

Automated Quality Assurance for Hand-held Tools via Embedded Classification and AutoML

Christoffer Löffler^{1,2}(✉), Christian Nickel², Christopher Sobel², Daniel Dzibela²,
Jonathan Braat², Benjamin Gruhler², Philipp Woller², Nicolas Witt², and
Christopher Mutschler^{2,3}
`{firstname.lastname}@iis.fraunhofer.de`

¹ Friedrich-Alexander University Erlangen-Nuremberg (FAU), Erlangen, Germany

² Precise Positioning and Analytics Dept., Fraunhofer IIS, Nuremberg, Germany

³ Dept. of Statistics, Ludwig-Maximilians-University (LMU), Munich, Germany

Abstract. Despite the ongoing automation of modern production processes manual labor continues to be necessary due to its flexibility and ease of deployment. Automated processes assure quality and traceability, yet manual labor introduces gaps into the quality assurance process. This is not only undesirable but even intolerable in many cases.

We introduce a process monitoring system that uses inertial, magnetic field and audio sensors that we attach as add-ons to hand-held tools. The sensor data is analyzed via embedded classification algorithms and our system directly provides feedback to workers during the execution of work processes. We outline the special requirements caused by vastly different tools and show how to automatically train and deploy new ML models.

1 Introduction

In modern industrial assembly processes all vitally important tasks are already highly automated and hence provide quality assurance by design. To satisfy strict quality requirements, expensive automation for high quantity production steps is standard. However, for many other tasks (such as fixing faceplates or other embellishments) this expensive automation is not worth its cost. Here, highly flexible human labor is much more efficient. Human workers can quickly adapt to new tasks, using different tools, and only need an initial instruction.

This human versatility to new tasks and tools is both the strong point of manual labor but also its deficit. Humans rarely function perfectly according to a task definition, and thus, the quality of the produced items cannot be guaranteed to a certain standard as it can be for an automated assembly. To overcome this quality issue, there are expensive solutions using a four-eye principle of human error detection, i.e., acceptance tests, or even automated camera-based approaches. Such mechanisms are expensive, error-prone, and add testing delays.

In the ideal case we track manual processes that use hand-held (power) tools, e.g., electric screwdrivers, and directly detect whether a task was performed according to its specification. While this allows an automated quality assessment of human labor existing smart tools are expensive and limited in their variety.

For this to work, the assessment of manual work requires a reliable classification of the performed tasks. However, these processes are often not known in advance. Workers may use wildly different tools (e.g. hammers, electric screwdrivers, etc.) and even configurations of such tools may differ. The tasks can be as diverse as fixing parts on a stationary workbench or of a car’s underbody in an assembly line. Finally, the working environment can differ, e.g., a metal hut environment influences the sensors differently from a saw mill.

Attaching external sensors to hand-held tools and using machine learning for classification enables the detection of high-quality process parameters [1]. Our proposed Tool Tracking system (<https://www.iis.fraunhofer.de/tooltracking>) uses an ML-based approach to fill the gap in quality assurance. It can be adapted to different tools and work processes, and can hence be used in combination with existing tools. We use inertial sensors, i.e., accelerometers and gyroscopes, a magnetic field sensor, and a microphone to assess the process. Our AutoML pipeline supports all steps starting from data collection for new tasks, over automatic training up to the deployment on the target sensor module.

2 System Architecture

The sensor module is attached to the current hand-held tool via an easily removable fixture that can be used on many tools. If required by a task an optional (optical or radio-based) positioning system can be added to the system. The sensor module and the user interface are wirelessly connected to a back-end server, hosting a service architecture. The worker interacts with our system via web browser, e.g., using a tablet computer.

Inferencing. Every sensor module analyzes the raw data directly, using efficient C/C++ implementations of the feature extraction and models, and only transmits relevant results. It is unaware of any work process logic, and solely detects and classifies actions in a more power-efficient two-step classification approach. A **stage-1** model continuously analyzes the sensor stream and detects the presence of *actions* with very low error rate and power consumption. A **stage-2** model is triggered by **stage-1** and classifies action intervals into distinct

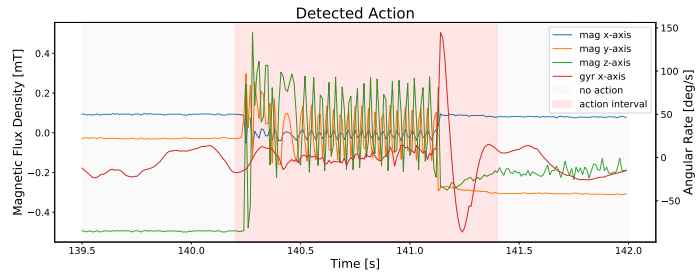


Fig. 1. The electric screwdriver’s magnetic field indicates activity, and the gyroscope’s *x-axis* spike that the torque is reached (*tightening* action highlighted).

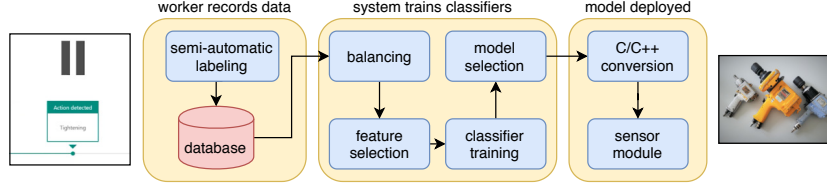


Fig. 2. The AutoML pipeline for training new classifier models.

classes, such as *tightening* or *riveting*. An example of raw data with action detection is given in Fig. 1. The back-end then uses a state machine to verify the work process logic, e.g., whether the actions were performed in the correct order, within the required pitch or roll angle interval, and given a positioning system, even at the correct position at a work piece.

Teaching. The users can define new work processes, e.g., sequences of *tightening*, that rely on previously trained models by recording demonstrations of successive steps, optionally enriching the state machine with position constraints. Moreover, they can teach new tools to the system that, in contrast to new processes, require the automatic training of new classifiers via our AutoML pipeline, see Fig. 2. First, an existing **stage-1** model labels new data with user-defined class labels. Next, we either extract features (similar to **tsfuse** [3]) and deploy the trained **scikit-learn** pipelines [4] to the embedded module (inspired by **sklearn-porter**¹), or we optimize deep learning models (e.g. fully convolutional networks (FCN), temporal convolutional networks (TCN), LSTMs, etc.) with Optuna [5]. Our C++ library receives the configuration and instantiates it.

Features. The AutoML pipeline includes a set of 32 statistical but also advanced features (for instance, we adapt **libxtract** [2]). We additionally create features encoding the temporal occurrences of the features within an action interval, e.g., a spike in rotation towards the end of the interval indicates reaching the torque of *tightening*. In total there are approx. 1,300 unique features that we then select by sensor type to retain diversity, e.g., based on Random Forest feature importance, and filter using the Pearson correlation test to eliminate strongly correlated features. From the remaining features for all sensors, the framework selects the best combinations via recursive feature elimination and trains the classifier. Note that the features are highly tool-dependent, e.g., electric tools generate strong magnetic fields, pneumatic tools produce loud hissing noises.

Experimental results. Actions from electric screwdrivers, such as e.g., *un-tightening*, are detected with 99% accuracy (trained on 20 minutes of data) with F1-scores of 93% for decision trees, and 96%, 98% for 94% for FCN, TCN or LSTM. We also detect short pneumatic rivet gun actions with error rates of 2 – 4% using just 850 samples of 200 ms. However, pneumatic screwdrivers are more difficult to classify: on 2,130 samples **stage-1** achieves 91 – 94% accuracy and **stage-2** only 73 – 80% F1-score.

¹ <https://github.com/nok/sklearn-porter>

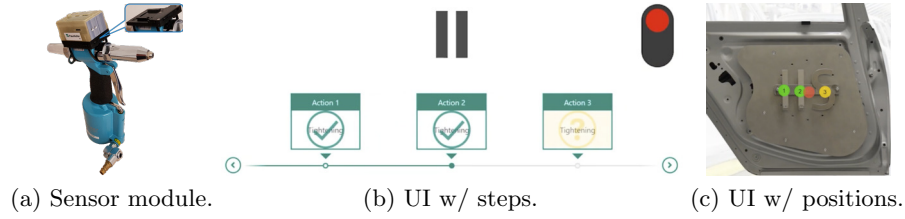


Fig. 3. Hardware used for the demonstration.

3 About the Demonstration

We demonstrate our system with two different hand-held tools, i.e., electric screwdriver and pneumatic rivet gun, see Fig. 3a. We record and execute new work processes, but rely on previously trained models, and detect successive un-/tightening or riveting actions on a work piece (Fig. 3c). The data is processed on the back-end server and visualized on a tablet computer (Fig. 3b). The **stage-1** classifier operates on 200 ms windows with 100 Hz sampling for inertial sensors, 150 Hz for the magnetic field sensor, and 8 kHz for the microphone. **Stage-2** operates on the whole action interval, and uses a more complex model. A demo video is available at <https://youtu.be/MCpC2i-ZbDs>.

Acknowledgements

This work was supported by the Bavarian Ministry of Economic Affairs, Infrastructure, Energy and Technology as part of the Bavarian project Leistungszentrum Elektroniksysteme (LZE) and through the Center for Analytics-Data-Applications (ADA-Center) within the framework of “BAYERN DIGITAL II”.

References

1. Dörr, M., Ries, M., Gwosch, T., Matthiesen, S.: Recognizing product application based on integrated consumer grade sensors: A case study with handheld power tools. In: *Procedia CIRP*, vol. 84, pp 798–803 (2019)
2. Bullock, J.: Libxtract: a Lightweight Library for audio Feature Extraction. In: *Intl. Computer Music Conf*, Santiago, Chile (2007)
3. De Brabandere, A., Robberechts, P., De Beéck, T., and Davis, J.: Automating Feature Construction for Multi-View Time Series Data. In: *ECML-PKDD Workshop on Automating Data Science*, pp. 1–16, Würzburg, Germany (2019)
4. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine Learning in Python. In: *J. Machine Learning Research*, vol. 12, pp. 2825–2830 (2011)
5. Akiba, T., Sano, S. and Yanase, T., Ohta, T. and Koyama, M.: Optuna: A Next-generation Hyperparameter Optimization Framework. In: *Proc. KDD* (2019)