applied sciences

MDPI

Article

Research on the Application of Pruning Algorithm Based on Local Linear Embedding Method in Traffic Sign Recognition

Wei Wang [1,2,*] and Xiaorui Liu [2]

[1] Foundation Department, Liaoning Technical University, Huludao 125105, China
[2] School of Electronics and Information Engineering, Liaoning Technical University, Huludao 125105, China; lxiaorui586@gmail.com
* Correspondence: wangwei@lntu.edu.cn; Tel.: +86-13591996798

Abstract: Efficient traffic sign recognition is crucial to facilitating the intelligent driving of new energy vehicles. However, current approaches like the Vision Transformer (ViT) model often impose high storage and computational demands, escalating hardware costs. This paper presents a similarity filter pruning method based on locally linear embedding. Using the alternating direction multiplier method and the loss of the locally linear embedding method for the model training function, the proposed pruning method prunes the operation model mainly by evaluating the similarity of each layer in the network layer filters. According to the pre-set pruning threshold value, similar filters to be pruned are obtained, and the filter with a large cross-entropy value is retained. The results from the Belgium Traffic Sign (BelgiumTS) and German Traffic Sign Recognition Benchmark (GTSRB) datasets indicate that the proposed similarity filter pruning based on local linear embedding (SJ-LLE) pruning algorithm can reduce the number of parameters of the multi-head self-attention module and Multi-layer Perceptron (MLP) module of the ViT model by more than 60%, and the loss of model accuracy is acceptable. The scale of the ViT model is greatly reduced, which is conducive to applying this model in embedded traffic sign recognition equipment. Also, this paper proves the hypothesis through experiments that "using the LLE algorithm as the loss function for model training before pruning plays a positive role in reducing the loss of model performance in the pruning process".

Keywords: visual transformer; local linear embedding; alternating direction multiplier method; Jensen–Shannon divergence; filter pruning

Citation: Wang, W.; Liu, X. Research on the Application of Pruning Algorithm Based on Local Linear Embedding Method in Traffic Sign Recognition. *Appl. Sci.* **2024**, *14*, 7184. https://doi.org/10.3390/app14167184

Academic Editor: Douglas O'Shaughnessy

Received: 3 July 2024
Revised: 8 August 2024
Accepted: 11 August 2024
Published: 15 August 2024

1. Introduction

Road traffic sign recognition is a research hotspot in the intelligent driving of new energy vehicles. "Made in China 2025" requires intelligent driving vehicles to reduce energy consumption by more than 10% compared with conventional vehicles. This is achieved by optimizing the driving path of the car, reasonably planning the speed of the vehicle, and rapidly recognizing traffic signs. Correct and rapid identification of traffic signs can help smart cars drive safely in accordance with traffic rules, improve energy utilization, and reduce polluting emissions, thereby preventing violations and accidents.

In 2011, Ciresan [1] applied convolutional neural networks to traffic sign recognition. Deep neural networks perform well in traffic sign experiments, but most classification networks for traffic sign identification are still limited by the performance of onboard hardware devices in practical applications. Additionally, these networks with good classification effects often require deeper network layers for training, which indicates that more parameters and computing resources are needed. Thus, in the case of limited hardware capabilities, using pruning algorithms to trim the network model of traffic sign recognition can reduce the number of redundant parameters and the scale of the network model, making it more widely applicable in real life. Therefore, research on pruning algorithms for traffic sign recognition models is significant.

Currently, methods proposed to reduce computing costs and scale down large-scale network models mainly include parameter quantization [2], low-rank decomposition [3], knowledge distillation [4], parameter sharing [5], and pruning [6–9]. Among these, the compression work for traffic sign recognition models includes:

Method	Specific	Advantage	Shortcoming
VGG [10]	Based on the VGG network, the channel pruning method based on Lasso regression is used to reduce the convolutional channel and compress the width of the network horizontally.	This method compresses the scale of the traffic sign recognition network, which is conducive to the real-time application of traffic sign systems in urban road conditions.	If there is a set of highly correlated features, Lasso regression tends to choose one of the features and ignore all the others, which leads to instability in the results.
Knowledge-based distillation framework [11]	Two lightweight convolutional neural network structures are designed; the first deep network model is used as the teacher model, and the second shallow model is used as the student model. Through knowledge distillation, the student model can learn the function mapping similar to the teacher model and then trim the redundant feature channels according to the scale factor that tends to zero in the Batch Normalization layer.	According to the experimental data in this paper, the compressed network model has a more compact structure and better performance than the mainstream traffic sign recognition methods.	The experimental data in this paper indicate that compared with the mainstream traffic sign recognition methods, the compressed network model has a more compact structure and better performance.
A lightweight YOLOv3 traffic sign detection algorithm [12]	Use both shallow and deep feature extraction in convolutional neural networks.	This method prunes the model while maintaining the accuracy of the model, thus establishing a lightweight traffic sign detection model with stronger robustness.	

Among the model compression methods for traffic sign recognition, model pruning is also a primary focus of current research. According to the structural analysis of pruning objects, there are mainly two types of model pruning methods: unstructured pruning and structured pruning. The target of unstructured pruning is the connection or neuron within the network layer of the model, characterized by the smallest pruning granularity. Although this method can achieve a higher compression rate while maintaining the experimental performance of the model, the model structure becomes a sparse matrix after pruning. This requires special support from the underlying hardware and computing acceleration libraries; otherwise, it is challenging to achieve substantial performance improvement in the application of the pruned model.

Currently, more researchers are focusing on structured pruning, specifically the filter pruning method, which does not require special support from software and hardware platforms. Filter pruning effectively reduces the number of network parameters and memory usage. This method directly prunes the filters in the network layer without introducing any special structure or additional operations, making filter pruning "orthogonal" to other model compression and acceleration techniques, and it can be easily combined with other technologies.

Filter pruning algorithms generally rank filters based on specific evaluation criteria and remove those deemed less important. However, within each network layer, there are often filters with similar functions, which limits the extent of model pruning. While filter pruning significantly impacts model performance, similar-function filters may still remain post-pruning. Additionally, neural network weight pruning algorithms require extra hardware for data storage, whereas filter pruning methods do not face this issue. These

pruning algorithms primarily focus on evaluating the importance of model filters but often overlook the impact of pruning on the manifold structure of the model's weight parameters.

To address these issues, this paper considers the influence of changes in the manifold structure of weight parameters on model performance during filter pruning. Using the Vision Transformer (ViT) model [13] to classify traffic sign images, we propose a similarity filter pruning algorithm based on local linear embedding (SJ-LLE). During model training, the Local Linear Embedding (LLE) method and the Alternating Direction Method of Multipliers (ADMM) are used as loss functions during iterative update training before model pruning. After obtaining the model with updated weights, the similarity of filters is evaluated by calculating the Jensen–Shannon (JS) divergence value between filters in each network layer. Finally, by calculating the cross-entropy of filters, those with small cross-entropy among similar filters are pruned. This compresses the model size, reduces the hardware requirements for model deployment, and ensures the ViT model remains effective. The experimental performance loss during traffic sign recognition is within an acceptable range.

This paper is organized as follows. The first part describes the overall framework and related theoretical basis of the proposed SJ-LLE pruning method. The second part provides the dataset used in the experiment and the preprocessing operations performed on the images before model training. The third part presents the related experiments and their results and analysis. The final part summarizes the research and provides an outlook for future research work.

2. Method

This paper employs the Alternating Direction Method of Multipliers (ADMM) to address the model pruning problem and accelerate model training. The primary feature of the Local Linear Embedding (LLE) method in manifold learning is to convert high-dimensional data into low-dimensional data without destroying the manifold structure. By using this feature of the LLE method as the loss function during model training and taking the weight parameters of the pre-trained model as the input of the loss function, the initial model is trained to obtain the pre-trained model. After multiple iterations of training, the similarity filter pruning method proposed in this paper is applied to prune the filters of the trained model. This reduces the number of model parameters, allowing the traffic sign recognition model to operate without relying on additional storage devices in practical applications, making it widely applicable on resource-limited platforms. The pruning operation of the Vision Transformer (ViT) model for traffic sign recognition proposed in this paper is shown in Figure 1, and the process of pruning the model with the proposed method is as follows:

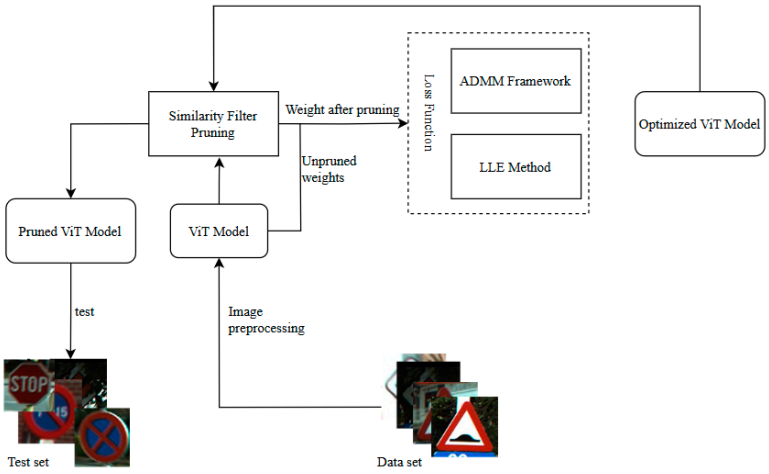


Figure 1. Overall framework of the algorithm.

Step 1: Image preprocessing, such as normalization and image enhancement of dataset images before model training.

Step 2: Pre-training: The model is trained on the dataset through a certain number of iterations to obtain the pre-trained model.

Step 3: Model optimization: ADMM and LLE methods are combined to optimize weights. Similarity filter pruning is performed on the pre-trained model in step 2, the weight parameters before and after the model pruning are used as the input of the loss function (the LLE method and ADMM), and the new model is obtained after iterative training.

Step 4: Pruning of similar filters: Prune the model after iterative training in Step 3 by using a similarity filter pruning method based on JS_ divergence. By calculating the JS_ divergence value among the filters in each layer of the network, similar filter pairs to be pruned are obtained by sorting the filters according to the JS_ divergence value. Then, the cross-entropy value of the filter is calculated, and the one with a smaller cross-entropy value in the middle of the similar filter to be pruned is cut out based on the preset pruning threshold β.

Step 5: Fine-tune and train the pruned model in step 4 through iterations to recover the performance loss of the model during pruning operations.

Step 6: Testing. Use the fine-tuned model in step 5 to test the experimental performance of the model on the test dataset.

2.1. ViT Model

The Transformer model achieves good performance in many computer vision tasks. If there is enough data for pre-training, the performance of ViT exceeds that of CNN, overcoming the Transformer's lack of inductive bias and obtaining a better migration effect in downstream tasks. However, when the training dataset is not large enough, ViT usually performs worse than ResNets under the same size, and this is because Transformer lacks an inductive bias, i.e., a priori knowledge or assumption made in advance, compared to CNN. CNN has two types of inductive bias: one is locality, i.e., adjacent regions on the image have similar characteristics; the other is translation invariance, $f(g(x)) = g(f(x))$, where g represents the convolution operation and f represents the translation operation. When the CNN has the above two inductive biases, there is a lot of prior information, and only relatively little data is needed to learn a better model. The flowchart of the ViT model is shown in Figure 2.

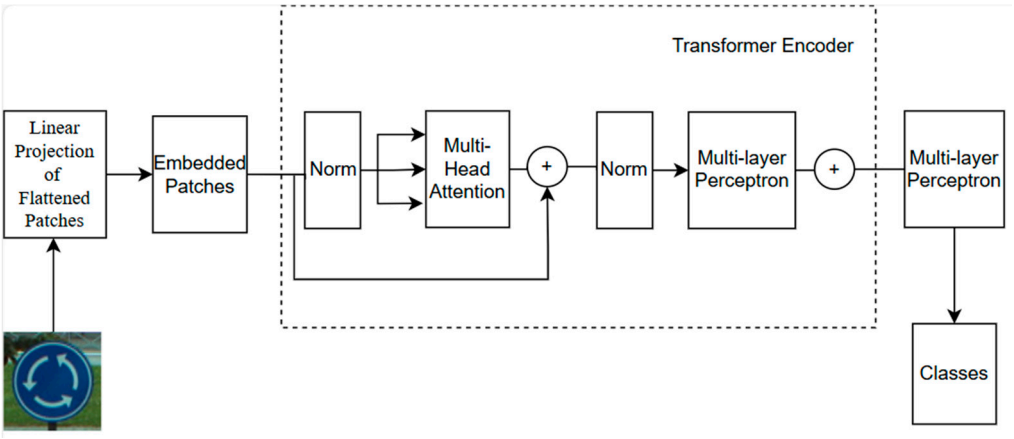


Figure 2. ViT model structure diagram.

A ViT block can be divided into the following four steps:

1. Patch embedding: For example, if the input image size is 224 × 224, the image is divided into fixed-size patches, and the patch size is 16 × 16. Then, each image will generate 224 × 224/16 × 16 = 196 patches, that is, the length of the input sequence is 196, the dimension of each patch is 16 × 16 × 3 = 768, and the dimension of the linear

projection layer is 768 × *N* (*N* = 768). Therefore, the dimension after the input passes through the linear projection layer is still 196 × 768, i.e., there are 196 tokens in total, and the dimension of each token is 768. A special character cls is also added here, so the final dimension is 197 × 768. In this way, a vision problem is transformed into a seq2seq problem with patch embedding.

2. Position coding: ViT also needs to add position coding, and the ViT position code has a total of N lines. Each line represents a vector, and the dimension of the vector and the embedded dimension of the input sequence is the same. So, after adding position code information, the dimension is still 197 × 768.
3. Linear layer/multi-head attention mechanism/linear layer: The output dimension of the linear layer is still 197 × 768. In the multi-head-attention mechanism, the input is mapped to QKV first. If there is only one head, the dimension of QKV is 197 × 768; if there are 12 heads (768/12 = 64), the dimension of QKV is 197 × 64, and there are 12 groups of QKV in total; finally, the output of 12 groups of QKV is spliced, the output dimension is 197 × 768, and then through a linear layer, the dimension is still 197 × 768.
4. MLP: Enlarge and scale down the dimension (197 × 768 to 197 × 3072, and then shrink to 197 × 768).

After a ViT block, the dimensions are still the same as the input, which is 197 × 768, so multiple blocks can be stacked. Finally, the output corresponding to the special character cls is used as the final output of the encoder, i.e., the final image presentation (another approach is to take an average of all tokens without adding cls characters), and the image is classified through the MLP module finally.

2.2. ADMM Principle

ADMM mainly solves the minimization problem of the objective function of two variables with equality constraints, and it mixes the decomposability of the dual rise algorithm with the superior convergence of the multiplier method, thus providing an effective method to solve the non-convex problem with combinatorial constraints [14,15]. In the process of model pruning, ADMM decomposes the original pruning problem into two subproblems. It first solves subproblem 1 by the traditional gradient descent method, and then it introduces a quadratic term iteration to solve subproblem 2; finally, it performs system weight pruning, and its weight-optimization problem is shown in Equation (1) [16]:

$$\underset{\text{subject to } W_i \in S_i,\ i=1,\ldots,N}{\text{minimize}} f(\{W_i\},\{b_i\}) + \sum_{i=1}^{N} g_i(W_i) \tag{1}$$

where $Z_i = W_i$, i = 1,..., N. $S_i = \{W_i \mid \text{card}(W_i) \leq l_i\}$, i = 1,..., N. S_i,..., S_N are a nonconvex set.

$$g_i(W_i) = \begin{cases} 0 & if\ \ W_i \in S_i \\ +\infty & else \end{cases}$$

where S_i is a set of weights for different pruning methods, and $g(\cdot)$ is an indicator function containing weight sparsity constraints. Then, Equation (1) is rewritten from the augmented Lagrange formula to Equation (2):

$$L_\rho(\{Z_i\},\{W_i\},\{\gamma_i\},\{b_i\}) = f(\{W_i\},\{b_i\}) + \sum_{i=1}^{N} g_i(Z_i) + \sum_{i=1}^{N} tr\left[\gamma_i^T(W_i - Z_i)\right] + \sum_{i=1}^{N} \frac{\rho_i}{2}\|W_i - Z_i\|_F^2 \tag{2}$$

where γ_i is the Lagrangian multiplier of the equal constraint of $Z_i = W_i$ in Equation (1), and it is the same as the W_i dimension. The positive scalar $\{\rho_1, \rho_2,..., \rho_N\}$ is the augmentation

parameter, tr(·) represents the trace of the matrix, and $\|\cdot\|_2^2$ represents the Frobenius norm. $Ui = (1/\rho)\gamma i$, and Equation (2) can be rewritten as Equation (3):

$$L_\rho(\{Z_i\},\{W_i\},\{\gamma_i\},\{b_i\}) = f(\{W_i\},\{b_i\}) + \sum_{i=1}^{N} g_i(Z_i) + \sum_{i=1}^{N} \frac{\rho_i}{2}\|W_i - Z_i + U_i\|_F^2 + \sum_{i=1}^{N} \frac{\rho_i}{2}\|U_i\|_F^2 \tag{3}$$

The following steps (4), (5), and (6) are repeated.

$$\left\{Z_i^{k+1}\right\} := \underset{\{Z_i\}}{argmin}\ L\rho\left(\left\{b_i^{k+1}\right\},\{Z_i\},\left\{W_i^{k+1}\right\},\left\{U_i^{k}\right\}\right) \tag{4}$$

$$\left\{W_i^{k+1}, b_i^{k+1}\right\} := \underset{\{Wi\},\{bi\}}{argmin}\ L\rho\left(\left\{Z_i^{k}\right\},\{W_i\},\{bi\},\left\{U_i^{k+1}\right\}\right) \tag{5}$$

$$U_i^{k+1} := U_i^{k} + \rho\left(W_i^{k+1} - Z_i^{k+1}\right) \tag{6}$$

until the following conditions are met:

$$\|W_i^{k+1} - Z_i^{k+1}\| \le \varepsilon_i,\ \|Z_i^{k+1} - U_i^{k+1}\| \le \varepsilon_i,$$

where k is the iteration argument, and Equation (4) can be expressed as Equation (7).

$$\underset{Zi}{min} E(x,y) \sim D[\max L(Z,x,y)] + \frac{\rho}{2}\sum_{i=1}^{N}\|Z_i - W_i^k + U_i^k\|_2^2 \tag{7}$$

Equation (5) can be expressed as Equation (8).

$$minimize\ \ f(\{W_i\},\{bi\}) + \sum_{i=1}^{N}\frac{\rho i}{2}\|W_i - Z_i^{k+1} + U_i^k\|_F^2 \tag{8}$$

The former term of Equation (8) is a loss function, and the latter term can be used as a special L2 regularization. Since the two terms before and after Equation (8) is differentiable, this paper uses the stochastic gradient descent method to solve the formula. From the gradient formula of the augmented Lagrangian of W_i and b_i, we have

$$\frac{\partial L_\rho\left(\{W_i\},\{bi\},\left\{Z_i^k\right\},\left\{U_i^k\right\}\right)}{\partial W_i} = \frac{\partial f(\{W_i\},\{bi\})}{\partial W_i} + \rho i\left(W_i - Z_i^k + U_i^k\right) \tag{9}$$

$$\frac{\partial L_\rho\left(\{W_i\},\{b_i\},\left\{Z_i^k\right\},\left\{U_i^k\right\}\right)}{\partial b_i} = \frac{\partial f(\{W_i\},\{b_i\})}{\partial b_i} \tag{10}$$

Since $g_i(\cdot)$ is an indicator function of the S_i set, the global optimal solution to the problem can be expressed as Formula (11).

$$W_i^{k+1} = \prod_{Si}\left(Z_i^{k+1} + U_i^k\right) \tag{11}$$

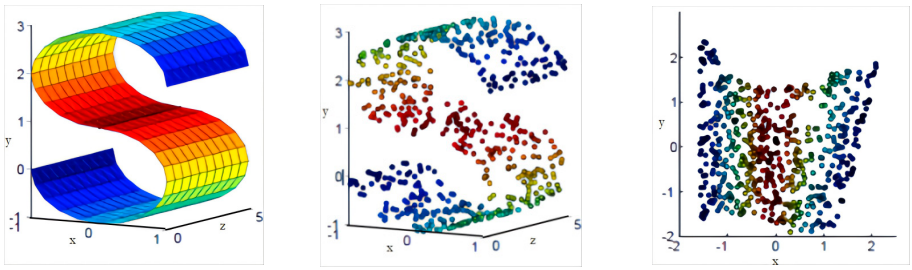
where $\prod_{S_i}(\cdot)$ denotes the Euclidean projection of $Z_i^{k+1} + U_i^k$ on the S_i set, and S_i is a non-convex set, so calculating $\prod_{S_i}(\cdot)$ is a difficult problem. However, considering the special structure of $Si = \{W|card(W) \le li\}$, the solution of Equation (4) is to maintain the maximum importance of the element l_i of $W_i^{k+1} + U_i^k$ and set the remaining elements to 0. Finally, the bivariate U_i is updated according to Equation (6). In this way, an iteration of ADMM is completed. The pruning algorithm based on ADMM is as follows (Algorithm 1).

Algorithm 1: ADMM-based pruning algorithm

Input: dataset D, ADMM iteration number K, N filters to be pruned layer by layer, pruning threshold β_i, augmented hyperparameter ρ, the weight set S_i of different pruning methods.
Output: Weight parameter W.
1: for k = 1, 2,..., K do
2: Sample from $D(x, y)$, solve for Z in Equation (7).
4: Applying the Adam optimizer to the outer minimum of Equation (9) yields $\{Z_i{}^k\}$
5: Solve Equation (11) using Formula (10) to obtain $\{W^i{}_k\}$
6: $U_i^{k+1} := U_i^k + \rho\left(Z_i^{k+1} - W_i^{k+1}\right)$
7: end for

2.3. Local Linear Embedding Method

In manifold learning, LLE methods aim to maintain linear relationships between samples within a neighborhood. High-dimensional data often represent complex real-world objects, such as gene expressions or image pixels. After these data are reduced by LLE, the distribution in the low-dimensional space reflects the relative positions and similarity relationships of the original high-dimensional data, thereby preserving both the local and global structure. Figure 3 illustrates the process of LLE mapping 3D data (B) to 2D data (C). The data points in the low-dimensional space maintain the structure of the high-dimensional data, demonstrating the effectiveness of LLE.



(**a**) (**b**) (**c**)

Figure 3. Mapping of high-dimensional data to low-dimensional data. (**a**) Raw data structure; (**b**) Three-dimensional data; (**c**) Two-dimensional data.

Figure 4 demonstrates that data points in the low-dimensional space should maintain the reconstruction relationship of the original high-dimensional data [17]. The "reconstruction relationship" means that, through the process of dimensionality reduction, the data points in the low-dimensional space can "reconstruct" or "reflect" the data structure of the original high-dimensional space to some extent. This involves retaining enough information in the low-dimensional space to recover important features of the high-dimensional data. If the dimensionality reduction is successful, similar sample points in the high-dimensional space should also remain similar in the low-dimensional space, effectively extracting and representing the key information of the data.

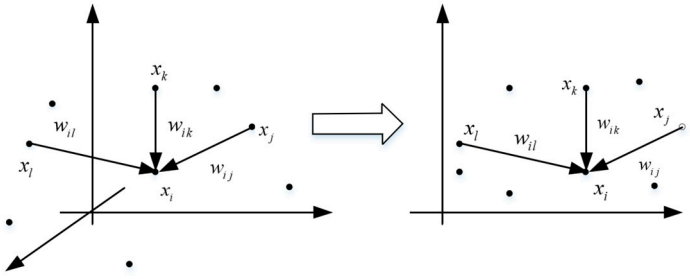


Figure 4. The samples in the high-dimensional space maintain reconstruction relations in the low-dimensional space.

Based on the above theory, a hypothesis is made: "using the LLE algorithm as the loss function trained before model pruning plays a positive role in reducing the loss of model performance in the pruning process". According to this assumption, the LLE method is used as the loss function during model pruning training, and the specific calculation steps are as follows:

1. The weight parameter of the filter in each layer of the network before the model is pruned is taken as the sample point, and the k nearest neighbor points of each sample point are calculated by the KNN strategy; meanwhile, the k sample points with the closest Euclidean distance relative to the sample point are specified as the k nearest neighbors of the sample point, and k is a pre-given value.
2. Calculate the local reconstruction weight matrix α of the sample points and use the mean squared error as the loss function of the regression problem. The calculation formula of mean squared error is shown in Equation (12).

$$J(\omega) = \sum_{i=1}^{N} \left|\left| w_i - \sum_{j \in Q(i)} \alpha_{ij} w_j \right|\right|_2^2 \tag{12}$$

where the sample points w_i and w_j represent the weight parameters before the model is pruned, w_j is the jth neighbor of w_i, a_{ij} is the weight coefficient of the linear relationship between the sample point w_i and its neighbor w_j, and $Q(_i)$ represents the set of k neighbors of w_i. The weight coefficient a_{ij} is normalized, and the weight coefficient needs to satisfy the condition in Formula (13).

$$\sum_{j \in Q(i)} \alpha_{ij} = 1 \tag{13}$$

For sample w_i that is not in the neighborhood of sample w_j, let the corresponding α_{ij} = 0. The weight coefficients of the linear relationship between w_i and its neighbors can be determined by Formulas (12) and (13). Then, the above optimization problems are solved by the Lagrangian multiplier method, and Formula (12) is obtained by matrixization.

$$J(\alpha) = \sum_{i=1}^{N} \left|\left| x_i - \sum_{j \in Q(i)} \alpha_{ij} w_j \right|\right|_2^2 = \sum_{i=1}^{N} \left|\left| \sum_{j \in Q(i)} \alpha_{ij} (w_i - w_j) \right|\right|_2^2 = \sum_{i=1}^{N} \gamma_i^T (w_i - w_j)(w_i - w_j)^T \gamma_i \tag{14}$$

where $\gamma_i = (\alpha_{i1}, \alpha_{i2}, \cdots \alpha_{ik})^T$, $Zi = (w_i - w_j)(w_i - w_j)^T$, and $j \in Q(i)$, Then, Equation (14) can be converted to Equation (15).

$$J(\gamma) = \sum_{i=1}^{k} \gamma_i^T Z_i \tag{15}$$

Through the Lagrangian multiplier method, Equations (14) and (15) are combined into one optimization goal, i.e., Equation (16).

$$L(\gamma) = \sum_{i=1}^{k} \gamma_i^T Z_i \gamma_i + \lambda(\gamma_i^T 1_k - 1) \tag{16}$$

where vector 1k is k b-dimensional all-1 vectors. The derivative of $L(\gamma)$ with respect to γ and its value to 0 gives $\gamma i = \lambda' Z_i^{-1} 1k$, where $\lambda' = -1/2\lambda$ is a constant, and the weight coefficient γ_i is obtained after normalization by using $\gamma_i^T 1_k = 1$. Based on this, Formula (17) is obtained.

$$\gamma_i = \frac{Z_i^{-1} 1_k}{1_k^T Z_i^{-1} 1_k} \tag{17}$$

where γ_i is the weight coefficient of the weight parameter and its neighboring points before pruning.

3. Assuming that the weight coefficient γ_i can still maintain the corresponding linear relationship after model pruning, i.e., the corresponding mean squared error loss function is the smallest. The calculation formula of the minimized loss function is shown in Formula (18).

$$L(\psi) = \sum_{i=1}^{N} \left|\left| \psi_i - \sum_{j=1}^{N} \alpha_{ij} \psi_j \right|\right|_2^2 \tag{18}$$

Equation (18) is similar to Equation (12), where Ψ_i and Ψ_j are the weight parameters after pruning the model, and α_{ij} is the weight coefficient obtained in Equation (12).

2.4. JS_ Divergence Values

The similarity filter pruning method based on local linear embedding (SJ-LLE) first measures the similarity of filters in each layer of the network according to a criterion, identifying pairs of filters with similar functions to be pruned. Then, one filter in each similar filter pair is removed. In traditional neural network pruning processes, the ADMM framework is combined with filter pruning methods based on the L2 norm. However, some filters with small L2 norm values still hold importance. The similarity filter pruning method can remove filters with similar functions, directly reducing the computational and parameter load of the network model while minimizing accuracy loss through model compression. The SJ-LLE method uses the Jensen–Shannon (JS) divergence value among the filters as the criterion for judging similarity, calculating the JS_ divergence value between each filter and others in each layer of the network. Filters with divergence values less than the pruning threshold are identified as similar filter pairs to be pruned. Subsequently, the method calculates the information entropy value of the filters and removes the filter with the smaller information entropy value in the similar filter pair.

The SJ-LLE method uses Equation (19) to calculate the JS_ divergence value between each filter and other filters in each layer of the network [18].

$$JS(P||Q) = \frac{1}{2} KL(P||\frac{P+Q}{2}) + \frac{1}{2} KL(Q||\frac{P+Q}{2}) \tag{19}$$

During the pruning process, the weights of the model are substituted into the JS_ divergence Formula (19) to obtain Formula (20). Meanwhile, the weights are arranged according to their positions, the weight distributions of the two filters are compared for similarity, and the filter with a JS_ divergence value less than the pruning threshold is used as a similar filter pair to be pruned.

$$JS(P||Q) = \frac{1}{2} \sum_{i=1}^{N} p(x_i) \log \frac{p(x_i) + q(x_i)}{2} + \frac{1}{2} \sum_{i=1}^{N} q(x_i) \log \frac{p(x_i) + q(x_i)}{2} \tag{20}$$

P and Q, respectively, represent the proportion of weights in the corresponding positions of the filter, and the weight distribution of the two filters is compared based on the JS_ divergence value and information entropy value in the comparison process.

To retain the filter with a richer amount of information in a similar filter pair, the information entropy value of the filter is taken as the evaluation criterion: the larger the information entropy value, the greater the amount of information involved. Thus, the information entropy of the filter is calculated, the filter with a larger information entropy value in the similar filter pair is retained, and the filter with a smaller information entropy value in the similar filter is removed. In this way, the weight element of the filter is zeroed, and the calculation formula of the information entropy [18] is shown in Formula (21).

$$H(P) = \sum_{i=1}^{N} p(x_i) \log\left(\frac{1}{p(x_i)}\right) \tag{21}$$

The specific steps of the pruning algorithm are as follows (Algorithm 2):

Algorithm 2: Similar filter pruning algorithm based on JS_ divergence

Input:
N: filters to be pruned layer by layer.
β_i: pruning threshold.
1: Obtain the absolute value of the weight parameter of N filters in layer i in the pre-trained model.
2: Calculate the JS_ divergence value between N filters in step 1 and between each filter and other filters, and sort these filters according to the JS_ divergence value.
3: Select the pre-$N\beta_i$ pair filter sorted in step 2 as the filter pair to be pruned.
4: Calculate the information entropy value of each filter, remove the filter with a small information entropy value from the filters to be pruned in step 3, and retain the filter with a larger information entropy value.

2.5. Similarity Filter Algorithm Based on Locally Linear Embedding

In the SJ-LLE algorithm proposed in this section, ADMM decomposition model pruning optimization is first used to accelerate model training. Combined with the LLE method from Section 2.3 as the loss function during the iterative training process before model pruning, the filter weights before and after pruning are used as the input for the loss function. The optimized model is trained over multiple iterations. Then, the SJ-LLE pruning method is applied to prune the filters of the optimized model. The JS_ divergence value between every two filters in each network layer is calculated, and the similarity of the filters in each group is evaluated based on the JS_ divergence value, where a smaller divergence value indicates higher similarity. The similar filter pair to be pruned is determined according to the preset pruning threshold. Finally, the information entropy value of the filters is calculated, and the filter with the smaller information entropy value in the similar filter pair is removed. In this way, the SJ-LLE algorithm reduces the number of model parameters while ensuring the accuracy of the model, without relying on additional storage devices when compressing the model. Consequently, the model can be more widely used in practical scenarios. The flow of the SJ-LLE algorithm is shown in Figure 5.

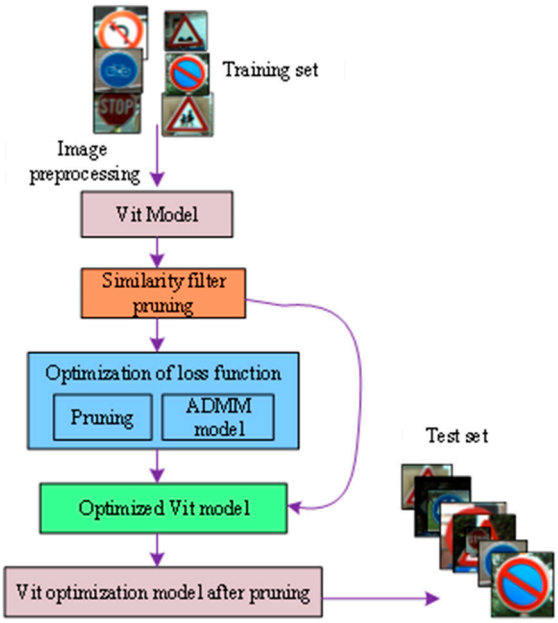


Figure 5. The flow chart of the SJ-LLE method.

The pseudocode of the SJ-LLE method is as follows (Algorithm 3):

Algorithm 3: Similar filter pruning algorithm based on local linear embedding

Input: dataset D, the number of pre-training iterations K', the number of model weight optimization iterations K'', the number of pruning iterations K, N filters to be pruned layer by layer, pruning threshold β_i, augmented hyperparameter ρ, the weight set S_i of different pruning methods.
//Output: Weight parameter W
1: for $k = 1, 2, ..., K'$ do
2: Sample from D (x, y)
3: Data normalization and data enhancement preprocessing of (x, y) yields (x', y')
4: Take (x', y') as input to initial training for iterative training
5: end for
//Get pre-trained model
6: for $k = 1, 2, ..., K''$ do
7: Take samples from D (x, y)
8: Data normalization and data enhancement preprocessing of (x, y) yields (x', y')
9: Take (x', y') as input for model weight optimization training.
10: Apply Algorithm 2 to obtain the pruned weight parameter Z by using the weight parameter W of the pre-trained model
11: Take the weight parameters W and Z as inputs to Algorithm 1 and solve for Z in Equation (7).
12: Applying the Adam optimizer to the outer minimum of Equation (11) yields $\{Z^i{}_k\}$
13: $U_i^{k+1} := U_i^k + \rho\left(Z_i^{k+1} - W_i^{k+1}\right)$
14: end for
15: //Get the weight optimization model
16: for $k = 1, 2, ..., K$ do
17: Take samples from D (x, y)
18: Data normalization and data enhancement preprocessing of (x, y) yields (x', y')
19: Take (x', y') as input for model weight optimization training.
20: Optimize the weight parameter of the model W' using Algorithm 2 to obtain the pruned weight parameter Z'
21: end for
22: //Get weight model after pruning

3. Dataset and Preprocessing

3.1. Dataset

This paper uses the BelgiumTS dataset [19] and GTSRB dataset [20] for model training and testing.

The BelgiumTS dataset includes 62 categories, where the training set contains 4575 images, the test set contains 2520 images, and some images in the BelgiumTS dataset are shown in Figure 6.

Figure 6. Some images in the BelgiumTS dataset.

There are 43 categories of traffic signs in the GTSRB dataset, where the training set contains 34,799 images, the validation set contains 4410 images, and the test contains 12,630 images. The image size is 32 × 32, with three color channels. The number of images in each category of the test set varies greatly, so the number of samples in each category is equalized to 50,690, and some images of the GTSRB dataset are shown in Figure 7.

Figure 7. Some images of the GTSRB dataset.

The CIFAR100 dataset has 100 classes. Each class has 600 color images of size 32 × 32, where 500 images are used as the training set and 100 images as the test set. Each image has two labels, namely fine_labels and coarse_labels, representing the fine-grained and coarse-grained labels of the image, respectively.

3.2. Image Preprocessing Methods

Before the model is trained, the size of the input image is reduced to 32 × 32 × 3. Since traffic signs do not have different meanings due to different sign colors, this paper ignores the color differences of images in the data when the model is trained for traffic sign recognition, and only the brightness differences of different parts of the image are considered. In this paper, the HSV and YUV color spaces are used: the V channel in the HSV space and the Y channel in the YUV space are taken. To enhance the brightness of the image, the Y channel and the V channel are respectively equalized in the histogram as follows: for an image with n pixels and pixel values ranging from [0, L−1], let r_k represent the k-th pixel level and nk represent the number of pixels in the image, and then the histogram probability density $p(r_k)$ of r_k is $p(rk) = \frac{nk}{n}$, k= 0, 1, 2 . . . , $L - 1$. The cumulative distribution function is shown in Formula (22).

$$Sk = \left(\sum_{j=0}^{k} p(rj)\right)^2 \times \left(1 + 2 \times \left(1 - \sum_{j=0}^{k} p(rj)\right)\right) \quad (22)$$

where, r_j represents the j-th pixel level, $p(r_j)$ represents the probability density of the j-th pixel level, and the final image is M = 255 × S_k.

As shown in Figure 8, Figure 8a shows the original image of the traffic sign, Figure 8b shows the image of the V channel in HSV through histogram equalization, and Figure 8c shows the image of the Y channel in YUV through histogram equalization. It can be seen from Figure 8b,c that after histogram equalization of the image, the image data is converted into a range of [−1, 1], and the poorly lit image has a better graphic outline. Through this preprocessing method, the contrast of parts of the image is enhanced without affecting the overall contrast.

(a) (b) (c)

Figure 8. The pre-processed traffic sign image. (**a**) shows the original traffic sign image; (**b**) shows the image with histogram equalization for the V channel in HSV; (**c**) shows the image with histogram equalization for the Y channel in YUV.

Due to the complex environment of traffic signs outdoors, the image of traffic signs will produce different changes. To eliminate the influence of external factors on the image, during model training, the maximum range of image rotation angle is set to 10 degrees, the maximum random magnification is set to 1.2, the maximum random horizontal offset is set to 0.08, and the range of up and down offset is set to 0.08. According to the characteristics of the deep learning model, images with a small size can be enhanced by geometric transformations (such as translation, rotation, scale stretching, contrast adjustment, and color transformation). For images with a large size, mean reduction is used to reduce the image size.

4. Experiments

4.1. Experimental Details

In this section, a variety of pruning methods are applied to the ViT model for traffic sign recognition, and multiple sets of comparative experiments are designed on different datasets. Meanwhile, a series of operations, such as the SJ-LLE pruning method and other pruning methods proposed in this paper, are clipped-fine-tuned under the same model pruning rate. Then, the fine-tuned model is tested on the BelgiumTS dataset and the GTSRB dataset, and the test results of different pruning methods on the ViT model are analyzed and compared. The effectiveness of the proposed ViT model pruning algorithm for traffic sign recognition is investigated. Also, for the CIFAR100 dataset, multiple sets of experiments are designed between the pruning method and its comparison method, and the experimental results are analyzed and compared to prove the hypothesis, "using the LLE algorithm as the loss function trained before model pruning plays a positive role in reducing the loss of model performance in the pruning process".

The pruning algorithm is applied to the existing ViT model for clipping, and the benchmark model is pre-trained based on the initial model of the ViT model with the initial learning rate set to 7×10^{-4} and the batch size of 256,100 epochs. The network structure used for the experiment is shown in Figure 2. In the pruning process, the network is cut with different pruning rates based on the pre-trained model. The learning rate is 7×10^{-4}, the number of iterations is 30, and other training parameter settings are the same as those of the pre-training. After the model is pruned, fine-tuning training is conducted, and the model is fine-tuned using the same parameter settings as the pre-training to restore the accuracy lost in some experiments. For the same data set and the same model with different cropping methods, the experimental parameters of pre-training and fine-tuning training are the same. In the experiment using the same pruning method, whether to combine the LLE method as the loss function for model training before pruning is the only difference, and the rest of the experimental parameter settings are the same.

4.2. Experimental Evaluation Index

Top-1 accuracy, Top-5 accuracy, pruning model parameters (Params), and floating-point operations (FLOPs) are usually adopted to measure the effectiveness of the model pruning algorithm.

(1) Top-1 accuracy

Top-1 accuracy refers to the accuracy of the first category consistent with the actual result, i.e., the predicted label takes the largest one in the last probability vector as the prediction result. If the classification with the greatest probability in the prediction result is correct, the prediction is correct; otherwise, the prediction is wrong.

(2) Top-5 accuracy

Top-5 accuracy refers to the accuracy of the top five categories containing the actual results, i.e., among the top five categories with the largest probability vector, the probability of correct occurrence indicates a correct prediction; otherwise, the prediction is wrong.

(3) Params

Params refers to the total number of parameters that need to be trained in model training, and the number of parameters in an ordinary convolutional layer is calculated as $(K_h \times K_w \times C_i + 1) \times C_o$, where K_h is the height of the convolution kernel, K_w is the width of the convolution kernel, C_i is the number of channels of the input, C_o is the number of channels of the output, and +1 is the bias. The formula for calculating the parameter quantity of the fully connected layer of the model is $(C_i + 1) \times C_o$, where C_i is the number of nodes in the input, C_o is the number of nodes in the output, and +1 is the bias.

(4) FLOPs

FLOPs represent the number of floating-point operations per second (FLOPs), which can be used to measure the complexity of an algorithm or model. The calculation formula of FLOPs is $[C_i \times K_w \times K_h + 1] \times W \times H \times C_o$, where C_i is the number of channels in the input; C_o is the number of channels in the output; $C_i \times K_w \times K_h$ is the calculation amount of a convolution multiplication operation; $C_i \times K_w \times K_h - 1$ is the calculation amount of a convolution addition operation; W and H are the length and width of the feature map, respectively; K_h is the height of the convolution kernel; and K_w is the width of the convolution kernel.

When other pruning methods are taken for comparing experiments, the difference between the accuracy of the pruned model and the accuracy of the initial model (the unpruned model) is used as the main basis for evaluating the effect of the pruning algorithm. Additionally, the compression and acceleration effects of the network model are evaluated by the two indicators of Params and FLOPs. When the pruning network contains fewer parameters, it is indicated that the compression ratio of the network is greater, and the reduction of parameter size leads to a decrease in FLOPs. In this case, the network running speed is improved because fewer FLOPs are required. The decline of the two standards of Params and FLOPs in the post-pruning model is controlled by the pruning rate β_i, and a better compression effect can be obtained with a larger pruning rate. When a convolutional layer is pruned at the pruning rate β_i, the parameter size of the layer will be reduced to $(1 - \beta_i) \times (K_h \times K_w \times C_i + 1) \times C_o$, and the amount of floating-point calculation needed will be reduced to $(1 - \beta_i) \times [C_i \times K_w \times K_h + 1] \times W \times H \times C_o$.

4.3. Experimental Results and Analysis

4.3.1. Experimental Results and Analysis of Hypothesis Proofs

This paper puts forward a hypothesis in Section 2.3: "using the LLE algorithm as a loss function for model training before pruning plays a positive role in reducing the loss of model performance during pruning". To prove this hypothesis, corresponding experiments are conducted on the CIFAR-100, BelgiumTS, and GTSRB datasets. Then, the influence of combining the LLE method in model pruning on the loss of model accuracy is analyzed, and the experimental results are shown in Tables 1 and 2.

Table 1. The ablation experiments of the ViT model with different pruning methods on the CIFAR-100 dataset.

Pruning Method\Pruning Rate	Baseline	60%	70%	80%	90%	95%
ASFP_LLE	67.27	64.82	63.54	60.98	45.64	37.93
ASFP	67.27	64.06	62.32	59.67	43.15	35.45
PFEC_LLE	67.27	66.26	65.43	61.56	48.46	36.98
PFEC	67.27	65.37	63.83	60.43	46.27	34.27
LWCE_LLE	67.27	64.38	63.16	60.44	48.73	33.10
LWCE	67.27	63.84	61.78	59.32	46.35	30.46
FPGM_LLE	67.27	65.32	63.41	60.85	43.96	33.52
FPGM	67.27	64.63	62.25	59.41	42.25	31.24

Table 2. The ablation experiments of the VGG-16 model with different pruning methods on the CIFAR-100 dataset.

Pruning Method\Pruning Rate	Baseline	60%	70%	80%	90%	95%
ASFP_LLE	94.47	90.77	90.24	83.98	63.54	50.93
ASFP	94.47	90.23	89.82	82.19	61.57	48.66
PFEC_LLE	94.47	91.74	91.03	85.91	65.34	49.76
PFEC	94.47	91.74	91.05	90.14	84.85	48.12
LWCE_LLE	94.47	90.72	89.97	83.06	66.45	49.75
LWCE	94.47	90.16	89.05	81.85	64.66	47.14
FPGM_LLE	94.47	91.37	89.64	86.13	86.13	47.32
FPGM	94.47	90.7	88.73	85.07	59.74	45.44

Table 1 compares ASFP [21] with ASFP_LLE methods, PFEC [22] with PFEC_LLE methods, LWCE [23] with LWCE_LLE methods, and FPGM with FPGM_LLE methods as the four groups of methods in the demonstration experiments in this section. In each group of comparison methods, whether to combine the LLE method as the loss function in the training process before pruning is used as the only variable in the experiment, and the other experimental parameters are the same, such as the clipped model, pruning method, data set, pruning ratio, etc.

ASFP pruning takes the L2 norm size of the filter as the evaluation standard, and the filter with a small norm value is removed;

The PFEC pruning method calculates the L1 norm value of the channel and then crops out the smaller channel. The amount of clipping depends on the acceleration ratio;

The LWCE pruning method reduces the parameters required by the network by pruning unimportant connections (L1/L2 norms) in the trained network.

In the experiment, the ASFP_LLE, PFEC_LLE, LWCE_LLE, and FPGM_LLE methods are compared with the baseline method, which is realized by combining the LLE method with the corresponding pruning method, i.e., the LLE method is used as the loss function before the model pruning. In this section, the ViT model is cut by using the above pruning method on the CIFAR-100 dataset, and the pruning ratios are 60%, 70%, 80%, 90%, and 95%, respectively. The experimental results are shown in Table 1.

In Table 1, the 1st row shows the clipping ratio of the model; the 1st column shows the pruning method of the clipping model; the 3rd, 5th, 7th, and 9th rows show the accuracy of the model tested on the dataset after the model is cut with the four baseline pruning methods ASFP, PFEC, LWCE, and FPGM; the 2nd, 4th, 6th, and 8th rows show the accuracy of the pruning model tested on the dataset after the model is cut by the corresponding comparative pruning method.

By analyzing the experimental results in Table 1, it can be found that when cutting the ViT model at 60% to 95% by using the ASFP_LLE pruning method, the accuracy of the model is 0.76%, 1.22%, 1.31%, 2.49%, and 2.48% higher than that of the cutting the model by using the ASFP pruning method, respectively. When cutting the model at 60% to 95% by the PFEC_LLE pruning method, the accuracy of the model test is 0.89%, 1.59%, 1.13%, 2.19%,

and 2.71% higher than that of the post-cutting model by using the PFEC pruning method, respectively. When cutting the model at 60% to 95% by using the LWCE_LLE pruning method, the accuracy of the model was 0.54%, 1.38%, 1.12%, 2.38%, and 2.64% higher than that of the cropped model by using the LWCE pruning method, respectively. When cutting the model at 60% to 95% by using the FPGM_LLE pruning method, the accuracy of the model is 0.69%, 1.16%, 1.44%, 1.71%, and 2.28% higher than that of the cropped model by using the FPGM pruning method, respectively. Under the same pruning ratio, the ViT model is pruned using the baseline method and the comparison method, respectively, and the experimental results indicate that the test results of the four comparison methods combined with the LLE method are better than those of the baseline method without the LLE method, which proves that the LLE algorithm plays a positive role in reducing the loss of model performance in the pruning process.

The experimental results presented in Table 2 demonstrate that the conclusions align with those in Table 1. When using the ASFP_LLE, PFEC_LLE, LWCE_LLE, and FPGM_LLE pruning methods on the ViT model, these methods achieved higher model accuracy on the dataset compared to their respective benchmark pruning methods. For instance, at a pruning rate of 95%, the accuracies were 2.27%, 1.64%, 2.61%, and 1.88% higher than those of the benchmark methods, respectively. These results indicate that combining the LLE algorithm with the loss function during model pre-pruning training achieves the desired model compression rate while effectively minimizing the loss of model accuracy.

Assuming that in the experiment, the experimental parameters of each set of baseline methods and their respective comparison methods are the same, the experimental results of 60% to 95% of the large-scale pruning of the model indicate that when the pruning ratio is the same, the comparison method combined with the LLE method can complete the pruning work of the model and improve the accuracy of the model after pruning compared with the original pruning method without the LLE method. Therefore, the experimental results can prove the validity of the hypothesis proposed in Section 2.3 that "using the LLE algorithm as a loss function for model training before pruning plays a positive role in reducing the loss of model performance during the pruning process".

4.3.2. Experimental Results and Analysis of SJ_LLE Method Pruning

In this experiment, the ASFP pruning, HYDRA pruning, FPGM pruning, and SJ-LLE pruning methods proposed in this paper are used to prune the model at different pruning ratios. Since excessive clipping of the model will cause the accuracy of the model to decrease significantly, and the model with excessive performance loss is not conducive to practical applications, the maximum pruning ratio is set to 70% in the experiment. The experiment is conducted on the BelgiumTS and GTSRB datasets, and the accuracy of the model after pruning is analyzed under the same parameters as those in other experimental conditions.

The SJ-LLE method is a similarity filter pruning algorithm based on JS_ divergence proposed in Section 2.5.

The ASFP [21] and Filter method uses the L2 norm size of the filter as the evaluation criterion and cuts the filter with a small norm value.

The HYDRA and Filter pruning method [6] regards the pruning target as an empirical risk-minimization problem and then uses SGD to solve the minimization problem. By letting the training target guide the search for connections to be pruned, it proposes a scaled initialization of importance score, which is a key driver behind the high robustness and accuracy of the compression network.

The FPGM pruning method [24] selects the filter with the largest replaceable contribution. Specifically, the median geometry of the filters within the same layer is calculated. Depending on the characteristics of the median geometric, the filter near it can be represented by the remaining filters.

The ViT model is pruned by using four pruning methods, namely SJ-LLE, ASFP, HYDRA, Filter, and FPGM, and the accuracy of the post-pruning model is tested on the BelgiumTS and GTSRB datasets, respectively. The results are shown in Tables 3 and 4. In

these two tables, different pruning ratios are set for the model pruned in the first experiment, and rows 2–5 show the accuracy of the model tested on the dataset after pruning at different ratios by using the four pruning methods, respectively.

Table 3. Comparison of accuracy (%) of the ViT model on the BelgiumTS dataset.

Pruning Method\Pruning Rate	Baseline	10%	20%	40%	50%	70%
SJ-LLE	98.78	98.72	98.13	97.95	96.88	96.09
ASFP	98.78	98.06	96.67	95.85	94.75	93.19
HYDRA	98.78	98.13	97.10	95.42	94.84	94.15
Filter	98.78	97.96	95.64	93.82	92.96	92.12
FPGM	98.78	98.03	97.15	95.39	95.04	94.51

Table 4. Comparison of accuracy (%) of the ViT model on the GTSRB dataset.

Pruning Method\Pruning Rate	Baseline	10%	20%	40%	50%	70%
SJ-LLE	99.03	99.09	98.72	98.05	97.58	96.04
ASFP	99.03	98.26	96.82	95.94	94.48	93.84
HYDRA	99.03	98.28	97.72	96.12	95.51	94.66
Filter	99.03	97.22	95.81	95.95	93.47	92.95
FPGM	99.03	98.53	98.01	96.95	96.03	95.18

On the BelgiumTS dataset, the ViT model is clipped by using four different pruning methods, and the results are shown in Table 2. It can be seen from this table that the model after pruning with the SJ-LLE method has obvious advantages in experimental accuracy. For example, when the model pruning ratio is 50%, the model accuracy is only lost by 1.90%. Under the same pruning ratio, the test accuracy loss of the model pruned with the ASFP pruning method, the HYDRA method, the Filter method, and the FPGM method is 2.12, 2.07, and 1.96 times that of the SJ-LLE method, respectively. When the model pruning ratio is 70%, the model accuracy loss is only 2.69%, which is 2.08, 1.72, 3.06, and 1.59 times higher than that of the model pruned with the ASFP pruning method, HYDRA method, and FPGM method, respectively.

Then, the model is pruned by using the SJ-LLE, ASFP, HYDRA, Filter, and FPGM pruning methods on the GTSRB dataset, and the results are shown in Table 3. From the experimental results in Table 3, it can be found that the model performance after pruning with the SJ-LLE method is better than that of other pruning methods under the same cutting ratio. For example, when the pruning ratio is 70%, the experimental accuracy loss of the model after pruning by the SJ-LLE method is 1.99%. Under the same pruning ratio, the test accuracy after pruning the SJ-LLE method is 2.20%, 1.38%, 3.09%, and 0.86% higher than that of the ASFP pruning method, HYDRA method, and FPGM method, respectively.

5. Conclusions and Future Works

In the field of intelligent driving for new energy vehicles, a traffic sign recognition model optimization method based on SJ-LLE pruning is proposed. This paper primarily introduces the general framework of the method, detailing its process and providing an in-depth exposition of the formula derivation and theoretical research. The experimental results fully verify the hypothesis. Experiments show that the SJ-LLE method achieves excellent pruning effects across multiple datasets, significantly improving the model's precision loss and compression ratio. The results indicate that this method achieves better model compression and enhances the recognition speed of traffic signs with minimal accuracy loss.

However, the correlation between the improved accuracy of the model and energy consumption was not considered. Compared to other pruning methods, the proposed method demonstrates superior accuracy in testing the pruned model under equal proportion pruning conditions. Nevertheless, experimental results reveal that, without signifi-

cantly sacrificing model accuracy, the pruning ratio achieved by this method is only 70%. Therefore, further improvements to the pruning method are necessary to achieve a higher model compression ratio. Future research will focus on the relationship between model performance and energy consumption in the context of new energy intelligent driving vehicles, combined with the traffic environment.

Author Contributions: Conceptualization, W.W.; Data curation, X.L.; Writing—original draft, W.W. and X.L.; Writing—review & editing, X.L.; Funding acquisition, W.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was financially support by the National Natural Science Foundation of China (Grant Nos. 61772249, 61702241), the Basic Research Projects of the Liaoning Provincial Department of Education (Grant Nos. LJKZ0362).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to data privacy restrictions.

Acknowledgments: The authors would like to thank all the reviewers who participated in the review during the preparation of this manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cireşan, D.; Meier, U.; Masci, J.; Schmidhuber, J. A committee of neural networks for traffic sign classification. In Proceedings of The 2011 International Joint Conference on Neural Networks, San Jose, CA, USA, 31 July–05 August 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 1918–1921.
2. Xu, Y.; Wang, Y.; Zhou, A.; Lin, W.; Xiong, H. Deep neural network compression with single and multiple level quantization. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
3. Qiu, Q.; Cheng, X.; Sapiro, G. DCFNet: Deep neural network with decomposed convolutional filters. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 4198–4207.
4. Shen, Y.; Xu, L.; Yang, Y.; Li, Y.; Guo, Y. Self-distillation from the last mini-batch for consistency regularization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 11943–11952.
5. Rahimian, E.; Javadi, G.; Tung, F.; Oliveira, G. DynaShare: Task and instance conditioned parameter sharing for multi-task learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 4535–4543.
6. Sehwag, V.; Wang, S.; Mittal, P.; Jana, S. Hydra: Pruning adversarially robust neural networks. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 19655–19666.
7. Tan, H.; Wu, S.; Du, F.; Chen, Y.; Wang, Z.; Wang, F.; Qi, X. Data pruning via moving-one-sample-out. *Adv. Neural Inf. Process. Syst.* **2024**, *36*, 18251–18262.
8. Khan, M.A.; Park, H.; Chae, J. A lightweight convolutional neural network (CNN) architecture for traffic sign recognition in urban road networks. *Electronics* **2023**, *12*, 1802. [CrossRef]
9. Xu, K.; Wang, Z.; Geng, X.; Wu, M.; Li, X.; Lin, W. Efficient Joint Optimization of Layer-Adaptive Weight Pruning in Deep Neural Networks. In Proceedings of the 2023 IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 1–6 October 2023; pp. 17401–17411.
10. Lin, X.; Zhang, Z.; Liu, L.; Yang, X.; Wang, S. VGG network pruning and quantization for urban road traffic sign recognition. *Small Microcomput. Syst.* **2021**, *6*, 1293–1296.
11. Wang, W. Traffic Sign Recognition Algorithm Based on Convolutional Neural Network Compression. Master's Thesis, Changsha University of Technology, Changsha, China, 2019.
12. Bai, S.L.; Yin, K.X.; Zhu, J.Q. Lightweight Yolov3 traffic sign detection algorithm. *Comput. Mod.* **2020**, *9*, 83–88+94.
13. DosoViTskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Houlsby, N. An image is worth 16 × 16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
14. Leng, C.; Dou, Z.; Li, H.; Zhu, S.; Jin, R. Extremely low bit neural network: Squeeze the last bit out with ad-mm. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
15. He, B.; Xu, S.; Yuan, X. Extensions of ADMM for separable convex optimization problems with linear equality or inequality constraints. In *Handbook of Numerical Analysis*; Elsevier: Amsterdam, The Netherlands, 2023; Volume 24, pp. 511–557.

16. Ye, S.; Xu, K.; Liu, S.; Cheng, H.; Lambrechts, J.H.; Zhang, H.; Zhou, A.; Ma, K.; Wang, Y.; Lin, X. Adversarial robustness vs. model compression, or both? In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 111–120.
17. Ghojogh, B.; Crowley, M.; Karray, F.; Ghodsi, A. *Elements of Dimensionality Reduction and Manifold Learning*; Springer: Berlin/Heidelberg, Germany, 2023.
18. Lei, M. *The Mathematics of Machine Learning: Information Theory*; Version 1; The People's Posts and Telecommunications Press: Beijing, China, 2021; pp. 298–316.
19. Takapoui, R.; Moehle, N.; Boyd, S.; Bemporad, A. A simple effective heuristic for embedded mixed-integer quadratic programming. *Int. J. Control.* **2020**, *93*, 2–12. [CrossRef]
20. Roweis, S.T.; Saul, L.K. Nonlinear dimensionality reduction by locally linear embedding. *Science* **2020**, *290*, 2323–2326. [CrossRef] [PubMed]
21. He, Y.; Dong, X.; Kang, G.; Fu, Y.; Yan, C.; Yang, Y. Asymptotic Soft Filter Pruning for Deep Convolutional Neural Networks. *IEEE Trans. Cybern.* **2019**, *50*, 3594–3604. [CrossRef] [PubMed]
22. Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; Graf Peter, H. Pruning filters for efficient convnets. *arXiv* **2016**, arXiv:1608.08710.
23. Han, S.; Pool, J.; Tran, J.; Dally, W. Learning both weights and connections for efficient neural network. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 1135–1143.
24. He, Y.; Liu, P.; Wang, Z.; Hu, Z.; Yang, Y. Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; IEEE: Piscataway, NJ, USA, 2019.