GA-CDAM: Genetic Algorithm Driven Compact Damage Assessment Model Using Post-Disaster Aerial Images

Iyed Dhahri*, Abdennour Azerine*, Mahmoud Golabi*, Karim Hammoudi*, and Lhassane Idoumghar*
* IRIMAS, University of Haute-Alsace, Mulhouse, France
Emails: {iyed.dhahri, abdennour.azerine, mahmoud.golabi, karim.hammoudi, lhassane.idoumghar}@uha.fr

***Abstract*—This article discusses the automated creation of a segmentation model from post-disaster aerial images for disaster analysis. Disaster damage assessment models must be efficient and accurate in order to provide relevant information on affected areas quickly. Manual design of such architectures is time-consuming and may not yield optimal results. To address this, we present a genetic algorithm that explores different components of UNet architectures to automatically design the best configuration. The goal is to maximize the accuracy of the segmentation model, which is the Mean Intersection over Union (MIoU) under explicit computational constraints. The algorithm explores a vast search space that contains various UNet architectural decisions (e.g., network depth, convolution type, oversampling strategies). It uses a fixed gene to generate a dynamic phenotype that serves as a neural network for the segmentation task. The results showed a significant reduction in model complexity, from over 40 million parameters for the State-Of-The-Art (SOTA) models to just 3.72 million parameters for our model. While maintaining good segmentation results, reaching 67.45% mIoU. We have successfully automated the design of the disaster damage assessment model by leveraging metaheuristic optimization and have provided a lightweight model that can be deployed in small devices for real-time disaster analysis.**

***Index Terms*—Genetic Algorithm, Deep Learning, Damage Assessment, Natural Disaster.**

I. Introduction

Nowadays, natural disasters are more frequent due to climate change [1], posing the challenge of preparing before they occur and responding afterwards to save lives [2]. These disasters cost a huge number of human lives and have a considerable impact on economies. In 2024, according to global disaster data, natural disasters killed more than 16,000 people worldwide and affected more than 160 million people, who were injured or displaced [3]. We focused on post-disaster response, where the lack of accurate information on the ground slows down relief efforts, resulting in a higher number of casualties [4]. This is why aerial images taken after a disaster are increasingly being used to reduce this gap [5]. However, manual interpretation of these images is time-consuming, which is why deep learning models are being used to automate this task and make it faster [6].

In recent years, numerous disaster images datasets have been published for various tasks, ranging from binary classification (does the image represent a damaged area or not?) to disaster type classification [7], but the most informative task is segmentation, which involves detecting the different environmental elements present in the image and quantifying the damage [8]. RescueNet was presented as a semantic segmentation dataset on images captured by UAVs after Hurricane Harvey in Florida, USA [9]. It annotates buildings with different levels of damage as well as other environmental elements (e.g., blocked roads, clear roads, vehicles). The reference models were based on UNet [10]. A Pyramid Pooling module was used to capture the different scales of objects such as buildings or vehicles, which can vary in size and shape [11]. Attention mechanisms were also used to capture fine details that allow for accurate segmentation of the contours of each object [12]. Transformer-based models were also trained on this dataset [13]. They are useful for capturing both the global and local context of the scene, as the model examines the image all at once, unlike CNN-based models that examine the image through small windows.

Many other architectures have been proposed with slight modifications for this dataset. The main challenge lies in the fact that manually designing such architectures is time-consuming, with no guarantee that the designed model will achieve better results [14]. In addition, most of these models use very resource-intensive base structures, such as ResNet50 [15], which slows down their training and evaluation [16]. Furthermore, in practice, these models can be deployed on small computing devices, such as drones, for real-time disaster analysis [17]. It is therefore necessary to have less complex models.

Metaheuristic optimization algorithms were applied to the tuning of machine learning hyperparameters by exploring different compositions in order to find the best solution based on a fixed objective [18]. For example, a population-based genetic algorithm was used to explore the hyperparameters of a PSPNet model in order to find the best configuration for improving the segmentation of certain classes in post-disaster images [19]. Another study applied GA to build a block-by-block UNet model for medical image classification [20]. However, it is still used with a limited search space that does not allow enough freedom to test more architectures or explore the components of each block.

Given the importance of damage assessment models using post-disaster images, we focused on improving segmentation results on the RescueNet dataset only for critical classes

directly related to rescue efforts. These are primarily buildings with varying levels of damage where the chances of finding victims are higher, as well as roads and vehicles useful for planning rescue efforts. Instead of designing the segmentation model architecture, we propose a GA-CDAM framework that uses a genetic algorithm [21] to build a damage assessment model from scratch. It explores a big search space that contains various architectural decisions (e.g., number of layers, filter sizes, sampling methods) to maximize the model's MIoU on the selected classes.

The results are compared to those of SOTA models. Our approach permits to build a model that is much less complex than the others, even though there is a trade-off between model complexity and accuracy. The discrepancy is acceptable, as it is possible to deploy these models on low-capacity devices.

II. Problem Definition

We consider the problem of automatically designing lightweight encoder-decoder neural architectures for semantic segmentation under explicit computational constraints. Let $\mathcal{A}$ denote the space of candidate architectures, where each architecture $a \in \mathcal{A}$ corresponds to a fully specified encoder-decoder network topology, including depth, width, skip connections, and segmentation head configuration.

Each architecture is evaluated by training it on a fixed dataset and measuring its segmentation performance on a held-out validation set. The objective is to identify an architecture that maximizes segmentation accuracy while remaining suitable for deployment on resource-constrained platforms, such as aerial or embedded systems commonly used in disaster response scenarios.

Formally, the problem can be expressed as a constrained optimization task:

$$\begin{aligned} a^* &= \arg\max_{a \in \mathcal{A}} \ \mathrm{mIoU}(a) \\ \text{subject to} \quad & \Theta(a) \leq \Theta_{\max}, \Phi(a) \leq \Phi_{\max} \end{aligned} \tag{1}$$

where $\mathrm{mIoU}(a)$ denotes the mean Intersection-over-Union achieved by architecture a on the validation set, $\Theta(a)$ represents the total number of learnable parameters bounded by $\Theta_{\max}$, and $\Phi(a)$ denotes the floating-point operations per forward pass bounded by $\Phi_{\max}$. These constraints ensure that discovered architectures remain suitable for deployment on resource-constrained platforms commonly encountered in emergency response scenarios.

III. Methodology

To solve the constrained architecture search problem defined in Section II, we propose GA-CDAM, a genetic algorithm-based neural architecture search framework for encoder-decoder semantic segmentation models. Candidate architectures are encoded as fixed-length discrete genomes and evolved using selection, crossover, mutation, and elitism. Fitness is computed from validation mIoU after training under explicit parameters and FLOP budgets. The following subsections detail the genome encoding and decoding process, genetic operators, fitness evaluation protocol, and termination criteria.

TABLE I: Genome Structure and Search Space

Gene	Parameter	Options
Global Architecture Parameters		
g_0	Number of stages S	3, 4
g_1	Base filters F_0	16, 24, 32
g_2	Skip connection type	concat, add
g_3	Decoder block type	simple, residual
Stage-Specific Parameters (per stage $s = 0, \ldots, S-1$)		
g_{4+10s}	Encoder blocks per stage	1, 2
g_{5+10s}	Convolution kernel size	3, 5
g_{7+10s}	Channel expansion factor	1.0, 1.5
g_{10+10s}	Encoder activation function	ReLU, LeakyReLU, GELU
g_{11+10s}	Decoder blocks per stage	1, 2
g_{12+10s}	Upsampling method	bilinear, transpose
g_{13+10s}	Decoder activation function	ReLU, LeakyReLU, GELU
Head and Training Parameters		
g_{49}	Segmentation head type	conv1×1, ASPP
g_{50}	Auxiliary loss enabled	false, true
g_{51}	Max training epochs	20, 30, 40, 50

A. Genome Representation

The genome encoding scheme partitions architectural choices into three categories: global parameters governing the overall network structure, stage-specific parameters defining layer configurations within each encoder-decoder stage, and head parameters specifying the final segmentation module. The complete genome comprises $L = 52$ genes organized as summarized in Table I.

1) Global Architecture Parameters: The first four genes encode high-level structural decisions. Gene g_0 determines the number of encoder-decoder stages $S \in \{3, 4\}$, controlling network depth and receptive field. Gene g_1 specifies the base filter count $F_0 \in \{16, 24, 32\}$, with subsequent stages following a doubling progression $F_s = F_0 \cdot 2^s$ to capture increasingly abstract features. Gene g_2 selects the skip connection type between encoder and decoder stages, with options including channel-wise concatenation and element-wise addition. Gene g_3 determines the decoder block style, choosing between standard and residual configurations.

2) Stage-Specific Parameters: Each of the S encoder-decoder stages is governed by a set of genes controlling both encoding and decoding operations. For the encoder portion, genes specify the number of convolutional blocks per stage $B_s^{\text{enc}} \in \{1, 2\}$, determining the representational capacity at each resolution level. The kernel size gene selects between $K_s \in \{3, 5\}$, trading off between local feature extraction and expanded receptive fields. The channel expansion factor $E_s \in \{1.0, 1.5\}$ controls the output channel multiplier, enabling the search to discover architectures with varying width profiles across stages. The activation function gene selects among ReLU, Leaky ReLU, and GELU, allowing the evolutionary process to identify activation characteristics best suited to different network depths. Batch normalization is employed throughout to stabilize training dynamics.

For the decoder portion, genes determine the number of refinement blocks $B_s^{\text{dec}} \in \{1, 2\}$ applied after feature fusion, the upsampling method selecting between bilinear interpolation and learned transposed convolution, and the decoder activation

function which may differ from the corresponding encoder stage. This stage-wise parameterization enables the genetic algorithm to discover architectures with heterogeneous configurations across depth levels, reflecting the observation that optimal design choices for high-resolution early stages may differ substantially from those appropriate for semantically-rich deeper representations.

3) Segmentation Head and Training Parameters: Gene g_{49} governs the segmentation head module that produces the final per-pixel class predictions. The search space includes a simple 1×1 convolutional classifier offering minimal computational overhead, and an Atrous Spatial Pyramid Pooling (ASPP) module that aggregates multi-scale contextual information through parallel atrous convolutions operating at dilation rates of 1, 6, and 12, concatenated with a global average pooling branch to incorporate image-level context. Gene g_{50} enables or disables deep supervision through an auxiliary classification head attached at an intermediate decoder stage, a technique that has been shown to improve gradient flow and accelerate convergence in deep encoder-decoder networks. Gene g_{51} specifies the maximum training duration $E_{\max} \in \{20, 30, 40, 50\}$ epochs, incorporating this critical training hyperparameter into the genome such that the evolutionary process jointly optimizes both architecture and training configuration.

B. Network Architecture

The decoded genome specifies an encoder-decoder network following the established U-shaped topology that has proven effective for dense prediction tasks. The encoder comprises S stages operating at progressively reduced spatial resolutions, with each stage containing B_s^{enc} convolutional blocks followed by max pooling for spatial downsampling by a factor of two. Each convolutional block applies a convolution operation with the genome-specified kernel size, followed by batch normalization and the selected activation function. Skip connections transmit encoder feature maps to corresponding decoder stages at matching resolutions, enabling the decoder to leverage fine-grained spatial information that would otherwise be lost through the encoding bottleneck.

The decoder mirrors the encoder structure with S stages, each performing spatial upsampling followed by fusion with skip-connected features and refinement through B_s^{dec} convolutional blocks. When concatenation-based skip connections are employed, encoder and decoder features are stacked along the channel dimension before processing by a fusion convolution. For additive skip connections, a 1×1 convolution projects encoder features to match decoder channel dimensions before element-wise summation.

The segmentation head produces per-pixel class predictions from the final decoder features. The simple head applies a 1×1 convolution directly, while the ASPP head employs parallel 1×1 and 3×3 convolutions at dilation rates of 6 and 12, concatenating the outputs with a global average pooling branch before final projection. If auxiliary supervision is enabled, an additional classification head attached to an intermediate decoder stage provides supplementary gradient signals during training.

C. Genetic Operators

The evolutionary search employs standard genetic operators adapted for the discrete genome representation. The initial population of N individuals is initialized with approximately one-third seeded from validated architectures to exploit prior knowledge, while the remainder is generated via uniform random sampling to ensure diversity. Parent selection utilizes tournament selection ($k = 3$) to balance selection pressure and population diversity. Recombination is performed using uniform crossover with probability $p_c = 0.80$, treating each gene position independently. To explore novel configurations, a per-gene mutation is applied with probability $p_m = 0.20$. Finally, an elitism strategy preserves the top 15% ($\rho = 0.15$) of individuals, alongside the overall best historical individual, ensuring monotonic improvement across generations

D. Fitness Evaluation and Training Protocol

Candidate architectures are evaluated based on their mean Intersection-over-Union (mIoU) on a held-out validation set, which provides a balanced assessment across all semantic categories without being dominated by majority classes. Training utilizes a composite loss function combining Focal Loss ($\gamma = 2.0$) to address class imbalance and Dice Loss to directly optimize segmentation overlap.Models are trained for up to E_{max} epochs using the AdamW optimizer (learning rate $\eta = 0.001$, weight decay $\lambda = 0.01$) with a cosine annealing schedule decaying to 10^{-5}. Early stopping terminates training if the validation metric fails to improve by at least $\delta = 0.005$ over five consecutive epochs. To enforce strict deployment constraints, architectures exceeding the parameter budget $P_{max} = 10^7$ or the computational budget $F_{max} = 10^{10}$ FLOPs are assigned zero fitness and excluded from training.

The evolutionary search incorporates multiple termination criteria to balance search quality against computational cost. The search terminates upon reaching a maximum generation limit G_{max}, if the best fitness stagnates for τ consecutive generations, or if a target threshold f^* is achieved. A minimum generation constraint prevents premature convergence on seeded solutions.

Algorithm 1 presents the complete genetic algorithm procedure, integrating population initialization, fitness evaluation, selection, crossover, mutation, and elitism with the described termination criteria.

IV. Experimental Setup

A. Dataset

Experiments are conducted on a disaster response benchmark comprising aerial imagery annotated with six semantic classes including background, buildings with 4 different levels of damage and roads [9]. Figure is a sample from the dataset, it illustrates the damage caused by the hurricane, with some buildings destroyed and others intact and a road in the middle. The dataset consists of high-resolution images from which

Algorithm 1 Genetic Algorithm for Neural Architecture Search

Require: Population size N, maximum generations $G_{\max}$, crossover rate p_c, mutation rate p_m, elitism ratio ρ, patience τ
Ensure: Best architecture genome $\mathbf{g}^*$

Initialize population $\mathcal{P} \leftarrow \{\mathbf{g}^{(1)}, \ldots, \mathbf{g}^{(N)}\}$ using seeded and random genomes
Evaluate fitness $f(\mathbf{g}^{(i)})$ for all $\mathbf{g}^{(i)} \in \mathcal{P}$
$\mathbf{g}^* \leftarrow \arg\max_{\mathbf{g} \in \mathcal{P}} f(\mathbf{g})$
$f^* \leftarrow f(\mathbf{g}^*)$, stagnation $\leftarrow 0$
for $t = 1$ **to** $G_{\max}$ **do**
    Select parents $\mathcal{M} \leftarrow \text{TournamentSelection}(\mathcal{P}, N, k)$
    Initialize offspring set $\mathcal{O} \leftarrow \emptyset$
    for $i = 1$ **to** $N - 1$ **step** 2 **do**
        $(\mathbf{c}_1, \mathbf{c}_2) \leftarrow \text{UniformCrossover}(\mathcal{M}^{(i)}, \mathcal{M}^{(i+1)}, p_c)$
        $\mathcal{O} \leftarrow \mathcal{O} \cup \{\mathbf{c}_1, \mathbf{c}_2\}$
    end for
    $\mathcal{O} \leftarrow \{\text{Mutate}(\mathbf{g}, p_m) \mid \mathbf{g} \in \mathcal{O}\}$
    Select elite set $\mathcal{E}$ as the top $\lceil \rho N \rceil$ individuals from $\mathcal{P}$
    Ensure $\mathbf{g}^* \in \mathcal{E}$
    Form new population $\mathcal{P}' \leftarrow \mathcal{E} \cup \{\mathcal{O}^{(1)}, \ldots, \mathcal{O}^{(N-|\mathcal{E}|)}\}$
    Evaluate fitness for all individuals in $\mathcal{P}'$
    $\mathcal{P} \leftarrow \mathcal{P}'$
    if $\max_{\mathbf{g} \in \mathcal{P}} f(\mathbf{g}) > f^*$ **then**
        $\mathbf{g}^* \leftarrow \arg\max_{\mathbf{g} \in \mathcal{P}} f(\mathbf{g})$
        $f^* \leftarrow f(\mathbf{g}^*)$
        stagnation $\leftarrow 0$
    else
        stagnation $\leftarrow$ stagnation $+ 1$
    end if
    if stagnation $\geq \tau$ **or** $f^* \geq f_{\text{target}}$ **then**
        break
    end if
end for
return $\mathbf{g}^*$

Fig. 1: Sample of Damage Scene from RescueNet. Buildings with different levels of damage as well as road parts.

overlapping patches are extracted at multiple scales to augment the training corpus. Patches of size 256×256 pixels are extracted with 50% overlap, generating a comprehensive set of training examples that capture diverse scene configurations and class distributions.

Data preprocessing includes resizing to the target resolution, normalization using ImageNet statistics, and application of geometric augmentations including horizontal and vertical flips, random rotation by multiples of 90 degrees, and affine transformations with limited shift and scale perturbations. Class weights are computed from training set statistics using inverse frequency weighting, with extreme values clipped to prevent numerical instability.

TABLE II: Genetic Algorithm Hyperparameters

Parameter	Symbol	Value
Population size	N	30
Maximum generations	$G_{\max}$	100
Tournament size	k	3
Crossover probability	p_c	0.80
Mutation probability	p_m	0.20
Elitism ratio	ρ	0.15
Stagnation patience	τ	20
Minimum generations	–	15
Training epochs	$E_{\max}$	20, 30, 40, 50
Batch size	–	8
Learning rate	η	0.001
Weight decay	λ	0.01
Training patience	–	5

B. Implementation

The evolutionary search and network training were implemented in Python using PyTorch, with all foundational hyperparameters (including genetic operator probabilities, population dynamics, and training schedules) summarized in Table II. To initialize the search, one-third of the population was seeded with pre-validated architectures to exploit prior knowledge, while the remainder was generated randomly to ensure structural diversity. To strictly enforce deployment constraints, any candidate architecture exceeding 10 million parameters or 10 billion FLOPs was assigned zero fitness and immediately rejected. All evaluations and theoretical computational analyses were conducted on a workstation featuring an NVIDIA RTX 4000 ADA GPU, 128 GB RAM, and an Intel Xeon W7-3555 processor.

V. Results and Evaluation

A. Genetic Algorithm Evolution Analysis

The evolutionary search demonstrated consistent fitness improvement across generations, as illustrated in Figure 2. The population mean fitness increased from 0.752 in the initial generation to 0.788 by generation 13, representing an improvement of 4.79%. More notably, the best individual fitness improved from 0.751 to 0.810, corresponding to a 7.84% gain over the search process.

The convergence behavior exhibited characteristics typical of well-tuned evolutionary algorithms [22]. Initial generations showed high fitness variance (standard deviation of 0.089), reflecting the diversity introduced by random initialization. As selection pressure accumulated, the variance decreased to 0.052 by generation 13, indicating population convergence toward favorable regions of the architecture space. The improvement rate was highest during generations 3–7, with diminishing returns observed in later generations as the population approached local optima.

The optimal architecture discovered by the genetic algorithm features four encoder-decoder stages with 32 base

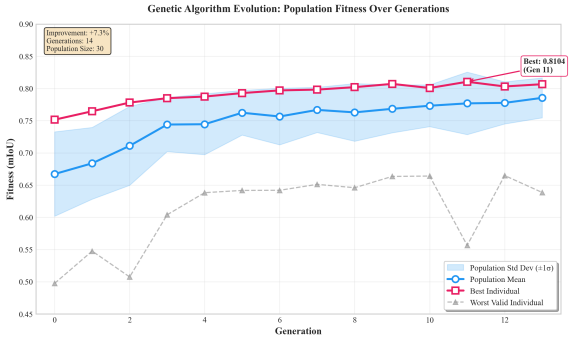

Fig. 2: Evolution of fitness (mIoU) across the first 14 generations of genetic algorithm search. The solid blue line represents population mean fitness with standard deviation bands, while the dashed green line indicates the best individual per generation. The population exhibits steady improvement with decreasing variance, indicating convergence toward high-performing architectures.

filters, concatenation-based skip connections, a bottleneck compression ratio of 0.5, and GELU activation functions. This configuration balances receptive field coverage with parameter efficiency, achieving 3.72 million parameters—a reduction of 92% compared to PSPNet's 46.5 million parameters [23] while maintaining competitive segmentation accuracy.

The best discovered architecture was trained for 336 epochs using the AdamW optimizer [24] with an initial learning rate of 0.0005 and cosine annealing scheduling. Data augmentation including random horizontal flips, rotations, and color jittering was applied to improve generalization. The training employed a combined loss function of focal loss and dice loss to address class imbalance inherent in disaster imagery [25].

Table III presents the quantitative segmentation results on the RescueNet test set. The proposed GA-discovered architecture achieved a mean intersection-over-union (mIoU) of 67.45% and pixel accuracy of 92.47%. These results demonstrate competitive performance relative to manually designed architectures while requiring substantially fewer computational resources.

TABLE III: Segmentation performance comparison on RescueNet test set. The proposed GA-CDAM method achieves competitive accuracy with significantly reduced model complexity, enabling efficient deployment on resource-constrained UAV platforms.

Method	mIoU (%)	Pixel Acc. (%)	Params (M)
RescueNet PSPNet	**91.98**	**94.2**	**46.5**
GA-CDAM (Proposed)	**67.45**	**92.47**	**3.72**

B. Per-Class Segmentation Analysis

Figure 3 presents the per-class IoU breakdown, revealing heterogeneous performance across damage categories. The background class achieved the highest IoU of 92.43%, attributable to its large presence in aerial imagery. Non-damaged buildings attained 82.19% IoU, benefiting from intact structural features that provide clear visual cues.

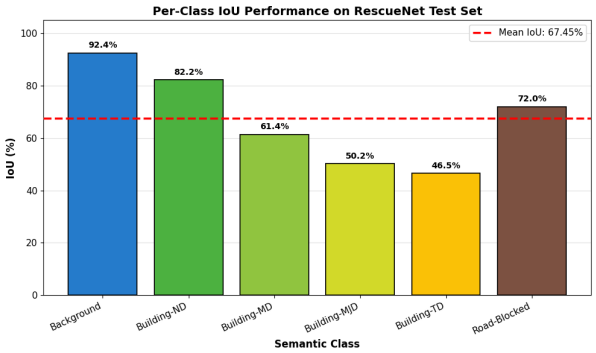

Fig. 3: Per-class intersection-over-union performance on the RescueNet test set. Background and non-damaged buildings achieve highest accuracy, while intermediate damage classes present greater challenge due to visual similarity and class imbalance.

The damage severity classes exhibited progressively lower accuracy, with minor-damaged buildings at 61.40%, major-damaged buildings at 50.25%, and totally-destroyed buildings at 46.55%. This pattern reflects the inherent difficulty of distinguishing intermediate damage states, where structural degradation produces ambiguous visual features. The blocked road class 71.99% IoU benefiting from the presence of distinctive debris patterns.

The class-specific performance highlights opportunities for targeted improvements through class-rebalancing strategies or specialized architectural components for damage assessment. Nevertheless, the proposed architecture provides a solid foundation for automated damage mapping that can inform resource allocation decisions in disaster response scenarios.

C. Computational Efficiency Analysis

The computational efficiency of the proposed GA-CDAM framework offers significant advantages for deployment in time-critical disaster response operations. Table IV compares the inference characteristics of the discovered architecture against established baseline methods. The proposed model achieves 89.3 frames per second on an NVIDIA RTX 3090 GPU at 128×128 input resolution, enabling real-time processing of UAV video streams.

TABLE IV: Computational efficiency comparison at 128×128 input resolution. The proposed architecture achieves superior throughput while maintaining competitive accuracy, making it suitable for real-time UAV deployment.

Method	FLOPs (G)	Params (M)	FPS	Memory (GB)
RescueNet PSPNet	**15.0**	**46.5**	**-**	**-**
GA-CDAM (Proposed)	**1.2**	**3.72**	**89.3**	**0.6**

The memory footprint of 0.6 GB enables deployment on embedded GPU platforms commonly used in autonomous

UAV systems, such as the NVIDIA Jetson series. This capability supports continuous aerial monitoring during disaster response, where persistent surveillance can track damage progression and identify emerging hazards.

In summary, the experimental results validate the effectiveness of genetic algorithm-based neural architecture search for disaster image segmentation. The evolutionary optimization successfully discovered compact architectures that balance accuracy and efficiency, addressing a critical requirement for operational deployment in resource-constrained environments.

The convergence analysis revealed that 14 generations with a population of 30 individuals provided sufficient exploration of the architecture space, with the search process requiring approximately 18 GPU-hours on an RTX 3090.

VI. Conclusion and Future Works

We proposed an GA-based approach to designing segmentation model architectures for post-disaster analysis. We automated the tedious stage of manual design with a two-fold objective to find the best model in terms of segmentation performance, but with constraints on model size. We used images taken by drone after the disaster to perform pixel-by-pixel segmentation. The experimental results show that our GA-CDAM model is much less complex than state-of-the-art models while maintaining good results in detecting the critical classes we selected for disaster analysis. This proves the effectiveness of our approach to designing accurate light segmentation models that can be deployed later on platforms with limited resources.

Our approach is focused on CNN blocks, and given the success of using attention mechanisms and pyramid pooling modules in image analysis, we plan to expand the search space to include these elements, which will allow GA to explore a greater number of architecture combinations.

References

[1] A. Sharma, "The impact of climate change on natural disasters," *International Journal for Multidisciplinary Research (IJFMR)*, vol. 7, no. 1, jan 2025, e-ISSN: 2582-2160. [Online]. Available: https://doi.org/10.36948/ijfmr.2025.v07i01.37719

[2] G. Khaspuria, A. Ranjan, Sahil, P. Soni, and K. Dadhich, "Natural disaster mitigation strategies: A comprehensive review," *Journal of Scientific Research and Reports*, vol. 30, no. 8, p. 20–34, Jul. 2024. [Online]. Available: https://journaljsrr.com/index.php/JSRR/article/view/2221

[3] Centre for Research on the Epidemiology of Disasters (CRED), "2024 disasters in numbers," Centre for Research on the Epidemiology of Disasters (CRED), Brussels, Belgium, Report, 2025. [Online]. Available: https://files.emdat.be/reports/2024_EMDAT_report.pdf

[4] H. J. Scholl, S. Ballard, S. Carnes, A. Herman, and N. Parker, "Informational challenges in early disaster response: The massive Oso/SR530 landslide 2014 as case in point," in *Proceedings of the 50th Hawaii International Conference on System Sciences (HICSS)*, Hawaii, USA, jan 2017. [Online]. Available: http://hdl.handle.net/10125/41458

[5] U. Verma, "Recent trends and challenges in analysis of uav aerial images for post-disaster scene understanding," in *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium*, 2022, pp. 4647–4649.

[6] A. F. Said, V. Kashyap, N. Choudhury, and F. Akhbari, "A cost-effective, fast, and robust annotation tool," in *2017 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, 2017, pp. 1–6.

[7] F. Alam, F. Ofli, and M. Imran, "Crisismmd: Multimodal twitter datasets from natural disasters," in *Proceedings of the 12th International AAAI Conference on Web and Social Media (ICWSM)*, 2018. [Online]. Available: https://crisisnlp.qcri.org/crisismmd

[8] R. Gupta, R. Hosfelt, N. Sajeev, S. B. Patel, B. Goodman, and R. Doshi, "xbd: A dataset for assessing building damage from satellite imagery," *arXiv preprint arXiv:1911.09296*, 2019. [Online]. Available: https://arxiv.org/abs/1911.09296

[9] M. Rahnemoonfar, T. Chowdhury, and R. Murphy, "Rescuenet: A high resolution uav semantic segmentation dataset for natural disaster damage assessment," *Scientific Data*, vol. 10, p. 913, 2023.

[10] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

[11] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2881–2890.

[12] O. Oktay, J. Schlemper, L. Le Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, S. McDonagh, N. Y. Hammerla, B. Kainz *et al.*, "Attention u-net: Learning where to look for the pancreas," in *arXiv preprint arXiv:1804.03999*, 2018.

[13] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[14] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019. [Online]. Available: http://jmlr.org/papers/v20/18-598.html

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[16] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64 270–64 277, 2018.

[17] A. Bouguettaya, H. Zarzour, A. Kechida, and A. M. Taberkit, "A survey on deep learning for aerial image analysis," *Computer Science Review*, vol. 34, p. 100180, 2020, highlights the challenges of on-board processing due to limited power and payload capacity of UAVs.

[18] K. Dilip, G. S. P. Ghantasala, M. Rathee, S. Kallam, and P. Bathla, "A brief comparative study of metaheuristic approaches for hyperparameter optimization of machine learning model," in *2023 International Conference on Computer Science and Emerging Technologies (CSET)*, 2023, pp. 1–5.

[19] I. Dhahri, M. Golabi, K. Hammoudi, and L. Idoumghar, "Detecting critical infrastructures in disaster images by combining pspnet and genetic algorithm-driven hyperparameter optimization," in *2025 11th International Conference on Control, Decision and Information Technologies (CoDIT)*, vol. 1, 2025, pp. 2155–2158.

[20] M. J. Ali, L. Moalic, M. Essaid, and L. Idoumghar, "Evolutionary neural architecture search for 2d and 3d medical image classification," in *International Conference on Computational Science (ICCS)*. Springer, 2024, pp. 131–146.

[21] A. Lambora, K. Gupta, and K. Chopra, "Genetic algorithm-a literature review, 2019 international conference on machine learning, big data, cloud and parallel computing (comitcon)," *IEEE*, pp. 380–384, 2019.

[22] D. B. Fogel, "Evolutionary algorithms in theory and practice," 1997.

[23] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2881–2890.

[24] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[25] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.