Multidimensional Systems and Signal Processing (2021) 32:239–262
https://doi.org/10.1007/s11045-020-00736-x

CLMIP: cross-layer manifold invariance based pruning method of deep convolutional neural network for real-time road type recognition

Qinghe Zheng[1] · Xinyu Tian[2] · Mingqiang Yang[1] · Huake Su[3]

Received: 22 June 2019 / Revised: 12 June 2020 / Accepted: 30 June 2020 / Published online: 17 July 2020

Abstract
Recently, deep learning based models have demonstrated the superiority in a variety of visual tasks like object detection and instance segmentation. In practical applications, deploying advanced networks into real-time applications such as autonomous driving is still challenging due to expensive computational cost and memory footprint. In this paper, to reduce the size of deep convolutional neural network (CNN) and accelerate its reasoning, we propose a cross-layer manifold invariance based pruning method named CLMIP for network compression to help it complete real-time road type recognition in low-cost vision system. Manifolds are higher-dimensional analogues of curves and surfaces, which can be self-organized to reflect the data distribution and characterize the relationship between data. Therefore, we hope to guarantee the generalization ability of deep CNN by maintaining the consistency of the data manifolds of each layer in the network, and then remove the parameters with less influence on the manifold structure. Therefore, CLMIP can be regarded as a tool to further investigate the dependence of model structure on network optimization and generalization. To the best of our knowledge, this is the first time to prune deep CNN based on the invariance of data manifolds. During experimental process, we use the python based keyword crawler program to collect 102 first-view videos of car cameras, including 137 200 images (320×240) of four road scenes (urban road, off-road, trunk road and motorway). Finally, the classification results have demonstrated that CLMIP can achieve state-of-the-art performance with a speed of 26 FPS on NVIDIA Jetson Nano.

Keywords Deep learning · Road type recognition · Model pruning · Model compression · Inference speedup

✉ Mingqiang Yang
imageinstitute@outlook.com

1 School of Information Science and Engineering, Shandong University, Qingdao 266237, China

2 College of Mechanical and Electrical Engineering, Shandong Management University, Jinan 250357, China

3 School of Microelectronics, Xidian University, Xian 710126, China

1 Introduction

Recently, a series of computer vision based intelligent vehicle systems have been developed to provide driver assistance and autonomous navigation, including the techniques of path planning (Jeong et al. 2018), traffic sign classification (Guan et al. 2018), road type recognition (Zhang et al. 2017), obstacle detection (Nguyen et al. 2017), and lane detection (Ozgunalp et al. 2017). In all cases, the system is primarily designed for enhancing the drivers' overall situational awareness, reducing driver workload and automating whole or partial driving process. As one of the most important cornerstone functions, road type recognition based on computer vision is defined as the process of determining road types according to the video content of the scene. This task is an important step in understanding road environment and is necessary for overall situational awareness and fully or semi-automatic driving system (Menouar et al. 2017). In such applications, it is essential to mine domain information and implicit expert knowledge. However, extracting specialized features from the perception of various road scenes is a challenge, which requires high quality image or video processing methods.

Over the past few decades, a lot of research and studies have been made to the field of computer visual based autopilot (Trittler et al. 2016), including the improvement of optimization methods for nonconvex models (Miguel et al. 2015), the development of regularization algorithms for improving generalization capabilities (Neyshabur et al. 2017; Rajeswaran et al. 2017) and the introduction of data augmentation methods for expanding training sets (Ding et al. 2016; Lemley et al. 2017), etc. These techniques help to move machine learning model toward real applications. As for road type classification problem, many high-level representations (Tang and Breckon 2011; Mioulet et al. 2013; Mohammad et al. 2015) that combine low-level features of color, texture, and shape have been designed and fed into classifiers to determine the types of road environments in images, such as off-road and motorway. The traditional low-level features do not have the adaptability to variable factors such as weather, illumination, angle of view, lens jitter and so on, which may bring challenges for the model. Nonetheless, building a robust and efficient road type recognition system remains an open problem.

Recently, deep learning has flourished and achieved significant progress in various computer vision related tasks, such as object detection and classification (Chen et al. 2015) and instance segmentation (Li et al. 2016). The over-parametrized deep neural networks that were trained on huge datasets have strong fitting capability, which can effectively resist noise and accurately learn the representations of samples. If the pruned deep neural network is applied to real-time vision-based road type recognition system, significant improvements may be achieved. However, the application of deep learning in real-time road type recognition needs to face the following problems: (1) Lack of large-scale datasets like ImageNet (Russakovsky et al. 2015) for model optimization; (2) The large size of deep neural network leads to large memory usage and slow reasoning speed, which make it difficult to apply in embedded devices. In order to apply deep neural networks to practical applications, a series of pruning methods have been developed to reduce their memory footprint and accelerate the reasoning of models. Pruning means to remove redundant network parameters that contribute little to the final output based on specific criteria, such as magnitude, gradient and activation value. Pruning low-contributing neurons or connections leads to certain accuracy loss, so pruned neural networks usually need to be retrained to ensure classification performance.

In this paper, we propose a cross-layer manifold invariance based pruning method called CLMIP for deep CNN compression and acceleration to help complete real-time road type

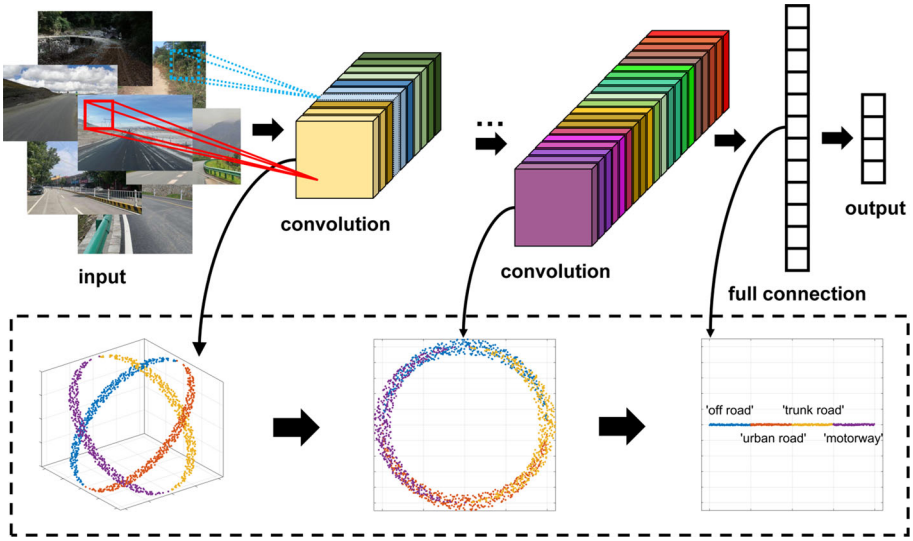


Fig. 1 Data manifold invariance based CLMIP method in a deep CNN. Manifold structures of different layers are captured and used to ensure the generalization ability of pruned networks

recognition in a low-cost vision system based on monocular camera. In the forward propagation process of neural networks, the output of convolutional and fully connected layers can be regarded as the indirect representation of data in various dimensions, containing the implicit knowledge of samples and represented in the form of features. They can be visually expressed by observing the shape of manifolds. Manifolds are higher-dimensional analogues of curves and surfaces, which can be self-organized to reflect the data distribution and characterize the relationship between data. Therefore, we hope to guarantee the generalization ability of deep CNN by maintaining the consistency of the data manifolds of each layer in the network, and then remove the parameters that have less influence on the manifold structure. Specifically, we try to reduce the reconstruction error of the data manifolds before and after pruning to ensure the network's ability to capture data distribution and utilize this as a standard to measure the contribution of parameters in the network, as shown in Fig. 1. Finally, redundant parameters with less contribution are removed and the pruned model is used for road type classification. To the best of our knowledge, this is the first time to prune deep CNN based on the invariance of data manifolds. The main contributions of this paper are summarized as follows.

- The proposed CLMIP method reduces the size (or memory footprint) of deep neural network for real-time road type recognition, so it can be deployed on the embedded device.
- CLMIP speeds up the inference of deep neural network and enables it to be used in the real-time automatic driving system, helping drivers understand the road environments and make corresponding driving decisions.
- It is generic and independent of network structures, so it can be easily extended to other complex models, such as GoogLeNet (Szegedy 2015) containing inception blocks and ResNet (He et al. 2016) with residual modules.
- Experiments on a video set containing four road types demonstrate the effectiveness of the pruned neural network, which achieves state-of-the-art classification results with a prediction speed of 26 frames per second.

The remainder of this paper is organized as follows. Section 2 gives a brief review to the related work of road type recognition and pruning methods for network compression and acceleration. Section 3 introduces the proposed cross-layer manifold invariance based pruning method. Extensive experiments and analysis are presented in Sect. 4. Finally, we conclude our work and propose future directions in Sect. 5.

2 Related work

In this section, we first present the difficulties, challenges and related methods for road type recognition in Sect. 2.1, and then introduce the pruning methods to accelerate the reasoning of deep CNNs and accomplish the real-time road type recognition in Sect. 2.2.

2.1 Road type recognition methods

Road type recognition faces the challenge of achieving high performance while ensuring computational efficiency. Thus, the simple and efficient representations of road environment is crucial. Previous work in the road type recognition field has studied the extraction of low-level features such as color distribution (Jansen et al. 2005) as the metric to classify different off-road terrain into approximate categories, whereas the other work in this field focuses on the estimation of geometry of road lane boundaries (Bosch et al. 2008). Alternative algorithms carried out the texture analysis of local pixel neighborhood (Rasiwasia and Vasconcelos 2008) to facilitate the derivation of two road scene clusters ('on-road' and 'off-road'), and complete the classification through neural network. Ess et al. (Ess et al. 2009) also investigated the field of feature-based road environment understanding, which focuses on the representation of road markings, on-road and road-side objects, and road type recognition according to the occurrence of junctions, roundabouts and turnings. Tang and Breckon (2011) further utilized the feature set of color and texture distribution extracted from multiple regions of interest, and combines them with trained classifiers (k-nearest neighbor and artificial neural network) to solve the four road type classification problem ('off-road', 'urban road', 'trunk road' and 'motorway'). Mioulet et al. (2013) improved the real-time texture classification by using a hardware-specific Gabor filter, which can successfully extract the Gabor feature of above four types of road environment for classification based on the random forest (Paul et al. 2018). Due to the selection of robust and representative features, the method in Mioulet et al. (2013) achieved better performance than the method in Tang and Breckon (2011). Mohammad et al. (2015) established a two-stage classification strategy to improve the road type classification accuracy. The first stage is to construct a statistical model for each road type offline, and the second stage is the online classification of new video frames. It can be seen that most methods still implement real-time road type classification based on the underlying visual features and manually designed classifier, such as random forest (RF) (Paul et al. 2018) and support vector machine (SVM) (Pan et al. 2018), rather than end-to-end solutions.

Recently, deep learning models (Gong et al. 2018; Yang et al. 2018; Wu and Prasad 2018; Zheng et al. 2018, 2020a, b) trained on big data can mine the implicit knowledge of samples and thus can deal with image classification problems well. However, the huge structures of such over-parameterized models make them hard to apply to embedded devices or mobile terminals. Therefore, reducing the network size of deep learning methods and accelerating their reasoning speed is the current research hotspot.

2.2 Network pruning for accelerating inference

Mathematical investigations (Tung and Mori 2020; Ko et al. 2018; Smith et al. 2018) have illustrated that weight removal of deep neural networks is highly efficient due to the redundancy across channels and kernels. At present, there are a large number of related publications (Wang et al. 2018; Yang et al. 2018; Yu et al. 2018; Li et al. 2019) on the removal of redundant parameters to reduce memory footprint and help model to accelerate reasoning. Existing pruning techniques either train baseline network from scratch with sparse regularization on parameters (or connections), or minimize the reconstruction error between feature maps before and after pruning. Both techniques suffer from some limitations: the former needs a large amount of computation and is difficult to converge, while the latter ignores the discriminative ability of filters.

In practice, it is usually difficult to determine which metric can be used to determine the importance of network parameters or connections. To overcome this challenge, Wang et al. (2018) proposed using the generative adversarial network (GAN) as a teacher to guide the construction of lightweight student neural network. Yang et al. (2018) proposed improving the information flow in deep neural networks to make more efficient usage of parameters, which help to alleviate the optimization difficulty of pruned network. Experiments on four public image classification benchmarks have shown that their method achieves the state-of-the-art performance with fewer parameters. Yu et al. (2018) designed ε-ResNet to allow the structure to automatically discard redundant layers with negligible loss of accuracy. Li et al. (2019) introduced a dynamic selection mechanism in CNN and designed a building block called Selective Kernel unit, which allowed neurons to adjust their receptive field adaptively based on multiple scales of input information. Experiments showed that SKNet is superior to the existing state-of-the-art architectures with lower model complexity. Lin (2019) jointly pruned filters and other structures in an end-to-end manner by introducing a new objective function that contained sparse regularization terms. Fast iterative shrinkage-thresholding algorithm is used to remove the corresponding structures quickly and reliably. Discrimination-aware channel pruning (DCP) method (Zhuang 2018) introduced additional recognition perception loss into the network to improve the recognition ability of the middle layer, and then chose the most recognizable channel for each layer by considering additional loss and reconstruction error. Experiments showed that the pruned ResNet-50 with 30% reduction of channels improved the accuracy by 0.39% on ILSVRC-2012. Liu et al. (2018) proposed a frequency-domain dynamic pruning scheme to explore spatial correlations and remove unimportant parameters. Specifically, the frequency-domain coefficients are removed dynamically in each iteration and various frequency bands are pruned differently due to their different influence on accuracy. It achieved a compression ratio of 8.4 × and a theoretical inference speedup of 9.2 × for ResNet-110, while the accuracy is even better than baseline model. Auto-balanced filter pruning method (Ding et al. 2018) transferred the representation ability of some convolutional layers to a part of filters to remove redundant parameters, and then retrains the remaining kernels to restore the accuracy. In this way, a smaller version of the original network can be learned and the floating-point operations (FLOPs) are reduced. Soft filter pruning method (He et al. 2018) developed a large architecture learning capacity and had less dependence on the pre-training model. Zhong et al. (2018) used long short term memory (LSTM) to learn the hierarchical characteristics of the network and generated a pruning decision for each layer, which is the main difference from previous work. Park et al. (2018) designed a new value-aware quantization method which applies aggressively reduced precision to the majority of samples while separately handling a small amount of large data in high precision, and results show that deep networks such as Inception-v3, ResNet-101

and DenseNet-121 can be quantized for inference with 4-bit weights and activations (with 1% 16-bit data) within 1% top-1 accuracy drop. The above methods still lack the guarantee in weight reduction rate and convergence time. Zhang (2018) presented a systematic weight pruning framework using the alternating direction method of multipliers, and achieved weight reduction of 21 × without accuracy loss ImageNet dataset.

In summary, the existing network pruning methods need to go through a long iteratively pruning and retraining process, which is instructive but lacks assurance of weight reduction rate and convergence time. Most of them rely on hierarchical multi-stage optimization for iteratively pruning and retraining, which may not be optimal and require a large amount of computation. In addition, these methods are usually designed to prune specific structures, such as filters or blocks in ResNet, without considering structural adaptability and jointly pruning heterogeneous structures.

3 CLMIP method

Pruning method requires minimizing the network size while keeping the performance of the model as much as possible. Slight pruning cannot meet the requirements, and excessive pruning may lead to the collapse of network structure and the inability to converge. In this section, we introduce proposed CLMIP method for CNN compression and acceleration in an iterative manner, which controls pruning rate in each iteration to prevent the above problems. The whole process consists of the following steps:

(1) Pre-train the original deep CNN model to converge on the training set;
(2) Remove redundant parameters in the network according to the reconstruction errors of data manifolds;
(3) Fine-tune the structure of pruned network as much as possible;
(4) Judge whether the pruned network meets the target trade-off between classification performance and pruning objective;
(5) Save sparse deep CNN and deploy it in embedded systems for road type recognition.

The alternate pruning and fine-tuning process of deep CNN using the CLMIP method is shown in Fig. 2.

3.1 Problem formulation

Given a L layer deep CNN model $\mathbb{M}_0 : f(\mathbf{x}; \boldsymbol{\theta}_0)$ optimized on training set $S : \left\{(\mathbf{x}_i,\ \mathbf{y}_i)\right\}_{i=1}^{N}$ where $(\mathbf{x}, \mathbf{y})$ and $\boldsymbol{\theta}_0 : \left\{\mathbf{W}_0^l, \mathbf{b}_0^l\right\}_{l=1}^{L}$ represent the inputs (i.e. the perceptual images of road environments and corresponding ground truth labels) and initialized network parameters, respectively. Parameters are organized in four dimensional tensors and two dimensional matrices in the convolutional and fully connected layers, respectively.

In the forward propagation process, sample images $\mathbf{x}$ are fed into the first layer and the corresponding output $\mathbf{h}_1$ can be calculated by

$$\mathbf{h}_1 = \sigma(\mathbf{W}_0^1\mathbf{x} + \mathbf{b}_0^1) \tag{1}$$

where $\sigma(\cdot)$ represents the element wise non-linear activation function, such as exponential linear unit (Yang et al. 2018), which is defined as

$$\sigma(x) = \begin{cases} x & if\ x > 0 \\ \tau e^x - \tau & if\ x \le 0 \end{cases} \tag{2}$$

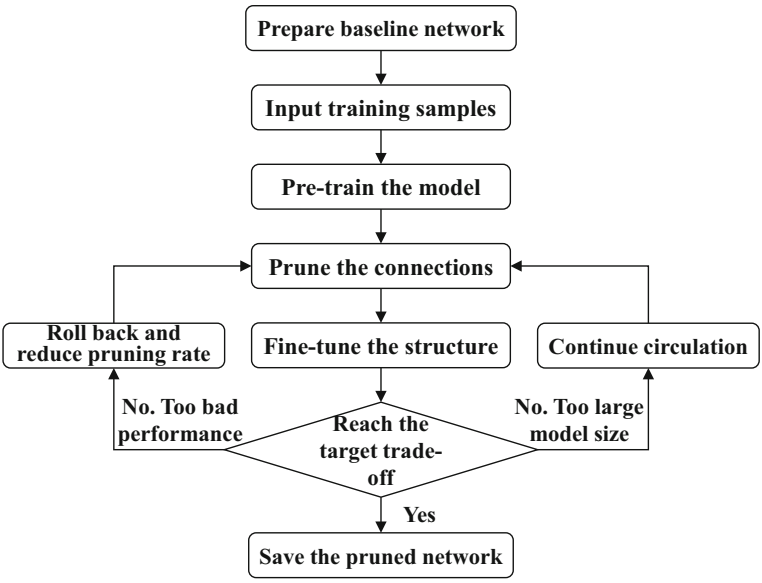


Fig. 2 The alternate pruning and fine-tuning procedure of deep CNN using the CLMIP method

where τ is an adjustable parameter that controls the saturation of activation function. Since the output of one layer is the input of the next, the output of l-th layer for $l = 2, \ldots, L-1$ is given by

$$\mathbf{h}_l = \sigma(\mathbf{W}_0^l \mathbf{h}_{l-1} + \mathbf{b}_0^l). \tag{3}$$

Then the final output (prediction result of road type) of deep CNN is

$$f(\mathbf{x}) = \text{softmax}(\mathbf{W}_0^L \mathbf{h}_{L-1} + \mathbf{b}_0^L). \tag{4}$$

At last, the total loss of deep CNN model can be calculated as

$$\mathcal{L}(\mathbf{x}_i, \mathbf{y}_i) = -\frac{1}{C} \sum_{j=1}^{C} \left[y_i^j \log f(\mathbf{x}_i)^j + (1 - y_i^j) \log(1 - f(\mathbf{x}_i)^j) \right] + \lambda \sum_{k=1}^{L} ||\mathbf{W}^k||_F \tag{5}$$

where $C = 4$ denotes the categories of road type, including 'urban road', 'off-road', 'trunk road' and 'motorway'. The first term is the loglikelihood loss and the second term is L2 regularization of all the weights. λ is a hyperparameter controlling regularization intensity and $\|\cdot\|_F$ means the Frobenius norm.

The optimization of the deep CNN is to minimize the loss by updating parameters $\boldsymbol{\theta}$ (i.e., weights $\mathbf{W}$ and biases $\mathbf{b}$). In the back propagation process, the network parameters in l-th layer at the t-th training iteration can be updated through mini-batch stochastic gradient descent (SGD) method (Zheng et al. 2019):

$$\boldsymbol{\theta}_t^l = \boldsymbol{\theta}_{t-1}^l - \alpha \cdot \frac{1}{M} \sum_{i=1}^{M} \frac{\partial \mathcal{L}(\mathbf{x}_i, \mathbf{y}_i)}{\partial \boldsymbol{\theta}_{t-1}^l} \tag{6}$$

where α and M represent the learning rate and batch size, respectively. Through successive iterations, a convergent deep CNN $\mathbb{M}_* : f(\mathbf{x}; \boldsymbol{\theta}_*)$ can be finally obtained.

The pruning objective is to remove the redundant weights of deep CNN, and therefore we try to minimize the loss subject to constraints on the cardinality of nonzero weights in the network. In other words, the mini-batch SGD training process is updated to solve the following problem.

$$\underset{\boldsymbol{\theta}}{\text{minimize}}\ \mathcal{L}(\mathbf{x}, \mathbf{y}),\ \text{subject to}\ \frac{prune(\mathbf{W})}{total(\mathbf{W})} \geq \xi \tag{7}$$

where *prune* (**W**) and *total* (**W**) return the number of pruned and total weights in deep CNN, respectively. ξ is the manually set total pruning rate that reflects our requirements for model size.

3.2 Parameter importance ranking criterion

Since we hope to ensure the generalization ability of CNN by maintaining the invariance of the data manifolds before and after pruning, the accurate representations of the cross-dimensional manifolds are crucial. Differential forms play a key role in manifold theory. They are, the first and foremost, intrinsic objects associated to any manifolds, and therefore can be used to construct diffeomorphism invariants of the manifold (Kwong 2016; Bône et al. 2018). Relative to vector fields, which are also intrinsic to the manifold, differential forms have a more abundant algebraic structures.

For the l-th convolutional layer, the cross-dimensional manifold descriptor of samples in a mini-batch set can be defined as

$$\boldsymbol{\varphi}_{conv}^{\ l} = (||\mathbf{h}_l^1 - \mathbf{h}_l^2||_F^2,\ ||\mathbf{h}_l^1 - \mathbf{h}_l^3||_F^2,\ \ldots,\ ||\mathbf{h}_l^i - \mathbf{h}_l^j||_F^2)_{1\leq i<j\leq M}^{M(M-1)/2}. \tag{8}$$

It characterizes the relationship between feature maps of the same dimension generated from mini-batch samples. In the process of dimensionality reduction layer by layer (i.e. semantics becoming abstract), the redundant information is gradually eliminated and critical implicit knowledge is retained and reorganized. Real projective space can be interpreted as a quotient of a sphere with antipodal points identified, or as the set of lines through the origin in a vector space (Tu 2011). The Frobenius norm of the difference of feature maps at the same layer can capture this distance and can be used to construct data manifolds in vector space.

The output dimension of fully connected layer is different from that of convolutional layer, which changes from the three-dimensional tensor to the two-dimensional vector. Therefore, $\boldsymbol{\varphi}$ for fully connected layer should be updated to

$$\boldsymbol{\varphi}_{full}^{\ l} = (|h^1{}_l - h_l^2|,\ |h^1{}_l - h_l^3|,\ \ldots,\ |h^i{}_l - h^j{}_l|)_{1\leq i<j\leq M}^{M(M-1)/2}. \tag{9}$$

Then the cosine similarity is used to measure the difference between manifolds spanning various dimensions, as given by

$$\phi = \frac{1}{2}\sum_{p=1}^{L}\sum_{\substack{q=1,\\ q\neq p}}^{L}\cos\langle\boldsymbol{\varphi}^p, \boldsymbol{\varphi}^q\rangle \tag{10}$$

where

$$\cos\left\langle \boldsymbol{\varphi}^p, \boldsymbol{\varphi}^q \right\rangle = \frac{\sum_{i=1}^{M(M-1)/2} \varphi_i^p \times \varphi_i^q}{\sqrt{\sum_{i=1}^{M(M-1)/2} (\varphi_i^p)^2} \times \sqrt{\sum_{i=1}^{M(M-1)/2} (\varphi_i^q)^2}}. \tag{11}$$

Cosine similarity is insensitive to the order of sequences and the absolute magnitude of a specific value, which can fairly describe the changes in the data manifold. Furthermore, it is worth noting that the cosine similarity is equivalent to the *Pearson correlation* for mean-centered normalized vectors. The frequently used Euclidean distance between two samples in high-dimensional feature space can no longer reflect the actual relationship between them, because the influence of noise often makes them sparse.

After the manifold descriptors of all types of layers in the network are obtained, we can calculate the average reconstruction error of manifolds on all samples before and after network pruning according to

$$\psi = \frac{M}{N} \sum_{i=1}^{N/M} |\phi_i - \tilde{\phi}_i| \tag{12}$$

where $\tilde{\phi}$ represents the manifold similarity of pruned network. Finally, the importance of network weights can be estimated according to the min–max normalized reconstruction errors of manifolds, which can be formulated by

$$\psi = \frac{\psi - \psi_{\min}}{\psi_{\max} - \psi_{\min}}. \tag{13}$$

Based on this ranking criterion, the weights in all convolutional and fully connected layers of deep CNN are ordered and smaller ones can be removed. The ranking process of weight importance is summarized in Algorithm 1. Since the output size of feature maps is not limited in the estimation of data manifolds, the pruning criterion is suitable for neural networks with any special structures, such as the Inception block (Szegedy 2015) containing convolutional kernels of different sizes and the residual modules (He et al. 2016).

The key issue of pruning is to evaluate the importance of parameters. Once the importance ranking criterion is established, the model can then be pruned and fine-tuned based on this to compress the model size and speedup the inference.

Algorithm 1 Ranking process of weight importance.

Input: training samples and corresponding labels $S: \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$.

Initialization: pre-trained CNN, batch size M, pruning rate ξ, pruning step size ζ, and threshold γ.

1: **for** l from 1 to L **do**

2: calculate the manifold descriptor $\boldsymbol{\varphi}^l$ of l-th layer;

3: **end for**

4: calculate the cosine similarity ϕ between manifolds;

5: calculate the normalized reconstruction errors $\boldsymbol{\psi}$ of

Output: the importance value ψ of each network weight.

3.3 Network pruning and fine-tuning

In this section, we introduce the process of deep CNN being pruned. To prevent the collapse of the model structure, we adopt iterative pruning and fine-tuning to gradually reduce the model size while ensuring the classification performance. In some cases, even if the classification accuracy of the model cannot reach the expected performance after a certain fine-tuning, the model can be rolled back to the previous version. Besides, fine-tuning is essential to compensate the accuracy loss from the previous pruning to suppress the accumulative error.

3.3.1 Pruning process

During the pruning process, a deep CNN model needs to be pruned at least $100\xi/\zeta$ times if ζ % of parameters is removed at each pruning iteration, in which ξ and ζ represent pruning rate and pruning step size, respectively. Suppose that the removed parameter set is $\tilde{\boldsymbol{\theta}} : \{\mathbf{W}_1, \ \mathbf{W}_2, \ldots \mathbf{W}_{100\zeta}\}$, then we can define a suppression mask $\mathbf{M}$ with the same dimension as weight matrix to specify the reserved or removed parameters. Before the next fine-tuning of the pruned network, the weight matrix is first updated by dot multiplying with the suppression mask:

$$\hat{\boldsymbol{\theta}} = \boldsymbol{\theta} \circ \mathrm{M} \tag{14}$$

where $\boldsymbol{\theta}$ is the parameter set of the neural network at the beginning of the current pruning round and $\circ$ represents the element-wise operation for computing the Hadamard product. By doing this, the redundant weights are removed by multiplying with 0, while the remaining weights can be preserved by multiplying with 1. In other words, the parameters of pruned network has been updated to $\hat{\boldsymbol{\theta}} : \{\mathbf{W} \in \boldsymbol{\theta} \& \mathbf{W} \notin \tilde{\boldsymbol{\theta}}\}$.

In the local preparation phase of the network, the computational complexity of pruning process is $\boldsymbol{O}\ (L\zeta + \xi \log\zeta)$ where $L\zeta$ and $\xi\log\zeta$ reflect the computing resources occupied by the layers of neural network and all pruning steps during the iterative process, respectively. Compared with the retraining process, the additional calculation can be ignored. When the pruned deep CNN is deployed on the embedded system for road type recognition, the memory footprint of suppression mask (about 15% under 50% pruning rate) is much smaller than that of the real-valued weights removed, which greatly reduces the storage and computation requirements.

3.3.2 Fine-tuning process

Starting from a pruned deep CNN, CLMIP updates the model $\mathbb{M}$ at the fine-tuning process. After each pruning is completed, an updated network $\mathbb{M}'$ whose parameters are transferred from the previous model is fine-tuned by using mini-batch SGD, as shown in Eq. (6). The learning policy of fine-tuning is consistent with the initial settings. The discriminative power of layers can be significantly improved after fine-tuning and another round of pruning can be performed. We only need a single fine-tuning step to retain the performance of the model and the convergence can be much faster than directly training the baseline deep CNN from scratch, since the starting position of the fine-tuning process is already close to the position which can obtain the original performance. On the other hand, many studies (Lemley et al. 2017; Chen et al. 2015; Li et al. 2016) have shown that fine-tuning the pruned network performs better than training sparse model from scratch. Although the learning capacity of sparse network is sufficient enough to fit plenty of training samples, it is more likely to fall

into a local minimum, while the denser network can help to find better initial positions in the parameter space. Once a good initialization is provided by previous version of the model, the pruned network is able to obtain its original classification ability by fine-tuning.

The end point of the entire process depends on whether the trade-off we set is satisfied. When the proportion of parameters removed from the model is larger than the pruning rate and the loss of validation set is less than the threshold, it ends otherwise entering the next loop, that is,

$$\begin{cases} \text{continue the iteration} & if \ \frac{prune(\mathbf{W})}{total(\mathbf{W})} < \xi \text{ or } validation\ loss\ > \gamma \\ \text{end loop} & if \ \frac{prune(\mathbf{W})}{total(\mathbf{W})} \geq \xi \text{ and } validation\ loss\ \leq \gamma \end{cases} \tag{15}$$

where γ is a hyperparameter to determine the availability of deep CNN. In other words, γ is a threshold used to determine whether the validation loss reaches our expectation. There are different requirements for model performance in different scenarios. For example, urban roads with high traffic flow require more sensitive algorithms than off-roads with less traffic. Finally, the network sequence $\{\mathbb{M}_1, \mathbb{M}_2, \ldots, \mathbb{M}_T\}$ with sparse connections specified by suppression masks $\{\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_T\}$ are saved. Among these candidates, the best configuration which minimizes the structural complexity while meeting accuracy requirements is selected and deployed on the mobile side.

The process of alternate pruning and fine-tuning is summarized in Algorithm 2. So far, we have introduced all the details of proposed CLMIP method.

Algorithm 2 Alternate pruning and fine-tuning of deep CNN.

Input: training samples and corresponding labels $S: \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$.

Initialization: baseline CNN, batch size M, learning rate α, Dropout rate, weight decay λ, pruning rate ξ, pruning step size ζ, threshold γ, suppression mask $\mathbf{M}$.

```
for t from 1 to T do
    transfer weights of M_t from M_{t-1}: θ_t ← θ_{t-1};
    update weights: θ̂ = θ ∘ M;
    if prune (W) / total (W) < ξ or validation loss > γ do
        compute the loss of pruned network;
        update weights by using mini-batch SGD;
    end if
end for
ξ_t ← ξ_{t-1} - ζ;
```

Output: the pruned deep CNN.

Table 1 Hyperparameters setting in ImageNet pre-training process

Hyperparameters	Learning rate	Batch size	Weight decay	Category	Number of samples	Convergent training loss
Value	0.01	64	0.0005	20	10^5	0.1

4 Experimental results and analyses

4.1 Experimental setup

4.1.1 Baseline deep CNN architecture

In this part, we introduce the specially designed baseline CNN for road type classification, as shown in Fig. 3. Considering the number of samples and image size of the target dataset. The model is consisted of 12 convolutional layers, 4 max pooling layers, 1 global pooling layer, 2 fully connected layers and a softmax layer to output classification results. Batch normalization layer (Li et al. 2016) and the exponential linear activation function (Yang et al. 2018) are added behind each convolutional and the fully connected layer, and the continuous Dropout (Shen et al. 2018) is used in fully connected layers to prevent the over-fitting problem and improve the generalization ability of the model. As for weight initialization, the model is initialized by MSRA (He et al. 2015) and then pre-trained on ImageNet (Russakovsky et al. 2015). The pre-training details are summarized in Table 1. Taking into account the depth of network and the number of convolution kernels, we set the pre-training task to be 20 categories, of which 5000 images per category, which is similar to the total number of training samples in the target dataset. Then the training loss is optimized as much as possible without considering the performance of validation set. This can ensure the optimization capacity of the deep learning model and transfer it from the source domain to the target domain.

Finally, the pre-trained network is retrained on the target dataset and is used for subsequent pruning and fine-tuning. The total number of parameters in the baseline CNN is about 0.19 million, as calculated by

$$\begin{aligned} &3 \times 3 \times 128 \times 2 + 3 \times 3 \times 256 \times 4 + 3 \times 3 \times 512 \\ &\quad \times 5 + 7 \times 7 \times 512 + 512 \times 256 + 256 \times 4 = 191744. \end{aligned} \tag{16}$$

The memory consumed by image transmission during network prediction is

$$\begin{aligned} &(320 \times 240 \times 3 + 320 \times 240 \times 128 \times 2 + 160 \times 120 \times 128 + 160 \times 120 \times 256 \times 2 \\ &\quad + 80 \times 60 \times 256 \times 3 + 40 \times 30 \times 256 + 40 \times 30 \times 512 \times 3 + 20 \times 15 \\ &\quad \times 512 \times 3 + 3 \times 2 \times 512 + 512 + 256 + 4 + 4)/10^6 \times 4 \approx 153.9 \text{ MB}. \end{aligned} \tag{17}$$

4.1.2 Benchmark dataset

During the sample preparation phase, we use the python based keyword crawler program to obtain a total of 102 first-view videos (about 10 h) of car cameras on various websites, including 137 200 images of four road scenes (‘urban road’, ‘off-road’, ‘trunk road’ and ‘motorway’). Some sample images are shown in Fig. 4. Then 62 videos (81 496 images) are divided into the training set, 20 videos (26 292 images) for the validation set, and 20

Baseline Network Architecture	
Layer type	**Output dimension**
input	320×240×3
convolution 1, 3×3, stride 1, padding 1	320×240×128
convolution 2, 3×3, stride 1, padding 1	320×240×128
max pooling 1, 2×2	160×120×128
convolution 3, 3×3, stride 1, padding 1	160×120×256
convolution 4, 3×3, stride 1, padding 1	160×120×256
max pooling 2, 2×2	80×60×256
convolution 5, 3×3, stride 1, padding 1	80×60×256
convolution 6, 3×3, stride 1, padding 1	80×60×256
max pooling 3, 2×2	40×30×256
convolution 7, 3×3, stride 1, padding 1	40×30×512
convolution 8, 3×3, stride 1, padding 1	40×30×512
convolution 9, 3×3, stride 1, padding 1	40×30×512
max pooling 4, 2×2	20×15×512
convolution 10, 3×3, stride 1, padding 1	20×15×512
convolution 11, 3×3, stride 1, padding 1	20×15×512
convolution 12, 7×7, stride 5, padding 1	3×2×512
global average pooling	1×1×512
full connection 1	1×1×256
full connection 2	1×1×4
softmax	1×1×4

Fig. 3 Baseline network architecture for 4 class road type classification

videos (29 412 images) for the test set. During the training phase, all images are first resized to 320×240 and then training samples are augmented by horizontal flipping and random cropping. The video based sample partitioning approach is more challenging, which can avoid information leakage and evaluate the model more fairly.

4.1.3 Pruning and fine-tuning details

Many hyperparameters have been introduced during the pruning and fine-tuning of deep CNN. The choice of hyperparameters is determined according to the classification performance of the model on validation set. Finally, the hyperparameters used for model pruning, including pruning rate ξ, pruning step size ζ and threshold γ, are set to 0.4, 2 and 0.25, respectively; and the hyperparameters used in the model fine-tuning process, including learning rate α, batch size M and weight decay λ, are set to 0.01, 64 and 0.0005, respectively. The learning rate is reduced by a factor of 10 when the training loss is no longer reduced, and the training loss can be calculated by Eq. (5).

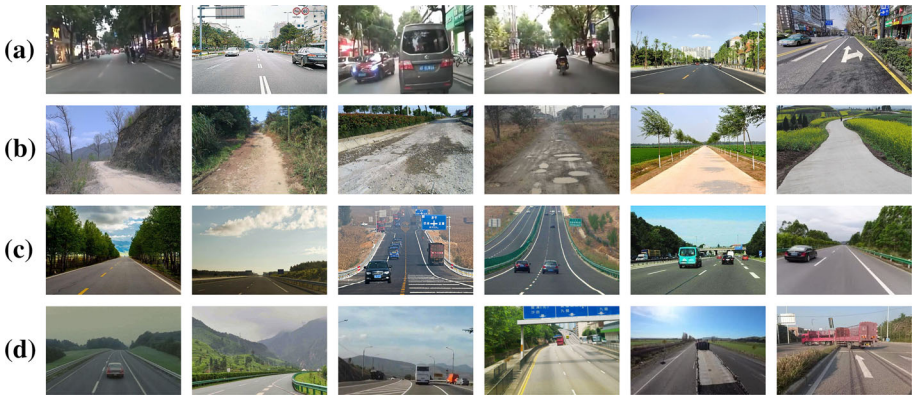


Fig. 4 Example images of four types of road environment. **a** Urban road, **b** Off-road, **c** Trunk road, **d** Motorway

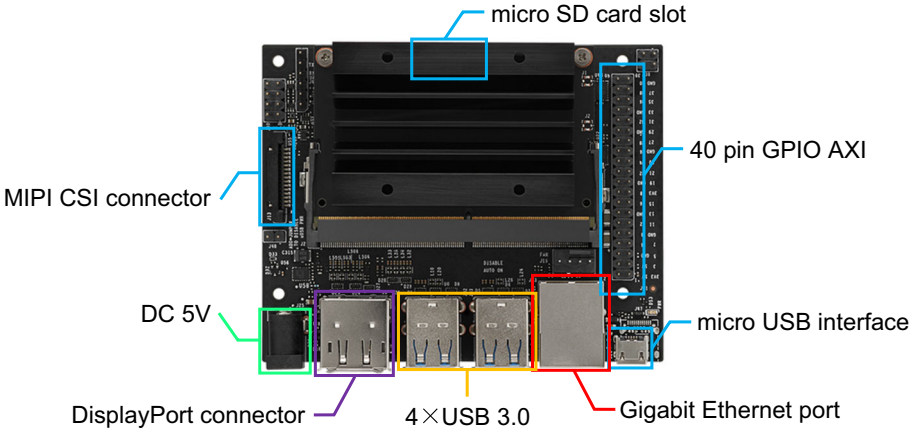


Fig. 5 The specific structure of developer kit NVIDIA Jetson Nano

4.1.4 Embedded device

The neural network model is implemented by PyTorch (Ketkar 2017) deep learning platform based on Python. In the test phase, the pruned network is deployed in the embedded device based on the developer kit NVIDIA Jetson Nano to test whether it can meet the requirements of real-time and accuracy.

The developer kit NVIDIA Jetson Nano is 70×45 mm in size and consists of a ARM Cortex A57 CPU with 4 cores, a Maxwell GPU with 128 cores, 16 GB storage, and 4 GB 64-bit memory. The structure of the development board is shown in Fig. 5. It can realize the computing power of 0.5 TFLOPs at 5 W power.

4.2 Performance of CLMIP

With the reduction of the parameter size of deep CNN, the optimization ability of the model will decrease gradually and lead to the underfitting problem, which is not what we expected. Multiple optimizations of pruned model in the fine-tuning process are shown in Fig. 6. It can

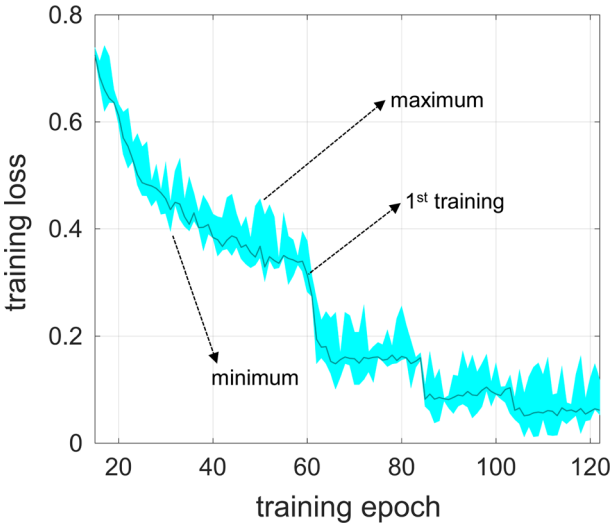


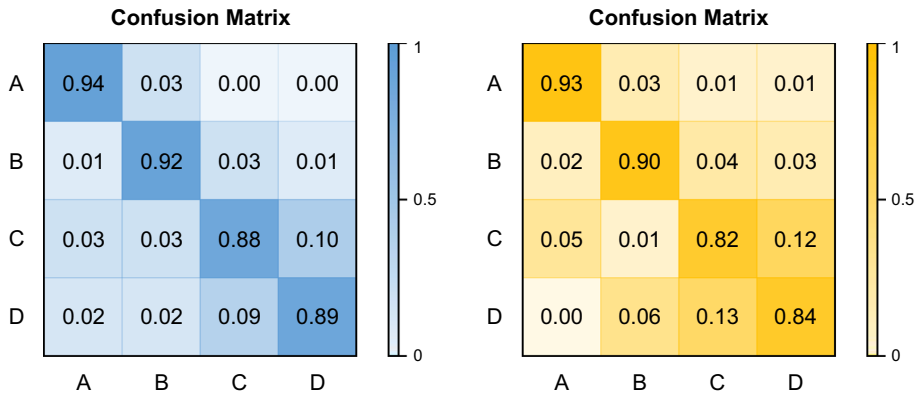
Fig. 6 Multiple optimizations of pruned model in the alternate pruning and fine-tuning process

be seen that the training loss fluctuates but never exceeds 0.2. Finally, after the original CNN model removed the redundant parameters by pruning, the number of the parameters is reduced to 60% (115 046), the memory consumption is reduced to 104.2 MB, and the computational requirements (i.e., FLOPs) are reduced by 46.72%. Tests on the platform Jetson Nano show that the pruned model only needs 31 ms to judge the road type of a single image, which means that the processing speed of the system can reach 26 frames per second and meet the real-time requirements. In a real-time system, the maximum response time of the system must be less than the required threshold. We have observed the maximum response time of the pruned model through testing all samples in the dataset, and the maximum response time of 0.039 s can meet the actual demand.

The performance of the original and pruned deep CNN on the test set are shown in Fig. 7. The confusion matrix shows the classification accuracy of CNNs for the four road types. The pruned deep CNN achieves the classification accuracy of 93%, 90%, 82% and 84% for 'urban road', 'off-road', 'trunk road' and 'motorway', respectively. It is worth noting that the model is more likely to confuse 'trunk road' and 'motorway'. Compared with the original model, the average classification accuracy i decreased approximately 3%, which can be acceptable considering the reduction of model size. In fact, we can further reduce the model size of deep CNN by limiting its classification performance.

4.3 Comparison with other pruning methods

In this part, we compare various pruning methods to illustrate the effectiveness of proposed CLMIP, including DCP (Zhuang 2018), Auto-balance (Liu et al. 2018), FDNP (Ding et al. 2018), etc. All methods are set up according to the description in original paper and implementation details are presented in Table 2. To demonstrate the acceleration of model reasoning, we report the classification accuracy and reduction of multiplication and parameters, which are denoted as [Classification accuracy (%)], [FLOPs (↓%)] and [Parameters (↓%)] in Table 3, respectively. FLOPs is one of the most commonly used metrics to compare the computa-



(a) original network **(b)** pruned network

Fig. 7 Visualization of confusion matrix of the recognition results. 'A', 'B', 'C' and 'D' represent 'urban road', 'off-road', 'trunk road' and 'motorway', respectively

Table 2 Implementation details of pruning methods for comparison

Methods	Implementation details
DCP (Zhuang 2018)	See https://github.com/SCUT-AILab/DCP
FDNP (Liu et al. 2018)	Compression rate: 0.5, relevant hyperparameters: 1.0 and 0.8
Auto-balance (Ding et al. 2018)	Compression rate: 0.5, accuracy drop threshold: 0.05
SFP (He et al. 2018)	See https://github.com/he-y/soft-filter-pruning
Value-aware (Park et al. 2018)	Compression rate: 0.5, accuracy drop threshold: 0.05
ADMM (Zhang 2018)	See https://github.com/KaiqiZhang/admm-pruning

Table 3 The classification accuracy and reduction of multiplication and parameters of deep CNN under various pruning methods

Pruning methods	Classification accuracy (%)	FLOPs (↓%)	Parameters (↓%)
No pruning	90.64	0.00	0.00
DCP (Zhuang 2018)	81.79	44.51	51.08
FDNP (Liu et al. 2018)	84.02	35.22	37.90
Auto-balance (Ding et al. 2018)	84.13	32.30	36.68
SFP (He et al. 2018)	82.66	48.40	41.80
Value-aware (Park et al. 2018)	79.40	38.34	35.51
ADMM (Zhang 2018)	77.52	50.66	43.47
CLMIP	87.38	46.72	40.00

tional complexity of various CNN architectures, which can be independent of the underlying hardware and software. The trade-off between classification accuracy and the reduction of FLOPs and parameters have empirically demonstrated the effectiveness of CLMIP method.

According to the experimental results, although the maximum reduction of FLOPs is achieved by ADMM (Zhang 2018), it leads to the sharp decline (from 90.64 to 77.52%) in the classification performance of pruned model. We speculate that a part of the core structure

Table 4 The memory footprint and the FPS of various lightweight networks and the corresponding classification performance

Lightweight CNNs	Classification accuracy (%)	Memory footprint (MB)	Frames/s
CliqueNet (Yang et al. 2018)	88.46	377.0	5
SKNet (Li et al. 2019)	83.17	91.0	22
SENet + ResNet (Hu et al. 2018)	84.89	170.3	14
SqueezeNet (Iandola 2017)	74.42	4.8	47
Xception (Chollet 2107)	87.20	260.6	7
ShuffleNet (Zhang et al. 2018)	84.21	143.0	13
Our CNN + CLMIP	87.38	104.2	26

of deep CNN may be destroyed, and the feature extraction and organization ability of the model from bottom layer to the upper level are affected. DCP method (Zhuang 2018) removes almost half of the parameters of the original model, while the classification performance decreases by only 8.85%, which shows stronger robustness to the change of model structure. Our proposed CLMIP method achieves the highest average classification accuracy (87.38%) on four road types, with 40% and 46.72% reduction in parameters and FLOPs, respectively. Although the classification accuracy of pruned deep CNN decreases slightly (3.26%), the resulting computation and memory savings are significant. CLMIP method can remove the redundant parameters in the CNN architecture as much as possible while ensuring model classification performance.

4.4 Comparison with other road type classification methods

To prove the necessity and value of pruning, we compare the performance of deep CNN with alternate pruning and fine tuning model with lightweight CNNs trained from scratch. The weight initialization and hyperparameter setting of these networks are set according to the description in the original papers. The memory footprint of each network and the corresponding classification performance are shown in Table 4. By pruning and fine-tuning the deep CNN we designed, it performs far better than the lightweight networks with smaller sizes, including SKNet (Li et al. 2019) and SqueezeNet (Iandola 2017). Even the performance of some larger lightweight networks such as SENet (Hu et al. 2018), Xception (Chollet 2107) and ShuffleNet (Zhang et al. 2018), has also been surpassed. Compared with our pruned model, the classification accuracy of above three lightweight networks is reduced by 4.21%, 0.18% and 3.17%, respectively. Only CliqueNet (Yang et al. 2018) performs better than network with CLMIP, which is not surprising considering that its model size is 3.6 times bigger than our pruned model.

We also report the comparison between the pruning network model and the traditional classification methods, including Gabor features based method (Mioulet et al. 2013) and Gaussian mixture models (GMMs) (Mohammad et al. 2015). The classification accuracy of the four road types is presented in Fig. 8. Compared with the conventional feature design strategy, our pruned deep CNN shows remarkable superiority and achieves the highest classification accuracy in all road types, i.e., 92.2% for ‘urban road’, 90.4% for ‘off-road’, 82.7% for ‘trunk road’, and 84.2% for ‘motorway’. Deep learning can automatically mine implicit

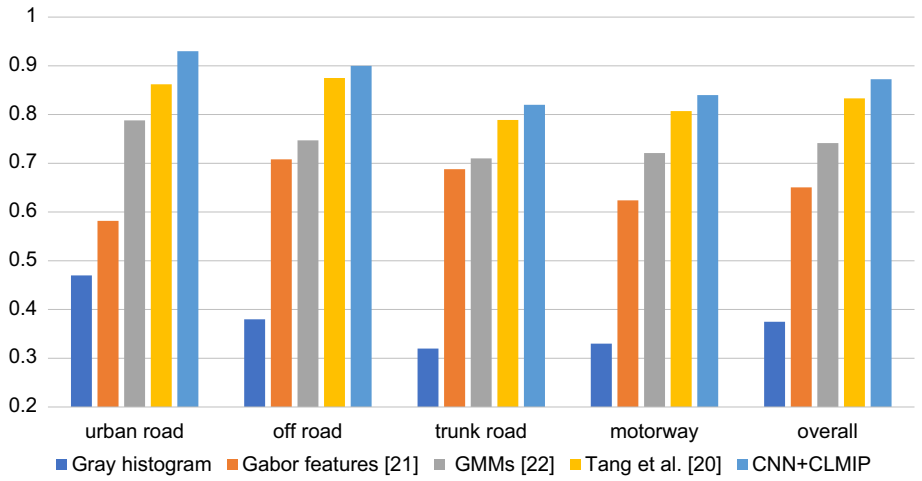


Fig. 8 Performance comparison of pruned CNN with other methods

knowledge from a large number of samples, which is helpful for the understanding of complex road environments.

4.5 Distribution of removed parameters

In cases where the classification accuracy loss can be ignored, the removed parameters reflect the redundancy of the CNN structure. Therefore, we plan to determine the location of structure redundancy and the reasons behind it by observing the distribution of removed parameters. The number of parameters removed from each layer of the network is counted, as shown in Fig. 9. It can be seen that more than half of the parameters come from the two fully connected layers. In fact, 68.9% (132 096/191 744) of the parameters in the original network come from the fully connected layers, which contain most of the structural redundancy. Too many parameters in the fully connected layers destroy the spatial structure and high-level semantic information of the image, resulting in the model cannot be generalized on the unseen test samples. On the other hand, the parameters of the bottom layer are more important in the convolutional layers, which can be seen from the fact that the first four convolutional layers have hardly been pruned. The underlying convolutional layer is dedicated to extracting basic features such as edges, colors, etc., which are the basis and key to the formation of subsequent high-level semantics.

In fact, we have tried to replace the final fully connected layer with a convolutional layer containing 1×1 filters, hoping to give the baseline network a better starting location in the structural space. In the experimental process, fully connected layer is replaced by convolutional layer with different number of filters, including 512, 1024 and 2048. Then the classification accuracy of the model in two cases (with or without pruning) is observed in Table 5. However, whether it is the original or pruned network, the performance cannot comparable to the original one. This reveals some of the properties of fully connected layer that have not yet been understood. Although many advanced networks [Yolo v2 (Redmon and Farhadi 2017) and SSD (Liu 2016)] gradually replace the fully connected layer as the global average pooling layer, which maps latent variables to category labels in the form of

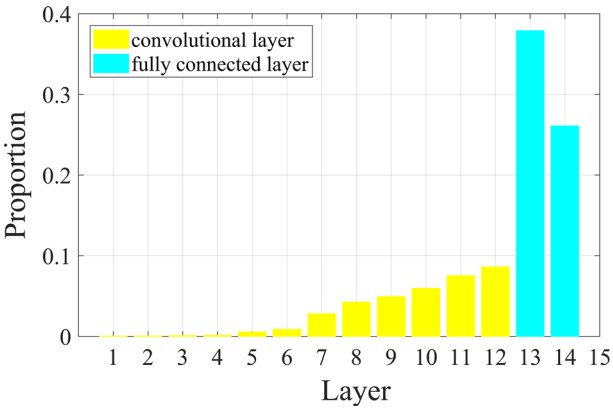


Fig. 9 Distribution of parameters removed from each layer of pruned CNN

Table 5 The classification accuracy of updated deep CNN with or without pruning

CNN structures	Classification accuracy (%)	Memory footprint (MB)
CNN (2048 1×1 filters)	85.50	190.7
CNN (1024 1×1 filters)	86.19	174.4
CNN (512 1×1 filters)	85.66	167.6
Pruned CNN (2048 1×1 filters)	81.20	91.5
Pruned CNN (1024 1×1 filters)	79.90	93.3
Pruned CNN (512 1×1 filters)	77.53	85.0
Original CNN + CLMIP	87.38	104.2

1×1 convolutional coding, we believe that the large-scale parameters brought by the fully connected layer may be beneficial in some cases. Maybe, the large structural space brought about by the massive parameters in fully connected layer provide more possibilities and better choices for the exploration of the model structure.

4.6 Manifold structure visualization

To observe the effect of manifold invariance limitation in the pruning process, we extract the deep convolutional activation features (DeCAF) (Donahue 2014) of pruned network and use t-SNE method (Pezzotti et al. 2017) to project them into a two-dimensional plane. The activation value of the last hidden layer is extracted as DeCAF, and the manifold structures of the features under different pruning methods are shown in Fig. 10. Consistent with the results shown in confusion matrix, the clusters of 'trunk road' and 'motorway' are easy to interfere with each other, especially in methods of DCP and ADMM. The other two road types are more likely to form their own clusters due to their distinctive characteristics. Obviously, the manifolds of data representation under CLMIP method is closer to the original model and more distinguishable than the other methods. This proves that the capture and description of data manifold by network can be related to its generalization ability. As the representation of implicit knowledge hidden in the training samples, data manifold can reflect the process of feature extraction and organization of model. On the other hand, we observe the presence

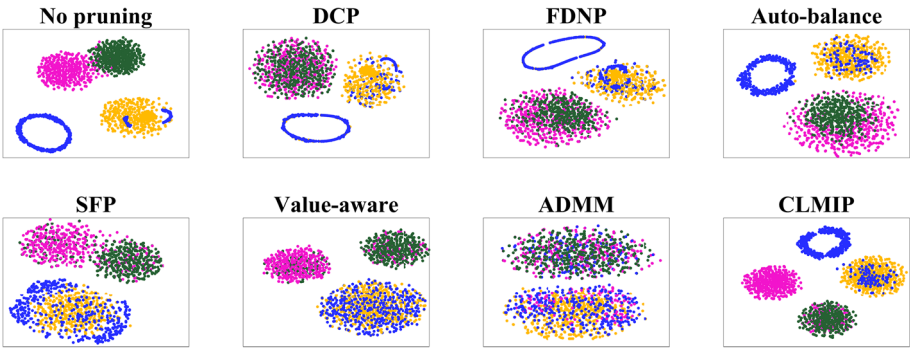


Fig. 10 The t-SNE (Pezzotti et al. 2017) visualization results of manifold structures of DeCAF features extracted by pruned CNNs with different pruning methods. The 'blue', 'yellow', 'pink' and 'green' points represent the 'urban road', 'off-road', 'trunk road' and 'motorway', respectively

of blue points (urban road) in the yellow cluster (off-road). DeCAF feature reflects the organizational structure of the samples, but there is still a decoding process between the sample and its category label. In other words, dimensionality reduction visualization can reflect the distribution characteristics of samples, but it cannot guarantee the accuracy. Even after decoding, there are classification errors in the results.

5 Conclusion

In this paper, we proposed a cross-layer manifold invariance based pruning method named CLMIP for network compression to help it complete real-time road type classification in low-cost vision system. To the best of our knowledge, this is the first time to prune deep CNN based on the invariance limitation of data manifolds. CLMIP is also generic and independent of network structures and therefore it can be easily extended to other complex models. Finally, experimental results on the image classification of four road types demonstrate the effectiveness of CLMIP, which achieves state-of-the-art performance with a prediction speed of 26 frames per second. Smaller memory and computational requirements of deep CNN are helpful for real-time image processing, making the model easier to be deployed on mobile systems or embedded devices. In addition, CLMIP can be regarded as a tool to furtherly investigate the dependence of model structure on network optimization and generalization.

In fact, the proposed CLMIP method can be also used in multiple neural network-based computer vision tasks in autonomous cars or mobile robots, such as pedestrian detection, obstacle avoidance and path planning. In the future, we plan to improve the road type classification performance of the pruned CNN from the following three aspects.

(1) A more reasonable baseline CNN can be designed to provide a good initial position in the structural space.
(2) A more accurate estimation of data manifold structures in different dimensions can be accomplished by using higher-order representations.
(3) The mathematical properties of the method can be analyzed, such as the expected value of the manifold consistency across dimensions.

This work will be further studied in the future.

Acknowledgements This work was supported by National Key R&D Program of China (Grant No. 2018YFC0831503), National Natural Science Foundation of China (Grant No. 61571275), China Computer Program for Education and Scientific Research (Grant No. NGII20161001), Shandong Provincial Natural Science Foundation (Grant No. ZR2014FM010 and No. ZR2014FM030), and Fundamental Research Funds of Shandong University (Grant No. 2018JC040).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Appendix

For the convenience of readers following and understanding this paper, we have annotated all variables in Table 6.

Table 6 Annotations of symbols

Symbols	Annotations
$\mathbf{x}$	Sample images
$\mathbf{y}$	Category labels of road types
$\mathbf{W}$	Weights in the deep CNN
$\mathbf{b}$	Bias in the deep CNN
$\mathbf{h}_i$	Output of i-th layer
$\sigma(\cdot)$	Element wise activation function
τ	Adjustable parameter
$f(\cdot)$	Final output of deep CNN
$\mathcal{L}$	Loss function
C	Number of categories
λ	Regularization intensity
α	Learning rate
M	Mini-batch size
ξ	Total pruning rate
ζ	Pruning step size
$\boldsymbol{\varphi}$	Cross-dimensional manifold descriptor
ϕ	Cosine similarity
ψ	Average reconstruction error
$\mathbf{M}$	Suppression mask
γ	Validation loss threshold
L	Number of layers in the deep CNN
$prune$ $(\mathbf{W})$	Number of pruned weights
$total$ $(\mathbf{W})$	Number of total weights
$\boldsymbol{\theta}_0/\boldsymbol{\theta}_*$	Initialized/ultimate weights of deep CNN
$\mathbb{M}_0/\mathbb{M}_*$	Initialized/convergent CNN model

References

Bône, A., Colliot, O., & Durrleman, S. (2018). Learning distributions of shape trajectories from longitudinal datasets: a hierarchical model on a manifold of diffeomorphisms. In *IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 9271–9280), IEEE Press: Salt Lake City.

Bosch, A., Zisserman, A., & Muñoz, X. (2008). Scene classification using a hybrid generative/discriminative approach. *IEEE Transactions Pattern Analysis and Machine Intelligence, 30*(4), 712–727.

Chen, Q., et al. (2015). Contextualizing object detection and classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 37*(1), 13–27.

Chollet, F. (2107). Xception: Deep learning with depthwise separable convolutions. In *IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1800–1807), IEEE Press: Honolulu.

Ding, J., Chen, B., Liu, H., & Huang, M. (2016). Convolutional neural network with data augmentation for SAR target recognition. *IEEE Geoscience and Remote Sensing Letters, 13*(3), 364–368.

Ding, X., Ding, G., Han, J., & Tang, S. (2018). Auto-balanced filter pruning for efficient convolutional neural networks. In *AAAI conference on artificial intelligence (AAAI)* (pp. 6797–6804), AAAI Press: New Orleans.

Donahue, J. et al. (2014). DeCAF: A deep convolutional activation feature for generic visual recognition. In *International conference on international conference on machine learning (ICML)* (pp. 647–655), ACM Press: Beijing.

Ess, A., Müller, T., Grabner H., & Van Gool, L. (2009). Segmentation-based urban traffic scene understanding. In *British machine vision conference (BMVC)* (pp. 84.1–84.11), BMVA Press: London.

Gong, C., Yang, C., Yao, X., Guo, L., & Han, J. (2018). When deep learning meets metric learning: Remote sensing image scene classification via learning discriminative CNNs. *IEEE Transactions on Geoscience and Remote Sensing, 56*(5), 2811–2821.

Guan, H., Yan, W., Yu, Y., Zhong, L., & Li, D. (2018). Robust traffic-sign detection and classification using mobile LiDAR data with digital images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 11*(5), 1715–1724.

He, Y., Kang, G., Dong, X., Fu, Y., & Yang, Y. (2018). Soft filter pruning for accelerating deep convolutional neural networks. In *International joint conference on artificial intelligence (IJCAI)* (pp. 1–8), Morgan Kaufmann Press: Stockholm.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *IEEE international conference on computer vision (ICCV)* (pp. 1026–1034), IEEE Press: Santiago.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 770–778), IEEE Press: Las Vegas.

Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. In *IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 7132–7141), IEEE Press: Salt Lake City.

Iandola, F. N. et al. (2017). SqueezeNet: AlexNet-level accuracy with 50 × fewer parameters and < 0.5 MB model size. In *International conference on learning representations (ICLR)*, Toulon, France. https://openreview.net/pdf?id=S1xh5sYgx.

Jansen, P., Mark, W., Heuvel, J. C., & Groen, F. C. A. (2005). Colour based off-road environment and terrain type classification. In *IEEE international conference on intelligent transportation systems (ICITS)* (pp. 61–66), IEEE Press: Vienna, Austria.

Jeong, S., Simeone, O., & Kang, J. (2018). Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning. *IEEE Transactions on Vehicular Technology, 67*(3), 2049–2063.

Ketkar, N. (2017). *Deep learning with python* (1st ed., pp. 195–208). Berkeley: Apress.

Ko, J., Kim, D., Na, T., & Mukhopadhyay, S. (2018). Design and analysis of a neural network inference engine based on adaptive weight compression. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 38*(1), 109–121.

Kwong, K. K. (2016). Some sharp Hodge Laplacian and Steklov eigenvalue estimates for differential forms. *Calculus of Variations and Partial Differential Equations, 55*(2), 38.

Lemley, J., Bazrafkan, S., & Corcoran, P. (2017). Smart augmentation learning an optimal data augmentation strategy. *IEEE Access, 5,* 5858–5869.

Li, K., Hariharan, B., & Malik, J. (2016). Iterative instance segmentation. In *IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 3659–3667), IEEE Press: Las Vegas.

Li, X., Wang, W., Hu, X., & Yang, J. (2019). Selective kernel networks. In *IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 510–519), IEEE Press: Long Beach.

Li, Y., et al. (2016b). Revisiting batch normalization for practical domain adaptation. *Pattern Recognition, 80,* 109–117.

Lin, S. et al. (2019). Towards optimal structured CNN pruning via generative adversarial learning. In *IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 2785–2794), IEEE Press: Long Beach.

Liu, W. et al. (2016). SSD: Single shot MultiBox detector. In *European conference on computer vision (ECCV)* (pp. 1–17), Springer: Amsterdam.

Liu, Z., Xu, J., Peng, X., & Xiong, R. (2018). Frequency-domain dynamic pruning for convolutional neural networks. In *Advances in neural information processing systems (NIPS)* (pp. 1–11), MIT Press: Vancouver.

Menouar, H., et al. (2017). UAV-Enabled intelligent transportation systems for the smart city: Ap-plications and challenges. *IEEE Communications Magazine, 55*(3), 22–28.

Miguel, J., et al. (2015). A general framework for constrained Bayesian optimization using information-based search. *Journal of Machine Learning Research, 17*(1), 5549–5601.

Mioulet, L., Breckon, T., Mouton, A., Liang, H., & Morie, T. (2013) Gabor features for real-time road environment classification. In *IEEE international conference on industrial technology (ICIT)* (pp. 1117–1121), IEEE Press: Cape Town.

Mohammad, M., Kaloskampis, I., & Hicks, Y. (2015). Evolving GMMs for road-type classification. In *IEEE international conference on industrial technology (ICIT)* (pp. 1670–1673), IEEE Press: Seville.

Neyshabur, B., Bhojanapalli, S., McAllester, D., & Srebro, N. (2017). Exploring generalization in deep learning. In *Advances in neural information processing systems (NIPS)* (pp. 5947–5956), MIT Press: Canada.

Nguyen, V. D., Nguyen, H. V., Tran, D. T., Lee, S. J., & Jeon, J. W. (2017). Learning framework for robust obstacle detection, recognition, and tracking. *IEEE Transactions on Intelligent Transportation Systems, 18*(6), 1633–1646.

Ozgunalp, U., Fan, R., Ai, X., & Dahnoun, N. (2017). Multiple lane detection algorithm based on novel dense vanishing point estimation. *IEEE Transactions on Intelligent Transportation Systems, 18*(3), 621–632.

Pan, X., et al. (2018). Safe screening rules for accelerating twin support vector machine classification. *IEEE Transactions on Neural Network and Learning Systems, 29*(5), 1876–1887.

Park, E., Yoo, S., & Vajda, P. (2018). Value-aware quantization for training and inference of neural networks. In *European conference on computer vision (ECCV)* (pp. 1–18), Springer Press, Munich.

Paul, A., et al. (2018). Improved random forest for classification. *IEEE Transactions on Image Processing, 27*(8), 4012–4024.

Pezzotti, N., et al. (2017). Approximated and user steerable tSNE for progressive visual analytics. *IEEE Transactions on Visualization and Computer Graphics, 23*(7), 1739–1752.

Rajeswaran, A., Lowrey, K., Todorov, E. V., & Kakade, S. M. (2017). Towards generalization and simplicity in continuous control. In *Advances in neural information processing systems (NIPS)* (pp. 6550–6561), MIT Press: Canada.

Rasiwasia, N., & Vasconcelos, N. (2008). Scene classification with low dimensional semantic spaces and weak supervision. In *IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1–6), IEEE Press: Anchorage.

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. In *IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 6517–6525), IEEE: Honolulu.

Russakovsky, O., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision, 115*(3), 211–252.

Shen, X., Tian, X., Liu, T., Xu, F., & Tao, D. (2018). Continuous dropout. *IEEE Transactions on Neural Networks and Learning Systems, 29*(9), 3926–3937.

Smith, J. S., Bo, W., & Wilamowski, B. M. (2018). Neural network training with levenberg-marquardt and adaptable weight compression. *IEEE Transactions on Neural Networks and Learning Systems, 30*(2), 580–587.

Szegedy, C. et al. (2015). Going deeper with convolutions. In *IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1–9), IEEE Press: Boston.

Tang, I., & Breckon, T. (2011). Automatic road environment classification. *IEEE Transactions on Intelligent Transportation Systems, 12*(2), 476–484.

Trittler, M., Rothermel, T., & Fichter, W. (2016). Autopilot for landing small fixed-wing unmanned aerial vehicles with optical sensors. *Journal of Guidance, Control and Dynamics, 39*(9), 1–11.

Tu, L. W. (2011). *An introduction to manifolds*. New York: Springer. https://doi.org/10.1007/978-0-387-48101-2.

Tung, F., & Mori, G. (2020). Deep neural network compression by in-parallel pruning-quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 42*(3), 568–579.

Wang, Y., Xu, C., Xu, C., & Tao, D. (2018). Adversarial learning of portable student networks. In *AAAI conference on artificial intelligence (AAAI)* (pp. 4260–4267), AAAI Press: New Orleans.

Wu, H., & Prasad, S. (2018). Semi-supervised deep learning using pseudo labels for hyperspectral image classification. *IEEE Transactions on Image Processing, 27*(3), 1259–1270.

Yang, Y., Zhong, Z., Shen, T., & Lin, Z. (2018). Convolutional neural networks with alternately updated clique. In *IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 2413–2422), IEEE Press: Salt Lake City.

Yang, X., et al. (2018b). Hyperspectral image classification with deep learning models. *IEEE Transactions on Geoscience and Remote Sensing, 56*(9), 5408–5423.

Yang, L., et al. (2018c). Improving deep neural network with multiple parametric exponential linear units. *Neurocomputing, 301,* 11–24.

Yu, X., Yu, Z., & Ramalingam, S. (2018). Learning strict identity mappings in deep residual networks. In *IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 4432–4440), IEEE Press: Salt Lake City.

Zhang, T. et al. (2018). A systematic DNN weight pruning framework using alternating direction method of multipliers. In *European conference on computer vision (ECCV)* (pp. 1–18), Springer Press, Munich.

Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In *IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 6848–6856), IEEE Press: Salt Lake City.

Zhang, J., et al. (2017). Road recognition from remote sensing imagery using incremental learning. *IEEE Transactions on Intelligent Transportation Systems, 18*(11), 2993–3005.

Zheng, Q., Tian, X., Jiang, N., & Yang, M. (2019). Layer-wise learning based stochastic gradient descent method for the optimization of deep convolutional neural network. *Journal of Intelligent and Fuzzy Systems, 37*(4), 5641–5654.

Zheng, Q., Tian, X., Yang, M., Wu, Y., & Su, H. (2020a). PAC-Bayesian framework based drop-path method for 2D discriminative convolutional network pruning. *Multidimensional Systems and Signal Processing, 31*(3), 793–827.

Zheng, Q., Yang, M., Tian, X., Jiang, N., & Wang, D. (2020b). A full stage data augmentation method in deep convolutional neural network for natural image classification. *Discrete Dynamics in Nature and Society*. https://doi.org/10.1155/2020/4706576.

Zheng, Q., Yang, M., Yang, J., Zhang, Q., & Zhang, X. (2018). Improvement of generalization ability of deep CNN via implicit regularization in two-stage training process. *IEEE Access, 6,* 15844–15869.

Zhong, J., Ding, G., Guo, Y., Han, J., & Wang, B. (2018). Where to prune: Using LSTM to guide end-to-end pruning. In *International joint conference on artificial intelligence (IJCAI)* (pp. 1–7), Morgan Kaufmann Press: Stockholm.

Zhuang, Z. et al. (2018). Discrimination-aware channel pruning for deep neural networks. In *Advances in neural information processing systems (NIPS)* (pp. 1–12), MIT Press: Vancouver.