2025 IEEE/SICE International Symposium on System Integration (SII)
January 21-24, 2025, Munich, Germany

Application of OSR for hardware accelerated intelligent manufacturing machines

Adwait RAWAT[†], Takayuki MUKAEDA[†‡], Keisuke SHIMA[†]

Abstract— **This study explores the application of open-set recognition (OSR) in intelligent manufacturing. The proposed method focuses on the integration of hardware-accelerated systems in manufacturing machines as they become more complex and versatile, because the management of both and known and unknown conditions is critical for better reliability and increased productivity. The proposed OSR method provides a novel hardware-accelerated system that enables better performance and reduces downtime in response to novel conditions.**

I. Introduction

In the domain of machine learning, supervised learning has rapidly advanced, demonstrating quick and accurate classification in a closed-set world. A closed-set world refers scenarios where training and test samples that share the same label space. This method is effective when all inputs to a system are known and can be clearly labeled. However, in many real-world cases, training models for all possible events and scenarios is impractical. Although not all possible events can be used for training, anomalies can occur in real-world scenarios. When events fall outside the training data, the machine learning system loses reliability. Consequently, supervised learning becomes less reliable when unknown conditions or label spaces appear. Open-set recognition (OSR) has gained relevance in addressing such challenging and realistic settings. OSR is a recognition system designed to process data outside the expected input domain, making it more resilient to unexpected inputs.

Owing to the robustness of OSR in handling unexpected information, it has emerged as a promising method for anomaly detection in the medical field [1] and the manufacturing industry [2]. In Industry 4.0, OSR can be applied to create intelligent manufacturing machines that can improve the quality of the manufactured parts or the operation status of machines to achieve higher productivity through tool optimization or fault detection, thereby resulting in higher overall robustness against failure. One such example of a manufacturing machine is the computer numerical control (CNC) machine which is a manufacturing machine that uses the computerized operation of machining tools. CNC milling machines are versatile because they can change the milling angle, speed, bits, and materials to produce various parts with different specifications. Because of the sheer number of parts and condition combinations involved in manufacturing, configuring a CNC machine often requires an expert with years of experience. This can result in significant losses owing to the time spent setting up the CNC milling machine. An ideal solution is a feedback system in which the machine automatically loops through its configuration to identify the optimal settings for machining a part within the provided tolerances, such as surface finish smoothness. For any feedback system to function, information from multiple sensors, such as accelerometers or cutting force sensors, is required for real-time classification of the machined surface. Prior studies[3] have successfully predicted machined surface smoothness under specific conditions. However, using such models on CNC machines that do not function under these specific conditions is not ideal, and traditional artificial neural network-based systems, such as those in [3] cannot adapt to newer conditions, such as a new sample type or a new drill bit. Therefore, a generalized model capable of predicting the smoothness of machined surface under diverse conditions would be of great use in the industry.

Mukaeda *et al.* [4], from our research group, have proposed a new OSR based model that demonstrates superior performance compared to traditional neural networks and can adapt to new conditions.

However, because CNC machines operate at high speeds, classification models that may induce lag are not suitable. Consequently, software implementations of OSR-based classification models are not completely suited for real-world applications, and a hardware-based classifier is required. A field programmable gate array (FPGA) can be used in this context. FPGA is a type of integrated circuit that can be programmed and reprogrammed after manufacturing, making it flexible for classification tasks. Additionally, as FPGA is an integrated circuit, it can process information in parallel, allowing for extremely fast operation.

In this paper, we propose a real-time FPGA-based OSR system for the rapid classification of sensor data from a CNC milling machine to assess the quality of milled part's based on surface smoothness.

II. Background and Preliminaries

A. Normal And Complimentary Gaussian Mixture Network and Gated Recurrent Unit

Normal and complimentary Gaussian mixture network (NAC-GMN) is a probabilistic network that estimates a Gaussian mixture model (GMM) corresponding to known classes and a complimentary Gaussian mixture model (CGMM) that excludes known classes. The GMM is expressed by a Gaussian distribution $g(x_t;k,m)$, while the

[†]Yokohama National University, 79-7 Tokiwadai, Hodogaya-ku, Yokohama-shi, Kanagawa, Japan
[‡]Kanagawa Institute of Industrial Science and Technology, 705-1 Shimoimaizumi, Ebina-shi, Kanagawa, Japan

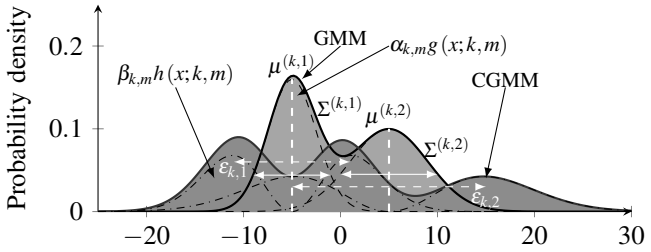


Fig. 1: GMM and CGMM examples.

CGMM is expressed by a complimentary Gaussian distribution $h(x_t;k,m)$.

To evaluate the likelihood of GMM and CGMM, nonlinear transformation is performed on the input $x_t = \{x_1(t),\cdots,x_D(t)\}^{\mathrm{T}}$ of dimension D, resulting in $X(t)$, as shown below:

$$X(t) = \left[1, x_t^{\mathrm{T}}, x_1^2, \cdots, x_1 x_D, x_2^2, \cdots, x_D^2\right]^{\mathrm{T}} \tag{1}$$

Here, $X \in \mathbb{R}^H (H = 1 + D(D+3)/2)$ represents the 1st layer of the NAC-GMN network. Further calculations of the likelihood from layer 1 onwards is are as follows:

$$^{(1)}O_h = X_h(t), \tag{2}$$

$$^{(2)}O_{k,m} = \sum_{h=1}^{H} {}^{(1)}O_h w_h^{(k,m)}, \tag{3}$$

$$^{(3)}O_{k,m} = {}^{(2)}O_{k,m} + w_N^{(k,m)} + \log\left[1 - p(k=0)\right], \tag{4}$$

$$^{(3)}O_{\overline{k,m}} = w_\varepsilon^{(k,m)}\,{}^{(2)}O_{k,m} + \log\left(-{}^{(2)}O_{k,m}\right) + w_C^{(k,m)} + \log\left[p(k=0)\right]. \tag{5}$$

Here, $^{(3)}O_{k,m}$ is the component-wise likelihood of the GMM, and $^{(3)}O_{\overline{k,m}}$ represents the component-wise likelihood of the CGMM. The output of the 3rd layer is a concatenated vector of the GMM and CGMM, as shown below:

$$^{(3)}O_t = \left\{ {}^{(3)}O_{k,m}(t), {}^{(3)}O_{\overline{k,m}}(t) \right\}_{\substack{k=1,\cdots,K,\\ m=1,\cdots,M_k}}. \tag{6}$$

B. Gated Recurrent Unit

Gated Recurrent Unit (GRU) [5] refers to a type of recurrent neural network methodology that uses gates to regulate the flow of information into the network. It uses a reset and an update gates for the regulatory purposes mentioned above.

The input layer includes data given by $x^{(t)} \in \mathbb{R}^d$, where $x^{(t)}$ represents the input vector at time $t \in [1,T]$ and T is the length of the time series.

GRU's forward propagation equations are as follows:

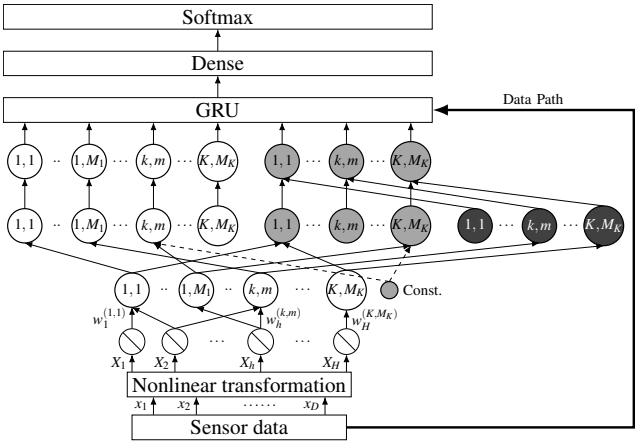


Fig. 2: The proposed classifier for estimating machined surface roughness [4]

$$r^{(t)} = \sigma\left(x^{(t)}W_r + h^{(t-1)}U_r + b_r\right), \tag{7}$$

$$z^{(t)} = \sigma\left(x^{(t)}W_z + h^{(t-1)}U_z + b_z\right), \tag{8}$$

$$\tilde{h}^{(t)} = \phi\left(x^{(t)}W_h + \left(r^{(t)} \odot h^{(t-1)}\right)U_h + b_h\right), \tag{9}$$

$$h^{(t)} = z^{(t)} \odot h^{(t-1)} + \left(1 - z^{(t)}\right) \odot \tilde{h}^{(t)}. \tag{10}$$

Here, $\odot$ denotes a Hadamard product, σ denotes the sigmoid function, and ϕ indicates the hyperbolic tangent function. d represents the number of input features and n represents the number of output features. $W_r, W_z \in \mathbb{R}^{d\times n}$ and $U_r, U_z \in \mathbb{R}^{n\times n}$ are weight parameters, and $b_r, b_z, b_h \in \mathbb{R}^n$ are biases.

$r^{(t)}$ represents the reset and $z^{(t)}$ the update gate vectors. $h^{(t-1)}$ represents the hidden state from the previous time-step, and $\tilde{h}^{(t)}$ indicates the candidate hidden state at the current time-step. $\tilde{h}^{(t)}$ uses the reset gate value to decide whether the older hidden state should be forgotten. Finally, $h^{(t)}$ represents the hidden state of the current time-step.

The reset gate regulates the amount of the previous hidden state to be forgotten, whereas the update gate regulates the amount of information from the previous hidden state to be carried over to the next time-step.

C. Prior researcher

The network proposed by Mukaeda *et al.* [6] is a probabilistic recurrent neural network comprised of NAC-GMN and GRU, as shown in Fig. 2. This network first performs a nonlinear transform on the input data, stabilizing the GMM and CGMM parameters for use in the network. Here, $\alpha_{k,m}$, $\beta_{k,m}$ represent the mixing coefficients of GMM and CGMM, and $p(k=0)$ represents the prior probability of the unknown class.

In its first half, the network calculates the likelihood vectors for GMM and CGMM, and in the second part, the likelihood vectors along with the time-series sensor data are used as inputs to perform the classification. The input to the GRU layer is expressed as follows:

$$o^{(t)} = \left[x^{(t)}, {}^{(3)}O_t\right] \in \mathbb{R}^{D_o},$$

where $D_o = D + 2\sum_{k=1}^{K} M_k$. The hidden state $h^{(t)}$ evaluated by the GRU is then passed through a dense layer containing two neurons. Here, the CGMM values from the OSR method, NAC-GMN, were also used as inputs to the GRU. The final result is obtained by passing the dense-layer values through a softmax function. This final result belongs to one of two classes, labeled 0 (smooth) or 1 (rough).

III. Model and Architecture

A. FPGA implementation of proposed method

The implementation of a GRU or GMM is challenging on an FPGA because both use complex activation or computationally expensive functions such as sigmoid, hyperbolic tangent, and exponential functions. Traditional implementation of these functions is either through IP cores that rely on clock cycles and therefore cannot be used to calculate the complete output in a single clock cycle, or through the implementation of a lookup table (LUT) [7] [8]. An LUT provides reasonable accuracy for hand-off performance; however, it also consumes significant memory, making it unsuitable for models with large numbers of parameters. By contract, IP core-based exponential functions rely on clock cycles; requiring multiple IP cores to evaluate output in a pipeline that delivers one output per clock cycle.

Here, approximation methods are appealing because they are neither computationally nor memory intensive [9] [10]. Furthermore, for offline model implementation, a software version trained using approximation method can have its resulting weights quantized [11] for FPGA implementation optimizing circuit resources.

1) Approximation method of probability distribution: Because the proposed method requires log linearization, an approximation of the logarithmic function is also required. This is expressed by the following equation:

$$\log_2(z) = \log_2(u \times 2^s) = s + \log_2 u, \tag{11}$$

When $1 \leq u < 2$ and $s \in \mathbb{Z}$, the logarithmic function can be expressed as follows:

$$\log_2(z) = s + \log_2 u \approx s + u + 1 \equiv \widehat{\log_2}(z). \tag{12}$$

Using the approximation method, the 3rd layer calculations for the GMM and CGMM are expressed as follows:

$${}^{(3)}O_{k,m} = {}^{(2)}O_{k,m} + w_N^{(k,m)} + \widehat{\log_2}[1 - p(k=0)], \tag{13}$$

$${}^{(3)}O_{\overline{k,m}} = w_\varepsilon^{(k,m)}\,{}^{(2)}O_{k,m} + \widehat{\log_2}\left(-{}^{(2)}O_{k,m}\right) + w_C^{(k,m)} + \rho, \tag{14}$$

where $\rho = \log[p(k=0)]$ is a constant.

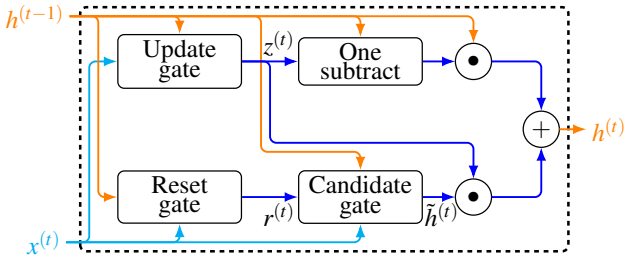


Fig. 3: GRU FPGA pipeline overview.

2) Approximation of activation functions: GRU uses the sigmoid and hyperbolic tangent activation functions; both use the exponential function as shown below:

$$\sigma = \frac{1}{1+e^{-x}}, \tag{15}$$

$$\phi = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{16}$$

Implementing an exponential function on FPGA is challenging because heavy reliance on memory or IP cores. Thus, in the proposed method, we use a hard-sigmoid ($\widehat{\sigma}$) and hard-hyperbolic tangent ($\widehat{\phi}$) implemented using combinational circuits in the FPGA for rapid evaluation. The proposed hard activation functions are determined as follows:

$$\widehat{\sigma} = \max\left(0, \min\left(1, \frac{x+2}{4}\right)\right) \tag{17}$$

$$\equiv \max\left(0, \min\left(1, (x+2) >> 2\right)\right) \tag{18}$$

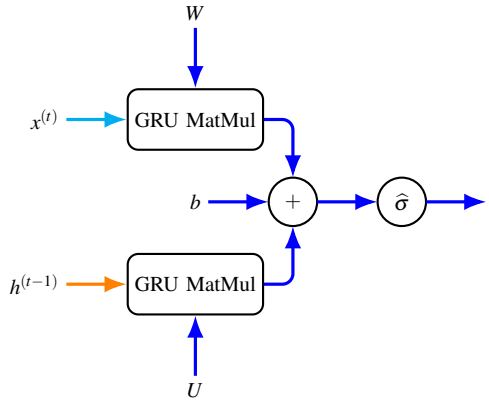
$$\widehat{\phi} = \max\left(0, \min\left(1, x\right)\right) \tag{19}$$

Furthermore, in the case of $\widehat{\sigma}$, the division is carried out using a bit-shift operation thus reducing the computational cost associated with division. This process enables rapid evaluation of the approximate values for both activation functions while preserving accuracy.

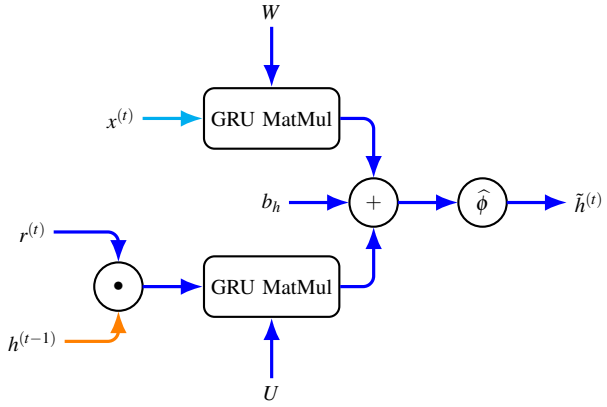
3) Pipeline: A GRU is a type of RNN that relies critically on the hidden state $h^{(t)}$. If the pipeline cannot return to a new hidden state before the next time-step calculations begin, the next step is halted, as mentioned in [12]. Therefore, a robust pipeline latency management is necessary to achieve a high operating frequency.

Here, we propose a combination-circuit architecture that performs each step of the calculation in a single clock cycle. Because all the modules are parallelized and function as a combination circuit, $h^{(t)}$ is completely dependent on the number of circuit resources available.

Fig. 4(a) illustrates the general pipeline of the "update gate" and "reset gate" from Fig. 3. A general purpose GRU matrix multiplication (GRU MatMul) module is constructed to multiply input values and weights. This module is designed to accept an input of dimension $\mathbb{R}^r$ and a weight matrix of dimension $\mathbb{R}^{r \times c}$, returning an output array of dimension $\mathbb{R}^c$, where r is the number of rows and c is the number of columns.



(a) Update gate and Reset gate



(b) Candidate gate

Fig. 4: (a) The update and reset gate pipeline, and (b) The candidate gate pipeline.

Here, $b = b_r, b_z$, $W = W_r, W_z$, and $U = U_r, U_z$. The final update and reset gate evaluation equations, including the hard activation functions, are as follows:

$$r^{(t)} = \hat{\sigma}\left(x^{(t)}W_r + h^{(t-1)}U_r + b_r\right), \tag{20}$$

$$z^{(t)} = \hat{\sigma}\left(x^{(t)}W_z + h^{(t-1)}U_z + b_z\right). \tag{21}$$

For the "candidate gate," the input from the reset gate $r^{(t)}$ is used along with $x^{(t)}$ and $h^{(t-1)}$. The pipeline of the candidate gate module is depicted in Fig. 4(b). Another module, represented by the $\odot$ operator, is constructed to perform the Hadamard product, or element wise multiplication. With this structure, calculation of $\tilde{h}^{(t)}$ begins when $r^{(t)}$ is evaluated.

The resulting candidate gate evaluation equation is as follows:

$$\tilde{h}^{(t)} = \hat{\phi}\left(x^{(t)}W_r + h^{(t-1)}U_r + b_r\right). \tag{22}$$

Finally, the "one subtract" module is used solely to perform subtraction operation, necessitated by the type of representation used in the FPGA design.

With the proposed pipeline, as outlined in Fig. 3, the output is evaluated when an input was provided. Furthermore, because the weights are initialized before the pipeline, reading the weights does not affect the processing time.

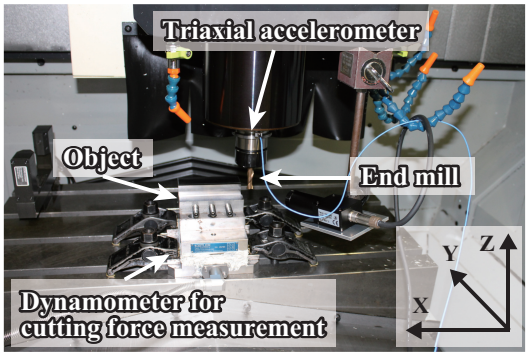


Fig. 5: Data acquisition environment [6].

TABLE I: CNC machine milling conditions.

End Mill	Diameter [mm]	Number of cutting tooth	Angle of twist [°]	Number of experiments
No.1	10	2	30	100
No.2	10	4	30	20
No.3	10	4	45	20
No.4	6	2	30	20
No.5	6	4	30	20
No.6	6	4	45	20

TABLE II: CNC machine configurable conditions.

Tool diameter [mm]	Cutting speed [m/min]	Feed amount per blade [mm/blade]	Radial depth of cut [mm]	Axial depth of cut [mm]	Cutting direction
10	50 ~ 100	0.05 ~ 0.15	0.2 ~ 1.0	3.0 ~ 15.0	Down cut
6	50 ~ 100	0.03 ~ 0.09	0.12 ~ 0.6	3.0 ~ 9.0	Down cut

IV. Simulation Experiments

A. Dataset

The dataset used in this study was collected from Makino V33 CNC machines and includes 3-axis accelerometer data, 3-axis force sensor data, and net force data, all sampled at a rate of 10 kHz. The data were obtained from six different CNC mills under 200 distinct milling conditions, as detailed in Table I. All machines operated dry, without lubrication, on a 100 mm × 32 mm × 5 mm S50C sample. The milling conditions were randomly set within the constraints listed in Table II for all 200 conditions.

As the milling speed varies for each measurement condition, the resulting data have a varying number of data points and are therefore not directly usable for training and testing purposes. Therefore, the raw data must be resampled spatially so that the number of data points per condition is equal. The data points per condition were resampled at a sampling frequency of 300 mm^{-1}, ensuring 300 sample points for each mm. The accelerometer data were full-wave rectified, and the force sensor values were passed through a low-pass filter with a spatial frequency of 100 mm^{-1}.

Finally, a data point is classified as smooth (and labeled "0") if $R_a < 1.6\mu\text{m} \cap R_z < 6.3\mu\text{m}$ according to JIS (Japanese Industrial Standards). Any value outside this constraint is considered as rough (labeled "1"). The final data used for

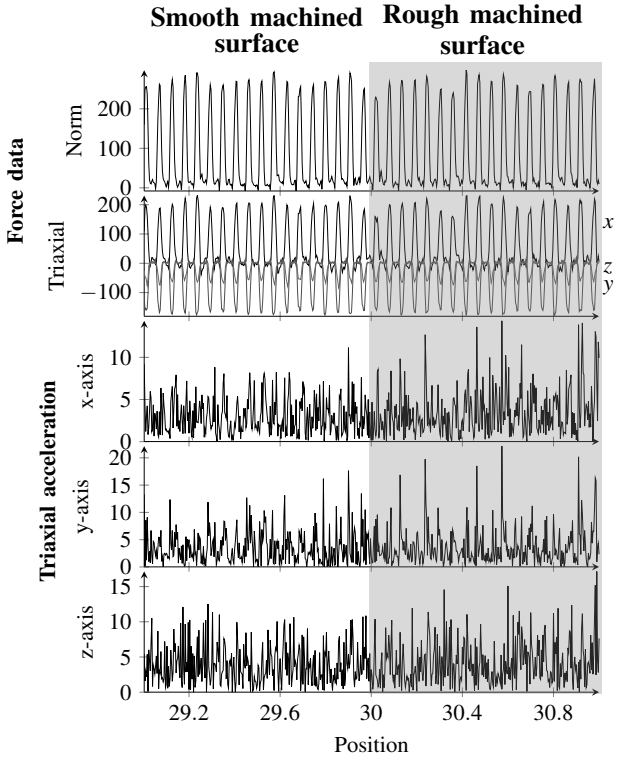


Fig. 6: Final data used to train the model.

training and testing the model are shown by fig. 6, where the shaded ares are labeled as rough, and the non-shaded areas are smooth. The force data includes 3-axis cutting force sensor data and normalized data, while the acceleration data consists of 3-axis measurement .

B. Experiments

In this study, the performance of the FPGA system was evaluated using simulations. Software simulations for model performance evaluation were conducted using the PyTorch framework. A custom GRU implementation was developed to enable the use of hard activation functions (Equations 17 and 19). For training, 120 out of 200 conditions were randomly selected, with the first (12 mm from the beginning) was used for training. Two test sets were used for testing: Testset-1 (TS1) comprised rest of the remaining 80 conditions not used for training, while Testset-2 (TS2) comprised data from the 120 conditions, excluding the 12 mm parts used for training. For each condition, the simulation experiment was conducted five times.The micro-F1 score (accuracy) was used for model performance evaluation. The proposed model was compared with a traditional Random Forest (TF) to the Random Forest (RF)-based model to assess its performance. The three RF models were trained using the following parameters:

Case A [6]:
Power spectrum

Case B [6]:
Power spectrum + milling conditions

Case C [6]:
Power spectrum + milling conditions + tool info.

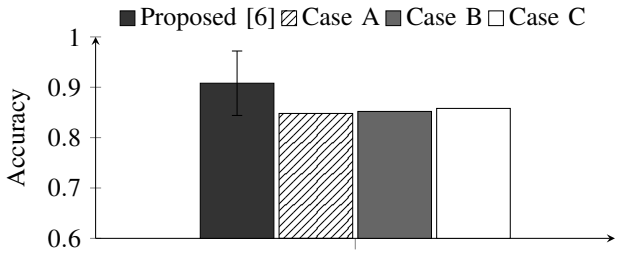


Fig. 7: Comparison of the performance between the proposed model and the traditional traditional RF based models in software implementations.

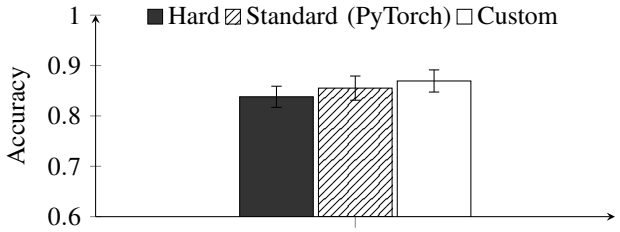


Fig. 8: Comparison of the performance between different versions of the model.

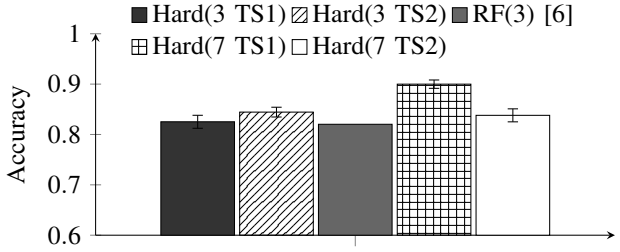


Fig. 9: Performance of the proposed model using only 3 features (3-axis acceleration) compared to using all features.

Here, the power spectrum refers to how the training data from the 120 conditions were processed to obtain the average power spectrum value. For this study, the FPGA used was an Intel Cyclone V 5CSXFC6D6F31C6. Circuit synthesis, timing analysis, and optimization were conducted using Quartus Prime Version 20.1.1 Build 72011/11/2020 SJ Lite Edition.

C. Results

Although the proposed model is trained solely on data from 120 conditions without any preprocessing, it outperforms traditional models trained on more detailed data, such as milling conditions and tool information, as shown in Fig. 7. In Fig. 8 we compare the accuracy results of the Standard PyTorch version, the custom GRU with the standard activation function version, and the custom GRU with the hard activation function version of the proposed model. The performance of the proposed FPGA architecture was comparable to that of traditional implementations, yielding similar accuracy. As shown in Fig. 9 when limited to acceleration

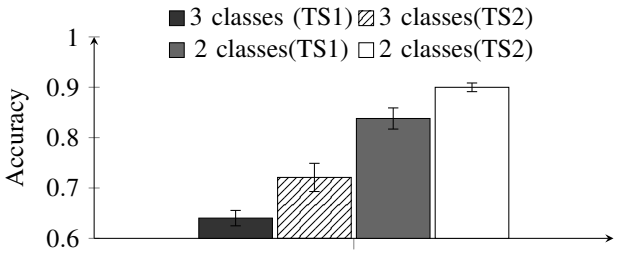


Fig. 10: Performance of proposed model when classifying 3 classes.

data alone, the proposed model outperforms the RF models. However, when the number of features is reduced by more than half, the performance significantly drops compared to the model trained with all seven features. Finally, Fig. 10, demonstrates that when the number of classes increased from two to three, the model's performance significantly decreased.

D. Discussion

From the results, it is evident that when using the approximated version of the model proposed by Mukaeda *et al.* [6], classification performance drops significantly. This is likely owing to the use of hard activation functions and other approximations. The errors introduced by these approximations in the model stack result in a higher overall error differential. Furthermore, the implementation of this model on an FPGA involves different hardware optimizations, which increases the complexity of hardware implementation. Acceleration sensors are cheaper and easier to apply in existing machines compared to force sensors, which are expensive and difficult to use. However, when using only acceleration data, a performance loss was observed owing to the lack of sufficient data. Similar results were observed with 3 class classification, where even with all features included, there was not enough information to distinguish one tool set from another using only sensor data. Compared to TS1, the TS2 results in Fig. 9 and Fig. 10 are higher. Thus, we can conclude that the training itself was successful. However, because the sensors values for different tools appeared similar, distinguishing between them was challenging. Despite this, the final surface profiles varied significantly. Our future research will focus on improving 3 class classification or developing a regression model for surface finish estimation. To enhance regression or multi-class classification performance, new sensors, such as inertial measurement units (IMUs) will be necessary to provide more detailed information during the milling process. This additional information will offer a new perspective on the currently available information.

V. Conclusion

In this study, we developed a probabilistic neural network using the Normal and Complementary Gaussian Mixture Model (NAC-GMN) combined with a Gated Recurrent Unit (GRU) for an FPGA-based implementation of open-Set recognition (OSR) method to classify the surface smoothness of parts fabricated/finished by a CNC milling machine under varying conditions. The inputs to the model included 3-axis acceleration, 3-axis force, and net force time-series data. We confirmed that the proposed model could successfully classify surface smoothness with a higher degree of accuracy when compared to traditional RF models.

Traditionally, CNC mill settings have be determined through trial and error, leading to wasted samples being used for testing, and lower productivity of production lines when new parts or conditions are introduced. In contrast, the proposed method can operate in real-time without preprocessing of the input data, making it suitable for a feedback system that automates the traditional trail and error process, thereby improving production line productivity. In the future, our goal is to develop a model with higher performance and reliability.

VI. Acknowledgement

I would like to express my deepest gratitude to Tomohiro Yokota and Makoto Okuda from the Kanagawa Institute of Industrial Science and Technology, 705-1 Shimoimaizumi, Ebina-shi, Kanagawa, Japan, for their invaluable assistance in collecting the data used in this research.

References

[1] T. Tsuji, O. Fukuda, H. Ichinobe, and M. Kaneko, "A log-linearized gaussian mixture network and its application to eeg pattern classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 29, no. 1, pp. 60–72, 1999.

[2] S. Soltani, Y. E. Sagduyu, R. Hasan, K. Davaslioglu, H. Deng, and T. Erpek, "Real-time and embedded deep learning on FPGA for RF signal classification," *CoRR*, vol. abs/1910.05765, 2019.

[3] D. Karayel, "Prediction and control of surface roughness in cnc lathe using artificial neural network," *Journal of Materials Processing Technology*, vol. 209, no. 7, pp. 3125–3137, 2009.

[4] T. Mukaeda and K. Shima, "Open-set motion recognition and adaptive structural modification of classifiers based on clustering of unknown motions," pp. 4307–4312, 10 2023.

[5] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014.

[6] T. Mukaeda, K. Shima, T. Yokota, and M. Okuda, "Surface roughness estimation of end-milled surfaces using probabilistic recurrent neural network incorporating complementary gaussian distributions (in japanese)." The 31st Symposium on Fuzzy, Artificial Intelligence, Neural Networks and Computational Intelligence, 2023.

[7] A. M. Dalloo, A. J. Humaidi, A. K. A. Mhdawi, and H. Al-Raweshidy, "Low-power and low-latency hardware implementation of approximate hyperbolic and exponential functions for embedded system applications," *IEEE Access*, vol. 12, pp. 24151–24163, 2024.

[8] A. X. M. Chang, B. Martini, and E. Culurciello, "Recurrent neural networks hardware implementation on fpga," 2016.

[9] B. H.-G. H. Amin, K.M. Curtis, "Piecewise linear approximation applied to nonlinear function of a neural network," *IEE Proceedings - Circuits, Devices and Systems*, vol. 144, pp. 313–317(4), December 1997.

[10] T. Kusaka and T. Tanaka, "Fast and accurate approximation methods for trigonometric and arctangent calculations for low-performance computers," *Electronics*, vol. 11, no. 15, 2022.

[11] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1," *CoRR*, vol. abs/1602.02830, 2016.

[12] Z. Que, H. Nakahara, E. Nurvitadhi, H. Fan, C. Zeng, J. Meng, X. Niu, and W. Luk, "Optimizing reconfigurable recurrent neural networks," in *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 10–18, 2020.