Deep learning methods for clinical workflow phase-based prediction of procedure duration: a benchmark study

Emanuele Frassini, Teddy S. Vijfvinkel, Rick M. Butler, Maarten van der Elst, Benno H. W. Hendriks & John J. van den Dobbelsteen

To cite this article: Emanuele Frassini, Teddy S. Vijfvinkel, Rick M. Butler, Maarten van der Elst, Benno H. W. Hendriks & John J. van den Dobbelsteen (2025) Deep learning methods for clinical workflow phase-based prediction of procedure duration: a benchmark study, Computer Assisted Surgery, 30:1, 2466426, DOI: 10.1080/24699322.2025.2466426

To link to this article: https://doi.org/10.1080/24699322.2025.2466426

COMPUTER ASSISTED SURGERY
2025, VOL. 30, NO. 1, 2466426
https://doi.org/10.1080/24699322.2025.2466426

RESEARCH ARTICLE

OPEN ACCESS

Deep learning methods for clinical workflow phase-based prediction of procedure duration: a benchmark study

Emanuele Frassini[a], Teddy S. Vijfvinkel[a,b], Rick M. Butler[a], Maarten van der Elst[a,b], Benno H. W. Hendriks[a,c] and John J. van den Dobbelsteen[a]

[a]Mechanical, Maritime and Materials Engineering, Delft University of Technology, Delft, The Netherlands; [b]Reinier de Graaf Hospital, Delft, The Netherlands; [c]Medical Systems, Philips Medical Systems, Best, The Netherlands

ABSTRACT
This study evaluates the performance of deep learning models in the prediction of the end time of procedures performed in the cardiac catheterization laboratory (cath lab). We employed only the clinical phases derived from video analysis as input to the algorithms. Our results show that InceptionTime and LSTM-FCN yielded the most accurate predictions. InceptionTime achieves Mean Absolute Error (MAE) values below 5 min and Symmetric Mean Absolute Percentage Error (SMAPE) under 6% at 60-s sampling intervals. In contrast, LSTM with attention mechanism and standard LSTM models have higher error rates, indicating challenges in handling both long-term and short-term dependencies. CNN-based models, especially InceptionTime, excel at feature extraction across different scales, making them effective for time-series predictions. We also analyzed training and testing times. CNN models, despite higher computational costs, significantly reduce prediction errors. The Transformer model has the fastest inference time, making it ideal for real-time applications. An ensemble model derived by averaging the two best performing algorithms reported low MAE and SMAPE, although needing longer training. Future research should validate these findings across different procedural contexts and explore ways to optimize training times without losing accuracy. Integrating these models into clinical scheduling systems could improve efficiency in cath labs. Our research demonstrates that the models we implemented can form the basis of an automated tool, which predicts the optimal time to call the next patient with an average error of approximately 30 s. These findings show the effectiveness of deep learning models, especially CNN-based architectures, in accurately predicting procedure end times.

KEYWORDS
Deep learning; time series; CNN; regression

1. Introduction

In cardiology, coronary artery disease is anticipated to rise by 40% by 2040 [1]. This increase puts strain on cardiac catheterization laboratories (cath labs), the primary treatment facility for minimally invasive cardiac procedures. Improving the workflow in the cath lab is necessary to anticipate this problem. The estimation of the duration of endovascular procedures is closely tied to the workflow. Deviations, such as adverse events or inefficiencies, can significantly influence procedure duration. Conversely, understanding and modeling the workflow can provide valuable insights for predicting the remaining duration. The workflow is influenced by a variety of factors. One factor is the estimation of the duration of endovascular procedures. Knowing the expected time to the completion (ETC) of a procedure is a difficult task since duration can vary substantially even across procedures of the same type. The expertise of the cardiologist, the comorbidities of the patient, or unexpected occurrences during the intervention may lead to inaccurate ETC estimates. Accurate real-time assessment of the duration can increase scheduling efficiency by allowing administrators to dynamically reschedule before a procedure has run overtime.

Currently, studies aiming for cath lab workflow optimization through ETC are lacking. In the past few years, substantial attention has been paid to programs aimed at minimizing 'door-to-balloon' time in the cath lab. For example, the Lean Six Sigma methodology has been implemented to treat patients with acute

CONTACT Emanuele Frassini e.frassini@tudelft.nl Mechanical, Maritime and Materials Engineering, Delft University of Technology, Delft, The Netherlands

coronary syndromes more quickly [2]. Although useful, these kinds of solutions still rely on scheduling and preparation. A continual understanding of the evolution of elective treatments, which are planned in the hospital, might help to better accommodate the treatment for emergency patients with acute coronary syndromes. The scheduling process is complicated by the presence of (semi-)urgent patients and the variability in procedure durations [3]. It has been reported that schedulers frequently obtain necessary information through verbal interactions with the staff, leading to unwanted interruptions in the procedure [4]. One potential application of an accurate model for ETC prediction involves the creation of an automated tool that informs hospital personnel when a procedure is nearing completion. The automation could optimize the initiation of subsequent procedures, making the schedule process more efficient.

Surgical process modeling has proven to be helpful in understanding procedure progression and predicting the end time of a surgery [5]. Despite the emphasis on surgical processes, modeling of the actual workflow in cath labs has been underexplored [6]. However, this time–critical environment might benefit most from advances in workflow efficiency [7]. In recent times, artificial intelligence algorithms have increased the capabilities of these workflow analyses, potentially making them suitable for real-time use in clinical practice [8].

1.1. Related work

In the past years, estimated time to completion studies assessed preoperative durations using data from surgeons, patients, or a combination of both [9,10]. Later, intraoperative estimates were developed, with some studies depending on manual annotations or additional external data [11–14]. Guédon et al. [4] used a Support Vector Machine to predict ETC based on electrosurgical device activation, achieving a Mean Absolute Error (MAE) of 14 min, outperforming the estimates of the surgeons of 19 min. Twinanda [15] reported solid ETC findings utilizing just video-based approaches, exceeding statistical studies of historical surgeon data. Recently, Ariel [16] employed a pre-trained Video Transformer Network as a feature extraction module to predict ETC from surgical steps, achieving best results with a Transformer model (Symmetric Mean Absolute Percentage Error (SMAPE) $\approx$ 20%).

In this work, we want to demonstrate that clinical phases derived from video data can be reliably used as input for deep learning models to accurately predict the end time of a CAG procedure in the cath lab. Furthermore, we want to investigate the error behavior of these models, with a focus on the final minutes leading up to the end of the procedure. Lastly, we want to investigate how the dataset size influences testing and training time, as optimizing these aspects is fundamental for making our model fast and lightweight, which is important for a clinical application.

2. Materials and methods

2.1. Data acquisition

The data used for training and testing our models consists of video recordings of 222 coronary angiographic (CAG) procedures performed at Reinier de Graaf Hospital, a large regional hospital in The Netherlands. The study received approval from the regional medical ethics committee (METC), and informed consent was obtained from all patients and staff prior to recording the catheterizations. During the observational study period, diagnostic CAGs were conducted by nine different cardiologists and recorded with four Axis M1125 video cameras placed in the cath lab. The median duration of a diagnostic CAG in the dataset was 42.49 min (interquartile range: 34.25—52.54).

Figure 1 shows an example of the videos recorded with four cameras at Reinier de Graaf Hospital. A medical doctor, T.V., annotated the clinical phase for every second of each procedure in collaboration with an experienced interventional cardiologist. The annotation process has been performed with the with the annotation software Noldus Observer XT 15, Noldus Information Technology, Wageningen, The Netherlands. The annotators derived the phases by looking at visual clues from the videos. Relevant clinical phases were additionally checked and defined in consultation with the expert cardiologists that performed the recorded procedures. A total number of 14 different phases were annotated. Every procedure has been treated as a time series, where each data point consists only of the clinical phase, labeled with the time to the end of the procedure, expressed in seconds.

Table 1 presents a detailed description of each annotated workflow phase and their relative duration, expressed in minutes. The phases defined above constitute the single-dimension input feature for the DL models. Phases Fa and Ha are supplementary phases that do not occur in all procedures. In particular, phase Fa is observed in 17 procedures, representing 8% of the total, while phase Ha occurs in 46 procedures, accounting for 21%.

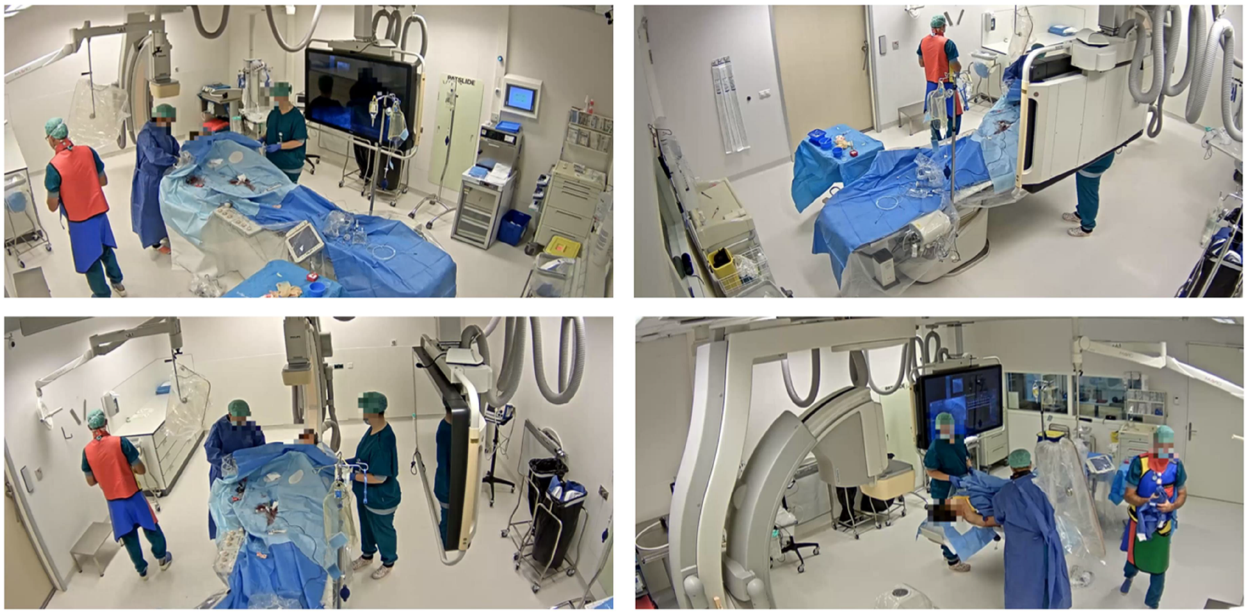

Figure 1. Example of anonymized video footage recorded during a diagnostic cardiac catheterization at Reinier de Graaf Hospital, used to annotate workflow steps.

Table 1. Definition of the clinical phases of a CAG procedure.

Clinical phase	Definition	Mean (IQR) duration [min]
A	Preparation prior to patient arrival	9.43 (3.84–12.87)
B	Patient arrival and transfer to table	0.56 (0.30–0.55)
C	Preparation with patient on table	10.89 (8.39–12.91)
D	Acquiring endovascular access	5.92 (3.32–6.70)
E	Guidance of first catheter to aortic root	0.39 (0.13–0.43)
F	Entering and recording of first coronary artery	5.20 (2.57–6.35)
Fa	Additional catheter required during phase F	4.52 (1.18–2.85)
G	Guidance of second catheter to coronary artery	1.01 (0.72–1.10)
H	Entering and recording of second coronary artery	6.52 (3.72–7.97)
Ha	Additional catheter required during phase H	3.40 (0.91–4.81)
I	Preparation of wound closure	1.79 (1.19–2.08)
J	Wound closure	3.46 (2.41–4.07)
K	Patient transfer off table and start cleaning	0.64 (0.32–0.72)
L	Cleaning after patient departure	1.33 (0.42–1.62)

2.2. Architectures

Deep learning (DL) has emerged as a potent tool for time series analysis, leveraging various architectures to capture complex temporal dependencies and patterns. It has been reported that sequential data can be effectively processed using deep neural networks (DNNs) to achieve state-of-the-art performance [17]. In this work, we focus on three main areas of DL for time series analysis. These include Recurrent, Convolutional, and Attention-based networks.

2.2.1. RNN models

Recurrent neural networks (RNNs) are the most commonly used NN architecture for sequence prediction problems [18]. For instance, RNN Encoder-Decoder (RNN-ED) frameworks, which use Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRUs), efficiently manage sequential data by making predictions at each time step based on previous states and outputs [18]. This enables robust anomaly identification in time series data, as demonstrated in multiple applications [19].

Among the RNN models, we decided to employ a Long Short-Term Memory network. LSTM are a type of recurrent neural network particularly well-suited for modeling sequential data and capturing temporal dependencies [20]. The core of our model consists of an LSTM layer with a specified number of units. This layer processes the input sequence and captures the temporal dependencies within the data. An LSTM layer consists of multiple LSTM units designed to produce entire sequences of hidden states. This ensures the preservation of temporal information throughout the network. The internal state of the model is reset after each procedure is analyzed. Each input sequence is therefore processed independently. A dropout regularization layer is incorporated following the LSTM layers to prevent overfitting. A masking layer is used to skip certain time steps in the input data, marked by a specified mask value (–1), to handle variable–length sequences. The output of the LSTM layer is fed into a dense layer with a ReLU activation function [21]. This layer reduces the dimensionality of the output to a single prediction value per time step. Together, these layers constitute the LSTM network.

2.2.2. CNN models

Convolutional neural networks (CNNs) are employed for time series analysis due to their effectiveness in feature extraction from raw data. They also showed efficiency in managing large networks with fully connected layers that can be challenging to train. CNNs decrease the number of parameters to be learned by limiting connections to local input regions and apply convolutional filters to extract hierarchical features from raw time series data. Furthermore, techniques like pooling provide translation invariance to the CNN, enhancing the robustness of the extracted features [22]. We employed two state-of-the-art CNN based architecture for time series regression, namely InceptionTime and LSTM-Fully convolutional network (LSTM-FCN).

The InceptionTime model is designed to efficiently capture multi-scale temporal patterns through parallel convolutions of different kernel sizes [23]. The core component of the architecture is the Inception module, which includes multiple layers. The bottleneck layer employs a 1×1 convolution to reduce dimensionality for input tensors with multiple channels. Parallel convolutions with kernel sizes of 41, 20, and 10 capture features at multiple scales simultaneously. Each convolution uses 32 filters and a stride of 1. A max pooling layer with a size of three captures dominant features while reducing dimensionality, followed by a convolution with 32 filters of size 1. Outputs from parallel convolutions and the max pooling path are concatenated along the feature dimension. Batch normalization is applied to stabilize and accelerate training, followed by a ReLU activation function. Residual connections are incorporated every three Inception modules to improve gradient flow and learning efficiency, mitigating the vanishing gradient problem and enabling the model to learn deeper representations [24].

In the LSTM-FCN architecture, the LSTM network was concatenated with a fully convolutional network. Three convolutional blocks were used, as initially proposed by Karim [25], with batch normalization following each convolutional layer. To prevent division by zero, a constant epsilon of 0.001 was added to the denominator of the batch normalization. The momentum, an optimization technique to determine the contribution of new batch statistics to the running mean and variance, was set to 0.99. ReLU activation was applied after batch normalization. The first convolutional block consists of a 1D convolutional layer with 128 filters and a kernel size of 8, followed by batch normalization, a ReLU activation function, and a squeeze-and-excite block to recalibrate channel-wise feature responses. The second convolutional block includes a 1D convolutional layer with 256 filters and a kernel size of 5, followed by batch normalization, a ReLU activation function, and another squeeze-and-excite block. Additionally, a modified squeeze-and-excite block, originally proposed by Wu [26], was implemented after the first two convolutional blocks to adaptively adjust the input feature maps. This block is crucial as different feature maps may affect subsequent layers differently during training. The original global pooling layer, acting as a majority vote within each time series, was removed from this block. The third convolutional block features a 1D convolutional layer with 128 filters and a kernel size of 3, followed by batch normalization and a ReLU activation function. Finally, the outputs from the LSTM and FCN branches are concatenated, combining temporal and spatial features. The concatenated features are passed through a dense layer with a ReLU activation function to produce the final prediction.

2.2.3. Attention-based models

Attention-based models are suitable for time series analysis because they model dependencies without regard for their location within the input or output sequences. The attention mechanism enhances the capacity to model intricate temporal relationships by dynamically focusing on relevant parts of the input sequence. For instance the Transformer model, which depends purely on attention mechanisms and eliminates recurrence, allows for higher parallelization and showed high performance in sequence modeling tasks [27]. We implemented two attention-based models, namely Transformer and LSTM with Attention mechanism (LSTM-Attention).

The Transformer model, originally designed for natural language processing, has shown remarkable success in various domains due to its ability to model long-range dependencies through self-attention [28]. We adapted this architecture for time series prediction. Our adapted architecture for time series prediction incorporates multiple components. The Transformer Encoder employs multi-head self-attention layers to capture dependencies across different time points, with each attention head focusing on a distinct subspace of the input to simultaneously analyze various aspects of the data. Layer normalization and residual connections are applied to stabilize training and improve gradient flow. Additionally, a position-wise feed-forward network, using 1D convolutions, introduces nonlinearity and further processes the outputs from the attention mechanism. This network, which follows the multi-head self-attention layers, includes two convolutional layers separated by a ReLU activation function.

Incorporating an attention mechanism into the LSTM network enhances its ability to focus on the most relevant parts of the input sequence. Here, we concatenated the LSTM layer, described in Section 2.2.1., with a dot-product attention layer, as originally proposed by Luong [29]. The attention mechanism picks information relevant to the current time step. The context vector is a weighted sum of column vectors from prior RNN hidden states. This mechanism computes alignment scores using a trainable weight matrix and bias. A softmax function [30] is then applied to obtain attention weights. A context vector is calculated as a weighted sum of the input sequence elements. This approach is well-suited to activities where each time step contains a single item of information [31]. The output of the attention mechanism is passed through a layer with a ReLU activation function, reducing the dimensionality to a single prediction value per time step.

2.2.4. Ensemble model

We employed an ensemble learning approach to enhance the prediction accuracy and robustness of our models. This technique combines the strengths of multiple models by averaging their outputs, and previously showed to reduce the variance and improve the accuracy of the predictions [32]. For our study, we selected the two best-performing models based on their validation performance and created an ensemble model by averaging their outputs. The two models selected for the ensemble were those with the lowest validation loss during training. In our case, the models were InceptionTime and LSTM-FCN. We chose to include only these two models since their performance achieved outstanding results compared to the other architectures. Each base model was trained independently using the same training and validation datasets, with early stopping and learning rate reduction callbacks employed to prevent overfitting and ensure optimal convergence.

After training, the predictions from each base model on the test dataset were obtained, and the final prediction for each time point in the time series was calculated by averaging the outputs of the two models. The advantages of this ensemble approach include increased robustness, since averaging the predictions mitigates individual model weaknesses. It reduces overfitting by combining models with different generalization errors. Furthermore, it improves accuracy by leveraging the complementary strengths of the individual models. The ensemble model was implemented by first training the selected models separately and then computing and averaging their predictions for the test set to obtain the final prediction.

2.3. Experimental setup

2.3.1. Implementation

The models were compiled using a SMAPE loss function. The loss function is computed as following:

$$\mathrm{SMAPE} = \frac{1}{T} \cdot \sum_{t=0}^{T-1} \left(\frac{\left| y^{t}_{\mathrm{pred}} - y^{t}_{\mathrm{true}} \right|}{\left| y^{t}_{\mathrm{true}} \right| + \left| y^{t}_{\mathrm{pred}} \right|} \right) \cdot 100, \tag{1}$$

where T is the total duration of the procedure, y^{t}_{true} and y^{t}_{pred} are the ground truth of the estimated time to the end of the procedure and the predictions of the models at the second t, respectively. SMAPE loss is particularly suited for the prediction of time series as it normalizes the absolute error by the sum of the actual and predicted values [16]. As a result, SMAPE is magnitude-invariant and maintains a consistent scale across videos of varying durations, providing a more comprehensive representation of ETC performance [33]. We evaluated our results considering both SMAPE score and the MAE metric. The latter is calculated as:

$$\mathrm{MAE} = \frac{1}{T} \cdot \sum_{t=0}^{T-1} \left(\left| y^{t}_{\mathrm{pred}} - y^{t}_{\mathrm{true}} \right| \right) \tag{2}$$

A limitation of MAE is its dependence on the magnitude of the values. This results in shorter videos typically exhibiting smaller errors, while longer videos tend to show larger errors. Furthermore, MAE does not account for the actual duration of the videos or the specific temporal locations where the predictions are made.

The Adam optimizer [34] was employed to provide an adaptive learning rate during training for efficient convergence. To prevent overfitting, early stopping was employed to end training if the validation loss did not improve for a specified number of epochs, set to 100, with the best weights restored. Additionally, the learning rate was halved if the validation loss plateaued within a factor of 0.001, for at least 50 epochs, enabling fine-tuning of the learning process. The training schedule consisted of a total of 100 different trials. In each trial, we trained and tested the six different models with a threefold cross validation approach. In each fold, we considered a consistent split for training and testing among all the different models. Thus, we were able to compare the performance without any bias given by a specific split of the data. During every fold, we computed the SMAPE and MAE of the models, and we then averaged the results among the threefolds. The repetition of this process for 100 times, each with a different split, is aimed at investigating the robustness of the models for the given task.

In our study, batches were constructed without shuffling to preserve the temporal sequence of data, which is critical for causal models designed for time-series predictions. For most architectures, batch normalization layers were not employed to avoid potential performance degradation when batches consist of continuous data points from the same sequence. [35] Instead, we used regularization techniques like dropout to improve generalization. However, the LSTM-FCN model includes batch normalization layers. To counteract potential issues, batches were constructed carefully to include diverse sequences rather than consecutive data points from the same source. This ensures that the statistics calculated by the batch normalization layers remained representative.

2.3.2. Research framework

We first conducted an initial pilot study on phase recognition. In this study, we employed DL models to predict the phases within the same CAG procedures that are used as input in this work. Our pilot study highlighted that a temporal window of five seconds in the prediction enabled the models to achieve high accuracy in phase recognition. Specifically, we found that analyzing one data point every 5 s was sufficient for accurate phase detection. Expanding the window size to up to 2 min produced similar accuracy, whereas increasing the frequency to one data point per second resulted in a decline in performance. Given that larger dataset sizes lead to longer computational times, we opted to investigate the effect of data sampling starting from one data point every 5 s.

Therefore, we examined the impact of varying dataset sizes on computational time to identify an optimal balance between minimizing prediction error and the time required for model training and testing. Specifically, we evaluated the effect of sampling data points at intervals of 5, 10, 30, 60, and 120 s. This leads to the analysis of five different datasets $\{D_i \mid i \in \{5,10,30,60,120\}\}$, for which the following relationship holds:

$$D_{120} \subset D_{60} \subset D_{30} \subset D_{10} \subset D_5 \quad (3)$$

Furthermore, our goal is to develop a model that predicts procedure end times and lay the foundations for an automated tool to inform hospital personnel, optimizing the process of calling the next patient. Therefore, we investigated the prediction errors of the models when 5, 10, 15, and 20 min are left to the end of a procedure. Our aim is to assess the feasibility of applying such a tool in real-life scenarios to reliably detect when a procedure is about to end.

The implementation, and training and testing of the algorithms has been done in Python programming language (Python Software Foundation, https://www.python.org/) with TensorFlow version 2.10.0 (TensorFlow, https://www.tensorflow.org/) and Keras version 2.10.0 (Keras, https://keras.io/). The computations were performed on a system equipped with a NVIDIA GeForce RTX 3090 GPU with 24GB of memory.

3. Results

3.1. Performance analysis

This section investigates the performance of various models in predicting the end time of a procedure. We employed MAE and SMAPE as metrics. The models assessed are InceptionTime, LSTM, LSTM-FCN, Transformer, LSTM-Attention, and an Ensemble approach. A complete list of all the performance metrics can be found in the Appendix. The notation “MAE_n” or “SMAPE_n” in the plot indicates the MAE or SMAPE of the relative model with one datapoint every n seconds.

Figure 2 shows the MAE of the models. Here, we report the metric with respect to the data sampling strategy that ranges from 5 s up to 2 min. InceptionTime and Ensemble models exhibit the lowest error across all prediction windows. The MAE of these models is less than 5 min, regardless of the data sampling. The InceptionTime model, in particular, significantly outperforms other models like LSTM-Attention, which shows the highest MAE values of around 20 min. The LSTM-FCN also performs well, consistently maintaining MAE values lower than 5 min.

The SMAPE metric is depicted in Figure 3. The variability in the errors that can be noticed changing the amount of data points reflect the importance of analyzing such scenarios. InceptionTime, LSTM-FCN, and the Ensemble continue to lead with the lowest percentage

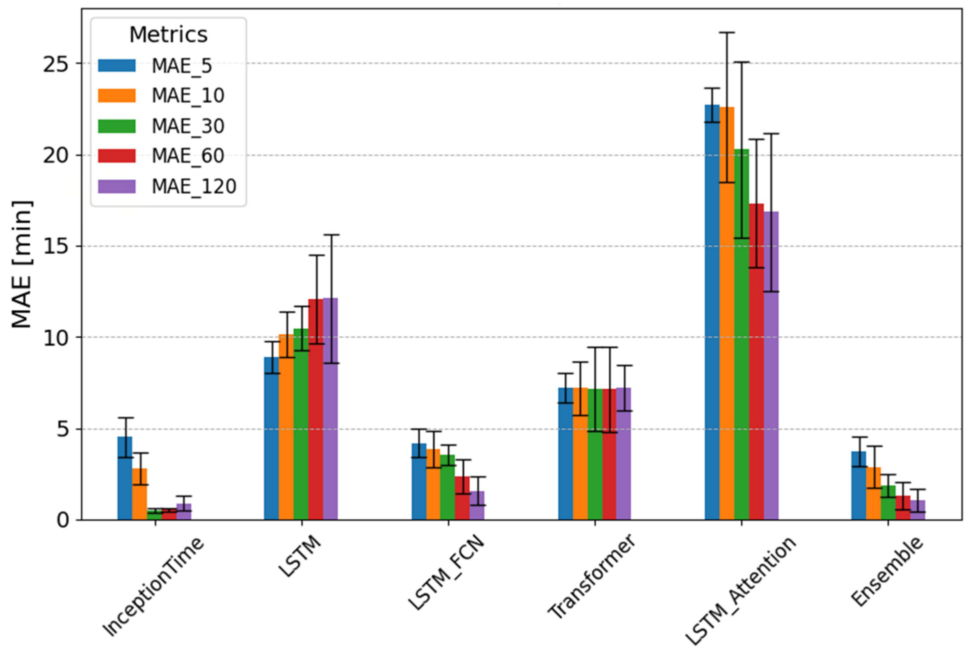

Figure 2. Mean absolute error in the predictions averaged over 100 different trials.

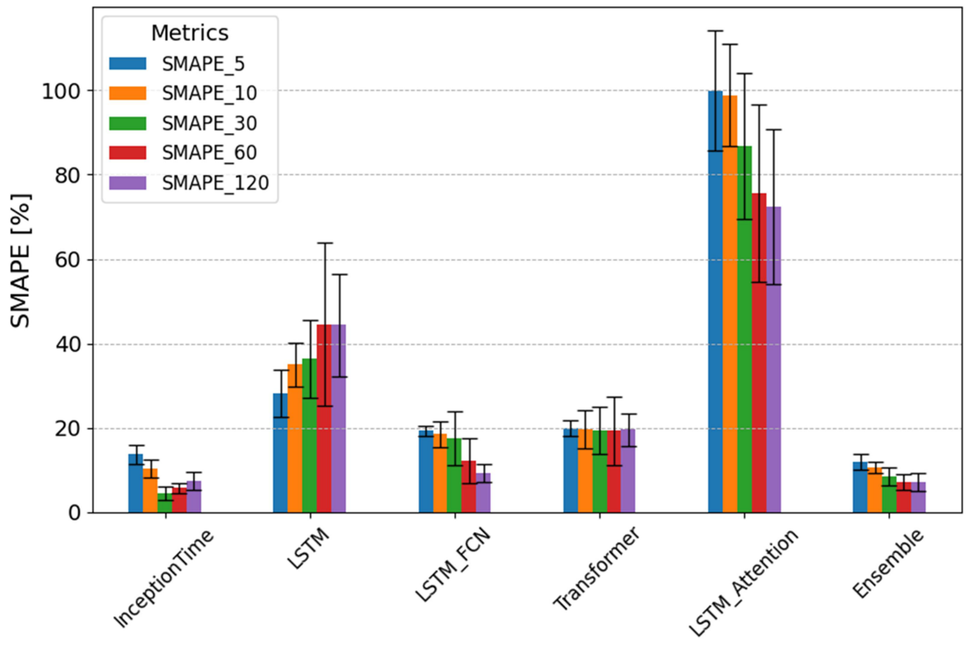

Figure 3. Symmetric mean absolute percentage error in the predictions averaged over 100 different trials.

errors. This indicates optimal performances in both absolute and relative terms. They consistently exhibit error values lesser than 20% across all sampling rates, despite some fluctuations. A slight decrease in SMAPE is observed as the sampling interval widens. In contrast, the SMAPE of the LSTM model increases when the dataset size decreases. The performance of the Transformer model is relatively stable, with a SMAPE of 20%.

The trends are consistent among both metrics. InceptionTime, Ensemble, and LSTM-FCN maintain low error values across all sampling intervals. Conversely, LSTM model exhibits larger increases in SMAPE with wider sampling intervals. The Transformer model is the most consistent in the metrics. The attention mechanism concatenated to the LSTM layer appears to decrease the performance of the models.

An example of the output of a model is presented in Figure 4. Here, we employed LSTM-FCN on a CAG which took 49 min. The performance metrics stand at 1.07 min for MAE, with a SMAPE of 4.91%. It can be observed that prediction accuracy improves as the procedure is near completion. Specifically, predictions tend to underestimate the ETC after the initial 10 min. However, alignment between the model predictions and ground truth significantly improves beyond the first half-hour, achieving nearly perfect correspondence. Moreover, the observed prediction jumps appear to align with phase transitions.

Table 2 shows the performance of all the models with one data point every 60 s. The results are averaged over 100 trials. InceptionTime stands out with an average SMAPE below 6%. Comparable performances can be noticed with LSTM-FCN and Ensemble model, with a SMAPE score of 12.3% and 7.2%, respectively.

3.2. Computational time analysis

The analysis of training and testing times for the models provides critical insights into their practical applicability, especially in real-world scenarios where both training efficiency and inference speed are important. Each model was trained with a specific routine, allowing for a maximum of 1000 epochs and incorporating early stopping criteria and learning rate adjustments based on improvements in SMAPE. The notation 'Training time_n' or 'Testing time_n' in the following plots indicate the training or testing time of the relative model with one datapoint every n seconds.

The training time for each model is presented in Figure 5. The values represent the average training time over onefold of data. InceptionTime shows a progressive increase in training time as the sampling interval widens, with the longest training time observed to be about 23 min at the 120-s interval. LSTM and LSTM-Attention models exhibit significantly shorter training times compared to CNN-based models, with an average training time of around 5 min per fold. LSTM-FCN, despite being a hybrid model combining RNN and CNN features, demonstrates longer training times similar to pure CNN models. The Transformer model shows moderate training times across all sampling intervals. However, the training time increases substantially at longer intervals. The Ensemble model, which aggregates predictions from multiple models, unsurprisingly exhibits the longest training times across all sampling intervals.

The inference time of the models over onefold is depicted in Figure 6. In real-life applications, testing time, or inference speed, is critical. The testing times for all models are much shorter than training times, as expected. All the proposed models take, on average, less than 1 s to be tested on onefold of data. Since onefold consists of 74 procedures, each procedure took less than 0.015 s to be inferred.

InceptionTime maintains low testing times around 0.3 s per fold across all sampling intervals. LSTM and LSTM-Attention models also exhibit similar testing times, with minor variations across different sampling rates. LSTM-FCN, while having longer training times, maintains moderate testing times, indicating that the convolutional layers do not significantly impact inference speed. The Transformer model proved to be the fastest model in inference phase, with an average testing time of around 0.2 s per fold. The Ensemble model, while exhibiting the longest training times, maintains reasonably low testing times across all sampling intervals, with a maximum of around 1 s per fold.

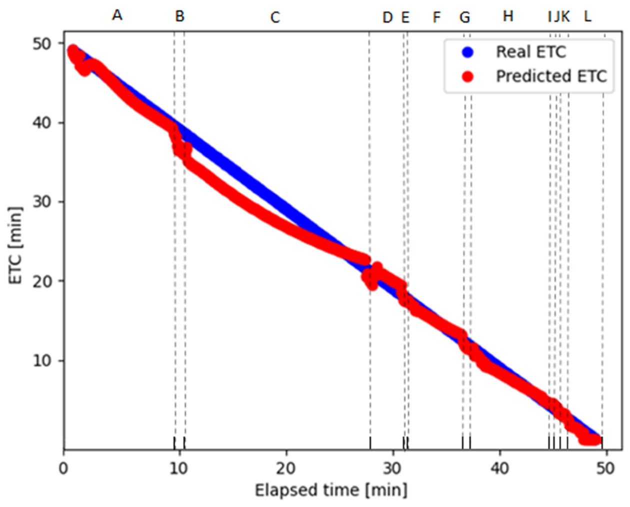

Figure 4. Example of end time prediction of the LSTM-FCN model. The workflow phases are also marked. This procedure was 49 min long and achieved a MAE of 1.07 min and SMAPE of 4.91%.

Table 2. MAE and SMAPE of the models with one data point every 60 s.

Model	MAE_60 [min]	SMAPE_60 [%]
InceptionTime	0.53 ± 0.09	5.8 ± 1.2
LSTM	12.06 ± 2.44	44.5 ± 19.3
LSTM-FCN	2.35 ± 0.92	12.3 ± 5.3
Transformer	7.13 ± 2.31	19.3 ± 8.2
LSTM-Attention	17.33 ± 3.54	75.6 ± 21.1
Ensemble	1.32 ± 0.76	7.2 ± 1.9

3.3. Quantitative error analysis

The best tradeoff between computational cost and performance has been found when sampling one data point every 60 s. Thus, here we focus on the absolute

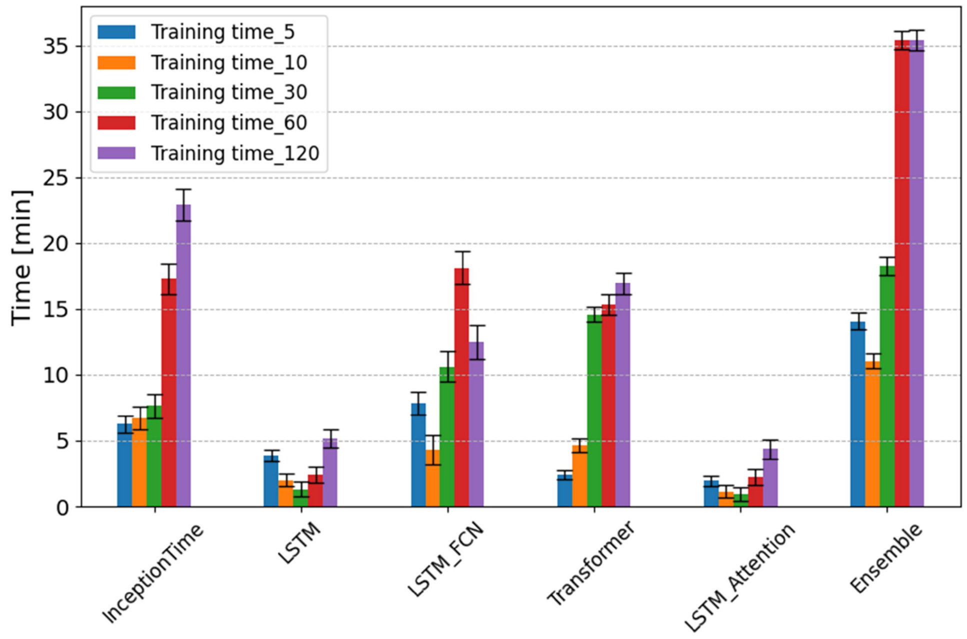

Figure 5. Onefold training time of the models averaged over 100 different trials.

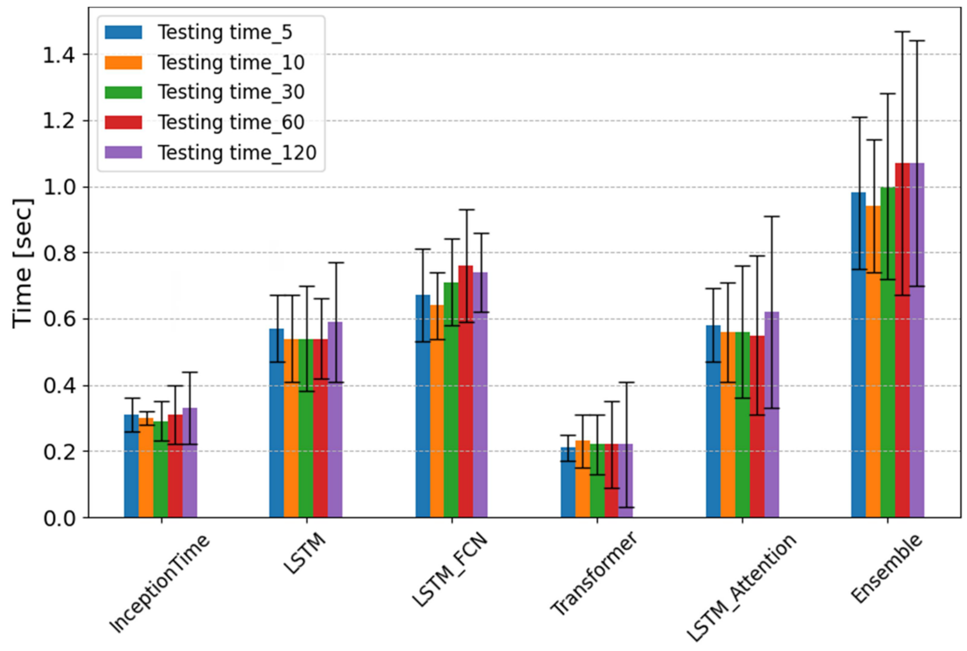

Figure 6. Onefold testing time of the models averaged over 100 different trials.

difference between predicted and actual time to the end of the procedure when 5, 10, 15, and 20 min are left to the end, with a 60-s sampling method. The notation 'Error @n' in the plot indicates the error of the relative model when n minutes are left to the end of the procedure.

Figure 7 illustrates the absolute error of the models. InceptionTime emerges as the standout performer in this analysis, consistently achieving the lowest errors across all prediction intervals. This CNN-based architecture excels in capturing temporal features from sequential data, with errors ranging from 25 s at 20 min to 31 s at 5 min. Similarly, LSTM-FCN shows promising performance (errors from 127 to 76 s), with a trend comparable to the Ensemble method. In contrast, LSTM and LSTM-Attention models showed worse results. LSTM struggles with higher errors, particularly at shorter intervals (692 s when 20 min are left to 207 s when 5 min are left). LSTM-Attention also faces similar issues with errors ranging from 899 to 247 s. The error

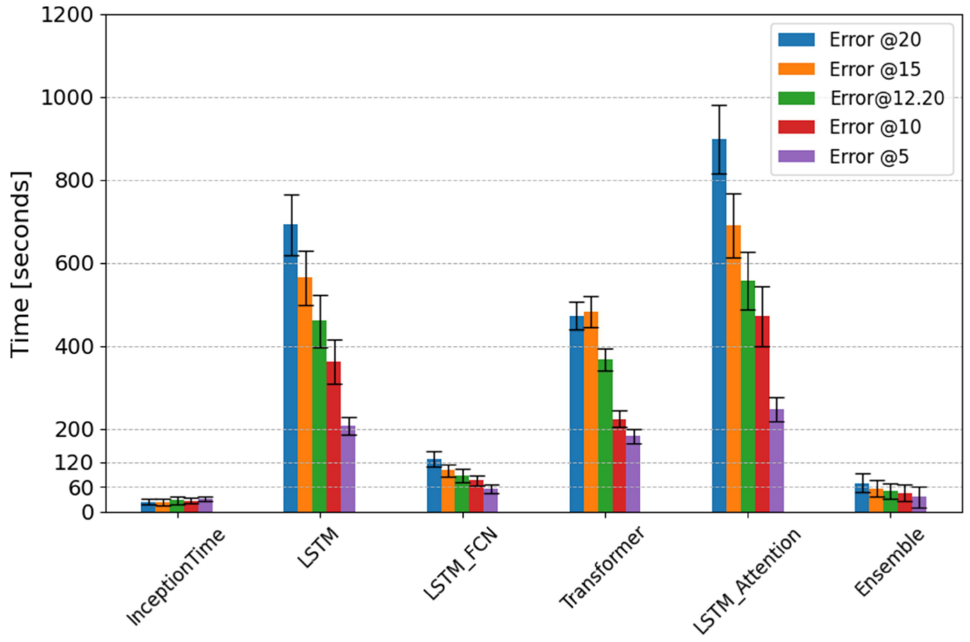

Figure 7. Absolute difference between predicted and actual ETC for all the models when 5 to 20 min are left to the end of the procedure.

of the Transformer model ranges from 482 to 183 s, showcasing moderate proficiency in capturing procedural dynamics.

Overall, all the models proposed seem to decrease the error as the procedure progresses. This finding is relevant and showcases the reliability of the predictions. The models tend to make less mistakes as more data becomes available during a procedure. However, the InceptionTime model exhibits an inverted trend in error rates, as shown in Figure 7. It is important to note that the observed errors remain very low, in the range of a few seconds.

4. Discussion

In this study, we implemented multiple DL models to predict the end time of CAG procedures in the cath lab. The InceptionTime, LSTM-FCN, and Ensemble models consistently exhibit superior performance across both MAE and SMAPE metrics. InceptionTime, in particular, achieves the lowest error rates, with MAE values consistently below 5 min and SMAPE percentages under 6% when sampling one data point every 60 s. The good performance achieved by LSTM-FCN and Ensemble models is reflected in a MAE lower than 5 min. On the other hand, the LSTM-Attention model demonstrates the highest SMAPE and MAE values, suggesting that the addition of an attention mechanism may not always enhance performance, particularly in scenarios requiring the integration of both long-term and short-term dependencies. The standard LSTM model also shows relatively poor performance, with significantly higher error rates than CNN-based models. This indicates the reliance of this model on more granular temporal data to maintain accuracy. The Transformer model, while very stable in performance, exhibits moderate proficiency, with consistent error values around 20%.

All models evaluated in this study are inherently causal, meaning they rely only on past data for predictions. This property is crucial for real-time applications, as it ensures the models make decisions based only on information available at the time, without the need for future data. Furthermore, the analysis of testing times provides crucial insights into the practical applicability of the models, as real-time predictions are essential for integration into clinical workflows. Training times, while important, are less critical for practical use since training is typically performed offline and infrequently. However, the environmental impact of training deep learning models should not be overlooked as long training times contribute to increased energy consumption. While CNN-based models excel in prediction accuracy, this comes with a tradeoff in computational time. InceptionTime, for instance, showed a progressive increase in training time, reaching up to 23 min per fold at the 120-s interval. This is significantly longer compared to the average training times of around 5 min per fold for RNN-based models. However, the increased computational cost is justified by the substantial reduction in prediction errors, making CNN-based models more suitable for applications where accuracy is a priority. LSTM-FCN, despite being a hybrid model, exhibits longer training times similar to pure CNN models, reflecting the additional computational cost of integrating convolutional layers. Nonetheless, its moderate testing times and strong performance metrics make it a robust choice for practical applications. The Transformer model proves to be the fastest in inference, with average testing times around 0.2 s per fold, demonstrating its efficiency in real-time applications. While this speed makes the Transformer particularly well-suited for real-time applications, it is important to note that all evaluated models have inference durations compatible with real-time requirements. The Ensemble model, while exhibiting the longest training times, maintains reasonably low testing times and balances the computational load with predictive accuracy. The total training time of every model seems to increase as the amount of data points decrease. This result is due to the training schedule of the models. On the one hand, less data is available, which means less computations are needed. On the other hand, fewer information is provided. Therefore, the training process takes more time as the algorithms take more iterations to converge to a optimal minimum value of loss function.

The quantitative error analysis further underscores the practical applicability of these models. In this study, we reported a sampling rate of one data point per minute as a tradeoff between computational efficiency and predictive accuracy. Initial experiments showed that this rate maintained consistent performance across all the architectures while significantly reducing the volume of data processed, simplifying the input of the model. While the computational savings may seem minor in isolation, they could become impactful when considering large-scale deployments or multiple models, contributing to resource efficiency and sustainability. LSTM and LSTM-Attention models struggle with higher errors, particularly at shorter intervals. Transformer showed moderate error rates which reflect its stable performance in capturing procedural dynamics. On the other hand, InceptionTime maintains low errors from 25 to 31 s as the procedure progresses. This is a strong indicator of its reliability in real-world settings. Such precision ensures that predictions become increasingly accurate as more data is available. However, we observed a 25% increase in error (from 25 s at a 20-min prediction interval to 31 s at a 5-min interval). This behavior contrasts with other models that show consistent decreases in error as prediction intervals shorten. While this limitation does not significantly impact the overall utility of the model, one possible explanation is that shorter intervals may amplify prediction noise or require the model to capture rapid transitions in workflow phases, which poses additional challenges. The LSTM-FCN model also demonstrated robust performance with moderate computational costs, making it a viable option when balancing accuracy and efficiency. These findings highlight the importance of adopting CNN-based models in practical applications, where the high accuracy and reliability of predictions can significantly impact operational efficiency and outcomes.

It is worth to notice that MAE provides insight into the absolute prediction accuracy. As a result, this metric can be biased by the length of the procedure, with longer procedures potentially skewing the error magnitude. SMAPE, on the other hand, normalizes the error, offering a clearer picture of model performance across varying procedure lengths. However, the consistent performance of the models across both MAE and SMAPE metrics underscores their robustness in predicting procedure durations. Our analysis of the standard deviation revealed an interesting trend: Models exhibit higher standard deviations when the evaluation metrics are higher, and lower standard deviations when the metrics are lower. This pattern suggests that variability in performance tends to align with the absolute error magnitude. Importantly, despite this observed trend, the standard deviation values remain well within an acceptable range across all models, ensuring the reliability of the results. The robustness of the results is indicator of the possibility of integrating such models in a clinical setting. We showed that the end time of a CAG procedure can be predicted with high accuracy. Thus, the models could be employed in the creation of an automated tool which could alert the personnel when the procedure is going to end without the need for any manual intervention. Possible applications could target, for instance, an automated call for the next patient in the room, leading to a more efficient workflow.

In comparison with existing literature, this study shows interesting results in the field of predicting ETC of procedures. Unlike previous efforts primarily focused on surgical environments [5,16], our work specifically addresses the unique operational challenges of the cath lab setting by relying on manually annotated workflow phases that describe the specific characteristics of this setting. Prior studies have predominantly leveraged Machine Learning techniques (SVM) and DL models (Transformer, LSTM) models for ETC predictions in surgical contexts, showing the potential of such approaches [4,15,16]. CNN-based models, particularly InceptionTime, showcases superior performance with MAE values consistently below 5 min and SMAPE under

6%. This marks a substantial improvement over the SVM-based prediction MAE of 14 min [4] and the moderate performance of Transformer models, with SMAPE around 20% [15]. However, the comparison with other studies is limited by differences in datasets and methodologies. For example, Ariel [15] used methods based on the extraction of visual features, while our approach relies on manually annotated workflow descriptions. On the other hand, Meeuwsen et al. [5] employed a laparoscopic dataset of 40 cases of hysterectomy while we derived our input phases from videos showing a comprehensive view of the entire cath lab. Similarly, Guédon [4] used as input the activation pattern of the electrosurgical device measured during 57 laparoscopic cholecystectomies, while our dataset is based only on clinical phase annotations. Additionally, Twinanda [14] validated their results on a collection of 170 bypass videos without any manual annotations, whereas our dataset consists of fully annotated videos. Furthermore, our research emphasizes the practical application of these models in enhancing the operational efficiency of cath labs. The automated tool envisioned in this study, which alerts hospital personnel when a procedure is nearing completion, represents a novel application within this context. We showed that the models we implemented could form the basis of an automated tool, achieving an average error of approximately 30 s in predicting the optimal time to call the next patient.

This study has some limitations. The dataset employed consists of CAGs only, one specific type of cardiac diagnostic procedure. However, in the cath labs multiple types of procedures are performed, such as percutaneous coronary intervention or loop recorder implantation. Therefore, the results would benefit from further validation on other types of operations. While the dataset used is derived from a single center, we recognize that this could impact the generalizability of the model, particularly in settings where the distribution of phases and procedures may differ. In real-world clinical environments, different hospitals or regions may have variations in procedural workflows and operational practices, which could influence phase distributions and, in turn, affect the performance of the model. Furthermore, the DL architectures employed here lack interpretability. Employing more transparent models could enhance the trust of the clinical personnel toward a medical application. While our approach relies on manually annotated workflow descriptions, we recognize that this method is not directly applicable to real operating rooms due to the time and effort required for manual annotation. The model was trained exclusively on sequences starting from the beginning of each procedure, as this approach ensured consistency in the training data and allowed the model to learn patterns comprehensively from the start of the workflow. However, we acknowledge the importance of handling incomplete sequences for practical applications in real-world operating room (OR) settings, where systems might be started late or restarted mid-procedure.

CNN-based models, particularly InceptionTime, outperform RNN-based and attention-based models in predicting the end time of procedures. The architectural design of CNNs allows for effective feature extraction across different scales, which is crucial for time-series predictions. In the presented study, the accuracy can be attributed to the capability of convolutional layers to identify and learn from patterns in the sequential data. In contrast, RNN-based models such as LSTM and LSTM-Attention struggled with higher error rates, particularly with increasing dataset sizes, indicating their dependence on finer temporal granularity and difficulty in maintaining accuracy over longer sequences. The attention mechanism, intended to enhance the performance of the LSTM layer, did not provide improvements, suggesting that the added complexity might not translate to better predictive accuracy.

5. Conclusion and future directions

The findings indicate that while InceptionTime and LSTM-FCN offer high accuracy and robust performance, their higher computational costs necessitate a consideration of the tradeoffs between accuracy and efficiency in practical applications. The Ensemble model, by combining multiple predictions, yields low MAE and SMAPE but at the cost of increased training time. The consistent performance improvements as the procedure progresses underscore the importance of real-time data integration for enhancing predictive accuracy.

Future research could explore optimizing these models for faster training times without sacrificing accuracy, potentially through advanced optimization techniques. This approach could become more resource-efficient without compromising performance by minimizing training time, especially in hospital settings where advanced hardware could be not available. On the other hand, additionally, investigating hybrid models that combine the strengths of different architectures might yield even more accurate and computationally efficient solutions. Furthermore, it could be interesting to investigate how these models could be applied in a real scenario. Future work could target the integration of such DL models in a scheduling scenario, where having reliable and real-time estimates of the duration of a procedure could improve the schedule planning of the cath labs. Additionally, future research should explore automated approaches to workflow annotation, such as using computer vision systems, to enable real-time integration and scalability in clinical environments. Finally, extending this analysis to diverse procedural contexts, such as other types of surgeries performed in the hospital, would enhance the generalizability and applicability of these predictive models. On the ensemble approach, one potential improvement involves dynamically weighting the contributions of the underlying models based on their performance at different stages of the procedure. For example, models like InceptionTime could be given higher weights during earlier stages, while other models better suited for later stages, such as LSTM-based architectures, could contribute more at that point.

In this work, we successfully demonstrated that clinical phases derived from video data can reliably inform deep learning models to predict the end time of CAG procedures in the cath lab with high accuracy. The InceptionTime model, in particular, achieved outstanding performance. MAE values consistently scored below 5 min and SMAPE percentages under 6% when sampling data every 60 s. The model consistently predicts the ETC of a procedure with an average error of approximately 30 s when there are between 20 and 5 min remaining. These findings highlight the efficacy of CNN-based models for ETC analysis.

Disclosure statement

The authors report there are no competing interests to declare.

Data availability

All the code and dataset used in this research is available at: https://github.com/emanuelefrassini/prediction_procedure_duration

Funding

The author(s) reported there is no funding associated with the work featured in this article.

References

[1] WHO. Cardiovascular diseases; 2021. Available from: https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)

[2] Agarwal S, Gallo JJ, Parashar A, et al. Impact of lean six sigma process improvement methodology on cardiac catheterization laboratory efficiency. Cardiovasc Revasc Med. 2016;17(2):95–101. doi: 10.1016/j.carrev.2015.12.011.

[3] van Heuven van Staereling I. a. Stochastic scheduling techniques for integrated optimization of catheterization laboratories and wards. PATAT 2018 - Proceedings of the 12th International Conference on the Practice and Theory of Automated Timetabling; 2018. Vienna: PATAT. p. 313–329.

[4] Guédon ACP, Paalvast M, Meeuwsen FC, et al. It is time to prepare the next patient' real-time prediction of procedure duration in laparoscopic cholecystectomies. J Med Syst. 2016;40(12):271. doi: 10.1007/s10916-016-0631-1.

[5] Meeuwsen FC, van Luyn F, Blikkendaal MD, et al. Surgical phase modelling in minimal invasive surgery. Surg Endosc. 2019;33(5):1426–1432. doi: 10.1007/s00464-018-6417-4.

[6] Miller DD. Machine Intelligence in Cardiovascular Medicine. Cardiol Rev. 2020;28(2):53–64. doi: 10.1097/CRD.0000000000000294.

[7] Anderson RD, Massoomi MR. Efficiency improvements in the catheterization laboratory: it's all about the team. JACC Cardiovasc Interv. 2018;11(4):339–341. doi: 10.1016/j.jcin.2017.10.031.

[8] Karalis VD. The integration of artificial intelligence into clinical practice. Appl Biosci. 2024;3(1):14–44. doi: 10.3390/applbiosci3010002.

[9] Ammori BJ, Larvin M, McMahon MJ. Elective laparoscopic cholecystectomy. Surg Endosc. 2001;15(3):297–300. doi: 10.1007/s004640000247.

[10] Macario A, Dexter F. Estimating the duration of a case when the surgeon has not recently scheduled the procedure at the surgical suite. Anesth Anal. 1999;89(5):1241–1245. doi: 10.1213/00000539-199911000-00030.

[11] Dexter F, Epstein RH, Lee JD, et al. Automatic updating of times remaining in surgical cases using bayesian analysis of historical case duration data and "instant messaging" updates from anesthesia providers. Anesth Analg. 2009;108(3):929–940. doi: 10.1213/ane.0b013e3181921c37.

[12] Maktabi M, Neumuth T. Online time and resource management based on surgical workflow time series analysis. Int J Comput Assist Radiol Surg. 2017;12(2):325–338. doi: 10.1007/s11548-016-1474-4.

[13] Marafioti A. a. CataNet: predicting remaining cataract surgery duration. Medical image computing and computer assisted intervention – MICCAI 2021. Strasbourg: Springer International Publishing; 2021. p. 426–435.

[14] Padoy Na-O. On-line recognition of surgical activity for monitoring in the operating room. Proceedings of the 20th National Conference on Innovative Applications of Artificial Intelligence - Volume 3; 2008. Chicago: AAAI Press. p. 1718–1724.

[15] Twinanda AP, Yengera G, Mutter D, et al. RSDNet: learning to predict remaining surgery duration from laparoscopic videos without manual annotations. IEEE Trans Med Imaging. 2019;38(4):1069–1078. doi: 10.1109/TMI.2018.2878055.

[16] Ariel B. a. Estimated Time to Surgical Procedure Completion: an Exploration of Video Analysis Methods. Medical Image Computing and Computer Assisted Intervention – MICCAI 2023. Berlin, Heidelberg: Springer-Verlag; 2023. p. 165–175.

[17] Hassan IF-A. Deep learning for time series classification: a review. Data Min Knowl Discov. 2019;33(4):917–963.

[18] Hewamalage H, Bergmeir C, Bandara K. Recurrent neural networks for time series forecasting: current status and future directions. Int J Forecast. 2021;37(1):388–427. doi: 10.1016/j.ijforecast.2020.06.008.

[19] Yu W, Kim IY, Mechefske C. Analysis of different RNN autoencoder variants for time series classification and machine prognostics. Mech Syst Sig Process. 2021;149:107322. doi: 10.1016/j.ymssp.2020.107322.

[20] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735–1780. doi: 10.1162/neco.1997.9.8.1735.

[21] Fukushima K. Visual feature extraction by a multilayered network of analog threshold elements. IEEE Trans Syst Sci Cyber. 1969;5(4):322–333. doi: 10.1109/TSSC.1969.300225.

[22] Gamboa JC. Deep learning for time-series analysis. 2017. ArXiv, abs/1701.01887

[23] Ismail Fawaz H, Lucas B, Forestier G, et al. InceptionTime: finding AlexNet for time series classification. Data Min Knowl Disc. 2020;34(6):1936–1962. doi: 10.1007/s10618-020-00710-y.

[24] Szegedy C. a. IEEE Conference on Computer Vision and Pattern Recognition (CVPR); Going deeper with convolutions, Boston, MA, USA; 2015. p. 1–9.

[25] Karim F, Majumdar S, Darabi H, et al. Multivariate LSTM-FCNs for time series classification. Neural Netw. 2019;116:237–245. doi: 10.1016/j.neunet.2019.04.014.

[26] Wu JH. Conference on Computer Vision and Pattern Recognition; 2019. Squeeze-and-Excitation Networks, Salt Lake City, Utah.

[27] Vaswani A. a. Advances in Neural Information Processing Systems; 2017. Attention Is All You Need. Curran Associates, Inc. (30)

[28] Choi SR, Lee M. Transformer architecture and attention mechanisms in genome data analysis: A comprehensive review. Biology (Basel). 2023;12(7):1033. doi: 10.3390/biology12071033.

[29] Luong T. a. Effective approaches to attention-based neural machine translation. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing; 2015. p. 1412–1421. Association for Computational Linguistics. doi: 10.18653/v1/D15-1166.

[30] Bridle JS. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. Proceedings of the 2nd International Conference on Neural Information Processing Systems; Morgan-Kaufmann; 1989. p. 211–217.

[31] Shih S-Y, Sun F-K, Lee H-y Temporal pattern attention for multivariate time series forecasting. Mach Learn. 2019; 108(8-9):1421–1441. doi: 10.1007/s10994-019-05815-0.

[32] Landassuri-Moreno VM. Neural network ensembles for time series forecasting. Proceedings of the 11th Annual. Conference on Genetic and Evolutionary Computation; 2009; New York, NY, US: Association for Computing Machinery. p. 1235–1242. doi: 10.1145/1569901.1570067.

[33] Armstrong J, Collopy F. Error measures for generalizing about forecasting methods: empirical comparisons. Int J Forecast. 1992;8(1):69–80. doi: 10.1016/0169-2070(92)90008-W.

[34] Ba DP. Adam: A Method for Stochastic Optimization. *3rd International Conference for Learning Representations*; San Diego, 2015. Ithaca, NY: ArXiv.

[35] Rivoir, D. a. (2024). On the pitfalls of Batch Normalization for end-to-end video learning: A study on surgical workflow analysis. Medical Image Analysis, 103126.

Appendix

Table 3. Performance one data point every 5 s.

Model	MAE_5 [min]	SMAPE_5 [%]
InceptionTime	4.53 ± 1.1	13.73 ± 2.35
LSTM	8.91 ± 0.88	28.19 ± 5.62
LSTM-FCN	4.20 ± 0.80	19.40 ± 1.24
Transformer	7.23 ± 0.79	19.98 ± 1.82
LSTM-Attention	22.71 ± 0.95	99.87 ± 14.2
Ensemble	3.75 ± 0.82	11.98 ± 1.91

Table 4. Performance one data point every 10 s.

Model	MAE_10 [min]	SMAPE_10 [%]
InceptionTime	2.79 ± 0.87	10.45 ± 2.08
LSTM	10.17 ± 1.24	35.02 ± 5.12
LSTM-FCN	3.85 ± 0.98	18.53 ± 3.11
Transformer	7.19 ± 1.46	19.61 ± 4.53
LSTM-Attention	22.57 ± 4.11	98.79 ± 12.1
Ensemble	2.89 ± 1.13	10.65 ± 1.32

Table 5. Performance one data point every 30 s.

Model	MAE_30 [min]	SMAPE_30 [%]
InceptionTime	0.48 ± 0.12	4.60 ± 1.56
LSTM	10.48 ± 1.23	36.35 ± 9.14
LSTM-FCN	3.54 ± 0.57	17.63 ± 6.32
Transformer	7.13 ± 2.33	19.4 ± 5.49
LSTM-Attention	20.27 ± 4.82	86.80 ± 17.32
Ensemble	1.86 ± 0.62	8.42 ± 2.12

Table 6. Performance one data point every 120 s.

Model	MAE_120 [min]	SMAPE_120 [%]
InceptionTime	0.89 ± 0.42	7.37 ± 2.1
LSTM	12.11 ± 3.54	44.41 ± 12.1
LSTM-FCN	1.58 ± 0.76	9.33 ± 2.14
Transformer	7.23 ± 1.25	19.60 ± 3.84
LSTM-Attention	16.85 ± 4.32	72.42 ± 18.34
Ensemble	1.07 ± 0.64	7.21 ± 2.13

Table 7. Error at different time points.

Model	Error @20 [s]	Error @15 [s]	Error @10 [s]	Error @5 [s]
InceptionTime	25 ± 6	24 ± 7	27 ± 7	31 ± 5
LSTM	692 ± 73	564 ± 65	363 ± 54	207 ± 21
LSTM-FCN	127 ± 19	100 ± 15	76 ± 12	56 ± 11
Transformer	473 ± 34	482 ± 38	225 ± 21	183 ± 18
LSTM-Attention	899 ± 82	691 ± 77	471 ± 72	247 ± 29
Ensemble	70 ± 23	57 ± 21	46 ± 20	36 ± 26