Research Article

An Automatic Architecture Designing Approach of Convolutional Neural Networks for Road Surface Conditions Image Recognition: Tradeoff between Accuracy and Efficiency

Mingjian Wu and Tae J. Kwon

Department of Civil and Environmental Engineering, University of Alberta, Edmonton, AB, Canada T6G 2W2

Correspondence should be addressed to Mingjian Wu; mingjian.wu@ualberta.ca

Received 8 October 2021; Revised 17 March 2022; Accepted 28 June 2022; Published 14 July 2022

Academic Editor: Zahid Mehmood

Copyright © 2022 Mingjian Wu and Tae J. Kwon. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Convolutional neural network (CNN) is a promising image recognition technique for winter road surface condition (RSC), a measure that is crucial for winter maintenance operations. In the past, researchers have designed RSC CNN models that displayed acceptable results but did so focusing solely on obtaining high classification accuracy without any consideration for efficiency. Furthermore, when it comes to model development itself, architecture design requires expertise in CNN as well as rich knowledge in the investigated problem itself. To rectify these issues, this paper proposes an innovative approach to automatically design RSC CNN architecture without compromising classification accuracy. The proposed approach uses a weighted sum method, which provides the freedom of choosing relative importance level between accuracy and efficiency.

Once the relative importance has been set, one of the most successful and widely adopted heuristics, namely, simulated annealing (SA), is employed to generate (sub)optimal solutions. Results show that both accuracy and efficiency of the automatically generated CNNs are better or at least comparable to the two selected state-of-the-art CNN models, ResNet50 and MobileNet, achieving as high as 93.44% classification accuracy. Ultimately, the outcome of this study fills the gap in existing CNN design methods that do not consider the tradeoff between accuracy and efficiency while providing insight into the effect varying architectures have on CNN model performance.

1. Introduction

Winter road maintenance (WRM) is a critical operation for meeting the safety and mobility needs of road users, especially for regions that reside in high latitudes. During the winter season, inclement weather events such as snow, sleet, ice, and frost lead to remarkable location and time variation in road surface conditions (RSC), which negatively affect drivers’ performance and threaten all passengers’ lives [1–3]. Due to the vast spatial distances covered by highway networks and the uncertain nature of the weather events, such variations are often hard to monitor and predict, making both WRM activities and public travel extremely challenging. Maintenance agencies are thus continuously making efforts to improve their WRM decision-making process by, for instance, deploying road weather information systems (RWIS) with individual stations equipped with advanced road weather/conditions detection sensors.

RWIS, available in both stationary and mobile forms, are deployed in many road networks around the world. RWIS can provide numerous weather data including temperature, humidity, and wind speed. Furthermore, most RWIS nowadays are equipped with cameras that provide users with a direct view of the road segment. However, determining the real-time RSC information via RWIS cameras requires well-trained and experienced maintenance personnel; and thus, this process is still being done manually. Due to the low efficiency and high cost of manual classification of RSC images, researchers have attempted to apply artificial intelligence (AI) techniques, especially convolutional neural network (CNN), to automate the process of RSC image recognition. A simplified representation of CNN architecture is
shown in Figure 1. CNN is a deep learning algorithm that labels an image into a user-defined category. It typically consists of two sections: first section (i.e., feature learning) has several filters/kernels consisting of trainable parameters which can convolve on a given image spatially to detect features, while the second section (i.e., classification) contains fully connected layers for learning the nonlinear combinations from the high-level features inputted by the previous section. [4].

Carrillo et al. [5] developed a CNN (baseline model) with a simple architecture from scratch and compared it with other pretrained CNN models in terms of their abilities to automatically classify winter RSC images (from stationary RWIS cameras). This experiment was performed using three categories: bare, partially snow covered, and fully snow covered. Results confirmed the effectiveness of CNN in determining RSC via imagery, as all CNN models produced high classification accuracies with the baseline model being the best. But the authors also identified that the results may only be indicative for their specific application. Pan et al. [6, 7] fine-tuned various state-of-the-art CNN models and trained them with stationary RWIS images, mobile RWIS images (i.e., in-vehicle images), and mixed images (i.e., stationary and mobile RWIS images together), and results of their studies showed that CNN was a promising technique for tackling the RSC recognition problems and can be helpful in assisting WRM decision-making. Several other studies [8–11] also demonstrated similar conclusions. However, there are two major limitations found in existing literatures regarding the selecting and designing of CNN models.

The first limitation is that the performance of CNNs highly relies on their architectures [12, 13]; however, there is currently no specific guideline on CNN design, not mentioning the effects of architecture on CNN performance. Typically, state-of-the-art CNNs are often manually designed with expertise in both CNNs and the investigated problems. It is therefore difficult for people, who have limited or no expertise in CNNs, to design optimal CNN architectures for their own image classification problems [12]. For this reason, transfer learning using preconstructed models (e.g., AlexNet, VGG16, and ResNet50) is popular as it provides a convenient way to directly take advantage of the preconstructed CNN models. However, their efficiencies cannot be controlled due to their complex architectures as they were not initially designed for individual specific problems [5, 6]. In other words, one cannot expect to get the optimal performance by applying the same architecture on various tasks, and CNN architecture would therefore need to be adjusted for each specific task.

As a result, there is now a surge of interest in automating the design of CNN architecture, especially for people who do not have a strong background in CNNs and/or rich domain knowledge of the investigated problem. To search for the optimal architecture of the CNN models, there are in general three strategies that can be followed: (1) random configuration inspired from literature and intuition; (2) grid search by systematically exploring the use of different architectures; and (3) heuristic algorithm looking for the optimal (or near optimal) architecture by a specific rule. Among them, random configuration lacks enough comparisons between alternative architectures. Grid search is an exhaustive operation since there are lots of hyperparameters (e.g., number of convolutional layers and learning rate) and parameters (e.g., feature maps and stride size of convolutional layers) involved in architecture design. Therefore, enumerating all use cases and selecting their optimal values will take a considerable amount of time, especially when the search space becomes bigger; this method becomes almost impossible to solve and converge. For these reasons, heuristic algorithms have gained more attention and recently been developed to automatically design model architecture without any human input [14–16].

Loussaief and Abdelkrim [16] proposed an innovative approach based on genetic algorithm (GA) to compute optimal CNN hyperparameters, including network depth, number of filters, and respective sizes in a given classification task with the objective of maximizing the classification accuracy. Wang et al. [17] utilized particle swarm optimization (PSO) to automatically search for the optimal architecture of CNNs that produced the highest classification accuracy. Within the searching process, architectures with different types and combinations of convolutional, pooling, and fully connected layers were built and compared. Similarly, Junior and Yen [18] proposed a novel algorithm based on PSO to quickly find good CNN architectures that achieved performance comparable to state-of-the-art designs. These studies showed that heuristic algorithm is a feasible approach to design CNN architecture automatically. However, a good CNN model should not be simply stacking layers to avoid the vanishing gradient (VG) and network degradation (ND) problems, especially when we want the architecture to go deeper [19]. For this reason, Sun et al. [13] applied GA in determining an optimal architecture of residual CNN with skip layers built into the blocks of convolutional layers. The experimental results indicated their proposed algorithm outperformed existing architecture design algorithms in terms of image classification accuracy.

Another major limitation is that existing literatures do not pay sufficient attention to the implementation efficiency of CNN models during the designing process. As WRM is a time-sensitive task that requires quick decision-making to guarantee the traffic safety and mobility along roadways, in order to provide a strong and efficient support for maintenance authorities [20], both classification accuracy and implementation efficiency needs to be included in the development process.

Therefore, to address these above-mentioned issues identified, this study proposes, for the first time in literature, to develop an automatic approach of designing CNN architecture tailored for RSC image recognition while considering both classification accuracy and implementation efficiency. In other words, it is the first time in literature that the automatic consideration of both accuracy and efficiency is involved in the designing process. In particular, our proposed method focuses on building a residual network without manual intervention using simulated annealing (SA) algorithm in an attempt to avoid VG and ND problems. SA is one of the
most widely used heuristic algorithms for solving combinatorial optimization problems that are difficult to solve analytically nor find exact solutions [21]. It is a probabilistic technique for approximating the global optimum by not only accepting better but also worse solutions whereby reducing the risk of falling prematurely into local optimum. Details of SA implementation, its objective function, and other settings are elaborated in the methodology part. Likewise, to objectively attest to the robustness of the method presented herein, our automatically generated CNN models are compared with two well-known state-of-the-art CNNs, namely, ResNet50 and MobileNet. To its end, this study can make the following contributions.

(i) For the first time in literature, this study fills the gap found in existing literatures by adding efficiency as another optimization objective in determining an optimal CNN architecture

(ii) Accurate and efficient CNN models developed specifically for RSC image recognition can be automatically generated, which is beneficial to the WRM decision making process

(iii) With the proposed approach developed, it can provide road maintenance authorities with an assisting tool for CNN architecture design without any specific knowledge or additional human efforts; and

(iv) Experimental results of the proposed approach can help better understand how changes in CNN architectures will affect the model performance of RSC image recognition

The remainder of this paper is organized as follows: the methodology section details our proposed approach for automatically designing a CNN architecture; afterwards, the next section presents and discusses the experimental results from our study; and lastly, all findings and contributions are summarized at the end alongside future research recommendations.

2. Methodology

In this section, the RSC image data used in this study and an overview of our proposed automatic CNN designing approach are described. Following this, residual network (ResNet), which is the basis for our proposed method, is introduced to facilitate a better understanding of why our study focuses on building a ResNet model instead of traditional CNNs. And lastly, details involved in the optimization algorithm are presented.

2.1. Experiment Data. As previously described, the main purpose of this study is to propose an innovative approach for automatically designing an optimal ResNet CNN architecture tailored for RSC image recognition. The RSC image datasets used were collected by Iowa, US’s automated vehicle location (AVL) system. The AVL is equipped with dash cameras that records RSC images along the highway every 5-10 minutes. All collected images (10,395 in total) were manually labeled into four categories, namely, bare pavement (3,786 or 36%), partially snow covered (3,166 or 31%), fully snow covered (1,327 or 13%), and undefined (2,116 or 20%). Examples of each category are shown in Figure 2.

2.2. Overview of the Proposed Approach. Figure 3 depicts a schematic diagram showing the overall procedures of our proposed approach. In general, one of the most widely adopted heuristic algorithms, namely, simulated annealing (SA), is employed to automatically search for the optimal CNN architecture among the candidate architectures [21, 22]. Within the searching process, candidate CNN architecture is generated iteratively and then fed in partial RSC images from our imagery dataset. Each candidate CNN is trained and validated using the same RSC images with 20 epochs; by doing so, the candidate model performance may not be truly reflected; however, previous literatures and experiments reveal that it is enough to obtain the major trend of each CNN architecture [23].

To determine the optimal solution, SA is set to find a CNN architecture with the highest accuracy (represented by model’s validation accuracy) and efficiency (represented by model’s FLOPs) among all candidates. Validation accuracy is the classification accuracy obtained when the developed CNN model is used to predict unseen images. It is the most important metric to measure true model accuracy. Based on our experiments and previous studies, training accuracy is nearly identical (i.e., 100%) for all candidate CNN models, while the validation accuracy has a much bigger variability. FLOPs (floating-point operations) are typically used to measure the inference time/complexity of a CNN architecture [12]. Generally, architectures with lower
FLOPs require smaller memory/space for implementation and storage. Unlike the length of training/validation time, which is heavily dependent on computer hardware and corresponding settings, FLOPs are a fixed measurement. Therefore, in this study, maximizing model efficiency can be considered equivalent to minimizing model FLOPs.

Since accuracy and efficiency are competing objectives, our proposed approach in CNN architecture design is formulated as a multiobjective optimization (MOO) problem. Methods of how it is formulated and solved are introduced in the following section. Once the optimal architecture has been determined, all RSC images are fed into it for complete training and validation (with 100 epochs); through this process, the final model performance can be assessed and used for comparison purposes. It is important to also note that all experiments done in this study used a 90/10 split; 90% of RSC images was used for training, and the remaining 10% was used for validation [9, 24]. In addition, proportions of each category in validation dataset were kept the same as the training dataset to minimize the bias of the output (i.e., the classification accuracy).

2.3. Residual Network. Much of the CNN’s success can be accredited to increasing depth of layers, which is especially true for the convolutional layers [25–27]. Essentially, as one increases layer depth, the model progressively learns more complex features, allowing for better image detection [28]. However, it is found that simply stacking additional convolutional layers would result in vanishing gradient (VG) and network degradation (ND) problems that worsen CNN model performance. VG occurs when weights of more advanced layers converge towards zero, which leads to saturation in the activation function during backpropagation.
ND refers to the phenomenon that increasing CNN architecture depth leads to diminishing accuracy values. Both of these problems were addressed in the invention of the residual learning framework or ResNet [19]. ResNet—CNN constructed using the residual learning framework—suggests that if convolutional layers are organized in residual blocks that use shortcut connections (or skip layers), vanishing learning effects can be avoided. In our approach, we incorporated the two most popular residual blocks, identity block and convolutional block (shown in Figure 4).

In the diagram above, CONV stands for the convolutional layer, while BatchNorm represents the batch normalization layer, used here to help stabilize the learning process by standardizing the inputs. ReLU stands for “rectified linear activation function,” which is considered the default activation function for many types of neural networks as it helps to streamline the training process and often achieves better performance [29]. As shown in Figure 4, the main paths for both residual blocks contain three convolutional layers; the number of filters can be customized for each of them. Typically, more filters mean more detailed information extracted from the image. For example, “Identity Block (64, 64, 256)” means the first and second convolutional layers having 64 filters while the third one having 256 filters. The difference between the identity block and convolutional block is that the latter has a convolutional and batch normalization layer in the shortcut connection. It is worth noting that, excluding the two used in this paper, there are many other types of residual blocks [30], and it is not mandatory to use the same architecture as shown in Figure 4. However, as previously mentioned, it is impossible to consider all combinations of layer/block types and parameters when designing the CNN architecture; thus, this study only uses identity and convolutional blocks in our proposed design approach. Details on how optimal architecture can be determined are found in the next subsection.

2.4. Architecture Optimization via Simulated Annealing Algorithm. The proposed architecture design approach iteratively generates candidate CNNs and selects the best one when the SA stopping criteria is met. Figure 5 depicts the procedures used in the optimization process.

2.4.1. Simulated Annealing Algorithm. Simulated annealing algorithm was created by Kirkpatrick et al. after he was inspired by the annealing procedure found in metal working [21]. And since its creation, SA has become one of the most recommended heuristic algorithms for solving combinatorial optimization problem. One notable benefit of SA is that it adopts the Metropolis criterion, which lets the algorithm search near the vicinity of the candidate solutions. This effectively prevents SA from being trapped in local minima or maxima.

To speed up the searching process, as depicted in Figure 5, we store and update the “global best” solution \( f_{\text{best}} \) along the entire process of SA. At the very beginning of the process, \( f_{\text{best}} \) is equal to the first generated solution. With every new iteration, the new solution will be compared to \( f_{\text{best}} \), and if the new solution is better than the \( f_{\text{best}} \), it will become the new \( f_{\text{best}} \). For SA to stop, one of the following criteria must be met: (a) \( f_{\text{best}} \) remains unchanged for 200 consecutive iterations, (b) the total iteration number have reached 10,000, and (c) annealing is done.

2.4.2. Generating Candidate CNN Architecture. As previously presented, the identity block and convolutional block are adopted as the basic residual blocks for building each candidate CNN architecture. In our proposed designing approach, top and bottom portions of ResNet50 are used as a fundamental structure to guarantee a certain level of model performance. On top of these structures, different number, type, and order of residual blocks are added within the optimization process. The rest of the layers involved in the classification section (i.e., fully connected layers) remains the same for all candidate CNNs, which consists of a flatten layer that converts the outputs from the previous layers into a simple vector for classification process and two ReLU-activated dense layers with 1,000 neurons each. Additionally, after each dense layer, a dropout layer with 50% dropout rate is added to avoid overfitting [7]. At the very end, a dense layer with four neurons (four RSC categories) will output 4 probability values that correspond to RSC [5, 23, 31]. Except for the changeable residual blocks (i.e., customized portion), all other layers involved in each candidate CNN architecture are the same, so we group them together as the “fixed
portion” (shown in Figure 6). Note that in Figure 6, “Dependent” means the value is dependent upon the output from the previous layer, and this number changes whenever a different architecture is generated.

In our proposed algorithm, a threshold (denoted as \( n \)) for restricting the maximum number of residual blocks, regardless of the type, that can be included in the customized portion is also set. For example, \( n = 5 \) means the customized portion will consist of no more than five residual blocks. To generate various architectures and make them evolvable along the optimization process, the customized portion of each candidate CNN architecture is assumed to have \( n \) residual blocks, while the number, type, and order of the residual blocks can be determined by sequentially assigned probabilities to each block. If the probability of the block is between 0 and 1/3, then there will be no residual block added. If the probability is between 1/3 and 2/3, an identity block will be placed in that position. If the probability is between 2/3 and 1, then a convolutional block will be placed. The sizes of the filters involved in each residual block are dependent on the previous layer/block. This idea is also borrowed from the architecture of ResNet50, but we have made a slight change that allows the whole architecture to be clearer. The first and second filter size of the current residual block are equal to half of the last filter size of the previous residual block, while the third filter size is equal to this value if it is an identity block; but if it is a convolutional block, this value is doubled. To avoid the candidate architecture being too large (hard to train), 2048 is set as the maximum filter size for all layers, which is the maximum size adopted in ResNet50. An example of a candidate CNN architecture generated in this process can be found in Figure 6.

2.4.3. Evaluating Candidate CNN Architectures. To determine the optimal architecture among all generated candidates, fitness value of each candidate needs to be evaluated. As previously mentioned, our proposed design approach is formulated as a MOO problem, where the objective function or the fitness value of each candidate CNN architecture includes their corresponding accuracy and efficiency. As depicted in Figure 3, the accuracy is represented by the experimental validation accuracy obtained by 20 epochs of training using partial RSC images (i.e., 5,000), while the efficiency is measured by FLOPs which is a fixed value once an architecture has been determined.

To incorporate these two measures into one objective function, we adopted a weighted sum method that allows investigators to choose weight values depending on their own needs [32, 33]. A higher weight value means more importance will be placed on that corresponding factor during the optimization process. Equation (1) shows the formulation of the objective function.

Minimize: 
\[
f = -w \cdot \text{normValAcc} + (1 - w) \cdot \text{normFLOPs},
\]

where \( w \) is the weight assigned to the accuracy and \( 1 - w \) is the weight assigned to the efficiency. The proposed design approach is formulated as a MOO problem with two objectives: performance and efficiency. To solve this problem, a two-objective genetic algorithm (MOGA) is adopted.
where $w$ is the weight value or the importance level (when $w = 0$, the objective function becomes solely minimizing the normFLOPs, meaning that the optimization process will not consider the accuracy piece at all, and vice versa); norm ValAcc and normFLOPs are the normalized experimental validation accuracy and FLOPs after max–min normalization method for fair comparison on an equivalent scale [34]. The fitness value of each candidate architecture is evaluated via this equation. Since we want to maximize the validation accuracy, a negative sign is added to this equation. Since we want to maximize the validation accuracy while minimizing FLOPs, a negative sign is added to this equation. Following the methods described in the previous section, all experiments and assessment of this study were implemented on Compute Canada with 32G GPU [35]; the CNN architectures were constructed using Tensorflow API [36]. By adopting the proposed architecture design approach, three CNNs were constructed by assigning different values to weight ($w$), and their corresponding experimental model performances (i.e., experimental validation accuracy and FLOPs) were obtained via the same training process through 20 epochs with 5,000 RSC images. Figure 7 depicts an example ($w = 1$) of how SA searches for the optimal solution (i.e., the architecture) by minimizing the fitness value. As can be seen from the search profile, the fitness value (orange line) changed at every single iteration, and this is because a different CNN architecture was evaluated, based on which a different fitness value was generated. The blue line representing the global best fitness value showed the changing profile of historical best fitness values that SA had ever evaluated and recorded during the searching process. Eventually, the final global best fitness value was found at 214th iteration (marked as the red dot in the figure) and then remained unchanged for the rest 200 iterations. This search profile implies that SA is successful in automatically finding the best (or at least a good enough) CNN architecture within a reasonable time frame.

Table 1 and Figure 8 show the customized portions and experimental model performances of the generated CNNs. As can be seen, the first scenario, or CNN1 ($w = 1$), only considered classification accuracy when searching for the optimal CNN architecture, which meant the candidate CNN with the highest experimental validation accuracy would be selected at the end. As this scenario did not consider model efficiency at all, two identify blocks and one convolutional block were added into the fixed portion of the architecture (as shown in Table 1). Its experimental validation accuracy (as shown in Figure 8) was the highest, while its FLOPs was the highest as well (meaning it had the lowest efficiency). By contrast, the third scenario, or CNN3 ($w = 0$), considers only the efficiency aspect when searching for the optimal architecture; hence, candidate with the lowest FLOPs would be selected. No additional residual blocks were added into the fixed portion in this scenario. As a result, it had the lowest FLOPs, meaning its efficiency was the highest, while at the same time, it had the lowest experimental validation accuracy. The second scenario, or CNN2 ($w = 0.5$), was the balanced scenario where accuracy and efficiency were considered equally important. Both its experimental accuracy and efficiency measurements were between the other two scenarios. This pattern makes intuitive sense that by setting a higher $w$, classification accuracy of the generated CNN model improved, but it did so at the cost of efficiency.

### 3. Results and Discussions

Following the methods described in the previous section, all experiments and assessment of this study were implemented on Compute Canada with 32G GPU [35]; the CNN architectures were constructed using Tensorflow API [36]. By adopting the proposed architecture design approach, three CNNs were constructed by assigning different values to weight ($w$), and their corresponding experimental model performances (i.e., experimental validation accuracy and FLOPs) were obtained via the same training process through 20 epochs with 5,000 RSC images. Figure 7 depicts an example ($w = 1$) of how SA searches for the optimal solution (i.e., the architecture) by minimizing the fitness value. As can be seen from the search profile, the fitness value (orange line) changed at every single iteration, and this is because a different CNN architecture was evaluated, based on which a different fitness value was generated. The blue line representing the global best fitness value showed the changing profile of historical best fitness values that SA had ever evaluated and recorded during the searching process. Eventually, the final global best fitness value was found at 214th iteration (marked as the red dot in the figure) and then remained unchanged for the rest 200 iterations. This search profile implies that SA is successful in automatically finding the best (or at least a good enough) CNN architecture within a reasonable time frame.

Table 1 and Figure 8 show the customized portions and experimental model performances of the generated CNNs. As can be seen, the first scenario, or CNN1 ($w = 1$), only considered classification accuracy when searching for the optimal CNN architecture, which meant the candidate CNN with the highest experimental validation accuracy would be selected at the end. As this scenario did not consider model efficiency at all, two identify blocks and one convolutional block were added into the fixed portion of the architecture (as shown in Table 1). Its experimental validation accuracy (as shown in Figure 8) was the highest, while its FLOPs was the highest as well (meaning it had the lowest efficiency). By contrast, the third scenario, or CNN3 ($w = 0$), considers only the efficiency aspect when searching for the optimal architecture; hence, candidate with the lowest FLOPs would be selected. No additional residual blocks were added into the fixed portion in this scenario. As a result, it had the lowest FLOPs, meaning its efficiency was the highest, while at the same time, it had the lowest experimental validation accuracy. The second scenario, or CNN2 ($w = 0.5$), was the balanced scenario where accuracy and efficiency were considered equally important. Both its experimental accuracy and efficiency measurements were between the other two scenarios. This pattern makes intuitive sense that by setting a higher $w$, classification accuracy of the generated CNN model improved, but it did so at the cost of efficiency.
Figure 8 also reveals that the variability of the experimental accuracy is not as big as what was observed for FLOPs, as CNN1 (having three residual blocks) only have a marginally better experimental validation accuracy (less than 1% improvement) than CNN2 (having one residual block), but their FLOPs are dramatically different (over 100%). In addition, although the maximum number of residual blocks that can be added into the customized portion is set by \( n = 5 \), the most complex architecture (i.e., CNN1) generated herein only contains three blocks, which means stacking residual blocks (i.e., more complex architecture with more residual blocks) may not necessarily produce better CNN model in terms of RSC images classification accuracy. The discovery that deeper model does not necessarily mean higher accuracy at the expense of computational efficiency is precisely why our study contributes to the better understanding of CNN. And by providing this model construction tool, people who are not CNN experts can easily create the most optimal model fit for their desired purpose.

With the architectures determined by the proposed design approach, all three models were trained completely using all RSC images; their final validation accuracies are shown in Figure 9(a). Furthermore, for comparison purposes, two state-of-the-art CNN architectures, namely ResNet50 and MobileNet, were trained using the same RSC images as well as the same number of epochs (i.e., 100). ResNet50, as introduced before, is one of the most well-known residual networks and produces very high classification accuracy [19], while MobileNet is famous for its portability and efficiency [37]. The fully connected layers and

<table>
<thead>
<tr>
<th>( w ) value</th>
<th>Block 1</th>
<th>Block 2</th>
<th>Block 3</th>
<th>Block 4</th>
<th>Block 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Identity block (128, 128, 256)</td>
<td>Identity block (128, 128, 256)</td>
<td>Convolutional block (128, 128, 512)</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>0.5</td>
<td>None</td>
<td>Convolutional block (128, 128, 512)</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>0</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

Figure 7: Example of SA searching profile.

Figure 8: Experimental model performances of the generated CNNs.

Table 1: Customized portion of the generated CNNs.
The comparisons clearly show that the CNNs automatically generated by our proposed design approach are better than or at least comparable to ResNet50 and MobileNet. Among all the models evaluated, ResNet50 has the highest FLOPs (meaning lowest efficiency), while MobileNet has the lowest value (meaning highest efficiency). CNN1 designed by placing all weight on the accuracy aspect has a better validation accuracy than both ResNet50 and MobileNet. Furthermore, CNN1 also outperformed ResNet50 in terms of efficiency as it has lower FLOPs; however, its efficiency was still lower than MobileNet. CNN2 designed with balanced weight on accuracy and efficiency also had a better final validation accuracy than ResNet50 and MobileNet, while its FLOPs were much lower than ResNet50 as well. CNN3 which focused on efficiency only in the designing process had a slightly lower classification accuracy than ResNet50 while being higher than MobileNet. In terms of FLOPs, CNN3 was less demanding than ResNet50 but still required more FLOPs than MobileNet, which is likely attributed to the mandatory fixed portion used in our proposed designing approach. It also implies that the building blocks in our design approach should be modified in future studies to investigate how building blocks and their corresponding parameters affect accuracy and efficiency. It is also interesting that CNN3 improved dramatically when one compares its final validation accuracy and its experimental one. One explanation is that CNN with simple architecture has less learning capabilities than more complex CNNs (e.g., CNN1 and CNN2). When the number of images and epochs are small, CNN3 is unable to acquire enough generalization power to make accurate classifications. In addition, the confusion matrixes show that the generated CNN models perform well in all four categories. The numbers in the diagonal line represent the validation accuracy of each category within which the predicted labels are equal to the true labels, while the numbers in the off-diagonal line are those mislabeled ratios [5]. Overall, these comparisons show that the CNNs generated from our proposed framework are better or at least comparable to the state-of-the-art CNN models in terms of both accuracy and efficiency, while they are also capable in dealing with imbalanced image data.

4. Conclusions and Future Recommendations

CNN is one of the most effective techniques that can automate RSC image recognition, which can assist the decision-making process of WRM activities. In this work, we propose a novel approach to automatically design CNN architecture tailored for RSC image recognition. In this approach, both the importance of accuracy and efficiency of the model can be adjusted by changing the weight values of the objective function (or fitness value) involved in the optimization process. Our proposed approach is the first time in literature that a designing approach is able to consider both accuracy and efficiency in automatically determining an optimal CNN architecture. In addition, residual blocks used in ResNet50 are used as the basic blocks in constructing all candidate CNN architectures, which is considered as a
Our results show that the proposed approach is able to automatically generate CNN architectures based on different weight values. The analyses undertaken in this study suggest that complex architectures tend to produce higher accuracy but sacrifice efficiency in the meantime, and simply stacking residual blocks will not necessarily improve the accuracy of the model. Through comparing between generated CNNs and two state-of-art CNNs, it is shown that our proposed approach can generate comparable or even better architectures for RSC image recognition, and they are able to deal with the imbalanced dataset as well. With its high fidelity proven herein, the proposed method will create new opportunities, particularly for people without expertise in CNNs to take advantage of automatically designing a CNN for their specific tasks, instead of spending more time using those preconstructed models that were not initially designed for them.

For future works, the building blocks can be expanded to include more types, and the proposed algorithm can be repeated to find even better architectures. More RSC images and other types of images can also be used to further attest if this proposed approach can be applied in other image recognition tasks. If so, this proposed approach can be beneficial to more people in other fields. Lastly, although automatic design is convenient and shows promising results, some of its hyperparameters and/or parameters may still need manual fine-tuning. Therefore, a sensitivity analysis of the hyperparameters and/or parameters found in the model can be an interesting topic in the future. Lastly, the constructed CNN model needs to be implemented in real production environment to test if they are accurate and efficient enough in handling real-time RSC image recognition tasks, which is one of the critical components for future autonomous transportation and smart city systems.

Data Availability

The processed image data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

Some parts of this work have been presented as a poster in “XVI World Winter Service and Road Resilience Congress”

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors’ Contributions

The authors confirm contribution to the paper as follows: study conception and design was contributed by Mingjian Wu and Tae J. Kwon; data collection and process were performed by Mingjian Wu; analysis and interpretation of results were performed by Mingjian Wu and Tae J. Kwon; draft manuscript preparation was performed by Mingjian Wu and Tae J. Kwon. All authors reviewed the results and approved the final version of the manuscript.

Acknowledgments

The authors would like to thank Tina Greenfield of Iowa DOT for providing the data used to complete this study. We would also like to thank the Aurora Project Team, Zach Hans and Neal Hawkins of Iowa State University, and Khyle Clute of Iowa DOT for providing continued support to the project. This research is funded by the Aurora Program (http://www.aurora-program.org)—an international research consortium for advancing road weather information systems technology.

References


