# Neural Inference Search for Multiloss Segmentation Models

Sam Slade, Li Zhang, *Senior Member, IEEE*, Haoqian Huang, *Member, IEEE*, Houshyar Asadi, *Member, IEEE*, Chee Peng Lim, Yonghong Yu, Dezong Zhao, *Senior Member, IEEE*, Hanhe Lin, and Rong Gao

***Abstract*— Semantic segmentation is vital for many emerging surveillance applications, but current models cannot be relied upon to meet the required tolerance, particularly in complex tasks that involve multiple classes and varied environments. To improve performance, we propose a novel algorithm, neural inference search (NIS), for hyperparameter optimization pertaining to established deep learning segmentation models in conjunction with a new multiloss function. It incorporates three novel search behaviors, i.e., Maximized Standard Deviation Velocity Prediction, Local Best Velocity Prediction, and $n$-dimensional Whirlpool Search. The first two behaviors are exploratory, leveraging long short-term memory (LSTM)-convolutional neural network (CNN)-based velocity predictions, while the third employs $n$-dimensional matrix rotation for local exploitation. A scheduling mechanism is also introduced in NIS to manage the contributions of these three novel search behaviors in stages. NIS optimizes learning and multiloss parameters simultaneously. Compared with state-of-the-art segmentation methods and those optimized with other well-known search algorithms, NIS-optimized models show significant improvements across multiple performance metrics on five segmentation datasets. NIS also reliably yields better solutions as compared with a variety of search methods for solving numerical benchmark functions.**

***Index Terms*— Convolutional neural network (CNN), hyperparameter optimization, multiloss function, semantic segmentation.**

This work was supported in part by the European Regional Development Fund (ERDF); in part by RPPTV Ltd., through the Joint Funding of a Ph.D. Studentship via the Intensive Industrial Innovation Program North East (IIIPNE) under Grant 25R17P01847; and in part by Innovate U.K. Smart Grants. *(Corresponding author: Li Zhang.)*

Sam Slade is with the Department of Computer and Information Sciences, Northumbria University, NE1 8ST Newcastle upon Tyne, U.K. (e-mail: samuel2.slade@northumbria.ac.uk).

Li Zhang is with the Department of Computer Science, Royal Holloway, University of London, TW20 0EX Surrey, U.K. (e-mail: li.zhang@rhul.ac.uk).

Haoqian Huang is with the College of Energy and Electrical Engineering, Hohai University, Nanjing 210098, China (e-mail: hqhuang@hhu.edu.cn).

Houshyar Asadi and Chee Peng Lim are with the Institute for Intelligent Systems Research and Innovation, Deakin University, Waurn Ponds, VIC 3216, Australia (e-mail: houshyar.asadi@deakin.edu.au; chee.lim@deakin.edu.au).

Yonghong Yu is with the College of Tongda, Nanjing University of Posts and Telecommunications, Nanjing 210049, China (e-mail: yuyh@njupt.edu.cn).

Dezong Zhao is with the James Watt School of Engineering, University of Glasgow, G12 8QQ Glasgow, U.K. (e-mail: Dezong.Zhao@glasgow.ac.uk).

Hanhe Lin is with the School of Science and Engineering, University of Dundee, DD1 4HN Dundee, U.K. (e-mail: hlin001@dundee.ac.uk).

Rong Gao is with the School of Computer Science, Hubei University of Technology, Wuhan 430068, China (e-mail: gaorong@hbut.edu.cn).

## I. Introduction

SEGMENTATION methods form a key component in many vision-related tasks, e.g., automated medical diagnosis, autonomous driving, and robotic navigation, all of which stand to revolutionize many industrial sectors. Poor segmentation performance causes incorrect medical diagnosis and dangerous course trajectories leaving apprehension in the uptake of these innovations. Consistently accurate segmentation algorithms are required to meet the stringent safety standards of these systems. Unfortunately, no existing methods can satisfy this requirement.

The existing segmentation techniques range from traditional methods such as $k$-means clustering [1] to modern deep learning methods [2], [3]. convolutional neural networks (CNNs) appear to be very successful in tackling segmentation problems with multiple semantic classes and complex shapes. Critically, this success depends upon selecting an appropriate loss function that sensibly measures the error and choosing appropriate hyperparameters for the gradient optimization algorithm and loss function. If these factors are not met, then CNN training is unlikely to produce well-generalized models. Acknowledging this, we seek to enhance the accuracy of CNN and transformer architectures through automated hyperparameter selection and appropriate loss function construction.

This research proposes a new search algorithm namely neural inference search (NIS) for fine-tuning hyperparameters of CNN and transformer-based segmentation models. Our research motivations are as follows. Swarm intelligence algorithms such as particle swarm optimization (PSO) employ fixed search parameters and do not adaptively adjust local and global search behaviors according to different search stages, therefore it constrains model capabilities in balancing between local exploitation and global exploration. By adopting global best solutions as the guiding signals, most swarm intelligence algorithms tend to converge prematurely. To tackle such limitations, NIS uses a neural network-based velocity updating strategy to dynamically predict optimal directions and magnitudes of velocities for updates, allowing for a better balance between diversification and exploitation. Such effects are further strengthened using an adaptive scheduler function. The neural network-based velocity prediction process also employs diverse local and global optimal signals for new velocity generation to overcome local optima traps. NIS introduces three new search strategies, namely: 1) maximized

standard deviation velocity prediction; 2) local best velocity prediction; and 3) $n$-dimensional Whirlpool Search, along with a Staged Discrete Adaptive Wave Function to schedule these behaviors. The first search action maximizes the standard deviation of the swarm to increase search territory and avoid repeatedly searching the same regions. The second operation moves particles to local optimal regions for faster convergence while diversifying the search process. The third strategy fine-tunes optimal regions around the global best solution using angle-driven whirlpool-style granular movements. The Staged Discrete Adaptive Wave Function dynamically adapts the contribution of each search behavior based on different search stages, emphasizing diversification and intensification in the early and final search stages, respectively.

Our proposed solution enhances CNN-based models' segmentation performance by integrating several error measures into a loss function. This approach increases the availability of various information and facilitates the selection of optimal hyperparameters for training. Additionally, our solution, NIS, addresses attention and expert knowledge issues that arise in manual searches while improving convergence rates and avoiding local optima traps. The prescribed search behaviors and staged scheduling method enable these benefits. The research's key contributions are summarized as follows.

1) A new multiloss function is proposed to capture multiple error measurements with respect to semantic segmentation, enhancing feedback during CNN training. Specifically, the combination of the mean of Cross Entropy, Focal, and Dice losses is exploited. Cross Entropy loss provides a measurement of the overall pixel-wise class accuracy. Focal loss introduces the term $\alpha$ to apply a weighting factor to the classes that are present or nonpresent in the ground truth (GT) masks, in order to prevent over-fitting. Additionally, it adds a $\gamma$ term to weight the contribution of well-classified examples, contrasting the error contribution of poorly classified ones. The Dice loss calculates a soft version of the mean intersection over union (mIoU) measurement, which can provide more insight into the discrepancy in shape and consistency of the classification. By combining these loss functions, multiple aspects of the error signal can be leveraged to design more effective training strategies.
2) To automate hyperparameter tuning, NIS is proposed. In particular, NIS incorporates three novel behaviors: 1) *maximized standard deviation velocity prediction*, which employs a long short-term memory (LSTM)-CNN to predict the velocity vectors that increase standard deviation of the particle positions, ensuring agents do not search in the same local areas; 2) *local best velocity prediction*, which predicts the velocity vectors that point to local areas of best fitness; and 3) *n-dimensional whirlpool search*, which produces the velocity vectors via a dot product of an $n$-dimensional rotation matrix at a defined angle with a vector pointing toward the global best position. As such, the particles are forced to explore multiple dimensions of the search space. The contribution of these three behaviors are adjusted at every iteration with a novel scheduling function, namely the Staged Discrete Adaptive Wave formula ensuring a better trade-off between exploration and exploitation. This is achieved by leveraging discrete stages factored with a slowly decreasing or increasing sinusoidal function, allowing each behavior to be disabled or emphasized whilst approaching global optimality. The scheduling function, in conjunction with these three innovative behaviors, operates synchronously to automate hyperparameter selection and address the issue of stagnation. To the best of our knowledge, we are the very first few works that use LSTM-CNN to predict optimal velocities to guide the search process.

The remaining of this article is structured as follows. Section II presents state-of-the-art related studies on image segmentation and optimization techniques. In Section III, the details of NIS are explained, including diverse proposed strategies and multiple loss functions. Following this, Section IV presents the evaluation of the proposed NIS algorithm for hyperparameter fine-tuning in semantic segmentation as well as solving benchmark functions. Finally, Section V presents the conclusions and suggestions for future work.

## II. Related Work

State-of-the-art related studies on image segmentation and PSO variants are discussed in this section.

### A. Segmentation

Many research studies on semantic segmentation methods are available in the literature, e.g., [4], [5] [6], for autonomous driving and robotic navigation. Zhang et al. [7] investigated the effect of early and late fusion of multimodal deep learning architectures to solve semantic segmentation for automated robotic navigation. They proposed a Complex Modality network (CMnet) which utilized a late fusion of two processing streams to handle both RGB images and supplementary features such as near-infrared images. Performance gains were obtained by using such dual stream architectures for diverse image segmentation tasks. Saire and Rivera [8] explored multitask learning with deep learning for semantic segmentation by introducing three related auxiliary tasks to be solved simultaneously by a single CNN model. A standard encoder-decoder network was adopted with predictive branches, one for the main segmentation task while the others for distinct contour prediction tasks. Each branch used a combination of cross entropy and soft IoU loss, which were weighted to control the contribution of the error signals.

Islam et al. [9] proposed a gated feedback refinement network (G-FRNet) for dense image labeling. Processing branches were inserted between spatially distinct encoding and decoding layers. Each branch contained a Gate block linked to a refinement block, providing spatially relevant features for the decoder stages. Each stage was supervised by spatially matching GT images. Jègou et al. [10] developed FC-DenseNet by appending DenseNet with upsampling blocks, while SegNet was proposed by Badrinarayanan et al. [11] which used the pooling indices of the maxpooling steps from a VGG16 encoder to perform nonlinear upsampling, for semantic segmentation.

Li et al. [12] introduced CTNet, a context-based tandem network for semantic segmentation. It made use of context information in both the channel and spatial dimensions of images through the use of the channel contextual model (CCM) and the spatial contextual model (SCM). The two models were connected for interactive training to exchange context information, where CCM acted as a prior knowledge for SCM. The SCM also introduced a self-attention mechanism for improved efficiency. CTNet outperformed the current state-of-the-art models on diverse segmentation tasks. Cheng et al. [13] proposed a revolutionary deep learning-based image segmentation model, i.e., masked-attention mask transformer (Mask2Former), which utilized a transformer decoder with masked attention. Their architecture was capable of addressing image segmentation tasks, such as panoptic, instance, and semantic segmentation, effectively. The masked attention component of the model restricted cross-attention within predicted mask regions, resulting in improved localized feature extraction. The model also included a multiscale strategy and optimization improvements, leading to superior performance compared to other state-of-the-art architectures. Strudel et al. [14] proposed a fully transformer-based encoder-decoder architecture, namely Segmenter, for semantic segmentation in images. The algorithm split the input image into patches, which were transformed into patch embeddings by a transformer encoder. These embeddings were then decoded by either a linear decoder or a mask transformer to produce pixel-level class annotations. The model was trained end-to-end and outputted a single class per pixel at inference time by applying the argmax to the upsampled output. The mask transformer decoder generated $K$ masks by computing the scalar product between the $L2$-normalized patch embeddings and class embeddings from the decoder. Qualitative results showed that Segmenter provided more consistent labels on large object instances and handled partial occlusions better compared to DeepLabv3+.

Sun and Li [15] introduced a new method called semantic structure aware inference (SSA) for object localization and multilabel tasks. It aimed to expand class activation maps (CAMs) to capture semantic structure information in images by incorporating the semantic structure modeling (SSM) module, which consisted of two self-affinity (SA) blocks and a smooth gate. The SSM module used feature maps from different stages of a CNN to expand the seed CAM, which was obtained by weighting the last convolution layer's feature map with the last classification layer's weights. The final CAM was created by combining the expanded CAMs from various stages of the network. SSA outperformed baseline state-of-the-art methods for diverse object localization tasks. Sun et al. [16] introduced a novel approach to few-shot segmentation (FSS) named singular value fine-tuning (SVF), which addressed the overfitting issue in FSS. In FSS, the task is to segment novel class objects with only a few densely annotated samples. Existing methods froze the pretrained backbone to prevent overfitting, but this leads to suboptimal performances. SVF fine-tuned a small part of the backbone parameters, instead of freezing the entire pretrained model, by decomposing the backbone parameters via singular value decomposition (SVD). The results showed the superiority of SVF over traditional fine-tuning methods in FSS. Du et al. [17] developed a SwinPA-Net with Swin Transformer as the backbone for medical image segmentation. Multiplicative feature fusion and multiscale attention aggregation were adopted to increase feature learning capabilities of their network. It employed a Swin Transformer as the encoder to extract multiscale feature maps, which were subsequently concatenated via a dense multiplicative connection (DMC) component. A local pyramid attention (LPA) module was exploited to extract discriminative spatial features from the multiplicative fused feature representations of DMC. A CNN was adopted as the decoder to upsample feature maps to generate the mask output.

Zhou et al. [18] developed multiobjective evolutionary schemes, genetic operators and filter elimination techniques for deep architecture compression for image segmentation. Their work employed multiobjective optimization algorithms to balance between multiple conflicting goals to yield a set of Pareto pruned networks. Li et al. [19] exploited a dual teacher-student architecture for semi-supervised image segmentation, where an exponential moving average of the student network trained using both labeled and unlabeled samples was used to construct the teacher model. Konar et al. [20] developed a shallow self-supervised quantum neural network for lesion segmentation, while a contextual learning network with autofocus and panorama embedding was studied by Wang et al. [21] for fine-grained lung infection segmentation.

### B. Optimization Algorithms

As a popular swarm intelligence algorithm, the PSO model simulates the flocking behaviors of birds. It initializes a number of agents with each occupying a position $\vec{x}_i^t$ in the search space. Each agent is updated in each iteration $t$ via calculating the velocity $\vec{v}_i^{t+1}$ and subsequent position $\vec{x}_i^{t+1}$ using (1) and (2). The velocity contains two key terms that affect the general search behaviors of the particles, i.e., the cognitive term $r_1c_1(\vec{p}_{\text{best}_i} - \vec{x}_i^t)$ and the social term $r_2c_2(\vec{g}_{\text{best}}^t - \vec{x}_i^t)$. The cognitive term encourages each agent to search around its personal best solution $\vec{p}_{\text{best}_i}$, while the social term directs each agent to move toward the global best position $\vec{g}_{\text{best}}$. The contributions of these terms are determined by the acceleration coefficients $c_1$ and $c_2$, randomized by $r_1$ and $r_2$ sampled from a uniform distribution $U(0, 1)$. In addition, the influence of the current velocity $\vec{v}_i^t$ to the new one is signified by $w$

$$\vec{v}_i^{t+1} = w\vec{v}_i^t + r_1c_1\left(\vec{p}_{\text{best}_i}^t - \vec{x}_i^t\right) + r_2c_2\left(\vec{g}_{\text{best}}^t - \vec{x}_i^t\right) \quad (1)$$

$$\vec{x}_i^{t+1} = \vec{x}_i^t + \vec{v}_i^{t+1}. \quad (2)$$

PSO shows great efficiency in identifying optimal CNN architectures and hyperparameters for vision and signal processing tasks. A PSO model embedded with multisurrogate schemes was proposed by Hu et al. [22] for feature selection, while an environmental PSO with probability-based fitness surface prediction was developed by Slade et al. [23] for human action recognition. Zhang et al. [24] exploited a PSO variant with super-ellipse formulae inspired hybrid leaders and root-finding algorithm-based local exploitation for audio respiratory abnormality classification. A swarm intelligence algorithm with crossover operators based on sine, cosine, and

tanh functions was also utilized by Zhang et al. [25] for bidirectional LSTM network generation pertaining to video action recognition. Lawrence et al. [26] developed a PSO variant with a residual group-based encoding mechanism for residual CNN generation. PSO with population aggregation measurement was integrated with generative adversarial networks (GANs) for facial image generation in Zhang and Zhao [27]. A multiobjective PSO combined with reinforcement learning was used by Zhang et al. [28] for multiUAV path planning.

## III. NIS-Optimized Multiloss CNN Model for Semantic Segmentation

We propose an NIS-optimized CNN model for image segmentation. It includes NIS and a multiloss function for CNN training. We use NIS to select optimal hyperparameters, such as learning rate, momentum, and loss coefficients $\alpha$ and $\gamma$. The hyperparameters $\alpha$ and $\gamma$, respectively, balance the effects of loss on present and absent classes in the GT masks and weigh the contributions of well/poorly classified examples. Using the identified hyperparameters, we train a new optimized CNN model for pixel-wise probabilistic class predictions for semantic segmentation. Moreover, the optimization of hyperparameters such as learning rate and weight decay has also been conducted for state-of-the-art transformer architectures to tackle more complex segmentation tasks. We present details on these components in Sections III-A and III-B.

### A. Proposed Search Algorithm

The proposed NIS algorithm encompasses three unique search behaviors, i.e., LSTM-CNN based Maximized Standard Deviation Velocity Prediction for global exploration, LSTM-CNN based Local Best Velocity Prediction for search diversification and $n$-Dimensional Whirlpool search for local exploitation of the optimal regions. These innovative methods for velocity vector generation incorporate machine learning techniques to better estimate/interpret different search spaces so that it can generate more effective velocity vectors, with the goal of improving both exploration and exploitation in the optimization process. A Discrete Adaptive Wave function is formulated to provide different emphasis of these three search behaviors at different search stages. The proposed velocity and position operations combining the above three search mechanisms scheduled by the Discrete Adaptive Wave function are defined in the following equations, respectively,

$$\vec{v}'^{t+1}_i = r_1 c_{d1}(t)\vec{u}^t_{\sigma_i} + r_2 c_{d2}(t)\vec{u}^t_{\beta_i} + r_3 c_{u1}(t)\vec{u}^t_{\theta_i} \quad (3)$$

$$\vec{x}^{t+1}_i = \vec{x}^t_i + \vec{v}'^{t+1}_i \quad (4)$$

where $\vec{v}'^{t+1}_i$ and $\vec{x}^{t+1}_i$ define the velocity and position vectors of the $i$th particle in the $t+1$th iteration, respectively. In (3), the velocity update operation consists of three behavioral terms as mentioned above. Specifically, the first component, i.e., $r_1 c_{d1}(t)\vec{u}^t_{\sigma_i}$, deals with exploration led by Maximized Standard Deviation Velocity $\vec{u}^t_{\sigma_i}$ to increase search territory. The second term, $r_2 c_{d2}(t)\vec{u}^t_{\beta_i}$, manages local exploration/exploitation of promising optimal regions guided by Local Best Velocity $\vec{u}^t_{\beta_i}$, while the third term, i.e., $r_3 c_{u1}(t)\vec{u}^t_{\theta_i}$, provides an exploitation mechanism to emphasize search intensification of well-established optimal regions using an angle-driven search velocity $\vec{u}^t_{\theta_i}$.

**Algorithm 1** NIS Algorithm

```
Initialise the swarm size S and particle positions
Initialise c_d1 and c_d2 using Equation 13
Initialise c_u1 using Equation 14
Initialise Velocity Prediction Model (VPM)
Initialise training data array A_data
while t < T do
    Update θ_t using Equations 11-12
    Update R_θ using Equation 10
    Collect input data from particle positions and fitnesses in array A_input
    Get VPM predictions M from A_input
    Update u⃗^t_σ from M_0
    Update u⃗^t_β from M_1
    Initialise target data array A_gt
    for each particle i = 1, ..., S do
        Update u⃗^t_θ using Equation 7
        Update velocity v⃗'^{t+1}_i using Equation 3
        Update position x⃗^{t+1}_i using Equation 4
        if f(x⃗^{t+1}_i) < f(p⃗^t_{best_i}) then
            p⃗^t_{best_i} = x⃗^{t+1}_i
        end if
        if f(x⃗^{t+1}_i) < f(g⃗^t_best) then
            g⃗^t_best = x⃗^{t+1}_i
        end if
        Generate and append target velocity vectors to A_gt
    end for
    Combine A_input and A_gt and append to A_data
    Train VPM with A_data
end while
return g⃗^t_best
```

The first two vectors, i.e., the Maximized Standard Deviation Velocity $\vec{u}^t_{\sigma_i}$ and Local Best Velocity $\vec{u}^t_{\beta_i}$, are derived from the LSTM-CNN predictions, with the third being determined by a novel $n$-dimensional spatial spiral algorithm.

These behavioral terms also contain a scheduling factor ($c_{d1}$, $c_{d2}$ or $c_{u1}$) implemented by the Discrete Adaptive Wave function. The aim is to determine the overall velocity contributions of the associated behavioral terms in regard to the current iteration. In addition, parameters $r_1$–$r_3$ contained in these terms are random scalar factors sampled from the uniform distribution $U(0.5, 1.5)$. They provide variations in the distance traveled between particles within the same iteration. As indicated in (4), after defining $\vec{v}'^{t+1}_i$, the particle's next position can be found by simply adding the velocity to the current particle position.

Algorithm 1 depicts the proposed NIS algorithm, the details of which are discussed in the following subsections.

*1) Velocity Prediction Using a Neural Network Model:* To generate the velocity vectors $\vec{u}_\sigma$ and $\vec{u}_\beta$, we employ an LSTM-CNN style network, named the velocity prediction model (VPM), as shown in Fig. 1. The key advantage of using the LSTM-CNN in this manner is its ability to directly adapt the velocity updates to the specific solution space. By learning from all previously evaluated positions, the model can develop

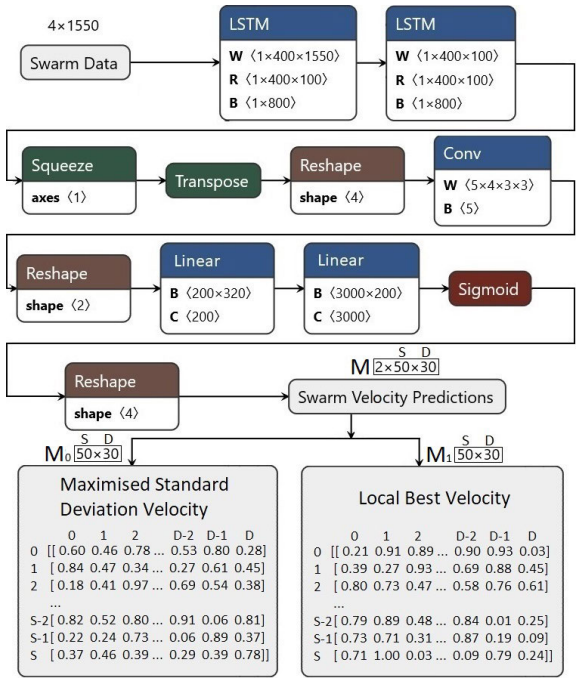



Fig. 1. VPM predicts velocity vectors for a number of particles ($S$) constrained within a $D$-dimensional search space. The swarm velocity predictions $M$ are split into $M_0$, i.e., the maximized standard deviation velocity predictions at index 0 axis 0, and $M_1$, i.e., the local best velocity predictions at index 1 axis 0.

an abstract representation of the search space, enabling the prediction of velocity vectors that maximize both exploration and exploitation.

Initially, the LSTM-CNN network employs two LSTM modules to extract sequential information from the network inputs, i.e., historical information of the particle position and the associated fitness information defined as $A_{\text{input}}$. A convolutional layer is subsequently applied to tackle spatial relationships between the particles. Fully connected linear layers are used to make the final predictions through a sigmoid layer to constrain the values between 0 and 1. By incorporating the LSTM-CNN design into the VPM, it enables the simultaneous prediction of both the $\vec{u}\sigma$ and $\vec{u}\beta$ vectors for individual particles. This is achieved by representing them as $M_0$ and $M_1$, respectively, within a single matrix $M$ of shape $(2, S, D)$, where $S$ denotes the swarm size and $D$ represents the dimensionality of the search space.

VPM training for velocity prediction with respect to global exploration is conducted from scratch, collecting and storing data samples at every iteration as $A_{\text{data}}$. Initially, the training data are sparse and the predictions rely on a few samples to produce the velocity vectors. As the search progresses, predictions improve since network training is conducted with comparatively more data collected at each iteration. Specifically, $A_{\text{input}}$ is collected at the beginning of the loop where the VPM's input data consist of the previous particle positions from the last four iterations with their associated fitness scores, forming a tensor of shape $(1, 4, (D+1)\times S)$. GTs for the target predictions of $\vec{u}_\sigma$ and $\vec{u}_\beta$ are stored in $A_{\text{gt}}$. Once the GTs of particle's velocity vector have been collected, $A_{\text{input}}$ and $A_{\text{gt}}$ are combined into a single data sample with shape $(2, S, D)$ and stored in $A_{\text{data}}$. This growing dataset serves to train the VPM before velocity prediction starts in the next iteration.

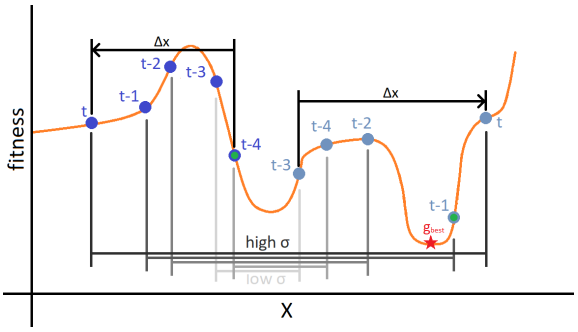



Fig. 2. Since the standard deviation of the particle positions at $t$ is higher than those at $t-1$, $\Delta x$ is taken for each particle as velocity vectors for the maximized standard deviation velocity GT target prediction. By training on these GT targets, the VPM causes particles to spread out in the search space (1-D in this graph). $\Delta x$ is the difference between the current particle position at iteration $t$ and the most distant previous particle position within $t-4$ iterations.

Sections III-A2 and III-A3 detail the GT collection processes for $\vec{u}_\sigma$ and $\vec{u}_\beta$, respectively.

*2) Maximized Standard Deviation Velocity:* The main intention of the Maximized Standard Deviation Velocity ($\vec{u}^t_{\sigma_i}$) in (3) is to ensure the searched territory of the swarm sufficiently encompasses the entire search space in a distributed manner. As such, promising areas for future exploitation can be identified. This is achieved by predicting global and local search velocity vectors using the VPM. To obtain the GT velocity vector required for the prediction of $\vec{u}^t_\sigma$, we compare the standard deviation of the particle positions in the current iteration with that from the previous iteration. If the standard deviation of the current iteration is higher, then a vector based on the difference between the particle's current position and its most distant previous position in the last four iterations is generated. Otherwise, a mirrored vector is yielded. This is repeated for each particle, yielding a GT that corresponds to the predictions defined as $M_0$ in Fig. 1, providing a tensor of shape $(S, D)$. The previous particle positions from the last four iterations in conjunction with these GT velocity vectors serve as inputs and outputs, respectively, to train the VPM as described in Section III-A1; enabling Maximized Standard Deviation Velocity prediction. To extract Maximized Standard Deviation Velocity predictions, the VPM swarm prediction matrix $M$ is indexed at 0 at axis 0 ($M_0$) yielding $\vec{u}^t_\sigma$, the predictions for every particle at iteration $t$, as indicated in (5). This LSTM-CNN-based Maximized Standard Deviation Velocity prediction guides the global search process starting from scratch and generates increasingly improved predictions to inform diversification of the swarm. As the search progresses, the predictions result in particles spreading out over the search space as indicated in Fig. 2

$$\vec{u}^t_\sigma = M_0. \tag{5}$$

*3) Local Best Velocity:* The second proposed search operation is the VPM based Local Best Velocity prediction ($\vec{u}^t_{\beta_i}$). This operation is conducted using the same VPM network, thus employs the same input training data (i.e., the previous particle positions from the last four iterations) as those used for Maximized Standard Deviation Velocity Prediction. The velocity vector target prediction aims to accelerate particle movements toward local best historical positions, instead of

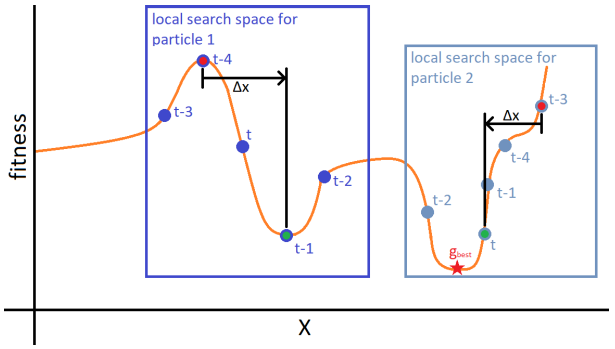



Fig. 3. Influence of the VPM accelerates the particles to move toward personal best positions where $t$ denotes the current iteration. The two boxes show two particles searching independently and how the velocity vectors ($\Delta x$) are collected and used as the training data in a 1-D search space.

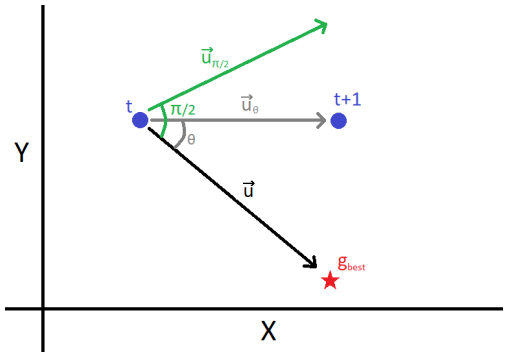



Fig. 4. $g_{\text{best}}$ refers to the global best position and $t$ represents the current iteration. The angle $\theta$ slowly decreases from $(\pi/2)$ (green line) to 0 (black line). $\vec{u}$ indicates the initial direction vector and $\vec{u}_\theta$ is $\vec{u}$ rotated by $\theta$. The magnitude of $\vec{u}$ is defined in (8). The 2-D case is displayed here but (7)–(12) show the generalization to $n$-dimensions.

global exploration as indicated in the first proposed action. The target prediction is the vector difference between the most recent best particle position and the most recent worst particle position. Target velocity vector predictions are collected for each particle giving a shape of $(S, D)$ and then used for Local Best Velocity prediction training via the $M_1$ output tensor of the VPM. The rationale behind this method is that the network can predict vectors that lead to the local optimal position of each particle across the search space, allowing the swarm to both explore and exploit distinct regions of the search space independently. This process is shown in Fig. 3. Similar to Maximized Standardized Deviation Velocity, Local Best Velocity predictions are taken from the VPM from the second index of $M$ at axis 0 ($M_1$) yielding $\vec{u}_\beta^t$, the Local Best Velocity Prediction for every particle at iteration $t$, as indicated in the following equation:

$$\vec{u}_\beta^t = M_1. \tag{6}$$

*4) n-Dimensional Whirlpool Search:* The proposed Whirlpool search is an $n$-dimensional spiral-like search. It provides exploitation of promising areas in the search space by applying an iteratively decreasing angular rotation about a unit direction vector pointing toward the global best solution, as indicated in the following equation:

$$\vec{u}_{\theta_i}^t = \vec{u}_i^t \cdot \mathbf{R}_\theta^\intercal \tag{7}$$

where $\mathbf{R}_\theta^\intercal$ is a transposed rotation matrix for rotating vectors by $\theta$, $\vec{u}_i^t$ is the nonrotated vector pointing from the $i$th particle position toward the global best position $\vec{g}_{\text{best}}^t$ at iteration $t$, and $\vec{u}_{\theta_i}^t$ being the resultant rotated vector. An example resultant rotation is displayed in Fig. 4. To increment particle positions toward the global best position, we define a direction vector, $\hat{u}_i^t$, with a variable magnitude that decreases over time to perform finer movements toward the end of the search. The largest possible movement is defined as the magnitude of the vector spanning from opposite corners of the search space ($\|\vec{b}_{\text{up}} - \vec{b}_{\text{low}}\|$). This magnitude is decreased by a cosine-based factor dependent on the maximum and current iterations. When combined with the direction vector $\hat{u}_i^t$, the complete angular rotation velocity vector ($\vec{u}_i^t$) is produced. This is formally expressed in the following equation:

$$\vec{u}_i^t = \cos\left(\frac{t}{2T}\pi\right)\hat{u}_i^t\left\|\vec{b}_{\text{up}} - \vec{b}_{\text{low}}\right\| \tag{8}$$

where $\cos((t/2T)\pi)$ is a decreasing factor. The direction vector $\hat{u}_i^t$ is obtained from the initial vector $\vec{u}_i^t$ by the division of its magnitude as in the following equation:

$$\hat{u}_i^t = \frac{\vec{g}_{\text{best}}^t - \vec{x}_i^t}{\|\vec{g}_{\text{best}}^t - \vec{x}_i^t\|} \tag{9}$$

where $\vec{g}_{\text{best}}^t$ and $\vec{x}_i^t$ are the global best position and the $i$th particle position at iteration $t$, respectively. Note that $\|\vec{g}_{\text{best}}^t - \vec{x}_i^t\|$ indicates the magnitude of the difference between $\vec{g}_{\text{best}}^t$ and $\vec{x}_i^t$.

As mentioned previously, the transposed rotation matrix $\mathbf{R}_\theta^\intercal$ defined in (7) enables $n$-dimensional rotation of the initial vector $\vec{u}_i^t$ by a given angle $\theta$. A new rotation matrix $\mathbf{R}_\theta$ is created at each iteration using two orthonormal vectors ($\hat{n}_1$ and $\hat{n}_2$) obtained through Gram-Schmidt Orthogonalization, as indicated in the following equation:

$$\begin{aligned}\mathbf{R}_\theta = \mathbf{I} &+ \hat{n}_2 \otimes \hat{n}_1 - \hat{n}_1 \otimes \hat{n}_2 \sin\theta \\ &+ \hat{n}_1 \otimes \hat{n}_1 + \hat{n}_2 \otimes \hat{n}_2(\cos\theta - 1)\end{aligned} \tag{10}$$

with $\mathbf{I}$ being an identity matrix whose rows and columns equal to the dimensionality of the search space, and $\otimes$ is the outer product operation. Besides that, $\theta$ is a dynamic angular value moving from $(\pi/2)$ to 0 in decreasing steps as shown in the following equation:

$$\theta_t = \frac{\pi}{2}\left(0.8 - \frac{t}{T}\right) \tag{11}$$

where $T$ is the total number of iterations. The factor of 0.8 ensures that $\theta$ has negative values in the last few iterations. This value is clipped to stay at 0 using the following equation, ensuring no rotation occurs toward the very end of the search leading to a linear global best search:

$$\theta_{t+1} = \begin{cases} 0, & \text{if } \dfrac{\theta_t}{\pi} < 0 \\ \theta_t, & \text{otherwise.} \end{cases} \tag{12}$$

This $n$-Dimensional Whirlpool search conducts angle-driven granular movements to exploit optimal regions around the global best solution to increase the chances of finding global optima.

TABLE I
Φ SETTINGS

| Coefficient | Behaviour | $s_1$ | $s_2$ | $s_3$ |
|---|---|---|---|---|
| $c_{d1}$ | $\vec{u}^t_{\sigma_i}$ Maximised Standard Deviation Velocity Prediction | 1.0 | 0.5 | 0.0 |
| $c_{d2}$ | $\vec{u}^t_{\beta_i}$ Local Best Velocity Prediction | 0.5 | 1.0 | 0.0 |
| $c_{u1}$ | $\vec{u}^t_{\theta_i}$ Whirlpool Search | 0.0 | 0.5 | 1.0 |

*5) Staged Discrete Adaptive Wave Function:* To maximize the exploration and exploitation capabilities of all three behavioral terms present in NIS [as defined in (3)], we introduce a function which adapts the contribution of each behavioral term based on sequential iteration ranges. These ranges, referred to as stages, allow each behavior to be configured with a weighting factor in each stage to increase or decrease its contribution. This enables bespoke macro behaviors to be created in each defined stage. Additionally, behavioral contributions to the overall velocity are increased or decreased via a sinusoidal function according to whether they are exploitative or exploratory, respectively. This ensures exploratory behaviors have a high contribution at the beginning of the search and no contribution near the end of the search, whereas exploitative behaviors do the opposite. Details of these mechanisms are shown in the following equations:

$$c_d = \frac{1}{2}\left(1 + \cos\left(\frac{t}{T}\pi\right)\right)\Phi(t, T, s_1, s_2, s_3) \tag{13}$$

$$c_u = \frac{1}{2}\left(1 - \cos\left(\frac{t}{T}\pi\right)\right)\Phi(t, T, s_1, s_2, s_3) \tag{14}$$

where $c_d$ and $c_u$ are the functions for producing the increasing or decreasing behavioral coefficients $c_d1$, $c_d2$, and $c_u1$ seen in (3). $(1/2)(1 - \cos((t/T)\pi))$ and $(1/2)(1 + \cos((t/T)\pi))$ are the decreasing and increasing sinusoidal factors and $\Phi$ is the Discrete Adaptive Wave Function. These equations are displayed in Fig. 5. The Discrete Adaptive Wave Function $\Phi$ is constructed through the addition of $n$th summations of $\psi$ which produces a function with discrete weighted stages to schedule NIS behaviors based on the current iteration $t$ of the search algorithm. In (15), we define three stages starting from 0 to the maximum iteration $T$ in increments of $(1/3)T$. Each stage has a corresponding weighting coefficient $s_1$, $s_2$ or $s_3$, which can be adjusted to increase or decrease the contribution of a particular behavior depending on the iterations falling within a given stage. The configurations of these weightings for Sections IV-A, IV-C, IV-D, and IV-E are displayed in Table I.

$$\Phi(t, T, s_1, s_2, s_3) = \sum_{n=0}^{\frac{1}{3}T} \psi(n, t, s_1) + \sum_{n=\frac{1}{3}T+1}^{\frac{2}{3}T} \psi(n, t, s_2) + \sum_{n=\frac{2}{3}T+1}^{T} \psi(n, t, s_3) \tag{15}$$

where $\psi$ is a function built upon sinc, often found in analog to digital signal conversion. We adopt this function as shown in the following equation for use with the discrete summation to enable an iteration-based scheduling as shown in (15):

$$\psi(n, t, s) = s \times \operatorname{sinc}(\pi(t - n)) = s \times \frac{\sin(\pi(t - n))}{\pi(t - n)}. \tag{16}$$

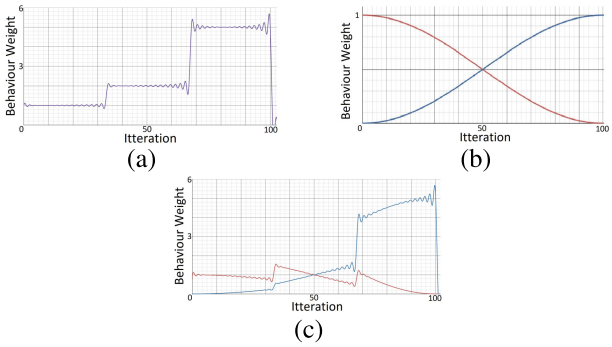



Fig. 5. (a) Equation (15) where $s_1 = 1$, $s_2 = 2$, $s_3 = 5$, and $T = 100$. (b) Factors $[1 + \cos((t/T)\pi)]$ (red) and $[1 - \cos((t/T)\pi)]$ (blue) from (13) and (14), respectively. (c) Equation (13) (red), and (14) (blue), i.e., the combination of (a) and (b).

As indicated in the velocity and position formulae in (3) and (4), the Scheduled Adaptive Coefficients ($c_{d1}$, $c_{d2}$, and $c_{d3}$) are combined with the three previously defined behaviors ($\vec{u}^t_{\sigma_i}$, $\vec{u}^t_{\beta_i}$, and $\vec{u}^t_{\theta_i}$) from (5)–(7), respectively. The full algorithm of NIS is indicated in Algorithm 1. Owing to this Discrete Wave function, the proposed algorithm employs an adaptive emphasis of the aforementioned three search behaviors driven by neural network based velocity prediction and angle rotation-based search movement to balance between diversification and intensification.

### B. Proposed Multiloss Function

Common loss functions for image segmentation include Cross Entropy, Soft Dice, and Focal loss. Cross Entropy measures the difference between GT and predicted masks, Soft Dice uses Sørensen-Dice coefficient to measure similarity, and Focal loss adjusts the impact of loss contributions for well and poorly classified examples and classes present and nonpresent in GT masks.

To take advantage of the loss information from the aforementioned variants, we propose a new multiloss function as shown in (17). It combines the complementary error signals from Cross Entropy, Dice, and Focal Loss schemes. Such a multiloss mechanism is able to provide compound loss indicators to advise the backpropagation process and adjust performance. In particular, to balance the effects of the well/poorly classified examples and contributions of the classes present/nonpresent in the GT masks, we optimize the $\gamma$ and $\alpha$ coefficients in the focal loss function using the proposed NIS algorithm

$$\text{ML} = \text{FL}(\gamma, \alpha) + \text{SDS} + \text{CE}. \tag{17}$$

Specifically, the $\alpha$ loss coefficient balances the influence between the classes that are present and nonpresent in the GT masks at the pixel level. A higher $\alpha$ emphasizes the classes present in the GT mask, while a lower value shifts the emphasis toward those that are not present. Each column in Fig. 6 indicates the impact of different $\alpha$ settings, where the blue and red lines show the loss contributions of the classes that are present and nonpresent in the GT masks, respectively. In each row, the loss graphs are generated using a fixed $\gamma$ value (i.e., $\gamma = 0, 1$, or 5), with (a) $\alpha = 0.1$, (b) $\alpha = 0.5$, and (c) $\alpha = 0.9$. To be specific, (a) indicates higher contributions of the classes that are not present in the GT masks,

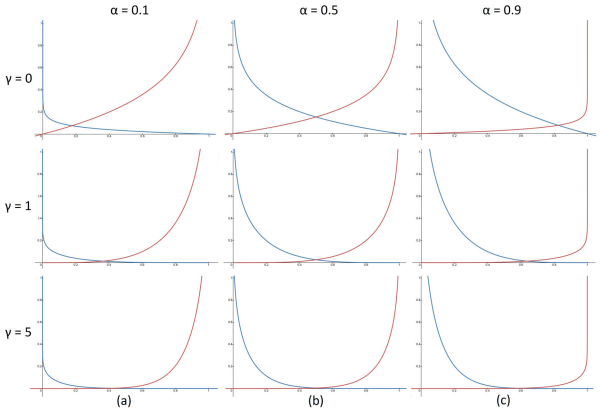



Fig. 6. Blue and red lines, respectively, show the loss contributions of the classes that are present and nonpresent in the GT masks, respectively, for the focal loss. The graphs are arranged into three columns with varied $\alpha$ values and three rows with different $\gamma$ values showing the resultant loss curves. The figure exhibits three columns of graphs (labeled a, b, c) illustrating loss contribution variations with different $\alpha$ and $\gamma$ values. (a) $\alpha = 0.1$, $\gamma = 0$, 1, 5 highlights higher contributions from non-present classes. (b) $\alpha = 0.5$, $\gamma = 0$, 1, 5 demonstrates balanced effects of present and non-present classes. (c) $\alpha = 0.9$, $\gamma = 0$, 1, 5 showcases a higher impact of the present classes.

and (b) shows the balanced effects of both classes that are present and nonpresent, while (c) implies higher impact of the classes that are present.

The $\gamma$ loss coefficient balances between pixels with true positive/true negative predictions and those with false positive/false negative predictions. We refer to classes with considerable true positive and true negative predictions as well classified, and those with substantial false positive and false negative predictions as poorly classified. A higher $\gamma$ emphasizes the contribution of the well-predicted classes at the pixel level, whereas a lower $\gamma$ setting increases the contribution of the poorly classified classes by reducing the influence of well-classified ones. Each row in Fig. 6 shows the effects of different $\gamma$ configurations.

As indicated in Fig. 6, different settings of $\alpha$ and $\gamma$ loss coefficients play significant roles in the resultant loss function behaviors defined in (17). We optimize these two hyperparameters along with the learning rate and momentum settings using the proposed NIS algorithm to further fine-tune model learning behaviors.

## IV. Evaluation

We employ five well-known semantic segmentation datasets, as well as mathematical numerical test functions to evaluate the proposed model against several baseline search methods.

### A. Segmentation Datasets

We employ five datasets, i.e., CamVid, Freiburg Forest, MESSIDOR, ADE20K, and Cityscapes for evaluating segmentation models. These datasets are detailed as follows.

The CamVid dataset has a total of 701 images in a resolution of 960 × 720. An official data split is provided with 369 training, 100 validation, and 232 test samples. To reduce the memory requirements, we compressed the original 32 categories into the following 12 semantically similar classes, i.e., Void, Sky, Building, Pole, Road, Pavement, Tree, SignSymbol, Fence, Car, Pedestrian, and Bicyclist. The Freiburg Forest dataset contains 366 images in a resolution of 882 × 490. It contains classes of object, trail, grass, tree, vegetation, and sky. The tree and vegetation classes are combined since the latter is not used in the test set. An official split of 230 training and 136 test samples is provided. The MESSIDOR dataset contains 1200 color retinal images and binary segmentation masks at multiple resolutions (1440 × 960, 2240 × 1488, and 2304 × 1536) for segmentation and detection of optic disks. This study resizes GTs and images to 640×480 using an 80–20 train-test split. ADE20K contains 20 210 and 2000 images for training and validation, respectively, with 150 semantic categories. Cityscapes consists of 2975 and 500 images for training and validation, respectively, with 19 semantic classes.

TABLE II
Results for Dice Score, GA, MIoU, and MCA for Models Trained With Different Losses on the CamVid Dataset With the Top Ten Results Highlighted

| Model | Cross Entropy | Dice | Focal | Dice (%) | GA (%) | MIoU (%) | MCA (%) |
|---|---|---|---|---|---|---|---|
| FCN | ✓ | × | × | **61.1** | **87.6** | **50.2** | **62.7** |
| | × | ✓ | × | **62.0** | **86.8** | **51.3** | **62.4** |
| | × | × | ✓ | **61.3** | **86.6** | **50.5** | **63.1** |
| DeeplabV3 | ✓ | × | × | **57.7** | **84.6** | **46.3** | **59.6** |
| | × | ✓ | × | **63.0** | **87.5** | **52.4** | **63.3** |
| | × | × | ✓ | **60.6** | **87.6** | **49.8** | **63.0** |
| Unet++ | ✓ | × | × | **54.3** | 81.4 | **43.5** | **57.8** |
| | × | ✓ | × | **55.2** | 79.6 | **44.3** | **56.3** |
| | × | × | ✓ | **55.0** | **83.8** | **44.8** | **58.1** |
| Linknet | ✓ | × | × | 43.1 | 83.5 | 34.5 | 49.0 |
| | × | ✓ | × | 33.4 | 79.3 | 27.5 | 38.2 |
| | × | × | ✓ | 36.8 | 81.8 | 30.4 | 45.2 |
| LR-ASPP | ✓ | × | × | 41.3 | 71.5 | 31.1 | 42.4 |
| | × | ✓ | × | 41.6 | 78.2 | 32.6 | 42.5 |
| | × | × | ✓ | 42.9 | 79.7 | 33.6 | 44.6 |
| MAnet | ✓ | × | × | 47.0 | **84.1** | 38.1 | 53.2 |
| | × | ✓ | × | **51.3** | **84.6** | **42.2** | **53.7*** |
| | × | × | ✓ | 47.9 | **85.5** | 38.8 | **53.7*** |
| PSPnet | ✓ | × | × | 39.7 | 58.4 | 28.5 | 37.8 |
| | × | ✓ | × | 44.0 | 63.9 | 32.2 | 42.0 |
| | × | × | ✓ | 39.8 | 58.7 | 28.7 | 38.0 |

* represents a shared tenth place result

### B. Loss Function Evaluations

We explore NIS-devised networks in combination with different loss functions and indicate efficiency of the proposed combined multiloss scheme.

To be specific, each model was trained with one of the existing three loss functions, i.e., Cross Entropy, Dice, and Focal loss, commonly used for semantic segmentation tasks, as well as a diverse combination of these loss functions, including the newly proposed one. The performance of these loss functions was evaluated across seven established segmentation models, i.e., FCN [4], DeeplabV3 [3], Unet++ [5], Linknet [29], LR-ASPP [6], MAnet [2], and PSPnet [30]. Each model was trained for 50 epochs using the stochastic gradient descent (SGD) with a learning rate of 0.01 and a momentum of 0.5. A train-validation split was taken from the original training sets to preserve the true test data. Four commonly used segmentation metrics were calculated from the test set for comparison, i.e., Dice Score, global accuracy (GA), mIoU, and mean class accuracy (MCA). These results can be found in Table II.

*1) Loss Function Comparison:* We present the segmentation results for the CamVid dataset using each of the three loss functions in Table II. As indicated in Table II, the top ten results for each metric for this dataset are mostly distributed between FCN, DeeplabV3 and Unet++ with the remaining good results being attained by MAnet. The top result (63%) for the Dice score is obtained by DeeplabV3 using the Dice loss function. For the GA rates, the top result (87.6%) is shared jointly between FCN with Cross Entropy loss and DeeplabV3 with Focal loss. The DeeplabV3 with Dice loss

TABLE III

DICE SCORE, GA, MIOU, AND MCA FOR MODELS TRAINED WITH DIFFERENT COMBINATIONS OF LOSSES ON THE CAMVID DATASET

| Model | Cross Entropy | Dice | Focal | Dice (%) | GA (%) | MIoU (%) | MCA (%) |
|---|---|---|---|---|---|---|---|
| FCN | ✓ | × | × | 61.1 | 87.6 | 50.2 | 62.7 |
| | × | ✓ | × | 62.0 | 86.8 | 51.3 | 62.4 |
| | × | × | ✓ | 61.3 | 86.6 | 50.5 | 63.1 |
| | ✓ | ✓ | × | 63.6 | 87.6 | 52.8 | 64.5 |
| | ✓ | × | ✓ | 61.5 | 87.2 | 50.6 | 63.2 |
| | × | ✓ | ✓ | **64.4** | 88.1 | 53.6 | **66.0** |
| | ✓ | ✓ | ✓ | 64.3 | **89.2** | **53.8** | 65.6 |
| DeeplabV3 | ✓ | × | × | 57.7 | 84.6 | 46.3 | 59.6 |
| | × | ✓ | × | 63.0 | 87.5 | 52.4 | 63.3 |
| | × | × | ✓ | 60.6 | 87.6 | 49.8 | 63.0 |
| | ✓ | ✓ | × | 63.5 | 87.8 | 52.6 | 65.7 |
| | ✓ | × | ✓ | 61.4 | 87.9 | 52.9 | 66.4 |
| | × | ✓ | ✓ | 63.2 | 87.4 | 52.2 | 65.0 |
| | ✓ | ✓ | ✓ | **64.3** | **89.5** | **53.7** | **67.6** |
| Unet++ | ✓ | × | × | 54.3 | 81.4 | 43.5 | 57.8 |
| | × | ✓ | × | 55.2 | 79.6 | 44.3 | 56.3 |
| | × | × | ✓ | 55.0 | 83.8 | 44.8 | 58.1 |
| | ✓ | ✓ | × | 61.8 | 83.1 | 50.0 | **64.2** |
| | ✓ | × | ✓ | 57.7 | 84.3 | 46.4 | 60.5 |
| | × | ✓ | ✓ | 60.0 | 81.4 | 47.6 | 61.4 |
| | ✓ | ✓ | ✓ | **62.3** | **86.2** | **51.0** | **64.2** |

model achieves both the highest MIoU (52.4%) and MCA (63.3%) scores. It is clear that the most consistently accurate models are DeeplabV3, FCN, and Unet++. Regarding loss functions, models trained with Dice loss often provide the highest results. To further assess the effectiveness of the combination of different loss functions, we use the three best-performing networks for subsequent experiments.

*2) Multiloss Function Results:* Since each of the previously used loss functions capture unique aspects of the error present in the dataset, further investigation was conducted to determine if the benefits provided by each loss function can be exploited simultaneously. Toward this end, the three best-performing networks, i.e., DeeplabV3, FCN, and Unet++, were trained with every combination of loss functions on the CamVid dataset using the same training regimen discussed previously. These results are provided in Table III. The results indicate that models trained with all three loss functions typically produce the top results, otherwise yielding the second best results. The lowest results are almost consistently associated with models trained with single loss functions. Of these models, DeeplabV3 and FCN networks typically outperformed Unet++ with higher metric scores, leading to top results. These observations justify using DeeplabV3 and FCN and the combined multiloss function for further evaluation of NIS optimization on CNN segmentation models.

### C. CNN Segmentation Model Evaluation Using CamVid, Freiburg Forest, and MESSIDOR

In this section, we employ the proposed NIS model for hyperparameter identification of the best-performing networks, i.e. DeeplabV3 and FCN. In particular, the multiloss function identified earlier is used as the fitness function, which integrates Cross Entropy, Dice, and Focal loss measures. The Firefly Algorithm (FA) and PSO are utilized as the baseline methods for optimal hyperparameter selection. The experimental setup is firstly explained. We then analyze the results and discuss notable patterns with respect to each dataset. A summary of the combined results from all datasets is also provided. The selected hyperparameters are presented and discussed with regard to their effect on model performance.

To evaluate each segmentation network, NIS, PSO, and FA are employed to identify the optimal settings of the learning rate, momentum, $\gamma$, and $\alpha$ in the multiloss function. In particular, $\gamma$ and $\alpha$ are the loss coefficients for the Focal loss function, as discussed earlier. The identified hyperparameters are then used to train the model on the combined training and validation sets, before being evaluated with the test set. The optimal hyperparameter identification process is performed five times. We present the mean results over five runs for DeeplabV3 and FCN with respect to each dataset in Section IV-C1.

TABLE IV

HYPERPARAMETERS TARGETED FOR OPTIMIZATION

| Hyper-Paremeter | Lower Bound | Upper Bound |
|---|---|---|
| Learning Rate | 0.0001 | 0.01 |
| Momentum | 0.0 | 1.0 |
| $\alpha$ (loss parameter balancing between the classes that are present and non-present in the GT masks) | 0.15 | 0.99 |
| $\gamma$ (loss coefficient weighting contributions of well and poorly classified examples) | 1.00 | 5.0 |

TABLE V

MEAN RESULTS OF FOUR COMMON METRICS OVER FIVE RUNS FOR THE NIS-OPTIMIZED DEEPLABV3 MODEL ON THREE DATASETS

| Dataset | Search | Dice (%) | GA (%) | MIoU (%) | MCA (%) |
|---|---|---|---|---|---|
| CamVid | Default | 61.0 | 84.9 | 51.8 | 63.6 |
| | FA | 72.8 | 85.1 | 59.7 | 70.2 |
| | PSO | 78.5 | 90.0 | 66.7 | 74.8 |
| | NIS | **79.8** | **90.5** | **68.3** | **76.3** |
| Freiburg | Default | 85.3 | 93.4 | 73.0 | 83.2 |
| | FA | 88.2 | 93.8 | 80.4 | 86.5 |
| | PSO | 88.3 | **94.2** | 80.7 | 86.7 |
| | NIS | **89.0** | 94.1 | **81.4** | **87.7** |
| MESSIDOR | Default | 81.1 | 99.3 | 74.4 | 72.0 |
| | FA | 92.5 | 99.7 | 87.6 | 90.5 |
| | PSO | 83.6 | 99.5 | 76.6 | 77.7 |
| | NIS | **95.6** | **99.8** | **92.1** | **93.9** |

As a reference, the default results of both DeeplabV3 and FCN without hyperparameter optimization are also provided. In the default experimental settings, instead of the multiloss function, a standard Cross Entropy loss is used. The networks are trained with an SGD optimizer with a default learning rate of 0.01 and a default momentum of 0.5. The final results of each network are obtained by taking the average of five runs.

The following settings remain constant throughout all experiments. All algorithms use a population of 10, a maximum iteration of 20, and a set of five runs. Each fitness evaluation trains each CNN model for two epochs before evaluation. Moreover, the VPM LSTM-CNN network is trained using the SGD optimizer with the following settings: Lr $= 0.0001$, momentum $= 0.9$, and weight decay $= 0.005$, which are determined using trial-and-error.

The search ranges for the optimization targets are shown in Table IV. The devised DeeplabV3 and FCN models with optimal settings are both trained with SGD for 50 epochs, along with a weight decay of 0.005 and a batch size of 4.

*1) Results:* The Dice score, GA, MIoU and MCA are used to measure the performance of the NIS optimized Multiloss CNN models. Evaluation results on the CamVid, Freiburg Forest and MESSIDOR datasets for the devised DeeplabV3 and FCN models are shown in Tables V and VI, respectively. We also analyze the selected hyperparameters in Section IV-C2.

Tables V and VI depict that NIS yields a superior performance over the standard PSO and FA methods across all three test datasets for both networks. The search strategies of NIS contribute toward improved hyperparameter selection, thus increase model prediction accuracy across all four metrics. Furthermore, the models trained with optimized

TABLE VI

MEAN RESULTS OF FOUR COMMON METRICS OVER FIVE RUNS FOR THE NIS-OPTIMIZED FCN MODEL ON THREE DATASETS

| Dataset | Search | Dice (%) | GA (%) | MIoU (%) | MCA (%) |
|---|---|---|---|---|---|
| CamVid | Default | 51.4 | 84.3 | 42.6 | 54.6 |
| | FA | 70.6 | 83.6 | 57.4 | 67.3 |
| | PSO | 74.9 | 87.6 | 62.5 | 71.3 |
| | NIS | **79.5** | **90.5** | **68.1** | **76.4** |
| Freiburg | Default | 62.8 | 90.1 | 55.7 | 62.2 |
| | FA | 85.9 | 92.5 | 77.1 | 84.5 |
| | PSO | 77.7 | 87.8 | 68.4 | 75.9 |
| | NIS | **86.7** | **92.7** | **78.0** | **85.5** |
| MESSIDOR | Default | 81.0 | 99.2 | 76.6 | 80.8 |
| | FA | 90.6 | 99.7 | 86.5 | 90.3 |
| | PSO | 85.8 | 99.6 | 80.2 | 82.4 |
| | NIS | **94.6** | **99.8** | **90.3** | **94.8** |

hyperparameters perform better than those trained with typical default configurations. Specifically, models trained with default settings use a single Cross Entropy loss function with default learning settings, while our devised networks are equipped with an optimized multiloss function in combination with more effective learning settings. As such, the latter shows great efficiency in learning from the backpropagation process to adjust the performance with customized learning behaviors. These observations are empirically shown across all three datasets, indicating that using NIS hyperparameter optimization together with a multiloss function provides benefits across multiple problems and model structures.

The improved performance gained from using a multiloss function results from the combination of diverse and unique loss calculation mechanisms. Each constituent loss function measures the error between the prediction and the GT differently making the error signal inherently more informative. This leads to better error correction during training as compared with using a single loss function, yielding improvements in the predictive capability of the resulting model. In addition, optimized parameters $\gamma$ and $\alpha$ enable the multiloss function to strike a balance between the impact of well/poorly classified instances and contributions of the classes present/nonpresent in the GT masks, thus preventing overfitting.

*2) Hyperparameter Selection:* An overview of the hyperparameter selection results of each optimization method across all three datasets are provided in Tables VII and VIII for the DeeplabV3 and FCN models, respectively. Each table displays the optimized learning rate (Lr), momentum, $\alpha$, and $\gamma$ configurations.

By analyzing Tables V–VIII, we found that lower values of learning rate, $\gamma$ and $\alpha$, along with higher momentum, lead to better accuracy for all models and datasets. NIS efficiently identifies these favorable hyperparameters, while FA and PSO are more affected by swarm initialization.

Most CNNs use high momentum and low learning rates as typical hyperparameters. This trend is reflected in the results of most algorithms. NIS identifies the importance of a low average learning rate, in addition to a high mean momentum, contributing to high accuracy in generated networks. A lower learning rate allows for finer adjustments to network weights during training, providing greater variation and granularity in potential internal network representations, resulting in higher accuracy. High momentum emphasizes the trajectory toward a global optimum, reducing the likelihood of weights being trapped in local optima or going beyond global optimum solutions.

TABLE VII

AVERAGE HYPERPARAMETERS IDENTIFIED OVER FIVE RUNS USING SEARCH METHODS BASED ON THE DEEPLABV3 MODEL ON THREE DATASETS

| Dataset | Search | Lr | Momentum | $\alpha$ | $\gamma$ |
|---|---|---|---|---|---|
| CamVid | FA | 0.0093 | 0.8297 | 0.4625 | 3.9188 |
| | PSO | 0.0093 | 0.8274 | 0.6043 | 3.4761 |
| | NIS | 0.0059 | 0.8707 | 0.3786 | 1.9099 |
| Freiburg | FA | 0.0072 | 0.8008 | 0.4230 | 1.5410 |
| | PSO | 0.0067 | 0.8636 | 0.4866 | 2.8988 |
| | NIS | 0.0054 | 0.7514 | 0.3506 | 2.0119 |
| MESSIDOR | FA | 0.0079 | 0.8732 | 0.6860 | 2.1909 |
| | PSO | 0.0083 | 0.9053 | 0.7007 | 3.1876 |
| | NIS | 0.0036 | 0.8345 | 0.4448 | 2.4869 |

TABLE VIII

AVERAGE HYPERPARAMETERS IDENTIFIED OVER FIVE RUNS USING SEARCH METHODS BASED ON THE FCN MODEL ON THREE DATASETS

| Dataset | Search | Lr | Momentum | $\alpha$ | $\gamma$ |
|---|---|---|---|---|---|
| CamVid | FA | 0.0077 | 0.5421 | 0.5848 | 3.9476 |
| | PSO | 0.0084 | 0.6732 | 0.4753 | 3.2803 |
| | NIS | 0.0065 | 0.8214 | 0.2815 | 2.6709 |
| Freiburg | FA | 0.0052 | 0.7462 | 0.6782 | 2.7889 |
| | PSO | 0.0052 | 0.4689 | 0.7493 | 3.7681 |
| | NIS | 0.0030 | 0.8105 | 0.3356 | 1.5221 |
| MESSIDOR | FA | 0.0073 | 0.6850 | 0.2335 | 2.5713 |
| | PSO | 0.0076 | 0.6550 | 0.4807 | 3.6714 |
| | NIS | 0.0024 | 0.8584 | 0.2420 | 2.3641 |

Low $\alpha$ loss coefficients configure the focal loss function to enable CNN to concentrate on classes not present in GT during training. Low $\gamma$ loss settings adjust weights to reduce incorrect predictions instead of improving correct ones, which can stabilize training in multiclass scenarios. For MESSIDOR dataset, low $\alpha$ and $\gamma$ values have a lesser impact due to the foreground-background binary segmentation process resulting in a lower number of classes. In such cases, the focus is on finer weight adjustments with lower learning rates and higher momentums to avoid local optima.

A detailed analysis of hyperparameters and results from DeeplabV3 networks was conducted to further examine the optimization algorithms. Tables V and VII show the segmentation results and hyperparameters identified using DeeplabV3. Results for the CamVid dataset indicate that NIS outperforms PSO and FA, as it selects high momentums with low $\alpha$, $\gamma$, and learning rate configurations, resulting in a Dice score of 79.8%. PSO and FA select moderate $\alpha$ and high $\gamma$ coefficients. Such a combination increases error signals from both correctly and incorrectly classified examples, while reducing the error signals for moderately classified examples. This leads to less informative error signals, producing instability in the network during training and encouraging under-fitting. High learning rates and momentums identified by PSO and FA, when combined with the above loss function settings, exacerbate the under-fitting issue. However, the other two loss functions may compensate for poor configurations identified by PSO and FA, preventing the models from failing completely.

The results of DeeplabV3 trained on the Freiburg forest dataset reveal that NIS prefers low $\alpha$ and $\gamma$ values with high momentums and low learning rates, yielding a Dice score of 89%. PSO selects high learning rates, momentums, and $\gamma$ values with moderate $\alpha$ values, resulting in a Dice score of 88.3%. FA stands out with a Dice score of 88.2%, selecting high learning rates, high momentums, moderate $\alpha$, and low $\gamma$ values. However, the performance gain over PSO is minor because of the high learning rates and momentums. Nonetheless, high momentums with low $\alpha$ and $\gamma$ settings produce superior results, with minimal variation across models.

TABLE IX
COMPARISON WITH OTHER REPORTED RESULTS

| Method | Dice (%) | GA (%) | MIoU (%) | MCA (%) |
|---|---|---|---|---|
| **MESSIDOR** | | | | |
| Abdulla et al. [31] | 93.39 | **99.89** | 87.9 | - |
| Morales et al. [32] | 89.50 | 99.49 | 82.28 | - |
| Kumar et al. [33] | 84.56 | - | - | - |
| Rehman et al. [34] | 85.1 | 98.8 | 74.7 | - |
| Zahoor and Fraz [35] | 90.3 | 99.1 | 84.4 | - |
| Fan et al. [36] | 91.96 | 97.7 | 86.3 | - |
| NIS-Deeplabv3 | **95.6** | 99.8 | **92.1** | **93.9** |
| **Freiburg Forest** | | | | |
| FC-DenseNet67 [8] | - | - | 74.47 | - |
| FCN8 [8] | - | - | 80.05 | - |
| ParseNet [8] | - | - | 80.89 | - |
| FastNet [8] | - | - | 81.00 | - |
| CGBNet [8] | - | - | 81.04 | - |
| SegNet [8] | - | - | 81.12 | - |
| Zhang et al. [7] | - | 92.07 | 79.87 | - |
| NIS-Deeplabv3 | **89.0** | **94.1** | **81.4** | **87.7** |
| **CamVid** | | | | |
| Segnet [11] | - | 90.40 | 60.10 | 71.20 |
| Dilation + FSO [37] | - | - | 66.12 | - |
| LRN [38] | - | - | 61.7 | **77.2** |
| G-FRNet [9] | - | - | 68.0 | - |
| FC-DenseNet103 [10] | - | **91.5** | 66.9 | - |
| DPDB-Net [39] | - | 85.2 | 54.7 | - |
| TD2-PSP50 [40] | - | - | 43.5 | 55.2 |
| MSSA [41] | - | - | 74.12 | 55.2 |
| NIS-Deeplabv3 | **79.8** | 90.5 | **68.3** | 76.3 |

The MESSIDOR dataset results show that high momentum, low $\alpha$, low $\gamma$, and low learning rate hyperparameters configurations are preferred by NIS, resulting in a Dice score of 95.6% for DeeplabV3 model. However, PSO chooses high values for learning rate, momentum, $\gamma$, and $\alpha$, leading to a Dice score of 83.6%. FA selects high values for learning rate and momentum and low values for $\gamma$ resulting in a Dice score of 92.5%. Although FA selects high values for learning rate and $\alpha$, they are still lower than those selected by PSO. Whilst FA-optimized configurations with reduced $\gamma$ values improve the performance compared to PSO, they have worse performance than those of NIS.

Similar observations for hyperparameter selection for FCN are also obtained from Tables VI and VIII. In summary, NIS outperforms FA and PSO in optimizing DeeplabV3 and FCN networks across three segmentation datasets by selecting low $\alpha$, $\gamma$, learning rate parameters, and high momentum settings.

We notice that both FA and PSO lack of an adaptive balance of exploration and exploitation search behaviors. FA also struggles with search oscillations and slow convergence since its agents are guided purely by local brighter individuals. NIS addresses this issue by introducing an iteration-based scheduling of novel diversification and intensification operations through the proposed Discrete Adaptive Wave Function. This function maximizes or minimizes different search behaviors in three separate stages using heuristic coefficients. The proposed search strategies, including LSTM-CNN-based optimal velocity prediction and Whirlpool search, effectively avoid early stagnation, as indicated by experimental results.

In Table IX, we present a comparison of the NIS-optimized DeeplabV3 network against state-of-the-art models on the three datasets, owing to the efficiency of the devised DeeplabV3 network. The empirical results indicate that our optimized network yields improved performance in comparison with those of other deep networks across multiple metrics.

### *D. Evaluation Using ADE20K and Cityscapes*

To further test model efficiency, two larger datasets, i.e. ADE20K and Cityscapes, are also employed in our experimental studies. A transformer network, i.e. Mask2Former [13], is employed as the base segmenter, because of its superior global feature learning capabilities in comparison with those of CNN-based models. We use the source code of Mask2Former released by its original study [13] in our experiments. Specifically, Mask2Former with Swin-Large (IN21k) as the backbone is adopted in our experiments owing to its impressive performance for semantic segmentation. Each search method is used to optimize hyperparameters, i.e. the learning rate and weight decay, of Mask2Former.

*1) Evaluation Using ADE20K:* For hypeparameter search, a subset of 2021 images (10%) is extracted from the training set of ADE20K. A population size of 10 and a maximum iteration number of 20 are employed for optimal parameter selection using NIS. The resulting optimal learning configurations are used to establish the optimized transformer network, which is trained using the whole training set of ADE20K with a large number of training iterations. The trained model is then tested using the official validation set. The above process is repeated five times for hyperparameter search. Besides implementing the NIS/PSO/FA-optimized Mask2Former, a baseline Mask2Former with the same backbone is also implemented where it is loaded with the ADE20K pretrained weights provided by its original study and subsequently tested using the validation set. This baseline model employs a learning rate of $1e{-}04$ and a weight decay of $5e{-}02$, provided by the original study. The optimized and baseline Mask2Former models employ 160 K training iterations to ensure a fair comparison. Table X illustrates the mean results of the ADE20K validation set over five runs for optimized and baseline Mask2Former models based on a single-scale inference. The symbol "*" in Table X indicates that the results are obtained in our own experiments. Recommended by [13], MIoU is used as the main performance indicator. Four NVIDIA 1080ti GPUs are used in our studies. As indicated in Table X, for ADE20K validation set, the NIS-optimized transformer model achieves state-of-the-art performance and outperforms the baseline, and PSO and FA-optimized transformers, as well as other state-of-the-art existing studies.

*2) Evaluation Using Cityscapes:* For Cityscapes, we subsample 298 images (10%) from the training set for hyperparameter search. The same experimental settings of ADE20K are also used in this experiment for parameter optimization, i.e. population $= 10$, iteration $= 20$ and trial $= 5$. Each optimized Mask2Former is evaluated using the official validation set. The baseline transformer with pretrained weights of Cityscapes provided by its original study is also evaluated using the validation set in our experiments. The mean performances of the optimized and baseline networks for the Cityscapes validation set over five runs based on single-scale inference are also provided in Table X. As indicated in Table X, the NIS-optimized transformer model also achieves state-of-the-art performance and outperforms the baseline, and PSO and FA-optimized transformers, as well as other existing studies.

*3) Hyperparameter Selection:* Table X also shows the mean hyperparameter selection results over five runs for each search method for both datasets. For ADE20K, as indicated in Table X, NIS, PSO and FA identify comparatively smaller mean learning rate settings in comparison with that of the baseline model. Such smaller learning rate configurations compensate well with the high loss from a large training set, which leads to delicate weight updates for model training.

TABLE X

MEAN RESULTS OVER FIVE RUNS FOR THE OPTIMIZED AND BASELINE MASK2FORMER MODELS ON ADE20K AND CITYSCAPES VALIDATION SETS

| Method | MIoU (%) |
|---|---|
| **ADE20K Validation** | |
| ISNet [42] | 47.55 |
| RegionContrast [43] | 46.85 |
| Seg-L-Mask [14] | 51.30 |
| CTNet [12] | 45.94 |
| TopFormer-B [44] | 37.8 |
| HRViT-b3 [45] | 50.20 |
| SegNeXt-L [46] | 51.0 |
| CSWin-L [47] | 54 |
| MaskFormer + FaPN [48] | 55.2 |
| SETR-PUP [49] | 48.58 |
| DFlatFormer [50] | 45.9 |
| Baseline Mask2Former* [13] (lr=1e-04, wd=5e-02) | 55.76 |
| PSO-optimised Mask2Former* (lr=1e-05, wd=2.783e-02) | 56.16 |
| FA-optimised Mask2Former* (lr=2.33e-05, wd=1e-04) | 55.96 |
| NIS-Optimised Mask2Former* (lr=1e-05, wd=3.698e-02) | 57.39 |
| **Cityscapes Validation** | |
| ISNet [42] | 81.10 |
| RegionContrast [43] | 81.3 |
| CTNet [12] | 82.2 |
| SegFormer (MiT-B5) [51] | 83.1 |
| HRViT-b3 [45] | 83.16 |
| SegNeXt-L [46] | 83.2 |
| FaPN [48] | 78.5 |
| SETR-PUP [49] | 79.34 |
| TrSeg [52] | 79.9 |
| DFlatFormer [50] | 80.6 |
| Baseline Mask2Former* [13] (lr=1e-04, wd=5e-02) | 83.3 |
| PSO-optimised Mask2Former* (lr=1e-05, wd=1.67e-02) | 83.34 |
| FA-optimised Mask2Former* (lr=4.28e-05, wd=4.03e-03) | 83.05 |
| NIS-Optimised Mask2Former* (lr=2.09e-05, wd=3.28e-02) | 83.67 |

* indicates results obtained through our own experiments.

On the contrary, large learning rates as in the baseline model may cause oscillations in gradient descent, therefore leading to less competitive performance. Moreover, NIS identifies a moderate mean weight decay in comparison with the default and those obtained using PSO and FA, which applies a reasonable penalty to the loss to maintain sufficient generalization capabilities. Applying very small weight decays as in FA and PSO may diminish the effects of the regularization term to cause overfitting, while adopting very large weight decay settings as in the baseline method may also squash the weights down too much to incur underfitting or premature convergence. The combination of a small mean learning rate and a moderate mean weight decay in NIS leads to optimal performance. A similar observation is also obtained for the hyperparameters identified by NIS for Cityscapes, i.e. a smaller mean learning rate and a moderate mean weight decay.

These experimental studies for ADE20K and Cityscapes indicate that the proposed NIS algorithm embedding LSTM-CNN based maximized standard deviation and local best velocity prediction and whirlpool search-based local intensification shows great efficiency in overcoming local optima traps for optimal hyperparameter search. The NIS-optimized transformer model is able to extract more precise long-range global dependencies to inform pixel-wise probabilistic class predictions for semantic segmentation.

*4) Optimization Cost:* The following analysis gives insight into the extra cost incurred by the optimization process during training. Since fitness evaluation involving CNN or transformer training and test is the most costly component in comparison with the dataflow of each search algorithm, and the same number of function evaluations is used as the termination criterion for all optimizers, the computational cost is mostly identical between different search methods. We provide the mean cost of one function evaluation along with one implementation of each search method over five runs

TABLE XI

COMPUTATIONAL COST (IN SECONDS) OF A SINGLE FUNCTION EVALUATION FOR EACH OPTIMIZATION METHOD AND DATASET

| Method | MESSIDOR | Frieburg | CamVid | Cityscapes | ADE20k |
|---|---|---|---|---|---|
| NIS | 21.11 | 25.12 | 31.90 | 401.77 | 471.83 |
| Firefly | 18.80 | 22.19 | 26.80 | 137.07 | 231.05 |
| PSO | 19.74 | 24.06 | 29.57 | 333.93 | 397.87 |

in Table XI, for cost comparison. Four NVIDIA 1080ti GPUs are used in parallel to generate the costs shown in Table XI.

As indicated in Table XI, NIS shows a similar average cost to those of other search methods. Because of the adaptive deployment of neural network-based velocity predictions and the whirlpool search based local exploitation, NIS shows fast convergence with comparable computational cost for hyperparameter search. NIS has a slightly higher cost owing to machine learning based velocity prediction in comparison with nonmachine learning based operations in FA and PSO. These extra costs are only incurred during the hyperparameter search in the training stage for each search method.

### *E. NIS Evaluation Using Benchmark Test Functions*

To validate the contributions of the Discrete Adaptive Wave Function scheduling and exploration/exploitation behaviors of NIS, we evaluate it against a total of 12 search methods in solving mathematical test functions, including PSO, FA [53], Random Search (RA) [54], Memetic Algorithm (MA) [55], Jaya [56], Gravitational Search Algorithm (GSA) [57], Genetic Algorithm (GA) [58], Flower Pollination Algorithm (FPA) [59], Cuckoo Search (CS) [60], Arithmetic Optimization Algorithm (AOA) [61], Adaptive Random Search (ARS) [62], and Autonomous Particle Groups PSO (AGPSO) [63].

A total of nine benchmark functions are utilized, i.e., Ackley, Dixon Price, Powell, Rastrigin, Rosenbrock, Rotated Hyper-Ellipsoid, Sphere, Sum Squares, and Zakharov. Each benchmark function provides a unique challenge for optimization algorithms by defining various shapes intended to trap the algorithm in local optima. The details of these numerical optimization functions are provided in Tan et al. [1]. We adopt the following experimental configurations for evaluating these test functions, i.e. 50 particles, 500 iterations and 30 dimensions. Each algorithm is executed for 30 trials. The mean results are summarized in Table XII.

The results from Table XII indicate that NIS performs better than all other methods for all the benchmark functions. This further indicates the strength of the three behaviors and the Discrete Adaptive Wave scheduling scheme. In particular, the effectiveness is highlighted by the performance of NIS as compared with those of PSO and AGPSO. To ensure statistical validity of these results, the Wilcoxon rank sum test is performed, and the results are presented in Table XIII. The statistical test results confirm that the $p$-value is smaller than 0.05 for nearly all test functions. The only exception is the Rastrigin function, where NIS and AGPSO achieve statistically similar results.

*1) Ablation Studies:* Ablation studies have been conducted to indicate the effectiveness of each proposed strategy in NIS using the benchmark functions. NIS embeds four operations, i.e.: 1) maximized standard deviation velocity prediction for

TABLE XII
MEAN RESULTS FOR BENCHMARK FUNCTIONS OVER 30 RUNS WITH DIMENSION = 30

| | RA | MA | Jaya | GSA | GA | FPA | CS | AOA | ARS | PSO | FA | AGPSO | NIS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Ackley | 1.82E+01 | 1.73E+01 | 1.91E+01 | 1.61E+01 | 1.91E+01 | 1.59E+01 | 1.88E+01 | 1.96E+01 | 1.78E+01 | 1.81E+01 | 1.96E+01 | 1.54E+01 | **5.39E+00** |
| DixonPrice | 6.26E+05 | 3.13E+04 | 1.42E+06 | 1.69E+06 | 9.34E+04 | 6.28E+04 | 8.50E+05 | 1.62E+06 | 5.56E+05 | 4.13E+03 | 1.64E+06 | 5.26E+02 | **2.37E+01** |
| Powell | 7.78E+08 | 8.78E+08 | 1.44E+12 | 1.60E+13 | 1.60E-02 | 1.41E+06 | 3.04E+10 | 3.18E+13 | 5.36E+08 | 1.24E+03 | 2.97E+12 | 1.99E+01 | **2.78E-05** |
| Rastrigin | 3.37E+02 | 3.40E+02 | 4.01E+02 | 2.10E+02 | 3.80E+02 | 2.93E+02 | 3.61E+02 | 4.29E+02 | 3.31E+02 | 1.73E+02 | 4.22E+02 | 1.11E+02 | **7.16E+01** |
| Rosenbrock | 4.29E+05 | 8.25E+04 | 1.02E+06 | 1.43E+06 | 5.29E+04 | 5.50E+04 | 7.70E+05 | 1.32E+06 | 3.85E+05 | 1.66E+05 | 1.45E+06 | 9.22E+04 | **2.29E+02** |
| Rotated Hyper Ellipsoid | 2.23E+05 | 6.39E+04 | 3.47E+05 | 3.98E+05 | 2.66E+05 | 5.91E+04 | 2.54E+05 | 3.97E+05 | 2.12E+05 | 2.04E+04 | 3.89E+05 | 7.85E+03 | **6.06E+02** |
| Sphere | 1.03E+02 | 1.05E+02 | 1.51E+02 | 7.69E+00 | 1.40E+02 | 2.60E+01 | 1.19E+02 | 1.73E+02 | 1.01E+02 | 9.20E+00 | 1.71E+02 | 5.65E+00 | **2.09E-01** |
| SumSquares | 1.43E+03 | 1.30E+03 | 2.23E+03 | 7.88E+01 | 1.56E+03 | 3.59E+02 | 1.59E+03 | 2.43E+03 | 1.35E+03 | 1.34E+02 | 2.40E+03 | 7.60E+01 | **1.78E+00** |
| Zakharov | 4.12E+02 | 2.44E+02 | 7.18E+02 | 8.35E+08 | 3.59E+02 | 4.24E+02 | 5.39E+02 | 4.91E+08 | 3.71E+02 | 2.30E+08 | 5.11E+07 | 2.20E+02 | **6.71E+01** |

TABLE XIII
WILCOXON RANK SUM TEST RESULTS FOR THE BENCHMARK FUNCTIONS OVER 30 RUNS

| | Ackley | DixonPrice | Powell | Rastrigin | Rosenbrock | RotHyp | Sphere | SumSquares | Zakharov |
|---|---|---|---|---|---|---|---|---|---|
| RA | 1.07E-07 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 |
| MA | 1.07E-07 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 |
| Jaya | 9.06E-08 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 |
| GSA | 1.07E-07 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 |
| GA | 8.35E-08 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 |
| FPA | 1.07E-07 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 |
| CS | 1.07E-07 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 |
| AOA | 1.55E-09 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 |
| ARS | 1.07E-07 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 |
| PSO | 1.07E-07 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 |
| FA | 4.18E-09 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 |
| AGPSO | 1.07E-07 | 3.02E-11 | 3.02E-11 | **7.73E-02** | 3.02E-11 | 3.02E-11 | 3.02E-11 | 3.02E-11 | 7.22E-06 |

TABLE XIV
MEAN RESULTS OF ABLATION STUDIES ON BENCHMARK FUNCTIONS OVER 30 RUNS WITH DIMENSION = 30

| | M1 (1) | M2 (1+2) | M3 (1+2+3) | M4 (1+2+3+4) |
|---|---|---|---|---|
| Ackley | 1.88E+01 | 1.90E+01 | 1.12E+01 | 5.39E+00 |
| DixonPrice | 5.32E+05 | 4.51E+05 | 5.63E+02 | 2.37E+01 |
| Powell | 3.97E+12 | 2.17E+12 | 1.42E-01 | 2.78E-05 |
| Rastrigin | 3.89E+02 | 3.71E+02 | 2.81E+02 | 7.16E+01 |
| Rosenbrock | 1.02E+06 | 1.02E+06 | 1.03E+03 | 2.29E+02 |
| RotatedHyperEllipsoid | 1.81E+05 | 1.77E+05 | 5.92E+04 | 6.06E+02 |
| Sphere | 1.60E+02 | 1.49E+02 | 5.98E-01 | 2.09E-01 |
| SumSquares | 2.22E+03 | 2.21E+03 | 2.49E+01 | 1.78E+00 |
| Zakharov | 3.75E+06 | 7.71E+02 | 1.87E+02 | 6.71E+01 |

global exploration; 2) local best velocity prediction for search diversification; 3) $n$-Dimensional Whirlpool search for exploiting optimal regions; and 4) the Staged Discrete Adaptive Wave formula for adjusting effects of the above methods. We implement four versions of the model with increasing numbers of operations, i.e., Model 1 (Operation 1), Model 2 (Operations 1 and 2), Model 3 (Operations 1–3), and Model 4 (all four operations).

These four models are tested using the above nine benchmark functions. Table XIV shows that all four models improve test function results incrementally in the majority of the test cases. Model 1 uses LSTM-CNN to maximize standard deviation velocity prediction and shows great efficiency in exploring the search space, but with limited capabilities in fine-tuning optimal solutions. Model 2 improves the search by adding LSTM-CNN-based local best velocity prediction, to increase search diversification while better exploiting local optimal regions. Model 3 further enhances local exploitation by adding an angle-driven whirlpool search mechanism to fine-tune the global best solution. Model 4 uses a Staged Discrete Adaptive Wave formula to emphasize local and global search behaviors adaptively and achieves the best performance.

## V. CONCLUSION

This research has proposed a novel NIS algorithm for optimizing learning and multiloss parameters for segmentation models. Three new search behaviors have been formulated, i.e. LSTM-CNN-based Maximized Standard Deviation and Local Best Velocity Prediction, as well as $n$-dimensional angle rotation, to improve search diversity. A Staged Discrete Adaptive Wave function has also been exploited for implementing a stage-based behavior scheduling to precisely balance the contribution of each behavior pertaining to intensification and diversification during the course of hyperparameter search.

The empirical results comparing NIS against PSO and FA on all five semantic segmentation datasets, in particular for ADE20K and Cityscapes, reveal the effectiveness of the proposed novel behaviors and new operation scheduling regimen introduced in NIS. This has been further confirmed by the notable performance of NIS against 12 well-known and modern search methods on several multimodal and unimodal benchmark functions. The aforementioned novel neural network-based velocity prediction, angle-driven whirlpool search, and the adaptive wave scheduling function work in tandem to further enhance performance.

Future work will investigate architecture generation for transformer networks using the NIS algorithm. Hybrid architectures embedding CNN, LSTM, and transformer models will be exploited for semantic segmentation and other vision and signal processing tasks.

## REFERENCES


[1] T. Y. Tan, L. Zhang, and C. P. Lim, "Adaptive melanoma diagnosis using evolving clustering, ensemble and deep neural networks," *Knowl.-Based Syst.*, vol. 187, Jan. 2020, Art. no. 104807.

[2] T. Fan, G. Wang, Y. Li, and H. Wang, "MA-Net: A multi-scale attention network for liver and tumor segmentation," *IEEE Access*, vol. 8, pp. 179656–179665, 2020.

[3] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017, *arXiv:1706.05587*.

[4] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.

[5] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "UNet++: A nested U-Net architecture for medical image segmentation," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. Cham, Switzerland: Springer, 2018, pp. 3–11.

[6] A. Howard et al., "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1314–1324.

[7] Y. Zhang, O. Morel, M. Blanchon, R. Seulin, M. Rastgoo, and D. Sidibé, "Exploration of deep learning-based multimodal fusion for semantic road scene segmentation," in *Proc. VISIGRAPP*, 2019, pp. 336–343.

[8] D. Saire and A. R. Rivera, "Empirical study of multi-task hourglass model for semantic segmentation task," *IEEE Access*, vol. 9, pp. 80654–80670, 2021.

[9] M. A. Islam, M. Rochan, N. D. B. Bruce, and Y. Wang, "Gated feedback refinement network for dense image labeling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4877–4885.

[10] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional DenseNets for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 1175–1183.

[11] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder–decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.

[12] Z. Li, Y. Sun, L. Zhang, and J. Tang, "CTNet: Context-based tandem network for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 9904–9917, Dec. 2022.

[13] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, "Masked-attention mask transformer for universal image segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 1280–1289.

[14] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, "Segmenter: Transformer for semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 7242–7252.

[15] Y. Sun and Z. Li, "SSA: Semantic structure aware inference for weakly pixel-wise dense predictions without cost," 2021, *arXiv:2111.03392*.

[16] Y. Sun et al., "Singular value fine-tuning: Few-shot segmentation requires few-parameters fine-tuning," 2022, *arXiv:2206.06122*.

[17] H. Du, J. Wang, M. Liu, Y. Wang, and E. Meijering, "SwinPA-Net: Swin transformer-based multiscale feature pyramid aggregation network for medical image segmentation," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Sep. 19, 2022, doi: 10.1109/TNNLS.2022.3204090.

[18] Y. Zhou, G. G. Yen, and Z. Yi, "Evolutionary compression of deep neural networks for biomedical image segmentation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 8, pp. 2916–2929, Aug. 2020.

[19] X. Li, L. Yu, H. Chen, C. Fu, L. Xing, and P. Heng, "Transformation-consistent self-ensembling model for semisupervised medical image segmentation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 2, pp. 523–534, Feb. 2021.

[20] D. Konar, S. Bhattacharyya, B. K. Panigrahi, and E. C. Behrman, "Qutrit-inspired fully self-supervised shallow quantum learning network for brain tumor segmentation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 11, pp. 6331–6345, Nov. 2022.

[21] R. Wang, C. Ji, Y. Zhang, and Y. Li, "Focus, fusion, and rectify: Context-aware learning for COVID-19 lung infection segmentation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 1, pp. 12–24, Jan. 2022.

[22] P. Hu, J.-S. Pan, S.-C. Chu, and C. Sun, "Multi-surrogate assisted binary particle swarm optimization algorithm and its application for feature selection," *Appl. Soft Comput.*, vol. 121, May 2022, Art. no. 108736.

[23] S. Slade, L. Zhang, Y. Yu, and C. P. Lim, "An evolving ensemble model of multi-stream convolutional neural networks for human action recognition in still images," *Neural Comput. Appl.*, vol. 34, no. 11, pp. 9205–9231, Jun. 2022.

[24] L. Zhang, C. P. Lim, Y. Yu, and M. Jiang, "Sound classification using evolving ensemble models and particle swarm optimization," *Appl. Soft Comput.*, vol. 116, Feb. 2022, Art. no. 108322.

[25] L. Zhang, C. P. Lim, and Y. Yu, "Intelligent human action recognition using an ensemble model of evolving deep networks with swarm-based optimization," *Knowl.-Based Syst.*, vol. 220, May 2021, Art. no. 106918.

[26] T. Lawrence, L. Zhang, K. Rogage, and C. P. Lim, "Evolving deep architecture generation with residual connections for image classification using particle swarm optimization," *Sensors*, vol. 21, no. 23, p. 7936, Nov. 2021.

[27] L. Zhang and L. Zhao, "High-quality face image generation using particle swarm optimization-based generative adversarial networks," *Future Gener. Comput. Syst.*, vol. 122, pp. 98–104, Sep. 2021.

[28] X. Zhang, S. Xia, X. Li, and T. Zhang, "Multi-objective particle swarm optimization with multi-mode collaboration based on reinforcement learning for path planning of unmanned air vehicles," *Knowl.-Based Syst.*, vol. 250, Aug. 2022, Art. no. 109075.

[29] A. Chaurasia and E. Culurciello, "LinkNet: Exploiting encoder representations for efficient semantic segmentation," in *Proc. IEEE Vis. Commun. Image Process. (VCIP)*, Dec. 2017, pp. 1–4.

[30] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6230–6239.

[31] M. Abdullah, M. M. Fraz, and S. A. Barman, "Localization and segmentation of optic disc in retinal images using circular Hough transform and grow-cut algorithm," *PeerJ*, vol. 4, May 2016, Art. no. e2003.

[32] S. Morales, V. Naranjo, J. Angulo, and M. Alcañiz, "Automatic detection of optic disc based on PCA and mathematical morphology," *IEEE Trans. Med. Imag.*, vol. 32, no. 4, pp. 786–796, Apr. 2013.

[33] J. R. H. Kumar, A. K. Pediredla, and C. S. Seelamantula, "Active discs for automated optic disc segmentation," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Dec. 2015, pp. 225–229.

[34] Z. U. Rehman, S. S. Naqvi, T. M. Khan, M. Arsalan, M. A. Khan, and M. A. Khalil, "Multi-parametric optic disc segmentation using superpixel based feature classification," *Expert Syst. Appl.*, vol. 120, pp. 461–473, Apr. 2019.

[35] M. N. Zahoor and M. M. Fraz, "A correction to the article 'fast optic disc segmentation in retina using polar transform,'" *IEEE Access*, vol. 6, pp. 4845–4849, 2018.

[36] Z. Fan et al., "Optic disk detection in fundus image based on structured learning," *IEEE J. Biomed. Health Informat.*, vol. 22, no. 1, pp. 224–234, Jan. 2018.

[37] A. Kundu, V. Vineet, and V. Koltun, "Feature space optimization for semantic video segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3168–3175.

[38] M. A. Islam, S. Naha, M. Rochan, N. Bruce, and Y. Wang, "Label refinement network for coarse-to-fine semantic segmentation," 2017, *arXiv:1703.00551*.

[39] G. L. Oliveira, W. Burgard, and T. Brox, "DPDB-Net: Exploiting dense connections for convolutional encoders," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 4525–4531.

[40] P. Hu, F. Caba, O. Wang, Z. Lin, S. Sclaroff, and F. Perazzi, "Temporally distributed networks for fast video semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 8815–8824.

[41] A. Sagar and R. Soundrapandiyan, "Semantic segmentation with multi scale spatial attention for self driving cars," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2021, pp. 2650–2656.

[42] Z. Jin, B. Liu, Q. Chu, and N. Yu, "ISNet: Integrate image-level and semantic-level context for semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 7169–7178.

[43] H. Hu, J. Cui, and L. Wang, "Region-aware contrastive learning for semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 16271–16281.

[44] W. Zhang et al., "TopFormer: Token pyramid transformer for mobile semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 12073–12083.

[45] J. Gu et al., "Multi-scale high-resolution vision transformer for semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 12084–12093.

[46] M.-H. Guo, C.-Z. Lu, Q. Hou, Z. Liu, M.-M. Cheng, and S.-M. Hu, "SegNeXt: Rethinking convolutional attention design for semantic segmentation," 2022, *arXiv:2209.08575*.

[47] X. Dong et al., "CSWin transformer: A general vision transformer backbone with cross-shaped windows," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 12114–12124.

[48] S. Huang, Z. Lu, R. Cheng, and C. He, "FaPN: Feature-aligned pyramid network for dense image prediction," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 844–853.

[49] S. Zheng et al., "Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 6877–6886.

[50] Y. Wang, C. Ho, W. Xu, Z. Xuan, X. Liu, and G.-J. Qi, "Dual-flattening transformers through decomposed row and column queries for semantic segmentation," 2022, *arXiv:2201.09139*.

[51] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "SegFormer: Simple and efficient design for semantic segmentation with transformers," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 12077–12090.

[52] Y. Jin, D. Han, and H. Ko, "TrSeg: Transformer for semantic segmentation," *Pattern Recognit. Lett.*, vol. 148, pp. 29–35, Aug. 2021.

[53] X.-S. Yang and X. He, "Firefly algorithm: Recent advances and applications," 2013, *arXiv:1308.3898*.

[54] Z. B. Zabinsky et al., "Random search algorithms," Dept. Ind. Syst. Eng., Univ. Washington, Seattle, WA, USA, Tech. Rep., 2009.

[55] F. Neri and C. Cotta, "Memetic algorithms and memetic computing optimization: A literature review," *Swarm Evol. Comput.*, vol. 2, pp. 1–14, Feb. 2012.

[56] E. H. Houssein, A. G. Gad, and Y. M. Wazery, "Jaya algorithm and applications: A comprehensive review," in *Metaheuristics and Optimization in Computer and Electrical Engineering*. Cham, Switzerland: Springer, 2021, pp. 3–24.

[57] M. Khajehzadeh and M. Eslami, "Gravitational search algorithm for optimization of retaining structures," *Indian J. Sci. Technol.*, vol. 5, no. 1, pp. 1821–1827, 2012.

[58] S. N. Sivanandam and S. N. Deepa, "Genetic algorithm optimization problems," in *Introduction to Genetic Algorithms*. Cham, Switzerland: Springer, 2008, pp. 165–209.

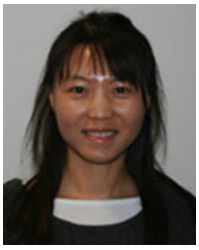
[59] X.-S. Yang, "Flower pollination algorithm for global optimization," in *Proc. Int. Conf. Unconventional Comput. Natural Comput.* Berlin, Germany: Springer, 2012, pp. 240–249.

[60] M. Mareli and B. Twala, "An adaptive cuckoo search algorithm for optimisation," *Appl. Comput. Informat.*, vol. 14, no. 2, pp. 107–115, Jul. 2018.

[61] L. Abualigah, A. Diabat, S. Mirjalili, M. A. Elaziz, and A. H. Gandomi, "The arithmetic optimization algorithm," *Comput. Methods Appl. Mech. Eng.*, vol. 376, Apr. 2021, Art. no. 113609.
[62] S. Andradóttir and A. A. Prudius, "Adaptive random search for continuous simulation optimization," *Nav. Res. Logistics (NRL)*, vol. 57, no. 6, pp. 583–604, Sep. 2010.
[63] S. Mirjalili, A. Lewis, and A. S. Sadiq, "Autonomous particles groups for particle swarm optimization," *Arabian J. Sci. Eng.*, vol. 39, no. 6, pp. 4683–4697, Jun. 2014.

**Sam Slade** received the B.Sc. and M.Sc. degrees in computer science from Northumbria University, Newcastle upon Tyne, U.K., in 2016 and 2017, respectively, where he is currently pursuing the Ph.D. degree.

His research interests include deep learning, computer vision, and evolutionary computation.



**Li Zhang** (Senior Member, IEEE) received the Ph.D. degree from the University of Birmingham, Birmingham, U.K., in 2004.

She is currently a Reader with the Department of Computer Science, Royal Holloway, University of London, Surrey, U.K. She holds expertise in deep learning, intelligent robotics, and image processing.

Dr. Zhang has served as an Associate Editor for *Decision Support Systems*.

**Haoqian Huang** (Member, IEEE) received the Ph.D. degree from Southeast University, Nanjing, China, in 2015.

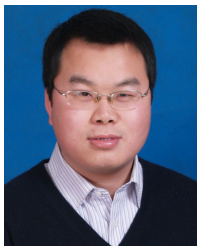
He is currently an Associate Professor with the College of Energy and Electrical Engineering, Hohai University, Nanjing. His main research interests include navigation technology applied to underwater vehicle, inertial navigation, and information fusion.

**Houshyar Asadi** (Member, IEEE) received the Ph.D. degree in human perception-based washout filtering using artificial intelligence (AI) from the Institute for Intelligent Systems Research and Innovation (IISRI), Deakin University, Waurn Ponds, VIC, Australia, in 2015.

He is currently an Associate Professor with IISRI. His current research interests include AI, motion-based simulator technologies, robotics, control, and human factors.

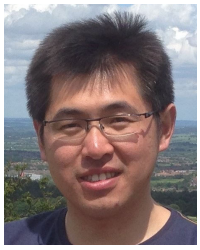
**Chee Peng Lim** received the B.Eng. degree in electrical engineering from the University of Technology Malaysia, Johor Bahru, Malaysia, in 1992, and the M.Sc. degree in control systems and the Ph.D. degree from the University of Sheffield, Sheffield, U.K., in 1993 and 1997, respectively.

He is currently a Professor with Deakin University, Waurn Ponds, VIC, Australia, and has published more than 500 technical papers in books, international journals, and conference proceedings. His research interests include soft computing, pattern recognition, medical prognosis and diagnosis, fault detection and diagnosis, and condition monitoring.



**Yonghong Yu** received the Ph.D. degree in computer science from Nanjing University, Nanjing, China, in 2017.

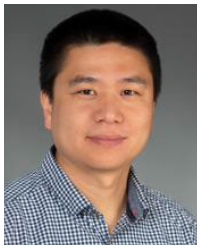
He is currently an Associate Professor with the TongDa College, Nanjing University of Posts and Telecommunications, Nanjing. His main research interests include machine learning, recommender systems, and image processing.



**Dezong Zhao** (Senior Member, IEEE) received the B.Eng. and M.S. degrees from Shandong University, Jinan, China, in 2003 and 2006, respectively, and the Ph.D. degree in control engineering from Tsinghua University, Beijing, China, in 2010.

Since 2020, he has been a Senior Lecturer in autonomous systems with the University of Glasgow, Glasgow, U.K. His research interests include connected and autonomous vehicles, robotics, machine learning, and control engineering.

Dr. Zhao has been an Engineering and Physical Sciences Research Council (EPSRC) Innovation Fellow since 2018 and a Royal Society-Newton Advanced Fellow since 2020.



**Hanhe Lin** received the Ph.D. degree from the Department of Information Science, University of Otago, Dunedin, New Zealand, in 2016.

From 2016 to 2021, he was a Post-Doctoral Researcher with the Department of Computer and Information Science, University of Konstanz, Konstanz, Germany, where he was working on project A05 (visual quality assessment) of SFB-TRR 161, funded by the German Research Foundation (DFG). Currently, he is a Lecturer in computing with the University of Dundee, Dundee, U.K. His research interests include visual quality assessment, computer vision, machine learning, and deep learning.

**Rong Gao** received the Ph.D. degree from Wuhan University, Wuhan, China, in 2018.

He is currently an Assistant Professor with the School of Computer Science, Hubei University of Technology, Wuhan. His research interests include machine learning and image processing.