Early failure prediction during robotic assembly using Transformers

Author Names Omitted for Anonymous Review. Paper-ID [add your ID here]

***Abstract*—Peg-in-hole assembly of tightly fitting parts often requires multiple attempts. Parts need to be put together by performing a wiggling motion of undetermined length and can get stuck, requiring a restart. Recognizing unsuccessful insertion attempts early can help in reducing the *makespan* of the assembly. This can be achieved by analyzing time-series data from force and torque measurements. We describe a transformer neural network model that is three times faster, i.e. requiring much shorter time series, for predicting failure than a dilated fully convolutional neural network. Albeit the transformer provides predictions with higher confidence, it does so at reduced accuracy. Yet, being able to call unsuccessful attempts early, makespan can be reduced by almost 40% which we show using a dataset with force-torque data from 241 peg-in-hole assembly runs with known outcomes.**

I. Introduction

Many manipulation tasks often require multiple attempts for robots and humans alike. Here, a new "attempt" might begin with a re-grasp action to get a better handle on an object, finding the correct pose for the object, such as when inserting a screwdriver into a screw, or restarting an activity as its initial conditions have been misjudged, such as during parallel parking. Watson and Correll [9] argue that the trial-and-error approach will remain a persistent feature of autonomy, as one-shot accuracy requires not only perfect perception and information of the environment but also perfect prediction of the dynamics of the physical world, which are intrinsically uncertain. They propose a deep-learning approach to the early detection of failure of a robotic system performing a peg-in-hole task with tight tolerances (Figure 1). This approach is based on dilated Fully Convolutional Networks (FCN) [4], which are trained to perform a binary classification task: given the Force-Torque data of the previous N time steps, make a prediction of the probability of failure (p_f) or success (p_s), i.e. classify if the robotic system is going to either fail or succeed at the task. A Markov Decision Process (MDP) is then used to make a prediction on the outcome using the computed probabilities of failure and success. By preempting trials that are likely to fail early, the overall *makespan* is reduced. The makespan is related to both the confusion matrix of the classifier and the amount of data it takes to make a classification. If the predictor is often wrong, tasks get preempted unnecessarily, thereby increasing the makespan. Similarly, the earlier a predictor is able to provide its judgment, the shorter the makespan can be.

This paper evaluates Transformer neural networks [8] that use attention models to learn salient information in time series data and studies their impact on makespan. Using machine learning has a long history in predicting robotic failure. Terra and Tinós [7] compare different artificial neural networks (ANN) for fault detection in manipulation. Cho et al. [3] propose an ANN to detect failure in a 2-degrees of freedom (DOF) manipulator. Due to the temporal nature of the task, Recurrent Neural Networks (RNNs) became the go-to architecture for time series analysis, which makes them suitable for failure detection using the stream of data generated by a robot's sensors [5]. With the success of the Transformer architecture for the analysis of natural language, Transformers have also successfully been used for time series analysis [10]. In particular, Liu et al. [6] introduce a two-tower Transformer architecture [6], where one of the towers performs step-wise attention while the other one performs channel-wise attention. The outputs of both towers are then concatenated through a gating layer which weighs the outputs of each tower. The Gated Transformer Network (GTN) outperforms other benchmark models in 7 out of 13 multivariate time series datasets.

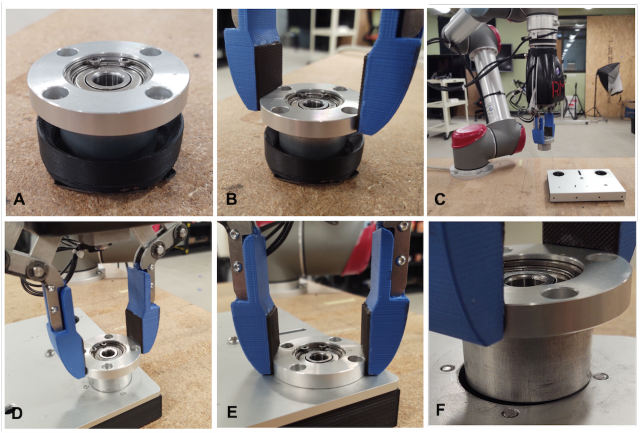

Fig. 1: A complete peg-in-hole assembly sequence: **A** The bearing is presented in a 3D- printed jig, **B** The bearing is picked up by the robot and transported to the assembly plate **C**. Force and torque measurements are used to **D** locate the hole **E** and complete insertion. Insertion failure due to misalignment **F**. Friction with the edge of the hole has caused the twisting action to pull the bearing further from the hole center.

In this paper, we are using a vanilla transformer implementation and show by drawing from a set of 241 timeseries from real robot experiments using a Universal Robot UR5, a Robotic Materials smart hand, and an Optoforce 6-axis force/torque (F/T) sensor that a transformer architecture is able

to make predictions significantly faster, thereby reducing the makespan by up to 40%.

II. Assembly makespan

As assembly is a task that requires a series of physical interactions with undetermined length and probalistic outcome ("trial-and-error"), the *makespan*, i.e. the time a task needs to complete is an important metric. Watson and Correll [9] propose and experimentally validate an analytical expression for the makespan t_{run} that is based on task timing statistics, failure rates, as well as the confusion matrix of a predictor that can be used for preempting a trial that is likely to fail:

$$t_{\text{Run}} = \frac{1 + \text{MTF}(\text{P}_{\text{FP}} + \text{P}_{\text{NCF}}) + \text{MTS}(\text{P}_{\text{TP}} + \text{P}_{\text{NCS}}) + \text{MTN}(\text{P}_{\text{FP}} + \text{P}_{\text{TN}})}{1 - \text{P}_{\text{FN}} - \text{P}_{\text{FP}} - \text{P}_{\text{TN}}} \tag{1}$$

Here, *Mean Time to Failure/Success* (MTF/MTS) are measured during the experiment and represent the average time length of failing episodes and average time length of succeeding episodes, respectively. *Mean Time to Negative/Positive* (MTN/MTP) represent the average times it takes the model to generate a negative (failing) or positive (succeeding) prediction. The probabilities P_{FP} (false positive), P_{TP} (true positive), P_{FN} (false negative), and P_{TN} (true negative) are obtained from the confusion matrix of the predictor. Finally, P_{NCF} and P_{NCS} represent the probabilities of non-classified failure and non-classified success respectively, i.e. the probability that, given the model prediction is a no-classification (NC), the episode is a failure or success. Eq. 1 shows that t_{run} is proportional to the time needed for classification (MTN/MTP) and the probabilities of false negatives and positives. We are therefore interested in early failure detectors that provide a favorable combination of these parameters that further reduce makespan.

III. Dilated FCN and Transformer models

We compare the transformer network to the dilated FCN, which has outperformed other RNN architectures in [9]. In order to provide a fair comparison, both architectures have a similar number of parameters.

a) Dilated FCN: The dilated FCN from [9] consists of two dilated convolutional layers followed by a max-pooling, flattening, and dense layers. The input layer is a 350×6 matrix which represents a window with 6 features ($F_{x,y,z}$ and $T_{x,y,z}$) and 350 time-steps (a window of 7 seconds, at 20ms intervals). The model consists of two convolutional layers with 16 filters, kernel size 4 and ReLU activation. The first layer uses a dilation rate of 4, the second layer a dilation rate of 8. Convolution is followed by max-pooling with size 4, and two dense layers with 50, and 2 units, respectively. The network has 63992 trainable parameters.

b) Transformer: The Transformer architecture used for this study consists of the encoder tower from the vanilla Transformer proposed in [8] and is shown in Figure 2 with parameters in Table I. The input to this model is the same as for the FCN, windows of 350 time-steps with 6 features

Multi-Head Attention	`head_size`	128	
	`num_heads`	4	
	`dropout`	0.2	
Feed Forward	Convolutional layer 1	`filters`	256
		`kernel_size`	1
		`activation`	ReLU
	Convolutional layer 2	`filters`	6 (d_{model})
		`kernel_size`	1
		`activation`	None
	`dropout`	0.4	
Linear out	Dense layer 1	`units`	128
		`activation`	ReLU
	Dense layer 2	`units`	2
		`activation`	Softmax
	`dropout`	0.4	
Parameters	68808		

TABLE I: Layer parameters of each Transformer block

at each step. We use a fixed positional encoding based on sinusoidal waves:

$$\begin{aligned} PE_{(pos,2i)} &= sin(pos/10000^{2i/d_{\text{model}}}) \\ PE_{(pos,2i+1)} &= cos(pos/10000^{2i/d_{\text{model}}}) \end{aligned} \tag{2}$$

where *pos* is the position of the token and i is the dimension (i.e. a different wave will be used to encode each dimension). We also use the standard Multi-Head Attention mechanism described in [8] defined by

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3}$$

IV. Assembly data

a) Dataset: We are using experimental data from [9], which is available online. Data was obtained using 241 real-robot experiments ("trials") as shown in Figure 1. Each trial has a varying number of timesteps and consists of the following data

- ***Time stamp***
- ***Force in X, Y, Z axis:*** numerical variables recording the force in Newtons (F_x, F_y, F_z)
- ***Torque in X, Y, Z axis:*** numerical variables recording the torque in Newton-meters (T_x, T_y, T_z)
- ***Ground truth:*** binary variable with the true outcome of the trial (1 = success and 0 = failure)

In the data set 49.79% of the trials represent successes while the remaining 50.21% represent failures.

b) Preprocessing: We perform a series of pre-processing steps, including standardization, truncation, window creation and class balancing. First, we take the full raw episodes and scale the Force-Torque variables with a Robust Scaler. We do this in order to prevent outliers from adding noise to the distribution of the data and therefore generating an incorrectly scaled dataset. The scaling is performed variable-wise, i.e.

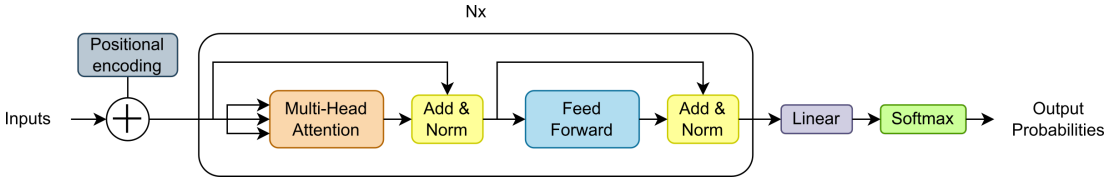

Fig. 2: Transformer architecture

for each axis x, y, z we take the corresponding Force-Torque values and scale only that variable (which means that for each episode, we scale $F_x, F_y, F_z, T_x, T_y, T_z$ individually).

Once data is scaled properly, we truncate the episodes by finding what we will call the *Episode beginning*. This is the first step in the trial in which the value of F_z exceeds a certain threshold and we do this to find the first point in time in which the robotic arm is exerting downward force (i.e. the robotic arm started inserting the bearing into the hole). We do this because data prior to the *Episode beginning*, i.e. before contact is made, will not really be useful for the prediction of the success/failure of the episode.

Instead of feeding the full episodes to the model, we take a rolling window approach This introduces a class imbalance with only 27.49% successful windows and the remaining 72.51% failing windows.

Class imbalance is a well-known issue in Machine Learning classification tasks [1]. We counter this problem with the Synthetic Minority Oversampling Technique [2], which allows us to generate a perfectly balanced dataset using an under-sample policy (i.e. we discard data points from the majority class until we have the same amount of negative and positive samples).

V. Results

We train both models for a maximum of 200 epochs. To prevent over-fitting, training is stopped early when we see no improvement in the validation loss metric within a span of 10 epochs. The FCN runs for about 45 epochs before being stopped, reaching a training accuracy of around 85% and a validation accuracy of around 80%. The Transformer runs for around 35 epochs. It reaches a training accuracy of over 95% while the validation accuracy peaks at 84% (Figures 3 and 4).

We can now use the trained models to predict failure window by window on the testing samples until we get a prediction or the episode ends, and compare the output with the ground truth of the episode. Results are shown in Table II.

	NC	Negative	Positive	NC	Negative	Positive
False	37.5%	0.82	0.0	0.0%	0.76	0.29
True		0.18	1		0.24	0.71

TABLE II: Confusion matrices for FCN (left) and Transformer (right). Albeit the FCN has higher accuracy, it is unable to reach a conclusion (NC) in 37.5% of the cases.

We observe that the FCN has a higher rate of both True Positives (TP) and True Negatives (TN), but it is worth noting

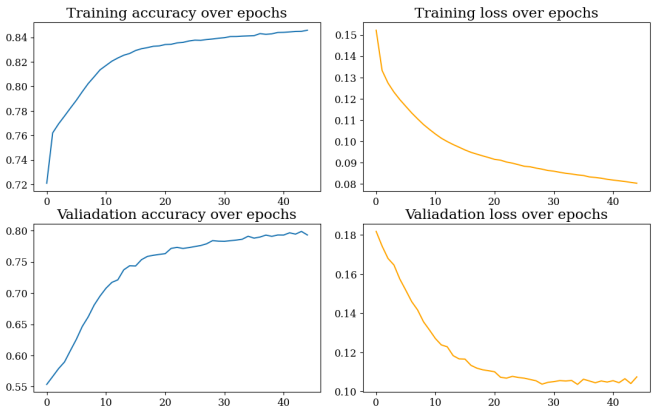

Fig. 3: Training progression of the FCN

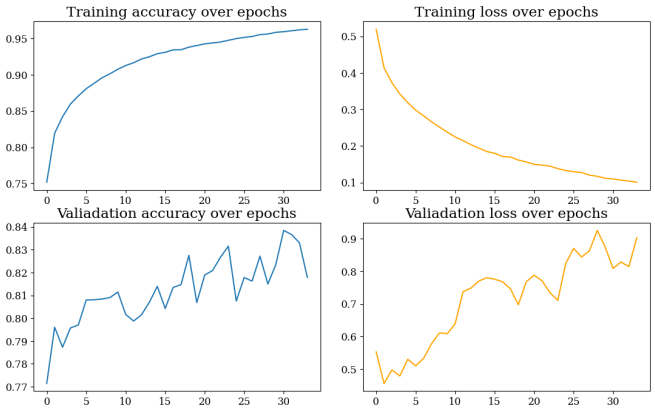

Fig. 4: Training progression of the Transformer

that it is unable to classify 37.5% of the episodes in the testing sample, while the Transformer has a no classification (NC) rate of 0%.

In order to further validate the models, we simulate assembly sequences by drawing from labeled episodes following the MDP shown in Figure 5. For this, we keep drawing time series at random from our dataset. Each time series is classified using a rolling window approach while keeping track of the makespan. A time series is preempted if the classifier predicts failure. The simulation ends once we draw a successful assembly sequence that is not preempted due to a false-negative. Figure 6 shows a histogram with the run-times of each iteration for a non-preempted approach ("reactive") as a baseline, as well as preemption policies using the FCN and Transformer models for $N = 150$ complete

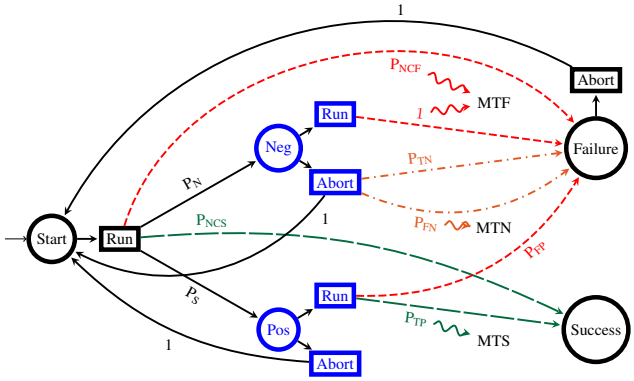

Fig. 5: Markov Decision Process for the Preemptive Scenario. Green, long-dashed paths represent time cost MTS. Red, short-dashed paths represent time cost MTF. Orange, dot-dash paths represent time cost MTN.

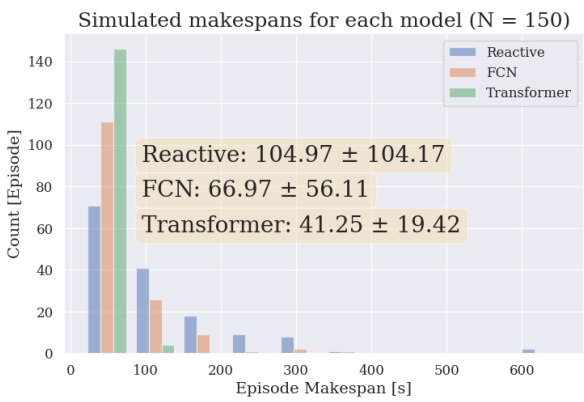

Fig. 6: Histogram showing the run-time distribution by each model through a simulation with $N = 150$

assembly sequences. The distribution of the makespan for the Transformer has a significantly lower mean. It achieves a decrease of around 38% when compared to the FCN and a decrease of around 60% when compared to the Reactive policy (Mann-Whitney U test with 5% significance level, p-value of 6.4×10^{-8}).

VI. Discussion and Conclusion

The transformer models are significantly faster in predicting a negative outcome, while having a much higher confidence than the dilated FCN classifier, expressed by their low no-classification rate. These properties lead to a reduction of makespan compared to previous methods by around 38% in the assembly simulation, which we predict by sampling from real robot assembly experiments for which the outcome is known and using both predictors to cut failing trials short. High confidence is traded with lesser accuracy, in particular for "false positives". This should not be confused with the false positive rate of the actual assembly experiment. It is simply the predictor who will not preempt a failing attempt, which is then treated by the robot as if no predictor and preemption mechanism were in place. Due to the promising nature of transformers for time series analysis, we are interested to extend this framework by additional dimensions, including image and range data, in future work.

References

[1] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural networks*, 106: 249–259, 2018.

[2] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[3] Chang Nho Cho, Ji Tae Hong, and Hong Ju Kim. Neural network based adaptive actuator fault detection algorithm for robot manipulators. *Journal of Intelligent & Robotic Systems*, 95:137–147, 2019.

[4] Pranav Khanna and Apurva Narayan. Light weight dilated cnn for time series classification and prediction. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2179–2183. IEEE, 2020.

[5] Shiwei Li, Yongping Zhao, and Mingli Ding. Mobile robot motor bearing fault detection and classification on discrete wavelet transform and lstm network. *Journal of Mechanics in Medicine and Biology*, 18(08):1840034, 2018.

[6] Minghao Liu, Shengqi Ren, Siyuan Ma, Jiahui Jiao, Yizhou Chen, Zhiguang Wang, and Wei Song. Gated transformer networks for multivariate time series classification. *arXiv preprint arXiv:2103.14438*, 2021.

[7] Marco Henrique Terra and Renato Tinós. Fault detection and isolation in robotic manipulators via neural networks: A comparison among three architectures for residual analysis. *Journal of Robotic Systems*, 18(7):357–374, 2001.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[9] James Watson and Nikolaus Correll. Optimal decision making in robotic assembly and other trial-and-error tasks. *arXiv preprint arXiv:2301.10846*, 2023.

[10] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*, 2022.