Autonomous Deep Quality Monitoring in Streaming Environments

Andri Ashfahani*
SCSE, NTU
Singapore
andriash001@e.ntu.edu.sg

Mahardhika Pratama***
SCSE, NTU
Singapore
mpratama@ntu.edu.sg

Edwin Lughofer
DKBMS, JKU
Austria

E. Y. K. Yee
SIMTech, A*STAR
Singapore

Abstract—The common practice of quality monitoring in industry relies on manual inspection well-known to be slow, error-prone and operator-dependent. This issue raises strong demand for automated real-time quality monitoring developed from data-driven approaches thus alleviating from operator dependence and adapting to various process uncertainties. Nonetheless, current approaches do not take into account the streaming nature of sensory information while relying heavily on hand-crafted features making them application-specific. This paper proposes the online quality monitoring methodology developed from recently developed deep learning algorithms for data streams, Neural Networks with Dynamically Evolved Capacity (NADINE), namely NADINE++. It features the integration of 1-D and 2-D convolutional layers to extract natural features of time-series and visual data streams captured from sensors and cameras of the injection molding machines from our own project. Real-time experiments have been conducted where the online quality monitoring task is simulated on the fly under the prequential test-then-train fashion - the prominent data stream evaluation protocol. Comparison with the state-of-the-art techniques clearly exhibits the advantage of NADINE++ with 4.68% improvement on average for the quality monitoring task in streaming environments. To support the reproducible research initiative, codes, results of NADINE++ along with supplementary materials and injection molding dataset are made available in https://github.com/ContinualAL/NADINE-IJCNN2021.

Index Terms—evolving intelligent systems, online quality monitoring, deep learning, data streams, quality classification.

I. INTRODUCTION

Condition monitoring plays a vital role in today's manufacturing industries due to its positive contribution toward the increase of productivity, the safety of manufacturing operations, reduction of manpower. It maximizes the lifespan of equipment thereby avoiding unnecessary downtime leading to significant cost saving while ensuring the quality of end product preventing costly scrap, damage of workpiece or surface finishing [1]. The worn tool is also inherent to high energy costs because it requires a high cutting force.[1] [2]

Real-time quality monitoring is highly demanded because the common practice in the industry is deemed too labor-intensive as a result of multi-staged visual inspection [2] [3]. Accurate quality monitoring is important not only to attain high customer satisfaction but also to meet the product's standard set by the relevant authorities. Manual quality checks

are limited in capacity, because of requiring time-intensive efforts in terms of personal stuff, which in turn increases costs for companies significantly. Moreover, in the complex quality control system, a manual inspection, although if conducted thoroughly and with sufficient manpower, would not cover all possible operation modes/phases with sufficient consistency and homogeneity (experts may judge the quality of items differently based on their experience levels, fatigue, mood, or even gender, etc. [2]). This issue has led to an in-depth study in utilizing the data-driven approaches to fully automate quality monitoring of manufacturing products, which abandons the usage of manual checks and induces human-like inconsistencies.

Data-driven quality monitoring is developed through two steps [4], indirect sensing and monitoring where the indirect sensing phase is designated to extract notable features of mounted sensors while the monitoring phase utilizes extracted features to train a model autonomously from the data, which circumvents high development time as is the case for analytical, physical-oriented models. The models are capable of diagnosing possible defects of end products [5] without laborious manual intervention.

Various approaches have been presented in the literature to deliver a reliable data-driven quality monitoring approach. In [6], a neural network is developed to predict the cutter's flank wear in the metal-turning process. A paper investigates the application of an acoustic emission sensor to produce informative features for condition monitoring of chemical-mechanical planarization (CMP) [7]. In [8], the vibration sensor is made use in predicting the tool condition of the ball-nose end-milling process and combined with the fuzzy regression technique. A quality monitoring approach in the multi-staged manufacturing process is proposed in [9], using the clustering approach and the support vector machine. The techniques proposed in [10], [11] focus on the detection and localization of faults in rolling mills by relying on an all-coverage data-driven modeling approach to integrate as many sensor signals as possible in a whole network of models.

Since the multi-sensors are implemented and often lead to the curse of dimensionality, feature selection is needed [12] [13] to reduce the dimension of feature space. Despite its rapid progress, all of these approaches are deemed as offline approaches and assume stationary environments and operation modes. A model is fixed once trained thereby resulting in

---

[1]*Equal Contribution
[2]**Corresponding Author

major performance deterioration in the presence of concept drift [14], an occurrence that leads to changing data distributions and even changing input-output (target) relationships (thus, older trained relations become outdated), see [15] for a survey. Consequently, a retraining phase has to be regularly carried out by experts/operators, which incurs extra costs and intensive computational resources.

The data-driven condition monitoring approach has been advanced in [16] via the deployment of data stream algorithm, pENsemble+ featuring the self-evolving structure and one-pass learning principle. Moreover, pENsemble+ is equipped with the online active learning strategy and the online feature selection approach using ensemble classifiers in an evolving data-streaming context [17] to permanently update their performance and to handle with non-stationary modes. Another online quality monitoring methodology is proposed in [18] for microfluidic chip quality. It integrates the incremental partial least square (iPLS) method into GEN-SMART-EFS [19] for online dimensionality reduction strategy of high-dimensional multi-sensor data. This approach is extended in [20] by introducing the forgetting strategy to handle the concept drift and the multi-objective evolutionary computation for process optimization. A collection of further dynamic data-driven QC approaches can be found in [21], which all serve as one key aspect in nowadays predictive maintenance systems. The online quality monitoring topic deserves further in-depth study due to at least two rationales:

1) Existing approaches heavily rely on the tedious feature engineering step of sensory data. This leads the overall framework to be application dependent with significant development time needed and high input dimension calling for the feature reduction mechanism. Such approaches cannot operate in an end-to-end manner. Notwithstanding that the application of deep learning starts to emerge in the literature where they resolve the issue of hand-crafted features and curse of dimensionality due to its implicit feature learning trait via multiple nonlinear transformation [22], approaches employing deep learning hardly scale to the streaming environments and become outdated quickly in the rapidly changing environments;

2) Existing approaches do not take into account the heterogeneous source of information leaving aside the use of multi-modal information, e.g., sensor+image, sensor+text. In order to assure accurate diagnosis of product's defect, especially in systems where the prediction quality depends on the mixture of various process phases or typically records different types of data, a multi-modal model can be utilized [23], [24].

This study presents an online quality monitoring of transparent mold within a non-stationary streaming environment using an extension of a recently developed deep learning algorithm for data streams, termed as *Neural Network with Dynamically Evolved Capacity++* (NADINE++). The modification of NADINE in [25] encompasses the integration of 1-D and 2-D convolutional layers (CNN) to bypass a complex feature engineering step in the condition monitoring task. The 2-D CNN is taken from the pre-trained residual network (ResNet) in the ImageNet problem. That is, only those generating general features are adopted leaving aside the object-specific layers. NADINE++ itself features an autonomous trait where the fully connected layer is self-evolved from scratch with the absence of a predefined network structure. In the classifier part, both the hidden nodes and hidden layers are automatically evolved or removed overtime via the network significance (NS) method and the drift detection method, respectively. The concept of adaptive memory exists to perform the experience replay mechanism combating performance loss during the insertion of a new layer. In addition, the soft forgetting mechanism is prepared to overcome the exploding gradient problem by controlling the step size of model updates.

The online quality monitoring task is performed in two facets: the use of sensory information from the sensor and the use of multi-modal information from both sensory and visual information. The first one is handled by amalgamating 1-D CNN while the second one is addressed by both 1-D and 2-D CNNs. This is established by creating a heterogeneous network in which the output of 2-D CNN is fused with the sensory data processed by 1-D CNN and in turn passed to the evolving multilayer perceptron (MLP) classifier. Furthermore, a multi-class online quality classification problem is studied here realizing 2 implementation scenarios: one-step-ahead and current-batch prediction. The explanation about the injection molding machine can be found in the supplemental document.

The major contribution of this paper is summed up into three folds: 1) this paper offers NADINE++ for online quality monitoring in truly streaming environments. NADINE++ is free from complex feature engineering step and enables the end-to-end training mechanism while being self-adaptive to track changing distributions of data streams; 2) this paper offers a comprehensive study for online quality monitoring of transparent mold from the injection molding machine where it investigates the use of sensory information, image information and multi-modal information. In addition, multi-class online quality classification schemes can be handled, providing a direct fault identification step (through the prediction of fault classes); 3) real-time experiments on the injection molding machine have been carried out where all real-world data, Python implementation of NADINE++, supporting materials are made publicly available for the convenience of reproducing our results. Comparisons with several state-of-the-art algorithms exhibit the advantage of NADINE++ in all simulation scenarios.

II. PROBLEM FORMULATION

Online quality monitoring is defined here as the detection problem of transparent mold quality produced by the injection molding machine. Unlike in the offline case where a predictive model $f(.)$ is crafted based on prerecorded samples $D = [X; Y] \in \Re^{(N \times (u+m))}$, where N, u, m respectively stand for the number of data points, input features and target

classes, and fixed once deployed, the online quality classifier is changed with streaming data. That is, there exist continuous information flows $B_1, B_2, \ldots, B_k \ldots, B_K \in \Re^{N \times u}$ where K denotes the number of batches. The typical characteristic of the data stream is seen in the rapidly changing data distributions, i.e. $P(X,Y)_k \neq P(X,Y)_{k+1}$ [26]. Adapting to such changes without a retraining phase from scratch may catastrophically erase previously valid knowledge as only forces to be trained to the new one. In addition, the retraining phase imposes considerable computational and memory footprints that cannot keep pace with the fast characteristics of the molding machine. A model is often forced to predict the data batch B_k first due to the issue of label latency. The prequential test-then-train protocol is implemented in this study where a model $f(.)$ is used to predict the target of the data batch B_k before updating it with the same data batch [26]. Numerical evaluation is executed per data batch B_k in order to check the effect of concept drift in the numerical results.

Input attributes $X \in \Re^{N \times u}$ arrive with the absence of target classes where it is collected from the indirect sensing mechanism [27] via built-in sensors of the injection molding machine generating time-series data and external static camera mounted at the end of manufacturing cycle delivering colored photos of the transparent mold. The data batch $B_k = X_k$ are collected in batch within the sampling time leading to $N = 50$ and $K = 59$. Predictions are made for this current batch of samples to produce classification statements (labels with uncertainties) based on which an expert/operator can provide feedback. These in turn can be used as annotation labels collected in a target vector Y_k, resulting in a supervised batch of samples $B_k = [X_k; Y_k]$. The labeled data batch B_k is then used for training the deep network in order to be up-to-date with the most recent batch for reliably classifying the next batch. Our implementation comprises two scenarios, one-step-ahead and current-batch prediction. Both are multi-class classification problems where it consists of three classes: good, short-forming and weaving, leading to $m = 3$ as explained in the supplemental document.

III. LEARNING POLICY OF NADINE++

A. Network Architecture of NADINE++

NADINE++ extends the original NADINE in [25] with the addition of the feature extraction layer alongside the self-evolving fully connected layer generalizing its feasibility into various condition monitoring problems regardless of the machine or sensor specification. That is, the raw input samples X collected from built-in sensors or external cameras are fed to the convolutional feature mapping $F(X)$ extracting the feature mapping. The residual mapping [28] is implemented here where the underlying goal is to approximate a residual function $H(X) - X$ where $H(X)$ is the desired mapping function. This concept is similar to the introduction of a shortcut connection in terms of the identity mapping to the convolutional feature $F(X) + X$. The use of such a connection is inspired by the increase of difficulty in training a very deep network under a standard structure due to the issue of

vanishing or exploding gradient problem. It is also supported by the finding in [29] where the mutual information of the natural features rapidly decreases across the network depth. The output of $l - th$ residual building block $Z \in \Re^{u'}$ is formalized as follows:

$$Z_l = F(X, W_{conv}^{l,i}) + X \tag{1}$$

where u' denotes the number of natural features, while $W_{conv}^{l,i}$ stands for the $i - th$ filter weight of the $l - th$ layer. Note that two variants of filters is deployed here, termed one dimensional $W_{conv}^{l,i} \in \Re^g$ filter and two dimensional $W_{conv}^{l,i} \in \Re^{g \times g}$ filter where g denotes the filter size. It is put forward to construct the 1-D and 2-D CNN network structure handling time-series and visual information, respectively [30].

After stacking L_{conv} residual layers, the natural features of the last residual layer is passed to the classification layer formed as a MLP network parameterized by the connective weight and bias $W_{in}^{i,l} \in \Re^{u' \times R_l}, b_{i,l} \in \Re^{R_l}$ where R_l stands for the number of hidden units in the $l - th$ fully connected layer. It is formally written as follows:

$$h_l = s(W_{in}^{i,l} Z_{L_{conv}} + b_l) \tag{2}$$

Note that the fully connected layer of NADINE++ features a self-adaptive trait allowing the number of hidden nodes R_l and the number of fully connected layers L_{fully} to be automatically developed during the training process. The classification decision is drawn by a softmax layer converting the activation degrees of the last hidden layer into the output posterior probability as follows:

$$\hat{y} = softmax(W_{out} h_{L_{fully}} + c) \tag{3}$$

where $W_{out} \in \Re^{R_{L_{fully}} \times m}, c \in \Re^m$ are the output weights and the output bias. The predicted label $\hat{Y}$ is taken from the output $\hat{y}$ having the highest multi-class probability. This can be written mathematically as $\hat{Y} = \text{argmax}_{\hat{y} \in \Re^m} \hat{y}$.

B. Structural Learning of NADINE++

NADINE++ is constructed by an evolving MLP classifier. It features an open structure where its hidden nodes and layers are automatically generated based on the network significance (NS) method and the drift detection method pinpointing possible changes in data distributions. The NS method is crafted using the bias-variance decomposition signifying the underfitting and overfitting situations. A new node is introduced in the case of underfitting or high bias while an inconsequential node is pruned if the overfitting or high variance condition is present. Network bias and variance are expressed as follows:

$$NS = E[(\hat{y} - E[\hat{y}]^2)] + (E[\hat{y}] - y)^2 \tag{4}$$

$$NS = Variance(\hat{y}) + Bias(\hat{y})^2 \tag{5}$$

where $Variance(\hat{y})$ and $Bias(\hat{y})$ respectively stand for the network variance and bias respectively. Equation (4) is solved by assuming that x is drawn from the normal distribution $p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{\sigma^2}\right)$ with the mean μ and variance σ.

Unlike in the original NADINE, x here denotes the features generated from the convolutional layer.

The statistical process control (SPC) condition [31] is extended by integrating the adaptive confidence level controlling the degree of confidence with respect to the level of network bias and variance. It is written as follows:

$$Growing: \mu_{bias}^t + \sigma_{bias}^t \geq \mu_{bias}^{min} + \kappa\sigma_{bias}^{min} \tag{6}$$

$$\kappa = 1.25\exp\left(-Bias^2\right) + 0.75 \tag{7}$$

$$Pruning: \mu_{var}^t + \sigma_{var}^t \geq \mu_{var}^{min} + 2\xi\sigma_{var}^{min} \tag{8}$$

$$\xi = 1.25\exp\left(-Var^2\right) + 0.75 \tag{9}$$

It is worth noting that the term 2 is inserted into (8) to avoid the direct-pruning-after-adding condition. This strategy enables dynamic confidence level in the range of $[68.2\%, 95.2\%]$ and $[68.2\%, 99.9\%]$. Furthermore, $\mu_{bias}^{min}, \sigma_{bias}^{min}, \mu_{var}^{min}, \sigma_{var}^{min}$ are reset once (6) and (8) are satisfied. It allows the node growing and pruning mechanisms to be carried out frequently in the case of high bias and variance. The two conditions are capable of capturing abnormal patterns leading to the increase of network bias and variance. It is worth mentioning that the node growing and pruning phase is only localized in the last layer to enable stable hidden representation.

The hidden layer growing strategy of NADINE++ is controlled by the drift detection approach based on the Hoeffding's bound strategy to control the appropriate level to declare a drift. It is modified from [16] where it utilizes the accuracy vector $A_k \in \Re^n$ recording the prequential error of the $k - th$ data batch B_k. Misclassified instance is marked as "1" while correct classification returns "0". As a result, the increase of population mean mirrors performance's deterioration of a model thereby signifying the concept drift. The first step is to determine the switching point cut signalling the start of concept drift cut:

$$\hat{A} + \epsilon_{\hat{A}} \leq \hat{B} + \epsilon_{\hat{B}} \tag{10}$$

where $A \in \Re^N$ and $B \in \Re^{cut'}$ and $cut' \leq N$. cut' stands for the hypothetical cutting point defined as $[25\%, 50\%, 75\%]*N$ to address the issue of false alarm while $\hat{A}$ and $\hat{B}$ denote the statistics of accuracy matrix A and B, respectively. The Hoeffding's bounds $\epsilon_{\hat{A}}, \epsilon_{\hat{B}}$ are derived as follows [32]:

$$\epsilon_{\hat{A},\hat{B}} = \sqrt{(1/(2n_{\hat{A},\hat{B}}))\ln(1/\alpha)} \tag{11}$$

where $n_{\hat{A},\hat{B}}$ denotes either number of sample in A or B; and α is the significance level of the Hoeffding's bound and is inversely proportional to the confidence level [32].

Once eliciting the cutting point cut, it enables the construction of another partition of accuracy matrix $C \in \Re^{(N-cut)}$, where $A = [B; C]$. A drift is signalled if $|\hat{B} - \hat{C}| \geq \epsilon_d$ holds in which it compares two possible distinct concepts carried in B and C. Another situation is formulated in the warning condition likely leading to the drift case but still calling for confirmation with next data streams. It is set as $|\hat{B} - \hat{C}| \geq \epsilon_w$.

The Hoeffding's bounds to confirm warning and drift condition is derived as follows [32]:

$$\epsilon_{d,w} = \sqrt{\frac{N - cut}{2 \times cut \times N}ln(\frac{1}{\alpha_{d,w}})} \tag{12}$$

Since $\epsilon_d > \epsilon_w$, the significance level is set as $\alpha_d < \alpha_w$ [32]. On the other hand, the stable condition is returned if the null hypothesis remains valid.

A new layer is inserted in the case of drift thereby deepening the network structure while a data buffer $B_w = [B_w; B_k]$ is created in the warning phase. The data buffer is to be replayed if the drift case is substantiated in the next data batch checking the consistency of the data buffer. Only the parameter learning phase is carried out during the stable phase. It is worth mentioning that the addition of a new layer must be undertaken carefully due to the risk of catastrophic forgetting. That is, an untrained layer is stacked to the last layer playing a vital role in the network's output. The adaptive memory concept is applied to overcome this issue inspired by the experience replay mechanism of continual learning [25].

C. Adaptive Memory Strategy

The addition of a new layer in the classifier induces the catastrophic forgetting problem since the final output is controlled by an untrained component missing previous concepts. The adaptive memory strategy is implemented here where it performs the experience replay-like mechanism [33]. The key idea is to memorize important samples describing the underlying data distribution. Important samples are defined as those satisfying one of the two conditions: 1) they must not be redundant samples leading to the over-fitting issue; 2) they must not be outliers portraying the low variance direction of data distribution. The concept of ellipsoidal anomaly detector is applied here [34] where it enables to select of anomalous but useful samples based on the multivariate Gaussian distribution. A sample is selected into the adaptive memory if the following condition is satisfied:

$$t_u^1 \leq M(x_k; C, Cov^{-1}) \leq t_u^2 \tag{13}$$

where C and Cov^{-1} respectively stand for the center and inverse covariance matrix of the multivariate normal distribution, while t_u^1 and t_u^2 denote the inverse of the chi-square distribution with u degrees of freedom $\chi_u^2(\alpha)$; M denotes the Mahalanobis distance. α is the confidence level and set such that $\alpha_1 < \alpha_2$ where $\alpha_1 = 0.99$, $\alpha_2 = 0.999$. This means that edge points of the data distribution are selected characterizing well the outer contour of the distribution, but no real outlying points. Note that the threshold selection from any unimodal distribution encompasses the majority of data points.

Since the main goal of (13) is to capture unique samples, it often leads to too few samples being captured thus making the experience-replay mechanism ineffective. In addition, hard samples are also picked up when they characterize low confident samples lying close to the decision boundary, i.e. $P(Y|X) \approx 0.5$. Such samples are important for the

training procedure in order to 'sharpen' the decision boundary appropriately and thus to reduce the likelihood of false classifications. A sample is stored in the adaptive memory if the following condition holds:

$$\frac{\hat{y}_1}{\hat{y}_1 + \hat{y}_2} \leq \delta \tag{14}$$

where y_1, y_2 denote the highest and second highest predictive output while δ stands for the predefined threshold fixed at 0.55. Uncertain output is resulted from low ratio between y_1 and y_2 or is close to 0.5.

D. Soft Forgetting Strategy

The soft forgetting mechanism is put forward to govern the learning intensity of MLP classifier components via the adjustment of learning rate. It copes with the catastrophic forgetting problem affecting the performance of NADINE++ and the exploding gradient problem leading to saturating hidden nodes. Intuitively, all relevant parameters to a batch B_k should be updated. On the other hand, the amount of update to all irrelevant parameters should be minimized [35], [36]. This is realized by measuring the correlation of every node of the $l-th$ layer to the target variable on every incoming batch B_k in which the current concept is embraced. It is followed by setting the highly correlated layer's learning rate to a high value while hindering other layers to accept the current concept. The learning rate of the $l-th$ layer η_l is formulated as follows:

$$\eta_l = 0.02 * \exp\left(-(1/\rho(h_l, \hat{y}) - 1)\right) \tag{15}$$

where $\rho(h_l, Y)$ denotes the average correlation between all nodes of the $l-th$ layer to all outputs calculated by the Pearson correlation index. In this study, η_l was capped at 0.02.

IV. RESULTS

This section demonstrates the classification performance of NADINE++ on injection molding data. A prequential test-then-train procedure [26] was used to demonstrate NADINE++'s performance where each data batch is tested first before it is learned. The performance of NADINE++ was simulated on 2 scenarios: One-step-ahead and current-batch prediction. The former scenario tests NADINE++'s ability to estimate the next batch product quality Y_{k+1} exploiting the current batch sensor data X_k. The latter scenario is the main proof of concept attempting to validate NADINE performance in predicting the current batch mold condition Y_k based on current sensor and image data X_k. Also, an ablation study is presented to assess NADINE++'s components.

A. One-step-ahead Prediction

1) Network Structure: The proposed NADINE++ network structure to handle the one-step-ahead prediction problem consists of 2 main components, i.e., 1-D CNN as a feature extractor and an evolving MLP as a classifier, as illustrated in Fig. 1. 1-D CNN was constructed by stacking 3 convolutional layers consisting of 1-D kernels with a pixel, padding and
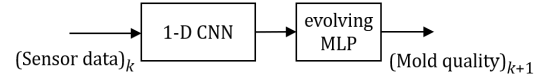

Fig. 1. The NADINE++ architecture used in the first scenario.

stride. The number of input and output channels in the first, second and third layers were respectively selected as $[48, 60]$, $[60, 40]$ and $[40, 20]$. The learned features were then forwarded to an evolving MLP which is able to construct its structure based on problem complexity. A ReLU non-linear activation function was used to decouple the layers. Note that in this scenario, NADINE++ predicts the next batch mold quality Y_{k+1} based on the current batch sensor data X_k.

2) Algorithms and Parameters: The proposed NADINE++ model is compared with other classification algorithm such as Online Deep Learning (ODL) [37], NADINE [25] and incremental bagging [38]. Those are online learning algorithms that use a single scan to process the incoming data. ODL is built upon MLP having a different-depth structure. It is also empowered by hedge backpropagation which is able to improve the performance of the deep networks. The comparison against NADINE aims to present the improvement over an evolving MLP classifier without a feature extractor. The performance of NADINE++ is also benchmarked against the performance of Incremental Bagging.

Those algorithms are re-implemented in the same simulation scenario and procedure to ensure a fair comparison. In the implementation, NADINE, ODL and NADINE++ are trained by Stochastic Gradient Descent (SGD). The hyperparameters are re-tuned for each algorithm attempting to obtain better performance, thereby providing a more competitive experimental setting for testing out our proposed algorithm. NADINE++ parameters α_d and α_w are set to 0.0001 and 0.0005 which controls the drift rate. The learning rate and momentum coefficient are set as $[0.001, 0.02]$ and 0.95. All of these parameters remain unchanged in all experiments. The documentation can be checked in the aforementioned link.

3) Results: Table I shows the performance of consolidated algorithms. The performance is evaluated according to two criteria: accuracy and number of layers (depth). These metrics enable the user to observe the predictive performance of an algorithm and its complexity. The tests were performed 10 times. The average performance across 10 consecutive runs is listed in the table. The t-test is conducted to further validate NADINE++'s predictive performance. The $\times$ mark in the table indicates that the t-test rejects the null hypothesis at the 5% significance level. These are also applied in the second case.

From Table I, it can be observed that NADINE++ outperforms NADINE, ODL and Incremental Bagging in terms of accuracy. Further, the t-test confirms that this result is statistically significant (P < 0.05). This is understood as NADINE++ employs 1-D CNN as a feature extractor. Note that the sensor data used in this experiment is the original normalized time-series signal. There is no signal processing method applied to the sensor data before 1-D CNN. NA-

TABLE I
CLASSIFICATION PERFORMANCE ON ONE-STEP-AHEAD PREDICTION
SCENARIO

MODEL	DEPTH	ACC. (%)
NADINE	9.2 ± 1.69	$76.46 \pm 0.03^{\times}$
ODL	5	$79.40 \pm 0.02^{\times}$
INCREMENTAL BAGGING	N/A	$81.65 \pm 0.00^{\times}$
NADINE++	6.4 ± 0.70	$\mathbf{83.40 \pm 0.02}$

$^{\times}$: Indicates that the numerical results of the respected baseline and
NADINE++ are significantly different.

TABLE II
PRECISION AND RECALL OF NADINE++ ON ONE-STEP-AHEAD
PREDICTION SCENARIO

LABELS	PRECISION	RECALL
NORMAL	0.84 ± 0.044	0.89 ± 0.032
WEAVING	0.83 ± 0.019	0.77 ± 0.071
SHORT-FORMING	0.84 ± 0.031	0.85 ± 0.063

DINE++ surpasses other methods, indicating the advantage of 1-D CNN when processing sensor data. By comparing NADINE++ and Incremental Bagging, we see the benefit of deep neural network architecture to execute a classification task in a streaming environment.

In terms of network complexity, NADINE++ introduced less number of layers compared to NADINE. It is worth mentioning that these algorithms utilize the same drift detection mechanism to govern the addition of depth. This finding signifies that 1-D CNN is able to produce more robust features for the classifier [39]. These features help NADINE++ to minimize performance deterioration due to concept drift. As a result, there is less drift detected by NADINE++ which in turn induces less number of the hidden layer. In industrial applications, it is important to maintain the network complexity low to attain high feasibility for real-time deployment [40].

The precision and recall of each class are evaluated. From Table II, it is depicted that there are gaps between precision and recall. However, the precision and recall values of NADINE++, which are never less than 0.75, can be considered good for a 3-class classification problem. Also, the gaps are quite small indicating that the predictive performance of NADINE++ is unbiased to one of the classes. One may re-tune the hyperparameters to improve the precision and recall metrics in one class. However, it should be conducted carefully as there is a trade-off between precision and recall [41].

B. Current-batch Prediction

1) Network Structure: The second scenario requires NADINE++ to predict the mold quality Y_k based on the current batch sensor and image data X_k. In the production system, this mechanism is useful as the technician can monitor the product quality without interrupting the molding process. As depicted in Fig. 2, NADINE++ put forwards three main components to handle this classification problem, those are 1-D CNN, 2-D
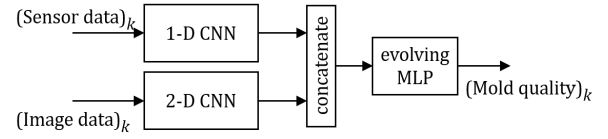

Fig. 2. The NADINE++ architecture used in the second scenario.

CNN and an evolving MLP classifier. The first and second component function to extract useful features from sensor and image data, respectively. The same 1-D CNN structure as used in the first scenario was utilized here, whereas the 2-D CNN structure adopted the ResNet18 feature extractor part. The simplest version of ResNet was selected attempting to reduce the risk of overfitting. ReLU activation function was used to introduce non-linearity. Finally, the generated features by 1-D and 2-D CNN were concatenated together into a long vector for the input of the evolving MLP classifier.

2) Algorithms and Parameters: We compared NADINE++ with four deep network architectures: NADINE [25], ODL [37], ResNet18 [28] and VGG11 [42]. The first two methods were designed to handle the classification problems in an online manner. These methods are able to incrementally improve predictive performance as the number of data increases. NADINE and ODL do not implement any feature extraction layer. As a result, they can only exploit raw sensor data in this scenario with the absence of automatic feature engineering trait. Note that the generated image data from our experiment are RGB images with a size of $150 \times 150 \times 3$. One may easily convert it to a vector with a size of 67500×1 to fit in NADINE and ODL. In practical implementation, however, a huge number of inputs may decrease the performance of NADINE and ODL due to the over-fitting problem [43].

ResNet18 and VGG11 are the prominent deep network architecture for computer vision. Both of them can achieve more than 80% accuracy on ImageNet with [44]. In this scenario, these networks only used image data to predict the mold quality as both of them did not incorporate 1-D CNN. We selected the simplest structure of ResNet and VGG because the number of collected data in our scenario is quite low. This may lead to overfitting if more complex networks were used [45]. All baselines were re-implemented in the same simulation scenario to provide an equal comparison. We used the SGD method to performs end-to-end training on all algorithms. For the baselines, their hyperparameters are hand-tuned and the best-performing results are reported here. For NADINE++, we used the same hyperparameters as used in the first scenario. These settings can also be observed in the raw numerical results which have been uploaded in the provided link.

3) Results: Table III shows the predictive performance of the second scenario. It is observed that NADINE++ achieves the best performance in terms of accuracy. From the t-test, we also found that NADINE++'s performance was statistically different ($P < 0.05$) compared to NADINE, ODL and VGG11. This is reasonable as NADINE++ is able to process both

TABLE III

MODEL	DEPTH	ACC. (%)
NADINE*	8.9 ± 1.45	$76.9 \pm 2.28^{\times}$
ODL*	5	$80.4 \pm 0.01^{\times}$
RESNET18[#]	17	86.5 ± 0.85
VGG11[#]	11	$84.1 \pm 0.99^{\times}$
NADINE++	20 ± 0.47	$\mathbf{87.1 \pm 0.74}$

*: Exploits sensor data, [#]: Exploits image data.

TABLE IV
PRECISION AND RECALL OF NADINE++ AND RESNET18 ON
CURRENT-BATCH PREDICTION SCENARIO

	LABELS	PRECISION	RECALL
N++	NORMAL	0.92 ± 0.003	0.88 ± 0.016
	WEAVING	0.84 ± 0.013	$\mathbf{0.86 \pm 0.000}$
	SHORT-FORMING	$\mathbf{0.87 \pm 0.000}$	0.90 ± 0.000
RN18	NORMAL	0.92 ± 0.011	0.88 ± 0.018
	WEAVING	0.84 ± 0.013	$0.83 \pm 0.014^{\times}$
	SHORT-FORMING	$0.83 \pm 0.011^{\times}$	0.90 ± 0.013

sensory and visual information [30]. Also, the end-to-end training mechanism enables NADINE++ to put more attention on which features help to improve the performance. The performance of NADINE++ is not statistically different from ResNet18. This is understood as NADINE++'s 2-D CNN adopted from ResNet18. However, it is presented in Table IV that our method achieved better precision and recall compared to ResNet18. In addition, NADINE++ is able to increase its capacity if the problem getting more complex.

Compared to NADINE, our method generated less number of layers. NADINE++ added around 3 hidden layers, whereas NADINE introduced more than 8 layers. Note that the reported number of layers in Table IV is the total number of layers including the 2-D CNN layers. Interestingly, the same approach was utilized to control the hidden layer growing mechanism. This behavior indicates that the feature extractor part of NADINE++ is able to generate a more generalizable feature representation [39]. This situation can prevent sudden performance decrease whenever a concept change occurs. As a result, it triggers less addition of depth. Note that it is important to maintain the network complexity as low as possible to help for real-time deployment [40].

Next, we present the precision and recall results of NADINE++ in Table IV to further evaluate its performance in predicting the mold quality. Overall, it is obvious that all classes achieved high recall with high precision. It was higher than 0.80 in all classes. Further, NADINE++ (N++) achieved statistically better F-score (P < 0.05) than ResNet18 (RN18) in weaving and short-forming labels at 0.85 ± 0.006 and 0.88 ± 0.000 respectively, whereas ResNet18 was at 0.84 ± 0.012

TABLE V
CLASSIFICATION PERFORMANCE ON ABLATION STUDY

ABLATION	DEPTH	ACC. (%)
ORIGINAL	20.0 ± 0.47	$\mathbf{87.1 \pm 0.74}$
A	14.0 ± 1.41	$83.4 \pm 1.26^{\times}$
B	20.7 ± 1.06	86.3 ± 1.70
C	18	86.9 ± 0.57

A: Reduce the number of 2-D CNN block layer, B: Without 1-D CNN, C: Without evolving mechanism.

and 0.86 ± 0.010. Note that F-score can be obtained easily from precision and recall. Intuitively, it measures how well NADINE++ predicts a class without confusion. This finding signifies that NADINE++'s prediction is not biased to one of the labels. This aspect is important in a prediction task [41]. Too many false negative and false positive diagnoses will reduce the production cost efficiency as the operator frequently needs to stop, to inspect and to adjust the machine.

C. Ablation Study

Since NADINE++ combines three main components, it has something in common with existing methods in the literature. As a result, it is required to study the effect of removing or adding components to present additional insight into what makes NADINE++ better. Specifically, we quantify the effect of *A*) reducing the number of 2-D CNN block layers from 4 into 2; *B*) removing 1-D CNN, which makes NADINE++ unable to process sensor data; *C*) turning off the evolving mechanism. The ablations were carried out in the second scenario; the results are depicted in Table V. We found that every component contributes to NADINE++'s predictive performance. Another significant outcome from the same table is that reducing the number of 2-D CNN layer dramatically deteriorates NADINE++'s performance. This is understood as 2-D CNN plays an important role to extract useful features from image data which makes the classification task easier. Further, from an information-theoretic view for deep learning the nature of ResNet architecture, which incorporates residual connection, is able to prevent loss of information which makes deep network training convenient [29].

V. CONCLUSION

We proposed NADINE++ a multi-modal deep learning approach for quality monitoring in streaming environments. NADINE++ is able to process both time-series and visual information utilizing 1-D CNN and 2-D CNN. The evolving MLP classifier was put forward to handle concept drift. The results indicate that NADINE++ achieved the best performance in all experiment scenarios. In terms of accuracy, it offered a 4.68% improvement on average. We also found that the evolving mechanism succeeded to generate competitive network structures. In future work, we are interested in incorporating additional ideas from time-series prediction and continuing to explore which mechanisms result in effective methods. The code, raw numerical results, supplementary materials and the

injection molding machine dataset can be accessed in this link https://github.com/ContinualAL/NADINE-IJCNN2021.

Acknowledgment

References

[1] E. P. Carden and P. Fanning, "Vibration based condition monitoring: a review," *Structural health monitoring*, vol. 3, no. 4, pp. 355–377, 2004.

[2] W. Heidl, S. Thumfart, E. Lughofer, C. Eitzinger, and E. Klement, "Machine learning based analysis of gender differences in visual inspection decision making," *Information Sciences*, vol. 224, pp. 62–76, 2013.

[3] S. Ravikumara, K. Ramachandran, and V. Sugumaran, "Machine learning approach for automated visual inspection of machine components," *Expert Sytems with Applications*, vol. 38, no. 4, pp. 3260–3266, 2011.

[4] A. Mitra, *Fundamentals of Quality Control and Improvement*. Hoboken, New Jersey: John Wiley & Sons, 2016.

[5] D. Montgomery, *Introduction to Statistical Quality Control (6th Edition)*. John Wiley & Sons, 2008.

[6] P. S. Paul and A. Varadarajan, "A multi-sensor fusion model based on artificial neural network to predict tool wear during hard turning," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 226, no. 5, pp. 853–860, 2012.

[7] H. Jeong, H. Kim, S. Lee, and D. Dornfeld, "Multi-sensor monitoring system in chemical mechanical planarization (cmp) for correlations with process issues," *CIRP annals*, vol. 55, no. 1, pp. 325–328, 2006.

[8] X. Li, B. Lim, J. Zhou, S. Huang, S. Phua, K. Shaw, and M. Er, "Fuzzy neural network modelling for tool wear estimation in dry milling operation," in *Annual conference of the prognostics and health management society*, 2009, pp. 1–11.

[9] A. Khormali and J. Addeh, "A novel approach for recognition of control chart patterns: Type-2 fuzzy clustering optimized support vector machine," *ISA transactions*, vol. 63, pp. 256–264, 2016.

[10] F. Serdio, E. Lughofer, K. Pichler, T. Buchegger, and H. Efendic, "Residual-based fault detection using soft computing techniques for condition monitoring at rolling mills," *Information Sciences*, 2014.

[11] F. Serdio, E. Lughofer, K. Pichler, M. Pichler, T. Buchegger, and H. Efendic, "Fuzzy fault isolation using gradient information and quality criteria from system identification models," *Information Sciences*, vol. 316, pp. 18–39, 2015.

[12] U. Stanczyk and L. Jain, *Feature Selection for Data and Pattern Recognition*. Heidelberg, New York, London: Springer, 2016.

[13] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, 2003.

[14] M. Pratama, J. Lu, E. Lughofer, G. Zhang, and M. Er, "Incremental learning of concept drift using evolving type-2 recurrent fuzzy neural network," *IEEE Transactions on Fuzzy Systems*, 2017.

[15] I. Khamassi, M. Sayed-Mouchaweh, M. Hammami, and K. Ghedira, "Discussion and review on evolving data streams and concept drift adapting," *Evolving Systems*, vol. 9, no. 1, pp. 1–23, 2017.

[16] M. Pratama, E. Dimla, T. Tjahjowidodo, E. Lughofer, and W. Pedrycz, "Online tool condition monitoring based on parsimonious ensemble+," *IEEE Transactions on Cybernetics*, vol. on-line and in press, 2018.

[17] M. Pratama, W. Pedrycz, and E. Lughofer, "Evolving ensemble fuzzy classifier," *IEEE Transactions on Fuzzy Systems*, 2018.

[18] E. Lughofer, A.-C. Zavoianu, R. Pollak, M. Pratama, P. Meyer-Heye, H. Zörrer, C. Eitzinger, J. Haim, and T. Radauer, "Self-adaptive evolving forecast models with incremental PLS space updating for on-line prediction of micro-fluidic chip quality," *Engineering Applications of Artificial Intelligence*, vol. 68, pp. 131–151, 2018.

[19] E. Lughofer, C. Cernuda, S. Kindermann, and M. Pratama, "Generalized smart evolving fuzzy systems," *Evolving Systems*, vol. 6, no. 4, 2015.

[20] E. Lughofer, A. Zavoianu, R. Pollak, M. Pratama, P. Meyer-Heye, H. Zörrer, C. Eitzinger, and T. Radauer, "Autonomous supervision and optimization of product quality in a multi-stage manufacturing process based on self-adaptive prediction models," *Journal of Process Control*, vol. 76, pp. 27–45, 2019.

[21] E. Lughofer and M. Sayed-Mouchaweh, *Predictive Maintenance in Dynamic Systems — Advanced Methods, Decision Support Tools and Real-World Applications*. New York: Springer, 2019.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[23] Y. Guo, Z. Wu, and Y. Ji, "A hybrid deep representation learning model for time series classification and prediction," in *2017 3rd International Conference on Big Data Computing and Communications (BIGCOM)*. IEEE, 2017, pp. 226–231.

[24] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, "Deep learning and its applications to machine health monitoring," *Mechanical Systems and Signal Processing*, vol. 115, pp. 213–237, 2019.

[25] M. Pratama, C. Za'in, A. Ashfahani, Y. S. Ong, and W. Ding, "Automatic construction of multi-layer perceptron network from streaming examples," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1171–1180.

[26] J. Gama, *Knowledge Discovery from Data Streams*, 1st ed. Chapman & Hall/CRC, 2010.

[27] E. Cazalas, B. K. Sarker, I. Childres, Y. P. Chen, and I. Jovanovic, "Modulation of graphene field effect by heavy charged particle irradiation," *Applied Physics Letters*, vol. 109, no. 25, p. 253501, 2016.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[29] J. Zhang, T. Liu, and D. Tao, "An information-theoretic view for deep learning," *arXiv preprint arXiv:1804.09060*, 2018.

[30] S. Du, T. Li, X. Gong, and S.-J. Horng, "A hybrid method for traffic flow forecasting using multimodal deep learning," *International Journal of Computational Intelligence Systems*, vol. 13, no. 1, p. 85, 2020.

[31] J. F. MacGregor and T. Kourti, "Statistical process control of multivariate processes," *Control Engineering Practice*, vol. 3, no. 3, pp. 403–414, 1995.

[32] I. Frías-Blanco, J. del Campo-Ávila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-Díaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on hoeffding's bounds," *IEEE Transactions on Knowledge and Data Engineering*, 2014.

[33] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[34] M. Moshtaghi, S. Rajasegarar, C. Leckie, and S. Karunasekera, "Anomaly detection by clustering ellipsoids in wireless sensor networks," in *2009 International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*. IEEE, 2009, pp. 331–336.

[35] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," *arXiv preprint arXiv:1708.01547*, 2017.

[36] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[37] D. Sahoo, Q. Pham, J. Lu, and S. C. Hoi, "Online deep learning: Learning deep neural networks on the fly," *arXiv preprint arXiv:1711.03705*, 2017.

[38] C. Oza Nikunj and J. Russell Stuart, "Online bagging and boosting. jaakkola tommi and richardson thomas, editors," in *Eighth International Workshop on Artificial Intelligence and Statistics*, 2001, pp. 105–112.

[39] N. Wang, S. Li, A. Gupta, and D.-Y. Yeung, "Transferring rich feature hierarchies for robust visual tracking," *arXiv preprint arXiv:1501.04587*, 2015.

[40] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "Amc: Automl for model compression and acceleration on mobile devices," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[41] S. Virtanen and M. Girolami, "Precision-recall balanced topic modelling," in *Advances in Neural Information Processing Systems*, 2019, pp. 6750–6759.

[42] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[43] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Transactions on knowledge and data engineering*, vol. 17, no. 4, pp. 491–502, 2005.

[44] E. Belilovsky, M. Eickenberg, and E. Oyallon, "Greedy layerwise learning can scale to imagenet," in *International Conference on Machine Learning*, 2019, pp. 583–593.

[45] E. B. Baum and D. Haussler, "What size net gives valid generalization?" in *Advances in neural information processing systems*, 1989, pp. 81–90.

Autonomous Deep Quality Monitoring
in Streaming Environments

Andri Ashfahani
School of Computer Science and
Engineering
Nanyang Technological University
Singapore
Email: andriash001@e.ntu.edu.sg

Mahardhika Pratama
School of Computer Science and
Engineering
Nanyang Technological University
Singapore
Email: mpratama@ntu.edu.sg

Edwin Lughofer
Department of Knowledge-Based
and Mathematical Systems
Johannes Kepler University
Austria

E. Y. K. Yee
Singapore Institute of Manufacturing Technology
Singapore

Abstract—**This document contains the supplementary material of our paper titled Autonomous Deep Quality Monitoring in Streaming Environments. It explains the injection molding machine which is involved in our experiment.**

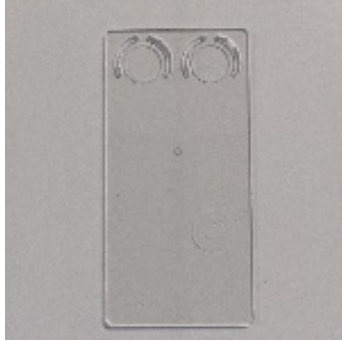## I. INJECTION MOLDING MACHINE

The injection molding machine is usually used to fabricate plastic products such as plastic trinkets, toys to automotive body parts, cell phone cases, water bottles, and containers [**?**]. The raw material, plastic, is injected through a nozzle into a mold cavity cooled and hardened as per the layout of the cavity [**?**]. Our experiment here makes use of the injection molding machine for the production of transparent mold as exhibited in Fig. 1. This machine possesses many parameters where 48 of which are considered here and are listed in the appendix. Two important parameters, namely the holding pressure and the injection speed, are varied here to simulate concept drifts. That is, the holding pressure is varied to be 900, 700, 500, 300, 100 psi while the injection speed is set to be 60, 70, 80, 90, 100 rpm.

Our online quality monitoring problem in this study considers two implementation scenarios: one-step-ahead and current-batch prediction problems. The former scenario requires a model to predict the next batch output $\hat{Y}_{k+1}$ based on the current batch sensor data X_k. It aims to estimate the possible next batch mold quality given the current batch machine condition from sensors. The latter problem focuses on predicting current batch mold quality $\hat{Y}_k$ based on the current batch sensor and image data. It functions as a monitoring tool where the operator is able to know the mold quality without turning-off the molding process. Both problems are multi-class classification problems consisting of 3 classes, i.e. good, weaving and short-forming. The number of data in good, weaving and short-forming classes is 1008, 1074 and 870, respectively. All those three mold qualities are displayed in Fig. 2. It is RGB image with size $150 \times 150 \times 3$. In practice, the injection molding machine requires around 9 minutes to complete a batch of molding process.

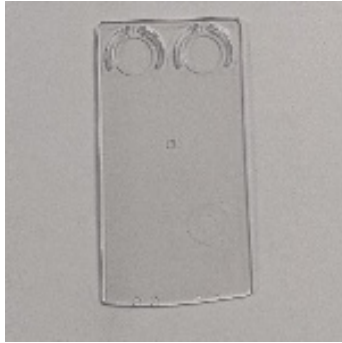Fig. 1. The injection molding machine used in this study.

Table I presents the list of input features captured from sensors. There are 48 input features from sensors utilized to predict the mold quality. To introduce concept changes, 2 variables were varied. Those are holding pressure and injection speed. Because of these variations, 3 classes indicating the mold quality were spotted by the operator. The observed qualities were normal, weaving and short-forming. All of those are presented in Fig. 2. Finally, MinMax normalization was applied to have all features falling within the same range.

Fig. 2. The transparent mold quality produced by the injection molding machine: (a) Normal, (b) weaving and (c) short-forming. Weaving class indicates that the mold surface is uneven, whereas short-forming class specifies that the mold shape is not in a perfect rectangular shape.

TABLE I
THE LIST OF FEATURES CAPTURED FROM SENSORS

NUMBER	DESCRIPTION
1	SCREW VOLUME, ACTUAL VALUE
2	MATERIAL CUSHION, ACTUAL VALUE
3	DOSAGE TIME, ACTUAL VALUE
4	CYCLE TIME, ACTUAL VALUE
5	MOULD HEATING CIRCUIT 1, ACTUAL VALUE
6	MOULD HEATING CIRCUIT 2, ACTUAL VALUE
7	MOULD HEATING CIRCUIT 3, ACTUAL VALUE
8	MOULD HEATING CIRCUIT 4, ACTUAL VALUE
9	MOULD HEATING CIRCUIT 5, ACTUAL VALUE
10	MOULD HEATING CIRCUIT 6, ACTUAL VALUE
11	PLOTTING POINT 2 (HOLDING PRESSURE)*
12	INJECTION SPEED*
13	INJECTION FLOW, ACTUAL VALUE
14	SWITCH-OVER VOLUME, ACTUAL VALUE
15	MAXIMUM INJECTION PRESSURE, ACTUAL VALUE
16	INJECTION TIME, ACTUAL VALUE
17	CYLINDER HEATING ZONE 1, ACTUAL VALUE
18	CYLINDER HEATING ZONE 2, ACTUAL VALUE
19	CYLINDER HEATING ZONE 3, ACTUAL VALUE
20	CYLINDER HEATING ZONE 4, ACTUAL VALUE
21	CYLINDER HEATING ZONE 5, ACTUAL VALUE
22	OPENING FORCE, ACTUAL VALUE
23	OPENING SPEED, ACTUAL VALUE
24	OIL TEMPERATURE, ACTUAL VALUE
25	MOULD TEMPERATURE CONTROL UNIT 1, ACTUAL VALUE
26	NOZZLE STROKE, ACTUAL VALUE
27	CLOSING SPEED, ACTUAL VALUE
28	ADVANCEMENT SPEED, ACTUAL VALUE
29	RETRACTION SPEED, ACTUAL VALUE
30	MOULD PROTECTION FORCE, ACTUAL VALUE
31	TEMPERATURE OF SUPPORT HOUSING, ACTUAL VALUE
32	CIRCUMFERENTIAL SPEED, ACTUAL VALUE
33	EJECTOR PRESSURE, NOMINAL VALUE
34	EJECTOR PRESSURE, ACTUAL VALUE
35	NOZZLE 1 FLOW, NOMINAL VALUE
36	NOZZLE 1 PRESSURE, ACTUAL VALUE
37	INJECTION TORQUE, ACTUAL VALUE
38	INJECTION ROTATIONAL SPEED, ACTUAL VALUE
39	INJECTION FORCE OF SCREW 1, ACTUAL VALUE
40	DOSAGE TORQUE, ACTUAL VALUE
41	DOSAGE ROTATIONAL SPEED, ACTUAL VALUE
42	HYDRAULIC ACCUMULATOR PRESSURE, ACTUAL VALUE
43	CHARGE PRESSURE OF ACCUMULATOR, MEASURED VALUE
44	MOULD-ENTRY TIME, ACTUAL VALUE
45	PART REMOVAL TIME, ACTUAL VALUE
46	MAXIMUM INJECTION PRESSURE, ACTUAL VALUE
47	BACK PRESSURE, ACTUAL
48	CLAMPING FORCE, ACTUAL

*: Indicate the varied features to introduce concept change.