

Entropy-driven adaptive neural architecture search for cell segmentation on edge devices

Yiwen Xu

UNSW Sydney

Tariq M. Khan

UNSW Sydney

Yanming Zhu

Griffith University

Yang Song

UNSW Sydney

Erik Meijering

`erik.meijering@unsw.edu.au`

UNSW Sydney

Article

Keywords:

Posted Date: October 16th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10168635/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Entropy-driven adaptive neural architecture search for cell segmentation on edge devices

Yiwen Xu¹, Tariq M. Khan¹, Yanming Zhu², Yang Song¹, and Erik Meijering^{1,*}

¹School of Computer Science and Engineering, University of New South Wales, Sydney, NSW, Australia

²School of Information and Communication Technology, Griffith University, Brisbane, QLD, Australia

*Corresponding author: erik.meijering@unsw.edu.au

ABSTRACT

Accurate and real-time segmentation of cells in microscopy images is a critical task in biomedical research. Incubator-like edge devices are increasingly being used for real-time monitoring of cellular activities, facilitating timely intervention, which is particularly important when studying dynamic cellular processes. However, adapting to the deployment requirements of diverse edge devices, particularly those with limited resources, presents significant challenges in network architecture design. This requires considering resource constraints and hardware metrics associated with different edge devices, necessitating rigorous control of model complexity and resource utilisation. To address these challenges, we propose a hardware-aware neural architecture search method based on the U-Net backbone to segment various cell types. By incorporating hardware-based penalty terms and an entropy-driven mechanism, our method optimizes the search strategy to maximize segmentation accuracy while ensuring compatibility with target hardware. We evaluated our approach on five resource-constrained hardware platforms and achieved competitive segmentation performance on ten 2D live-cell imaging datasets from the Cell Tracking Challenge, while maintaining favourable hardware efficiency.

1 Introduction

Microscopy imaging techniques, such as fluorescence microscopy, enable the study of cellular dynamics at single-cell resolution. This has allowed researchers to investigate cell migration and proliferation during tissue development and organogenesis in early embryonic stages. Live-cell microscopy image segmentation is a crucial step toward quantitative analysis of the data produced in these studies.¹ In recent years, advances in computer vision and deep learning have significantly improved the accuracy of cell segmentation.^{2,3} Integrating these techniques with edge computing can further enhance analysis efficiency and optimise workflows, particularly in scenarios requiring continuous, real-time monitoring. For instance, an incubator microscope based on edge devices can independently and distributedly monitor and manage cellular dynamics, thus avoiding the computational and bandwidth constraints of desktop workstations or servers. This capability is especially important in experiments that involve large numbers of cell samples or prolonged periods of continuous observation.⁴⁻⁶ Recent studies have demonstrated that edge devices can be successfully deployed for cell-related tasks, such as cell counting, phenotype analysis, tracking, and segmentation.⁷⁻⁹ By processing image data directly on edge devices, one can achieve stable real-time monitoring and timely intervention in rapidly changing cellular processes. This is particularly valuable for investigating dynamic phenomena such as cell migration, proliferation, and apoptosis.^{10,11}

Although deep learning offers powerful tools for cell segmentation, customising optimal network architectures for different cell types and staining methods remains challenging (Fig. 1), as it requires expertise and is time-consuming, hindering widespread adoption. Moreover, edge devices typically have limited resources, such as computational power, memory, and energy consumption, making it difficult to find network architectures suitable for different hardware configurations.¹² The task may be facilitated by neural architecture search (NAS) methods. In recent years, various NAS methods have been proposed for medical image segmentation.¹³⁻¹⁷ The predominant approach is to adopt a U-Net-based backbone and then search for optimal basic operations (BOs) in the upsampling and downsampling blocks. For instance, NAS-UNet,¹³ which was developed for the segmentation of 2D prostate MRI, liver CT, and neurosonography images, simplifies the search space by employing repeated upsampling and downsampling blocks. The method has been optimised and applied also to cell segmentation.¹⁴ BiX-NAS¹⁵ employs a bidirectional architecture search strategy with iterative encoder-decoder blocks. However, most NAS methods do not account for hardware constraints of edge devices, and thus the architectures discovered may have high computational complexity or memory demands, rendering them inefficient for deployment on edge devices.

The incorporation of hardware configurations into the search process is referred to as hardware-aware NAS.¹⁸⁻²⁰ Currently, hardware-aware NAS methods optimise network architectures by considering both network performance (e.g., accuracy) and hardware metrics (e.g., latency, memory usage) during the search. For example, ProxylessNAS¹⁸ estimates the inference speed

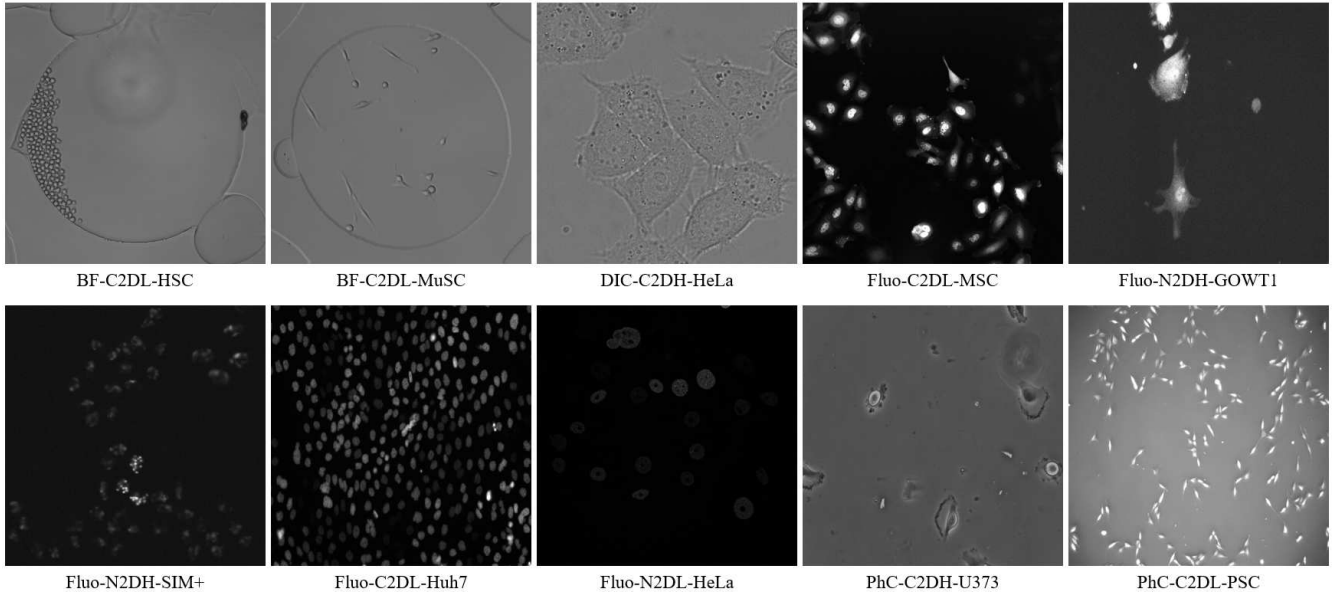


Figure 1. Example images from the 2D Cell Tracking Challenge datasets, illustrating different imaging modalities, including bright-field (BF), differential interference contrast (DIC), fluorescence (Fluo), and phase contrast (PhC), and diverse cell types: HSC (mouse hematopoietic stem cells), MuSC (mouse muscle stem cells), HeLa (Henrietta Lacks human uterine cervical carcinoma immortalized cells), MSC (rat mesenchymal stem cells), GOWT1 (mouse embryonic stem cells), SIM+ (second-generation simulated cells), Huh7 (human hepatocarcinoma-derived cells), U373 (human glioblastoma-astrocytoma cells), and PSC (pancreatic stem cells). Each dataset exhibits distinct imaging characteristics (e.g., signal-to-noise ratio, contrast, cell density, and morphological features), posing varying degrees of complexity for cell segmentation.

of a model on target hardware using a latency predictor and, in combination with model size evaluation, identifies efficient models suitable for edge devices. FBNet¹⁹ employs a differential-based architecture search approach, which measures the latency of BOs and creates a device-specific lookup table (LUT), incorporating hardware latency as a constraint during the search. This approach has been successful in generating efficient architectures for a variety of mobile devices. However, multiscale feature extraction and fusion are critical requirements for cell segmentation, especially when processing high-resolution microscopy images on edge devices. Current hardware-aware NAS methods are not tailored for microscopy images, limiting them in handling multiscale features for cell segmentation tasks.

To address these limitations, we propose a hardware-aware NAS method for cell segmentation based on the U-Net backbone architecture. Specifically, our approach constructs the U-Net network with nonrepetitive building blocks. This means that the BOs within each block can differ, allowing the network to flexibly and selectively capture features at different scales. Additionally, we create a LUT by measuring the latency and memory usage data of each BO on the target edge device. During the search process, we introduce penalties based on latency and memory usage data, ensuring that the discovered architecture can adapt to the target hardware. We found that introducing hardware-based penalties effectively constrains the model’s complexity but also excessively impacts segmentation accuracy. To address this issue and better balance segmentation performance with hardware-related metrics, we introduce an entropy-driven mechanism to ensure that each selected basic operation maximises its contribution to segmentation accuracy. We conducted tests on five different devices: Jetson Nano, PYNQ-Z2, ZUBoard-1CG, Raspberry Pi 5, and Raspberry Pi 5 with a HAILO-8 accelerator. Compared with other NAS methods for medical image segmentation, our approach achieves state-of-the-art performance.

2 Related work

Before presenting the proposed method, we briefly review related work in relevant areas, including microscopy cell segmentation, medical neural architecture search, hardware-aware neural architecture search, and the maximum entropy principle.

2.1 Microscopy cell segmentation

Recent studies on microscopy cell segmentation have addressed several challenges that are also relevant to live-cell analysis. To reduce annotation effort, Sadanandan et al.²¹ trained deep convolutional networks using automatically generated labels from fluorescently stained cells, and Din and Yu²² trained single-cell segmentation models without manual annotation. To improve

robustness, Moshkov et al.²³ applied test-time augmentation to deep-learning-based cell segmentation. For time-lapse analysis, Annasamudram et al.²⁴ combined multiscale detection, learning-based cell-region detection, and graph-based tracking. These methods improve annotation efficiency, robustness, or temporal modelling, but they generally rely on fixed or manually designed architectures and do not explicitly optimise latency or memory usage on target edge devices. Our work is complementary, as it focuses on hardware-aware NAS for live-cell segmentation under device-specific constraints.

2.2 Medical neural architecture search

With the advancement of deep learning (DL) in medical applications, medical NAS has emerged to bridge DL methods and the complex demands of medical data analysis.²⁵ In NAS, the search space defines the scope of network structures that can be explored. Many medical NAS studies tailor the search space according to the task and data type. These search spaces can be categorised into two levels: topology and operation. The topology level defines the backbone architecture. For instance, most medical image segmentation methods follow a U-Net-like structure,^{13,15,16,26} while others, such as tumour detection methods, employ chain-like search spaces.^{27,28} On the other hand, the operation level encompasses the BOs within the building blocks of the backbone architecture, such as convolution and pooling operations.

In this work, we focus on representative NAS methods specifically developed for 2D medical image segmentation based on the U-Net architecture. NAS-UNet¹³ serves as a foundational approach by employing a recursive U-Net design with a downsampling path to capture fine details and an upsampling path to recover spatial resolution. Although the recursive topology allows for the reuse of building blocks, it can also constrain performance improvements. Building on this framework, Zhu and Meijering¹⁴ enhanced the design for time-lapse microscopy image segmentation by refining the operation-level search space to include BOs better suited for cell images, and by assigning independent weights to each block in the upsampling and downsampling paths, thereby alleviating the limitations of block repetition.

Similarly, BiX-NAS¹⁵ leverages the U-Net topology and introduces bidirectional skip connections, which enable repeated utilisation of previously searched BOs. Its two-stage search strategy, combining differentiable architecture search (DARTS)²⁹ with a progressive evolutionary algorithm for skip connection optimisation, improves multiscale feature fusion and enhances overall segmentation performance.

In addition, we consider Auto-DeepLab,³⁰ a representative NAS method capable of searching U-Net architectures, and we also compare with the widely used nnU-Net model.²⁶ Although nnU-Net does not systematically search the network architecture space but instead focuses on hyperparameter tuning and pipeline configuration, it remains a strong baseline.

Live cell analysis on edge devices is often constrained by computational power, memory, and energy resources.^{4,31} Current methods are typically designed for resource-rich computing platforms, making their deployment in low-resource environments challenging. Therefore, we develop hardware-aware architecture search methods that can operate efficiently in resource-constrained environments. These methods must consider not only the accuracy and robustness of the network, but also the hardware constraints of edge devices, such as latency, memory usage, and power consumption.

2.3 Hardware-aware neural architecture search

Incorporating hardware constraints into NAS has garnered considerable attention.²⁰ Early methods such as MnasNet³² employed reinforcement learning to balance model accuracy with mobile device latency, whereas Single-Path One-Shot NAS³³ integrated resource constraints into a single-path search paradigm. Although these methods provide promising solutions, they typically focus on classification tasks designed for edge devices. Extending them to medical image segmentation, which often requires multiscale search spaces, can pose challenges in terms of flexibility and prolonged search time.

Among the hardware-aware approaches, two widely cited and influential methods stand out: ProxylessNAS¹⁸ and FBNet.¹⁹ They both possess the flexibility to explore multiscale search spaces, and their effectiveness has been widely validated in practical deployments on various devices. ProxylessNAS introduces hardware constraints, allowing for the simultaneous optimisation of network accuracy and hardware performance (e.g. latency and memory usage) during the search. It adopts a direct search strategy that targets specific hardware, ensuring the searched architectures perform well on the target device. During the search process, ProxylessNAS employs a differentiable optimisation framework, enabling efficient searching and avoiding the computational complexity associated with traditional NAS methods based on discrete sampling. FBNet adopts an end-to-end approach, simultaneously considering hardware and model performance requirements during training, significantly improving search efficiency. Specifically, it prebuilds a LUT recording the hardware metrics of various BOs, such as different types of convolution layers and activation functions, on specific hardware. This eliminates the need to repeatedly run the model on the hardware for each candidate architecture evaluation.

Although originally designed for image classification, ProxylessNAS and FBNet still offer a robust strategy that underpins our method, particularly through their integration of hardware constraints into the search process.

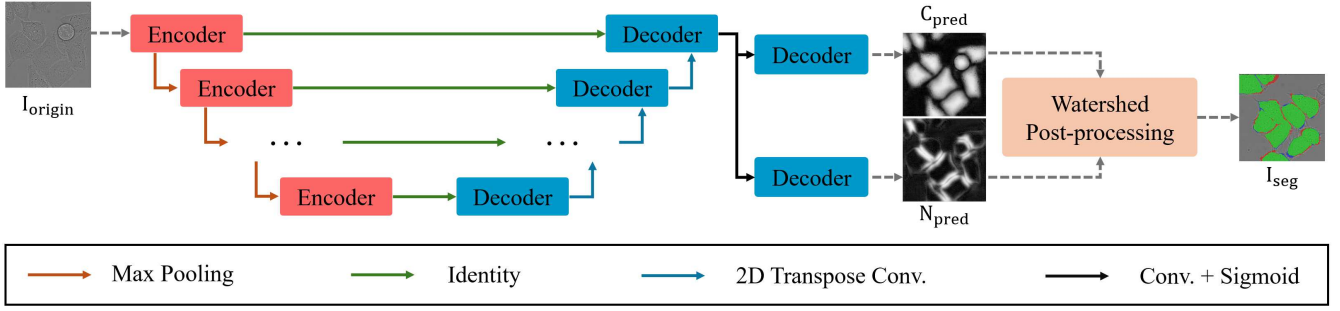


Figure 2. Architecture of our proposed method.

2.4 Maximum entropy principle

The maximum entropy principle is widely used in information theory.^{34,35} Many studies have explored the relationship between information entropy and neural network architectures. For instance, Saxe et al.³⁶ studied the information bottleneck phenomenon in deep networks, analysing entropy distribution and information flow within networks. Other researchers^{37–40} enhanced network performance by maximising the entropy of multiscale feature maps. In our method, we explore optimising neural network architectures by maximising information entropy to improve feature utilisation under various hardware constraints, while balancing the performance degradation caused by hardware-aware constraints.

3 Proposed method

Our proposed method is a multi-objective NAS framework consisting of two main components: the search space and the search strategy. The search space defines the set of candidate architectures that can be generated within a given range. The search strategy refers to the optimisation methods used to generate candidate architectures and select the optimal solution. In our framework, the search space considers both the topology-level and the operation-level configurations of segmentation architectures. For the topology level, we specify the network depth and the number of BOs by defining the building blocks in the encoder and decoder paths. At the operation level, we search for suitable BOs within each block. Our NAS method for live-cell segmentation constructs segmentation architectures by considering multiple objective functions: expected segmentation error, hardware-related metrics (inference speed and memory usage), and information entropy. We first introduce the segmentation model and search space, which includes the topology and operation levels, and then discuss the hardware-aware and entropy-driven search strategy.

3.1 Segmentation model

We adopt a U-Net structure to construct the topological layer for live-cell segmentation (Fig. 2). U-Net has not only proven effective in biomedical image segmentation, but has also undergone extensive validation in medical NAS.^{13–15,41} Our network comprises multiple encoder–decoder pathways operating at different scales, thus facilitating effective feature extraction and recovery. Specifically, the encoder pathway receives 2D cell images as input and applies a series of BO computations to learn hierarchical representations. The corresponding decoder pathway then reconstructs spatial details at each scale, ensuring the preservation of fine-grained boundary information. Within the encoder, max pooling is employed to progressively reduce the spatial resolution while enlarging the receptive field, thus enabling the capture of both local and global features across multiple scales. As the encoder pathway deepens, the number of feature channels increases, allowing more expressive representations of cellular structures.

Each decoder block mirrors its encoder counterpart by first employing transposed convolution to restore the spatial resolution, followed by fusing the upsampled features with the high-resolution features from the corresponding encoder block through skip connections. This synergy between deep (semantic) and shallow (spatially detailed) features is a key strength of U-Net-like architectures, as it preserves subtle details of cell boundaries. Subsequent convolution layers within the decoder further refine these fused features before passing them along to the following layers.

At the final stage of the network, two distinct prediction heads produce two distance maps: the cell distance C_{pred} , which denotes the distance from each pixel to its nearest cell boundary, and the neighbour distance N_{pred} , which represents the separation distance between adjacent cells. These distance maps are subsequently utilised by a watershed-based postprocessing module to delineate the actual cell boundaries. By integrating both distance predictions, the watershed step effectively discriminates tightly packed or partially overlapping cells. This approach mitigates common segmentation pitfalls such as under-segmentation (multiple cells merging into one) or over-segmentation (one cell splitting into multiple).⁴² Ultimately, the watershed output produces the final segmentation map I_{seg} , accurately outlining each individual cell in the image.

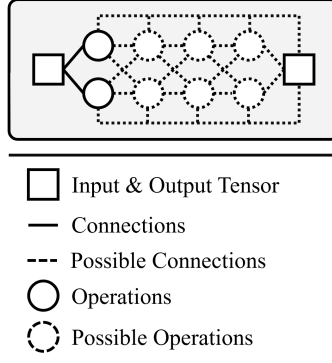


Figure 3. Architecture of the encoder and decoder.

3.2 Search space

Building on the encoder-decoder framework, we decouple upsampling and downsampling operations from the encoder and decoder modules, thereby substantially reducing the search space for the operational layer and making the network structure more amenable to optimisation. Specifically, the operational layer manages the internal set of BOs at each scale of the encoder-decoder paths. Each encoder/decoder unit (Fig. 3) is represented as a directed acyclic graph (DAG), where the input tensor (left) flows through n nodes, each corresponding to a BO, and each directed edge denotes the flow of tensors between operations.

The available BOs (Table 1) include conventional convolution (“Conv”), atrous convolution,⁴³ shuffle convolution,⁴⁴ depthwise-separable convolution,⁴⁵ and the squeeze-and-excitation attention module.⁴⁶ In addition, we also incorporate “Identity” and “None” operators. We focus on these operators not only because they are widely employed in medical image segmentation tasks, but also because their efficiency and extensive hardware support make them ideally suited to our target edge devices. Specifically, we designed and optimised the BO set according to the run-time constraints of the target hardware. For example, while a long short-term memory (LSTM) operator can effectively capture spatio-temporal patterns in time-lapse microscopy data,¹⁴ we opted to not incorporate it into our BO set because of its high computational cost and latency under our target hardware constraints. Evaluations on the PYNQ-Z2 and ZUBoard-1CG platforms indicated that LSTM operators incur excessive memory and runtime costs relative to alternative operations. By contrast, the selected operators benefit from native support and optimised acceleration in the ONNX runtime, ensuring real-time performance on resource-constrained edge devices. Consequently, we prioritised operators that deliver competitive segmentation accuracy whilst satisfying the stringent efficiency requirements of our deployment environment.

Compared to NAS-UNet¹³ and NAS-UNet-LSTM,¹⁴ we also optimised the search space for upsampling and downsampling operations. By analysing the frequency of the selected upsampling and downsampling operations on the target device, we observed an unbalanced usage of certain fundamental operators. Hence, to keep the search tractable under the deployment constraints of the target edge devices, we pruned the search space for resolution-changing operators after preliminary profiling and pilot searches. Specifically, max pooling and transpose convolution were retained as shared downsampling and upsampling operators, respectively, while the architecture search focused on the internal BO composition of each encoder/decoder unit. This way, we arrived at a set of BOs tailored to our target edge devices, which is more compact than those of NAS-UNet and NAS-UNet-LSTM, further reducing search cost.

Moreover, because we include a “None” operator in the BOs, certain nodes can be completely bypassed, resulting in more flexible routes for feature propagation and thereby accommodating greater structural flexibility. Ultimately, all node outputs are aggregated via an element-wise sum into the output tensor, a choice that not only alleviates GPU memory demands, but also aligns with existing NAS studies showing a preference for addition over concatenation.^{47,48} This design captures a broad range of architectural variations, from more elaborate graphs incorporating multiple operators in series to simpler paths that omit entire processing steps. By allowing the “None” operator as a candidate, our framework reduces the computational burden and enables the network to learn optimally sparse or dense connections, depending on the target deployment constraints.

3.3 Search strategy

Our objective is to search for a cell segmentation network that can be efficiently deployed on edge devices, optimising to minimise segmentation error, latency, and memory usage. To achieve this, we employ a hardware-aware search strategy that optimises network performance while taking into account the resource constraints of edge devices.

Table 1. Basic operators for operation levels.

Operator
Conv
Atrous conv
Shuffle conv
Depthwise-separable conv
Squeeze-and-excitation
Identity
None

In order to jointly minimise segmentation error, latency, and memory usage, we use a multi-objective function:

$$L_{\text{total}} = w_1 \cdot L_{\text{ce}} + w_2 \cdot \text{Latency} + w_3 \cdot \text{Memory} - w_4 \cdot H, \quad (1)$$

where L_{ce} represents the cross-entropy loss, which corresponds to the segmentation error, Latency is the inference latency, Memory is the network’s memory usage, H is defined as the average differential entropy of the feature maps over selected layers, and w_1, w_2, w_3, w_4 are the respective weighting coefficients for each objective.

We regard the determination of w_1, w_2, w_3 and w_4 as a multi-objective hyperparameter optimisation problem, aiming to balance segmentation accuracy, inference latency, and memory usage while ensuring efficient deployment on edge devices. Concretely, we define several discrete candidate values for each weight under predefined hardware constraints and then evaluate segmentation accuracy and hardware metrics on a validation set. Any combinations exceeding the hardware limits are discarded, and we rank the remaining feasible sets across these multiple criteria. We select the configuration that achieves the best trade-off between accuracy and resource cost, and finally validate its effectiveness and stability on the target device and test set to ensure reliable performance and resource efficiency in real-world deployment.

Measurement of the actual runtime of an architecture on the target device is highly time-consuming. For a network with a depth of 5, and 8 nodes per block, covering the entire search space would require measuring approximately 10^{34} architectures. To more efficiently account for latency and memory usage, we employ device-specific LUTs to estimate the overall latency and memory usage of the network based on the runtime and memory consumption of each operator. More formally, we assume the following:

$$\text{Latency}_{\text{arch}} = \log \sum_{\text{BO}} \text{Latency}(\text{BO}), \quad (2)$$

which implies that the runtime of each operator on the target processor is taken to be independent of other operators during the search. This LUT-based formulation is used as an efficient approximation rather than an exact simulation of end-to-end hardware behaviour. It is most suitable for platforms where operators are executed predominantly sequentially, although its accuracy may be reduced on systems with stronger parallelism or more complex memory interactions. By benchmarking the latency of a few hundred operators used in the search space, we can efficiently estimate the performance of architectures within the search space.

Throughout the search, we use a LUT to evaluate the hardware performance of each candidate architecture in real time, thereby avoiding network structures that exceed the device’s resource limits. The LUT covers the latency and memory usage of various operations, and the optimised network is adapted to the hardware characteristics of the device by the following constraints:

$$\text{Latency} \leq T_{\text{latency}}, \quad \text{Memory} \leq T_{\text{memory}}, \quad (3)$$

where T_{latency} and T_{memory} represent the maximum allowable values for latency and memory usage, respectively.

In practice, imposing a strict cutoff for architectures exceeding these thresholds can be detrimental for two reasons. First, if a promising design only slightly surpasses the threshold, a hard cutoff will discard it outright, thereby losing a potentially high-accuracy model that could have been refined or pruned back under the limit. In addition, in gradient-based or iterative optimisation, a hard cutoff provides no gradient feedback, preventing corrections that might otherwise bring the solution back into the feasible domain. Instead, we adopt a soft-constraint strategy by introducing a regularisation term that penalises any violation of T_{latency} and T_{memory} . The penalty grows quadratically with the extent of the violation, strongly discouraging infeasible architectures, while still allowing them to be adjusted in subsequent search steps. Consequently, our approach favours architectures that remain within resource constraints without prematurely discarding borderline designs.

The penalty for exceeding the latency threshold is:

$$\text{Latency}_{\text{reg}} = \max(0, \text{Latency} - T_{\text{latency}})^2. \quad (4)$$

Architectures that breach the threshold incur this penalty, which does not outright forbid them but dramatically lowers their likelihood of being selected unless they achieve substantial gains in accuracy or other metrics. By permitting slight threshold violations under a heavy penalty, we preserve gradient information and avoid prematurely discarding near-feasible solutions during the search.

After introducing the regularisation term, the actual latency of the network consists of two parts: the first is the inference latency $\text{Latency}_{\text{arch}}$ of the architecture itself on the target device, and the second is a penalty term $\text{Latency}_{\text{reg}}$ for exceeding the latency threshold. These two parts are combined to form the total latency expression, which is used to evaluate the final hardware performance of the network and ensure that the resulting model is both efficient and deployable on resource-constrained edge devices:

$$\text{Latency} = \text{Latency}_{\text{arch}} + \text{Latency}_{\text{reg}}. \quad (5)$$

3.4 Entropy-driven optimisation mechanism

To maintain high segmentation accuracy while adhering to hardware resource constraints, we introduce an entropy-driven optimisation mechanism. In the search space, we allow the model to select the “None” operation, which effectively skips certain nodes, reducing the complexity and computational load of the model. This helps to meet the latency and memory limitations of edge devices, but can lead to a decrease in the model’s feature representation capacity, which in turn could affect segmentation performance.

To address this trade-off, we incorporate information entropy as one of the optimisation objectives during the search process. In information theory, entropy measures the uncertainty and information content of a random variable. Here, the computed entropy H is integrated into the overall NAS objective as a reward term, counterbalancing the penalties from latency and memory constraints. Entropy is used to assess the diversity and richness of the network’s feature representations. By maximising the entropy of the feature maps, we encourage the model to extract more diverse and informative features, thereby improving segmentation accuracy while maintaining computational efficiency.

In practice, we calculate the differential entropy of the feature maps during the forward pass. Assuming the feature map activations follow a Gaussian distribution, the entropy $H^{(l)}$ of the feature map from layer l can be expressed as:

$$H^{(l)} = \frac{1}{2} \log \left(2\pi e \left(\sigma^{(l)} \right)^2 \right), \quad (6)$$

where e denotes Euler’s number, the base of the natural logarithm, and $\sigma^{(l)}$ is the standard deviation of the feature map activations:

$$\sigma^{(l)} = \sqrt{\frac{1}{N} \sum_{i=1}^N \left(x_i^{(l)} - \mu^{(l)} \right)^2}, \quad (7)$$

where $\mu^{(l)}$ is the mean of the feature map, N is the number of elements in the feature map, and $x_i^{(l)}$ is the value of the i th element. The total entropy H of the network is computed by averaging the entropy across selected layers:

$$H = \frac{1}{L} \sum_{l=1}^L H^{(l)}, \quad (8)$$

where L is the number of layers used for entropy calculation. The entropy is computed at each forward pass by estimating the variance of the feature maps. This entropy-driven mechanism acts as a practical regularisation term that encourages the search process to favour architectures that meet hardware constraints while preserving richer intermediate representations. Empirically, our ablation results suggest that this regularisation term is associated with improved convergence behaviour and competitive segmentation performance under resource constraints.

4 Experiments and results

The proposed method was extensively evaluated on a public dataset and various types of hardware. We describe the specifics of the dataset, model training, hardware platforms, and evaluation metrics, and the results of the experiments and an ablation study.

Table 2. Overview of the ten 2D subsets of the Cell Tracking Challenge dataset.

Subset	Modality	Cell type	Frame dimensions (pixels)	Number of frames	Contrast ratio	Signal-to-noise ratio
BF-C2DL-HSC	Bright-field	Mouse hematopoietic stem cells	$1,010 \times 1,010$	1,764	0.86	2.01
BF-C2DL-MuSC	Bright-field	Mouse muscle stem cells	$1,070 \times 1,036$	1,376	0.95	1.09
DIC-C2DH-HeLa	Differential interference contrast	Henrietta Lacks human uterine cervical carcinoma immortalised cells	512×512	84-115	1.00	0.45
Fluo-C2DL-MSC	Fluorescence	Rat mesenchymal stem cells	$992-1,200 \times 782-832$	48	1.56	3.18
Fluo-N2DH-GOWT1	Fluorescence	Mouse embryonic stem cells	$1,024 \times 1,024$	92	9.99	4.92
Fluo-N2DH-SIM+	Fluorescence	Second-generation simulated cells	$628-739 \times 690-790$	65-150	1.23	6.35
Fluo-C2DL-Huh7	Fluorescence	Human hepatocarcinoma-derived cells	$1,024 \times 1,024$	30	8.63	9.54
Fluo-N2DL-HeLa	Fluorescence	Henrietta Lacks human uterine cervical carcinoma immortalised cells	$1,100 \times 700$	92	1.02	22.68
PhC-C2DH-U373	Phase contrast	Human glioblastoma-astrocytoma cells	696×520	115	1.10	2.90
PhC-C2DL-PSC	Phase contrast	Pancreatic stem cells	720×576	300	1.44	3.12

4.1 Image dataset

The Cell Tracking Challenge (CTC)⁴⁹ dataset is specifically designed to benchmark cell segmentation and tracking algorithms under various imaging conditions. Encompassing a wide range of cell types and imaging modalities, the CTC provides a rigorous evaluation framework for segmentation algorithms. We used all ten 2D subsets from the CTC. These comprise time-lapse image sequences of motile and dividing cells recorded using fluorescence, bright-field, phase-contrast, or differential interference contrast microscopy, covering diverse biological conditions such as stem cells and cancer cells.

Each subset of the CTC dataset (Table 2) differs in terms of cell density, cell morphology, and imaging artefacts, thus forming a comprehensive benchmark to assess the generalisability of cell segmentation methods. For instance, BF-C2DL-HSC and BF-C2DL-MuSC contain sparsely distributed, low-contrast cells, while Fluo-N2DH-SIM+ simulates tightly packed cells and challenging illumination conditions. These varying conditions mimic realistic scenarios in microscopy, enabling an in-depth evaluation of the robustness of segmentation algorithms.

In addition, each subset includes a separate training set and a held-out test set. We further divided the training set into five folds, using four folds for training and the remaining fold as validation, and repeated this procedure so that each fold served as the validation set once. This five-fold cross-validation enabled robust hyperparameter tuning while mitigating overfitting. After determining the best hyperparameters, we retrained the model on the entire training set and we reported performance exclusively on the held-out test set. For every type of cell image, we adopt the same preprocessing, data augmentation, and postprocessing strategy as used by.⁴²

4.2 Model training

Based on the target hardware configuration and the memory constraints of the devices, we set the maximum depth of our network architecture to 5, with the maximum number of BOs in each stage capped at 8. The batch size was fixed to 4, and an early-stopping mechanism was employed in both the architecture search and the training phases. Specifically, training was terminated when the architecture remained unchanged for 15 consecutive epochs or when the maximum of 200 epochs was reached. We employed the Adam optimiser⁵⁰ with a learning rate of 0.0003 and a weight decay of 0.0001 to learn both architecture and kernel parameters. The architecture search and training processes were performed on four NVIDIA V100 GPUs.

The models were implemented in PyTorch⁵¹ and subsequently converted to the Open Neural Network Exchange (ONNX)^a format for deployment on various hardware platforms. The ONNX format facilitates seamless model portability across hardware platforms, while ONNX runtime optimisations were applied during deployment to ensure that the models met the real-time requirements imposed by the resource limitations of each device.

4.3 Hardware platforms

Trained models were then tested on five different hardware platforms: Jetson Nano,^b PYNQ-Z2,^c ZUBoard-1CG,^d Raspberry Pi 5,^e and Raspberry Pi 5 with a HAILO-8 accelerator.^f These platforms span a broad range of computational power, memory capacity, and energy efficiency, enabling us to assess the adaptability of our approach under diverse resource constraints. Specifically, the Jetson Nano combines a quad-core ARM Cortex-A57 CPU with a 128-CUDA-core Maxwell GPU and 4 GB of RAM, typically operating at 5-10 W. The PYNQ-Z2, built around a Xilinx Zynq-7000 SoC, includes a dual-core ARM Cortex-A9 CPU and 512 MB of external DDR, with average power consumption of approximately 2-5 W. The ZUBoard-1CG

^a<https://onnx.ai/>

^b<https://developer.nvidia.com/embedded/jetson-modules>

^c<https://www.xilinx.com/support/university/xup-boards/XUPPYNQ-Z2.html>

^d<https://www.avnet.com/wps/portal/us/products/avnet-boards/avnet-board-families/zuboard-1cg>

^e<https://www.raspberrypi.com/products/raspberry-pi-5/>

^f<https://hailo.ai/products/ai-accelerators/hailo-8-ai-accelerator/>

integrates a Xilinx Zynq UltraScale+ MPSoC with ARM Cortex-A53 and ARM Cortex-R5F cores, 1 GB of DDR, and power usage generally below 10 W. The Raspberry Pi 5 offers a quad-core ARM Cortex-A76 CPU and up to 8 GB of LPDDR4X SDRAM, drawing around 2–10 W depending on workload. Finally, augmenting the Pi 5 with the HAILO-8 accelerator delivers up to 26 TOPS at a power budget of roughly 2.5–10 W, providing a powerful yet energy-efficient platform for neural inference. Some earlier studies have further discussed the design considerations and performance trade-offs of these systems.¹²

4.4 Evaluation metrics

We used both segmentation and hardware efficiency metrics to evaluate the performance of the proposed method. To evaluate segmentation performance, we adopted two metrics officially used by the CTC,⁴⁹ namely SEG and DET, both of which range in $[0, 1]$ with higher values indicating better performance. The SEG metric quantifies the overlap between predicted and ground-truth masks by computing the average Jaccard index across all matched reference objects, where for each pair of predicted S and ground-truth object R the Jaccard index is calculated as $J(S, R) = |S \cap R| / |S \cup R|$. Any reference object that does not meet a minimum overlap threshold (at least 50% of R) is treated as unmatched. By contrast, DET measures how well individual cells are detected using a normalised acyclic-oriented graph matching measure, defined as $DET = 1 - \min(AOGM-D, AOGM-D_0) / AOGM-D_0$, where AOGM-D is the cost of transforming the computed set of detected objects into the reference set according to the acyclic oriented graph measure (AOGM), and AOGM-D₀ denotes the cost of creating the reference set from scratch.

Regarding hardware efficiency, we measured several key factors pertinent to model deployment on edge devices. Latency is defined as the average time required to process a single image during inference, directly reflecting the model's real-time capability on resource-constrained hardware. By monitoring memory usage, we captured the mean memory consumption during inference, which indicates how effectively the model manages limited memory resources. We also compared the energy consumption required for inference. To more precisely capture minor fluctuations in memory usage and energy consumption, we continuously measured the model at a sampling frequency of 1 Hz and averaged the measurements over a 10-minute interval. Finally, we evaluated the model size to determine the overall storage requirements—an important consideration when deploying models to devices with restricted storage capacities or performing remote updates.

4.5 Comparison methods

We compared our proposed method with several state-of-the-art methods, including established NAS-based models commonly employed in 2D medical image segmentation. Specifically, we considered nnU-Net,²⁶ which has shown strong performance in various challenges such as brain and liver tumour segmentation by automatically adapting its pipeline to different datasets. Furthermore, we included NAS-UNet,¹³ developed primarily for abdominal CT and MRI segmentation, and BiX-NAS,¹⁵ which aims at cross-domain generalisation for MRI and CT scans. We also added AdwU-Net,¹⁷ which refines U-Net through NAS techniques for ultrasound data, and Auto-DeepLab,³⁰ which applies automated architecture search to various large-scale 2D benchmarks (e.g. Cityscapes⁵²). Finally, we included NAS-UNet-LSTM,¹⁴ which specifically addresses live cell segmentation tasks but does not emphasise edge-focused resource constraints. Although these models achieved high performance in their respective domains, their suitability for resource-limited environments such as on-device cell analysis remains underexplored.

4.6 Evaluation results

Our proposed method performed favorably in SEG and DET segmentation metrics on most CTC subsets compared to competing NAS-based frameworks (Table 3). Moreover, our model has a smaller size, leading to faster inference speed and lower energy consumption. To further investigate the differences revealed by the quantitative scores, we performed a qualitative assessment of the segmentation outputs. Representative visual examples (Fig. 4) indicate that, even in challenging imaging scenarios characterised by heterogeneous illumination, blurred cell edges, and densely packed cell clusters, our model produced a more coherent and accurate delineation of cell boundaries. We note that all methods (including ours) used the same watershed postprocessing. The superiority of our model stems from its ability to learn more discriminative multiscale features and capture subtle intensity variations between adjacent cells. This enhanced representation ensures that, under the same postprocessing conditions, our final segmentation outputs remain more consistent and precise.

Compared to the leaderboard of the latest CTC segmentation benchmark,⁸ our method performs highly competitively in all data subsets (Table 4). The discrepancies arise primarily because current leaderboard methods typically prioritise segmentation accuracy without stringent hardware constraints and real-time requirements. In contrast, our proposed method achieves a balance between segmentation accuracy and practical hardware efficiency, with a search space specifically optimised for deployment scenarios, making it well-suited for real-world biomedical applications that require immediate analysis.

Because the LUT serves as a search-time latency proxy, we further examined its consistency with the measured end-to-end hardware latency. This comparison is intended to quantify how closely the LUT latency matches the real deployment

⁸<https://celltrackingchallenge.net/latest-csb-results/>

Subset	Model	SEG	DET	Jetson Nano (ms/W)	PYNQ-Z2 (ms/W)	ZUBoard (ms/W)	RPi5 (ms/W)	RPi5+HAILO-8 (ms/W)	Params (M)	Size (MB)
BF-C2DL-HSC	NAS-UNet	0.791	0.913	59/2.736	109/0.403	95/0.451	84/3.254	49/3.831	4.37	11
	NAS-UNet-LSTM	0.811	0.967	68/2.925	120/0.415	104/0.466	92/3.327	55/4.013	5.06	12
	nnU-Net	0.775	0.893	278/6.464	485/0.618	423/0.713	369/4.802	223/5.990	30.53	37
	BiX-NAS	0.817	0.970	59/2.715	105/0.402	94/0.443	81/3.257	47/3.806	4.22	11
	AdwU-Net	0.787	0.914	345/7.421	620/0.638	526/0.738	466/4.861	281/6.094	41.38	46
	Auto-DeepLab	0.823	0.975	228/5.731	408/0.602	357/0.682	314/4.636	191/5.732	24.42	30
	Ours	0.825	0.990	58/2.624	104/0.399	91/0.437	80/3.251	48/3.813	4.03	10
BF-C2DL-MuSC	NAS-UNet	0.740	0.903	70/2.928	128/0.420	111/0.474	98/3.371	57/4.008	5.38	12
	NAS-UNet-LSTM	0.760	0.954	80/3.157	142/0.435	126/0.492	108/3.459	65/4.218	6.27	13
	nnU-Net	0.725	0.881	283/6.538	494/0.620	431/0.715	376/4.817	228/6.008	31.25	39
	BiX-NAS	0.765	0.955	63/2.807	114/0.411	101/0.457	87/3.314	50/3.892	4.70	11
	AdwU-Net	0.737	0.900	377/7.787	676/0.644	573/0.744	508/4.903	306/6.137	46.20	48
	Auto-DeepLab	0.770	0.960	245/6.000	438/0.612	383/0.692	337/4.706	205/5.816	26.75	32
	Ours	0.772	0.977	62/2.706	112/0.408	98/0.455	86/3.256	52/3.926	4.60	11
DIC-C2DH-HeLa	NAS-UNet	0.851	0.953	62/2.782	114/0.407	99/0.457	88/3.283	51/3.875	4.61	11
	NAS-UNet-LSTM	0.861	0.959	67/2.899	118/0.413	102/0.463	91/3.313	54/3.990	4.93	11
	nnU-Net	0.824	0.943	272/6.381	475/0.615	414/0.710	362/4.784	219/5.970	29.74	37
	BiX-NAS	0.830	0.951	59/2.734	107/0.401	95/0.446	82/3.269	50/3.824	4.32	11
	AdwU-Net	0.842	0.947	335/7.294	601/0.636	510/0.736	452/4.845	272/6.076	39.83	42
	Auto-DeepLab	0.864	0.934	205/5.357	367/0.586	321/0.665	283/4.528	172/5.597	21.37	27
	Ours	0.869	0.961	56/2.592	101/0.397	88/0.440	78/3.183	46/3.818	3.98	10
Fluo-C2DL-MSC	NAS-UNet	0.675	0.821	56/2.678	104/0.397	90/0.444	80/3.218	46/3.775	4.07	11
	NAS-UNet-LSTM	0.681	0.827	81/3.172	143/0.436	124/0.494	110/3.468	66/4.231	6.35	13
	nnU-Net	0.647	0.724	283/6.534	493/0.620	430/0.715	376/4.816	227/6.007	31.21	39
	BiX-NAS	0.652	0.753	52/2.572	92/0.388	81/0.427	70/3.165	41/3.667	3.49	9
	AdwU-Net	0.654	0.790	356/7.552	639/0.641	542/0.740	480/4.877	289/6.111	43.05	48
	Auto-DeepLab	0.673	0.845	301/6.820	538/0.637	471/0.717	414/4.876	252/6.012	34.76	41
	Ours	0.683	0.852	50/2.489	90/0.383	79/0.423	70/3.171	42/3.702	3.42	9
Fluo-N2DH-GOWT1	NAS-UNet	0.903	0.943	62/2.773	115/0.411	100/0.458	89/3.290	52/3.886	4.67	11
	NAS-UNet-LSTM	0.925	0.960	67/2.900	118/0.415	102/0.463	91/3.313	54/3.991	4.93	11
	nnU-Net	0.875	0.896	275/6.412	479/0.616	417/0.711	364/4.791	221/5.978	30.03	39
	BiX-NAS	0.882	0.931	64/2.820	115/0.409	102/0.456	88/3.322	54/3.904	4.76	11
	AdwU-Net	0.872	0.896	358/7.567	642/0.641	544/0.741	482/4.879	291/6.112	43.25	48
	Auto-DeepLab	0.881	0.951	229/5.756	411/0.603	359/0.683	316/4.643	192/5.740	24.63	33
	Ours	0.930	0.963	63/2.754	113/0.408	98/0.453	86/3.261	52/3.854	4.64	11
Fluo-N2DH-SIM+	NAS-UNet	0.788	0.939	70/2.933	128/0.420	111/0.475	99/3.374	58/4.012	5.41	12
	NAS-UNet-LSTM	0.801	0.965	75/3.049	132/0.426	114/0.480	101/3.399	60/4.125	5.71	12
	nnU-Net	0.790	0.932	284/6.551	495/0.620	432/0.715	377/4.819	228/6.011	31.37	39
	BiX-NAS	0.794	0.928	58/2.711	105/0.397	93/0.443	80/3.254	49/3.802	4.20	11
	AdwU-Net	0.781	0.922	367/7.678	659/0.643	559/0.742	495/4.891	298/6.125	44.72	47
	Auto-DeepLab	0.796	0.933	216/5.529	386/0.594	337/0.673	297/4.580	181/5.662	22.75	28
	Ours	0.803	0.963	57/2.614	103/0.399	90/0.443	79/3.197	47/3.811	4.10	10
Fluo-C2DL-Huh7	NAS-UNet	0.741	0.915	62/2.780	114/0.407	99/0.456	87/3.281	51/3.872	4.61	11
	NAS-UNet-LSTM	0.762	0.939	66/2.893	118/0.413	102/0.462	90/3.309	54/3.984	4.90	11
	nnU-Net	0.730	0.853	272/6.371	474/0.615	413/0.710	361/4.782	218/5.968	29.64	37
	BiX-NAS	0.767	0.938	59/2.738	108/0.401	96/0.448	84/3.272	52/3.868	4.34	11
	AdwU-Net	0.735	0.886	389/7.919	698/0.646	592/0.745	524/4.915	316/6.150	48.09	53
	Auto-DeepLab	0.771	0.942	260/6.231	465/0.620	407/0.700	358/4.760	218/5.880	28.86	37
	Ours	0.779	0.953	59/2.642	106/0.404	92/0.446	82/3.215	49/3.823	4.25	10
Fluo-N2DL-HeLa	NAS-UNet	0.908	0.975	63/2.813	117/0.410	101/0.460	90/3.302	52/3.903	4.77	11
	NAS-UNet-LSTM	0.912	0.981	67/2.903	118/0.413	103/0.464	91/3.315	54/3.993	4.95	11
	nnU-Net	0.897	0.968	286/6.581	499/0.621	435/0.716	380/4.825	230/6.018	31.68	39
	BiX-NAS	0.915	0.977	60/2.753	109/0.402	97/0.448	83/3.281	51/3.842	4.42	11
	AdwU-Net	0.894	0.962	370/7.705	663/0.643	562/0.743	498/4.894	300/6.128	45.08	51
	Auto-DeepLab	0.918	0.980	161/4.591	289/0.547	253/0.621	222/4.259	135/5.246	15.73	22
	Ours	0.916	0.984	56/2.602	102/0.398	89/0.441	78/3.189	47/3.820	4.03	10
PhC-C2DH-U373	NAS-UNet	0.905	0.955	55/2.668	103/0.396	89/0.442	79/3.211	46/3.766	4.01	11
	NAS-UNet-LSTM	0.922	0.973	79/3.126	139/0.432	120/0.489	106/3.442	64/4.192	6.11	13
	nnU-Net	0.894	0.957	286/6.569	498/0.621	434/0.716	379/4.823	229/6.015	31.56	39
	BiX-NAS	0.916	0.961	56/2.670	100/0.393	88/0.440	77/3.229	47/3.763	3.99	10
	AdwU-Net	0.900	0.952	372/7.737	668/0.644	567/0.743	502/4.897	303/6.132	45.52	48
	Auto-DeepLab	0.915	0.958	252/6.108	451/0.616	394/0.696	347/4.731	211/5.847	27.73	33
	Ours	0.925	0.962	56/2.597	101/0.396	90/0.438	78/3.182	46/3.809	3.97	10
PhC-C2DL-PSC	NAS-UNet	0.721	0.853	67/2.876	123/0.415	107/0.468	94/3.339	55/3.961	5.10	12
	NAS-UNet-LSTM	0.728	0.960	86/3.260	151/0.444	131/0.503	116/3.516	69/4.305	6.82	13
	nnU-Net	0.718	0.942	278/6.465	485/0.618	423/0.713	369/4.802	223/5.991	30.55	38
	BiX-NAS	0.720	0.916	57/2.694	103/0.395	92/0.441	79/3.244	48/3.786	4.11	10
	AdwU-Net	0.714	0.837	342/7.382	614/0.638	521/0.737	461/4.856	278/6.089	40.90	43
	Auto-DeepLab	0.726	0.926	173/4.807	311/0.559	272/0.635	239/4.342	145/5.356	17.25	23
	Ours	0.731	0.963	54/2.602	102/0.398	89/0.443	78/3.189	47/3.820	4.03	10

Table 3. Quantitative comparison of segmentation (SEG) and detection (DET) performance for various state-of-the-art models across multiple microscopy live-cell subsets. Results are evaluated on diverse embedded platforms, covering both latency (in ms) and power efficiency (in W). Bold highlights the best value in each column of each subset. If multiple models or hardware configurations yielded the same best value, all are bolded.

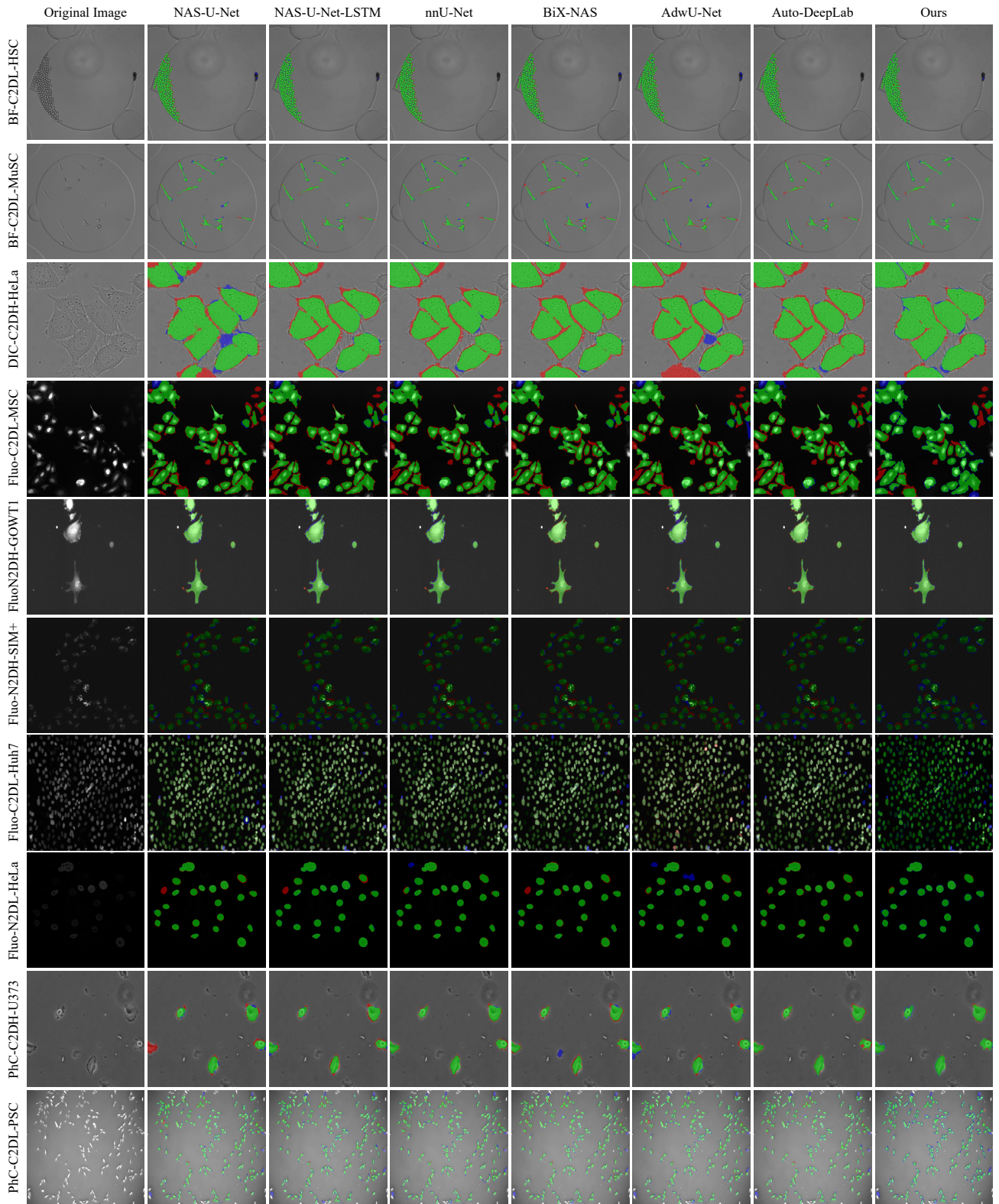


Figure 4. Qualitative segmentation results on ten 2D subsets of the Cell Tracking Challenge dataset. Each row corresponds to a different subset, and the columns display the original image along with the results from the compared methods. The coloured overlays indicate true positives (green), false positives (red), and false negatives (blue).

Table 4. Performance of our proposed method compared to the state of the art in the latest CTC segmentation benchmark at the time of writing. For the latter we report the range of values on the leaderboard per subset for the SEG and DET metrics.

Subset	SEG		DET	
	CTC	Ours	CTC	Ours
BF-C2DL-HSC	0.826–0.855	0.825	0.994–0.997	0.990
BF-C2DL-MuSC	0.774–0.784	0.772	0.982–0.984	0.977
DIC-C2DH-HeLa	0.871–0.877	0.869	0.960–0.975	0.961
Fluo-C2DL-MSD	0.655–0.687	0.683	0.832–0.876	0.852
Fluo-N2DH-GOWT1	0.933–0.938	0.930	0.970–0.980	0.963
Fluo-N2DH-SIM+	0.830–0.832	0.803	0.980–0.981	0.963
Fluo-C2DL-Huh7	0.782–0.811	0.779	0.956–0.968	0.953
Fluo-N2DL-HeLa	0.922–0.923	0.916	0.992–0.994	0.984
PhC-C2DH-U373	0.927–0.931	0.925	0.988–0.990	0.962
PhC-C2DL-PSC	0.741–0.756	0.731	0.973–0.975	0.963

Table 5. Validation of the LUT-based latency estimator on the target edge devices. T_{real} denotes the measured end-to-end latency, while $\hat{T}_{\text{LUT}}$ denotes the raw LUT latency estimate obtained by summing the latency of the selected BOs. MAPE and Max APE quantify the average and largest relative discrepancies between the LUT-estimated and measured latency, respectively. Spearman’s ρ is reported as a secondary indicator of ranking consistency among the evaluated architectures.

Device	Runtime/backend	No. arch.	Measured latency T_{real} (ms)	LUT latency $\hat{T}_{\text{LUT}}$ (ms)	MAPE (%)	Spearman ρ	Max APE (%)
Jetson Nano	ONNX Runtime	20	59.0 ± 7.9 (41–74)	64.3 ± 9.4 (44.1–80.0)	8.9	0.9766	16.0
PYNQ-Z2	ONNX Runtime	20	106.4 ± 14.6 (78–137)	118.0 ± 16.2 (90.9–150.4)	10.9	0.9751	19.0
ZUBoard-1CG	ONNX Runtime	20	93.5 ± 12.2 (70–119)	102.6 ± 14.1 (79.2–126.5)	9.6	0.9797	14.9
Raspberry Pi 5	ONNX Runtime	20	82.1 ± 11.6 (57–106)	86.8 ± 12.8 (57.4–110.1)	5.7	0.9574	10.5
Raspberry Pi 5 + HAILO-8	HAILO runtime	20	48.6 ± 6.5 (35–60)	57.4 ± 7.9 (42.8–75.3)	18.2	0.9434	28.3

latency. Specifically, we computed the mean absolute percentage error (MAPE) between the measured latency and the raw LUT-estimated latency, obtained by summing the BO latencies before applying the logarithmic scaling used in the search objective:

$$\text{MAPE}^{(d)} = \frac{100}{N} \sum_{i=1}^N \left| \frac{T_{\text{real}}^{(d)}(a_i) - \hat{T}_{\text{LUT}}^{(d)}(a_i)}{T_{\text{real}}^{(d)}(a_i)} \right|, \quad (9)$$

where $T_{\text{real}}^{(d)}(a_i)$ is the measured end-to-end latency of architecture a_i on device d , $\hat{T}_{\text{LUT}}^{(d)}(a_i)$ is the raw LUT latency estimate, and N is the number of evaluated architectures. Spearman’s rank correlation ρ is also reported as a secondary indicator of ranking consistency, measuring whether the latency ordering predicted by the LUT agrees with the ordering observed from real hardware measurements. The resulting consistency check (Table 5) shows that the LUT-estimated latency is consistent with the measured hardware latency across the target devices. This supports the use of the BO-level LUT as an efficient search-time proxy, while minor discrepancies are expected due to runtime-level deployment effects.

4.7 Ablation study

To validate the effectiveness of each component in our framework, we performed two ablation experiments, first on hardware-aware penalties and the second on entropy-driven optimisation.

4.7.1 Ablation on hardware-aware penalties

We first examined the impact of imposing hardware-aware penalties during the search process. Introducing these penalties markedly affected both the choice of BOs and the overall size of the model (Fig. 5). Specifically, comparing the operator distributions before and after applying these penalties, we noticed that the frequency of the operator “None” increased significantly, indicating that the search mechanism adaptively balanced complexity and efficiency when penalising for hardware-related costs. Comparing model sizes across different subsets, we found that hardware-aware constraints effectively prune superfluous parameters, making the model more amenable to resource-limited scenarios. In summary, these findings confirm that hardware-aware penalties guide the search towards architectures with lower computational overhead.

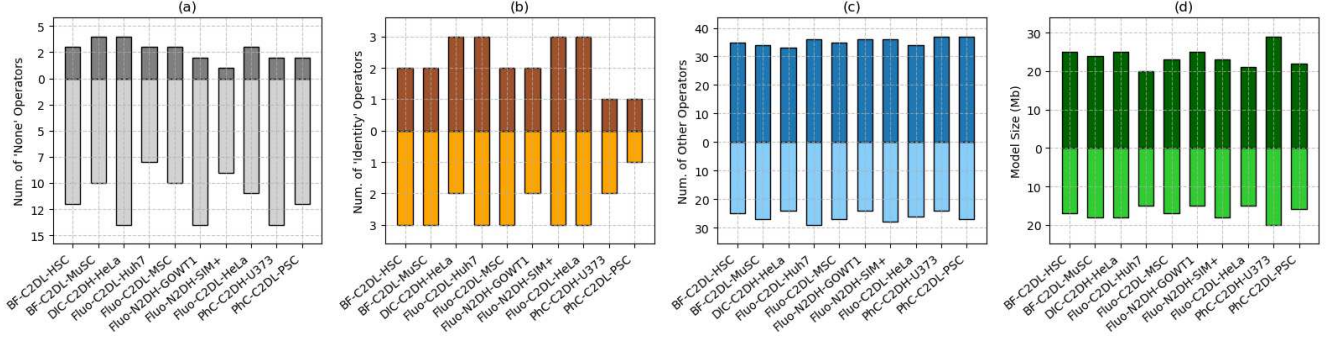


Figure 5. Operator usage and model size across various datasets in two scenarios: (top) without hardware-aware penalty and (bottom) with hardware-aware penalty. (a) Number of “None” operators, (b) number of “Identity” operators, (c) number of other (non-None, non-Identity) operators, and (d) final model size (MB).

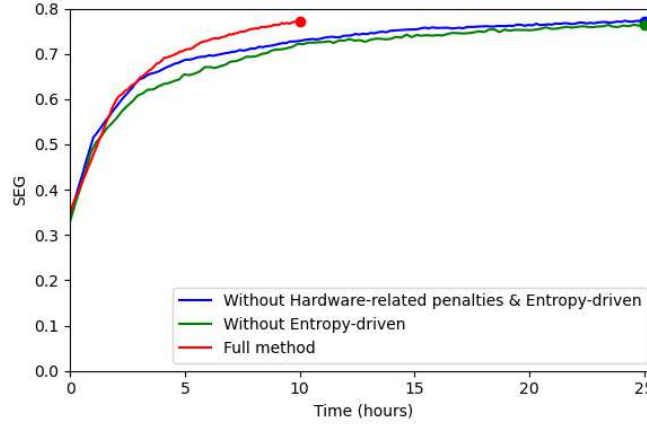


Figure 6. Ablation study of our method on the BF-C2DL-HSC subset, showing the search performance (SEG) over time.

Table 6. Comparison of model size (in MB) and performance (SEG, DET) under different ablation settings on the BF-C2DL-HSC subset.

Model	Model size (MB)	SEG	DET
Ours (w/o hardware-related penalties & w/o entropy-driven)	25	0.820	0.983
Ours (w/o entropy-driven)	16	0.811	0.974
Ours (full model)	17	0.825	0.990

4.7.2 Ablation on entropy-driven optimisation

We also investigated how the entropy-driven term affects convergence and the final performance. From the loss curves during the search phase on the BF-C2DL-HSC subset, comparing scenarios with and without entropy-driven optimisation (Fig. 6), we observed that incorporating the entropy term accelerates convergence in the initial stages, significantly reducing the time needed to reach the same loss level, while also yielding more stable results in later iterations. We attribute these benefits to the enhanced feature diversity introduced by the entropy term, which prevents the network from collapsing into overly sparse configurations and encourages more discriminative representations.

To quantify these improvements, we also report the performance of the final architectures under different configurations on the BF-C2DL-HSC subset (Table 6). Compared with the baseline, our entropy-driven method achieves better SEG performance. Crucially, this is accomplished without inflating model size or increasing latency, as the hardware-aware component continues to limit redundant complexity. This demonstrates that entropy-driven optimisation effectively offsets any potential reduction in representational capacity arising from aggressive hardware constraints.

5 Conclusion

In this study, we introduced a hardware-aware NAS framework for cell segmentation on resource-constrained hardware platforms. By integrating hardware latency and memory constraints into a U-Net-based search space, our method efficiently identifies architectures that balance segmentation accuracy with real-time performance. An entropy-driven component further ensures that the learned networks preserve sufficient feature diversity, preventing detrimental performance trade-offs when enforcing hardware constraints. Experiments on diverse live-cell microscopy images and deployments on multiple hardware platforms demonstrated a favourable trade-off between segmentation accuracy and practical hardware efficiency, supporting the reliability of the proposed framework for real-time cell image analysis on resource-constrained platforms.

Acknowledgements

The authors are grateful to Matthew Sheedy, George Bou-Rizk and Geoff Sizer from Genesys Electronics Design, Sydney, Australia, for fruitful discussions and providing the hardware platforms used in the experiments.

References

1. Meijering, E. Cell segmentation: 50 years down the road. *IEEE Signal Process. Mag.* **29**, 140–145 (2012).
2. Xing, F. & Yang, L. Robust nucleus/cell detection and segmentation in digital pathology and microscopy images: a comprehensive review. *IEEE Rev. Biomed. Eng.* **9**, 234–263 (2016).
3. Liu, Z. *et al.* A survey on applications of deep learning in microscopy image analysis. *Comput. Biol. Medicine* **134**, 104523 (2021).
4. Zeng, X. & Chen, Y. Analysis of cytotoxic effect using the live cell imaging system. *Int. Core J. Eng.* **8**, 340–344 (2022).
5. Yan, H. *et al.* Establishment of the microscope incubation system and its application in evaluating tumor treatment effects through real-time live cellular imaging. *Front. Bioeng. Biotechnol.* **12**, 1447265 (2024).
6. Talebipour, A., Saviz, M., Vafaiee, M. & Faraji-Dana, R. Facilitating long-term cell examinations and time-lapse recordings in cell biology research with CO2 mini-incubators. *Sci. Rep.* **14**, 3418 (2024).
7. Centofanti, C., Lozzi, D., Cococetta, C. & Marotta, A. An efficient edge-based system for nucleated oval-shaped red blood cell counting. *IEEE Access* **12**, 195629–195641 (2024).
8. Carpenter, A. E. *et al.* Cellprofiler: image analysis software for identifying and quantifying cell phenotypes. *Genome Biol.* **7**, 1–11 (2006).
9. Hu, T., Xu, S., Wei, L., Zhang, X. & Wang, X. CellTracker: an automated toolbox for single-cell segmentation and tracking of time-lapse microscopy images. *Bioinformatics* **37**, 285–287 (2021).
10. Ogundokun, R. O., Misra, S., Akinrotimi, A. O. & Ogul, H. MobileNet-SVM: a lightweight deep transfer learning model to diagnose BCH scans for IoMT-based imaging sensors. *Sensors* **23**, 656 (2023).
11. Datta Gupta, K. *et al.* A novel lightweight deep learning-based histopathological image classification model for IoMT. *Neural Process. Lett.* **55**, 205–228 (2023).
12. Xu, Y., Khan, T. M., Song, Y. & Meijering, E. Edge deep learning in computer vision and medical diagnostics: a comprehensive survey. *Artif. Intell. Rev.* **58**, 1–78 (2025).
13. Weng, Y., Zhou, T., Li, Y. & Qiu, X. NAS-UNet: neural architecture search for medical image segmentation. *IEEE Access* **7**, 44247–44257 (2019).
14. Zhu, Y. & Meijering, E. Automatic improvement of deep learning-based cell segmentation in time-lapse microscopy by neural architecture search. *Bioinformatics* **37**, 4844–4850 (2021).
15. Wang, X. *et al.* BiX-NAS: searching efficient bi-directional architecture for medical image segmentation. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 229–238 (2021).
16. Yu, Q. *et al.* C2FNAS: coarse-to-fine neural architecture search for 3D medical image segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4126–4135 (2020).
17. Huang, Z., Wang, Z., Yang, Z. & Gu, L. AdwU-Net: adaptive depth and width U-Net for medical image segmentation by differentiable neural architecture search. In *International Conference on Medical Imaging with Deep Learning (MIDL)*, 576–589 (2022).
18. Cai, H., Zhu, L. & Han, S. ProxylessNAS: Direct neural architecture search on target task and hardware. *arXiv:1812.00332* (2018).

19. Wu, B. *et al.* FBNet: hardware-aware efficient ConvNet design via differentiable neural architecture search. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10734–10742 (2019).
20. Benmeziane, H. *et al.* A comprehensive survey on hardware-aware neural architecture search. *arXiv:2101.09336* (2021).
21. Sadanandan, S. K., Ranefall, P., Le Guyader, S. & Wählby, C. Automated training of deep convolutional neural networks for cell segmentation. *Sci. Rep.* **7**, 7860 (2017).
22. Din, N. U. & Yu, J. Training a deep learning model for single-cell segmentation without manual annotation. *Sci. Rep.* **11**, 23995 (2021).
23. Moshkov, N., Mathe, B., Kertesz-Farkas, A., Hollandi, R. & Horvath, P. Test-time augmentation for deep learning-based cell segmentation on microscopy images. *Sci. Rep.* **10**, 5068 (2020).
24. Annasamudram, N., Zhao, J., Oluwadare, O., Prashanth, A. & Makrogiannis, S. Scale selection and machine learning based cell segmentation and tracking in time lapse microscopy. *Sci. Rep.* **15**, 11717 (2025).
25. Benmeziane, H., Hamzaoui, I., Cherif, Z. & El Maghraoui, K. Medical neural architecture search: survey and taxonomy. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1 (2024).
26. Isensee, F., Jaeger, P. F., Kohl, S. A., Petersen, J. & Maier-Hein, K. H. nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. *Nat. Meth.* **18**, 203–211 (2021).
27. Xiao, A. *et al.* Intraoperative glioma grading using neural architecture search and multi-modal imaging. *IEEE Trans. Med. Imaging* **41**, 2570–2581 (2022).
28. Peng, Y., Bi, L., Fulham, M., Feng, D. & Kim, J. Multi-modality information fusion for radiomics-based neural architecture search. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 763–771 (2020).
29. Liu, H., Simonyan, K. & Yang, Y. DARTS: differentiable architecture search. *arXiv:1806.09055* (2018).
30. Liu, C. *et al.* Auto-DeepLab: Hierarchical neural architecture search for semantic image segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 82–92 (2019).
31. Azghadi, M. R. *et al.* Hardware implementation of deep network accelerators towards healthcare and biomedical applications. *IEEE Trans. Biomed. Circuits Syst.* **14**, 1138–1159 (2020).
32. Tan, M. *et al.* MNASNet: platform-aware neural architecture search for mobile. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2820–2828 (2019).
33. Guo, Z. *et al.* Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision (ECCV)*, 544–560 (2020).
34. Jaynes, E. T. Information theory and statistical mechanics. *Phys. Rev.* **106**, 620 (1957).
35. Kullback, S. *Information theory and statistics* (Courier Corporation, 1997).
36. Saxe, A. M. *et al.* On the information bottleneck theory of deep learning. *J. Stat. Mech. Theory Exp.* **2019**, 124020 (2019).
37. Lin, M. *et al.* Zen-NAS: a zero-shot NAS for high-performance deep image recognition. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 347–356 (2021).
38. Sun, Z. *et al.* MAE-DET: revisiting maximum entropy principle in zero-shot NAS for efficient object detection. *arXiv:2111.13336* (2021).
39. Sun, Z. *et al.* Entropy-driven mixed-precision quantization for deep network design. In *Entropy-Driven Mixed-Precision Quantization for Deep Network Design* (2022).
40. Wang, J. *et al.* Maximizing spatio-temporal entropy of deep 3D CNNs for efficient video recognition. In *International Conference on Learning Representations (ICLR)* (2023).
41. Yan, X., Jiang, W., Shi, Y. & Zhuo, C. MS-NAS: multi-scale neural architecture search for medical image segmentation. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 388–397 (2020).
42. Scherr, T., Löffler, K., Böhlend, M. & Mikut, R. Cell segmentation and tracking using CNN-based distance predictions and a graph-based matching strategy. *PLoS One* **15**, e0243219 (2020).
43. Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K. & Yuille, A. L. DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Trans. Pattern Anal. Mach. Intell.* **40**, 834–848 (2017).
44. Zhang, X., Zhou, X., Lin, M. & Sun, J. ShuffleNet: an extremely efficient convolutional neural network for mobile devices. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6848–6856 (2018).

45. Chollet, F. Xception: Deep learning with depthwise separable convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1251–1258 (2017).
46. Hu, J., Shen, L. & Sun, G. Squeeze-and-excitation networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7132–7141 (2018).
47. Liu, C. *et al.* Progressive neural architecture search. In *European Conference on Computer vision (ECCV)*, 19–34 (2018).
48. Zoph, B., Vasudevan, V., Shlens, J. & Le, Q. V. Learning transferable architectures for scalable image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 8697–8710 (2018).
49. Maška, M. *et al.* The Cell Tracking Challenge: 10 years of objective benchmarking. *Nat. Meth.* **20**, 1010–1020 (2023).
50. Kingma, D. P. & Ba, J. Adam: a method for stochastic optimization. In *International Conference for Learning Representations (ICLR)* (2015).
51. Paszke, A. *et al.* PyTorch: an imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* **32** (2019).
52. Cordts, M. *et al.* The Cityscapes dataset for semantic urban scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3213–3223 (2016).

Author contributions

Y.X.: Conceptualization, methodology, software, experimentation, and writing—original draft. T.M.K.: Conceptualization and writing—review and editing. Y.Z.: Software and manuscript review. Y.S.: Conceptualization, methodology, supervision, and writing—review and editing. E.M.: Conceptualization, methodology, supervision, funding acquisition, resources, and writing—review and editing.

Data availability

The ten 2D Cell Tracking Challenge datasets used in this study are publicly available from the Cell Tracking Challenge repository (<https://celltrackingchallenge.net/2d-datasets/>). The data supporting the findings of this study are available from the corresponding author upon reasonable request.

Funding

This work was supported by the ARC Research Hub for Connected Sensors for Health (grant number IH210100040). Partial financial support was also provided by Genesys Electronics Design.

Competing interests

This study received research funding and hardware support from Genesys Electronics Design. The authors declare no other competing interests.