RESEARCH-ARTICLE

Evolutionary Quadtree Pooling for Convolutional Neural Networks

POWEI HARN, National Central University, Taoyuan, Taiwan

BO HUI, The University of Tulsa, Tulsa, OK, United States

LIBO SUN, Auburn University, Auburn, AL, United States

WEI-SHINN KU, Auburn University, Auburn, AL, United States

Published: 14 July 2025

Citation in BibTeX format

GECCO '25: Genetic and Evolutionary Computation Conference
July 14 - 18, 2025
Malaga, Spain

Conference Sponsors:
SIGEVO

Evolutionary Quadtree Pooling for Convolutional Neural Networks

Po-wei Harn
National Central University
Taoyuan, Taiwan, ROC
harnpowei@ncu.edu.tw

Bo Hui
The University of Tulsa
Tulsa, OK, USA
boh1704@utulsa.edu

Libo Sun
Auburn University
Auburn, AL, USA
lzs0101@auburn.edu

Wei-Shinn Ku
Auburn University
Auburn, AL, USA
weishinn@auburn.edu

Abstract

Despite the success of Convolutional Neural Networks (CNNs) in computer vision, it can be beneficial to reduce parameters, increase computational efficiency, and regulate overfitting. One such reduction technique is the use of so-called pooling, which gradually reduces the spatial dimensions of the data throughout the network. Recently, Quadtree-based Genetic Programming has achieved state-of-the-art results for optimizing spatial areas on customized requirements in different grid structures. Motivated by its success, we propose to extend this approach to pooling layers of CNNs. In this direction, this paper introduces a new way to look at each pooling layer. Specifically, we propose an Evolutionary Quadtree Pooling (EQP) method that can identify the best pooling scheme. By embedding multiple quadtrees set as a pooling scheme in the pooling layers of a CNN, we are able to operate crossover and mutation on the feature maps. The evolutionary process of EQP guides the search to provide more reliable evaluations, where each individual can be seen as a CNN with a new type of pooling scheme. Our experimental results show that the best candidate network of EQP outperforms state-of-the-art max, average, stochastic, median, soft, and mixed pooling in accuracy and overfitting reduction while maintaining low computational costs. Our codes are available at https://github.com/poweiharn/EQP.git.

CCS Concepts

• **Computing methodologies → Search methodologies**; **Genetic programming**; **Machine learning**; **Neural networks**; **Computer vision**.

Keywords

Evolutionary Computation, Genetic Programming, Convolutional Neural Networks, Image Classification

ACM Reference Format:
Po-wei Harn, Bo Hui, Libo Sun, and Wei-Shinn Ku. 2025. Evolutionary Quadtree Pooling for Convolutional Neural Networks. In *Genetic and Evolutionary Computation Conference (GECCO '25), July 14–18, 2025, Malaga, Spain.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3712256.3726325

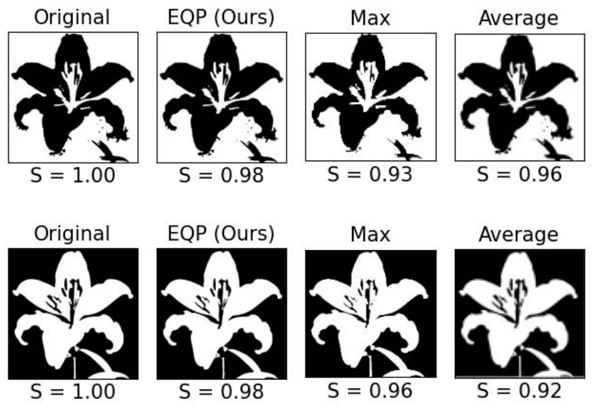

Figure 1: The comparisons of EQP, max pooling, and average pooling on two separate images. Score S represents the pairwise cosine similarity from CLIP [26]. A value closer to 1 indicates that the pooled image is more similar to its original.

1 Introduction

Convolutional Neural Networks (CNNs) are the state-of-the-art industry standard for various computer vision tasks [23]. Due to the layer-by-layer structure that adapts to the shape of the input, it is widely used in computer vision and image analysis such as image classification [1, 6, 39] and segmentation [10, 13, 34, 42]. The key components of CNN consist of multiple convolutional layers and pooling layers, where they are continuously improved to push the accuracy and efficiency of CNNs above historical benchmarks.

In addition to the aforementioned merits, CNN suffers from the undesirable behavior of overfitting when the model performs well with training data and fails to perform on test data. More specifically, the model learns to recognize the noise patterns presented in the training data. Therefore, a large gap is seen between the training and test error. To mitigate this issue, pooling is considered one of the fundamental techniques to reduce overfitting [2].

Many pooling methods have been proposed, such as max pooling [14, 28], average pooling [18], mixed pooling [37], and stochastic pooling [40, 41]. However, only max and average pooling are widely employed among users because of their effectiveness in sharping or lowering sensitivity in local and global pooling layers. Despite that, they have their shortcomings [37, 40]. Figure 1 illustrates the pooling effect of EQP, max pooling, and average pooling on two separate images: one with black flower (top) and the other with white flower (bottom). As shown in the figure, the image with black flower vanishes when max pooling is employed. This is because max pooling can erase important features if the main details have less intensity than the insignificant details. As for the white flower, average pooling diminishes feature characteristics if the averaging of the data with values is much lower than the significant ones. Meanwhile, our EQP approach is able to overcome the shortcomings of both max and average pooling.

Constructing a series of pooling methods bundled as a pooling scheme for multiple tasks is very challenging. First, the pooling scheme needs to capture representative information, such as correlation between features. Second, the design should also be applied to multiple pooling layers in a CNN to have a high-accuracy prediction. Third, even if overfitting regulation is achieved in a CNN model, there are still other purposes that may need to be satisfied, such as the reduction of pooling operations to increase computational efficiency [7]. Therefore, instead of designing such a pooling scheme to satisfy the conditions, we intend to use an evolutionary approach [4, 5] to automate the design of the pooling scheme in CNNs that can handle different types of requirements.

Quadtree-based Genetic Programming (Quadtree-GP) [9] has gained immediate success in optimizing spatial regions on different grid structures. Unlike traditional genetic programming (GP), where binary trees or decision trees are used as representations of individuals, the individuals are represented as quadtrees [3], where quadtree crossover and mutation are applied. Motivated by its ability to address the grid region-by-region, we propose applying the idea of Quadtree-GP on the feature maps in the pooling layers of a CNN. More specifically, the pooling schemes are generated using a population of multiple sets of quadtrees. The quadtrees in a quadtree set are then embedded in the pooling layers, where the candidate network is evaluated through training. The candidates with the best validation accuracy are selected for offspring generation. The evolutionary process guides the search to provide more reliable evaluations, where each individual can be seen as a CNN with a new type of pooling scheme.

To the best of our knowledge, this is the first paper to consider evolutionary pooling design using areas of feature maps on CNNs. We term our approach as Evolutionary Quadtree Pooling (EQP). In this paper, we intend to answer the following research questions: **Q1**: Can the pooling scheme produced by EQP give CNN a high-accuracy classifier with minimized pooling operations? **Q2**: Can the pooling scheme generated from EQP help ease the fundamental issue of overfitting? **Q3**: Can the pooling scheme be transferred from one CNN model to another CNN model? The answers to these questions are affirmative through our empirical study. Our contributions are listed as follows:

- **Evolutionary Quadtree Pooling:** Our method EQP, for the first time, can identify the best pooling scheme for a high-accuracy classifier in a CNN model.
- **Overfitting Reduction:** The solution produced by EQP is able to accomplish the minimized gap between the training accuracy and the test accuracy.
- **Competitive Generated Pooling Scheme:** Our experiments show that EQP can achieve both good accuracy and minimized pooling operations in comparison to state-of-the-art max, average, stochastic, median, soft, and mixed pooling on four popular benchmark datasets of MNIST, CIFAR10, CIFAR100, and SVHN.
- **Transferable Pooling Scheme:** Our pooling scheme obtained from EQP is transferable from one CNN model to another CNN model. The transferred CNN model is able to outperform existing pooling methods such as max, average, and stochastic pooling.

2 Related Work

Pooling is an approach used in CNN to down-sample input images into feature maps. There are two primary purposes for pooling. The first is to preserve important features that are needed for processing by the subsequent layers. The second is to regulate overfitting of neural networks. From each pooling window, the features are aggregated within one value, thereby improving the accuracy and sensitivity of feature translation for smaller input data. This reduces computational complexity and memory requirements. Pooling keeps the important information active while the irrelevant information is removed.

The well-known pooling methods are max pooling [14, 28] and average pooling [18], where the respective maximum and mean values are evaluated. Mixed pooling [37] that randomly performs max or average pooling in a CNN has also been used. Wu *et al.* [36] propose Max-Pooling-Dropout, which consists of a dropout function to cover the pooling layers. Lee *et al.* [21] combine three pooling strategies: mixed max-average pooling, gated max-average pooling, and tree pooling. The mixed max-average pooling is the same as mixed pooling. In gated max-average pooling, a gating mask based on a sigmoid function is applied for choosing the mixing proportion between max and average. In tree pooling, the leaves of a binary tree are associated with the pooling filters learned during training, and the overall output is returned from the leaves to the root node. However, these pooling methods limit themselves to max pooling, average pooling, or their certain combination. As a result, important feature characteristics may be lost.

Different types of pooling methods have also been used for other purposes. Soft pooling is proposed to take advantage of both max and average pooling [8, 25, 32, 35]. Stochastic pooling [40, 41] has been introduced to reduce overfitting. Spatial pyramid pooling [11] is used to capture the structure information in input images. Higher-order pooling [33, 38] is proposed to obtain higher-order statistical information from feature maps. Ranked-based pooling [30] takes the average of the top-k elements in each pooling region as the pooled representation. In contrast to these works, the pooling layers of a CNN model are not restricted to a specific type of pooling. We can give a combination of existing pooling methods set in the pooling layers of a CNN.

3 Problem Formulation

Definition 1. *Given a CNN model $\mathcal{N}$, a pooling scheme P is a set of pooling methods deployed in all quadrants of feature maps for all pooling layers of $\mathcal{N}$, where the area of a quadrant is larger than or equal to the pooling window.*

Definition 2. *Given the set of all possible pooling schemes $\mathbb{P}$ and a CNN model $\mathcal{N}$, the optimal pooling scheme problem is to find the best pooling scheme $P^* \in \mathbb{P}$ that satisfies Equation 1. In the equation, $ACC_t(\mathcal{N}, \Pi)$ represents the test accuracy of $\mathcal{N}$ under a fixed set of hyperparameters $\Pi = \{n, opt, W^o, lr, D_{train}, D_{test}\}$, where*

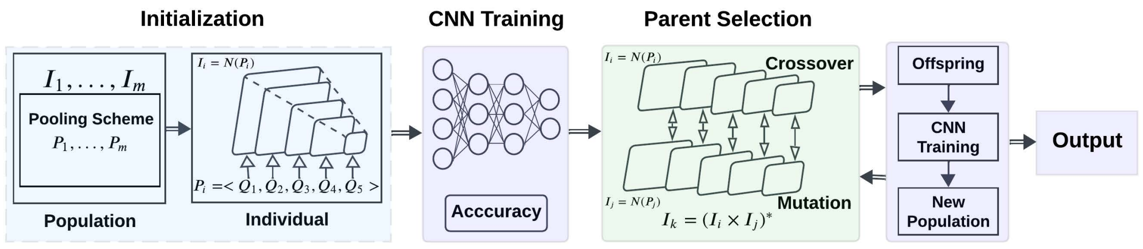

Figure 2: The overall framework of the proposed EQP.

$n, opt, W^o, lr, D_{train}, D_{test}$ denote the number of training epochs, optimizer, initial weights, learning rate, training dataset, and test dataset, respectively.

$$P^* = \arg\max_{\mathbb{P}} ACC_t(\mathcal{N}, \Pi) \tag{1}$$

According to Definition 2, different pooling schemes are applied on $\mathcal{N}$ to result in distinct structures of the feature maps with respect to their pooling layers. The main objective is to search for the optimal pooling scheme that achieves the best test accuracy under a given set of hyperparameters.

4 The Proposed Method

4.1 Evolutionary Quadtree Pooling Overview

Figure 2 illustrates the overall framework of EQP. As shown in the figure, the EQP approach can be divided into two phases: evolutionary pooling search and CNN training. In the first phase, the evolutionary pooling search covers the search space $\mathbb{P}$ by sampling different pooling schemes. The main idea is that through evolution, the EQP can select the individuals with the best pooling scheme for each generation. In the second phase, each pooling scheme is inserted into the pooling layers of a CNN model for evaluation. Candidate networks are trained for n epochs on D_{train}, and the network with the highest validation accuracy on D_{valid} is expected to be considered during the next stage.

The algorithm for EQP is shown in Algorithm 1. As can be seen in the algorithm, the population $\tilde{I} = \{I_1, I_2, ..., I_m\}$ is initialized by deploying a set of pooling schemes $S_P = \{P_1, ..., P_m\}$ into the CNN model (line 1). More specifically, each pooling scheme is a set of quadtrees, where each quadtree is addressed on a corresponding pooling layer of the CNN model. For every individual (i.e., the same CNN with a different pooling scheme), the hyperparameters of the optimizer, initial weights, and learning rate are fixed during training. The network is trained for n epochs on D_{train} and the best classification accuracy on D_{valid} is derived as the individual's fitness (lines 3-6). After the first evaluation, a parent selection strategy is employed to select well performing individuals as the parents. The parents generate offspring O based on the proposed crossover and mutation strategies $(I_i \times I_j)^*$ on the individuals I_i and I_j (lines 8-10). Subsequently, the offspring are evaluated through training by following the similar evaluation method mentioned above (lines 11-14). Then, the offspring are merged with the population for survival selection, where a new population is generated (line 15). Note that a new population is constructed for each generation until gen reaches $maxGen$ in the evolutionary cycle (lines 7-17). Finally, P is returned according to the calculation of Equation 1 with respect to D_{valid} (lines 18-19). If a large number of generations is allowed, the returned pooling scheme P eventually converges to the optimal pooling scheme P^*. However, since the unseen dataset of D_{valid} and D_{test} are not identical, the returned pooling scheme P does not guarantee to be optimal on D_{test}.

Algorithm 1 Evolutionary Quadtree Pooling
Input: $\mathcal{N}$: CNN model, Opt: Optimizer, W^0: Initial weights, lr: Learning rate, D_{train}: Dataset for training, D_{valid}: Dataset for validation, m: Population size, λ: Offspring size, $maxGen$: Maximum generation
Output: Pooling Scheme P

```
1:  Initialize population $\tilde{I} = \{I_1, I_2, ..., I_m\}$ with a set of pooling
    scheme $S_P = \{P_1, ..., P_m\}$, where $I_i = \mathcal{N}(P_i)$.
2:  $gen = 0$
3:  for $I_i \in \tilde{I}$ do
4:      Set $Opt$, $W^0$, and $lr$.
5:      Train the network $\mathcal{N}(P_i)$ for $n$ epochs on $D_{train}$.
6:  end for
7:  while $gen < maxGen$ do
8:      for $k \in \{1, 2, ..., \lambda\}$ do
9:          $O \leftarrow I'_k = (I_i \times I_j)^*$, where $I_i \in \tilde{I}$ and $I_j \in \tilde{I}$
10:     end for
11:     for $I'_k \in O$ do
12:         Set $Opt$, $W^0$, and $lr$.
13:         Train the network $\mathcal{N}(P'_k)$ for $n$ epochs on $D_{train}$.
14:     end for
15:     $\tilde{I} \leftarrow UpdatePopulation(\tilde{I} \cup O)$
16:     $gen \leftarrow gen + 1$
17: end while
18: $P = \arg\max_{S_P} ACC_v\{I_1, I_2, ..., I_m\}$
19: return $P$
```

4.2 Individual Representation

The key insight of the individual design for EQP is to address each pooling layer with a corresponding quadtree. Figure 3 shows the block diagram of a VGG16 [31]. As presented in the figure, a total of five pooling layers is specified, where the dimensions of the feature map are down-sampled by the pooling window of 2×2 for each pooling layer. Based on the observation, we assign a series of five quadtrees in a set as a pooling scheme on VGG16, where they are displayed at the top of Figure 3. The details of the quadtree pooling based on the leaf nodes of the quadtree will be further discussed in the next section. According to the number of quadtrees in a pooling scheme assigned on VGG16, the general case of an individual over a CNN model is shown in Definition 3.

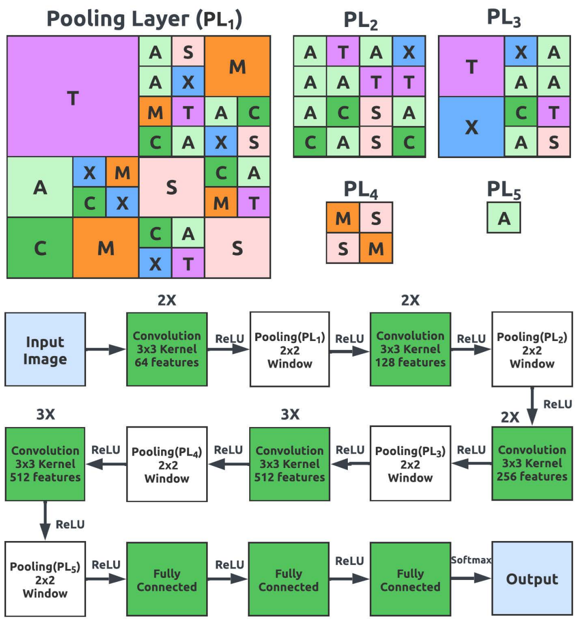

Figure 3: An example of five Quadtrees addressed on each of the pooling layers of the VGG16 architecture. The areas of M, A, S, C, T, **and** X **represent max, average, stochastic, median, soft, and mixed pooling, respectively.**

Definition 3. *Given a CNN model* $\mathcal{N}$ *with* k *pooling layers, an individual of EQP is represented as* $I = \mathcal{N}(P)$*, where* $P = \langle Q_1, ..., Q_k \rangle$ *is a pooling scheme and* Q_i *is the quadtree addressed on the i-th the pooling layer of* $\mathcal{N}$.

Definition 4. *Given two CNN models* $\mathcal{N}$ *and* $\mathcal{N}'$*, and a pooling scheme P, a transfer learning of pooling scheme is to utilize the knowledge of* $\mathcal{N}(P)$ *to improve the performance on* $\mathcal{N}'(P)$.

4.3 The Quadtree Pooling Method in CNN

In this section, we present the quadtree pooling approach. Figure 4 shows an example of the quadtree pooling on a given image, where the areas of M, A, S, C, T, and X represent max, average, stochastic, median, soft, and mixed pooling, respectively. As displayed in the figure, the original image is transformed into its pooled representation. The implementation of quadtree pooling is displayed in Algorithm 2. As shown in the algorithm, the inputs are PL with feature map dimensions of $2^\delta \times 2^\delta$ and a quadtree Q with maximum depth $\delta - 1$, where $\delta > 0$ (See Appendix B). If the current node of the quadtree is a leaf node, the feature map in the pooling layer is addressed with the corresponding pooling method of $Q.node$ using the function $get\_pooling()$ (lines 1-2). Otherwise, $Q.node$ is an internal node of a quadtree (lines 3-17). If $Q.node$ is an internal node, we iterate through all its connected edges 00, 01, 10, and 11 (line 4). According to the edge identifiers, the feature map of the pooling layer is split into four equal quadrants PL^{00}, PL^{01}, PL^{10}, and PL^{11}. For each quadrant, Algorithm 2 is called recursively on the sub-tree to address the partitioned region (line 14). All four quadrants are merged once they have a returned result (line 16). Algorithm 2 will keep on running until no partitioned region remains. Finally, PL is returned with quadtree pooling on the feature map. Note that Algorithm 2 is applied to all the pooling layers.

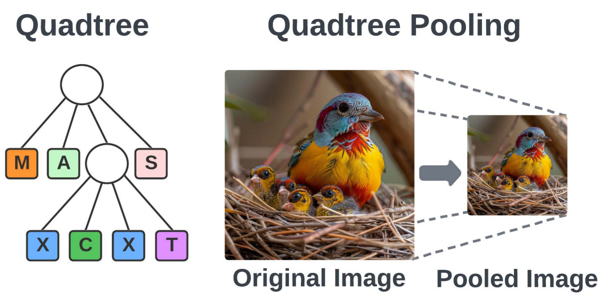

Figure 4: An example of quadtree pooling. The image is downsampled to the pooled image using quadtree pooling, where M, A, S, C, T, **and** X **represent max, average, stochastic, median, soft, and mixed pooling, respectively.**

Algorithm 2 $QT\_Pool(PL_{[0\sim 2^\delta]\times[0\sim 2^\delta]}, Q)$

Input: $PL_{[0\sim 2^\delta]\times[0\sim 2^\delta]}$: Pooling layer with feature map of size $2^\delta \times 2^\delta$, where $\delta > 0$ and $[0 \sim 2^\delta]$ are the indexes ranged from 0 to 2^δ.
Q: Quadtree with $D_{max} = \delta - 1 \geq 0$.
Output: PL with quadtree pooling

```
1:  if $Q.node$ is leaf then
2:      $PL \leftarrow get\_pooling(PL_{[0\sim 2^\delta]\times[0\sim 2^\delta]}, Q.node)$
3:  else
4:      for $Q.edge \in \{00, 01, 10, 11\}$ do
5:          if $Q.edge = 00$ then
6:              $PL^{00} \leftarrow PL_{[0\sim 2^{\delta-1}]\times[0\sim 2^{\delta-1}]}$
7:          else if $Q.edge = 01$ then
8:              $PL^{01} \leftarrow PL_{[0\sim 2^{\delta-1}]\times[2^{\delta-1}\sim 2^\delta]}$
9:          else if $Q.edge = 10$ then
10:             $PL^{10} \leftarrow PL_{[2^{\delta-1}\sim 2^\delta]\times[0\sim 2^{\delta-1}]}$
11:         else if $Q.edge = 11$ then
12:             $PL^{11} \leftarrow PL_{[2^{\delta-1}\sim 2^\delta]\times[2^{\delta-1}\sim 2^\delta]}$
13:         end if
14:         $PL^{Q.edge} \leftarrow QT\_Pool(PL^{Q.edge}, Q.st[Q.edge])$
15:     end for
16:     $PL \leftarrow Merge(PL^{00}, PL^{01}, PL^{10}, PL^{11})$
17: end if
18: return $PL$
```

4.4 Offspring Generation

In EQP, the proposed crossover and mutation strategies are operated on the parent individuals to diversify the construction of pooling schemes. The developed crossover and mutation methods are shown in Definition 5.

Definition 5. *Given a CNN model* $\mathcal{N}$ *with* k *pooling layers and two individuals* $I = \mathcal{N}(P)$ *and* $I' = \mathcal{N}(P')$ *with* $P = \langle Q_1, ..., Q_k \rangle$ *and* $P' = \langle Q'_1, ..., Q'_k \rangle$*, the crossover of* I *and* I' *are defined as* $I \times I' = \mathcal{N}(\langle Q_1 \otimes Q'_1, ..., Q_k \otimes Q'_k \rangle)$*, where* $\otimes$ *is the quadtree crossover operator. The mutation of* I *is defined as* $(I)^* = \mathcal{N}(\langle (Q_1)^\odot, ..., (Q_k)^\odot \rangle)$*, where* $(.)^\odot$ *is the quadtree mutation function.*

The details of the crossover and mutation of two parent individuals are displayed in Figure 5. As can be seen in the figure, since the VGG16 model consists of 5 pooling layers, the basic unit for crossover and mutation between two parents is the feature map.

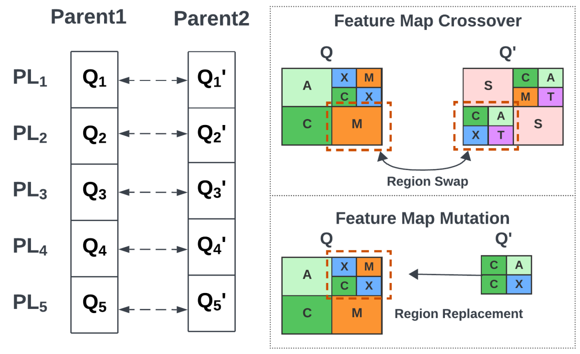

Figure 5: An example of the crossover and mutation of two parent individuals of VGG16. The regions of the feature map are swapped and replaced during evolution.

The quadtrees from two parent individuals addressed in each pooling layer are to be crossed over separately. More specifically, two sub-regions of their respective feature maps are selected from two sub-trees of their parents. Next, the chosen sub-trees swap with each other resulting in region swap between the feature maps. While two quadtrees are generated, one quadtree is randomly selected and preserved to address its respective pooling layer of the offspring network. After the crossover operation is completed on two parents, mutation is provided on the offspring to improve the exploration of EQP. In the mutation operation, a sub-tree, which is a sub-region of the feature map, is randomly selected for each quadtree of their respective pooling layers. In this case, a total of 5 sub-trees are selected and replaced by 5 newly generated quadtrees resulting in region replacement over the feature maps.

4.5 Overfitting and Computational Reduction

In the CNN training phase, the candidate individuals are trained on the training dataset D_{train} for n epochs. For each iteration, the images are processed through the layer-by-layer structure of the CNN, where the values of the feature maps are aggregated based on quadtree pooling. The weights of the CNN model are updated by gradient descent. The fitness value of the candidate individuals is then assessed on the validation dataset D_{valid}, with validation accuracy as performance metric. However, we expect the pooling schemes not only to reduce the effect of overfitting but also to minimize the number of computational operations. Therefore, we modify the fitness function to make the search gradually focus on multiple objectives [27, 29].

$$F(I) = \alpha ACC_v(I) + \beta GAP(I) + \gamma OP(I) \tag{2}$$

In theory, the overfitting issue occurs because the model performs well with the training data and fails to perform on unseen data. The margin between the training accuracy and the validation accuracy increases during training. Thus, the measurement of overfitting can be based on the generalization gap [15, 24], where it is defined as the difference between the training accuracy and the validation accuracy. A large generalization gap indicates that the model tends to overfit. Hence, the design of the fitness function is to penalize the degree of generalization gap as well as the number of pooling operations. Equation 2 shows the modified fitness function $F(I)$, where $ACC_v(.)$ is the validation accuracy accessed on the candidate network, $GAP(.)$ is the generalization gap for overfitting measurement, $OP(.)$ is the operation count on the pooling scheme in a dataset, and α, β, and γ are hyperparameters that can be fine-tuned based on customized requirements. Note that the test dataset D_{test} is not used during CNN training or fitness calculation but is considered as the final evaluation of the best candidate.

5 Experimental Setup

The experiments are conducted on a machine with Nvidia P100 GPU (16GB memory), 2-core Intel(R) Xeon(R) CPU (2.30 GHz) and 12 GB of RAM. All the implementations are written in Python. The experiment results collected from each configuration are averaged over 10 runs with different random seeds.

5.1 Benchmark Datasets

We use four standard benchmark image datasets: MNIST [20], CIFAR10, CIFAR100 [16], and SVHN [22]. MNIST consists of unbalanced handwritten digits (0 to 9) with a training set of 60,000 images and a test set of 10,000 images. Each image is 28×28 grey-scale pixels. Both CIFAR10 and CIFAR100 contain 60,000 RGB images of size 32 × 32. Each dataset has a training set of 50,000 images and a test set of 10,000 images. CIFAR10 comprises 10 classes, each with an equal number of images, while CIFAR100 includes 100 classes. SVHN consists of 10 classes of house numbers digits (0 to 9) with a training set of 73,257 and a test set of 26,032 images, where each image is RGB of size 32 × 32. We divide D_{train}, D_{valid}, and D_{test} of all the benchmark datasets according to the ratio of $4:1:1$, where D_{train} and D_{valid} are split from training set and there is no intersection between them. Note that the images of the MNIST dataset are converted to size 32 × 32 to match the input dimensions of our CNN model.

5.2 Parameter Settings

Considering the number of pooling layers, VGG16 [31] is used as our standard CNN model. Compared to other CNN models such as LeNet [19], AlexNet [17] and ResNet [12], VGG contains the most numbers of pooling layers of five. Hence, a total of five quadtrees with D_{max} set from 4 to 0 are deployed in each pooling scheme using ramped half-and-half [9]. The elements of the leaf nodes consist of the set $\{M, A, S, C, T, X\}$, where M, A, S, C, T, and X represent max, average, stochastic, median, soft, and mixed pooling, respectively. All the pooling methods have their stride set to 2 and a pooling window of 2×2. The population size of EQP is set to 20, the offspring size λ is set to 10, the crossover probability is set to 0.5 and the mutation probability is set to 0.5. A tournament size of 5 is used for parent selection. The maximum number of generations is set to 10 considering the computational cost. In CNN training and the final evaluation, the total number of epochs is set to 20. The hyperparameters of α, β, and γ are set to 1, -10, and -10^{-8}, respectively. Besides the proposed EQP, two EQP variants are considered: one without crossover (EQP w/o Cross) and the other without mutation (EQP w/o Mu.). Genetic Algorithm (GA) [5] and random search are represented as search baselines. Both are unprecedented in scope and they are capable of searching pooling scheme candidates in a reasonable scale. In addition, M, A, S, C, T, and X are set as pooling baselines. More experimental details and settings are presented in **Appendices C, D, E, and F**.

Table 1: Best of EQP vs. Best of GA and Best of random search, and various State-of-the-Art pooling methods. Mean ± standard deviation test accuracy (%) of 10 runs and the number of operations (B) required on MNIST, CIFAR10, CIFAR100, and SVHN, respectively. The best values are in bold. Column ST denotes the statistical test results based on the Wilcoxon rank sum test.

Dataset	MNIST			CIFAR10			CIFAR100			SVHN		
	Acc.	#Ops.	ST	Acc.	#Ops.	ST	Acc.	#Ops.	ST	Acc.	#Ops.	ST
EQP	**99.19 ± 0.03**	11.64B (4th)	+	**80.78 ± 0.04**	6.88B (2nd)	+	**43.22 ± 0.68**	9.90B (3rd)	+	**92.25 ± 0.22**	18.87B (5th)	+
EQP w/o Cross.	98.98 ± 0.10	15.98B	+	79.17 ± 0.68	15.98B	+	38.26 ± 0.64	15.67B	+	91.25 ± 0.33	15.98B (4th)	+
EQP w/o Mu.	98.95 ± 0.14	10.44B (3rd)	+	79.96 ± 0.46	14.05B (5th)	+	37.91 ± 0.37	11.46B (4th)	+	91.52 ± 0.22	11.78B (2nd)	+
GA	98.91 ± 0.09	22.68B	/	78.85 ± 0.75	23.15B	/	38.37 ± 0.61	22.25B	/	91.57 ± 0.43	23.12B	/
Random	98.90 ± 0.07	23.92B	-	78.62 ± 0.46	23.57B	-	37.99 ± 0.82	23.75B	-	91.18 ± 0.30	23.57B	-
Max Pooling	99.09 ± 0.06	**6.56B** (1st)		80.59 ± 1.00	**5.62B** (1st)		41.10 ± 1.19	**5.62B** (1st)		91.68 ± 0.39	**9.30B** (1st)	
Average Pooling	99.03 ± 0.07	8.74B (2nd)		79.99 ± 0.86	7.49B (3rd)		42.37 ± 1.12	7.49B (2nd)		91.51 ± 0.49	12.40B (3rd)	
Stochastic Pooling	98.75 ± 0.20	26.23B		76.22 ± 1.34	22.48B		33.36 ± 1.65	22.48B		89.83 ± 0.63	37.21B	
Median Pooling	98.77 ± 0.07	34.98B		76.36 ± 0.80	29.98B		35.96 ± 1.20	29.98B		89.82 ± 0.60	49.62B	
Mixed Pooling	99.02 ± 0.09	15.30B (5th)		80.75 ± 1.04	13.11B (4th)		43.06 ± 1.07	13.11B (5th)		91.69 ± 0.39	21.71B	
Soft Pooling	99.09 ± 0.07	17.49B		80.20 ± 0.93	14.99B		40.35 ± 1.58	14.99B		91.76 ± 0.37	24.81B	

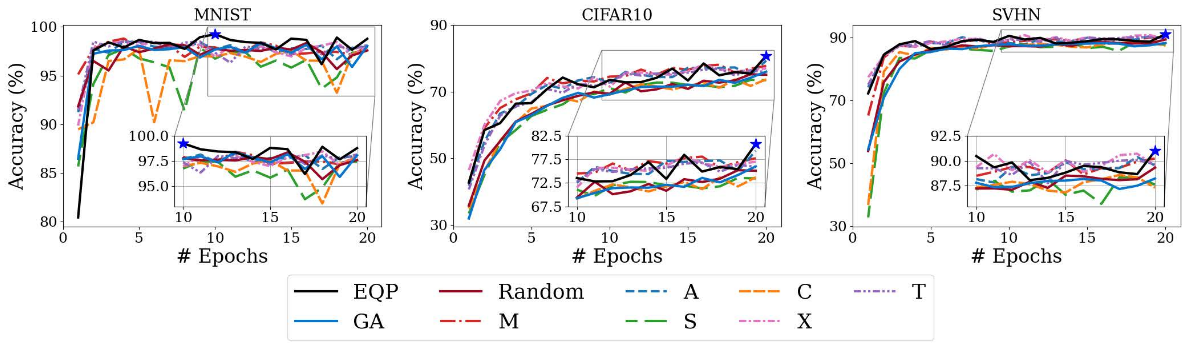

Figure 6: The performances of the Best of EQP vs. Best of GA and Best of random search, and various State-of-the-Art pooling methods on MNIST, CIFAR10, and SVHN, where M, A, S, C, T, and X represent max, average, stochastic, median, soft, and mixed pooling, respectively. Marker ★ denotes the best test accuracy.

6 Experimental Analysis

This section presents the results and the analyses of the proposed EQP on MNIST, CIFAR10, CIFAR100 and SVHN datasets. We compare the classification accuracy, number of operations, and generation gap with peer competitors on the test dataset. Statistical test and fitness analysis are provided to demonstrate the effectiveness of EQP. In addition, we also apply pooling scheme transfer from a source to a target CNN model.

6.1 Comparison on Classification Accuracy

Table 1 presents the comparison of the best candidate of EQP against GA, random search, and the pooling baselines on MNIST, CIFAR10, CIFAR100 and SVHN. Specifically, the best candidate of EQP has the highest test accuracy on all datasets due to its generalization capabilities on both gray-scale and colored images. In gray-scale images of MNIST, max pooling obtains better results than average pooling. This is because of the drawbacks of average pooling on images with black background. Because GA, random search, mixed pooling, and soft pooling have a proportion of average pooling, their softening effects are not that strong. Stochastic and median pooling, on the other hand, add noise to the images, which results in low accuracies. As for colored images in CIFAR10, CIFAR100 and SVHN, the combination of max and average pooling performs well on declaring contrast balance on datasets. Mixed pooling is effective on CIFAR10 and CIFAR100, while soft pooling is strong on SVHN. Because of the max and average pooling are extreme cases of softening and sharpening on images, their performances are behind their mixtures. Meanwhile, stochastic and median pooling have the same issues as gray-scale images. Since GA and random search contain both of their effects, their accuracies are slightly above them.

Figure 6 shows the best candidates of EQP versus GA, random search, and pooling baselines on MNIST, CIFAR10, and SVHN during the final evaluation. As displayed in the figure, the best accuracy (★) of all three datasets belongs to EQP. In MNIST, EQP has the best performance on the 10th epoch and continues to perform extraordinarily until the end of training. GA, random search, max pooling, average pooling, and their mixtures perform solidly with small fluctuations, while stochastic and median pooling are unsteady because of the generated noise. For CIFAR10 and SVHN, the performances of the best candidate of EQP and its baselines converge at the 5th epoch. Eventually, EQP stands out compared to all the baselines at the 20th epoch of training. Due to the randomness of pooling schemes and limited exploration capabilities, GA and random search are only as far as stochastic and median pooling. Even though the best candidate of EQP is evaluated based on D_{valid}, this does not guarantee that it is the best solution on D_{test}. However, from experimental results, the best candidate of EQP is sufficient enough to outperform GA, random search, and existing pooling baselines.

6.2 Statistical Tests

The Wilcoxon rank sum test is conducted in Table 1 to confirm the significance of EQP. Column ST shows the results of this test

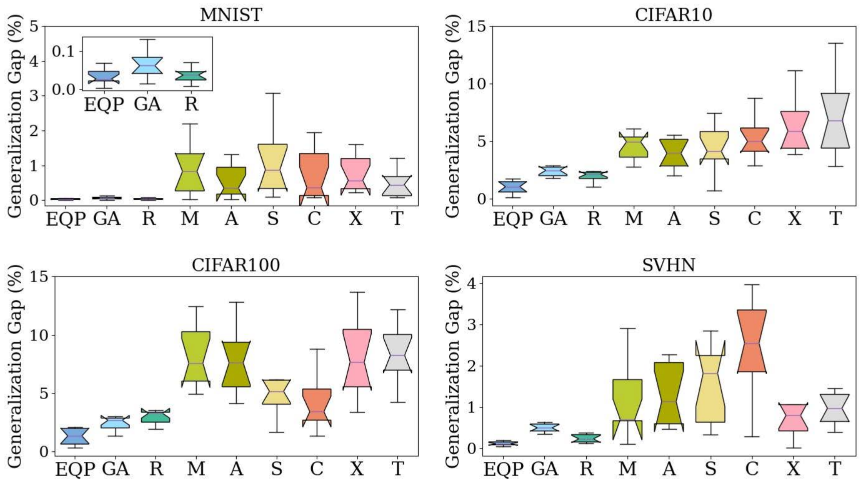

Figure 7: The generalization gap of the Best of EQP, Best of GA and Best of random search (R), and various State-of-the-Art pooling methods (M,A,S,C,T,X) on MNIST, CIFAR10, CIFAR100 and SVHN.

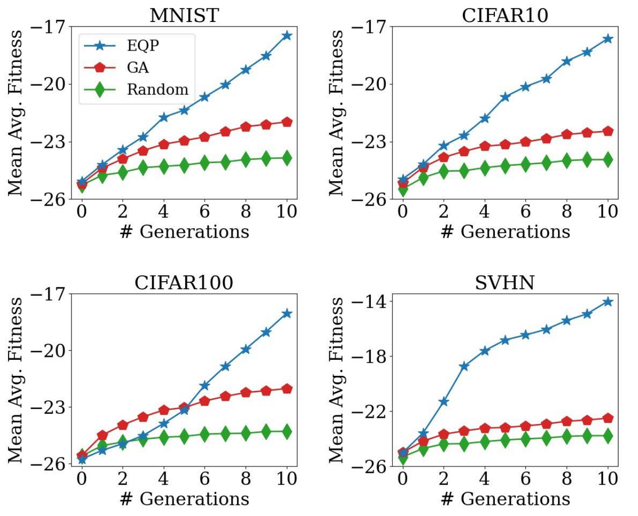

Figure 8: The mean average fitness of EQP, GA, and random search on MNIST, CIFAR10, CIFAR100 and SVHN.

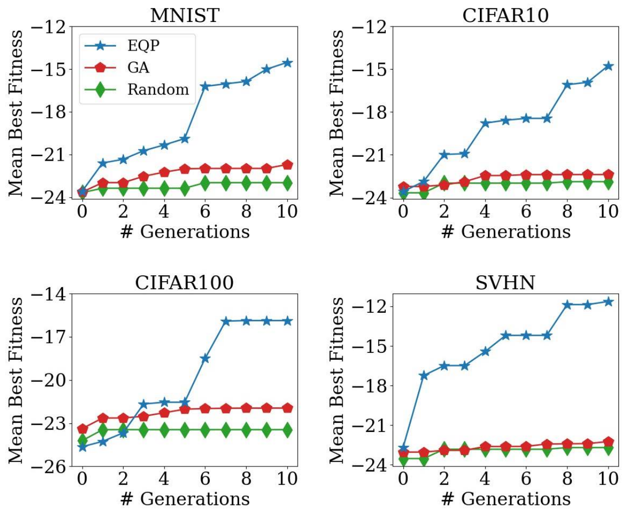

Figure 9: The mean best fitness of EQP, GA, and random search on MNIST, CIFAR10, CIFAR100 and SVHN.

comparing each search method against baseline GA. The level of significance is set to 0.05. The notations "+" and "−" are used if there are significant differences between samples, otherwise "=" if there are no significant differences. The results show that EQP consistently outperforms GA across all datasets. Even without crossover or mutation, both variants of EQP are able to dominate GA (one-point crossover and bit string mutation) and random search (random generation), respectively. This is because swapping or replacing the regions of the feature maps is more effective than overwriting most of the pooling windows. As a result, both variants of EQP produce candidates with better accuracy and fewer amount of operations than GA. This confirms that the contribution of the proposed crossover and mutation strategies is extremely significant to EQP.

6.3 Computational Complexity

In this section, we compute the computational complexity of baseline pooling methods in terms of mathematical operations (multiplications, additions, logical, etc.) to compare them with the best pooling schemes generated by EQP. All the images in a dataset, which is processed by forward propagation for each pooling layer of the CNN, are taken into consideration. The number of operations in terms of billions as a unit for each pooling method with respect to all four datasets is shown in Table 1. Note that the values for baseline pooling methods in CIFAR10 and CIFAR100 are the same because of the equal size of the datasets. As displayed in the table, max pooling requires the fewest operations because of its simplicity. Average pooling achieves the second best in MNIST and CIFAR100, while the best candidates of EQP and its variants are not far behind. For CIFAR10 and SVHN, the second place belongs to EQP or its variants because of their tendencies of minimizing the number of operations. It is interesting to see that EQP and at least one of its variants are able to make the top five spot. Mix and soft pooling belong in the same category because they both need to calculate the average along with a mixture of summation. As for stochastic and median pooling, they require multiple iterations on the feature map, which result in the least computational efficiency.

Table 2: The comparison of pooling scheme transfer from VGG16 to VGG11, VGG13, and VGG19 of EQP against baseline pooling methods. Mean ± standard deviation classification accuracy (%) of 10 runs on MNIST, CIFAR10, CIFAR100 and SVHN. Best values are in bold. Marker * represents the performance of the target network is better than the source network.

Method	MNIST	CIFAR10	CIFAR100	SVHN
Transferred Pooling Scheme on VGG11	98.86 ± 0.05	77.66 ± 0.85	43.01 ± 1.09	90.63 ± 0.68
Max Pooling	**99.06 ± 0.10**	**79.60 ± 0.94**	**45.38 ± 0.48**	90.48 ± 0.42
Average Pooling	99.05 ± 0.12	78.53 ± 0.97	44.65 ± 0.93	**91.17 ± 0.59**
Stochastic Pooling	98.24 ± 0.18	71.80 ± 0.70	34.02 ± 1.11	85.80 ± 0.37
Transferred Pooling Scheme on VGG13	**99.23 ± 0.11***	81.23 ± 1.31*	45.65 ± 0.64*	**91.97 ± 0.50**
Max Pooling	99.10 ± 0.08	81.49 ± 0.87	**47.79 ± 1.53**	91.87 ± 0.38
Average Pooling	99.12 ± 0.11	**81.61 ± 1.26**	46.95 ± 0.98	91.92 ± 0.56
Stochastic Pooling	98.69 ± 0.18	76.72 ± 1.19	38.20 ± 0.93	89.10 ± 0.49
Transferred Pooling Scheme on VGG19	**99.04 ± 0.06**	78.53 ± 0.80	**35.61 ± 1.21**	**91.95 ± 0.65**
Max Pooling	98.97 ± 0.13	**79.19 ± 0.63**	34.94 ± 1.68	91.32 ± 0.45
Average Pooling	98.99 ± 0.12	78.78 ± 1.80	35.17 ± 1.12	91.34 ± 0.66
Stochastic Pooling	98.68 ± 0.18	75.16 ± 1.16	28.46 ± 1.53	89.85 ± 0.71

Even though the best candidate of GA and random search contain all pooling methods, their operation count lies close to stochastic pooling in most cases.

6.4 Overfitting Reduction

We examine the generalization gap comparison between the best candidate of EQP and its baselines presented in Figure 7. As shown in the figure, EQP, GA and random search stand out among the existing baseline pooling methods. EQP is resistant to overfitting since it has the lowest generalization gap on average for all datasets. The performance of random search exceeds GA in most cases because of the full diversity of pooling schemes which results in better generalization. For MNIST, CIFAR10 and CIFAR100, average, stochastic, and median pooling are effective to reduce overfitting, while max, mix, and soft pooling are behind. This is because the CNN model is able to generalize well on new unseen data when noise is added. In SVHN, the sharpening effect of max pooling surprisingly increases generalization. This is because the dataset already contains distracting features due to real life sources. Hence, the performances of max pooling and its mixtures are able to reduce overfitting compared to average, stochastic and median pooling.

6.5 Fitness Analysis

In this section, we analyze the mean average fitness and mean best fitness of EQP, GA, and random search on MNIST, CIFAR10, CIFAR100 and SVHN. The higher the fitness value indicates that the search method is able to find better solutions for optimizing classification accuracy, generalization gap, and the number of pooling operations. The results are illustrated in Figure 8 and Figure 9. As shown in both figures, all methods start from a similar state but as evolution proceeds, EQP is able to obtain better results and continues to improve later on. The mean average fitness and mean best fitness of EQP show a steeper increase than GA and random search because of its exploration capabilities of crossover and mutation. In MNIST, CIFAR10, and SVHN, EQP has already dominated search baselines from generation 1. As for CIFAR100, the performance of EQP on mean average fitness exceeds random search on generation 3 and GA on generation 6. Meanwhile, the mean best fitness of EQP outperforms both GA and random search on generation 5. The fitness value of GA surpasses random search because of the evolutionary process that random search lacks.

6.6 Transferred Pooling Scheme

We explore the effectiveness of transferring a pooling scheme on a source to a target CNN model. The best pooling scheme found by EQP using VGG16 is transferred to two shallow networks of VGG11 and VGG13 and a deep network of VGG19. All the transferred networks are trained for 20 epochs with the same parameter settings in CNN training. The weights are randomly generated for each network but are initialized the same for the pooling baselines. The results of the transferred pooling scheme are shown in Table 2. As displayed in the table, the transferred pooling schemes achieve good performances on VGG13 and VGG19. Most importantly, the accuracy of VGG13 on MNIST, CIFAR10, and CIFAR100 surpasses the best candidate of EQP on the source VGG16. As for VGG11, its performance falls behind max and average pooling. We think this is because the knowledge gained from source network is inversely proportional to the layer differences of the target. Because VGG13 and VGG19 are more similar models to VGG16 than VGG11, the utilization of knowledge transfer from VGG16 is more compatible with VGG13 and VGG19 than VGG11.

7 Conclusion

In this paper, we propose a novel EQP approach to search for the optimal pooling scheme for CNN models. From our approach, multiple quadtrees set as a pooling scheme are embedded in the pooling layers of a CNN. The experiment results verify that the best candidate network of EQP outperforms state-of-the-art pooling methods in accuracy and overfitting reduction while maintaining low computational costs. Statistical results further strengthen our proposed evolution strategies against GA and random search. Furthermore, we demonstrate that the pooling schemes are also transferable from a source to a target CNN model. However, due to the NP-hardness of the optimal pooling scheme problem (See Appendix A), our solution is only the best candidate discovered. In future work, we intend to solve this problem by investigating deep into the network architecture hopefully to reveal the relationship between pooling schemes and decision boundaries.

Acknowledgments

This research was partially supported by National Science and Technology Council through Grant NSTC 113-2222-E-008-009.

References

[1] Muhammad Junaid Ali, Laurent Moalic, Mokhtar Essaid, and Lhassane Idoumghar. 2023. Designing Convolutional Neural Networks using Surrogate assisted Genetic Algorithm for Medical Image Classification. In *GECCO Companion*. ACM, 263–266.

[2] Bilal Alsallakh, David Yan, Narine Kokhlikyan, Vivek Miglani, Orion Reblitz-Richardson, and Pamela Bhattacharya. 2023. Mind the Pool: Convolutional Neural Networks Can Overfit Input Size. In *ICLR*. OpenReview.net.

[3] Srinivas Aluru. 2004. Quadtrees and Octrees. In *Handbook of Data Structures and Applications*. Chapman and Hall/CRC.

[4] Aaryan Dubey, Alexandre Hoppe Inoue, Pedro Terra Fernandes Birmann, and Sammuel Ramos da Silva. 2022. Evolutionary feature selection: a novel wrapper feature selection architecture based on evolutionary strategies. In *GECCO*. ACM, 359–366.

[5] A. E. Eiben and James E. Smith. 2015. *Introduction to Evolutionary Computing, Second Edition*. Springer.

[6] Arjun Ghosh, Nanda Dulal Jana, Swagatam Das, and Rammohan Mallipeddi. 2023. Two-Phase Evolutionary Convolutional Neural Network Architecture Search for Medical Image Classification. *IEEE Access* 11 (2023), 115280–115305.

[7] Ritam Guha, Wei Ao, Stephen Kelly, Vishnu Boddeti, Erik D. Goodman, Wolfgang Banzhaf, and Kalyanmoy Deb. 2023. MOAZ: A Multi-Objective AutoML-Zero Framework. In *GECCO*. ACM, 485–492.

[8] Çaglar Gülçehre, KyungHyun Cho, Razvan Pascanu, and Yoshua Bengio. 2014. Learned-Norm Pooling for Deep Feedforward and Recurrent Neural Networks. In *ECML/PKDD (1) (Lecture Notes in Computer Science, Vol. 8724)*. Springer, 530–546.

[9] Po-Wei Harn, Bo Hui, Sai Deepthi Yeddula, Libo Sun, Min-Te Sun, and Wei-Shinn Ku. 2023. A Novel Quadtree-Based Genetic Programming Search for Searchable Encryption Optimization. In *GECCO Companion*. ACM, 583–586.

[10] Tahereh Hassanzadeh, Daryl Essam, and Ruhul A. Sarker. 2023. EEvoU-Net: An ensemble of evolutionary deep fully convolutional neural networks for medical image segmentation. *Appl. Soft Comput.* 143 (2023), 110405.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 37, 9 (2015), 1904–1916.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. IEEE Computer Society, 770–778.

[13] Akiyoshi Hizukuri, Yuto Hirata, and Ryohei Nakayama. 2023. Semantic Face Segmentation Using Convolutional Neural Networks With a Supervised Attention Module. *IEEE Access* 11 (2023), 116892–116902.

[14] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. 2009. What is the best multi-stage architecture for object recognition?. In *ICCV*. IEEE Computer Society, 2146–2153.

[15] Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. 2019. Predicting the Generalization Gap in Deep Networks with Margin Distributions. In *ICLR (Poster)*. OpenReview.net.

[16] Alex Krizhevsky. 2009. Learning Multiple Layers of Features from Tiny Images. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*. 1106–1114.

[18] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. 1989. Handwritten Digit Recognition with a Back-Propagation Network. In *NIPS*. Morgan Kaufmann, 396–404.

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[20] Yann LeCun and Corinna Cortes. 2005. The mnist database of handwritten digits. https://yann.lecun.com/exdb/mnist

[21] Chen-Yu Lee, Patrick W. Gallagher, and Zhuowen Tu. 2018. Generalizing Pooling Functions in CNNs: Mixed, Gated, and Tree. *IEEE Trans. Pattern Anal. Mach. Intell.* 40, 4 (2018), 863–875.

[22] Yuval Netzer, Tao Wang, Adam Coates, A. Bissacco, Bo Wu, and A. Ng. 2011. Reading Digits in Natural Images with Unsupervised Feature Learning. https://api.semanticscholar.org/CorpusID:16852518

[23] M.A. Nielsen. 2015. *Neural Networks and Deep Learning*. Determination Press. https://books.google.com/books?id=STDBswEACAAJ

[24] Svetlana Pavlitskaya, Joël Oswald, and J. Marius Zöllner. 2022. Measuring Overfitting in Convolutional Neural Networks using Adversarial Perturbations and Label Noise. In *SSCI*. IEEE, 1551–1559.

[25] Pedro H. O. Pinheiro and Ronan Collobert. 2015. From image-level to pixel-level labeling with Convolutional Networks. In *CVPR*. IEEE Computer Society, 1713–1721.

[26] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *ICML (Proceedings of Machine Learning Research, Vol. 139)*. PMLR, 8748–8763.

[27] Nian Ran, Bahrul Ilmi Nasution, Claire Little, Richard Allmendinger, and Mark J. Elliot. 2024. Multi-objective evolutionary GAN for tabular data synthesis. In *GECCO*. ACM.

[28] Marc'Aurelio Ranzato, Y-Lan Boureau, and Yann LeCun. 2007. Sparse Feature Learning for Deep Belief Networks. In *NIPS*. Curran Associates, Inc., 1185–1192.

[29] Lennart Schneider, Bernd Bischl, and Janek Thomas. 2023. Multi-Objective Optimization of Performance and Interpretability of Tabular Supervised Machine Learning Models. In *GECCO*. ACM, 538–547.

[30] Zenglin Shi, Yangdong Ye, and Yunpeng Wu. 2016. Rank-based pooling for deep convolutional neural networks. *Neural Networks* 83 (2016), 21–31.

[31] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*.

[32] Alexandros Stergiou, Ronald Poppe, and Grigorios Kalliatakis. 2021. Refining activation downsampling with SoftPool. In *ICCV*. IEEE, 10337–10346.

[33] Qilong Wang, Zilin Gao, Jiangtao Xie, Wangmeng Zuo, and Peihua Li. 2018. Global Gated Mixture of Second-order Pooling for Improving Deep Convolutional Neural Networks. In *NeurIPS*. 1284–1293.

[34] Jiahong Wei, Bing Xue, and Mengjie Zhang. 2024. EZUAS: Evolutionary Zero-shot U-shape Architecture Search for Medical Image Segmentation. In *GECCO*. ACM.

[35] Zhen Wei, Jingyi Zhang, Li Liu, Fan Zhu, Fumin Shen, Yi Zhou, Si Liu, Yao Sun, and Ling Shao. 2019. Building Detail-Sensitive Semantic Segmentation Networks With Polynomial Pooling. In *CVPR*. Computer Vision Foundation / IEEE, 7115–7123.

[36] Haibing Wu and Xiaodong Gu. 2015. Max-Pooling Dropout for Regularization of Convolutional Neural Networks. In *ICONIP (1) (Lecture Notes in Computer Science, Vol. 9489)*. Springer, 46–54.

[37] Dingjun Yu, Hanli Wang, Peiqiu Chen, and Zhihua Wei. 2014. Mixed Pooling for Convolutional Neural Networks. In *RSKT (Lecture Notes in Computer Science, Vol. 8818)*. Springer, 364–375.

[38] Kaicheng Yu and Mathieu Salzmann. 2018. Statistically-Motivated Second-Order Pooling. In *ECCV (7) (Lecture Notes in Computer Science, Vol. 11211)*. Springer, 621–637.

[39] Gonglin Yuan, Bing Xue, and Mengjie Zhang. 2023. An Effective One-Shot Neural Architecture Search Method with Supernet Fine-Tuning for Image Classification. In *GECCO*. ACM, 615–623.

[40] Matthew D. Zeiler and Rob Fergus. 2013. Stochastic Pooling for Regularization of Deep Convolutional Neural Networks. In *ICLR*.

[41] Shuangfei Zhai, Hui Wu, Abhishek Kumar, Yu Cheng, Yongxi Lu, Zhongfei Zhang, and Rogério Schmidt Feris. 2017. S3Pool: Pooling with Stochastic Spatial Sampling. In *CVPR*. IEEE Computer Society, 4003–4011.

[42] Wenying Zhang, Min Dong, and Li Jiang. 2023. Image segmentation using convolutional neural networks in multi-sensor information fusion. *Soft Comput.* 27, 23 (2023), 18353–18372.