

## Sequential Training Data Production Allocation for AI-based Abnormal Event Diagnosis in Nuclear Power Plants

Seung Gyu Cho, Seung Jun Lee\*

Ulsan National Institute of Science and Technology, 50, UNIST-gil, Ulsan, 44911

\*Corresponding author: sjlee420@unist.ac.kr

**\*Keywords :** training data production allocation, sequential allocation, abnormal event diagnosis, nuclear power plant

### 1. Introduction

Artificial intelligence (AI)-based diagnosis of abnormal events in nuclear power plants (NPPs) has been studied extensively [1]. Its data situation, however, is unusual: abnormal-event data from operating plants are essentially unavailable, so every training file must be produced on a simulator. Producing a single scenario is not free; beyond the simulation run itself, it requires human effort for scenario setup, malfunction injection, and review. The production budget is therefore finite, and a planning question arises before any model is trained: if only  $n$  files can be produced, how many should be produced for each event class? Current practice produces the same number for every class ( $n/C$  files for  $C$  classes), an unjustified default whose allocation itself has never been validated.

Prior work on training-data optimization has been inductive and post-hoc: an optimal subset is selected out of an already-existing pool. Active learning is the representative paradigm, querying the most informative samples from an unlabeled pool to reduce labeling cost [2], and it has recently been applied to NPP condition diagnosis, where a Bayesian-neural-network-based information metric selects informative training samples and halves the samples needed for equivalent performance [3]. In every such approach, however, the pool must exist before selection begins: for simulator-produced diagnosis data the production cost has already been fully paid, and no production saving is possible. What is missing is a methodology for the production-planning stage, which decides the allocation before the data exist.

This study asks whether observations accumulated sequentially, starting from a small pilot, can decide (i) how many files to produce per class (allocation) and (ii) when to stop producing (stopping). The proposed method contributes: (1) a deficit-proportional sequential allocation in which observed per-class deficits shape the production distribution; (2) a self-signaled stopping rule that never references full-data performance; and (3) actionable by-products, namely a production-request list and a data-quality audit. On three simulator datasets, the proposed method maintained full-data diagnostic accuracy while producing only 9.4–57% of the files, shortening the total data-preparation time by up to a factor of 9.

### 2. Methodology

The proposed framework is a sequential procedure of five steps, summarized in Fig. 1. Starting from a minimal pilot, the procedure alternates between producing a small batch, training an ensemble, and reallocating the next batch according to the deficits it has just observed.

#### 2.1. Pilot Production

Two files per class are produced uniformly. Two is the measured minimum for observation: with a single file, a bad representative collapses its class, whereas with two files the bad-representative risk drops and intra-class variation exists, so deficits become signal rather than noise. Because two files per class is itself the first rung of a uniform schedule, the comparison with uniform production starts from an identical state. As a guard rule, no validation split is made for classes holding fewer than five files. The pilot is for observation, not performance; it is the minimum investment that draws the deficit map

#### 2.2. Training and Deficit Measurement

Each round trains an ensemble of 10 models, one per random seed, and measures two per-class metrics. Confirmed accuracy ( $conf_{acc}$ ) is the fraction of test files whose diagnosis is confirmed correctly, where a diagnosis is confirmed after five consecutive seconds of the same abnormal prediction. Per-second accuracy ( $step_{acc}$ ) is the correct fraction of every-second predictions and serves as a continuous auxiliary signal. The corresponding deficits are  $conf_d = 1 - conf_{acc}$  and  $step_d = 1 - step_{acc}$ .

#### 2.3. Class Judgment

Deficits are compared not to fixed thresholds but to each class's own noise floor  $\varepsilon = \max(\text{SEM}, 0.005)$ , computed per metric. A class is active, and thus eligible for further production, if  $conf_d > \varepsilon_c$  or ( $conf_d > 0.005$  and  $step_d > \varepsilon_s$ ); the dual signal catches deficits hidden by stepwise noise. A class whose deficit is within its noise floor is satisfied. A class showing two consecutive ineffective increments ( $\Delta conf \leq 0$  and  $\Delta step \leq 0$  after added files) is frozen and receives no further production, and a class whose available pool is used up is exhausted.

#### 2.4. Budget Allocation

Each round produces a fixed quota  $Q$  of new files, set to the difference between the next rung of the uniform ladder and the files already held; for example, moving from two to four files per class over  $C$  classes gives  $Q = 2C$ . The sequential production therefore spends exactly the same budget as the uniform schedule at every step, and only the distribution differs.  $Q$  is split over the active classes in proportion to their confirmed deficits,  $share = Q \times conf_d \div \sum conf_d$ , integerized by flooring and assigning the remaining files to the largest fractional parts. Each class's share is capped at +10 files per round and at its remaining pool availability; whenever a proportional share exceeds either cap, the excess becomes an overflow that is redistributed to the other active classes in the same proportional manner. If files are still left after every active class is served, they are spent as exploration probes: one file per round (at most two in total per class) to satisfied classes whose  $step_d$  still exceeds 0.005, guarding against deficits hidden by the confirmed metric. In the first allocation round of one experiment (quota 36), for example, class A's proportional share exceeded the round cap, so it received +10 and its excess overflowed to the others;

class B (deficit 0.35) received +8; class C (0.29) +7, its own share of 6 plus the one file redistributed from class A's cap; class D (0.14) +3; and the four remaining active classes +2 each. The allocation is purely a function of observed deficits.

### 2.5. Stop Logic

After each judgment the procedure asks whether any active class remains. If none does, production stops by self-signal (Stop A): every class is satisfied, frozen, or exhausted, there is no reason left to produce, and the remaining budget is unspent cost saved. If active classes remain but no budget rung does, the procedure stops at the budget ceiling (Stop B) and the remaining active classes form a priority list for any future budget increase. Note that neither rule ever references full-data performance. Besides the executed allocation recipe, the procedure emits two outputs that require human action: production requests for exhausted-but-deficient classes, and a quality audit of detected data defects and detection limits.

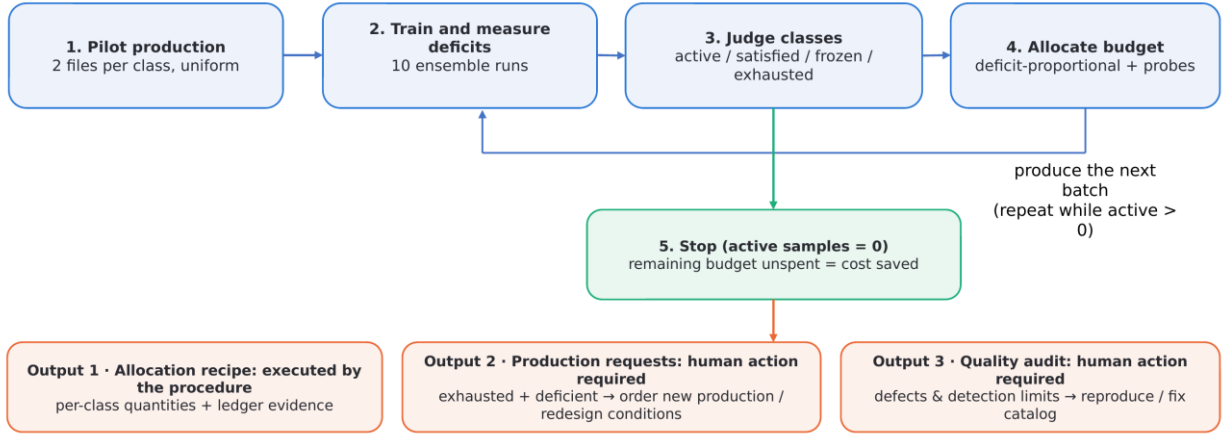


Fig. 1. The sequential production-allocation framework and its three outputs.

## 3. Experimental Settings

The methodology was evaluated on three simulators of different scale (Table I): the 3KEYMASTER full-scope simulator (3KM), the KINGS full-scope simulator, and the compact nuclear simulator (CNS), for which only classes holding at least 50 files were used. The diagnosis model is a 1D convolutional neural network [1]; the methodology itself is model-agnostic. Every condition was run 10 times with 10 distinct random seeds, and each run's seed is reused identically across all conditions and stages, so runs are paired; all comparisons use the paired  $z$  statistic on same-run differences with  $|z| \geq 1.96$  as the significance criterion. The primary metric is macro-averaged confirmed accuracy under the 5-s rule, with per-second accuracy as an auxiliary.

Throughout this paper, abnormal-event classes are denoted by system-based labels of the form SYSTEM[failure mode], and Table II defines every class label that appears. Because the procedure identifiers of

the KINGS dataset are confidential, its events are mapped to the same labeling scheme.

Table I. Datasets and budget ladders.

| Item                 | 3KM                               | KINGS                    | CNS                                |
|----------------------|-----------------------------------|--------------------------|------------------------------------|
| Scope                | Full                              | Full                     | Compact                            |
| Classes              | 28                                | 53                       | 18                                 |
| Full pool (files)    | 1,044                             | 764                      | 880                                |
| Test files           | 450                               | 118                      | 220                                |
| Variables            | 636                               | 785                      | 574                                |
| Ladder (files/class) | 2→3→4→<br>5→7→10<br>→14→20<br>→40 | 2→4→6→<br>8→10→12<br>→14 | 2→4→6→<br>8→10→14<br>→20→28<br>→40 |

**Table II.** Abnormal-event class labels referred to in this paper.

| Label     | Event                                                               |
|-----------|---------------------------------------------------------------------|
| CDS[LV]   | Loss of condenser vacuum                                            |
| LTDN[LK]  | Letdown line leakage                                                |
| MSIV[VC]  | Main steam isolation valve positioner failure                       |
| POSRV[VO] | Pilot-operated safety relief valve leakage                          |
| PZR[VO]   | Pressurizer spray valve positioner failure                          |
| PZR[SF]   | Pressurizer level input-signal failure (automatic channel transfer) |
| RCP[LC]   | Loss of component cooling water to the RCP                          |
| RCP[LS]   | Loss of RCP seal injection                                          |
| RVHF[LK]  | Reactor vessel head flange leakage                                  |

## 4. Results

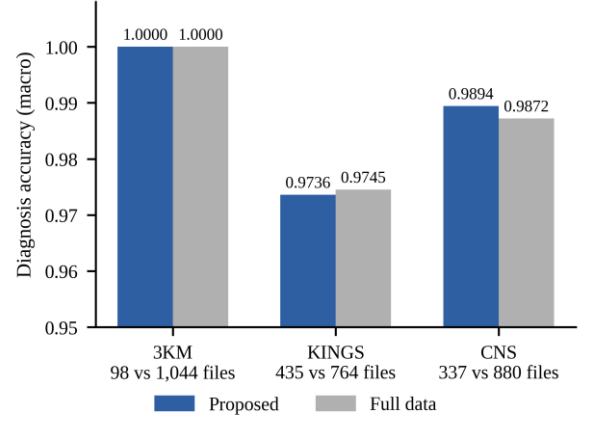
### 4.1. Equal-Budget Performance

Fig. 2 and Table III summarize the outcome. On 3KM the procedure self-stopped after producing 98 files, 9.4% of the current full production of 1,044, and reached perfect macro confirmed accuracy in 10 of 10 runs. At the previous round, where both schedules had produced 82 files, the procedure had already reached 0.9996 while uniform production remained at 0.9893 (paired  $z = +9.0$ ); uniform production reached the same perfect accuracy only at 264 files, nearly three times the procedure's total. On KINGS the procedure stopped at 435 files (57% of 764) with 97.36% accuracy, statistically indistinguishable from the full-data 97.45% (paired  $z = -0.56$ ), while outperforming uniform production at every ladder step (paired  $z = +2.8$  to  $+8.5$ ). On CNS the procedure stopped at 337 files (38% of 880) with 98.94%, nominally above the full-data 98.72% (paired  $z = +1.08$ ), and significantly above uniform production at several rounds (paired  $z = +3.2$  to  $+4.4$ ).

The allocations concentrate where deficits were observed. On CNS, the classes CDS[LV], POSRV[VO], LTDN[LK], and MSIV[VC] grew to 36–40 files, while classes that were already perfect at the pilot, such as PZR[VO] and RCP[LC], remained at their two pilot files. On 3KM, only the deficient classes grew and all 28 classes were perfect at self-stop, with zero production requests and zero freezes.

**Table III.** Sequential production vs. full production (macro confirmed accuracy, mean of 10 runs).

| Sim.  | Files used (share) | Proposed | Full data | $z$   |
|-------|--------------------|----------|-----------|-------|
| 3KM   | 98 / 1,044 (9.4%)  | 1.0000   | 1.0000    | —     |
| KINGS | 435 / 764 (57%)    | 0.9736   | 0.9745    | -0.56 |
| CNS   | 337 / 880 (38%)    | 0.9894   | 0.9872    | +1.08 |

**Fig. 2.** Diagnosis accuracy of the proposed sequential production at self-stop vs. full-data production.

### 4.2. Unresolved Classes Become Work Orders

Classes the procedure cannot fix are not silent failures; their exit path identifies the appropriate remedy (Table IV). On KINGS, class PZR[SF] (pressurizer level control, single input-signal failure with automatic channel transfer) never responded to added data and was misconfirmed in all runs as the closely related event of pressurizer level increase, indicating overlapping signatures; it is auto-listed as the top-priority production request and a candidate for scenario redesign. Class RVHF[LK] (reactor vessel head flange leakage) did respond (+0.30 confirmed accuracy) before its 17-file pool was exhausted, so further production has a high expected return. On CNS, the lowest-severity files of LTDN[LK] and POSRV[VO] are confirmed as Normal, revealing a detection limit; the automated remedy is to reproduce these events at lower severities. RCP[LS] was frozen after two ineffective increments, saving 15 files, and its residual failure is a boundary case with MSIV[VC] that more data cannot fix. Notably, the failures are structural rather than random: across 10 runs there were zero no-diagnosis failures on KINGS, and misconfirmations concentrate among closely related events of the same system.

**Table IV.** Representative unresolved classes and the automatically generated actions.

| Class            | Exit      | Used/a vail. | Acc. | Action                 |
|------------------|-----------|--------------|------|------------------------|
| PZR[SF] (KINGS)  | Exhausted | 6/6          | 0.00 | Produce more; redesign |
| RVHF[LK] (KINGS) | Exhausted | 17/17        | 0.60 | Produce more           |
| LTDN[LK] (CNS)   | Exhausted | 40/40        | 0.91 | Lower severity         |
| POSRV[VO] (CNS)  | Exhausted | 40/40        | 0.97 | Lower severity         |
| RCP[LS] (CNS)    | Frozen    | 25/40        | 0.93 | No data response       |

### 4.3. Cost

With measured unit times of roughly 2 min per produced file on 3KM, 4 min on KINGS, 80 s on CNS,

and 45 s per trained model, the procedure's added training time is negligible against the production it avoids: 0.5 h of training vs. 31.5 h of avoided production on 3KM (946 of 1,044 files avoided,  $\geq 67\times$ ), 0.6 h vs. 21.9 h on KINGS (329 of 764 avoided,  $\geq 38\times$ ), and 0.7 h vs. 12.1 h on CNS (543 of 880 avoided,  $18\times$ ) — in wall-clock terms, one full sequential run costs only 28, 36, and 41 min of training on 3KM, KINGS, and CNS, respectively (RTX 3080). The total time, production plus training, shortens from 34.8 h to 3.7 h on 3KM (9.3 $\times$ ), from 50.9 h to 29.6 h on KINGS (1.7 $\times$ ), and from 19.6 h to 8.2 h on CNS (2.4 $\times$ ). Training overhead would begin to matter only if a file could be produced in a few seconds, far below realistic scenario production. Fig. 3 and Table V summarize the measured time cost of the two paths.

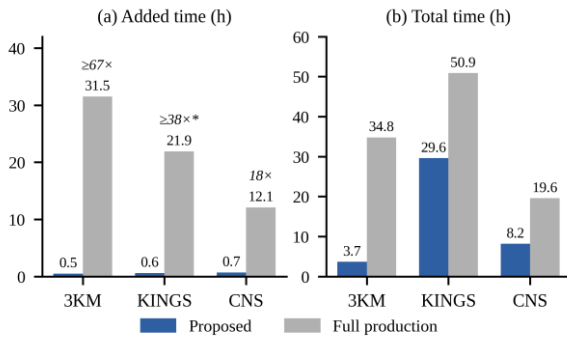


**Fig. 3.** Time cost of the proposed sequential production vs. full production: (a) training time added by the procedure against production time avoided, (b) total time.

**Table V.** Measured time cost: sequential production vs. full production.

| Item                        | 3KM             | KINGS            | CNS            |
|-----------------------------|-----------------|------------------|----------------|
| Production time per file    | $\approx 2$ min | $\approx 4$ min* | $\approx 80$ s |
| Training time per model     | 45 s            | 45 s             | 45 s           |
| Files avoided               | 946             | 329              | 543            |
|                             | (91%)           | (43%)            | (62%)          |
| Training time added (h)     | 0.5             | 0.6              | 0.7            |
| Production avoided (h)      | 31.5            | 21.9             | 12.1           |
| Total time — proposed (h)   | 3.7             | 29.6             | 8.2            |
| Total time — full prod. (h) | 34.8            | 50.9             | 19.6           |

\* Production time per file on KINGS is an estimate.

## 5. Conclusions

This study proposed a sequential production-allocation methodology that decides, before the data exist, how many training files to produce per abnormal-event class and when to stop producing. Starting from a two-file pilot, the procedure allocates each round's quota in proportion to observed per-class deficits, judges classes against their own noise floors, and stops by self-signal without ever referencing full-data performance. Across three simulators the procedure matched full-data diagnostic performance with 9.4% (3KM), 57% (KINGS), and 38% (CNS) of the current uniform production, while meeting or exceeding uniform allocation at equal budget and cutting the total data-preparation time by factors of 1.7–9.3. Beyond the

allocation itself, the procedure turns residual deficits into actionable outputs: a prioritized production-request list and a data-quality audit that localizes detection limits. Future work includes severity-aware production requests and application of the framework to the production planning of new plant datasets.

## ACKNOWLEDGEMENTS

This work was supported by KOREA HYDRO & NUCLEAR POWER CO., LTD (No. 2024-Tech-09)

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. RS-2022-00144042)

## REFERENCES

- [1] J. H. Shin, J. M. Kim, and S. J. Lee, "Abnormal state diagnosis model tolerant to noise in plant data," *Nuclear Engineering and Technology*, Vol. 53, No. 4, pp. 1181-1188, 2021.
- [2] B. Settles, *Active Learning Literature Survey*, Computer Sciences Technical Report 1648, University of Wisconsin-Madison, 2009.
- [3] Y. H. Chae, S. R. Koo, J. Choi, and J. Kim, "Enhanced learning for nuclear power plant condition diagnoses using information metric based on Bayesian neural networks and UMAP," *Nuclear Engineering and Technology*, Vol. 58, 103886, 2026.