocation and shorter waiting times; and for video-editing knowing where an activity begins and ends helps with automatic cropping of the desired video ranges. Here, we define activity progress prediction as the task of predicting the percentage of completion of an activity in a video in an online setting, *i.e*.: without access to the length of the video. 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 progress prediction is extracting meaning from the visual inputs, which, ideally relates to the specific phases of the activity and, thus, enables predicting progress.

To address this challenge, current methods rely on deep networks, such as *VGG-16* [23], *ResNet* [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 embeddings and recurrent connections are useful for extracting visual information and keeping track of the activity progression over time, they may also overfit to uninformative artifacts. Here, we aim to analyze if such undesirable learning strategies are occurring when performing progress prediction.

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 simple non-learning baseline methods such as simply frame counting.

We evaluate models on various dataset types and regimes. We examine the learning methods when they are presented with the full videos during training. In addition, to avoid overfitting to absolute time/frame progression, we also evaluate methods when trained on randomly sampled video segments. For randomly sampling video segments, it is not possible to do frame-counting, and only the visual information is available 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 frame-counting. Finally, we design a precisely controlled synthetic progress prediction dataset, *Progress-bar*, on which the visual information is directly related to the progress.

2473-9944/23/$31.00 ©2023 IEEE
DOI 10.1109/ICCVW60793.2023.00318

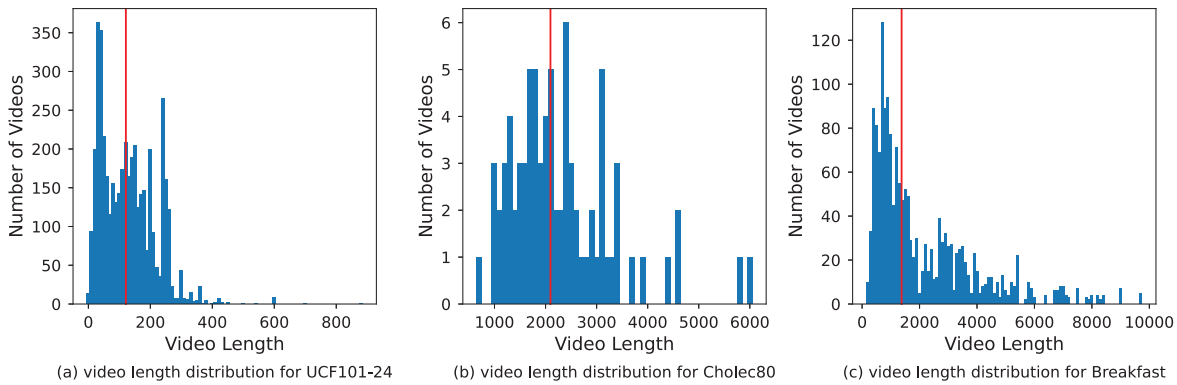


Figure 1. Length distributions for *UCF101-24*, *Cholec80*, and *Breakfast*. *UCF101-24* are grouped into bins of size 10, for *Cholec80* and *Breakfast* the bins are of size 100. Most notable is the long-tail distribution of the video lengths in the *Breakfast* dataset, which makes progress prediction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

Difficulties in current progress prediction. Progress prediction methods [3, 17, 26] evaluate on complicated and realistic datasets such as *UCF101-24* [24], *Breakfast* [15, 16], and *Cholec80* [25]. The appearance of the activities in these videos is diverse. And the activity length drastically varies between videos in these datasets, as shown Fig. 1. *UCF101-24* and *Breakfast* follow a long-tail distribution, with few videos containing long activities. Moreover, there can be unexpected activity progressions: *e.g.* the pancake gets burned, or there is a surgery lag. Also, 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 observer. Therefore, we arrive at two main questions we aim to address here: (i) *How well can methods predict activity progression on the current datasets?* 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 analyze their performance on the currently used datasets.

Remaining Duration. A topic closely related to progress prediction is Remaining Duration (RD) prediction. While the goal of progress prediction is to predict the course of the activity as a percentage value in $[0, 100\%]$, RD aims at predicting the 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. If an action consistently consists of separate sub-tasks or phases of similar duration, then recognizing the current phase gives a good approximation of the progress. Previous work jointly performs phase-based progress prediction and surgical gesture recognition [27], jointly predicting the phase and the surgery tools [25], or by 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 indica-

tor 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 focus on the more fine-grained targets of activity progression at every frame.

3. Activity progress prediction

We formulate activity 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 1 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 denote the video sampling strategy implementing both points (a) and (b), as *video-segments*. On *video-segments* the methods can only rely on 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 the 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.

Frame-counting. Finally, we consider a non-learning baseline which computes training-set statistics. It is a frame-counting strategy that makes per-frame average progress predictions. 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 different datasets: *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 of *ProgressNet* first trains on *telic* activities, and then fine-tunes 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.

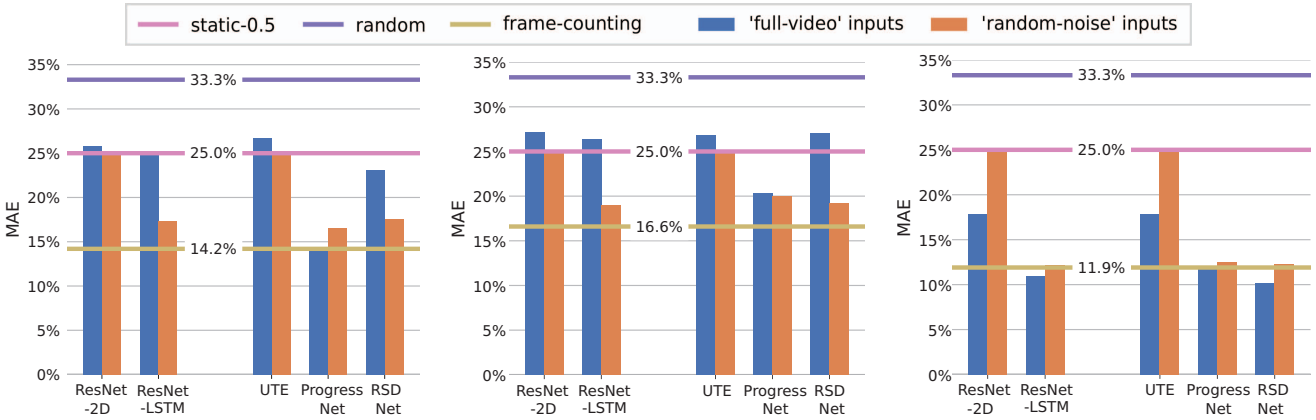
UCF101-24 contains training videos of up to 599 frames, while *Cholec80* and *Breakfast* contain videos with thousands of frames. When training on *full-videos*, we could not train the *ProgressNet* model on the original *Cholec80* and *Breakfast* datasets, because of the long videos leading to memory problems. Thus, for the experiments on *full-videos*, we use a subsampled version of *Cholec80* from 1 fps to 0.1 fps (the original fps is 25; [26] subsamples this down to 1 fps); and we subsample the *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 training. For all experiments we report the MAE (Mean Absolute Error) in percentages. Our code is available online at https://github.com/Frans-db/progress-prediction.

[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



(a) *UCF101-24* on *full-videos*. (b) *Breakfast* on *full-videos*. (c) *Cholec80* on *full-videos*.

Figure 2. **MAE scores on *full-videos***. We plot the MAE in percentages for all learning methods when inputting both *full-video* data and *random-noise*. (a) MAE for the *UCF101-24* dataset: For all methods except *ProgressNet* inputting *random-noise* performs on par or better than inputting *full-videos*. (b) MAE for the *Breakfast* dataset: When using *random-noise* as input to the methods, they perform on par or better than when inputting *full-videos*, indicating that the methods overfit to training artifacts. (c) MAE for the *Cholec80* dataset: On this dataset, using visual *full-videos* is better than inputting *random-noise*, however the *frame-counting* baseline remains hard to exceed.

4.3. *(i) How well can methods predict activity progression on the current datasets?*

(i.1) Progress predictions on *full-videos*. Here we want to test how well the learning-based models perform when trained on *full-videos*. We compare this with using *random-noise* as input – we replace each frame with randomly sampled noise. Intuitively, learning from *random-noise* over complete videos will give recurrent models access to frame indices, and this should reach the *frame-counting* baseline, which computes dataset statistics per frame. 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 *frame-counting*.

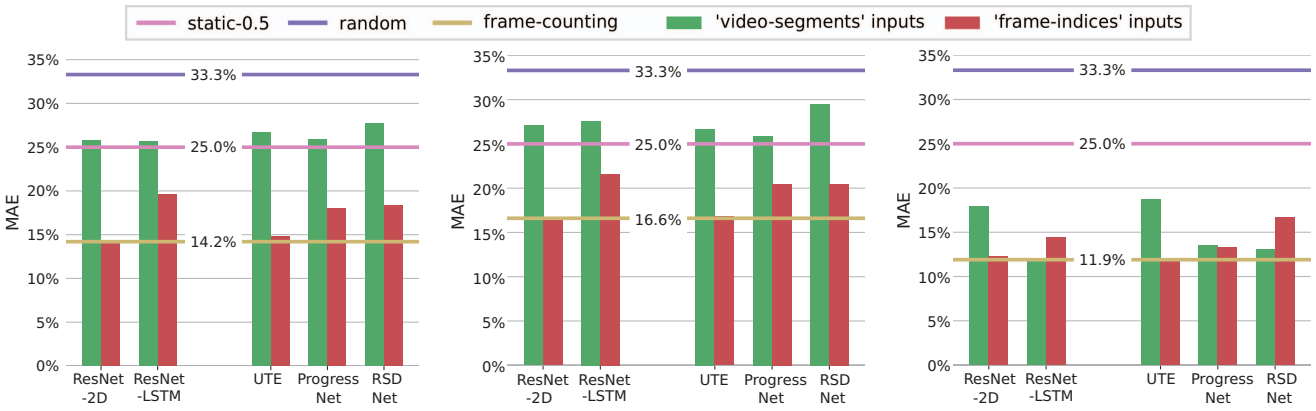
Fig. 2(a) shows that the *full-video* visual information (blue bars) is less useful than inputting *random-noise* (orange bars). When training on the *full-videos* of *UCF101-24* both the *ResNet-2D* and *UTE* models perform on par with the *static-0.5* baseline. This is because these spatial-only networks 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 on *full-videos*, *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. When these recurrent networks are provided with *random-noise* they can only rely on the number of frames seen so far, and thus reach the *frame-counting* baseline. *ProgressNet* is the only outlier here: when given *full-videos* it performs better than when given *random-noise* as input. However, *ProgressNet* still cannot outperform the non-learning *frame-counting* baseline.

For *Breakfast* in Fig. 2(b) 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 on *full-videos*, indicating that they are overfitting on the training data. When provided with *random-noise* as input, they again can only rely on the number of frames seen, and thus approach the *frame-counting* naive baseline.

Cholec80 in Fig. 2(c) is the only dataset where the spatial-only networks *ResNet-2D* and *UTE* perform better than the *static-0.5* baseline. Here, we see that using the *full-videos* (blue bars) is better than inputting *random-noise* (orange bars). This hints to the visual information present in this dataset being indicative of the activity progress. When inputting *random-noise* the spatial-only methods again perform on par with the *static-0.5* baseline, as expected. However, this dataset still remains challenging as the methods are not far from the *frame-counting* baseline. *ResNet-LSTM* and *RSDNet* are the only who perform slightly better than this naive baseline, indicating that they can extract some useful visual information from the video frames.

Observation: *The current datasets make it difficult for the progress prediction methods to extract useful visual information. Therefore, the methods overfit to training set artifacts, and are outperformed by simple baselines based on dataset statistics.*

(i.2): Progress predictions on *video-segments*. We test the performance of learning methods when trained to predict progress from *video-segments*. Using *video-segments*



(a) *UCF101-24* on *video-segments*. (b) *Breakfast* on *video-segments*. (c) *Cholec80* on *video-segments*.

Figure 3. **MAE scores on *video-segments***. We plot the MAE in percentages for all considered methods when inputting both *video-segments* and *frame-indices*. (a) MAE for the *UCF101-24* dataset: 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. (b) MAE for the *Breakfast* dataset: Also here *frame-indices* are more informative than the visual data. (c) MAE for the *Cholec80* dataset: All learning methods perform on par with the *frame-counting* baseline, except for *RSDNet* which is slightly worse. This could be due to suboptimal hyperparameter settings.

should encourage the methods to focus more on the visual information and not on the temporal position of the frames, as this is no longer informative. We compare inputting *video-segments* with inputting *frame-indices* – the absolute index of each frame, replicated as images. Intuitively, learning from *frame-indices* should be on par with the *frame-counting* baseline, since the only information available is the dataset statistics per frame. Ideally, we would expect all methods to solve the progress prediction task by relying on visual information, and therefore having lower errors than when inputting *frame-indices*. Again, we also compare with the naive baselines: *static-0.5*, *random* and *frame-counting*.

Fig. 3(a) shows that the visual information encoded in *video-segments* (green bars) is less useful than knowing the current frame index (red bars). When trained on *video-segments* of *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(a) is better than the *frame-counting* baseline, however, here it fails to learn when trained on *video-segments*. 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.

The results on the *Breakfast* dataset in Fig. 3(b) are similar to those of *UCF101-24* in Fig. 3(a). None of the networks can extract useful information from the *video-segments*. All methods improve when trained on *frame-indices*. The improvement is again more obvious for *ResNet-2D* and *UTE*. Moreover, all results are on par with, or worse than the *frame-counting* baseline.

On *Cholec80* in Fig. 3(c) all results are close to the *frame-counting* baseline. This is dissimilar to Fig. 2(c) where inputting visual data was better than inputting random noise of the same length as the full video. Again, *ResNet-2D* and *UTE* improve when provided with *frame-indices* as input. For *ResNet-LSTM* and *ProgressNet* and *RSDNet* the performance is on par with the *frame-counting* baseline when trained on *video-sequences* indicating that these methods overfit to the training data. When trained on *frame-indices* most methods approach the *frame-counting* baseline, as this is the information encoded in the frame indices across the full training set. *RSDNet* performs worst when given *frame-indices* as inputs; we hypothesise that this is due suboptimal hyperparameter settings.

Observation: *When restricting the models to rely only on visual information, the models are outperformed by simply considering the current frame index, and performing dataset statistics. This is due to the current progress prediction datasets not containing sufficient visual information to guide progress prediction.*

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

We observe that current progress prediction datasets are not well-suited for the task, as the learning models fail to extract useful information from visual data alone. To test that the problem is indeed with the datasets used for evaluation and not the learning models, we test here if progress prediction is possible from visual data alone. For this we aim to construct a synthetic dataset in such a way that the learning-based methods perform optimal using visual infor-

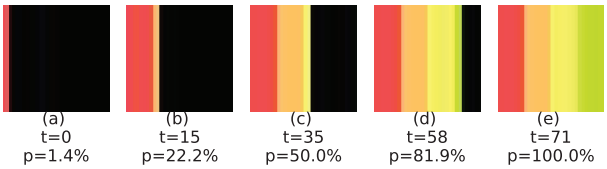


Figure 4. Visualisation of a progress bar from our synthetic *Progress-bar* dataset at timesteps t=0, t=15, t=35, t=58, and t=71. Each coloured section indicates visually a 25% section, but due to variance in the speed, the actual video progress may differ at these points.

mation, and outperform all the naive baselines.

Our synthetic *Progress-bar* dataset, shown in Fig. 4, 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. Therefore, even on this simple dataset it is impossible to make no errors. 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 *frame-counting* 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 temporal information 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 overfitting, we use *ResNet-18* as a backbone for *ResNet-2D*, *ResNet-LSTM*, and *RSDNet*. *ProgressNet* and *UTE* remain unchanged.

Fig. 5 shows the results of all the learning-based methods when predicting progress from both *full-videos* and *video-segments*. For this dataset the *frame-counting* 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* relies on 3D convolutional embeddings over a temporal-window of size 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

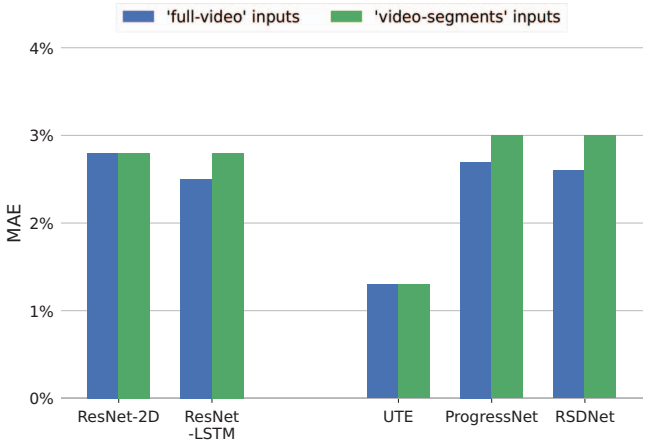


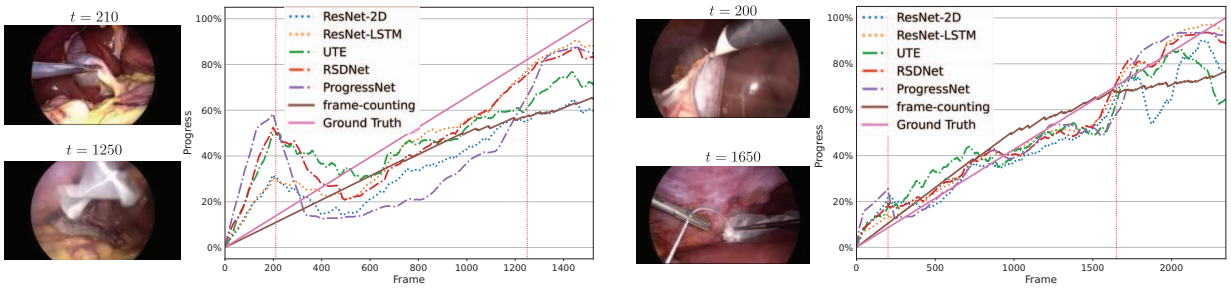
Figure 5. **MAE scores on our synthetic *Progress-bar* dataset, when training on *full-videos* and *video-segments***. The *frame-counting* baseline has an MAE of 12.9%, while the *static* baseline is at 25% and the *random* baseline at 33.3%. We see that all methods outperform the *frame-counting* baseline. *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.

are less likely to get sampled, thus the progress prediction methods will have a higher error there.

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

5. Discussion and limitations of our analysis

Discussion. This paper empirically shows that the current progress prediction datasets do now allow for learning useful visual information, and methods are outperformed by naive baselines relying on dataset statistics. We observe that the progress prediction models perform well on the training data, yet cannot generalize well to the unseen test data. As future research, it would be interesting to pinpoint the photometric artifacts that the models overfit to. However, we also saw that some useful visual information was learned on the *Cholec80*. This may be due to the presence of clear visual phase delineators. Fig. 6(a) and Fig. 6(b) show examples of predictions on the *Cholec80* dataset. The first frames highlighted in these videos (t=210 and t=200) are the moment when the first medical tool is present in the video, and the progress prediction methods adjust their predictions to this new visual information. Similarly, the second highlighted timesteps (t=1250 and t=1650) represent the moment the collection bag is present which signals the end of the procedure. On our synthetic *Progress-bar* dataset



(a) Video-04 of *Cholec80*

(b) Video-05 of *Cholec80*

Figure 6. Activity progress prediction examples of *Cholec80*. (a) Video-04 at timestamps t=210 and t=1250. (b) Video-05 at timestamps t=200 and t=1650. At t=210 and t=200 the methods recognize the medical tool, and correct their progress downwards to signal the start of the medical procedure. At t=1250 and t=1650 the methods recognize the collection bag and correct their progress to signal the end of the procedure.

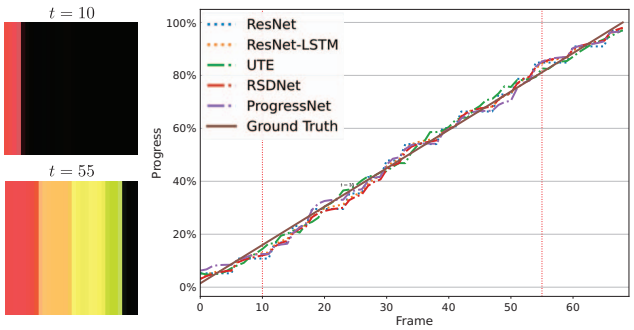


Figure 7. Progress prediction example on Video-00015 of our synthetic *Progress-bar* dataset at timestamps t=10 and t=55. The learning methods can almost perfectly follow the ground truth.

in Fig. 7 we also show the predictions and highlight two moments in the videos. 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 analyze, 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 embeddings, as well as recurrent 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 *frame-counting* 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 on the currently used datasets, the progress prediction methods can fail to extract useful information from visual data, and are exceeded by simple non-learning baselines based on frame counting. Additionally, we evaluate all the methods on a synthetic dataset we specifically design 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 naive, non-learning baselines. We conclude that in its current form the task of progress prediction is ill-posed, as the currently used datasets for progress prediction are not suitable for this task.

Acknowledgements. This work was done with the support of the Eureka cluster Program, IWISH project, grant number AI2021-066. Jan van Gemert is financed by the Dutch Research Council (NWO) (project VI.Vidi.192.100).

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

[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

[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. 2

[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. 2

[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