applied sciences

MDPI

Article

A 280-Parameter Embedding-Constrained Network for Autonomous Rate-of-Penetration Optimization on a Microcontroller

Zhi Geng [1,*], Changmin Fu [2], Bo Zheng [3], Zhijing Bai [1] and Yanfei Wang [1,*]

[1] Key Laboratory of Deep Petroleum Intelligent Exploration and Development, Institute of Geology and Geophysics, Chinese Academy of Sciences, Beijing 100029, China
[2] CAS Engineering Laboratory for Deep Resources Equipment and Technology, Institute of Geology and Geophysics, Chinese Academy of Sciences, Beijing 100029, China
[3] Key Laboratory of Shale Gas and Engineering, Institute of Geology and Geophysics, Chinese Academy of Sciences, Beijing 100029, China
* Correspondence: gengzhi@mail.iggcas.ac.cn (Z.G.); yfwang@mail.iggcas.ac.cn (Y.W.)

Abstract

Rate of penetration (ROP) prediction is essential for drilling efficiency, yet prevailing deep learning models employ millions of parameters and rely on surface-based pipelines constrained by low-bandwidth telemetry and round-trip delays measured in minutes to hours. These delays make surface-computed optimization decisions obsolete before they reach the bit, leaving real-time drilling intelligence infeasible. We present ROP MCU_model, an encoder–regressor neural network with only 280 trainable parameters that performs autonomous ROP prediction and optimization entirely on a microcontroller. Discovered through genetic-algorithm neural architecture search and trained with supervised contrastive learning, the model imposes structural constraints on the latent embedding space to maintain accuracy under extreme parameter scarcity. Full-integer quantization compresses the network to 5.57 KB, enabling inference in 156 µs on an ESP32-S3 via TensorFlow Lite Micro. An integrated on-device optimization engine searches controllable drilling parameters within a bounded search envelope without surface communication. Validated under leave-two-well-out cross-validation on field data from the Taranaki Basin, the model achieves a mean weighted mean absolute percentage error (WMAPE) of 0.687 over nine seeded evaluation splits, the lowest prediction error among all competing models. Recomputing all normalization bounds from training-stage wells only reproduces this result within 0.2%, indicating no split-dependent leakage and a negligible effect. The on-device optimizer delivers a mean surrogate-model-predicted ROP improvement of 46.2% (median 18.5%). This work demonstrates closure of the sense–predict–optimize–decide decision loop on a commodity microcontroller; transferring the methodology to other telemetry-constrained downhole tasks remains future work.

Keywords: edge AI; TinyML; rate of penetration; drilling optimization; microcontroller

PACS: J0101

Check for updates

Academic Editor: Dan Sui

Received: 5 August 2026
Revised: 19 October 2026
Accepted: 19 October 2026
Published: 22 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

The rate of penetration (ROP) achieved during rotary drilling operations is among the most consequential determinants of well construction economics, with drilling time representing 40–60% of total expenditure in deep and complex wells. Optimizing ROP in real time has long been a central objective of drilling engineering. Yet the feasibility of real-time ROP optimization is constrained by a fundamental physical bottleneck: under current

industrial practice, high-fidelity downhole measurements must traverse thousands of meters of drill string via mud-pulse telemetry, a channel with effective bandwidth typically below 10 bits per second [1] and degrading to single-digit bits per second in extended-reach wells [2]. The resulting round-trip delay is measured in minutes; by the time a surface-computed recommendation returns to the bit, the drill string may have penetrated several meters of new formation. This latency barrier affects not only ROP economics but also the responsiveness of any downhole decision support, underscoring a broader architectural question: where should the decision-making node reside when telemetry cannot support real-time surface control [3–5]? Even wired drill pipe (57,600 bps) [6,7] does not eliminate this constraint, because the optimization algorithm still runs on surface servers and the control loop remains bounded by the physical distance between bit and rig floor. At the surface, data aggregation systems further degrade the signal by downsampling high-frequency sensor streams, discarding the vast majority of original measurements. The bottleneck is not data acquisition; it is the absence of decision-making intelligence at the point of measurement.

This study takes ROP optimization as its specific object: it is one of the most extensively researched drilling optimization problems, supported by public datasets and established validation protocols that enable rigorous benchmarking. By solving ROP prediction and parameter optimization entirely on a near-bit microcontroller, we provide a concrete, reproducible demonstration that autonomous edge decision-making is computationally viable in drilling operations. The methodological pipeline, comprising hardware-aware neural architecture search, supervised contrastive learning for regression, 8-bit integer (INT8) quantization, and on-device optimization, is generalizable in principle to other telemetry-constrained downhole tasks, although each application will require task-specific feature engineering and safety validation.

Machine learning methods have substantially improved ROP prediction accuracy over classical physical models. Data-driven approaches, including support vector regression, tree ensembles, and shallow neural networks, consistently outperform empirical formulations by capturing nonlinear interactions without per-formation recalibration [8–12]. More recent learning-based architectures, such as attention-based recurrent networks [13], Informer-based transformers [14], and physics-constrained hybrid models [15], have pushed predictive performance further, with several studies reporting coefficients of determination exceeding 0.95 [15,16]. However, these advances have deepened a deployability gap rather than closing it. The prevailing deep learning paradigms for ROP prediction employ hundreds of thousands to tens of millions of parameters and rely on sequential input windows to model temporal dependencies, such as long short-term memory (LSTM) hidden states, gated recurrent unit (GRU) memory vectors, or transformer attention caches, that consume runtime memory proportional to sequence length. Even under aggressive quantization, their combined parameter and activation footprints remain in the tens-of-kilobytes to megabytes range, far beyond the tight kilobyte-scale budgets feasible for near-bit MCU (microcontroller unit) deployment. None of these studies characterize inference latency, memory footprint, or edge deployability; computational infrastructure is treated as an external given, leaving a critical gap between algorithmic performance and operational realizability.

Parallel advances in model compression, efficient kernel design, and embedded inference frameworks have made neural network inference feasible on microcontroller-class hardware. Lin et al. [17] demonstrated ImageNet-level accuracy on off-the-shelf commercial microcontrollers through system–algorithm co-design; Jacob et al. [18] formalized INT8 full-integer quantization, which reduces model memory footprint by approximately 4× and substantially reduces inference latency; and David et al. [19] developed TensorFlow Lite Micro, an inference runtime engineered for deeply embedded systems with no op-

erating system and no dynamic memory allocation. Standardized benchmarks [20] and optimized ARM kernels [21] have matured the TinyML ecosystem across image classification and keyword-spotting domains [22–24]. Despite this progress, two gaps persist that are particularly relevant to industrial sensing. First, existing TinyML deployments focus overwhelmingly on classification tasks with discrete output spaces; resource-constrained regression, where predictions must respect continuous physical scales, ordinal relationships, and operational safety bounds, remains largely unexplored. Second, the intersection of TinyML and drilling optimization is almost entirely vacant. To the authors' knowledge, no published work has demonstrated a neural network for ROP prediction that simultaneously achieves competitive accuracy and deployability on a microcontroller with kilobyte-scale memory footprint and sub-millisecond inference. The drilling automation roadmap articulated by Macpherson et al. [25] advocates embedded intelligence and autonomous downhole control, yet the sensor and AI technologies required to fulfill this vision have developed largely in isolation.

Closing this gap demands more than simply compressing an existing architecture. Under an extreme parameter budget of fewer than 1000 trainable weights, standard regression losses such as mean absolute error provide insufficient constraints on the latent embedding space; the optimizer wanders across an unbounded landscape, and run-to-run training variability inflates several-fold because the model lacks any structural mechanism to impose geometric order on the embedding without an explicit memory of previously drilled formations. Supervised contrastive learning, originally developed for classification [26], offers a principled solution: by forcing drilling states of similar ROP-level structure to cluster on a well-behaved manifold, the embedding geometry itself encodes the missing contextual information. Its adaptation to regression, however, requires an order-aware loss that respects the ordinal structure of ROP magnitudes, an extension that has not previously been explored under kiloparameter constraints. Likewise, discovering an architecture that remains expressive under such extreme parameter constraints is not amenable to manual design. Neural architecture search [27,28], including hardware-aware variants [29,30], provides a computational route to such topologies, yet no prior work has combined genetic-algorithm neural architecture search (NAS) with supervised contrastive learning for resource-constrained industrial regression.

Three research questions guide our investigation. RQ1: Can a neural network with fewer than 1000 parameters achieve ROP prediction accuracy competitive with shallow machine learning baselines and established deep learning architectures? RQ2: Can such a model execute inference in less than one millisecond on a 240 MHz microcontroller, and if so, what quantization and kernel optimization strategies are required to reach this target? RQ3: Can the prediction–optimization–decision pipeline for ROP be closed entirely on-device, enabling near-bit decision-making without surface communication?

This paper makes four contributions toward answering these questions. First, we show that supervised contrastive learning, adapted to the ordinal structure of continuous ROP targets, acts as a consistent stabilizer rather than an auxiliary regularizer: by anchoring optimization to a well-ordered embedding manifold, it enables stable convergence to competitive accuracy in every tested loss ratio even when the latent dimension is compressed to two units and the parameter budget to 280. The constraint reduces prediction error relative to pure regression across all nine tested contrastive-to-regression weight ratios (WMAPE 0.588–0.639 versus 0.6645 under the set_2 ablation protocol; 11.5% for the best ratio) and reduces the run-to-run standard deviation of training outcomes from 0.0845 to 0.027–0.049. Second, we present a genetic-algorithm neural architecture search methodology that discovers the 280-parameter encoder–regressor on which this constraint operates, approximately 1000× fewer parameters than comparable deep learning models. Third,

we demonstrate the complete INT8 quantization pipeline that compresses the model to 5.57 KB and deploy it on the ESP32-S3 using TensorFlow Lite Micro, achieving 156.38 µs inference latency at 6394 inferences per second, well under the one-millisecond threshold. Fourth, we evaluate the integrated prediction–optimization system on field data from the Taranaki Basin, demonstrating mean surrogate-model-predicted ROP improvements of +46.2% (median +18.5%) across 41 evaluation points when the on-device optimizer is activated. Together, these contributions establish, to the authors' knowledge, one of the first instances of on-device ROP decision generation on a microcontroller-class device. They also demonstrate that kilobyte-scale edge models can generate advisory decisions without surface dependency, establishing a methodological template for extending edge autonomy to other telemetry-constrained drilling tasks.

The remainder of this paper is organized as follows. Section 2 presents the ROP MCU_model architecture, the genetic algorithm-based NAS procedure, the contrastive learning training protocol, the INT8 quantization workflow, and the on-device optimization engine. Section 3 reports the results of accuracy benchmarking, the contrastive learning ablation study, quantization impact analysis, MCU deployment profiling, and field optimization performance. Section 4 discusses the implications of our findings, answers the three research questions with quantitative evidence, and identifies directions for future work. Section 5 concludes the paper.

2. Methodology

2.1. Problem Formulation

This work addresses autonomous decision-making for rate-of-penetration optimization executing entirely on a microcontroller-class edge device. The system functions as an intelligent near-bit advisor that, given the current drilling state, autonomously predicts ROP and determines optimal controllable parameters, generating operational decisions locally without surface communication. The architecture, training protocol, and safety constraints are designed specifically for the ROP prediction task, with the encoder–regressor topology and contrastive learning objective co-evolved to match the geological heterogeneity and operational safety requirements of this problem. The end-to-end workflow is illustrated in Figure 1. The problem is formulated as a constrained encoder–regressor regression pipeline: given an input feature vector $\mathbf{x}_d \in \mathbb{R}^8$ comprising eight drilling parameters, namely Measured Depth (MD), True Vertical Depth (TVD), bit size, weight on bit (WOB), pipe pressure, torque, rotary speed (RPM), and flow rate, an encoder $E_\theta : \mathbb{R}^8 \to \mathbb{R}^{d_{\text{latent}}}$ first projects the drilling state onto a low-dimensional latent manifold, where supervised contrastive learning imposes geometric structure that clusters states exhibiting similar ROP-level structure (the geological proxy). A downstream regressor $R_\phi : \mathbb{R}^{d_{\text{latent}}} \to \mathbb{R}^+$ then predicts the scalar ROP y, representing penetration velocity in meters per hour. The physical constraint $y > 0$ is enforced at all times. The entire inference pipeline, including the forward pass through E_θ and R_ϕ, must execute within sub-millisecond latency on a resource-constrained MCU with kilobyte-scale memory, and the on-device optimization engine must close the prediction–optimization loop without surface communication.

Figure 1 organizes the methodology into four integrated stages. Stage (a) performs genetic-algorithm NAS on the USROP dataset [31] to evolve the encoder–regressor topology under extreme parameter constraints. Stage (b) trains the selected architecture on the target dataset with supervised contrastive learning to structure the latent manifold. Stage (c) quantizes the converged model to INT8 and deploys it on the ESP32-S3 for sub-millisecond inference. Stage (d) closes the loop with an on-device optimization engine that searches controllable drilling parameters to maximize predicted ROP within operational safety bounds.

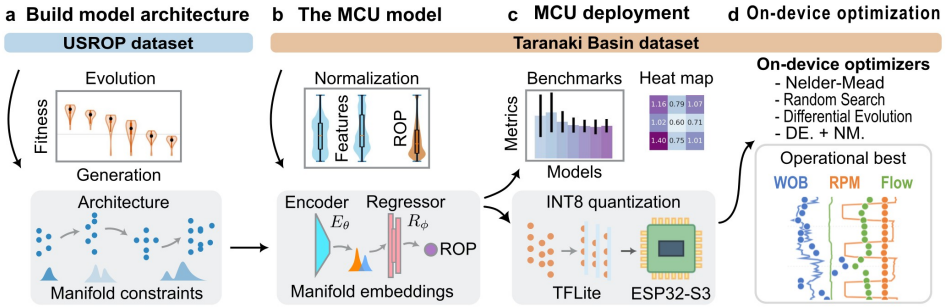


Figure 1. Integrated workflow of the proposed ROP MCU_model. (**a**) Neural architecture search (NAS) on the USROP dataset evolves encoder–regressor topologies under extreme parameter constraints; the discovered architecture transfers directly to the Taranaki Basin scenario. (**b**) The selected encoder E_θ projects drilling states onto a low-dimensional latent manifold structured by supervised contrastive learning, forcing states with similar ROP-level structure to cluster; the downstream regressor R_ϕ maps the structured embedding to scalar ROP. (**c**) The trained model undergoes INT8 full-integer quantization and is deployed on an ESP32-S3 microcontroller via TensorFlow Lite Micro for sub-millisecond on-device inference. (**d**) The MCU-integrated optimization engine searches over controllable drilling parameters (WOB, RPM, and flow rate) to maximize predicted ROP within bounded intervals enforced on-device, while ROP-floor clamping and upstream command authorization belong to the deployment pipeline, delivering advisory operational decisions.

2.2. Dataset and Preprocessing

Two distinct datasets serve complementary roles in this study. The USROP dataset [31], comprising standardized drilling records from seven offshore wells in the Norwegian North Sea Volve field with twelve measured features, provides the evolution arena for the neural architecture search process. Its uniform feature dimensions, consistent unit system, and ample sample size make it suitable for architecture evolution without the confounding effects of unit heterogeneity. The architecture discovered on USROP, including layer topology and activation selections, is subsequently transferred directly to the Taranaki Basin evaluation without re-evolution.

The Taranaki Basin dataset, drawn from eleven wells in New Zealand's Taranaki Basin, forms the primary evaluation benchmark. This dataset presents substantially greater challenges: features span eight drilling parameters, data originate from multiple logging systems with inconsistent units, and the basin's position at an active convergent plate margin produces extreme geological heterogeneity across wells. The dataset's multi-source, multi-unit character makes it a rigorous test of generalization capability.

Raw drilling logs from different sources use mixed units, which must be standardized before modeling. The raw log data contain measurements in ton-force, kips (kilopounds-force), psi, lbf·ft, and gallons per minute. We apply the following conversion factors: ton-force to kilonewtons via $x_{\mathrm{kN}} = x_{\mathrm{ton}} \times 9.80665$; kilopounds-force to kilonewtons via $x_{\mathrm{kN}} = x_{\mathrm{klb}} \times 4.44822$; psi to kilopascals via $x_{\mathrm{kPa}} = x_{\mathrm{psi}} \times 6.89476$; lbf·ft to kilonewton-meters via $x_{\mathrm{kN\cdot m}} = x_{\mathrm{lbf\cdot ft}} \times 0.00135582$; and gallons per minute to liters per minute via $x_{\mathrm{L/min}} = x_{\mathrm{gal/min}} \times 3.78541$. Where multiple measurements represent the same physical quantity, such as pump pressure and standpipe pressure (which differ by less than 5% under normal drilling conditions), they are mapped to a single unified feature (pipe pressure) to eliminate redundancy.

All input features then undergo a min–max normalization with fixed physical bounds that maps each feature to the interval $[\varepsilon, 1+\varepsilon]$, where $\varepsilon = 10^{-8}$ is a small constant introduced to prevent exact zero or unity values that could destabilize neural network training. The normalization bounds are not estimated from any modeling split; they are fixed engineering bounds defined a priori, at the data-standard stage, and applied uniformly to

every well. Their exact values for all eight inputs, together with the ROP target range and the tail-clipping quantiles, are listed in Appendix A. Formally, for each feature dimension $j \in \{1, \ldots, 8\}$,

$$x_{\text{norm}}^{(j)} = \text{clip}\left(\frac{x^{(j)} - q_{\text{low}}^{(j)}}{q_{\text{high}}^{(j)} - q_{\text{low}}^{(j)}},\ \varepsilon,\ 1 + \varepsilon \right) \tag{1}$$

where $q_{\text{low}}^{(j)}$ and $q_{\text{high}}^{(j)}$ denote the fixed lower and upper bounds of feature j: tail-clipped historical operational limits that exclude sensor glitches and extreme states, computed once over the full field-wide dataset before any modeling split and then held fixed for every well and every split (exact values and clipping quantiles in Appendix A). The use of such clipped physical bounds, rather than data-driven min–max limits, provides robustness against outliers that are prevalent in downhole sensor data due to measurement noise, calibration drift, and transient operational events.

These normalization constants constitute a data contract: they are fixed before any modeling begins, carry the same epistemic status as the unit conversions above, and are never re-estimated per well or per split. Evaluation separation is enforced at the whole-well level by the leave-two-well-out protocol of Section 2.5, in which two wells are entirely withheld from training and model selection. The same contract anchors the contrastive learning target: because ROP is normalized by its fixed physical range of 0–80 m/h, the uniform bin edges of the contrastive loss (Section 2.4) are definitionally identical under any well partition. At deployment, the constants ship with the firmware, so the on-device model requires no field statistics at inference time. This design choice is empirically inconsequential: recomputing all bounds for each split from the training-stage wells only, the strictest leakage-avoidance protocol, reproduces the nine-split benchmark within 0.2% with no statistically significant difference (Appendix A).

The spatial distribution of the eleven Taranaki Basin wells is illustrated in Figure 2.

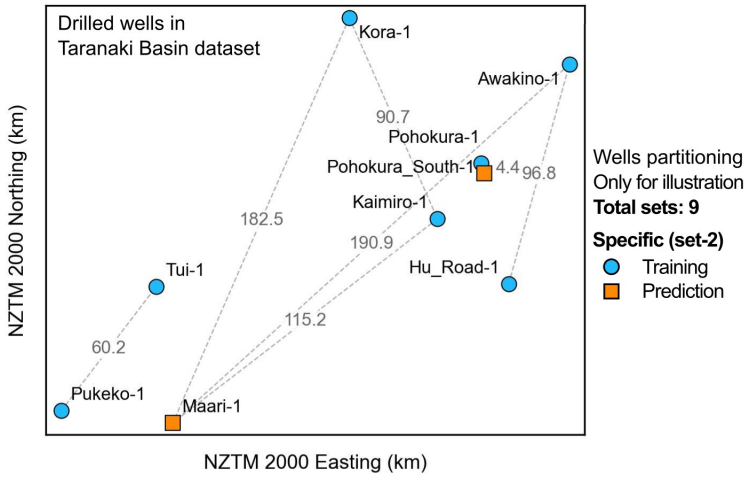


Figure 2. Taranaki Basin well distribution for the leave-two-well-out protocol, shown for set_2 as a conceptual illustration of the nine seeded evaluation splits. Blue circles: training wells; orange squares: the two test wells held out in set_2 (Pohokura_South-1, 4.4 km from Pohokura-1; Maari-1, 60–180 km from all training wells). Set_2 corresponds to Repeat 2 in the benchmark heatmap and represents a mixed-distance generalization scenario combining an intra-field proximal well with a basin-margin distal well.

2.3. Neural Architecture Search via Genetic Algorithm

Drawing on our prior work demonstrating genetic algorithm-based neural architecture search for embedded intelligence in resource-constrained environments [32], we employ

a genetic algorithm to discover an optimal encoder–regressor architecture under extreme parameter constraints. Unlike conventional NAS that searches over hundreds of layers and millions of parameters, our search targets sub-kiloparameter architectures suitable for microcontroller deployment.

Each candidate architecture is encoded as a chromosome, with each layer represented by seven genes that jointly determine its configuration. The unit gene $u \in (-1024, 1024)$ determines the number of neurons via $n_{\text{units}} = \lfloor u + 0.5 \rfloor$, where $n_{\text{units}} \leq 0$ signals layer termination. The batch normalization gene $b \in (0, 10)$ determines whether batch normalization is applied via has_BN $= \lfloor b/10 + 0.5 \rfloor > 0$. The dropout gene $d \in (0, 10)$ similarly controls whether dropout is applied via has_DO $= \lfloor d/10 + 0.5 \rfloor > 0$. The dropout rate gene $r \in (0, 1)$ sets the dropout probability. The activation gene $a \in (0, 2)$ selects the nonlinearity: $\lfloor a + 0.5 \rfloor = 0$ for exponential linear unit (ELU), 1 for rectified linear unit (ReLU), and 2 for hyperbolic tangent (tanh). The regularization gene $t \in (0, 2)$ selects weight regularization: 0 for none, 1 for L1, and 2 for L2. Finally, the output dimension gene $d_{\text{out}} \in (0.51, 128)$ sets the layer's output size.

The chromosome is bisected at its midpoint: the first half encodes the encoder $E_\theta : \mathbb{R}^8 \to \mathbb{R}^{d_{\text{latent}}}$, and the second half encodes the regressor $R_\phi : \mathbb{R}^{d_{\text{latent}}} \to \mathbb{R}^1$. For the regressor, the output dimension is overridden to 1 regardless of the gene value. The complete data flow is therefore

$$\mathbf{x}_d \xrightarrow{E_\theta} \mathbf{z} \in \mathbb{R}^{d_{\text{latent}}} \xrightarrow{R_\phi} \hat{y} \in \mathbb{R}^+ \tag{2}$$

The genetic algorithm fitness function jointly penalizes prediction error and model complexity. For each chromosome, three independent training runs are executed, and the robust mean of the middle 60% of WMAPE values is computed following the protocol described in Section 2.5; at this repetition count the trim rule drops no run. The fitness is then

$$\text{fitness} = 2 \times \left[\text{clip}(\bar{e}_{\text{test}},\ \varepsilon_e,\ E_{\max})\right]^{p_{\text{acc}}} + \frac{\max\left(0,\ \log_2(N_{\text{params}}) + b_{\text{offset}}\right)}{s_{\text{base}}} \tag{3}$$

where $\bar{e}_{\text{test}}$ is the robust-mean WMAPE, $\varepsilon_e = 0.0001$ is the error floor, $E_{\max} = 2$ is the error ceiling, N_{params} is the total trainable parameter count, and the hyperparameters are set to $p_{\text{acc}} = 2$, $b_{\text{offset}} = -7$, and $s_{\text{base}} = 20$. The first term doubles the squared prediction error, while the second term imposes a logarithmic penalty on model size that becomes active once the parameter count exceeds $2^7 = 128$; the penalty grows slowly beyond this threshold, ensuring that architectures with similar accuracy are discriminated primarily by their parameter efficiency.

Each candidate architecture is evaluated through the complete contrastive-regression training pipeline described in Section 2.4. The continuous ROP targets are first discretized into $K = 12$ uniform-width bins to enable contrastive learning; the encoder–regressor is then trained with the hybrid loss combining contrastive and regression objectives; and the robust-mean WMAPE is computed to score the chromosome. The NAS therefore co-evolves not only the network topology but also the latent space geometry that the contrastive loss imposes.

2.4. Contrastive Learning for Regression

The contrastive learning pipeline begins with the discretization of continuous ROP values into uniform-width bins spanning the fixed physical ROP range of 0–80 m/h. During NAS evolution on the USROP dataset, we set $K = 12$ intervals to provide sufficient granularity for the genetic algorithm to discriminate encoder geometries; for subsequent training and evaluation on the Taranaki Basin dataset, $K = 8$ intervals are used as an empirically determined trade-off between bin resolution and sample density. This discretization assigns

each sample to a ROP interval $l_i \in \{0, 1, \ldots, K-1\}$, where the bin boundaries divide the full ROP range into equal-width segments.

Conventional regression training via mean absolute error (MAE) optimizes only pointwise prediction accuracy, providing no explicit constraint on the structure of intermediate representations. Building on Khosla et al. [26] supervised contrastive learning and Hadsell et al. [33] dimensionality reduction by learning an invariant mapping, we introduce a hybrid loss function that imposes geometric structure on the latent embedding space. The core insight is that samples with similar ROP values should occupy proximal regions in the embedding space, while dissimilar samples should be separated, with the additional constraint that the embedding distance should respect the ordinal ordering of ROP magnitudes. Unlike conventional regularizers, which constrain weights, activations, or the input distribution, the contrastive term constrains the relations between embeddings directly; Table 1 positions it among the common regularization mechanisms.

Table 1. Positioning of the supervised contrastive constraint among common regularization mechanisms.

Regularizer	Constrains	Effect on Embedding Manifold	Evidence Here
L1/L2 weight decay	Individual weights	None direct	—
Dropout	Neuron activations	Indirect noise	—
Data augmentation	Input distribution	None	Not applicable to fixed sensor channels
Supervised contrastive (this work)	Relations between embeddings	Direct ordinal structuring	All nine ratios 0.588–0.639 vs. pure regression 0.6645; run std 0.0845 → 0.027–0.049

The complete loss combines three terms with adaptive weighting:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{contrast}} \cdot \mathcal{L}_{\text{contrast}} + \lambda_{\text{regress}} \cdot \mathcal{L}_{\text{MAE}} + \lambda_{\text{negative}} \cdot \mathcal{L}_{\text{neg}} \tag{4}$$

The contrastive loss $\mathcal{L}_{\text{contrast}}$ enforces embedding-space structure, the MAE loss $\mathcal{L}_{\text{MAE}}$ drives regression accuracy, and the negative penalty $\mathcal{L}_{\text{neg}}$ enforces physical non-negativity of ROP predictions. During NAS evolution, the negative penalty weight is set to $\lambda_{\text{negative}} = 0.1$; for final training and evaluation, it is raised to $\lambda_{\text{negative}} = 0.5$.

A critical design decision is the two-phase weighting strategy. During NAS evolution, we set $\lambda_{\text{contrast}} = 0.1$ and $\lambda_{\text{regress}} = 1.0$, yielding a 1:10 contrast-to-regression ratio. This conservative weighting keeps the GA focused on regression accuracy while imposing a weak geometric prior on the latent space; architectures that rise to the top of the population under this restrained regime are those whose encoders naturally produce structurally organized embeddings, rather than architectures that require an overpowering contrastive signal to maintain latent order. During final training and evaluation, the weights shift to $\lambda_{\text{contrast}} = 0.5$ and $\lambda_{\text{regress}} = 2.0$, a 1:4 ratio that strengthens the embedding constraint while preserving the regression signal, thereby refining the latent geometry discovered during evolution.

The contrastive loss itself comprises a triplet loss and an order loss:

$$\mathcal{L}_{\text{contrast}} = \lambda_{\text{dist}} \cdot \mathcal{L}_{\text{triplet}} + \lambda_{\text{order}} \cdot \mathcal{L}_{\text{order}} \tag{5}$$

with internal weights $\lambda_{\text{dist}} = 1.0$ and $\lambda_{\text{order}} = 3.0$. The pairwise Euclidean distance between embeddings is computed as $d_{ij} = \|\mathbf{z}_i - \mathbf{z}_j\|_2$, clipped to $[10^{-6}, 10^{6}]$ for numerical stability.

To handle class imbalance across ROP bins, adaptive interval weights are computed as $w_k = 1/(\log(f_k + 2) + \varepsilon)$, where f_k is the sample count in bin k. This weighting gives higher importance to underrepresented bins, preventing the dominant sand-interval samples from overwhelming the contrastive signal.

For triplet construction, positive samples for anchor i are defined as $\mathcal{P}_i = \{j : |l_i - l_j| \leq 1, j \neq i\}$ (samples from the same or adjacent bins), while negative samples are $\mathcal{N}_i = \{j : |l_i - l_j| > 1\}$ (samples from non-adjacent bins). Hardest negative mining selects the most challenging negative $n_i^* = \arg\max_{n \in \mathcal{N}_i}(d_i^+ - d_{in})$. To prevent densely populated bins from dominating the aggregate loss, each bin's contribution is further scaled by an inverse sample weighting $\omega_k = 1/(N_k + \varepsilon)$. The triplet loss for bin k with margin $m = 1.0$ is

$$\mathcal{L}_{\text{triplet}}^{(k)} = \frac{\omega_k}{N_k} \sum_{i \in \mathcal{I}_k} \max\left(0,\ m + d_i^+ - d_{i,n_i^*}\right) \tag{6}$$

The order loss enforces monotonic structure in the embedding space by penalizing pairwise distances between samples whose ROP bins are ordinally separated:

$$\mathcal{L}_{\text{order}} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{l_i} \cdot \max\left(0,\ d_{ij} \cdot \mathbb{I}(l_i < l_j)\right) \tag{7}$$

Here $\mathbb{I}(l_i < l_j)$ is the indicator function that activates only when sample i belongs to a lower ROP bin than sample j. Because the pairwise distance d_{ij} is symmetric and the double summation traverses all ordered pairs, this formulation applies the order constraint to every ordinally separated pair exactly once. With $\lambda_{\text{order}} = 3.0$, the ordering constraint receives three times the weight of the triplet distance constraint, reflecting its importance for maintaining a semantically meaningful latent geometry.

The regression loss is standard MAE: $\mathcal{L}_{\text{MAE}} = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|$. The negative penalty $\mathcal{L}_{\text{neg}} = \sum_{i=1}^{N} \max(0, -\hat{y}_i)$ provides a soft physical constraint that drives the model toward non-negative outputs without requiring post hoc clipping at inference time.

2.5. Training and Benchmark Protocol

All models are trained using the Adam optimizer with an initial learning rate $\alpha_0 = 10^{-4}$. Gradient norms are clipped via a dual mechanism: the optimizer is configured with clipnorm = 1.0, and an additional explicit clip is applied within the training step via tf.clip_by_norm($\mathbf{g}$, 1.0). This dual clipping provides redundancy against gradient explosion, which is particularly important given the multi-component loss landscape with competing contrastive and regression objectives. The learning rate schedule follows ReduceLROnPlateau, monitoring validation regression loss with a decay factor of 0.2, patience of 3 epochs, and minimum learning rate 10^{-9}. Early stopping triggers after 7 epochs without improvement exceeding $\delta = 10^{-6}$, restoring the best observed weights. Training operates with a batch size of 128, a minimum of 10 epochs, and a maximum of 50 epochs, with an absolute time limit of 15 min per training run.

The evaluation adopts a leave-two-well-out design over nine seeded evaluation splits (set_0–set_8). In each split, seven wells are used for training, two for validation, and two are entirely withheld as blind test wells. The validation wells serve only as the stopping criterion for early stopping and learning-rate scheduling; they do not contribute to gradient computation or weight updates. This protocol follows the recommendation of Tunkiel et al. [31] that well-based partitioning, as opposed to random point-wise splitting, is essential for realistic ROP benchmarking, because random splits violate the temporal and spatial dependence structure of drilling data and inflate accuracy metrics artificially. The simultaneous exclusion of two wells reduces training diversity and amplifies generalization stress, particularly when the held-out wells are spatially distant from the training population.

Two disclosures bound how the cross-split statistics in this paper should be read. First, the nine partitions are not independent observations: wells must recur across training, validation, and test roles from one split to the next, and any well not held out in a given partition reappears among that partition's training or validation wells, so dispersion across the split series describes

repeated re-partitions of one field inventory rather than independent field trials; cross-split comparison should be interpreted with that dependence in view. Second, the choice of nine seeded partitions is a cost-coverage decision rather than an exhaustive enumeration: the set of all two-well pairings over the available well inventory is combinatorially far larger than the number of partitions evaluated, and exhaustive enumeration would multiply the repeated-run training budget accordingly, so the partitions were drawn from the documented base seed to spread coverage across proximal and distal held-out pairings. Operationally, every partition aggregates the same 50 repeated training runs, each partition's failure tally is taken against the collapse-level criterion reported with Table 2, the full per-split error distribution itself is given in that table, and the paired pure-regression-versus-deployed-configuration runs completed with return code zero on every partition.

Table 2. Complete per-run error distribution of ROP MCU_model under the leave-two-well-out benchmark: all aggregated runs per split; sample standard deviation; failures counted against the collapse-level criterion (WMAPE ≥ 1.0).

Split	Mean	Median	Std	IQR	Min	Max	Failed	Runs
set_0	0.6748	0.6634	0.0860	0.0590	0.5490	1.0972	1	50
set_1	0.6386	0.6262	0.1437	0.0581	0.4301	1.4940	1	50
set_2	0.6517	0.6133	0.1622	0.0925	0.5023	1.4796	2	50
set_3	0.6183	0.6200	0.0665	0.0745	0.4817	0.8411	0	50
set_4	0.9816	0.9925	0.1811	0.2038	0.6003	1.6191	24	50
set_5	0.6371	0.6586	0.0835	0.0644	0.3428	0.7569	0	50
set_6	0.6926	0.6920	0.0605	0.0595	0.5675	0.8675	0	50
set_7	0.7140	0.6886	0.0958	0.1233	0.5619	1.0475	1	50
set_8	0.6797	0.6415	0.1649	0.0613	0.5354	1.6442	2	50
All	0.6987	0.6585	0.1611	0.1049	0.3428	1.6442	31	450

Following the definition of Tunkiel et al. [31], we adopt the weighted mean absolute percentage error (WMAPE) as the primary regression metric (Equation (8)):

$$\text{WMAPE} = \frac{\sum_{t=1}^{n} |A_t - F_t|}{\sum_{t=1}^{n} |A_t|} \tag{8}$$

where A_t and F_t denote the actual and predicted ROP at sample t. WMAPE is a globally weighted sum-ratio, not the per-sample mean of ratios, so low-ROP samples cannot dominate the score. Tunkiel et al. demonstrate that random point-wise train–test splits can inflate R^2 to near-unity values even in the absence of genuine predictive signal, because such splits leak spatial autocorrelation into the test set; they further observe that many reported $R^2 > 0.99$ claims remain unverifiable when the underlying data are undisclosed. WMAPE evaluated under well-based cross-validation provides an operationally interpretable alternative that measures trend agreement, rather than a correlation score that conflates interpolation with true generalization.

The baseline suite is selected to span the standard ROP estimation paradigms: classical ML estimators (RF, XGBoost, Gradient Boosting, AdaBoost, k-NN) as the conventional machine learning references; the Bingham model [34] as a physics-based reference; and a four-hidden-layer MLP ([128, 256, 96, 32] units with batch normalization, dropout 0.2, and L1 regularization; 64,033 parameters, approximately 229× the 280-parameter ROP MCU_model) as the conventional neural baseline. To the authors' knowledge, no published TinyML architecture exists for ROP regression; MCUNet [17] and µNAS [29] target classification tasks, so the MLP serves as the deep-learning reference point.

The statistical evaluation protocol is governed by the distinction between deterministic and stochastic model classes, and it further differs by analytical purpose, namely statistical

benchmarking versus case-study visualization. Classical ML estimators (Random Forest, XGBoost, Gradient Boosting, AdaBoost, and KNeighbors) and the Bingham physical model [34] are evaluated with library default hyperparameters under the fixed data-split seed, with no hyperparameter-search stage; the ensemble and boosting estimators retain stochastic training elements, so each configuration contributes a single fitted result per evaluation split, which is reported directly, while the closed-form Bingham reference remains deterministic. Neural-network models, including both the MLP baseline and ROP MCU_model, exhibit training stochasticity arising from random weight initialization and mini-batch ordering, which can occasionally produce training-failure or weight-collapse runs. To obtain a robust estimate, each neural-network configuration undergoes repeated independent training runs, and the robust mean of the middle 60% (discarding the top and bottom 20%) is reported as the primary metric. This protocol filters out training collapses inherent to neural-network optimization. For final benchmarking (Figure 3), 50 runs per evaluation point are used; for NAS fitness evaluation, three training runs per chromosome are used. For the case-study prediction-curve visualization (Figure 4), the ROP MCU_model curve plots the statistically representative individual of the repeated training runs, designated before and independently of any test-well result under the project's original reporting convention, whereas the competing-model MLP curve plots the individual recorded for that suite under the production benchmark's archived reporting convention. Well splits derive from base seed 88; the architecture search used seed 2025; network training is intentionally unseeded, and all reported metrics aggregate 50 independent training runs per configuration (mean and best-model statistics).

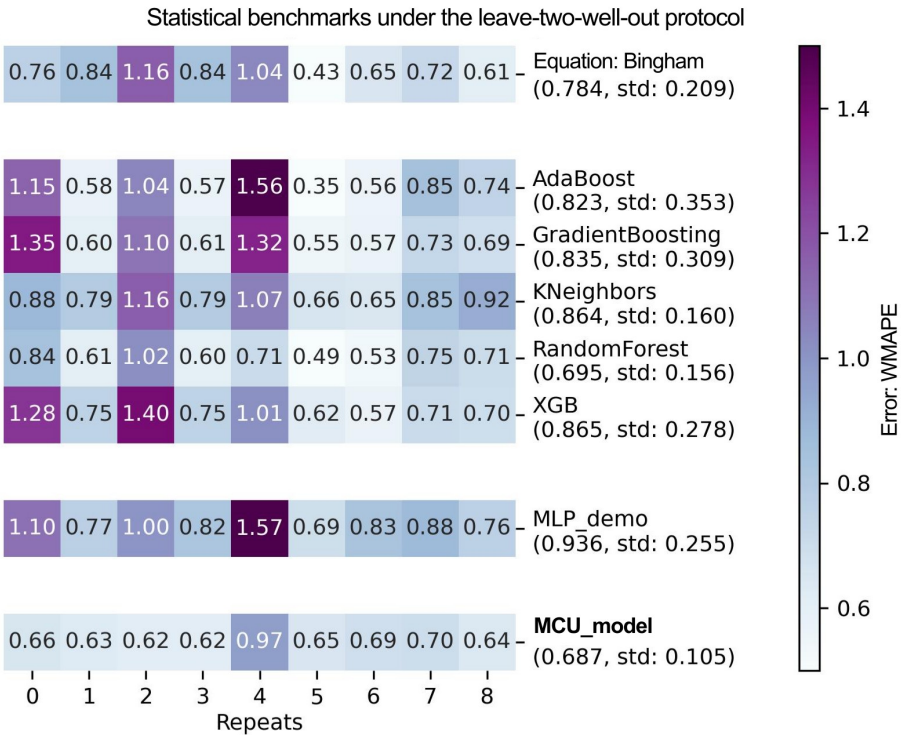


Figure 3. WMAPE heatmap under the leave-two-well-out protocol (nine repeats, 0–8). Neural-network cells report the robust mean of the middle 60% across 50 runs; deterministic estimators report a single evaluation. The right-hand annotation gives the cross-repeat mean and standard deviation. ROP MCU_model achieves the lowest mean WMAPE (0.687) and the smallest standard deviation (0.105), indicating the lowest prediction error and stable generalization across diverse well combinations.

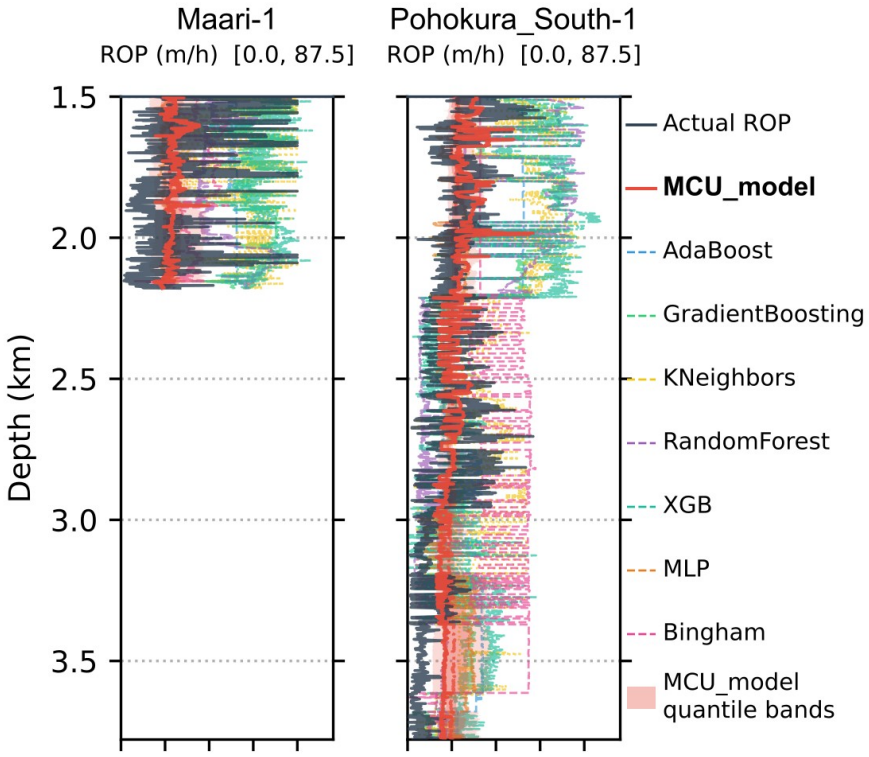


Figure 4. Full-well ROP prediction curves for the two held-out wells in set_2. (**Left**) Maari-1, located >60 km from all training wells. (**Right**) Pohokura_South-1, located 4.4 km from the nearest training well. Black: actual ROP. Red: ROP MCU_model. Other colors: competing models (AdaBoost, Gradient Boosting, KNeighbors, Random Forest, XGB, MLP, Bingham). The ROP scale starts at zero. The ROP MCU_model curve shows the statistically representative individual of the repeated training runs, recorded under the project's original reporting convention rather than chosen afterwards from test-well performance. The shaded region is the median with quantile bands aggregated across the 50 archived runs. For the displayed representative individual the per-well WMAPE is 0.5576 on Maari-1 and 0.4791 on Pohokura_South-1, and its pooled two-well WMAPE is 0.5023. The competing-model MLP curve shows the individual recorded under the production benchmark suite's archived reporting convention. Portions of the competing-model curves below zero are clipped by the new axis origin.

2.6. MCU Deployment and Quantization Pipeline

The proof-of-concept deployment target is the ESP32-S3 microcontroller (Espressif Systems, Shanghai, China; Xtensa LX7 dual-core at 240 MHz, 520 KB SRAM, 8 MB PSRAM). Surviving a high-temperature high-pressure wellbore currently restricts the usable computing hardware to microcontroller-class devices with extremely limited computational resources, so this study adopts a consumer-grade device that others can readily obtain and reproduce to demonstrate that the algorithm concept is feasible under constraints of that class. The inference framework is TensorFlow Lite Micro (TFLM) [19] for accelerated kernel execution. The runtime allocates a statically reserved Tensor Arena in SRAM for model state, activation buffers, and temporary storage during inference. The firmware stack runs on FreeRTOS, with model inference and optimization engine executing as cooperative tasks.

Following the full-integer quantization methodology of Jacob et al. [18], the trained FP32 model is converted to INT8 representation using the TensorFlow Lite converter. Input quantization converts floating-point features to 8-bit integers via $q_x = \text{round}(x/s_{\text{in}} + z_{\text{in}})$, where s_{in} is the input scale factor and z_{in} is the zero-point offset. After inference, output dequantization recovers the predicted ROP value via $\hat{y} = (q_y - z_{\text{out}}) \cdot s_{\text{out}}$. The TFLM interpreter receives and returns INT8 tensors; quantization and dequantization are performed

manually in the inference wrapper before and after invocation, enabling deployment on integer-only microcontrollers.

The deployment validation workflow spans three platforms to ensure cross-architecture portability. The INT8 model is first verified on an x86_64 workstation for numerical correctness against the FP32 Keras reference. It is then profiled on an ARM Cortex-A72 (Raspberry Pi 4) to validate kernel behavior on a reduced-instruction-set architecture. Finally, the model is flashed to the ESP32-S3 for on-device inference benchmarking and optimization testing. This tiered validation strategy ensures that accuracy, latency, and memory behavior are characterized across the full spectrum from development host to target microcontroller.

To complement the embedded firmware, a browser-based offline replay dashboard was developed as a supplementary validation and demonstration tool. The application ingests JSON logs exported from the MCU, containing sensor inputs, predicted ROP values, optimizer trajectories, and recommended parameters, and renders interactive time-series visualizations and optimization-convergence curves. The tool serves two purposes: it enables qualitative inspection of MCU execution traces without physical hardware access, and it lets domain engineers interactively inspect safety-constraint enforcement and optimizer convergence. The dashboard is provided as a supplementary file and runs entirely client-side without external dependencies.

2.7. On-Device Optimization Engine

The MCU firmware integrates a real-time optimization engine that treats the deployed ROP MCU_model as a surrogate objective function. Given a fixed formation state vector comprising Measured Depth, TVD, bit size, pipe pressure, and torque, the engine searches over the three controllable drilling parameters (WOB, RPM, and flow rate) to maximize predicted ROP. The optimization problem is

$$\max_{\mathbf{p}} \hat{y}_{\mathrm{ROP}} = f_{\mathrm{INT8}}(\mathbf{x}_s, \mathbf{p}), \quad \mathbf{p} = (\mathrm{WOB}, \mathrm{RPM}, \mathrm{Flow}) \tag{9}$$

where f_{INT8} denotes the INT8-quantized ROP MCU_model and $\mathbf{p}$ is the vector of decision variables. The budget is fixed at 50 function evaluations per optimization point.

Four optimization algorithms are implemented on the MCU, switchable at runtime via JSON command. Nelder–Mead operates via simplex deformation with a working memory of 16 floats (simplex vertices plus objective values). Random Search samples uniformly within the parameter bounds with zero persistent state memory. Differential Evolution maintains a population of candidate solutions and evolves them through mutation and crossover, requiring 40 floats of working memory (population plus fitness scores). DE + Nelder–Mead is a two-stage hybrid that allocates 30 evaluations to DE for global exploration followed by 20 evaluations to NM for local refinement; its peak working memory is 40 floats, because the NM stage reuses the stack frame after the DE phase completes. The memory footprints and algorithmic characteristics of these four variants are summarized in Table 3 and evaluated empirically in Section 3.

Table 3. MCU optimization algorithms comparison (budget = 50 evaluations).

Algorithm	Memory (Floats)	Mean Time (µs)	Std. Dev. (µs)	Algorithm Type
Nelder–Mead	16	12,860	103	Local (simplex)
Random Search	0	14,853	74	Uniform sampling
DE + NM	40	18,489	163	Two-stage hybrid
Differential Evolution	40	21,407	256	Global evolution

All optimization recommendations are confined to bounded intervals and pass a preliminary feasibility pre-screen before being issued; this mechanism is a data-supported

search envelope and does not constitute an engineering safety certification. Percentile and change-rate bounds are enforced on the microcontroller side, whereas the ROP floor and the upstream command issuer's constraints are enforced in the deployment pipeline rather than in firmware. Layer 1 enforces parameter bounds anchored to both the current operating point and the global training distribution. The global percentiles (p10 and p90) are derived from the Taranaki Basin training set distribution for each controllable parameter (WOB, RPM, Flow), providing an empirical, data-driven safety envelope. Specifically, for each decision variable, $\text{param}_{\text{min}} = \max(\text{current} \times 0.2, \text{global}_{p10})$ and $\text{param}_{\text{max}} = \min(\text{current} \times 1.8, \text{global}_{p90})$. The current-value multiplier ($\pm 80\%$) sets the width of a conservative, data-supported search envelope as an engineering heuristic rather than a dynamics-validated constraint, while the global percentile constraint ensures the optimizer does not extrapolate beyond the training distribution; residual operational safety remains with the Layer 2 ROP-floor pre-screen and with the Layer 3 upstream command issuer and driller, who validate each recommendation before it is executed. Layer 2 enforces an ROP floor: if the optimized ROP falls below the baseline predicted ROP, the recommendation is clamped to the baseline value with zero improvement, ensuring that the system never recommends a parameter set predicted to degrade performance. Layer 3 delegates constraint enforcement to the upstream command issuer rather than embedding hard limits in the firmware; the optimization engine receives bounds as part of each optimization command and trusts their validity, keeping the on-device binary minimal and allowing safety logic to evolve without hardware re-flashing. The optimization engine autonomously generates operational decisions, specifically optimal WOB, RPM, and flow-rate setpoints, for the specific objective of maximizing predicted ROP under the given safety constraints. These decisions are intended to advise the driller or a supervisory control system, while also demonstrating that a complete prediction–optimization–decision pipeline can be closed on a microcontroller-class device.

3. Results and Analysis

This section presents the quantitative evaluation of the proposed contrastive-regression neural architecture for ROP prediction, its quantization-aware deployment profile, and the on-device optimization performance on a resource-constrained microcontroller. Prediction metrics are evaluated under the leave-two-well-out protocol across nine seeded evaluation splits (set_0–set_8), with the cross-split mean WMAPE serving as the primary generalization metric. The case study focuses on set_2, where Pohokura_South-1 and Maari-1 are simultaneously withheld. On this split, ROP predictions from competing models are compared along the full depth trajectories of both held-out wells. The Pohokura_South-1 interval 2300–2500 m (200 samples, actual mean ROP 18.42 m/h) is selected as the on-device demonstration scenario: the MCU executes real-time inference and searches controllable drilling parameters at 5-m increments to maximize predicted ROP.

3.1. Optimized Network Architecture

The reported architecture was selected from the population snapshot at generation 12 of the genetic-algorithm search. Based on the fitness values across the evolutionary trajectory, the representative individual was identified as Generation 10, Individual 177 (Gen-10, ID-177), yielding a 280-parameter encoder–regressor with latent dimension of 2.

Figure 5 presents the optimized architecture and its learned embedding structure in three complementary views. The views are read in sequence: panel (a) the encoder–regressor topology that consumes the eight normalized drilling parameters, panel (b) the two-dimensional embedding, and panel (c) its one-dimensional projection. Figure 5a depicts the encoder–regressor topology discovered by the genetic algorithm. The input layer

receives eight normalized drilling parameters; the encoder compresses these through a single dense layer (7 units, ReLU, batch normalization, dropout 0.57) into a two-dimensional latent representation. The regressor then expands this embedding through a 43-unit ELU hidden layer to the scalar ROP output. The grayscale connection lines encode the learned weight magnitudes, with darker tones indicating stronger coupling between neurons, a pattern that directly reflects the parameter relationships explored in the case-study prediction curves of Section 3.

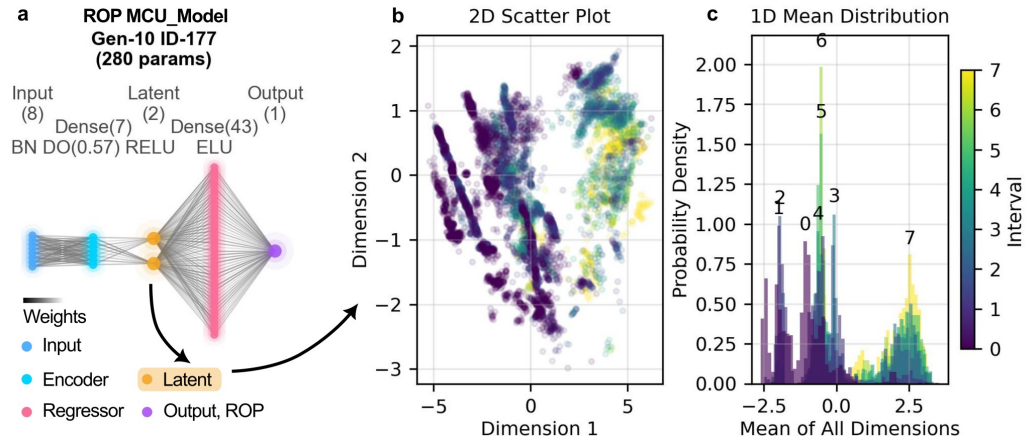


Figure 5. (**a**) Network architecture diagram of the proposed ROP MCU_model (Gen-10, ID-177) showing the encoder–regressor topology. Grayscale connection lines encode the learned parameter weight magnitudes, with darker tones indicating stronger coupling between neurons. (**b**) Two-dimensional embedding space after training, with each point colored by its ROP uniform interval (0–7). Low-ROP intervals (purple, 0–2) occupy the lower-left region, while high-ROP intervals (yellow–green, 5–7) cluster in the upper-right, with intermediate intervals (3–4, teal) bridging the two extremes. (**c**) One-dimensional mean projection of the 2-D embeddings, showing probability density distributions for each ROP interval. The clear separation between low-interval peaks (negative mean values) and high-interval peaks (positive mean values) confirms that the contrastive objective has structured the latent space to preserve the ordinal ROP relationship. In panel (**a**) the pathway runs from the eight normalized drilling inputs, across the encoder bottleneck, to the scalar ROP readout; panels (**b**,**c**) examine the embedding learned in that architecture, first in two dimensions and then in projection.

Figure 5b visualizes the two-dimensional embedding space after training, with each point representing a drilling sample and its color indicating the ROP uniform interval to which it belongs. The contrastive learning objective has structured the latent manifold such that low-ROP intervals (purple, intervals 0–2) occupy the lower-left region, while high-ROP intervals (yellow–green, intervals 5–7) cluster in the upper-right. The intermediate intervals (3–4, teal) bridge the two extremes along a roughly diagonal trajectory, confirming that embedding distance correlates with ROP magnitude.

Figure 5c collapses the 2-D embeddings to a one-dimensional mean projection, revealing the probability density distribution for each ROP interval. The low-interval distributions peak at negative mean values (approximately −2.5 to −1.0), whereas the high-interval distributions peak at positive values (approximately +1.5 to +3.0). The near-complete absence of overlap between the low-interval and high-interval density peaks demonstrates that the contrastive embedding has achieved linear separability in the projected space, a geometric property that enables the downstream regressor to discriminate ROP states reliably despite the extreme parameter scarcity.

The architecture specification of the selected individual is detailed in Table 4.

Table 4. Model architecture specification for the ROP MCU_model (Gen-10, ID-177). Parameter counts include dense weights, biases, and batch-normalization statistics.

Component	Layer	Units	Activation	Batch Norm	Dropout	Regul.	Parameters
Encoder E_θ	Dense	7	ReLU	Yes	0.57	L1	91
	Output	2	—	No	No	None	16
Regressor R_ϕ	Dense	43	ELU	No	No	L1	129
	Output	1	—	No	No	None	44
Total							**280**

The architecture in Table 4 reflects the extreme parameter efficiency discovered by the genetic algorithm. The encoder compresses the 8-dimensional drilling feature vector through a single hidden layer (7 units with batch normalization and dropout at rate 0.57) into a compact two-dimensional latent space. The regressor then maps this 2-D embedding through a single hidden layer (43 units with ELU activation) to the scalar ROP prediction. The genetic algorithm allocated the majority of parameters (129) to the regressor's hidden layer, suggesting that expanding the latent representation is more critical than deepening the encoder under the 280-parameter constraint. Batch normalization and dropout appear only in the encoder hidden layer, while L1 regularization is applied to both trainable dense layers. The resulting 280-parameter count represents a reduction of more than three orders of magnitude compared to conventional 4-layer MLP baselines with comparable prediction error. This count was selected autonomously by the fitness value defined in Section 2.3: the extreme parameter budget of fewer than 1000 trainable weights is a design target that steers the search rather than a hard constraint inside the fitness function (the size term is logarithmic and activates only past 128 parameters), and what the search returned was the two-dimensional latent bottleneck with its 43-unit regressor head. The representational capacity of this pair is compact but sufficient for the task: the bottleneck retains the dominant, ROP-ordered factors of the instantaneous drilling state, the head expands that two-dimensional embedding before the scalar readout, and the resulting model remains on par with conventional ensemble methods that use orders of magnitude more parameters.

3.2. Comparative Statistical Evaluation

Prediction metrics are evaluated under the leave-two-well-out protocol over nine seeded evaluation splits (Repeats 0–8). Figure 3 reports the WMAPE heatmap for all competing models across the nine repeats.

Table 2 reports the complete per-run error distribution of the proposed model under the benchmark protocol: the conventional mean, median, sample standard deviation, interquartile range, minimum, and maximum over all aggregated runs of every split, together with the count of runs meeting the collapse-level failure criterion (WMAPE ≥ 1.0). The headline benchmark statistic is unchanged wherever it is reported: its per-split values already appear in the physical-bounds column of the appendix protocol comparison, so the full distribution and that headline value are now readable side by side instead of the distribution remaining invisible. Where a single archived run is named in this paper, namely the MCU curve of Figure 4, that run is recorded in the archived results log under the project's original reporting convention, which designates a statistically representative individual rather than a post-test pick, decoupling run identification from the test wells.

Figure 3 reveals two broad patterns. First, conventional ML models exhibit high cross-repeat variance: AdaBoost (mean 0.823, std 0.353), Gradient Boosting (0.835, 0.309), and XGBoost (0.865, 0.278) all show large WMAPE swings depending on which wells are withheld, reflecting their sensitivity to training-set composition. The MLP baseline (0.936,

0.255) suffers from the same instability despite its neural-network formulation, indicating that architectural depth alone does not confer robustness when parameters are scarce and training data are heterogeneous. The baselines' weakness is not merely a worse scalar score: on blind wells their predictions deviate from the overall ROP trend, as the case-study curves of Figure 4 illustrate. Second, ROP MCU_model achieves the lowest mean WMAPE (0.687) and, critically, the smallest standard deviation (0.105, computed as the population standard deviation over the nine split means) among all tested methods. Its heatmap row is uniformly pale, with WMAPE values ranging narrowly from 0.62 to 0.97 across all nine repeats. The widest deviation occurs on set_4, which pairs the two lowest-ROP wells of the dataset and thereby jointly minimizes the WMAPE denominator; the same split is the worst for all benchmarked model families (e.g., AdaBoost 1.564), indicating a property of the well pair rather than of the proposed model. This stability is the direct consequence of the contrastive embedding constraint: by forcing drilling states of similar ROP-level structure to cluster on a structured manifold, the model generalizes reliably even when training diversity is sharply reduced by the simultaneous exclusion of two wells.

Among the nine repeats, Repeat 2 (set_2) is selected for the detailed case study that follows. This choice is not driven by statistical extremity, as ROP MCU_model's WMAPE on Repeat 2 (0.62) lies well within its narrow observed range, but by the typicality of its well-pair geometry. Set_2 simultaneously withholds Pohokura_South-1 (4.4 km from the nearest training well, representing intra-field proximal generalization) and Maari-1 (>60 km from all training wells, representing basin-margin distal generalization). This mixed-distance combination makes set_2 a representative stress test for spatial heterogeneity: a model that performs reliably on set_2 must handle both interpolation within a field and extrapolation across a basin. The following sections examine this capability in detail, first through full-well prediction curves for both held-out wells, and then through interval-specific on-device optimization on Pohokura_South-1.

Figure 4 presents the predicted versus actual ROP across the full depth trajectories of both held-out wells. On Maari-1 (left), the actual ROP (black) exhibits substantial variation with depth. The ROP MCU_model prediction (red) tracks the dominant trend of the actual ROP while suppressing high-frequency fluctuations, yielding a curve with median-filter-like statistical characteristics. The competing models produce highly fragmented trajectories: AdaBoost, Gradient Boosting, XGB, and KNeighbors display pronounced high-amplitude oscillations that deviate substantially from the actual ROP envelope, while the MLP and Bingham predictions drift away from the true trend in the lower depth intervals.

On Pohokura_South-1 (right), where the well is proximal to the training population, the actual ROP shows larger dynamic range. ROP MCU_model again captures the principal trend, maintaining alignment with the actual ROP through both gradual transitions and sharp excursions. The conventional ML models continue to exhibit noisy outputs: Random Forest and XGB produce scattered predictions with large variance, Bingham generates systematic offsets in several depth bands, and the MLP shows intermittent divergence from the baseline. The contrastive embedding constraint appears to act as an implicit smoothing prior, enabling ROP MCU_model to extract the coherent underlying signal while rejecting spurious fluctuations that destabilize the competing estimators.

3.3. Cross-Platform Inference Profiling

Post-training uniform quantization to 8-bit signed integers (INT8) was applied to both the proposed ROP MCU_model and the MLP baseline. On-device validation of the INT8-quantized variant confirms negligible quantization-induced degradation relative to the full-precision predictions (see the set-2 case study in Section 3). The quantized INT8

model occupies 5704 bytes (5.57 KB) and fits the 32 KB tensor arena with ample margin; the MLP baseline at 77.45 KB does not (Table 5).

On the desktop-class Intel processor, ROP MCU_model INT8 completes inference in 1.57 µs. On the Raspberry Pi 4 (ARM Cortex-A72), latency rises to 29.32 µs; the MLP baseline requires 33.37 µs on the same platform. On the ESP32-S3, ROP MCU_model INT8 completes in 156.38 µs, while the MLP cannot be benchmarked because its 77.45 KB footprint exceeds the available tensor-arena allocation. All ESP32-S3 figures are real on-device measurements at 240 MHz with a statically allocated 32 KB tensor arena over 200 timed repetitions; each timing wraps a concise I/O-quantized inference loop (input quantization, interpreter invocation, and output dequantization).

Table 5. Model size comparison and MCU deployability.

Model	Size	MCU Deployable
ROP (INT8)	5.57 KB	Yes
MLP (INT8)	77.45 KB	Partial
AdaBoost	30.70 KB	Yes
Gradient Boosting	193.24 KB	No
XGBoost	422.83 KB	No
Random Forest	103.40 MB	No

Table 5 shows a five-order-of-magnitude span in model size, from 5.57 KB to 103.40 MB. Were the models statistically equivalent, this deployability gap would be decisive; however, Figure 3 separates the classical estimators into two groups on accuracy grounds. AdaBoost, Gradient Boosting, and XGBoost combine high mean error (0.823–0.865) with high cross-split instability (standard deviations of 0.278–0.353), which rules them out regardless of memory footprint. Random Forest remains close to ROP MCU_model in mean error (0.695 versus 0.687), but its 103.40 MB footprint is five orders of magnitude larger and equally excludes deployment on MCU-class hardware. The MLP baseline is partially deployable yet exceeds the ESP32-S3 tensor-arena allocation at 77.45 KB. Only ROP MCU_model INT8 satisfies both the statistical constraint, namely the lowest mean WMAPE (0.687) and the smallest standard deviation (0.105), and the design constraint of sub-100 KB footprint.

Figure 6 places throughput and memory footprint on a single comparative plane. The left axis (log scale) reports inference throughput in inferences per second for three platforms, x86_64 (dark blue), aarch64 (light blue), and ESP32-S3 (green), while the right axis (log scale) overlays memory footprint in kilobytes (orange markers). The shaded grey band highlights the region where memory exceeds 100 KB, corresponding to the conventional "not deployable on MCU" threshold.

On x86_64, ROP MCU_model INT8 achieves 636,000 inferences per second, 19% faster than the MLP INT8 baseline (535,000 inf/s). The margin widens on aarch64 (Raspberry Pi 4): 34,000 versus 30,000 inf/s. On the ESP32-S3, ROP MCU_model INT8 sustains 6394 inferences per second, exceeding the one-millisecond real-time target by a factor of 6.4. No other model achieves non-zero throughput on this platform; the MLP exceeds the tensor-arena limit, and the tree-based models occupy 193 KB to 103 MB, well inside the grey exclusion zone.

A dedicated benchmark was executed on the ESP32-S3 to characterize the real-time performance envelope under production conditions. Two hundred consecutive inferences were performed using the TFLM interpreter with a statically allocated tensor arena of 32 KB at the 240 MHz system clock. Each timing wraps a concise I/O-quantized inference loop (input quantization, interpreter invocation, and output dequantization) taken with the ESP32 hardware timer for cycle-accurate timing; these are real on-device measurements, not host-side estimates. The average inference latency is 156.38 µs, with individual mea-

surements spanning 148–167 µs (coefficient of variation approximately 3%). The 19 µs peak-to-peak jitter is attributable to interrupt handling and cache effects inherent to the FreeRTOS scheduler. Even in the worst-case observed latency of 167 µs, the inference component consumes less than 0.02% of a 100 ms control cycle, leaving the overwhelming majority of the computational budget available for the optimization routine and I/O operations.

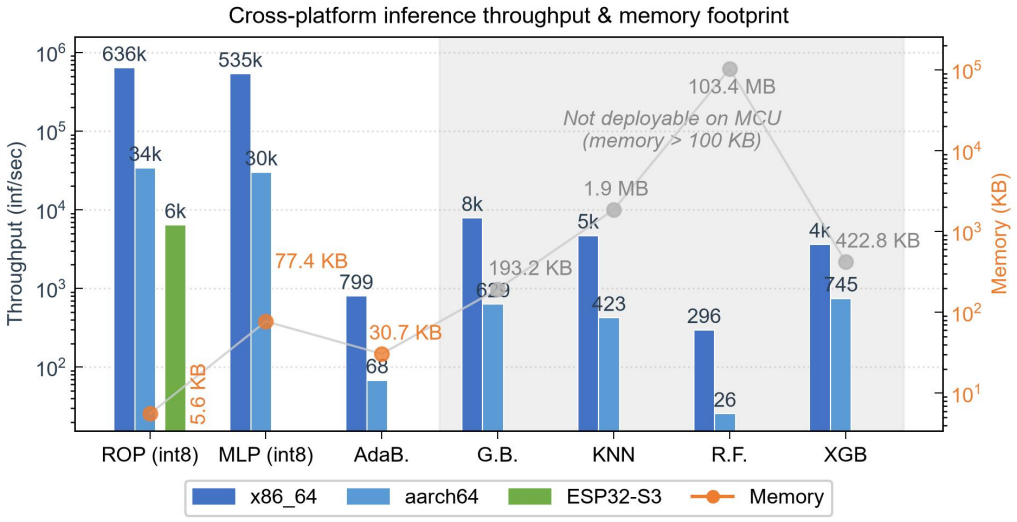


Figure 6. Cross-platform inference throughput and memory footprint. Bars show throughput (left axis, log scale) on x86_64, aarch64, and ESP32-S3. Orange markers show memory footprint (right axis, log scale). The grey shaded band denotes the memory->100 KB region conventionally regarded as non-deployable on MCU.

3.4. Optimization Evaluation for Microcontroller

The computational cost of the optimization loop on the MCU was characterized by timing each algorithm variant across repeated executions. Table 3 reports the mean execution time and standard deviation for all four algorithms under a function-evaluation budget of 50.

On the ESP32-S3, Nelder–Mead is the fastest at 12.9 ms pure compute time, while the DE + NM hybrid achieves a balanced 18.5 ms. All four algorithms complete within 52.5 ms when the cooperative-multitasking yield is accounted for, sitting comfortably below the 100 ms control-loop deadline.

Figure 7 presents the x86_64 pre-screening results used to guide the final algorithm selection. Figure 7a shows convergence trajectories for six candidate algorithms over 50 function evaluations. Nelder–Mead (red) converges to the lowest cumulative best ROP (≈0.28) and shows the widest variance band, indicating high sensitivity to initial simplex geometry. Random Search (blue) reaches ≈0.34 within the first five evaluations and then levels off. Differential Evolution (green) and DE + NM (purple) both reach ≈0.34–0.35; DE + NM shows slightly smoother convergence with a narrower variance band. CMA-ES (grey dashed) reaches ≈0.35, comparable to the MCU-compatible group. Bayesian optimization (yellow dotted) achieves the highest final value (≈0.38) but requires more than 10 s per run and is excluded from MCU deployment on memory and runtime grounds.

Figure 7b quantifies the runtime gap. Random Search is fastest at ≈0.7 ms, followed by Nelder–Mead (≈1.5 ms), DE (≈2 ms), and DE + NM (≈2.5 ms). CMA-ES requires ≈10 ms, while Bayesian optimization exceeds 10 s, four orders of magnitude slower than the MCU-compatible quartet.

Figure 8 shows the surrogate-model-predicted ROP improvement delivered by each optimizer across the Pohokura_South-1 2300–2500 m interval. The violin plots show the

distribution of percentage ROP improvement for 41 optimization points under a budget of 50 evaluations.

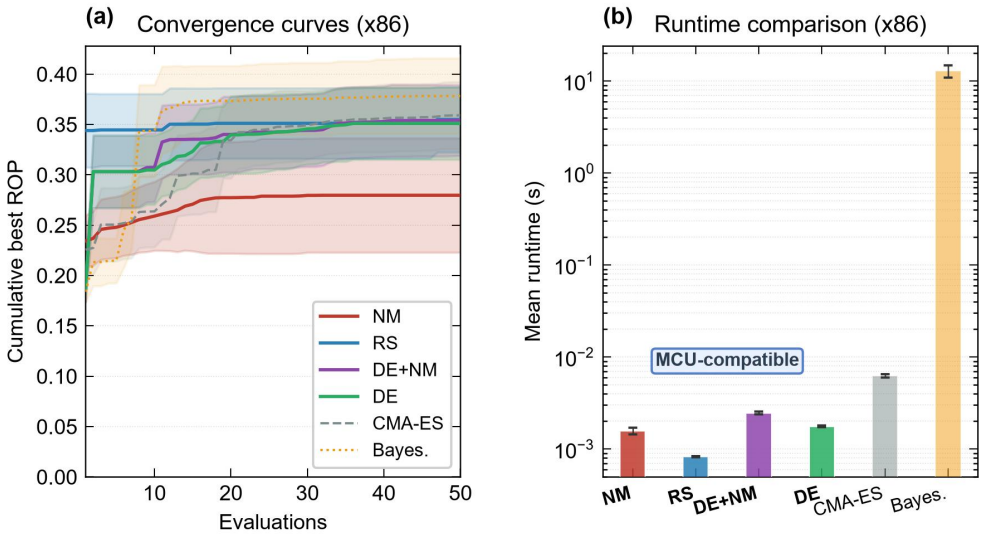


Figure 7. (**a**) Convergence curves on x86_64 for six candidate optimization algorithms (budget = 50 evaluations). Cumulative best ROP is plotted against function-evaluation count. Shaded bands indicate inter-run variance. (**b**) Mean runtime comparison on x86_64. The "MCU-compatible" bracket encompasses NM, RS, DE, and DE + NM. Bayesian optimization exceeds 10 s and is omitted from the bracket.

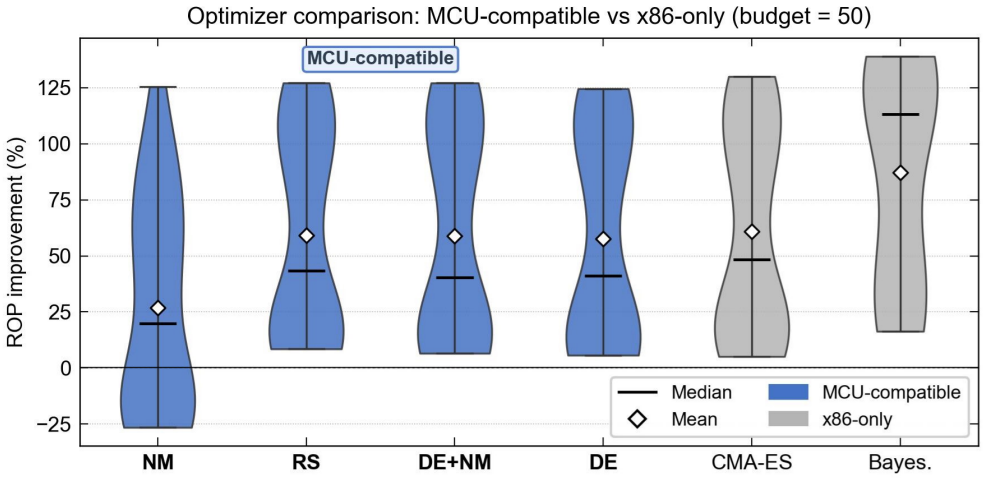


Figure 8. Optimizer comparison: surrogate-model-predicted ROP improvement distributions across 41 evaluation points (budget = 50). Blue violins denote MCU-compatible algorithms; grey violins denote x86_64-only algorithms. Horizontal lines mark medians; diamonds mark means. Negative improvement values occur for Nelder–Mead: its simplex is sensitive to the initial geometry, and at some evaluation points the optimized point-prediction falls below the current operating point. No ROP floor is applied in these study-side comparisons; the floor of the deployment pipeline is a host-side software policy in the replay and command layers, not a firmware-side clamp.

The MCU-compatible algorithms (blue) cluster tightly: RS, DE, and DE + NM all achieve mean surrogate-model-predicted improvements of 57–58%, with medians near 40%. Nelder–Mead is the outlier within the MCU-compatible group: its mean improvement is only 25% and its distribution spans from −25% to +125%, reflecting the algorithm's sensitivity to initial simplex geometry. The x86_64-only algorithms (grey), CMA-ES and Bayesian optimization, achieve higher means (61% and 87%, respectively), but the gain is insufficient to justify sacrificing MCU deployability. DE + NM is therefore selected as the

production optimizer: it matches the surrogate-model-predicted ROP improvement of pure DE and RS while offering more stable convergence behaviour, and its 18.5 ms pure compute time leaves ample margin within the 100 ms control cycle.

3.5. Case Study: Autonomous ROP Decision-Making On-Device

This on-device demonstration adopts a host-to-MCU offline replay protocol: the x86 host streams pre-recorded drilling-state vectors to the ESP32-S3 MCU via USB-Serial, emulating the downhole sensor acquisition layer, while all inference and optimization computations execute exclusively on the MCU. No physical feedback loop is exercised in this demonstration; all MCU input is drawn solely from the previously recorded run. This configuration provides unambiguous proof that the complete sense–predict–optimize–decide pipeline can be closed entirely within the microcontroller's resource envelope. The x86 host acts only as a passive data injection terminal; it does not participate in prediction or parameter optimization.

The Pohokura_South-1 interval 2300–2500 m serves as the dedicated demonstration scenario. At each of 41 points spaced at 5 m increments, the DE + NM optimizer searches WOB, RPM, and flow rate to maximize the INT8 model's predicted ROP, subject to the MCU-side bounded intervals and the deployment-pipeline feasibility pre-screen.

Figure 9 presents the complete on-device execution trace. Figure 9a compares the INT8-quantized prediction (orange) against the actual ROP (black) as recorded by the field logging system. The INT8 curve tracks the actual ROP with negligible visible deviation, confirming that full-integer quantization preserves the model's predictive fidelity on the MCU. Figure 9b overlays the full-precision Float32 reference (red) on the same axes; the red and orange curves are visually coincident, providing direct evidence that quantization loss is negligible within this interval. The green dots mark the optimized ROP at each evaluation point, consistently lying above both the actual and predicted curves. Figure 9c displays the normalized optimized drilling parameters (WOB in blue, RPM in orange, Flow in green) as stepwise trajectories, reflecting the 5 m discrete optimization grid.

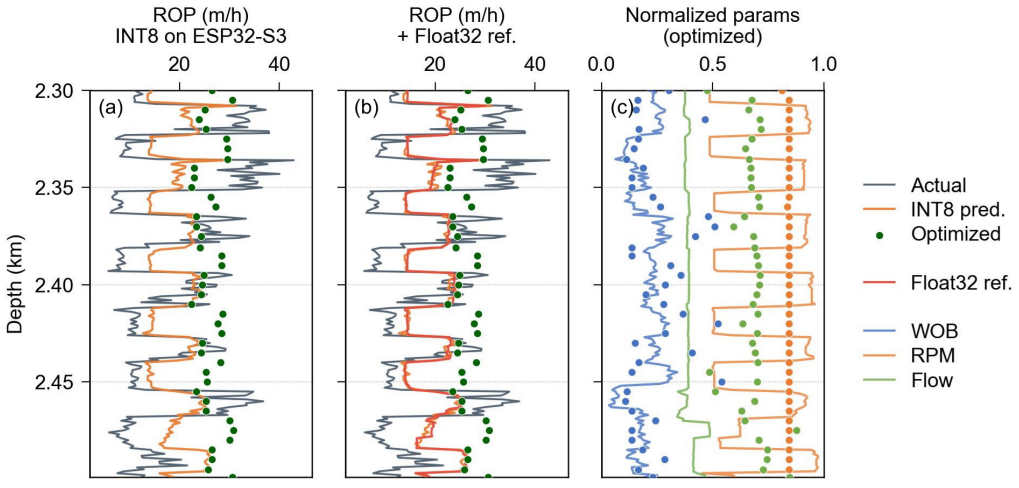


Figure 9. On-device autonomous optimization and decision-generation trace for the Pohokura_South-1 2300–2500 m interval. (**a**) Actual ROP (black) versus INT8 prediction on the ESP32-S3 (orange); green dots denote optimized ROP. (**b**) Same as (**a**) with the Float32 reference (red) overlaid, demonstrating negligible quantization deviation. (**c**) Normalized optimized drilling parameters: WOB (blue), RPM (orange), and Flow (green).

Figure 10 summarizes the optimization statistics and timing budget. Figure 10a shows the surrogate-model-predicted ROP improvement distribution for DE + NM executed

entirely on the ESP32-S3. The distribution of surrogate-model-predicted gains is right-skewed: mean +46.2%, median +18.5%, standard deviation 43.0%, maximum +124.6%. The disparity between mean and median indicates that several points achieve exceptionally large gains, while the majority of improvements are moderate.

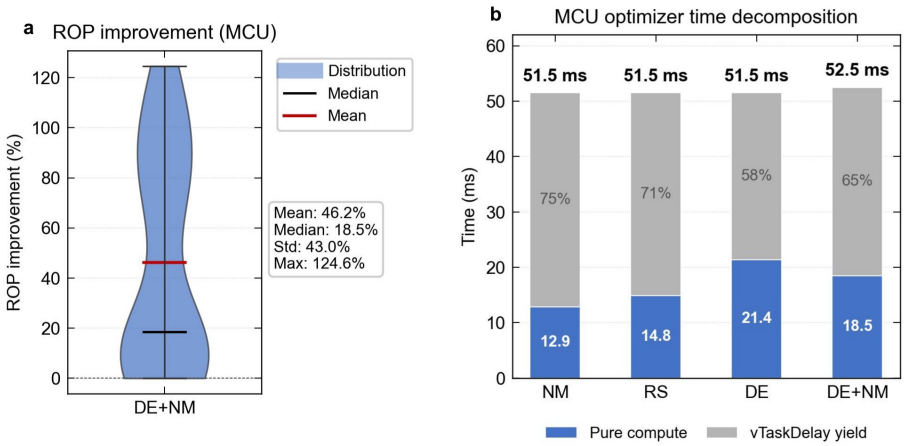


Figure 10. (**a**) Surrogate-model-predicted ROP improvement distribution for DE + NM on the ESP32-S3 across 41 evaluation points (Mean: 46.2%, Median: 18.5%, Std: 43.0%, Max: 124.6%). (**b**) End-to-end optimization latency decomposition for the four MCU-compatible algorithms, showing pure compute time (blue) and vTaskDelay yield overhead (grey). All four algorithms complete within 52.5 ms.

Figure 10b decomposes the end-to-end optimization latency. NM consumes 12.9 ms of pure compute plus 38.6 ms of yield (51.5 ms total); RS 14.8 ms plus 36.7 ms (51.5 ms); DE 21.4 ms plus 30.1 ms (51.5 ms); and DE + NM 18.5 ms plus 34.0 ms (52.5 ms). The yield fraction ranges from 58% (DE) to 75% (NM), reflecting the FreeRTOS scheduler's cooperative multitasking design. Even the slowest configuration (DE + NM at 52.5 ms) leaves a 47.5 ms margin within the 100 ms control-loop budget.

A concrete illustration is provided by the first evaluation point (depth 2300.0 m, shown as Sample 1 of 200 in the dashboard replay). Under the current drilling parameters (WOB = 52.9 kN, RPM = 120 rev/min, flow rate = 1817 L/min), the model predicted an ROP of 14.33 m/h. The optimizer converged to WOB = 72.0 kN, RPM = 167 rev/min, and flow rate = 2090 L/min, yielding a predicted ROP of 26.51 m/h, a surrogate-model-predicted improvement of 85.0%. The parameter shifts are physically interpretable: increased weight on bit and rotary speed apply greater mechanical energy to the rock face, while the elevated flow rate enhances cuttings transport efficiency, preventing balling and preserving bit tooth sharpness.

The mean surrogate-model-predicted ROP improvement across all optimization points was +46.2%, with a median of +18.5%, the disparity between mean and median indicating a right-skewed distribution in which several points achieved exceptionally large gains. The maximum single-point improvement reached +124.6%, while the minimum was 0.00%, occurring at points where the Layer 2 floor was active. This clamp is a deployment-pipeline policy rather than a statistical device: the recommendation pipeline never recommends a parameter set predicted to be worse than the current operating point, which is a deployment requirement for any advisory optimization system.

All optimization gains reported here are surrogate-model-predicted: the optimizer is a decision layer whose recommendations precede execution by definition, and its improvements are computed from the surrogate model's output rather than measured in the field. The surrogate itself is validated by trend consistency on blind whole wells: on set_2, the archived repeated-run distribution centers on a mean WMAPE of 0.6517 and a median of 0.6133 (Table 2), with Figure 4 serving only as a qualitative trend illustration (the

displayed run's per-well WMAPE values are disclosed in its caption), whereas the baseline estimators deviate from the overall trend. WMAPE quantifies exactly this trend agreement, which is why it, and not R^2-style correlation scores, serves as the primary accuracy metric (Section 2.5). Field confirmation of the predicted gains remains future work.

Figure 11 presents the on-device smart-sensor validation dashboard for this interval. The top panel shows the laboratory setup; the MCU autonomously executes the complete decision pipeline, comprising TFLite Micro INT8 inference (5.6 KB, 280 parameters) followed by DE + NM optimization (budget = 50), with the ×86 host acting only as a surrogate sensor-injection terminal. The bottom-left well-log profile panel visualizes the 2300–2500 m interval, showing the actual ROP (black), INT8 prediction on the ESP32-S3 (orange), and optimized ROP (green dots) alongside the normalized engineering parameters. The right-hand panels display the current-sample state at 2300.0 m: predicted ROP 14.33 m/h, an end-to-end per-sample processing time of 0.42 ms, and the full optimization result: WOB from 52.9 to 72.0 kN, RPM from 120 to 167 rev/min, and Flow from 1817 to 2090 L/min, yielding a surrogate-model-predicted ROP improvement of +85.0% (from 14.33 to 26.51 m/h) in 18.8 ms. This dashboard embodies the concept of an autonomous edge decision node for ROP optimization: a microcontroller-class device that not only acquires sensor data but also performs real-time inference, on-device optimization, and decision generation entirely at the point of data generation, without telemetry dependency or surface intervention. The demonstration establishes that a commodity MCU can host a complete decision-generation pipeline for a continuous industrial regression task, a result that encourages future inquiry into whether similarly constrained edge systems can support other downhole decision tasks.

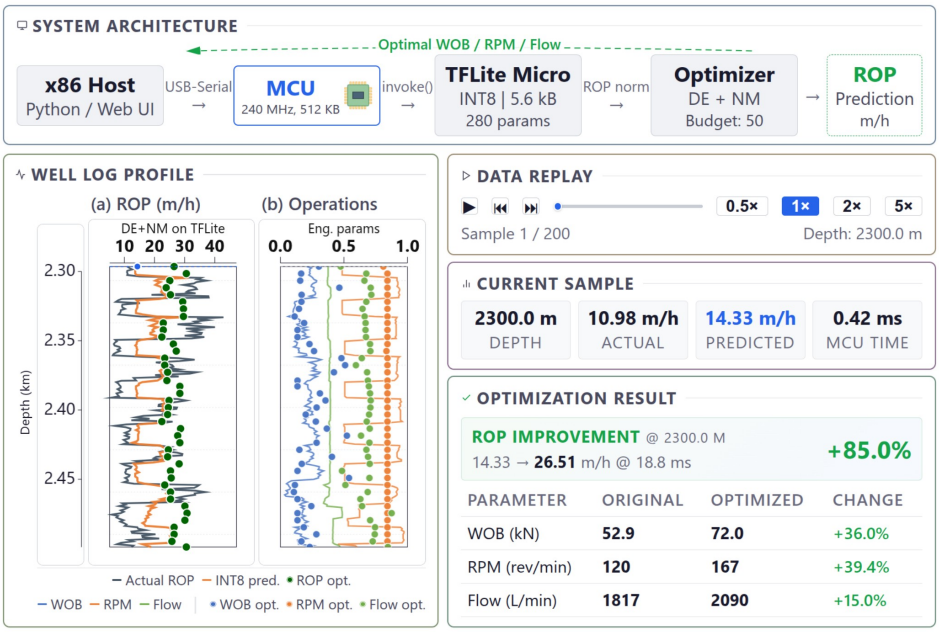


Figure 11. On-device smart-sensor validation dashboard for the Pohokura_South-1 2300–2500 m interval. Top: host-to-MCU offline replay setup; the x86 host injects drilling-state vectors into the ESP32-S3 MCU and receives the resulting logs, but does not participate in inference or optimization. Bottom-left: well-log profile with actual ROP (black), INT8 prediction on the ESP32-S3 (orange), and optimized ROP (green dots), alongside normalized engineering parameters. Bottom-right: current-sample readout at 2300.0 m showing the predicted ROP (14.33 m/h), end-to-end per-sample processing time (0.42 ms, including serialization and USB-serial logging overhead; distinct from the 156.38 µs concise I/O-quantized inference-loop benchmark of Table 6), and optimization result (ROP improvement +85.0%, total latency 18.8 ms).

Table 6. Cross-platform INT8 inference latency (microseconds, per sample).

Platform	CPU	ROP INT8	MLP INT8
x86_64 PC	Intel Ultra 9 @ 5 GHz	1.57	1.87
Raspberry Pi 4	ARM Cortex-A72 @ 1.5 GHz	29.32	33.37
ESP32-S3	Xtensa LX7 @ 240 MHz	156.38	—

4. Discussion

4.1. Performance Under Extreme Parameter Constraints

The experimental results consistently address each of the three research questions in Section 1.

With respect to generalization performance under extreme parameter scarcity, ROP MCU_model employs exactly 280 trainable parameters, a figure that stands in sharp contrast to the hundred-thousand to fifty-million parameter budgets typical of contemporary deep learning approaches for ROP prediction [13,16]. Despite this three-to-six-order-of-magnitude reduction in model size, on the set_2 test split the full-precision ROP MCU_model achieves a best WMAPE of 0.502 and a mean-model WMAPE of 0.616. The mean of the per-split robust-mean WMAPEs across the nine seeded evaluation splits is 0.687, which is the operationally relevant figure for assessing generalization across diverse well combinations. The leave-two-well-out protocol deliberately withholds two wells per split, reducing training diversity and amplifying generalization stress. That ROP MCU_model achieves WMAPE 0.616 on set_2, a split that simultaneously combines the proximal Pohokura_South-1 (4.4 km from the nearest training well) and the distal Maari-1 (>60 km from all training wells), demonstrates that the contrastive embedding compensates for both extreme parameter scarcity and spatial heterogeneity. The mechanism underlying this capability is examined in detail in Section 2.4 and further quantified through the ablation study presented in Section 4.2.

Regarding inference latency on resource-constrained hardware, the ESP32-S3 implementation achieves an average of 156.38 microseconds per prediction, with individual measurements falling within a narrow 148–167 microsecond range across 200 sequential evaluations. This corresponds to a sustained throughput of 6394 inferences per second, exceeding the one-millisecond target by a factor of 6.4. The INT8-quantized model at 5.7 KB occupies less than 0.1% of the available flash, and the 32 KB tensor arena leaves 93% of the ESP32-S3's SRAM budget unallocated for application logic.

Concerning computational autonomy, the on-device optimization engine completes a full parameter search with a budget of 50 function evaluations in 12.9–21.4 ms of pure compute time, or approximately 51.5 ms when accounting for cooperative multitasking yields. Both figures sit comfortably within a 100–200 ms window, which is quoted here only as a computational reference budget for on-device search, not as a drilling control period validated by this study. The mean surrogate-model-predicted ROP improvement of +46.2% (median +18.5%) across 41 field-data points, with all solutions screening within the MCU-side bounded intervals and the deployment-pipeline feasibility pre-screen, establishes that the on-device portion of the prediction–optimization–decision pipeline can execute on a microcontroller positioned at or near the bit, while floor clamping and command authorization remain in the deployment pipeline, though actuation and rig-level control remain outside the present scope.

Finally, regarding scalability, the computationally expensive stages of the methodology, namely genetic-algorithm architecture search and the repeated-run training protocol, are offline, one-time costs incurred before deployment. The recurring cost of the deployed artifact is fully characterized: 5704 bytes of storage and 156.38 µs per inference on the

ESP32-S3. Extending the system to additional wells or rigs therefore requires only offline retraining and firmware re-flashing, with no change to the on-device resource envelope.

4.2. Contrastive Embedding as a Consistent Stabilizer Under the Set-2 Ablation Protocol

To isolate the contribution of the contrastive regularization term, we conducted an ablation study spanning ten loss–weight configurations. Each configuration was evaluated using the Gen-10 architecture with individual ID-177, trained on the seven training wells of set_2, and every configuration underwent 50 repeated training runs to ensure statistical reliability; the reported WMAPE is the robust mean of the middle 60% of runs, evaluated as the mean model over the two blind wells of set_2 (Pohokura_South-1 and Maari-1).

The results show a clear pattern: the pure regression configuration C01 ($\lambda_{\text{contrast}} = 0.0$, $\lambda_{\text{regress}} = 1.0$) produced the highest prediction error of all ten configurations (WMAPE = 0.6645), and every contrastive configuration outperformed it, with WMAPE values spanning 0.588–0.639. The benefit of the embedding constraint therefore does not depend on a finely tuned loss–weight ratio. The best configuration, C04 (1:1 ratio), achieves a WMAPE of 0.5881, an 11.5% reduction from the pure-regression baseline. The deployed configuration C10 (1:4 ratio) yields a 3.9% reduction (0.6386) under this protocol. Two additional points are worth stating explicitly. First, the values in Table 7 derive from the set_2 ablation protocol (mean-model evaluation over the two blind wells of set_2) and are not directly comparable to the deployed-model benchmark figure of 0.687, which is the mean over nine seeded evaluation splits. Second, the deployed 1:4 ratio is not the ablation optimum: its 0.6386 lies within run-to-run variability of the deployed model's set_2 estimate (0.616) in the nine-split benchmark, and its selection is inherited from earlier experiments conducted before the full ratio sweep.

Table 7. Ablation study: effect of contrastive-to-regression loss ratio on prediction errors.

Config	$\lambda_{\text{contrast}}$	λ_{regress}	C:R Ratio	Test WMAPE	Error Reduction vs. C01
C01	0.0	1.0	0:1 (pure regression)	0.6645	—
C02	0.2	1.0	1:5	0.5948	+10.5%
C03	0.5	1.0	1:2	0.6430	+3.2%
C04	1.0	1.0	1:1 (best)	0.5881	+11.5%
C05	1.0	0.5	2:1	0.5943	+10.6%
C06	2.0	0.2	10:1	0.6132	+7.7%
C07	2.0	0.5	4:1	0.6021	+9.4%
C08	2.0	1.0	2:1	0.6050	+9.0%
C09	3.0	1.0	3:1	0.5923	+10.9%
C10	0.5	2.0	1:4 (deployed)	0.6386	+3.9%

Beyond raw prediction error, the contrastive term exerts a pronounced stabilizing effect on training dynamics. Across the 50 repeated training runs, the run-to-run standard deviation of the evaluation statistic falls from 0.0845 under pure regression (C01) to 0.027–0.049 across the nine contrastive configurations, indicating that without the embedding-space structure the optimizer wanders across an unbounded latent landscape, occasionally stumbling upon a favorable initialization but more often settling in suboptimal basins. The contrastive term acts as a geometric prior that collapses the latent manifold into an ordered, ROP-monotonic structure, effectively reducing the effective search dimension and making the regression task well-posed despite the extreme parameter scarcity.

A further interpretation of the C01 failure mode highlights the temporal structure of drilling data. Conventional time-series models (LSTM, GRU, and Transformer) rely on an explicit memory of recently drilled formations to infer lithological transitions and predict ROP. ROP MCU_model deliberately decouples this temporal dependency: its eight inputs are purely instantaneous engineering parameters with no historical depth context. Under

this severe information restriction, the contrastive manifold becomes the sole mechanism through which embedding continuity is shaped; the embedding clusters effectively stand in for an explicit record of prior context at the resolution of ROP-level similarity structure. The ablation results support this interpretation: when the manifold structure is removed (configuration C01), the model loses its only source of contextual information and the run-to-run standard deviation of the evaluation statistic rises from 0.027–0.049 (C02–C10) to 0.0845. Conversely, the success of configurations C02–C10 demonstrates that a well-ordered latent geometry encodes sufficient ROP-level similarity structure to permit accurate prediction from a single instantaneous state vector. This temporal decoupling is also what enables the kilobyte-scale memory footprint, because eliminating recurrent states or attention caches avoids the parameter inflation typical of sequence models. A natural extension is to relax this decoupling: feeding a short historical buffer of recent formation states into the encoder would combine the geometric regularization of contrastive learning with the sequential inference capability of time-series models, and is expected to sharpen prediction accuracy in thinly interbedded intervals where lithology changes rapidly with depth.

A per-stage ablation of the remaining pipeline components would not be informative: neural architecture search, INT8 quantization, and the on-device optimizer are orthogonal enablers whose outcomes are measured independently: parameter count and topology (Table 4), model size and inference latency (Tables 5 and 6), and predicted ROP improvement (Table 8), and none of them interacts with the objectives of the others. The contribution-relevant ablation is therefore the contrastive constraint itself (C01–C10), which isolates the effect of embedding-space structure on prediction error and training stability.

Table 8. Summary of surrogate-model-predicted drilling parameter optimization results (41 points, DE + NM, budget = 50).

Metric	Value
Optimization points	41
Mean surrogate-model-predicted ROP improvement	+46.2%
Median surrogate-model-predicted ROP improvement	+18.5%
Maximum ROP improvement	+124.6%
Minimum ROP improvement	0.00% (Layer 2 floor)
Pre-optimization mean ROP	19.12 m/h
Post-optimization mean ROP	25.98 m/h

4.3. Closing the Loop On-Device

A common misconception in drilling automation is that the information gap stems from an inability to acquire downhole data. In reality, modern measurement-while-drilling (MWD) and logging-while-drilling (LWD) systems continuously acquire weight-on-bit, torque, rotary speed, annular pressure, temperature, and vibration measurements at sampling rates of 25 Hz or higher, often from sensors positioned within meters of the bit. The five engineering parameters required for ROP prediction, namely WOB, RPM, torque, flow rate, and standpipe pressure, are already measured downhole by standard MWD suites.

The bottleneck is not acquisition; it is the location of the decision-making node. Because current practice streams all measurements to surface before processing, the control loop is bounded by round-trip telemetry latency (Section 1). By the time a parameter adjustment returns to the bit, the formation state at which it was computed may no longer exist. For time-critical tasks such as pressure control or lost-circulation response, the consequences are more severe than for ROP. Alternative telemetry technologies offer only partial relief: electromagnetic telemetry offers higher bandwidth than mud pulse but remains depth-limited by signal attenuation [35], and wired drill pipe, while higher bandwidth [6], still leaves the optimization algorithm running on surface servers with the control loop bounded by

physical distance. At the surface, data aggregation systems downsample the high-frequency sensor streams before they reach the optimizer, compounding the information loss.

This study uses historical surface-archived data from the Taranaki Basin to train and validate the ROP MCU_model, consistent with standard post-well analysis workflows. However, the architectural design target is fundamentally different. Rather than streaming raw sensor values through a telemetry bottleneck, the proposed system relocates ROP optimization decision-making to a microcontroller positioned at or near the bit. The MCU ingests full-resolution downhole sensor data locally, executes the INT8-quantized neural network in 156 µs, runs the DE + NM optimizer in 18.5 ms, and autonomously outputs optimized WOB, RPM, and flow-rate decisions, all without surface communication. The payload transmitted upward is reduced from continuous high-frequency data streams to a few bytes of optimized decisions per cycle. To contextualize this bandwidth reduction quantitatively, we first estimate the raw sensor data flux at the downhole acquisition layer. Modern MWD/LWD suites simultaneously acquire approximately ten channels of drilling data, including weight on bit, torque on bit, rotary speed, standpipe and annular pressures, flow rate, downhole temperature, triaxial vibration, inclination, and azimuth, with channel-specific sampling rates ranging from 1 Hz for temperature and directional parameters to 50–100 Hz for mechanical and dynamic measurements. Taking a weighted average sampling rate of approximately 20 Hz per channel and 32-bit floating-point encoding (4 bytes/sample), the aggregate raw data flux is $10 \times 20 \times 4 = 800$ bytes/s. This figure represents the data volume accumulated in downhole memory prior to any compression or telemetry-dependent downsampling; the actual mud-pulse uplink capacity is typically below 10 bps (approximately 1–3 bytes/s), which is why MWD data are aggressively downsampled for real-time transmission.

By contrast, the proposed on-device optimization engine produces a decision packet comprising three optimized controllable parameters (WOB, RPM, and flow rate) plus a scalar predicted ROP value. Encoded as four 8-bit integers, each packet occupies 4 bytes. At a typical optimization cadence of one decision every 60 s (consistent with operational drilling control cycles), the average uplink requirement is $4/60 \approx 0.067$ bytes/s, comfortably below the 0.1 bytes/s threshold. The present study provides a host-to-MCU offline replay proof-of-concept validation of this architecture using historical surface-archived drilling records; the bandwidth figures are engineering extrapolations based on standard MWD acquisition specifications, reflecting the theoretical sensing–communication co-design gain [36] when the full decision loop is relocated to a near-bit microcontroller. This yields a reduction exceeding four orders of magnitude relative to raw streaming.

This volume fits comfortably within a single mud-pulse telemetry frame even at the lowest bandwidth. The computational cost of this edge decision node is roughly one-thousandth of conventional surface-based infrastructure, and the same architectural principle of relocating the decision-making node to the point of measurement is expected to benefit other telemetry-constrained downhole tasks.

4.4. Comparison with Alternative Approaches

Table 9 provides a multi-dimensional comparison of ROP MCU_model against the principal competing paradigms for drilling intelligence.

The comparison reveals a clear pattern. Surface-based deep learning approaches achieve high predictive performance on benchmark datasets (R^2 typically exceeding 0.95) [16], yet they are inseparable from high-performance computing infrastructure, requiring continuous telemetry and human-in-the-loop oversight. Edge computing systems reduce latency relative to cloud architectures but remain surface-bound and unable to operate during telemetry interruptions. Traditional physics models are lightweight yet

require per-formation recalibration and suffer from poor generalization in heterogeneous geology. ROP MCU_model occupies a previously empty region of this design space: robust generalization (mean WMAPE 0.687 over nine seeded evaluation splits, outperforming every tested baseline in both mean error and cross-split stability), extreme portability, and full on-device ROP decision autonomy simultaneously.

Table 9. Multi-dimensional comparison of drilling intelligence approaches. Note: the 100 ms figure is a computational reference budget for on-device search, not a drilling control period validated by this study, and the 156 µs inference and 18.5 ms optimization figures demonstrate on-device computational feasibility.

Dimension	Surface-Based DL [16]	Edge Computing [37]	Traditional Physics [38]	ROP MCU_Model (Ours)
On-device compute latency	100–300 ms	<10 ms	Moderate	156 µs (inf.) + 18.5 ms (opt.)
Hardware cost	>$5000 (GPU server)	$5000–$25,000 (ruggedized)	Negligible	<$5 (consumer-grade MCU)
Power consumption	100–500 W	10–50 W	Negligible	<1 W (typically 0.5 W)
Bandwidth dependency	Continuous high-bandwidth telemetry	Moderate (local aggregation)	None	None (fully offline)
Autonomous on-device decision-making	No	Limited (advisory only)	No	Yes (full local surrogate-model-predicted ROP decision loop)
HTHP tolerance	N/A (surface-based)	N/A (surface-based)	N/A	Portable to industrial MCUs rated 150/175 °C

4.5. Implications for Multiple Communities

The implications of this work extend across three distinct technical communities. For the sensors and edge AI community, this study provides, to the authors' knowledge, one of the first demonstrations that a neural network with fewer than one thousand parameters can generate advisory decisions on a real-world industrial regression task. Standard TinyML benchmarks focus overwhelmingly on classification; the present work shows that resource-constrained regression, where predictions must respect continuous physical scales, ordinal relationships, operational safety bounds, and subsequent decision-making, is equally feasible on microcontroller-class hardware. The combination of hardware-aware neural architecture search, INT8 quantization, and supervised contrastive learning in one deployed system offers a design methodology that may inspire similarly constrained edge decision systems in other industrial sensing domains.

For the petroleum engineering community, this work offers an on-device case study of surrogate-model-predicted near-bit decision generation. ROP MCU_model delivers lower mean WMAPE than all tested machine learning baselines, yet its computational cost is roughly one-thousandth that of conventional surface-based approaches. It generates advisory decisions without surface dependency. These results establish an important feasibility boundary: if kilobyte-scale models can support on-device ROP decisions, then the computational barrier to edge-autonomous drilling intelligence is lower than previously assumed. This encourages future, task-specific investigations into whether similarly lightweight edge systems can autonomously manage pressure anomalies, detect incipient kicks, or respond to lost circulation.

For the drilling automation community, the advance is concrete but carefully bounded. Current field practice treats ROP optimization as an advisory function: the surface system recommends parameters and a human driller decides whether to apply them [25]. Our implementation moves the ROP decision loop onto the MCU: the model predicts, the optimizer searches, each recommendation is checked against the bounded search envelope and the preliminary feasibility pre-screen, and operational decisions are autonomously generated locally, all without cloud connectivity. It provides, to the authors' knowledge, one of the first empirical demonstrations that the cognitive loop (sense, predict, optimize, and decide) can be closed on commodity hardware. The present work removes the argument that such edge autonomy is computationally infeasible.

A further route concerns carrying the demonstrated stages from simulation and offline replay onto field hardware. Every stage that can be exercised without a well has already been closed on the deployed device itself: recorded field logs are replayed end-to-end through the host-to-MCU offline replay protocol, so sensing, prediction, optimization, and decision generation execute on the microcontroller rather than only in simulation, while desktop testbench measurement stands in for the downhole effects that still await a full-scale drilling simulator or a live well. Moving the same quantized network onto specialized hardware is an integration exercise rather than a re-modeling exercise: porting to a different qualified microcontroller requires only recompilation for that target because the weights, the optimizer, and the quantized inference path carry over unchanged, while the companion work is supply regulation for the toolstring, packaging and thermal paths for the electronics, immunity to electromagnetic interference, and the telemetry interface along the assembly. The temperature-qualification argument that motivates choosing such a device is stated in the Limitations and Future Directions section (Section 4.6), and is deliberately not restated here; this paragraph only records how the simulation-to-implementation route builds on the offline replay already reported.

4.6. Limitations and Future Directions

Several limitations of the current work point naturally toward future research directions. A first limitation is that the current work remains a laboratory proof-of-concept. The ESP32-S3 is a consumer-grade microcontroller validated on a desktop testbench only; it has not been subjected to the shock, vibration, pressure, and electromagnetic interference encountered in a real borehole. Downhole temperatures routinely exceed 150 °C and may reach 200 °C in high-pressure high-temperature formations, whereas the ESP32-S3 is rated only to 85 °C in its standard consumer-grade configuration (industrial-grade variants reach 105 °C). These are hardware packaging and qualification problems, not algorithmic barriers: the 280-parameter architecture and the INT8 quantization pipeline impose no intrinsic thermal or mechanical constraints. Scaling to a real implementation follows a staged validation ladder whose well-free stages are already closed on the deployed device: offline replay of recorded field logs, with desktop testbench measurement standing in for the downhole effects (Section 4.5). No live-well closed-loop trial has been performed in this study, and accordingly no field-test numbers are reported here; full-scale drilling-simulator and live-well closed-loop validation therefore constitute the defined next milestone, where the model must demonstrate robustness to the vibrations, pressure pulses, and thermal transients of the downhole environment. For conventional high-temperature service, future work will port the deployed model to industrial-grade microcontrollers qualified for operation at least to 175 °C (for example, ceramic-packaged high-temperature variants), and for high-pressure high-temperature service to a radiation-tolerant high-temperature system-on-chip; porting requires only recompilation for the target instruction set, because the trained weights, the optimizer, and the INT8 inference path carry over unchanged.

A second limitation concerns the gap between autonomous decision-making and field execution. The system autonomously computes optimal drilling parameters and generates advisory decisions on-device; however, the reported ROP improvements are surrogate-model-predicted gains that require field validation before operational reliance, and formation heterogeneity may limit how much of a predicted gain translates into realized penetration rate. Moreover, the translation of these decisions into physical rig movements must account for bottom-hole assembly mechanical response delay, mud circulation system lag, and rig hydraulic actuation limits. These dynamics are formation-specific and beyond the scope of the present decision-support module.

A related evidential boundary should be stated plainly. The controllable parameters are observational variables chosen in response to formation, bit condition, and operating constraints, so the surrogate learns associations that can be confounded by those conditions; a high surrogate-predicted ROP may mark an operating state that coincides with the recommended parameters rather than one caused by them, and an optimizer driven purely by surrogate output can in principle exploit this mismatch outside the support of the training record. The study includes no randomized or intervention-based drilling data, no high-temperature or high-pressure experiments, and no closed-loop trials on a live well; empirical support is limited to host-to-MCU offline replay of recorded field logs and desktop testbench measurements, and no further validation is claimed. Comparing recommended parameter changes with observed ROP responses to comparable historical changes under similar formation conditions is left to future work.

Third, and most fundamentally, the architecture, training protocol, and safety constraints developed in this study are specific to the ROP optimization task. The encoder–regressor topology was evolved under a fitness function tailored to ROP prediction error; the contrastive bins were derived from uniform ROP intervals; and the safety bounds were anchored to the Taranaki Basin WOB, RPM, and flow-rate distributions. Extending this approach to other downhole decision tasks, such as well-control risk assessment, managed-pressure drilling, or lost-circulation response, would require task-specific re-engineering of the input features, objective functions, and safety constraints, along with acquisition of appropriate training data. The present work is therefore understood as a rigorous feasibility study for ROP-specific edge autonomy, with the understanding that each downstream task will require its own domain-specific adaptation. Moreover, the bounded search envelope and preliminary feasibility pre-screen cover only the controllable parameters modeled here; constraints such as torque, vibration, mechanical specific energy, differential pressure, stick-slip, bit wear, hole cleaning, and equipment ratings are not enforced by the present mechanism and are left to the upstream command layer and to future work.

Fourth, the model input space is restricted to eight routinely measured drilling parameters (MD, TVD, bit size, WOB, pipe pressure, torque, RPM, and flow rate) and therefore omits several physical factors that strongly influence ROP. Bit type (e.g., PDC versus roller-cone), bit wear state, and bottom-hole assembly configuration are not encoded; neither are drilling dynamics such as stick-slip oscillations, whirl, or axial vibrations, nor rock-mechanical properties such as unconfined compressive strength or Poisson's ratio. In the current TinyML regime, simply adding these variables as extra input nodes would inflate the parameter count and break the kilobyte memory budget. A promising alternative is to fold them into the contrastive manifold as physics-informed constraints: rather than expanding the input vector, one could regularize the embedding geometry so that geologically similar states cluster more tightly and wear-induced drift is captured as a slow trajectory on the latent manifold. Developing such physics-aware regularizers within a sub-kiloparameter budget is a natural extension of the present work.

A separate design boundary concerns the search and training configuration itself. The deployed contrastive-to-regression loss weight has not been re-ablated: no experiment re-selected that weight using training and validation data alone, and the deployed choice should be read as inherited rather than as a validated optimum. The neural architecture search space and its training hyperparameters have not been studied under cross-dataset transfer, and we claim no separation between hyperparameter selection and evaluation wells beyond the whole-well split boundaries of Section 2.5.

Fifth, the Taranaki Basin dataset represents a single geological province. While the USROP dataset was employed during NAS to ensure topology generalization across at least two basins, multi-basin validation on Middle East carbonates, North Sea clastics, and Permian Basin unconventional formations would strengthen generalization claims. We anticipate that the contrastive embedding structure will facilitate such transfer, as states of similar ROP-level structure should cluster regardless of basin-specific parameter ranges.

Sixth, the deployed model is fixed after training. Continuous adaptation through lightweight on-device learning or federated fleet-wide updates would enable the model to evolve with changing geology without surface intervention. A complementary enhancement is to incorporate the sequential time-depth structure of the drilled section: by maintaining a short historical memory of recent formation states in the encoder input, the model could adapt its latent geometry to local lithology and achieve stronger fitting accuracy when the bit traverses thinly interbedded or rapidly changing strata.

A further boundary concerns how far the reported results extrapolate. The evidence rests on a single field dataset acquired within one structural setting of the Taranaki Basin and standardized to a single pre-split data contract, so the accuracies and improvements reported here characterize one instrumentation and logging context; performance across other instruments, structural settings, or depth ranges has not been established. This uncertainty is about data breadth rather than about physical content: drilling dynamics enter the model only through the measured parameter time series, and the acknowledged feature limitations remain the unmodeled bit wear and rock properties noted above. Applying the methodology to a different field or downhole decision task proceeds by re-running the neural architecture search and retraining offline on local data rather than by online adaptation or transfer fine-tuning of a trained model; the present results contain no such weight adaptation.

5. Conclusions

We present ROP MCU_model, a neural network with only 280 parameters that predicts rate of penetration and optimizes drilling parameters on a commodity microcontroller. Discovered through genetic-algorithm architecture search and trained with supervised contrastive learning, the model maps each drilling state to a point in a low-dimensional embedding space where states of similar ROP-level structure cluster together. Full-integer quantization compresses the network to 5.7 KB, allowing inference in 156 microseconds on an ESP32-S3. Tested under a strict leave-two-well-out protocol across nine seeded evaluation splits, the model achieves an average WMAPE of 0.687, on par with conventional ensemble methods that use orders of magnitude more parameters, and the on-device optimizer raises surrogate-model-predicted ROP by roughly 46% while keeping all solutions within the bounded search envelope and preliminary feasibility pre-screen.

Two design choices make this possible. First, the model responds only to the current instantaneous drilling state, eight parameters measured at a single depth, rather than maintaining a memory of previously drilled formations. The contrastive loss supplies the missing ROP-level similarity structure by structuring the embedding space so that nearby points of similar ROP-level structure lie close together. The contrastive constraint reduces

prediction error relative to pure regression across all nine tested loss–weight ratios under the set_2 ablation protocol (by 11.5% for the best ratio) and cuts the run-to-run standard deviation of training outcomes from 0.0845 to 0.027–0.049, indicating a consistent benefit of the embedding geometry across the ablated ratios when both parameters and history are scarce. Second, the architecture search explicitly co-optimizes accuracy and model size, preventing the parameter inflation typical of deep learning approaches.

The outcome is a drilling intelligence system that autonomously predicts ROP and generates surrogate-model-predicted optimization recommendations entirely on-device, eliminating dependence on surface telemetry for this specific task. Because the algorithmic stack is hardware-agnostic, the same 280-parameter network can be ported to industrial microcontrollers rated for 150 °C or higher, opening a path to practical near-bit deployment for ROP optimization. Its contribution is, to the authors' knowledge, one of the first rigorous demonstrations that on-device edge decision-making is achievable for a continuous industrial regression task under extreme resource constraints. We hope this result encourages future task-specific investigations into edge-autonomous well-control, pressure management, and geological hazard response, each of which will require its own models, data, and safety frameworks but may now proceed with greater confidence that the underlying computational platform is sufficient. Future directions include adding bit wear, drilling dynamics, and rock-mechanical properties as physics-informed constraints; relaxing the temporal decoupling by feeding short historical buffers into the encoder; and validating the decision-to-actuation interface on a full-scale drilling simulator or live well.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/app162110465/s1, MCU_Offline_Replay_Dashboard.html—a browser-based offline visualization tool for replaying and inspecting MCU execution logs, including sensor time-series, optimizer trajectories, and recommended parameter tables.

Author Contributions: Conceptualization, Z.G.; Methodology, Z.G.; Formal analysis, Z.G.; Investigation, Z.G., B.Z. and Z.B.; Writing—original draft, Z.G.; Writing—review & editing, C.F., B.Z., Z.B. and Y.W.; Supervision, C.F. and Y.W.; Funding acquisition, Z.G. and Y.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (grant number 42102351) and the Institute of Geology and Geophysics, Chinese Academy of Sciences (grant number E5555801). The APC was funded by the Institute of Geology and Geophysics, Chinese Academy of Sciences (Grant No. E5555801).

Data Availability Statement: The data used in this study derive from New Zealand's publicly available petroleum exploration data packages. The complete architecture of the proposed model is disclosed in the main text (Table 4, Figures 1 and 2). The trained demonstration model is shared with the supplementary material MCU_Offline_Replay_Dashboard.html, in which the inference–optimization workflow can be run and reproduced.

Acknowledgments: We thank the contributors of New Zealand Government Projects for Petroleum Exploration Data Pack, and the Center of Big Data and AI in Earth Sciences of IGGCAS in supplying facilities for simulation and practical data processing is greatly acknowledged.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Normalization-Protocol Robustness Analysis

To examine whether the fixed physical normalization bounds of Section 2.2 introduce any measurable information transfer from held-out wells, the complete evaluation was repeated under the strictest leakage-avoidance protocol. For each of the nine splits, all normalization bounds were recomputed using only the nine training-stage wells (seven training and two validation wells), applying the identical per-feature method (tail-clipped

operational bounds for the sensor channels, min–max ranges for the depth-related features, and the fixed physical range of 0–80 m/h for ROP). The two blind test wells of each split were therefore scaled entirely by training-stage statistics. All other settings were kept identical to the benchmark of Section 3: the Gen-10 ID-177 architecture, the deployed 1:4 loss weights, and the 50-run repeated-training protocol. The exact normalization constants employed by every model in this study are compiled in Table A1.

Table A1. Fixed normalization bounds for the eight model inputs and the ROP target.

Feature	Lower	Upper	Unit
Measured depth (MD)	1500	4999	m
True vertical depth (TVD)	1239.4416	4728.9948	m
Bit size	6	17.5	in
Weight on bit (WOB)	12.0102	209.8623	kN
Pipe pressure	6853.5293	28,081.6680	kPa
Torque	0	23.1845	kN m
Rotary speed (RPM)	49	195	rpm
Flow rate	776.0091	3546.9292	L/min
Rate of penetration (target)	0	80	m/h

All bounds were fixed once, before any well partitioning, as a single field-wide prior applied uniformly across the full field dataset; this is the sense in which they are *a priori* for modeling while still derived from field history. For the sensor channels the bounds are tail-clipped historical operational limits with $q_{\text{down}} = 0.01$ and $q_{\text{up}} = 0.95$ (the lower 1% and upper 5% tails are trimmed); the depth features use min–max ranges, and ROP uses its fixed 0–80 m/h physical range. An honest reproducibility note: re-deriving these constants on the ML-features partition (15,713 rows) against the full drilling-stage population (17,703 rows) reproduces them to numerical tolerance except the WOB lower bound, which differs by 12.96%, because merge-stage row dropping makes the partition a proper subset; these constants are therefore the record values, not a claim of bit-exact recomputation.

Table A2 compares the two protocols on the primary generalization metric. The training-stage-only protocol reproduces the physical-bounds benchmark within 0.2% on the nine-split mean (0.6855 versus 0.6869), and the per-split differences are not statistically significant (paired t-test, $p = 0.76$; Wilcoxon signed-rank test, $p = 0.91$). The largest single-split deviation (0.026 on set_4) lies within the run-to-run variability of a single configuration (standard deviation 0.03–0.05 across repeated runs, Section 4.2). Removing two wells from approximately 15,700 samples moves the tail-clipped physical bounds by far less than the model's own training stochasticity, which is why the measured effect of the normalization contract is negligible.

The set_2 ablation of Section 4.2 was likewise repeated under the training-stage-only protocol (Table A3). The central finding is preserved: the best contrastive ratio (1:1, WMAPE 0.606) outperforms pure regression (0.628), and every protocol comparison is consistent within the run-to-run variability. Individual ratios fluctuate between protocols, as the deployed 1:4 ratio lands below pure regression under the physical-bounds protocol and slightly above it under the training-stage-only protocol, which is the ratio-selection sensitivity already disclosed in Section 4.2, not a normalization artifact. The values in Table A3 are robust means over the 50 runs (middle 60%), the same statistic as the training logs of Section 4.2; they are not directly comparable to the mean-model evaluation values in Table 7, and the two statistics are reported separately throughout.

Table A2. Benchmark WMAPE under two normalization protocols (deployed 1:4 configuration; robust mean of the middle 60% of 50 runs per split).

Split	Physical Bounds (Fixed)	Training-Stage-Only Bounds	Difference
set_0	0.6599	0.6704	+0.0105
set_1	0.6299	0.6209	−0.0090
set_2	0.6159	0.6245	+0.0086
set_3	0.6158	0.6330	+0.0172
set_4	0.9734	0.9470	−0.0264
set_5	0.6546	0.6506	−0.0040
set_6	0.6897	0.6897	+0.0000
set_7	0.6997	0.6825	−0.0172
set_8	0.6436	0.6505	+0.0069
Mean	**0.6869**	**0.6855**	**−0.0014**
Pop. std	0.1050	0.0953	—

Table A3. Set_2 ablation under the two normalization protocols (robust mean of the middle 60% of 50 runs).

Configuration	Physical Bounds (Fixed)	Training-Stage-Only Bounds
C01 (0:1, pure regression)	0.6533	0.6282
C04 (1:1, best)	0.6019	0.6061
C10 (1:4, deployed)	0.6270	0.6507

Appendix B. Blind-Well Inventory and Statistical Comparisons

This section tabulates the field inventory behind the leave-two-well-out protocol described in the main text. The inventory covers eleven wells evaluated over nine seeded splits, with each split holding out two blind test wells and two validation wells while the remaining seven wells form the training set; the role counts accumulated by every well across the nine splits are given in Table A4, which also records per-well survey details, sample counts, depth range, ROP quartile display, and mean per-evaluation WMAPE. Table A5 reports the planar separations between the blind wells and their separation from the training pool, computed in the New Zealand Transverse Mercator coordinate system. Table A6 lists the exact well sets assigned to the test, validation, and training roles in all nine splits. All three tables were compiled from the archived benchmark metadata rather than re-derived by hand.

All comparisons were computed from the archived per-run error records, with no new model training; the per-run tests pool the archived runs of each split. Across the pooled set of 450 pairs, the model minus the MLP reference has mean paired delta −0.3097 with 95% CI from −0.351 to −0.2684; Wilcoxon $p < 0.001$ and paired t $p < 0.001$ indicate a statistically significant difference, and the model records lower error in 357 of 450 archived runs. Applied within each of the nine seeded splits, the same test against the MLP is significant in all nine splits; the smallest margin has mean paired delta −0.0983, Wilcoxon $p = 0.024$, and paired t $p = 0.005$, also statistically significant.

The pooled paired runs of the deployed configuration against pure-regression training cover 90 pairs, with mean paired delta −0.0114, CI −0.048 to 0.0253, Wilcoxon $p = 0.786$, and paired t $p = 0.539$; this margin is small and not statistically significant, consistent with the cautious wording retained in the main text, and the deployed configuration records lower error in 45 of 90 pairs. Cross-split uncertainty of the headline error over the nine re-partitions of one field inventory is mean 0.6884 across the split medians with CI 0.5981 to 0.7787 and mean 0.6987 across the split means with CI 0.6139 to 0.7835; these intervals

quantify dispersion over alternate partitions of a single inventory rather than independent field trials and bound the headline figure only under that reading.

Table A4. Eleven-well field inventory: leave-two-well-out role counts (test/validation/train appearances across the nine seeded splits), sample counts, measured-depth intervals, and ROP distributions; the mean per-evaluation WMAPE of the proposed model over every archived run in which the well served as a blind test well.

Well	Survey	Test/Val/Train	Samples	MD Range (m)	ROP Median [min–max] (m/h)	Mean WMAPE
Awakino-1	KAHU	1/0/8	520	1505–2897	9.9 [2.4–33.0]	0.842
Kaimiro-1	KAIMIRO-NGATORO	0/0/9	2191	1505–4998	12.9 [3.8–80.0]	–
Kora-1	KORA	0/0/9	407	1505–3413	27.2 [8.1–80.0]	–
Kea-1	MAARI	2/7/0	624	1545–3132	36.4 [5.8–80.0]	0.654
Maari-1	MAARI	2/0/7	680	1501–2180	20.0 [1.3–80.0]	0.605
Pukeko-1	PIPELINE	3/4/2	1048	3141–4189	10.0 [1.1–80.0]	0.831
Pohokura-1	POHOKURA	1/0/8	2731	1501–4231	30.5 [0.8–80.0]	0.674
Pohokura-3	POHOKURA	3/6/0	2440	1501–3814	28.4 [0.3–80.0]	0.631
Pohokura_South-1	POHOKURA	3/0/6	2379	1501–3779	14.8 [0.1–62.3]	0.635
Tui-1	TUI	1/1/7	2391	1501–3891	36.4 [2.1–80.0]	0.641
Hu_Road-1	WAIHAPA	2/0/7	302	2383–3346	4.5 [1.4–21.0]	1.615

Table A5. Blind-well pairings per split with planar distances (NZTM 2000): separation between the two withheld wells and the minimum distance from each blind well to any training or validation well of the same split.

Split	Blind Well A	Blind Well B	Pair Distance (km)	Min Distance to Train/Val Pool (km)
set_0	Hu_Road-1	Maari-1	111.4	34.5 (Hu_Road-1), 11.0 (Maari-1)
set_1	Pohokura_South-1	Pohokura-3	11.4	4.4 (Pohokura_South-1), 7.1 (Pohokura-3)
set_2	Maari-1	Pohokura_South-1	138.8	11.0 (Maari-1), 4.4 (Pohokura_South-1)
set_3	Pohokura_South-1	Pohokura-3	11.4	4.4 (Pohokura_South-1), 7.1 (Pohokura-3)
set_4	Hu_Road-1	Pukeko-1	136.5	34.5 (Hu_Road-1), 31.6 (Pukeko-1)
set_5	Tui-1	Pohokura-3	105.6	49.3 (Tui-1), 7.1 (Pohokura-3)
set_6	Pukeko-1	Pohokura-1	159.0	31.6 (Pukeko-1), 4.4 (Pohokura-1)
set_7	Pukeko-1	Kea-1	34.8	31.6 (Pukeko-1), 11.0 (Kea-1)
set_8	Awakino-1	Kea-1	180.6	46.3 (Awakino-1), 11.0 (Kea-1)

Table A6. Full split structure for all nine seeded partitions: the two blind test wells, the two validation wells, and the seven training wells of every split, as recorded in the benchmark metadata of record.

Split	Test Wells	Validation Wells	Training Wells
set_0	Hu_Road-1, Maari-1	Pohokura-3, Kea-1	Kora-1, Pohokura_South-1, Pohokura-1, Kaimiro-1, Awakino-1, Tui-1, Pukeko-1
set_1	Pohokura_South-1, Pohokura-3	Pukeko-1, Kea-1	Kora-1, Pohokura-1, Maari-1, Kaimiro-1, Awakino-1, Hu_Road-1, Tui-1
set_2	Maari-1, Pohokura_South-1	Pohokura-3, Kea-1	Kora-1, Pohokura-1, Kaimiro-1, Awakino-1, Hu_Road-1, Tui-1, Pukeko-1
set_3	Pohokura_South-1, Pohokura-3	Pukeko-1, Kea-1	Kora-1, Pohokura-1, Maari-1, Kaimiro-1, Awakino-1, Hu_Road-1, Tui-1
set_4	Hu_Road-1, Pukeko-1	Pohokura-3, Kea-1	Kora-1, Pohokura_South-1, Pohokura-1, Maari-1, Kaimiro-1, Awakino-1, Tui-1
set_5	Tui-1, Pohokura-3	Pukeko-1, Kea-1	Kora-1, Pohokura_South-1, Pohokura-1, Maari-1, Kaimiro-1, Awakino-1, Hu_Road-1
set_6	Pukeko-1, Pohokura-1	Pohokura-3, Kea-1	Kora-1, Pohokura_South-1, Maari-1, Kaimiro-1, Awakino-1, Hu_Road-1, Tui-1
set_7	Pukeko-1, Kea-1	Tui-1, Pohokura-3	Kora-1, Pohokura_South-1, Pohokura-1, Maari-1, Kaimiro-1, Awakino-1, Hu_Road-1
set_8	Awakino-1, Kea-1	Pukeko-1, Pohokura-3	Kora-1, Pohokura_South-1, Pohokura-1, Maari-1, Kaimiro-1, Hu_Road-1, Tui-1

References

1. Jarrot, A.; Gelman, A.; Kusuma, J. Wireless digital communication technologies for drilling: Communication in the bits/s regime. *IEEE Signal Process. Mag.* **2018**, *35*, 112–120. [CrossRef]
2. Klotz, C.; Kaniappan, A.; Thorsen, A.K.; Nathan, E.; Jahangir, M.; Lie, L. A New Mud Pulse Telemetry System Reduces Risks When Drilling Complex Extended Reach Wells. In Proceedings of the IADC/SPE Asia Pacific Drilling Technology Conference and Exhibition, Jakarta, Indonesia, 25–27 August 2008; Paper No. SPE-115203-MS. [CrossRef]
3. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge computing: Vision and challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646.
4. Deng, S.; Zhao, H.; Fang, W.; Yin, J.; Dustdar, S.; Zomaya, A.Y. Edge intelligence: The confluence of edge computing and artificial intelligence. *IEEE Internet Things J.* **2020**, *7*, 7457–7469. [CrossRef]
5. Qiu, T.; Chi, J.; Zhou, X.; Ning, Z.; Atiquzzaman, M.; Wu, D.O. Edge computing in industrial Internet of Things: Architecture, advances and challenges. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 2462–2488. [CrossRef]
6. Craig, A.D.; Jackson, T.A.; Ramnarace, D.; Schultze, R.; Briscoe, M. The Evolution of Wired Drilling Tools: A Background, History and Learnings from the Development of a Suite of Drilling Tools for Wired Drillstrings. In Proceedings of the SPE/IADC Drilling Conference and Exhibition, Amsterdam, The Netherlands, 5–7 March 2013; Paper No. SPE-163416-MS. [CrossRef]
7. Siahaan, H.B.; Bjørkevoll, K.S.; Gravdal, J.E. Possibilities of Using Wired Drill Pipe Telemetry During Managed Pressure Drilling in Extended Reach Wells. In Proceedings of the SPE Intelligent Energy Conference & Exhibition, Utrecht, The Netherlands, 1–3 April 2014; Paper No. SPE-167856-MS. [CrossRef]
8. Hegde, C.; Daigle, H.; Millwater, H.; Gray, K. Analysis of rate of penetration (ROP) prediction in drilling using physics-based and data-driven models. *J. Pet. Sci. Eng.* **2017**, *159*, 295–306. [CrossRef]
9. Barbosa, L.F.F.M.; Nascimento, A.; Mathias, M.H.; de Carvalho, J.A., Jr. Machine learning methods applied to drilling rate of penetration prediction and optimization—A review. *J. Pet. Sci. Eng.* **2019**, *183*, 106332. [CrossRef]
10. Soares, C.; Gray, K. Real-time predictive capabilities of analytical and machine learning rate of penetration (ROP) models. *J. Pet. Sci. Eng.* **2019**, *172*, 934–959. [CrossRef]
11. Kor, K.; Altun, G. Is support vector regression method suitable for predicting rate of penetration? *J. Pet. Sci. Eng.* **2020**, *194*, 107542. [CrossRef]
12. Ahmed, O.S.; Adeniran, A.A.; Samsuri, A. Computational intelligence-based prediction of drilling rate of penetration: A comparative study. *J. Pet. Sci. Eng.* **2019**, *172*, 1–12. [CrossRef]
13. Zhang, C.; Song, X.; Su, Y.; Li, G. Real-time prediction of rate of penetration by combining attention-based gated recurrent unit network and fully connected neural networks. *J. Pet. Sci. Eng.* **2022**, *213*, 110396. [CrossRef]
14. Wang, Y.; Lou, Y.; Lin, Y.; Cai, Q.; Zhu, L. ROP prediction method based on PCA-Informer modeling. *ACS Omega* **2024**, *9*, 23822–23831. [CrossRef]
15. Jiao, S.; Li, W.; Li, Z.; Gai, J.; Zou, L.; Su, Y. Hybrid physics-machine learning models for predicting rate of penetration in the Halahatang oil field, Tarim Basin. *Sci. Rep.* **2024**, *14*, 5957. [CrossRef]
16. Bai, K.; Jin, S.; Zhang, Z.; Dai, S. Drilling rate of penetration prediction based on CBT-LSTM neural network. *Sensors* **2024**, *24*, 6966. [CrossRef]
17. Lin, J.; Chen, W.M.; Lin, Y.; Cohn, J.; Gan, C.; Han, S. MCUNet: Tiny deep learning on IoT devices. *Proc. Adv. Neural Inf. Process. Syst.* **2020**, *33*, 11711–11722. [CrossRef]
18. Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 2704–2713. [CrossRef]
19. David, R.; Duke, J.; Jain, A.; Reddi, V.J.; Jeffries, N.; Li, J.; Kreeger, N.; Nappier, I.; Natraj, M.; Regev, S.; et al. TensorFlow Lite Micro: Embedded machine learning on TinyML systems. *Proc. Mach. Learn. Syst.* **2021**, *3*, 800–811.
20. Banbury, C.; Reddi, V.J.; Torelli, P.; Holleman, J.; Jeffries, N.; Kiraly, C.; Montino, P.; Kanter, D.; Ahmed, S.; Pau, D.; et al. MLPerf Tiny Benchmark. *arXiv* **2021**, arXiv:2106.07597. [CrossRef]
21. Lai, L.; Suda, N.; Chandra, V. CMSIS-NN: Efficient neural network kernels for ARM Cortex-M CPUs. *arXiv* **2018**, arXiv:1801.06601.
22. Warden, P.; Situnayake, D. *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*; O'Reilly Media: Sebastopol, CA, USA, 2020; ISBN 978-1492052043.
23. Abadade, Y.; Temouden, A.; Bamoumen, H.; Benamar, N.; Chtouki, Y.; Hafid, A.S. A comprehensive survey on TinyML. *IEEE Access* **2023**, *11*, 96892–96922. [CrossRef]
24. Capogrosso, L.; Cunico, F.; Cheng, D.S.; Fummi, F.; Cristani, M. A machine learning-oriented survey on Tiny Machine Learning. *IEEE Access* **2024**, *12*, 23406–23426. [CrossRef]
25. Macpherson, J.D.; de Wardt, J.P.; Florence, F.; Chapman, C.D.; Zamora, M.; Laing, M.L.; Iversen, F.P. Drilling-systems automation: Current state, initiatives, and potential impact. *SPE Drill. Compl.* **2013**, *28*, 296–308. [CrossRef]

26. Khosla, P.; Teterwak, P.; Wang, C.; Sarna, A.; Tian, Y.; Isola, P.; Maschinot, A.; Liu, C.; Krishnan, D. Supervised contrastive learning. In *NIPS'20: Proceedings of the 34th International Conference on Neural Information Processing Systems*; ACM: New York, NY, USA, 2020; Volume 33, pp. 18661–18673. [CrossRef]
27. Zoph, B.; Le, Q.V. Neural architecture search with reinforcement learning. *arXiv* **2016**, arXiv:1611.01578. [CrossRef]
28. Liu, H.; Simonyan, K.; Yang, Y. DARTS: Differentiable architecture search. In Proceedings of the 7th International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019. [CrossRef]
29. Liberis, E.; Dudziak, Ł.; Lane, N.D. µNAS: Constrained neural architecture search for microcontrollers. In *Proceedings of the 1st Workshop on Machine Learning and Systems (EuroMLSys '21)*; Association for Computing Machinery: New York, NY, USA, 2021; pp. 70–79. [CrossRef]
30. Burrello, A.; Risso, M.; Motetti, B.A.; Macii, E.; Benini, L.; Pagliari, D.J. Enhancing neural architecture search with multiple hardware constraints for deep learning model deployment on tiny IoT devices. *IEEE Trans. Emerg. Top. Comput.* **2024**, *12*, 780–794. [CrossRef]
31. Tunkiel, A.T.; Sui, D.; Wiktorski, T. Reference dataset for rate of penetration benchmarking. *J. Pet. Sci. Eng.* **2021**, *196*, 108069. [CrossRef]
32. Geng, Z.; Wang, Y.; Pan, W.; Yu, C.; Bai, Z.; Zhang, H. Real-time discrimination of earthquake signals by integrating artificial intelligence technology into IoT devices. *Commun. Earth Environ.* **2025**, *6*, 73. [CrossRef]
33. Hadsell, R.; Chopra, S.; LeCun, Y. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*; IEEE: Piscataway, NJ, USA, 2006; Volume 2, pp. 1735–1742. [CrossRef]
34. Bingham, M.G. *A New Approach to Interpreting Rock Drillability*; The Petroleum Publishing Co.: Tulsa, OK, USA, 1964. [CrossRef]
35. Schnitger, J.; Macpherson, J.D. Signal Attenuation for Electromagnetic Telemetry Systems. In Proceedings of the SPE/IADC Drilling Conference and Exhibition, Amsterdam, The Netherlands, 17–19 March 2009; Paper No. SPE-118872-MS. [CrossRef]
36. Fu, R.; Chen, J.; Lin, Y.; Zou, A.; Chen, C.; Guan, X.; Ma, Y. Smart sensing and communication co-design for IIoT-based control systems. *IEEE Internet Things J.* **2024**, *11*, 3994–4014. [CrossRef]
37. Chen, J.; Ran, X. Deep learning with edge computing: A review. *Proc. IEEE* **2019**, *107*, 1655–1674. [CrossRef]
38. Bourgoyne, A.T.; Young, F.S. A multiple regression approach to optimal drilling and abnormal pressure detection. *SPE J.* **1974**, *14*, 371–384. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.