Article:

Ople, J.J.M., Chen, S.-F., Chen, Y.-Y. et al. (4 more authors) (2022) Controllable model compression for roadside camera depth estimation. IEEE Transactions on Intelligent Transportation Systems. ISSN 1524-9050

https://doi.org/10.1109/tits.2022.3166873

Controllable Model Compression for Roadside Camera Depth Estimation

Jose Jaena Mari Ople, Shang-Fu Chen, Yung-Yao Chen, Kai-Lung Hua, *Senior Member, IEEE,* Mohammad Hijji, Po Yang, *Senior Member, IEEE* Khan Muhammad, *Senior Member, IEEE*

***Abstract*—In the Cooperative Intelligent Transportation System (C-ITS) paradigm, vehicles could communicate with roadside units to augment their traffic knowledge. Smart roadside units could provide second-order information (e.g., vehicle count) from raw first-order data (e.g., visual feed, point clouds), and this ”smart” feature is usually provided using deep neural network models. However, implementing these useful models implies a cost for computational complexity that could hinder the future deployment of smart roadside units needed for sustainability in transportation systems. In this paper, we propose to use model compression on deep image processing models to promote its feasibility for usage in smart sensors. We formulated a controllable convolutional model compression (CCMC) algorithm that can perform filter-wise evolutionary pruning on image processing networks, along with a predefined compression ratio. CCMC is applicable for image processing networks, which have multiple possible traffic data sources (e.g., road camera surveillance). Furthermore, CCMC has a definable target compression ratio that is useful for controlling the trade-off between resource consumption and output performance. We tested our proposed method on depth estimation, which is useful for scene understanding and mapping the locations of objects in the 3D space. Our experiments show that the pruned model has minimal performance discrepancy from the original one, supporting the sustainability features needed for intelligent transportation systems.**

***Index Terms*—Smart sensors, neural network compression, depth estimation, genetic algorithm, sustainable solutions, intelligent transportation systems.**

I. INTRODUCTION

C-ITS involves multiple components (i.e., vehicles, roadside units, traffic commands centers) that communicate with each other to minimize traffic disruptions. A subset of roadside units are *sensors* that gather environment and traffic information for other components so that they can perform decisions accordingly [1]. Traditionally, the sensors utilized are camera surveillance systems that are manually observed by human operators to perform traffic management. In the C-ITS paradigm, the camera sensors can automatically provide information not only limited to visual feed but also smartly compute second-order information such as environmental structure [2]–[4], and pedestrian flow [5], [6]. The information processing is usually performed using deep neural models due to their excellent performance. However, these models have computational complexity that serves as an issue on the availability of smart roadside sensors.

Using a camera surveillance system is a common traffic management approach, which implies that there is an availability of image sensors that could be re-purposed to smart sensors. Furthermore, image processing models typically employ convolutional layers that—despite being able to automatically learn meaningful and high-level features—inherit a considerable amount of redundancy [7], [8]. The combination of hardware accessibility and model redundancy make computer vision models suitable targets for optimization.

To generate computer vision models with different levels of compression, we propose our method Controllable Convolutional Model Compression (CCMC), which can prune convolutional models until their size matches a specified compression ratio. CCMC performs two-phase compression using multiple evolutionary filter pruning. First, the initialization phase acquires a compressed model with a minimal performance drop. We use a Genetic Algorithm (GA) [9] to prune those convolutional filters that least contribute to the model's performance score. To automatically generate a model architecture with a balance of performance and compression, we compute the fitness in this phase as the weighted sum of both the model's performance score and compression ratio. Since we let the GA dictate the compression process, we may not obtain the model with the desired compression ratio. Hence in the second phase, the initially compressed model from the first phase will be further adjusted to match the compression ratio specified beforehand. We run another set of GA that optimizes two populations to search the following: (1) filter activations that have maximal performance increase and (2) filter deactivations that yield minimal performance decrease. We use the results from the second phase to loop back and adjust the initially compressed network. If the network size is *below* the compression ratio, we *activate* filters that have *maximal* performance *increase*. If the network size is *above* the compression ratio, we *deactivate* filters with a *minimal* performance *decrease*. To be more specific, in this paper, we focus on depth estimation as the computer vision task that we want to optimize. Specifically, we perform compression on monocular depth estimation networks, Monodepth2 [3]

Manuscript received December 29, 2021; Revised March 24; Accepted April 6, 2022; Published XXXX. This paper was recommended by Associate Editor XYZ. (Corresponding authors: Kai-Lung Hua and Khan Muhammad)

Jose Jaena Mari Ople, Shang-Fu Chen, Yung-Yao Chen, and Kai-Lung Hua are with the National Taiwan University of Science and Technology, Taiwan (e-mail: josmople@gmail.com, chenshungfu@gmail.com, yungyaochen@mail.ntust.edu.tw, kailunghua@gmail.com).

M. Hijji is with the Computer Science Department, Faculty of Computers and Information Technology, University of Tabuk, Tabuk 47711, Saudi Arabia (e-mail: m.hijji@ut.edu.sa).

Po Yang is with the Department of Computer Science, University of Sheffield, United Kingdom (e-mail: po.yang@sheffield.ac.uk).

Khan Muhammad is with the Visual Analytics for Knowledge Laboratory (VIS2KNOW Lab), Department of Applied Artificial Intelligence, School of Convergence, College of Computing and Informatics, Sungkyunkwan University, Seoul 03063, Republic of Korea (e-mail: khan.muhammad@ieee.org).

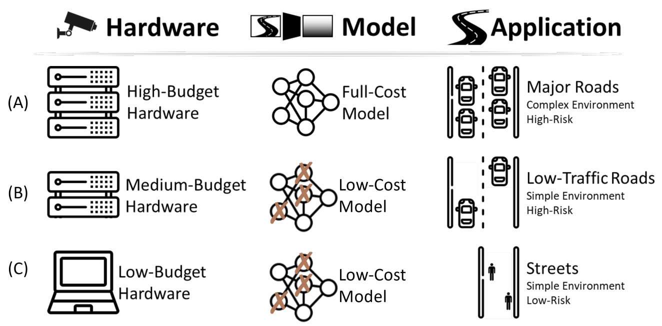


Fig. 1. Different model sizes could be used in different scenarios. (A) Ideal scenario of having a full model deployed on a high-budget hardware. (B) Low-cost model could be deployed for surveying simple road environments. (C) With low-cost model and hardware, smart surveillance could be employed on numerous minor roads and streets .

and GLPDepth [4]. We have chosen these models due to the following reasons: (1) depth estimation from a single image is a complex task that requires deep models, in which we could showcase the compression ability of CCMC, (2) monocular images are more common than other modalities that use specialized hardware (e.g., point clouds from LiDAR and stereo images). (3) In the context of a roadside camera, depth estimation offers a 3D understanding of a scene, which is useful for other traffic-related tasks (e.g., navigation [10], [11], parking management [12], and traffic flow estimation [13]). (4) Compressed depth estimation models have reduced fidelity, however, they could still be applied for smart surveillance of simple environments (e.g., few scene objects), see Figure 1.

Our main contributions can be summarized as follows:

1) We proposed a novel evolutionary filter-wise pruning algorithm for convolutional models with a controllable compression ratio that can be used as sustainable solution in transportation systems.
2) We implemented an evolutionary filter search algorithm that finds performance-optimal filter activations and deactivations.
3) We explored the effects of different levels of compression on depth estimation.
4) We conducted a series of experiments from different perspectives and our experimental results show that neural compression can reduce the computational complexity with minimal performance degradation. Pruned models could still work even if compressed to 20% of their original size, which has applications in different domains, including ITS.

II. Related Literature

The goal of our paper is to increase the availability of smart roadside units (i.e., monocular camera) with model compression. We explore depth estimation and different available compression methods.

A. Depth Estimation

Scene understanding is the process of interpreting and analyzing the 3D dynamic scene [14], and this could be done using different approaches such as semantic segmentation and depth estimation. In this paper, we focus on depth estimation towards scene understanding, in which there are multiple approaches. Point clouds from LiDAR can be used to generate depth maps [2], as well as images from stereo cameras [15]. However, these approaches use complicated and expensive equipment. With the goal of this paper to increase the availability of smart roadside units, we explore a depth estimation algorithm using a monocular camera.

The work by [16] uses a feed-forward network to estimate depth maps but its training is augmented with an additional semantic segmentation network. Another paper from [17] uses different networks to estimate low-depth and high-depth areas. Other research [18] utilized a network architecture with multiple aggregations of different convolutional layers. In [19], they proposed an unsupervised adversarial depth estimation network. Monodepth2 [3] uses multi-scale features and multiple modalities to learn depth estimation. GLPDepth [4] deploys a transformer to encode the global context and a lightweight decoder that estimates depth map while considering local connectivity.

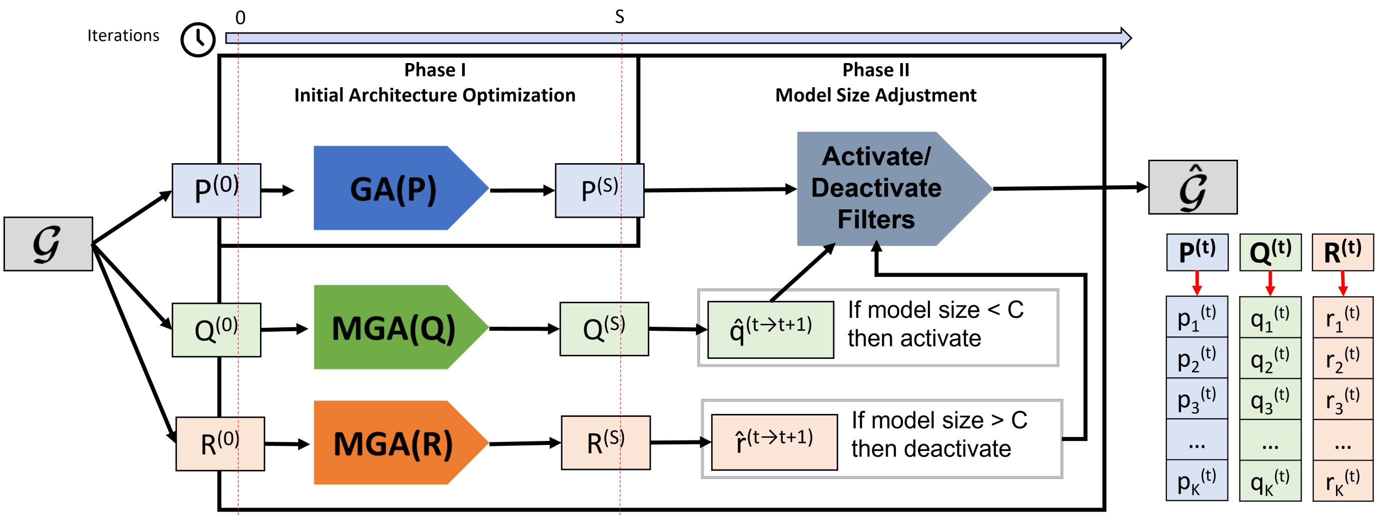
Depth estimation could be used in multiple applications for C-ITS, such as autonomous navigation [11], parking management [12], and traffic flow estimation [13]. These approaches typically use depth map estimations for generating the 3D geometry of a scene.

B. Model Compression

There are multiple approaches for model compression, including network pruning, sparse representation, bits precision, and knowledge distillation [20]. In this work, our chosen approach is network pruning, which reduces the model size of a neural network by removing some of its parameters. Existing pruning algorithms employ different search methods (e.g., random, greedy [21], gradient [22], [23], and GA [24]) for parameter removal. For example, Elkerdawy *et al.* [22] learned the pruning mask by joint optimization with the layer weights. On the other hand, we focus on GA-based approaches because of their advantages for neural architecture search, such as flexibility for navigating complex search spaces [25], and improved generalization through sparsity by evolutionary pruning [26]. An example of this approach is by Wang *et al.* [24], where they used GA to prune convolutional filters that least contribute to the overall performance of the model. To the best of our knowledge, there is no evolutionary filter-wise pruning that allows a controllable compression ratio; hence, we propose our method, CCMC.

III. The Proposed Method

CCMC can compress convolutional models using two-phase compression based on GA. The first phase is called *Initial Architecture Optimization* (Sec. III-A), which yields an initially compressed network. Whereas the second phase is *Model Size Adjustment* (Sec. III-B), which further modifies the compressed network to match the defined compression ratio (within the range $[0, 1]$). Further processing of the model is



Module	Algorithm	Objective	Fitness Function	Application
GA(P)	Genetic Algorithm	Generate architecture proposals with minimal parameter count and performance decrease	Minimize param count Minimize performance drop	Generate architecture proposals to be modified later
MGA(Q)	Microbial Genetic Algorithm	Find filter activations that cause a large performance increase	Minimize performance drop	Activate good-performing filters to optimally increase the memory sizes of proposed architectures
MGA(R)	Microbial Genetic Algorithm	Find filter activations that cause a large performance decrease	Maximize performance drop	Deactivate bad-performing filters to decrease the memory sizes of proposed architectures

Fig. 2. Overview of our proposed method, CCMC. GA is used to evolve the populations $P^{(t)}$, $Q^{(t)}$, and $R^{(t)}$, where t is the generation count. The original network $\mathcal{G}$ will be propagated to population $P^{(0)}$, $Q^{(0)}$, and $R^{(0)}$, iteration $t = 0$. Population P will evolve by minimizing both parameter count and performance drop. Q will try to find the filter changes that have the most performance improvement. R will maximize performance drop to find filter changes that provide the least improvement. After iteration S, the filter changes from Q and R are used as the heuristic to decide which filters should be activated or deactivated when the model size does not match the compression ratio C.

discussed in Sec. III-C (*Fine-Tuning the Model*). The high-level visualization of our pipeline is presented in Figure 2, while the pseudo-code is provided in Algorithm 1.

A. Initial Architecture Optimization

Given the original convolutional architecture $\mathcal{G}$, we aim to construct a compressed architecture $\hat{\mathcal{G}}$ that has fewer trainable parameters. To perform this task, we used GA [9] to optimize the architecture of our chosen network. Optimization, in this context, means the pruning of convolutional filters that least contribute to the output. We closely follow the methodology of [24] for the initial optimization of population P.

Bit String Representation. Every individual p in population P represents a solution (*chromosome*) that is manipulated by GA. This *chromosome* is a bit string (i.e., a list of 0s and 1s) that represents the architecture of a generator network. Each bit determines whether its corresponding convolution filter is discarded (if 0) or retained (if 1). Considering the network $\mathcal{G}$ with I convolutional layers, F_i is the i-th convolution layer, where $i \in [1, 2, \ldots, I]$ and $F_i \in \mathbb{R}^{H_i \times W_i \times C_i \times N_i}$. Here, H_i, W_i, C_i, and N_i are the height, width, channel size, and number of filters for the i-th layer, respectively. The total length of the bit string is defined as $\sum_{i=1}^{I} N_i$.

The input channels of the initial layer and the output channel of the final layer cannot be pruned. This is to prevent zero values as input and zero values as output. Additionally, we imposed a soft restriction in which each convolutional layer should have at least 10% of its filter active. Otherwise, a layer with less than 10% of its filter will almost output nothing. Note that this restriction is not mandatory, however, empirically, layers that output almost nothing will cause the entire model to have a bad performance.

Performance Metric. We compute the performance L of an individual p using its depth estimation accuracy at 1.25 threshold (i.e., $\delta < 1.25$). The estimation for a pixel is considered correct if the relative error δ is within the threshold (i.e., 1.25).

Fitness Function. In GA, the fitness function determines the quality of an individual p in the population P. In the context of our work, a higher fitness should reflect a better modeling performance of the compressed network. The fitness of p is computed as:

$$F(p) \;=\; L(\hat{G}|p) + \gamma(1 - N(p)), \tag{1}$$

where $\hat{\mathcal{G}}$ is the equivalent generator architecture from the bit string of individual p, $L(\hat{G})$ is the performance metric function, $N(p)$ is the compression ratio for p, and γ is the hyperparameter for balancing between performance $L(\cdot)$ and compression($N(\cdot)$. A high fitness score is achieved by maximizing performance and minimizing compression ratio.

Genetic Algorithm. Using the fitness function $F(\cdot)$ from Eq (1), GA is used to find the fittest individual p through S search generations. The roulette wheel selection is used to pick the parents for the next evolutionary generation. After S iterations, we have the latest population $P^{(S)}$, which represents the generator architectures that possess the optimal

Algorithm 1 CCMC

Require: Pretrained convolutional network $\boldsymbol{\mathcal{G}}$, Parameters: K (population size), T (number of epochs), C (Target compression ratio), S (Iterations before Q and R affects P); $S < T$

Initialize a population $P^{(0)}$ w.r.t. $\boldsymbol{\mathcal{G}}$ with K individuals;
Copy population $P^{(0)}$ to population $Q^{(0)}$ and $R^{(0)}$;
for $t = 1$ to T **do**
    Calculate the fitness of each individual in $P^{(t)}$ using Eq. 1;
    Obtain probabilities using roulette wheel selection;
    for k = 1 to K **do**
        Conduct GA's selection, crossover, and mutation on $P^{(t)}$ for generating new individuals according to a pre-defined probability;
    end for
    Calculate the fitness of each individual in $Q^{(t)}$ using Eq. 2;
    for k = 1 to K **do**
        Conduct Microbial GA's selection, evaluation, recombination, mutation on $Q^{(t)}$ for generating new individuals according to a pre-defined probability;
    end for
    Find out which individual in $Q^{(t)}$ gained the most fitness after Microbial GA and record in $\hat{Q}$
    Calculate the fitness of each individual in $R^{(t)}$ using Eq. 3;
    for k = 1 to K **do**
        Conduct Microbial GA's selection, evaluation, recombination, mutation on $R^{(t)}$ for generating new individuals according to a pre-defined probability;
    end for
    Find out which individual in $R^{(t)}$ gained the most fitness after Microbial GA and record in $\hat{R}$, see Figure 3;
    if $t > S$ **then**
        if $N(p^{(t)}) > \dfrac{N(G)}{(C * (1 - 2\%))}$ **then**
            Check every convolution filter state in each individual $p^{(t)}$ and deactivate those recorded in $\hat{R}$;
        else if $N(p^{(t)}) < \dfrac{N(G)}{(C * (1 + 2\%))}$ **then**
            Check every convolution filter state in each individual $p^{(t)}$ and reactivate those recorded in $\hat{Q}$;
        end if
    end if
end for
Update fitnesses of individuals in P_t
Establish compressed Generator $\hat{G}$ by choosing the best individual $\hat{p}^{(T)}$

Ensure: Compressed Generator $\hat{G}$ after fine-tuning using the entire training set.

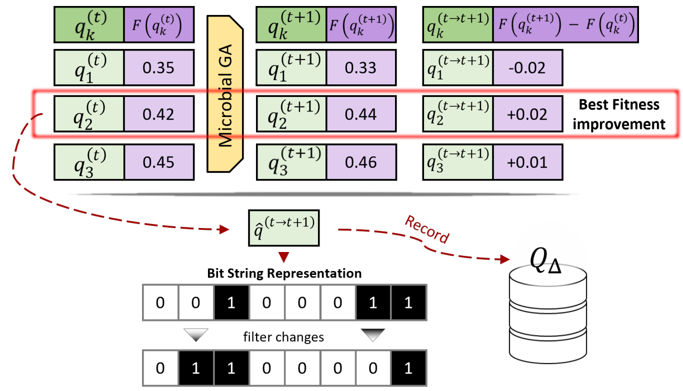


Fig. 3. Model size adjustment using population Q. Individual $q_k^{(t)}$ is the k-th member of population Q at evolution step t. $F(\cdot)$ is the fitness function defined in Eq. (2). Filter changes of $\hat{q}^{(t \to t+1)}$ with the best fitness improvement will be recorded to cache Q_Δ. Items in Q_Δ are used to adjust the model size of the generator network. We use MGA since it improves the population via self-mutation, which allows one-to-one correspondence to the succeeding generations.

ratio between model size and performance, as per the selected value of γ.

B. Model Size Adjustment

Even after the initial optimization (Sec. III-A), it is not guaranteed that the final architectures (i.e., $P^{(S)}$) has the specified compression ratio. After the S-th generation of the first phase, the network architecture is further modified by either activating filters if the model size is below the compression ratio or deactivating filters if the model size is above the ratio. To do this task, we use Microbial Genetic Algorithm (MGA) to optimize additional populations, Q and R, as guidance for further filter changes. We use MGA instead of GA because we want to record the gradual development of the individuals of populations Q and R, since MGA improves upon the population via self-mutation. Specifically, the individual $q_i^{(s)}$ should correspond to the evolved individual $q_i^{(s+1)}$ (this should also apply to the individuals of p). In MGA, the individual is modified by accepting genes from other individuals with higher fitness; therefore, there is a correspondence between individuals of succeeding generations. For GA, this does not hold true.

Increase model size. We define population Q whose purpose is to find filter activations with maximal performance improvement. The fitness function for Q is defined as:

$$F_{pos}(p) = L(\hat{\boldsymbol{\mathcal{G}}}|p). \tag{2}$$

F_{pos} is similar to Eq. (1) but has no regard for the compression ratio. As illustrated in Figure 2, we populate the cache Q_Δ by finding the filter activations (i.e., $0 \to 1$ in the bit string representation) with the best fitness increase for each evolutionary generation of Q. For each item in Q_Δ, if there is a similar configuration in $P^{(S)}$, we perform that filter change to increase the model sizes of each individual in $P^{(S)}$.

TABLE I
QUANTITATIVE RESULTS OF CCMC IN COMPRESSING DEPTH ESTIMATION MODELS

Model		Compression			Performance Metrics						
Arch.	Dataset	Ratio	Actual Ratio	Actual Size	Abs REL	Sq REL	RMSE	RMSE log	$\delta < 1.25^1$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	Cityscapes	0.8	0.8079	9.96 MB	0.0477	0.0001	0.0027	0.0681	0.9862	0.9987	0.9999
		0.6	0.6196	7.64 MB	0.0494	0.0001	0.0026	0.0781	0.9733	0.9996	0.9999
		0.4	0.4198	5.18 MB	0.0547	0.0002	0.0028	0.0845	0.9680	0.9985	0.9998
		0.2	0.2200	2.71 MB	0.0684	0.0003	0.0040	0.0990	0.9561	0.9953	0.9998
	CityCam	0.8	0.8013	9.88 MB	0.0545	0.0002	0.0030	0.753	0.9821	0.9992	0.9998
		0.6	0.6134	7.56 MB	0.0560	0.0002	0.0032	0.0803	0.9789	0.9993	0.9997
		0.4	0.4086	5.04 MB	0.0581	0.0002	0.0033	0.0888	0.9664	0.9979	0.9996
		0.2	0.2141	2.64 MB	0.0628	0.0002	0.0031	0.0914	0.9660	0.9971	0.9996
GLPDepth	Cityscapes	0.8	0.8132	192.29 MB	0.0833	0.0375	0.3550	0.1116	0.9823	0.9989	0.9999
		0.6	0.6054	143.16 MB	0.0570	0.0140	0.2008	0.0756	0.9457	0.9978	0.9998
		0.4	0.4113	97.26 MB	0.0992	0.0529	0.4289	0.1417	0.9073	0.9808	0.9976
		0.2	0.2189	51.76 MB	0.1172	0.0741	0.4998	0.1660	0.8872	0.9700	0.9953
	CityCam	0.8	0.8172	193.24 MB	0.0166	0.0032	0.1265	0.0267	0.9998	0.9999	0.9999
		0.6	0.6163	145.73 MB	0.0308	0.0075	0.1911	0.0407	0.9992	0.9999	0.9999
		0.4	0.4049	95.74 MB	0.0584	0.0245	0.2990	0.0795	0.9791	0.9998	0.9998
		0.2	0.2177	51.48 MB	0.0585	0.0252	0.3130	0.0817	0.9788	0.9998	0.9998

Decrease model size. We define population R whose purpose is to find filter deactivations with minimal performance improvement. The fitness function for R is defined as:

$$F_{neg}(p) = \frac{1}{L(\hat{\mathcal{G}}|p)}. \tag{3}$$

F_{neg} encourages the network to have worse performance than the original convolutional network $\mathcal{G}$. We populate the cache $R_\triangle$ by following a similar process in Figure 2 but using a different fitness function. Since R is optimizing for architecture with worse performance, the filter activations found in $R_\triangle$ have a negative contribution to the overall performance $L(p)$ for individuals p in $P^{(S)}$. Therefore, for each item in $R_\triangle$, if there is a similar configuration in $P^{(S)}$, we deactivate (i.e., $1 \rightarrow 0$) the filter to reduce its model size.

C. Fine-tuning the Model

After finding a suitable compressed network architecture $\hat{\mathcal{G}}$ that has good performance and satisfies the compression ratio, we fine-tune $\hat{\mathcal{G}}$ using the training dataset of the initial network $\mathcal{G}$. The original weights of $\mathcal{G}$ are preserved and transferred to $\hat{\mathcal{G}}$ but the removed parameters negatively affect the performance. We fine-tune the network $\hat{\mathcal{G}}$ using a subset from the original training dataset. The network is trained using different numbers of fine-tune batches, in which the batch size is based on the original training procedures of the generator.

IV. EXPERIMENTS

In this section, we qualitatively and quantitatively evaluate our proposed GA-based compression, CCMC, on depth estimation task.

A. Implementation Details

The models and the GA pipeline are implemented in PyTorch. For the fine-tuning step, the pruned models are trained with the same configurations as their respective original networks. The fine-tuning is performed with the configurations stated in their respective implementations. The experiments are run on a computer with NVIDIA GeForce GTX 2080 Ti GPU and Intel Core i7-8700 CPU.

B. Models.

Using CCMC, we compress the depth estimation models, Monodepth2 [3] and GLPDepth [4], to specific compression ratios (i.e., 0.8, 0.6, 0.4, 0.2). For Monodepth2, we used the model trained on the KITTI dataset [27] with the monocular modality of resolution 640×192. For GLPDepth, we used the model trained on [28]. Do note that we fine-tune the pruned architectures derived from the original pretrained models.

C. Dataset.

The testing is performed using Cityscapes [29] and CityCam datasets [30]. To compute performance metric $L(\hat{\mathcal{G}}|p)$ in the fitness formulas (Eq. 1,2,3), we use a randomly selected 10-item subset from their test dataset. Only 10 items were chosen for fast computation of the fitness since it will run for multiple instances of pruned architectures. For computing the depth metrics, the entire test images are used. For both datasets, we post-process their images such that the longer side has a length of 640 pixels while also preserving the aspect ratio.

Cityscapes [29] is a dataset that focuses on semantic understanding of urban street scenes. This dataset has similarities with the training dataset [27], which is, both of them are from

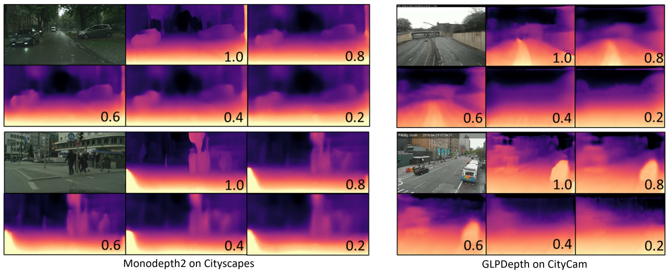


Fig. 4. Examples of generated depth maps. (Left) Depth maps produced by Monodepth2 models [3] on Cityscapes [29] dataset. (Right) Depth maps produced by GLPDepth models [4] on CityCam [30] dataset. The compression ratio is indicated by the numbers at the lower right of each image. The upper-left image from each image group is the input. The ratio of 1.0 indicates that this is the output of the original network.

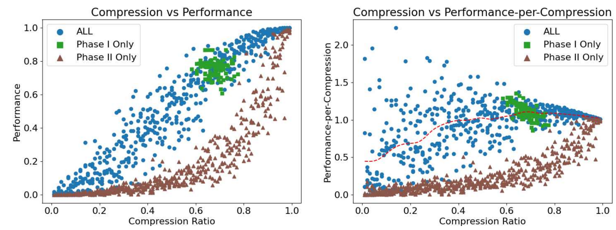


Fig. 5. Statistics of the proposed solutions. (Left) Compression (x-axis) vs Performance (y-axis). (Right) Compression (x-axis) vs Performance-per-Compression (y-axis). *Blue dots* indicate proposed solutions of the entire pipeline (Phase I and Phase II combined). *Green boxes* indicate outputs of Phase I. *Brown triangles* indicate generated architectures of Phase II but with a random population as input—Phase II only. The red dashed line (at the graph on the right) indicates average performance for *Blue dots*. Best viewed in color.

the point of view of the car. Cityscapes have stereo images (i.e., left and right) but we just used the left test images for the evaluation metrics. Images in this dataset have a resolution of 2048×1024.

CityCam [30] contains images from the point-of-view of road surveillance cameras. This dataset fits more with the desired use-case of this paper (i.e., more smart roadside units). The original image resolution for this dataset is 352×240.

D. Metrics

We use common evaluation metrics for depth estimation such as Absolute Relative Difference (Abs Rel), Squared Relative Difference (Sq Rel), Root Mean Squared Error (RMSE), and accuracy with certain thresholds. For accuracy, we consider the depth estimation for a pixel to be correct if its error ($\delta = max(\frac{pred}{gt}, \frac{gt}{pred})$) is within a certain threshold. Specifically, the thresholds are 1.25^1, 1.25^2, and 1.25^3—these values are standard for depth estimation. The comparison is performed relative to the original network. To reduce verbosity, we refer to the compression ratio as C (e.g., $0.8C$ is 80% compression ratio), from hereon.

E. Performance

In this section, we show the quantitative and qualitative results of the model pruned to different compression ratios. We compressed the models with the architectures from Monodepth2 [3] and GLPDepth [4], then used these pruned models to perform depth estimation on the datasets, Cityscapes [29] and CityCam [30]. View the quantitative results in Table I and the qualitative results view Figure 4.

Based on the table, the obvious trend is that performance degrades as the compression ratio is minimized. However, we could see that outputs of the pruned model have minimal discrepancy from the original network. If we look at Figure 4, the details of the depth map become blurrier with the lower target compression size. But still, the general outline of the depth map remains.

Do note that the performance of the compressed model is derived from the original model. If the base model has a bad performance to a dataset, this will carry over to the pruned model.

F. Memory

In Table I (Compression columns), we can see the actual memory sizes of the pruned models used in the evaluation. We could see that the actual compression ratio is not exactly the same as the specified ratio. This is for the following reasons. Evolutionary algorithms will take too long to generate the desired filter activations with a specific number of activations; hence, there is a margin of error for the compression ratio. In this case, we set the margin of error as 0.02. The final size of the pruned model still fits the specified ratio.

G. Ablation Study

We graphed the outputs of proposed solutions generated by different configurations (i.e., Phase I only, Phase II only, and full pipeline) in Figure 5. On the left side of the figure, the compression (x-axis) and the performance (y-axis) values of the proposed solutions present an upward trend. It is not immediately obvious why the outputs of Phase I (*green boxes*) are located around 0.6 to 0.8 compression values. However, their coordinates make sense, if we look at the right side of the figure, which graphs the compression (x-axis) and the performance-per-compression (y-axis) scores. There is a noticeable apex, hovering around 0.6 to 0.7. In other words, Phase I outputs efficient pruned model architectures characterized by high values for performance-per-compression scores. On the other hand, Phase II allows the initial solutions (i.e., Phase I outputs) to be adjusted to other compression ratios, albeit with a sacrifice on the performance-per-compression efficiency. If we don't use Phase I proposals as Phase II inputs, the generated solutions for Phase II have lower performance (see *brown triangles*). Additionally, in Figure 5 (left), for compression ratios less than 0.2, the actual performance of the proposals hovers closely to zero.

H. Comparison with Other Compression Methods

To the best of our knowledge, we are the first paper addressing controllable compression for depth estimation models. However, a previous work [22] already exists if limited only for general model pruning. We compared our methods and found that we could have comparable performance at the following compression ratios: 0.15 for [22] and 0.28 (Ours). We hypothesize that their improved performance is due to the joint learning of the weights and pruning mask. For our method, we find the pruning mask first, then perform fine-tuning.

V. Conclusion

We proposed Controllable Convolutional Model Compression (CCMC) that can perform evolutionary filter-wise pruning to computer vision model, in which the desired compression size can be specified. With CCMC, we increased the deployability of image processing models so that more "smart" roadside sensors could be established. CCMC works by using a Genetic Algorithm (GA) to generate an initial pruned architecture with a balance between performance and compression. Then CCMC employs two additional GA that attempts to perform the objectives: search for (GA.1) filter activations that have maximal performance increase and (GA.2) filter deactivations that have minimal performance decrease. With these GA pipelines, we achieved efficient pruning of the base model. Furthermore, we directly controlled the model size of the final output by committing the findings of GA.1 for upsizing or GA.2 for downsizing. We tested our method on monocular depth estimation models and found that CCMC can generate pruned depth estimation models with minimal performance discrepancy from the original model. Furthermore, even with 20% compression, the pruned depth estimator could still somewhat work, albeit only on the coarse-level estimation. In addition to reducing hardware requirements for the deployment of "smart" roadside sensors, CCMC partially solved the hardware compatibility issue for model sharing between C-ITS agents.

References

[1] Z. Lv, L. Qiao, and I. You, "6g-enabled network in box for internet of connected vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[2] L. Chen, Y. He, J. Chen, Q. Li, and Q. Zou, "Transforming a 3-d lidar point cloud into a 2-d dense depth map through a parameter self-adaptive framework," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 1, pp. 165–176, 2016.

[3] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3828–3838.

[4] D. Kim, W. Ga, P. Ahn, D. Joo, S. Chun, and J. Kim, "Global-local path networks for monocular depth estimation with vertical cutdepth," *CoRR*, vol. abs/2201.07436, 2022. [Online]. Available: https://arxiv.org/abs/2201.07436

[5] P. Yang, G. Zhang, L. Wang, L. Xu, Q. Deng, and M.-H. Yang, "A part-aware multi-scale fully convolutional network for pedestrian detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 1125–1137, 2020.

[6] B. Han, Y. Wang, Z. Yang, and X. Gao, "Small-scale pedestrian detection based on deep neural network," *IEEE transactions on intelligent transportation systems*, vol. 21, no. 7, pp. 3046–3055, 2019.

[7] K. Kahatapitiya and R. Rodrigo, "Exploiting the redundancy in convolutional filters for parameter reduction," in *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 1410–1420.

[8] S. Chakraborty, S. Paul, R. Sarkar, and M. Nasipuri, "Feature map reduction in cnn for handwritten digit recognition," in *Recent Developments in Machine Learning and Data Analytics*. Springer, 2019, pp. 143–148.

[9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[10] Z. Lv, D. Chen, H. Feng, H. Zhu, and H. Lv, "Digital twins in unmanned aerial vehicles for rapid medical resource delivery in epidemics," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[11] R. de Queiroz Mendes, E. G. Ribeiro, N. dos Santos Rosa, and V. Grassi Jr, "On deep learning techniques to boost monocular depth estimation for autonomous navigation," *Robotics and Autonomous Systems*, vol. 136, p. 103701, 2021.

[12] M.-R. Lee and D.-T. Lin, "Vehicle counting based on a stereo vision depth maps for parking management," *Multimedia Tools and Applications*, vol. 78, no. 6, pp. 6827–6846, 2019.

[13] F. Brickwedde, S. Abraham, and R. Mester, "Mono-sf: Multi-view geometry meets single-view depth for monocular scene flow estimation of dynamic traffic scenes," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2780–2790.

[14] Z. Lv, Y. Li, H. Feng, and H. Lv, "Deep learning for security in digital twins of cooperative intelligent transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[15] W. Chuah, R. Tennakoon, R. Hoseinnezhad, and A. Bab-Hadiashar, "Deep learning-based incorporation of planar constraints for robust stereo depth estimation in autonomous vehicle applications," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[16] P.-Y. Chen, A. H. Liu, Y.-C. Liu, and Y.-C. F. Wang, "Towards scene understanding: Unsupervised monocular depth estimation with semantic-aware representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2624–2632.

[17] H. Ren, M. El-Khamy, and J. Lee, "Deep robust single image depth estimation neural network using scene understanding." in *CVPR Workshops*, 2019, pp. 37–45.

[18] W. Su, H. Zhang, Q. Zhou, W. Yang, and Z. Wang, "Monocular depth estimation using information exchange network," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[19] A. Pilzer, D. Xu, M. Puscas, E. Ricci, and N. Sebe, "Unsupervised adversarial depth estimation using cycled generative networks," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 587–595.

[20] R. Mishra, H. P. Gupta, and T. Dutta, "A survey on deep neural network compression: Challenges, overview, and solutions," *CoRR*, vol. abs/2010.03954, 2020. [Online]. Available: https://arxiv.org/abs/2010.03954

[21] J. Yu and T. S. Huang, "Universally slimmable networks and improved training techniques," in *International Conference on Computer Vision*, 2019, pp. 1803–1811.

[22] S. Elkerdawy, H. Zhang, and N. Ray, "Lightweight monocular depth estimation model by joint end-to-end filter pruning," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 4290–4294.

[23] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 734–10 742.

[24] Y. Wang, C. Xu, J. Qiu, C. Xu, and D. Tao, "Towards evolutionary compression," in *24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2476–2485.

[25] A. D. Martinez, J. Del Ser, E. Villar-Rodriguez, E. Osaba, J. Poyatos, S. Tabik, D. Molina, and F. Herrera, "Lights and shadows in evolutionary deep learning: Taxonomy, critical methodological analysis, cases of study, learned lessons, recommendations and challenges," *Information Fusion*, vol. 67, pp. 161–194, 2021.

[26] R. C. Gerum, A. Erpenbeck, P. Krauss, and A. Schilling, "Sparsity through evolutionary pruning prevents neuronal networks from overfitting," *Neural Networks*, vol. 128, pp. 305–312, 2020.

[27] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[28] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[29] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[30] S. Zhang, G. Wu, J. P. Costeira, and J. M. Moura, "Understanding traffic density from large-scale web camera data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5898–5907.

Jose Jaena Mari Ople received his B.S. degree in Computer Science from De La Salle University, Philippines, in 2018, and an M.S. degree from National Taiwan University of Science and Technology (NTUST), in 2020. He is currently pursuing a Ph.D. degree with the Department of Computer Science and Information Engineering, NTUST. His research interests include digital image processing and deep learning applied to computer vision.

Shang-Fu Chen received his B.S. degree from Department of Computer Science and Information Engineering in National Taiwan University of Science and Technology (NTUST), on 2021. He is currently pursuing a M.S. degree with the Department of Computer Science and Information Engineering in NTUST. His research interests include image processing and deep learning applied to computer vision.

Yung-Yao Chen received his B.S. (2004) and M.S. (2006) degrees in Electrical and Control Engineering from the National Chiao Tung University from Hsinchu, Taiwan, and his Ph.D. (2013) degree in Electrical Engineering from Purdue University, USA. Before being a faculty, he has worked in HP Labs - Printing and Content Delivery Lab (HPL - PCDL) about one year. He is currently an Associate Professor in the Department of Electronic and Computer Engineering and the Co-Director of Taiwan Tech Smart Electric Vehicle Research Center, National Taiwan University of Science and Technology, Taipei, Taiwan. His current research interests include vision-based automation, automated/wisdom factory, self-driving car, and human-computer interaction. Dr. Chen was the recipient of the Best Paper Award of the International conference on Advanced Robotics and Intelligent Systems (2015 and 2020), the Rotary Foundational Scholarship, and the Ta-Yu Wu Memorial Award from Taiwan's Ministry of Science and Technology (MOST). He is a member of Golden Key International Honor Society and Phi Tau Phi.

Kai-Lung Hua received the B.S. degree in electrical engineering from National Tsing Hua University in 2000, and the M.S. degree in communication engineering from National Chiao Tung University in 2002, both in Hsinchu, Taiwan. He received the Ph.D. degree from the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, in 2010. Since 2010, Dr. Hua has been with the National Taiwan University of Science and Technology, where he is currently a professor in the Department of Computer Science and Information Engineering. He has also been the vice dean of the College of Electrical Engineering and Computer Science since 2019. He is the director of the Artificial Intelligence Research Center. Dr. Hua is a member of Eta Kappa Nu and Phi Tau Phi, as well as a recipient of MediaTek Doctoral Fellowship. His current research interests include digital image and video processing, computer vision, and machine learning. He has received several research awards, including the 2019 Outstanding Research Award of Taiwan Tech, 2018 Young Scholar Award of Taiwan Tech, Top Performance Award of 2017 ACM Multimedia Grand Challenges, Top 10% Paper Award of 2015 IEEE International Workshop on Multimedia Signal Processing, the Second Award of the 2014 ACM Multimedia Grand Challenge, the Best Paper Award of the 2013 IEEE International Symposium on Consumer Electronics, and the Best Poster Paper Award of the 2012 International Conference on 3D Systems and Applications.

Mohammad Hijji [Member, IEEE] is currently an Assistant Professor with the Faculty of Computers and Information Technology, University of Tabuk, Saudi Arabia. He is also the Chairman of the Computer Science Department. He works with a range of research centers, government sectors and global companies related to "Artificial Intelligence", "Smart Cities", and "Disaster/Emergency Management". He is also responsible for developing and managing postgraduate programs with the Faculty of Computers and Information Technology, University of Tabuk. His research interests include Artificial Intelligence, Cyber Security, Internet of Things (IoT), and Smart City.

Po Yang [Senior Member, IEEE] is a Senior Lecturer in Large Scale Data Fusion in the Department of Computer Science at the University of Sheffield. He graduated with a BSc (Hons) in Computer Science from Wuhan University in China in 2004, before being awarded his MSc in Computer Science from the University of Bristol in 2006. In 2010 he graduated with a PhD in Electronic Engineering from the University of Staffordshire.

From February 2015 to July 2019, he was a Senior Lecturer in Computer Science at Liverpool John Moores University. He worked as a Post-doc Research Fellow in Computer Science at the University of Bedfordshire from January 2012 to January 2015. Previously, he has also held the positions of Research Associate in Computer Science at the University of Teeside from September 2008 to February 2010, a Research Assistant in image processing at the University of Salford from March 2010 to December 2011. Since 2006 he has generated over 90 international journal and conference papers in the fields of Pervasive Healthcare, Image Processing, Parallel Computing and RFID related internet of things (IoT) applications.

He serves as an Associate Editor in IEEE Journal of Translational Engineering in Health and Medicine and IEEE Access.He has over 12 years full time research experience in computing areas (recent three years working on Pervasive Healthcare), which includes the key participation and local leadership of 6 EU funded projects CALLAS (RA in Affective Computing at Teeside University), IMPACT (RA in Image Processing at Salford University), GPSME, DRINVENTOR, MHA and CHIC (RF in Computer Science at Bedfordshire University) and 3 EPSRC/TSB funded projects.

Khan Muhammad [S'16, M'18, SM'22] received his PhD degree in Digital Contents from Sejong University, Republic of Korea in February 2019. He was an Assistant Professor at the Department of Software, Sejong University from March 2019 to February 2022. He is currently the director of Visual Analytics for Knowledge Laboratory (VIS2KNOW Lab) and an Assistant Professor (Tenure-Track) with the Department of Applied AI, School of Convergence, College of Computing and Informatics, Sungkyunkwan University, Seoul, Republic of Korea. His research interests include intelligent video surveillance, medical image analysis, information security, video summarization, multimedia data analysis, computer vision, IoT/IoMT, and smart cities. He has registered 10 patents and has contributed 200+ papers in peer-reviewed journals and conference proceedings in his areas of research. He is an Associate Editor/Editorial Board Member of more than 14 journals. He is among the highly cited researchers in 2021 according to the Web of Science.