

CONTEXT-AWARE PRUNING FOR SEMANTIC SEGMENTATION

Technical Field

The present invention relates, in general terms, to systems, neural networks, methods for image pattern recognition. The present invention also relates to methods of pruning of neural networks.

Background

Semantic segmentation involves the prediction of a semantic label for all pixels in an image. Semantic segmentation plays a vital role in computer vision applications such as autonomous driving, robotic navigation, etc. Conventional models for semantic segmentation have a large number of parameters and require high computational power for the dense predictions involved in semantic segmentation. The complexity and computational costs of such conventional models hinder their deployment on mobile devices or embedded devices that have limited resources for computation, storage, and may have a strict requirement on inference latency.

Over-parameterization of most existing Deep Neural Networks (DNNs) such as CNNs is conventionally considered necessary for training the DNNs. Pruning of networks within DNNs aims to remove redundancies in the well-trained and over-parameterized models for faster inference. The pruning of neural networks may also be referred to as sparsification.

Conventional non-structured pruning methods require specialized library or hardware support. Conventional structured pruning approaches focus on pruning the entire structure (e.g., kernel, filter, and even layer). Some conventional pruning methods sparsify the structures together with group lasso, while other conventional sparsification methods involve identifying redundancies in the networks on multi-level structures by imposing sparsity. Filter L1-norm and Geometric Median are regarded as indicators for redundant convolution filters by some conventional pruning methods.

Some conventional pruning methods selectively switch the channels on/off based on the runtime activation. However, such conventional pruning methods still require the deployment of the entire complex model to a target machine to maintain the representation capacity of the

DNNs. Accordingly, such conventional pruning methods are not suitable for systems with tight memory or computational power constraints.

Further, most conventional pruning methods are evaluated for image-level classification networks. Conventional pruning methods when applied to the problem of semantic segmentation suffer from substantial performance degradation.

It would be desirable to overcome or address at least one of the above-described problems, or at least to provide a useful alternative.

Summary

The invention provides a system for image pattern recognition, comprising memory storing a neural network model, the neural network model comprising:

- an input layer for receiving the image;
 - an output layer for outputting the image pattern; and
 - a context-aware guiding module (CAGM) between the input layer and output layer, the CAGM:
 - receiving feature maps supplied by a layer of the neural network;
 - forming a context guiding vector from a layer-wise integrated channel interdependency of the feature maps; and
 - directing contextually informative channel selection based on the context guiding vector,
- wherein the output layer outputs the image pattern based on features of channels selected using the context guiding vector.

In some embodiments, the CAGM forms the context guiding vector by computing a channel affinity matrix and forming the context guiding vector from the channel affinity matrix.

In some embodiments, the CAGM forms the context guiding vector by squeezing the channel affinity matrix based on dimensions of a spatial size of the layer supplying the feature maps.

In some embodiments, the context guiding vector carries an adaptive penalty strength for selectively penalising channels of the neural network.

In some embodiments, the CAGM comprises a normalisation layer and the adaptive penalty strength modifies one or more scaling factors in the normalisation layer.

In some embodiments, the CAGM obtains the context guiding vector β_j^i by:

$$\beta_j^i = \frac{\sum_k a_{j,k}^i - \text{Min}(\sum_k a_{:,k}^i)}{\text{Max}(\sum_k a_{:,k}^i) - \text{Min}(\sum_k a_{:,k}^i)} \in [0,1]$$

wherein $\text{Max}()$ and $\text{Min}()$ compute a maximum and minimum value along a channel dimension of channel k , β_j^i is a guiding factor for channel j in layer i , and $a_{j,k}^i$ is an affinity level between channel j and channel k in layer I , and $a_{:,k}^i$ is an affinity level between channel k and remaining channels in layer i .

In some embodiments, the CAGM directs contextually informative channel selection based on the context guiding vector by applying a modifying scaling factor to each channel based on the context guiding vector, to perform context-aware guided sparsification (CAGS).

In some embodiments, the CAGM performs CAGS based on a channel-to-channel interdependency as determined from the context guiding vector.

In some embodiments, the system further comprises a pruning module for pruning one or more channels of the neural network based on the modified scaling factor for the respective channel.

The invention also provides a neural network model for image pattern recognition, comprising:

- an input layer for receiving the image;
- an output layer for outputting the image pattern; and
- a context-aware guiding module (CAGM) between the input layer and output layer, the CAGM:
 - receiving feature maps supplied by a layer of the neural network;
 - forming a context guiding vector from a layer-wise integrated channel interdependency of the feature maps; and
 - directing contextually informative channel selection based on the context guiding vector,

wherein the output layer outputs the image pattern based on features of channels selected using the context guiding vector.

In some embodiments, CAGM directs contextually informative channel selection based on the context guiding vector by applying a modifying scaling factor to each channel based on the context guiding vector, to perform context-aware guided sparsification (CAGS).

The invention also provides, a method of neural network pruning, comprising:

- receiving, at a context-aware guiding module (CAGM), a plurality of feature maps for an image, each feature map corresponding to a channel of the neural network;
- forming, using the CAGM, a context guiding vector from a layer-wise integrated channel interdependency of the feature maps; and
- directing contextually informative channel selection using the CAGM, based on the context guiding vector.

In some embodiments, directing contextually informative channel selection comprises applying a weight to each channel based on an individual integrated interdependent level of the respective channel relative to one or more other said channels.

The invention also provides a method of image pattern recognition, comprising:

- receiving an image at an input layer;
- formulating a plurality of feature maps for the image, each feature map corresponding to a channel;
- performing the method of neural network pruning according to the disclosure;
- and
- outputting the image pattern from an output layer, based on features of channels selected using the context guiding vector.

Brief description of the drawings

Embodiments of the present invention will now be described, by way of non-limiting example, with reference to the drawings in which:

Figure 1 is a block diagram of a Context-aware Guiding Module (CAGM);

Figure 2 is a block diagram of a CAGM positioned in a Pyramid Pooling Module;

Figure 3 illustrates a method 300 for neural network pruning;

Figure 4 illustrates a method 400 for image pattern recognition;

Figure 5 is a schematic diagram illustrating an unpruned and a pruned neural network;

Figure 6 illustrates an input image and an output image obtained by semantic segmentation;

Figure 7 illustrates a theoretical contextual relationship for the segments of the input image of Figure 6;

Figure 8 illustrates a representation of feature maps and channels obtained from an input image;

Figure 9 illustrates a flowchart of a part of a method of pruning neural networks;

Figure 10 illustrates a series of images on which image segmentation operations are applied using pruned networks according to the embodiments and baseline methods;

Figure 11 illustrates images of a crowd counting operation that may be performed using neural networks pruned using the disclosed methods of pruning;

Figure 12 illustrates images of a crowd counting operation that may be performed using neural networks pruned using the disclosed methods of pruning and corresponding unpruned neural networks;

Figure 13 illustrates a graph of a layer-wise comparison of a part of a SegNet based neural network structure pruned according to the disclosed methods of pruning;

Figure 14 illustrates a graph of a layer-wise comparison of a part of an ICNet based neural network structure pruned according to the disclosed methods of pruning;

Figure 15 illustrates a graph of a layer-wise comparison of a part of a PSPNet50 based neural network structure pruned according to the disclosed methods of pruning;

Figure 16 illustrates a graph of a layer-wise comparison of a part of a PSPNet101 based neural network structure pruned according to the disclosed methods of pruning;

Figure 17 illustrates in graphs the mIoU values and number of parameters (y-axis) for various pruning ratios (x-axis) for the various networks when pruned according to the disclosed methods; and

Figure 18 illustrates graphs of ablation studies on different values of λ_1 and λ_2 pair for the various segmentation networks.

Detailed description

Systems and Methods according to the embodiments perform structured pruning on DNNs (including CNNs) for semantic segmentation. Structured pruning is performed by taking into account the context embedded in feature maps, and by leveraging the channel associations as a cue to guide pruning. When pruning segmentation networks, emphasis is given to the preservation of informative contextual features, whilst channels that exhibit less important contextual properties are discarded by the pruning methods of the embodiments.

Some methods of pruning according to the present disclosure may be referred to as Context-aware Pruning (CAP). CAP comprises channel pruning while considering the context embedded in feature maps. The CAP framework is based on the insight of semantic parsing, wherein the determination of pixel semantics requires abundant aggregation of local abstract features with its surrounding information.

A Context-aware Guiding Module (CAGM) is disclosed. The CAGM quantifies the contextual information among channels into a guiding vector. Next, to distinguish the critical channels in the original model, a Context-aware Guided Sparsification (CAGS) approach is disclosed. The CAGS approach comprises sparsifying the channel-wise scaling factors in the batch normalization (BN) layers under the guidance of CAGM from different inputs. By forcing the scaling factors to zero, the corresponding channels can be regarded as redundant since their

corresponding output will be scaled to zero and hence these filters can be potentially removed. Since the BN layer is generally employed in most networks, the disclosed pruning framework can be easily applied to existing models. Moreover, for CNNs with no normalization layers, simple pseudo scaling factors can be introduced to apply the disclosed pruning methods to such CNNs.

Advantageously, some embodiments incorporate contextual information in intermediate features for guiding channel pruning tailored to semantic segmentation models. The contextual information provided by the CAGM is leveraged to emphasize or de-emphasize the structured sparsification operations. CAGS induces channel-wise sparsity under the contextual guidance from CAGM, and to adaptively reveal the informative channels in the cumbersome model. After removing the redundant channels, a sparsified model can preserve comparable accuracy and information while using a much lesser number of parameters. The sparsified model of the embodiments may therefore be more portable and be suitable for deployment in devices with limited memory, processing power or battery capacity.

The disclosure also exposes opportunities for pruning both large and lightweight segmentation models. A good generalization over different semantic segmentation models is demonstrated via quantitative results. The pruning methods of the embodiments not only effectively remove redundancies in large networks like PSPNet, but also prune lightweight networks like ICNet.

The pruning methods or frameworks when applied to various benchmarks (e.g., CamVid, Cityscapes) demonstrated the generation of compact models for various state-of-the-art segmentation networks with significantly fewer parameters and FLOPs (Floating Point Operations Per Second), while maintaining better performance to (at times outperforming) the original models, compared to all baseline methods.

Unlike image classification tasks, semantic segmentation emphasizes more on local-to-global features aggregation. Systems and methods of the present disclosure exploit the property wherein spatial semantic contextual information can be captured via multiscale pooling or downsampling, which are further fused via various strategies to facilitate pixel-wise prediction. However, from the high-level semantic features that embed such contextual information, the associations or combinations of different channel maps may also contribute differently towards useful contextual information. For example, in a well-trained network, a particular association

of channels activation in the feature maps may represent specific useful context with semantic meaning (e.g., driving lane) for a semantic class (e.g., car), while a different channels association may provide another contextual hint. Thus, to condense such knowledge, channels that always provide useful contextual clues under diverse inputs are preserved, while others can be considered for removal by the embodiments during pruning of networks.

Conventional pruning methods ignore such channel associations on features, and thus the pruned semantic segmentation models performs poorly when channels with influential contextual information are removed. However, it is not straightforward to directly measure channel contextual importance, since different contextual information may come from different channels and they may not be independent to each other. The framework of some embodiments leverages the channel maps affinity for guiding structured sparsification to discover contextual informative channels and exploits the contextual redundancy for pruning semantic segmentation networks.

Context Aware Guiding Module (CAGM)

Figure 1 illustrates a block diagram of a CAGM incorporated in a neural network 100. The neural network 100 embodies several elements of a CNN. There may be one or more intermediate layers of neurons between the various layers identified in Figure 1. A solid line 102 corresponds to a forward pass during training or execution of the neural network 100. The dashed line 104 corresponds to a backward pass during the training of the neural network 100. The backward pass comprises backpropagation in the neural network 100 during training of the neural network.

The neural network 100 may be deployed as a part of a system (not shown) for image pattern recognition. The neural network 100 is provided in a memory and/or storage of the system and is executable by at least one processor of the system. In some embodiments, the neural network 100 is deployed in an edge-computing system (not shown) operating in concert with an edge sensor such as a camera or a surveillance camera capturing images that are processed by the neural network model 100. In some embodiments, the neural network 100 is deployed in an edge-computing system (not shown) provided in an autonomous vehicle and operating in concert with a camera provided in the autonomous vehicle to process image data and generate signals that the systems of the autonomous vehicle receive as input for autonomous navigation.

The neural network 100 has an input layer 124 for receiving the image, an output layer 140 for outputting the image pattern. The neural network 100 also has a context-aware guiding module (CAGM) 110 between the input layer and output layer. The CAGM 110 receives feature maps supplied by a layer of the neural network and forming a context guiding vector 138 from a layer-wise integrated channel interdependency of the feature maps. The CAGM 110 contextually directs informative channel selection based on the context guiding vector 138. The output layer 140 outputs the image pattern based on features of channels selected using the context guiding vector 138.

122 is an input image provided to the input layer 124. The output of the input layer 124 is processed by the pooling layer 126. The pooling layer 126 processes feature maps received as inputs to generate a pooled feature map corresponding to the input image 122. The pooling layer 126 outputs a dimensionally reduced feature map for the input image 122 and this dimension reduction may also be referred to as downsampling. The output of the pooling layer may be processed by one or more convolution layers 128.

A batch normalization (BN) layer 130 processes the output of the convolution layer 128. In some embodiments, the BN layer 130 is implemented according to the publication 'Sergey Ioffe and Christian Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift arXiv preprint arXiv:1502.03167, 2015' the contents of which are hereby incorporated by reference. The output of the BN layer 130 is processed by the activation layer 132 to generate a high-level feature map 134. Layer 140 is an output layer that produces a segmentation prediction exemplified by image 142. Image 144 is a ground truth image that allows the calculation of a loss that is backpropagated through network 100.

The CAGM 110 is applied to the feature maps adjacent to the pooling layer 126 on top of the original network 100. CAGM 110 measures the integrated channel interdependency into a layer-wise vector, namely Context Guiding Vector 138, which is used to direct the contextual informative channels selection in Context-aware Guiding Sparsification (CAGS).

Mathematically, the feature maps in layer i are denoted as $\mathbf{M}^i \in \mathbb{R}^{C_i \times H_i \times W_i}$, where C_i is the number of channels, H_i and W_i represent the spatial size. Channels are the depth of matrices

involved in the convolution process. CAGM first calculates the symmetrical Channel Affinity Matrix 136 $\mathbf{A}^i \in \mathbb{R}^{C_i \times C_i}$ from $\mathbf{M}^i$.

The feature maps $\mathbf{M}^i \in \mathbb{R}^{C_i \times H_i \times W_i}$ are reshaped into $\mathbf{P}^i \in \mathbb{R}^{C_i \times N_i}$, where $N_i = H_i \times W_i$ is the squeezed spatial size. Next, each element in $\mathbf{A}^i$ is obtained via a dot product of the reshaped feature maps:

$$a_{j,k}^i = \mathbf{P}_{j,:}^i \cdot \mathbf{P}_{k,:}^i, \quad (1)$$

where $a_{j,k}^i$ indicates the affinity level between channel j and channel k in layer i . Here, the dot product similarity is adopted, which considers the vectors in-between angle and the magnitude. In alternative embodiments, other similarity measures are adopted to determine the affinity matrix 136.

Secondly, from the obtained Channel Affinity Matrix 136 $\mathbf{A}$, CAGM computes the Contextual Guiding Vector 138 β in each layer, which integrates the affinity of channels into scalars. Considering row j in $\mathbf{A}^i \in C_i \times C_i$, each element $a_{j,:}^i$ represents the affinity level between channel j and the remaining channels in layer i . Then, $a_{j,:}^i$ is normalized into the same scale, i.e., between zero to one, and then $a_{j,:}^i$ is reduced into one dimension using the summation $\sum_k a_{j,k}^i$ for integration.

Finally, the Contextual Guiding Vector 138 β^i is obtained by:

$$\beta_j^i = \frac{\sum_k a_{j,k}^i - \text{Min}(\sum_k a_{:,k}^i)}{\text{Max}(\sum_k a_{:,k}^i) - \text{Min}(\sum_k a_{:,k}^i)} \in [0,1] \quad (2)$$

where $\text{Max}(\cdot)$ and $\text{Min}(\cdot)$ compute the maximum and minimum value along the channel dimension. Each scalar in β is denoted as the Guiding Factor, where β_j^i is the Guiding Factor for channel j in layer i . Iterating all channels in this layer i , their corresponding Guiding Factor are obtained, which represents individual integrated interdependence level with other channels given the network input.

Since the activation on feature maps varies from the network inputs, the corresponding β will be input-dependent as well. To incorporate the mini-batch training, additional min-max normalization on β along the batch dimension each mini-batch before CAGS is conducted to achieve the batch-dependent integration. In this way, both the numerical stability and scale of β are maintained. In some embodiments, no additional trainable parameters or supervision are introduced in the CAGM 110.

The CAGM 100 may be applied to the feature maps in the encoder part of a SegNet based network architecture for image segmentation. Multiple pooling and sub-sampling operations are implemented in the encoder part of a SegNet based network to extract the potential spatial context and to achieve translation invariance over spatial dimension. Similarly, for PSPNet or ICNet based segmentation networks with a dilated backbone, the CAGMs 110 are concatenated to the Pyramid Pooling module of the respective networks, where the features carry richer sub-region contextual information in different scales.

Figure 2 illustrates a block diagram of a plurality of CAGMs 210 incorporated in a neural network 200 wherein each CAGM 210 is positioned next to a Pyramid Pooling Module. Each CAGM 210 comprises components similar to the CAGM 110 of Figure 1. Each CAGM 210 comprise a channel affinity matrix 236 (similar to channel affinity matrix 136) that generates a context guiding vector 238 (similar to context guiding vector 138). The CAGM 110 forms the context guiding vector 138 by computing the channel affinity matrix 136 and forming the context guiding vector 138 from the channel affinity matrix 136. The CAGM 110 forms the context guiding vector 138 by squeezing the channel affinity matrix 136 based on dimensions of a spatial size of the layer supplying the feature maps within the neural network 100.

Context-Aware Guided Sparsification (CAGS)

The CAGM 110 directs contextually informative channel selection based on the context guiding vector 138 by applying a modifying scaling factor to each channel based on the context guiding vector 138. The contextually informative channel selection is also referred to as context-aware guided sparsification (CAGS). The CAGM 110 performs CAGS based on a channel-to-channel interdependency as determined from the context guiding vector 138. A pruning module (CAGM 110) is provided for pruning one or more channels of the neural

network 100 based on the modified scaling factor for the respective channel of the neural network 100.

Figure 3 illustrates a method 300 for neural network pruning or CAGS. At step 310, the CAGM receives a plurality of feature maps for an image. Each feature map corresponds to a channel of the neural network 100. At step 320, the CAGM, forms a context guiding vector β from a layer-wise integrated channel interdependency of the feature maps. The CAGM 110 directs contextually informative channel selection based on the context guiding vector 138 by applying a modifying scaling factor to each channel based on the context guiding vector 138, to perform CAGS.

As the CAGM 110 at step 310 is applied over the well-optimized unpruned models, the initial β reflects the meaningful channel-to-channel interdependency information from different contexts. However, the β computed at step 320 is dependent on each input. In other words, β is computed for each input in a training dataset. To leverage β using all available training data, some embodiments include Context-aware Guided Sparsification (CAGS). CAGS is empirically valid for contextual informative channels selection.

For notation, given a model ϕ with the parameters set θ , the following vanilla loss function for semantic segmentation is used:

$$L(\theta) = \frac{1}{n} \sum_{k=1}^n J(\mathbf{Y}_k; \Phi(\mathbf{X}_k, \theta)) + R_w(\theta) \quad (3)$$

where $\mathbf{X}_k \in \mathbb{R}^{3 \times H \times W}$ and $\mathbf{Y}_k \in \mathbb{R}^{H \times W}$ are the k th input RGB image and its pixel-wise semantic labels, $k = 1, \dots, n$. $J(\cdot)$ is the pixel-level standard segmentation loss. the loss function is intended to minimize the standard loss together with the regularization term R_w (e.g., L2 regularization) on the parameters set to achieve a better generalization and to avoid over-fitting.

To enable the CNNs to obtain better generalization and faster convergence, batch normalization (BN) is employed in most modern architectures, and it can be formulated as follows:

$$y = \gamma \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \quad (4)$$

where μ and σ are the statistical mean and standard deviation over the mini-batch input features. The affine parameters γ and β are the learnable scaling and shifting factors. For channel pruning, the scaling factor γ in the BN layers can be an importance indicator for the corresponding channels. Channels with close-to-zero γ can be regarded as redundant since it is scaled to near-zero activation and they contribute less to the prediction.

At step 330, within the neural network 100, the CAGM directs contextually informative channel selection, based on the context guiding vector β . Contextually informative channel selection includes pruning one or more channels of the neural network 100 based on modified scaling factors ζ for the respective channel. Directing contextually informative channel selection comprises applying a weight to each channel based on an individual integrated interdependent level of the respective channel relative to one or more other said channels.

To learn the redundant channels, the straightforward attempt is to induce sparsity on the whole scaling factors set ζ with vanilla L1 regularization [15] along with $L(\theta)$:

$$\min_{\theta} L(\theta) + \lambda \sum_{\gamma \in \zeta} R_s(\gamma) \quad (5)$$

where $R_s(\cdot) = |\cdot|$ is the regular sparsification term and λ is a constant to determine the global sparsification strength.

While this approach can enforce the selective γ to zero, it may lead to unnecessary performance loss and misclassification in semantic segmentation because the scaling factors of informative channels are penalized equally to others. This may lead to unsatisfactory pruning results. Such an approach is considered a baseline approach for comparison as illustrated in Figure 10.

The context guiding vector 138 carries an adaptive penalty strength for selectively penalising channels of the neural network based on the contextual information of each channel. The CAGM 110 comprises a normalisation layer 130 and the adaptive penalty strength modifies one or more scaling factors in the normalisation layer 130.

When different channels combination or associations provide different contextual hints, this provides a prior to help distinguish useful channels from less informative channels. The

sparsification penalty is de-emphasized for the channels that give overall useful information from the input images.

As such, the Contextual Guiding Vector β is incorporated as a prior to the sparsification of channel-wise scaling factors to adaptively impose different penalty strength, namely Context-aware Guided Sparsification (CAGS):

$$\min_{\theta} L(\theta) + \lambda_1 \sum_{\gamma \in \zeta} R_s(\gamma) + \lambda_2 \sum_{\substack{\gamma \in \delta \\ \beta \in \delta'}} (1 - \beta) R_c(\gamma) \quad (6)$$

where ζ and δ are the scaling factors set over all the BN layers in ϕ , and over the selective BN layers with the CAGM respectively. δ' is the corresponding set of guiding factors. Note that $R_c(\cdot)$ is the contextual penalty term for sparsifying the scaling factor with the context guiding factor β , where $R_c(\cdot) = |\cdot|$ and is multiplied with $(1 - \beta)$ accordingly. Furthermore, λ_1 and λ_2 are the hyperparameters to determine the basic penalty strength (also referred to as adaptive penalty strength) on the regular penalty term $R_s(\cdot)$ and the contextual penalty term $R_c(\cdot)$.

When feeding different inputs during the forward pass, the channels indicating overall high channel-to-channel interdependency are considered to be more highly contextually informative. Moreover, the contextual importance that is measured by β can also provide guidance for sparsification by using term $(1 - \beta)$ in Equation 6. $(1 - \beta)$ is denoted as the channel-wise contextual sparsification guidance. The larger the integrated interdependency level, the smaller the sparsification penalty strength that will be enforced due to the smaller sparsification guidance. The CAGS tends to preserve these channels during the scaling factors sparsification and penalize the rest of the channels relatively more. During back-propagation along with the standard loss, CAGS enables the model to learn to balance the segmentation target with the aim of selecting informative channels under the guidance prior. Each guidance $(1 - \beta)$ adaptively scales the L1 penalty gradient for the channel-wise $\gamma \in \delta$, i.e. imposing less force on the contextually important channels.

CAGS prunes channels and preserve important contextual information as much as possible. After several epochs inducing sparsity with CAGS, the whole scaling factors set ζ of the cumbersome network become sparse, enabling the determination of less informative channels

with the smallest scaling factors γ . Finally, a compact model is obtained after pruning and finetuning. Ablation studies disclosed below illustrate the effectiveness of the CAGS framework.

Figure 4 illustrates a method 400 for image pattern recognition. Method 400 is performed by a neural network, such as the neural network 100 or 200 of Figure 1 or Figure 2 respectively. A step 410, the neural network receives an image at an input layer, such as the input layer 124. At step 420, the neural network formulates a plurality of feature maps for the image. Each feature map corresponds to a channel in the neural network. At step 430, the neural network is pruned or sparcified based on the method of pruning described with reference to Figure 3. In some embodiments, step 430 are performed before step 410. At step 440, the neural network outputs an image pattern corresponding to the image received at 410 from an output layer of the neural network. The image pattern may include image segmentation information, i.e. information regarding a class to which each pixel of the image received at 410 belongs.

Figure 5 is a schematic diagram illustrating an unpruned neural network 510 and a pruned neural network 520. The pruned network 520 may be obtained by applying the method of pruning of Figure 3 on network 510. As demonstrated in Figure 5, the unpruned network 510 comprises a significant number of nodes or neurons 512 with a significant number of connections 514. In comparison, the pruned network 520 comprises a significantly lower number of neurons 522 and connections 524.

Figure 6 illustrates an input image 610 and an output image 620 obtained by semantic segmentation operations by a neural network model according to the present disclosure. Segment 622 is identified as a boat in the output image 620. Other segments such as sky, trees, water are also identified in the input image 610.

Figure 7 illustrates a theoretical contextual relationship for the components or segments of the input image 610 of Figure 6. Information regarding contextual relationships between the various segments in an image enables the CAGM to generate the context guiding vector. In the example of Figure 7, there is a strong relationship between the channel encoding the features relating to a boat and water (because boats are typically observed on water). Channels encoding the relationship between the boat and water features may be strengthened during the CAGS process. Less relevant channels such as channels encoding the relationship between boat

features and grass or sky features may be weighted less or discarded during CAGS. Based on the contextual relationship, a neural network trained to segment boats, water, sky and trees can be pruned by CAGS while retaining the accuracy of the pruned neural network with respect to the unpruned neural network.

Figure 8 is a representation of feature maps and channels obtained from an input image. An input image can be processed by a CNN to generate the feature map 800 comprising 64 channels. Activations on different channels during segmentation may provide differently significant information for each relevant segment label. For example, channels 1, 62, 63 may be more relevant for the segment water 810. Similarly, channels 1 and 2 may be more relevant for segment tree 820, channels 2 and 3 may be more relevant for segment sky and channels 62 and 5 may be more relevant for segment grass 830.

Figure 9 illustrates a flowchart 900 of a part of a method of pruning neural networks. At stage 910 channel-wise scaling factors γ of each channel in a neural network are determined. The scaling factors γ can be an importance indicator for the corresponding channels. Based on the channel wise scaling factors a context guiding vector β is calculated. At stage 920, the relative weight of each channel is determined based on the context guiding vector β and the scaling factors γ . Between stages 920 and 930, channels that are highly contextually informative are retained. Channels that have low contextual information are dropped or de-emphasized. The channels with scaling factors γ close to 0 are considered less contextually informative. Between stages 930 and 940, fine-tuning operations on the pruned neural network are performed. There may be a drop in the performance of the neural network after pruning between stages 920 and 930. The pruned models of 930 are fine-tuned for several epochs using the same setting as the training stage of the neural network, but with a smaller learning rate to obtain the fine-tuned model at stage 940.

Experiments

Performance of the pruning methods and neural network according to some embodiments was empirically evaluated on various networks and two benchmark datasets CamVid and Cityscapes. The implementation of the pruning framework included three stages as follows: normal training, sparsity induction, and finally pruning including finetuning.

CamVid is a road scene dataset, which contains 367 training and 233 testing images of the size 360X480. The dataset includes ground truth labels associating each pixel with one semantic class. The semantic classes included 11 semantic classes are of common interest, including pavement, pedestrian, tree, building, sky, etc.

Cityscapes is a large-scale urban driving scene dataset. Cityscapes includes 2,975 well-annotated and high-resolution images in the training set, 500 images in the validation set, and 1,525 images in the test set. The dataset includes 19 common classes for semantic segmentation. The classes included humans, vehicles, constructions, objects, nature, sky, etc.

Normal Training

For CamVid dataset, the initial learning rate was set to 0.01, and the cosine annealing decay policy of $\frac{1}{2} \cdot \text{initial_lr} \cdot \left(1 + \cos\left(\frac{\text{iter}}{\text{total_iter}}\pi\right)\right)$ was applied for 450 epochs training in mini-batch size 8. A stochastic gradient descent (SGD) optimizer with momentum coefficient 0.9 and weight decay coefficient 0.0005 was employed.

For the Cityscapes dataset, a SegNet based network was trained with 512x1024 resized input images for 450 epochs in batch size 8, and the Adam optimizer is used. For a PSPNet based network, the input images were randomly cropped into the size 713x713. The PSPNet based model was trained using the Inplace-ABN and the SGD optimizer with momentum for 200 epochs. An ICNet based network was trained using the same setting used for the PSPNet based network but with inputs in full sizes. A poly decay strategy with power 0.9 on the learning rate, which is multiplied by $\left(1 - \frac{\text{iter}}{\text{total_iter}}\right)^{0.9}$ was employed in each iteration of the training of the ICNet based network. The initial learning rates were set to 0.001 for SegNet, and 0.01 for PSPNet and ICNet. In addition, multiple data augmentations were adopted, such as random scaling, random rotation, random translation, and random flipping. Due to the performance loss when using the pre-trained PSPNet50 backbone in the Caffe framework for ICNet, an ICNet based network was reimplemented and trained from scratch for the experiments.

Sparsity Induction

Before pruning, the CAGM was applied on the optimized models after the normal training stage. Subsequently, sparsity was induced on the scaling factors γ with CAGS in a few epochs, to distinguish channels based on the richness of their respective contextual information for the given training data.

After applying CAGS to the networks or models, the magnitude of scaling factors indicated the channel-wise saliency considering the contextual information. The scaling factors are subsequently used for pruning. The hyperparameters λ_1 and λ_2 were set to 0.0001 and 0.001. An ablation study on such settings is presented in Figure 17, where the sparsity level with λ in different values are shown, and $\lambda_1 = 0$; $\lambda_2 = 0$ represents the original model. λ_2 is always 10 times as large as λ_1 , to balance the effect of multiplying the guidance term $(1-\beta) \in [0,1]$. Since the CAGMs are applied to provide pruning guidance only, they can be harmlessly removed after inducing sparsity.

Pruning and Fine-tuning

Each scaling factor γ in BN regulates the channel outputs into various magnitudes. The lesser the scaling factor, the lesser contribution its channel makes to the final prediction. Hence, the channels with the smallest γ are discarded in a global and greedy manner. Instead of setting layerwise prune ratios, one global prune ratio is assigned for reference. Hence, the pruned architectures of the disclosed method are determined automatically based on their channel-wise importance global ranking. The pruned architectures are shown in Figures 13 to 15.

The removal of a specific channel during the pruning step is equivalent to removing a convolution kernel in a previous layer of the networks being pruned. To maintain the network architecture, it is also necessary to remove the following channel in all the incoming convolution kernels. To avoid the case where all pruning candidate channels are within the same layer, 10% of channels (or some other proportion) in each layer may be preserved. For pruning the residual block, the last convolution layer and downsampling layers are preserved to match the feature volume for summation. For a SegNet based network, the max-pooling indices are shared in-between layers. Thus, for channels that are determined to be removed in either the Encoder or the Decoder part, their corresponding channels for indices sharing are also be pruned.

Since there is an inevitable performance drop after pruning, the pruned models are fine-tuned for several epochs using the same setting as the training stage, but with a smaller learning rate. Finally, the compact models are optimized and evaluated based on the given metrics.

Baseline

Conventional pruning methods are not specifically tailored for semantic segmentation networks. The experiments evaluated the impact of popular baseline pruning methods (originally evaluated for image-level classification) on widely-used semantic segmentation models to show the advantage of the disclosed methods and models. The baselines include BN-Scale, NS (Network Slimming), FPGM ('Filter pruning via geometric median for deep convolutional neural networks acceleration' by Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang published in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4340–4349, 2019) and the conditional computing based method CCGN ('Batch-shaping for learning conditional channel gated networks' by Babak Ehteshami Bejnordi, Tijmen Blankevoort, and Max Welling published in International Conference on Learning Representations, 2020).

FPGM is a pruning method for image classification, which prunes filters based on their Euclidean distance with other filters layer-wise. The experiments follow the pruning criterion formulation and the optimal predefined layer-wise prune ratio in the literature.

BN-Scale and NS methods both use the scaling factors in BN as the pruning indicator. The former serves as a naïve baseline, and the latter is the widely-used pruning method. In BN-Scale, the original model is directly pruned based on the scaling factors magnitude after normal training. In NS, the regular sparsity is induced on all the scaling factors before pruning.

The disclosed pruning method and the first two baselines belong to the automatic pruning approach, while FPGM performs pruning given the manually specified pruned ratio for each layer and results in a predefined pruned architecture. The overall pruning results comparison is illustrated in Table 1, and show the pruning results when FPGM is implemented in an automatic pruning manner. Although CCGN does not prune the original network, it was adopted as a baseline as it is a network acceleration method that provides comprehensive results on semantic segmentation. The acceleration comparison is shown in Table 3.

Results

Quantitative and qualitative comparisons are presented in Table 1 and Figure 10.

Dataset		Methods	mIoU(%)	#Params(M)(%↓)	#FLOPs(G)(%↓)
CamVid	SegNet	Unpruned	55.60	29.45	106.73
		FPGM[11]	52.54	15.63(46.92%↓)	32.95(69.13%↓)
		NS-20%[15]	54.78	12.25(58.40%↓)	47.81(55.20%↓)
		BN-Scale-20%	55.73	12.50(57.39%↓)	44.54(58.27%↓)
		Ours-20%	57.12	11.47(61.05%↓)	53.14(50.21%↓)
		Ours-30%	56.37	6.03(79.52%↓)	30.01(71.88%↓)
Cityscapes	PSPNet101	Unpruned	78.40	70.44	557.04
		FPGM[11]	74.84	36.09(48.76%↓)	280.68(49.61%↓)
		NS-60%[15]	75.70	48.47(31.19%↓)	368.03(33.93%↓)
		BN-Scale-60%	74.88	48.81(30.71%↓)	370.49(33.49%↓)
		Ours-60%	77.82	47.84(32.08%↓)	363.21(34.80%↓)
		Ours-70%	75.27	39.74(43.58%↓)	296.25(46.82%↓)
	PSPNet50	Unpruned	76.99	51.45	403.00
		FPGM[11]	74.59	27.06(47.40%↓)	207.31(48.56%↓)
		NS-50%[15]	73.57	23.61(54.11%↓)	199.78(50.43%↓)
		BN-Scale-50%	73.85	23.59(54.15%↓)	199.43(50.51%↓)
		Ours-60%	75.59	27.31(46.92%↓)	233.67(42.02%↓)
		Ours-70%	73.94	23.78(53.78%↓)	203.19(49.58%↓)
	ICNet	Unpruned [†]	64.59	12.21	40.13
		FPGM[11]	62.00	6.45(47.18%↓)	22.96(42.79%↓)
		NS-60%[15]	60.02	6.90(43.49%↓)	22.75(43.31%↓)
		BN-Scale-60%	59.68	6.96(43.00%↓)	22.82(43.13%↓)
		Ours-60%	62.38	5.56(54.46%↓)	21.16(47.27%↓)
	SegNet	Unpruned	56.10	29.45	326.59
		FPGM[11]	51.60	15.63(46.92%↓)	100.51(69.22%↓)
		NS-20%[15]	56.85	11.85(59.76%↓)	188.16(42.39%↓)
		BN-Scale-20%	59.95	11.92(59.52%↓)	150.47(53.93%↓)
		Ours-20%	61.16	10.76(63.46%↓)	178.23(45.43%↓)

[†] Train from scratch

Table 1. Quantitative pruning results on CamVid and Cityscapes test set.

In Table 1 above, each row corresponds to a model or network or a model or network pruned according to a specific technique. In BN-Scale, the original model is directly pruned based on the scaling factors magnitude after normal training. The percentage values in the method column of Table 1 refers to the pruning ratio value applied to the various models during the pruning process.

The reduction in the number of model parameters and FLOPs may not be consistent across the distinct models or networks due to the pruned channels selections on different layers. The pruned models with a similar number of parameters are compared. Additionally, the actual runtime speedups after pruning with the present methods and the per class prediction performance comparison is illustrated in Table 4 and Table 5.

Table 1 includes results of experiments on pruned models using the mean Intersection-over-Union (mIoU), the number of parameters (#Params), and the floating-point operations (#FLOPs). FLOPs of SegNet are reported based on the input size 512x1024 and 360x480 for Cityscapes and CamVid, respectively. While for PSPNet and ICNet, the FLOPs are reported for input size 713x713 and 1024x2048. All the test results on Cityscapes are submitted and evaluated by a benchmark server. Ours-x% denotes the pruning using the disclosed pruning method with a global prune ratio x%. The same notation applies for the baselines BN-Scale and NS. Note that in the same prune ratio x%, the reduction in #Params and #FLOPs vary for different methods, due to the selection differences on channels to be pruned. Table 1 shows the pruning results for specific pruning ratios x% that have similar #Params reduction and the closest performance with the unpruned model.

The results of Table 1 demonstrate that the disclosed pruning methods can effectively reduce the number of parameters and #FLOPs, compared to all the baselines. Specifically:

1. In terms of the reduction in #Params, the disclosed method achieves the best pruning performance in comparison to rest of the methods tested in the experiment. The method allow determination of efficiently pruned network architectures. For instance, for Cityscapes and PSPNet101, Ours-60% achieves 77.82 mIoU while NS-60% and BN-Scale-60% only achieved 75.70 and 74.88 mIoU with a larger number of parameters. For SegNet, Ours-20% is able to outperform the original model with 61% lesser parameters and 45% fewer FLOPs. The pruned models obtained using the disclosed pruning methods comprise a similar number of parameters to the pruned models used as a baseline for comparison.
2. The images 1015, 1025, 1035 in Figure 10 demonstrate that the pruned models of obtained using the disclosed method can preserve most of the prediction precision on small objects in comparison to the original model. In comparison, the image

segmentation results obtained using the baselines methods lead to information loss and misclassification by varying degrees.

Ablation Studies

Ablation studies were performed in relation to the various elements of the disclosed methods of network pruning.

Pruning Ratio

The use of a large pruning ratio may result in high model capacity loss leading to the inability to recover the segmentation performance. On the other hand, small pruning ratios will not lead to effective compression for the given set of compression requirements. Hence, the right balance between the model size and performance may be struck based on pruning ratios.

Figure 17 illustrates in graphs the mIoU values and number of parameters (y-axis) for various pruning ratios (x-axis) for the various networks when pruned according to the disclosed methods. Graphs 1710, 1720, 1730 and 1740 correspond to graphs of ablation studies on PSPNet101, PSPNet50, ICNet and SegNet based neural networks respectively. The graphs of Figure 17 demonstrate that it is possible to maintain original accuracy by keeping a maximal pruning ratio within certain intervals (e.g., between 0.5 and 0.7 for PSPNet101 in graph 1710).

Hyperparameters λ_1 and λ_2

The hyperparameters λ_1 and λ_2 are used to adjust the strength on the regular sparsification term and the contextual sparsification term, and in CAGS, pair $\lambda_1 = 0.0001$, $\lambda_2 = 0.001$ was considered preferable.

Figure 18 illustrates graphs of ablation studies on different values of λ_1 and λ_2 pair for the various segmentation networks. The histograms in different shades indicate the frequency of scaling factors γ in the models under different λ pairs after the application of CAGS. Graphs 1810, 1820, 1830 and 1840 correspond to graphs of ablation studies on PSPNet101, PSPNet50, ICNet and SegNet based neural networks respectively.

As shown in Figure 18, different λ pair results in models with different sparsity levels on γ . When a larger λ pair forces more γ towards zero, the model's performance will be negatively

impacted as well. For instance, in the PSPNet101 experiments on Cityscapes validation set, while smaller λ pair, i.e., $\lambda_1 = 0.0001$, $\lambda_2 = 0.001$ (series 1812), results in mIoU of 77.57 and induces suitable sparsity from the unpruned model (series 1814), larger λ pair, i.e., $\lambda_1 = 0.001$, $\lambda_2 = 0.01$ (series 1816), leads to a significantly sparser model with 63.63 mIoU.

Position of CAGM

The position of the CAGM within the various networks another factor that was subjected to an ablation study. The CAGM was placed adjacent to the pooling layers, where the feature maps with the potential spatial context information are leveraged. To evaluate the effectiveness of this positioning an ablation experiment was conducted that involved applying CAGM on all layers of the various networks.

Models	CAP	CAGM on All	Prune Ratio	mIoU(%)
PSPNet101	✓	✓	0.6	78.23
			0.6	77.99
PSPNet50	✓	✓	0.7	74.31
			0.7	73.21
SegNet	✓	✓	0.2	60.98
			0.2	57.85
ICNet	✓	✓	0.6	63.25
			0.6	61.22

Table 2. Ablation study on the position of CAGM. *CAP* is our proposed framework, and *CAGM on All* is when we apply the CAGM on all layers. The mIoU in different architectures are reported on Cityscapes **validation set**.

Column 'CAGM on All' in Table 2 relates to the experiments wherein CAGM was applied on all layers of the respective networks. Column 'CAP' in Table 2 relates to experiments wherein the CAGM was placed adjacent to the pooling layers. Results in Table 2 demonstrate that positioning adjacent to the pooling layer in the disclosed framework is sufficient, since it consistently leads to better pruning performance compared to CAGM on All, especially on lightweight models. The lightweight models have a relatively larger downsampling rate within fewer layers. Feature maps after each downsampling operation capture richer spatial information. Using such information as guidance benefits the evaluation of the contextual informative channels. And when other feature maps that contain less useful information are

also utilized by CAGM, the advantage of pruning may be less noticeable and more computation in run-time memory may be required.

Table 3 below compares the performance of the disclosed pruning methods (listed 'Ours-60%') with the pruning methods of CCGN.

Methods	mIoU(%)	#Params(M)(%↓)	#FLOPs(G)(%↓)
Unpruned	76.57	51.45	403.0
CCGN*[2]	71.90	70.44(0%↓)	23.70%↓
CCGN-1 [2]	74.40	70.44(0%↓)	23.50%↓
CCGN-2 [2]	74.70	70.44(0%↓)	5.00%↓
Ours-60%	75.65	27.31(46.92%↓)	233.67(42.02%↓)

* Without pretrain

Table 3. Comparison with CCGN [2] on Cityscapes validation set.

Table 4 below tabulates the runtime acceleration results of models obtained by applying the disclosed pruning methods with unpruned models. Table 4 demonstrates that the pruning methods generate models that are significantly faster without a significant sacrifice to the mIoU% values. For example, a SegNet (30% pruning ratio) is 93.41% faster than an unpruned SegNet model while retaining a similar mIoU% value.

Dataset	Methods	mIoU(%)	Time(ms)	Speed(fps)(%↑)
CamVid	SegNet(Unpruned)	55.60	18	54.79
	SegNet(Ours-20%)	57.12	12	81.34(48.46%↑)
	SegNet(Ours-30%)	56.37	9	105.97(93.41%↑)
Cityscapes	PSPNet101(Unpruned)	78.40	885	1.13
	PSPNet101(Ours-60%)	77.82	658	1.52(34.51%↑)
	PSPNet101(Ours-70%)	75.27	595	1.68(48.67%↑)
	PSPNet50(Unpruned)	76.99	735	1.36
	PSPNet50(Ours-70%)	73.94	485	2.06(51.47%↑)
	ICNet(Unpruned)	64.59	14	69.91
	ICNet(Ours-60%)	62.38	13	76.27(9.10%↑)
	SegNet(Unpruned)	56.10	83	12.02
	SegNet(Ours-20%)	61.16	59	17.04(41.76%↑)

Table 4. Runtime acceleration per image inference in Cityscapes (1024×2048) and CamVid (360×480) **test set**. All inference speeds are measured by a single Tesla V100 GPU.

Table 5 below tabulates per class segmentation results for the Cityscapes dataset when segmented by the unpruned and pruned models according to the disclosed pruning methods and the comparative pruning methods.

The results of Table 5 demonstrate the advantage of the disclosed method of pruning that is able to preserve the closest accuracy from the original model in general and reduce the redundancy efficiently. Note that for pruning lightweight models, some baselines like NS may suffer from unrecoverable performance loss on uncommon classes (for example for classes rider, truck), while the models pruned according to the disclosed methods can still maintain the discriminative ability on different classes.

Method	Average	road	swalk	build.	wall	fence	pole	light	sign	veg.	terrain	sky	person	rider	car	truck	bus	train	minike.	bike	
PSPNet101	Unpruned	78.40	98.60	86.20	92.90	50.80	58.80	64.00	75.60	79.00	95.40	72.30	95.40	86.50	71.30	95.90	68.20	79.50	73.80	69.50	77.20
	PPGM	74.84	98.34	84.56	92.16	53.15	56.94	60.44	69.28	74.58	93.04	70.38	94.74	83.74	65.73	95.35	58.97	75.12	59.89	62.80	72.91
	NS-60%	75.70	98.21	83.09	91.99	51.85	56.00	60.45	69.23	73.82	92.84	69.04	95.02	83.51	64.94	95.05	63.55	80.45	74.21	62.49	72.62
	BN-Scale-60%	74.88	98.15	83.06	91.83	49.09	54.50	60.62	68.91	73.56	92.93	71.20	94.73	83.37	64.79	95.09	65.04	81.07	61.96	60.75	72.08
	Ours-60%	77.82	98.40	84.78	92.05	56.74	57.79	63.58	72.05	76.57	93.22	71.60	95.04	84.91	67.91	95.60	68.81	83.82	75.66	65.23	74.18
	Ours-70%	75.27	98.24	83.65	92.08	55.53	53.93	60.93	69.95	73.86	92.94	71.25	94.82	83.58	65.64	95.16	64.38	77.62	64.33	60.21	72.03
PSPNet50	Unpruned	76.99	98.41	84.52	92.04	51.01	56.53	64.13	73.19	77.18	93.27	70.45	95.21	83.43	69.80	93.72	69.55	79.52	65.86	66.14	74.33
	PPGM	74.59	98.17	83.27	92.00	54.96	54.08	59.20	70.06	72.80	92.96	70.91	94.76	83.82	65.61	95.22	62.71	75.29	56.24	62.43	72.76
	NS-50%	73.57	98.15	83.20	91.46	50.19	54.16	59.48	68.91	73.09	92.65	69.77	94.26	82.31	63.63	94.75	62.54	71.84	57.95	58.05	71.40
	BN-Scale-50%	73.85	98.19	83.50	91.54	51.64	52.85	58.84	69.02	72.96	92.73	70.31	94.33	82.15	63.27	94.86	63.20	74.54	61.54	57.14	70.64
	Ours-60%	75.59	98.38	84.71	92.58	56.14	56.64	63.14	72.50	76.23	93.23	71.19	94.90	84.46	66.09	95.38	60.05	73.80	61.27	62.35	73.27
	Ours-70%	73.94	98.21	83.55	91.95	50.88	54.67	61.34	70.90	74.81	92.89	70.86	94.76	83.73	64.79	94.95	63.06	71.55	51.13	58.90	71.87
ICNet	Unpruned	64.59	97.71	79.95	88.87	37.52	40.78	43.87	51.12	57.80	90.71	67.13	93.83	72.65	52.47	92.08	50.65	60.16	48.17	43.33	58.45
	PPGM	62.00	97.39	77.72	87.63	35.95	38.51	39.66	44.13	53.36	90.04	64.84	93.05	70.02	47.32	91.22	49.45	56.48	47.74	38.89	54.62
	NS-60%	60.02	97.20	76.73	87.08	40.95	36.56	37.27	40.28	49.25	89.44	63.56	92.68	67.79	46.05	90.18	39.96	53.68	42.49	35.64	53.59
	BN-Scale-60%	59.68	97.21	76.83	87.10	35.56	34.46	37.66	40.64	49.13	89.49	64.43	92.70	67.71	45.69	90.21	43.98	50.07	39.56	36.55	54.73
	Ours-60%	62.48	97.36	78.00	87.88	41.32	37.93	40.83	43.93	52.29	89.98	65.58	92.75	69.91	48.46	90.79	46.17	59.01	47.19	40.27	55.51
	Unpruned	56.09	95.65	70.10	82.81	29.87	31.88	38.06	43.05	44.58	87.32	62.30	91.68	67.28	50.75	87.89	21.70	29.03	34.73	40.47	56.63
SelfNet	PPGM	51.60	96.20	71.11	84.02	26.19	26.67	33.26	32.17	43.78	88.24	61.92	91.10	57.16	32.50	88.28	21.68	28.31	29.41	21.25	47.17
	NS-20%	56.85	96.32	77.51	88.43	35.00	37.18	40.41	53.27	60.57	91.18	67.07	93.91	71.57	46.32	91.42	27.67	0.00	0.00	35.52	57.89
	BN-Scale-20%	59.95	97.12	77.03	88.17	33.31	36.35	48.80	50.34	59.19	90.80	66.30	93.68	70.51	44.78	91.50	29.46	41.20	29.36	33.68	57.66
	Ours-20%	61.16	97.19	77.32	88.37	32.68	37.15	49.88	53.64	61.89	90.95	66.23	93.74	71.69	45.26	91.31	30.39	42.66	37.59	34.95	59.21

Table 5. Per-class results after pruning on Cityscapes test set.

Table 6 below tabulates a comparison of the performance of models pruned according to the disclosed methods with models pruned according to FPGM variants in an automatic pruning manner.

	Methods	mIoU(%)	#Params(M)(%↓)	#FLOPs(G)(%↓)
PSPNet101	Unpruned	77.48	70.44	557.04
	FPGM-A-20%	70.94	53.10(24.62%↓)	397.22(28.69%↓)
	FPGM-A-30%	67.20	46.81(33.54%↓)	346.25(37.84%↓)
	Ours-60%	78.23	47.84(32.08%↓)	363.21(34.80%↓)
	Ours-70%	75.40	39.74(43.58%↓)	296.25(46.82%↓)
PSPNet50	Unpruned	76.57	51.45	403.0
	FPGM-A-30%	60.14	35.17(31.63%↓)	273.53(32.13%↓)
	FPGM-A-40%	61.36	23.56(54.21%↓)	185.69(53.92%↓)
	Ours-60%	75.65	27.31(46.92%↓)	233.67(42.02%↓)
	Ours-70%	74.31	23.78(53.78%↓)	203.19(49.58%↓)
	Ours-80%	70.83	21.16(58.87%↓)	179.79(55.39%↓)
ICNet	Unpruned	64.59	12.21	40.13
	FPGM-A-20%	49.20	10.47(14.30%↓)	27.03(32.64%↓)
	FPGM-A-30%	43.01	8.86(46.57%↓)	24.62(38.65%↓)
	Ours-60%	63.26	5.56(54.46%↓)	21.16(47.27%↓)
SegNet	Unpruned	56.10	29.45	326.59
	FPGM-A-10%	44.91	27.41(6.93%↓)	178.09(45.46%↓)
	FPGM-A-40%	37.17	10.85(63.18%↓)	43.74(86.61%↓)
	Ours-20%	60.98	10.76(63.46%↓)	178.23(45.43%↓)

Table 6. Quantitative pruning results on Cityscapes validation set.

Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration (denoted as FPGM) indicates filter importance via its Euclidean distances to other filters in the same layer. As FPGM was originally implemented in the image classification task only, it was re-implemented for semantic segmentation for the purpose of evaluating the disclosed pruning methods. Table 1 demonstrates that the performance of FPGM wherein the filter pruning ratio in each layer was predefined. The pruned architectures of FPGM in Table 1 are shown in Figures 13 to 16. Table 6 tabulates the results of models pruned using FPGM, where only a global pruning ratio was set, and prune filters were implemented in a global and greedy manner. This method is denoted as FPGM-A. FPGM-A-x% stands for the global threshold ratio. The 10% filters were reserved to prevent pruning out entire layers.

Batch-Shaping for Learning Conditional Channel Gated Networks (denoted as CCGN in Table 3) is a conditional computing method, which estimates channel saliency by introducing a gated module similar to Dynamic Channel Pruning, but provides a better trade-off between simple and complex example inferences. Although CCGN is not strictly a network pruning method, it provides comprehensive network acceleration results on the large-scale semantic segmentation benchmarks. CCGN(without pre-train) stands for the model that undertakes training without ImageNet-pre-trained model, while CCGN-1(with pre-train) and CCGN-2(with pre-train) are with the pre-trained models and reduce FLOPs to different percentages to balance the performance.

Table 6 demonstrates that the disclosed methods for the pruning method outperform the above mentioned state-of-the-art pruning methods. It can also be observed that when FPGM is implemented in an automatic pruning manner (i.e., FPGM-A-x%), the performance become worse (compared to FPGM in Table 1). From this observation, it is evident that the pruning criteria of the disclosed method serve as a better global indicator to identify the importance of channels, while some pruning criteria in the image classification task may not be effective for semantic segmentation. Figures 13 to 16 visualize the pruned architectures using the disclosed framework and baseline architectures, i.e., CAP, and the original FPGM.

Practical Applications

The proposed methods of pruning of networks or models enable compression of various deep convolution neural networks, especially those that target computer vision tasks that are sensitive to contextual information, e.g., semantic segmentation and crowd counting.

The potential application areas include pruning or compressions of models for computer vision operations for autonomous driving, robot navigation, smart surveillance, public security, etc., which have strict requirements for the models to be computationally inexpensive, low-complexity, or power efficient. The disclosed methods enable the compression of complex deep neural network models to make them suitable for deployment on edge devices that have limited computational or power resources, e.g., embedded systems like wearable devices or mobile devices. By deploying the models on the edge, high communication bandwidth due to data transfer between the cloud and the edge device can be avoided, while still achieving satisfactory performance.

The proposed Context-aware Pruning framework utilizes channels association to exploit parameters redundancy in terms of contextual information for accelerating and compressing semantic segmentation models without sacrificing accuracy. The disclosed methods effectively preserve contextual informative channels after pruning. Experiments on two challenging datasets demonstrated the advantages of the disclosed methods over the baseline pruning methods for both large and lightweight state-of-the-art architectures. The disclosed framework can also be used to complement other pruning schemes (e.g., iterative pruning) or compression techniques (e.g., quantization) to further improve the performance of the pruned models.

Figure 10 illustrates a series of images on which image segmentation operations are applied using pruned networks as described herein and baseline methods. Images 1010, 1020 and 1030 are the images input to the respective neural networks. Images 1011, 1021 and 1031 are ground truth images representing the segmentation ground truth for each input image. The segmentation ground truth comprises a label or annotation for each distinct segment in the input image.

Images 1012, 1022, 1032 are obtained by segmentation operations using a neural network pruned using the network slimming techniques described in 'Learning efficient convolutional networks through network slimming' in Proceedings of the IEEE International Conference on Computer Vision, pages 2736–2744, 2017 by Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang.

Images 1013, 1023 and 1033 are obtained by segmentation operations using a neural network pruned BN-Scale, wherein the original model is directly pruned based on the scaling factors magnitude after normal training.

Images 1014, 1024 and 1034 are obtained by segmentation operations using an unpruned neural network. Images 1015, 1025 and 1035 are obtained by segmentation operations using a neural network pruned according to the methods of this disclosure. As is observable from the white rectangles in images 1035 and 1034, the segmentation output (1035) of a neural network pruned according to the disclosed method closely matches the segmentation output (1034) of an unpruned model. In comparison, the segmentation output (1032, 1033) of the baseline methods, substantially deviates from the segmentation output (1034) of the unpruned model.

Figure 11 illustrates images of a crowd counting operation that may be performed using neural networks pruned using the disclosed methods of pruning. As surveillance cameras may be deployed across large parts of an urban area, neural network models pruned according to the disclosed methods may be more suitable for deployment on the edge in concert with the surveillance cameras. When image data is processed on the edge, the bandwidth consumption in the transmission of raw image data is substantially reduced.

Images 1110, 1120 and 1130 are raw images captured by a surveillance camera. Images 1112, 1122 and 1132 are output images generated by image segmentation operations by neural network pruned according to the disclosed methods. An identified segment (such as segment 1140 in image 1132) in the output images corresponds to a person (or a person's head) in the respective input images. Based on the number of identified segments or the density of identified segments, an estimate of the size of the crowd in the respective input images may be determined.

Figure 12 illustrates images of a crowd counting operation that may be performed using neural networks pruned using the disclosed methods of pruning and corresponding unpruned neural networks. Images 1210 and 1212 are input images that were captured by a surveillance camera. Images 1220 and 1222 are image segmentation output images generated by an unpruned neural network. Images 1230 and 1232 are image segmentation output images generated by a neural network pruned according to the disclosed pruning methods. The crowd count estimate based on images 1230 (count 309.1) and 1232 (count 41) are reasonably close to the estimates obtained using the unpruned model (315.6, 46.2 respectively) and the ground truth (298, 44 respectively) as well. Therefore, Figure 11 demonstrates an example where models pruned using the disclosed pruning methods perform relatively accurately despite the removal of channels and simplification of the network due to the pruning process.

Figure 13 illustrates graph 1300 of a layer-wise comparison of a part of a neural network structure pruned according to the disclosed methods of pruning. The example in Figure 13 is based on a SegNet neural network. The x-axis of graph 1300 corresponds to a layer of a neural network and the y axis corresponds to a number of filters in a respective layer. Series 1310 correspond to a number of filters in a model pruned according to the disclosed pruning methods at a 20% pruning ratio. Series 1320 corresponds to a number of filters in a model pruned

according to a baseline FPGM pruning method. Series 1330 corresponds to an unpruned model. As is observable from graph 1300, the model pruned according to the disclosed method comprises a substantially lower number of filters in comparison to the baseline for layers such as down5.conv1, up5.conv1, up5.conv2.

Figure 14 illustrates graph 1400 of a layer-wise comparison of a part of a neural network structure pruned according to the disclosed methods of pruning. The example in Figure 14 is based on an ICNet neural network. The x-axis of graph 1400 corresponds to a layer of a neural network and the y axis corresponds to a number of filters in a respective layer. Series 1410 correspond to a number of filters in a model pruned according to the disclosed pruning methods at a 60% pruning ratio. Series 1420 corresponds to a number of filters in a model pruned according to a baseline FPGM pruning method. Series 1430 corresponds to an unpruned model.

Figure 15 illustrates graph 1500 of a layer-wise comparison of a part of a neural network structure pruned according to the disclosed methods of pruning. The example in Figure 15 is based on a PSPNet50 neural network. The x-axis of graph 1500 corresponds to a layer of a neural network and the y axis corresponds to a number of filters in a respective layer. Series 1510 correspond to a number of filters in a model pruned according to the disclosed pruning methods at a 60% pruning ratio. Series 1520 corresponds to a number of filters in a model pruned according to a baseline FPGM pruning method. Series 1530 corresponds to an unpruned model.

Figure 16 illustrates graph 1600 of a layer-wise comparison of a part of a neural network structure pruned according to the disclosed methods of pruning. The example in Figure 16 is based on a PSPNet101 neural network. The x-axis of graph 1600 corresponds to a layer of a neural network and the y axis corresponds to a number of filters in a respective layer. Series 1610 correspond to a number of filters in a model pruned according to the disclosed pruning methods at a 60% pruning ratio. Series 1620 corresponds to a number of filters in a model pruned according to a baseline FPGM pruning method. Series 1630 corresponds to an unpruned model.

It will be appreciated that many further modifications and permutations of various aspects of the described embodiments are possible. Accordingly, the described aspects are intended to

embrace all such alterations, modifications, and variations that fall within the spirit and scope of the appended claims.

Throughout this specification and the claims which follow, unless the context requires otherwise, the word “comprise”, and variations such as “comprises” and “comprising”, will be understood to imply the inclusion of a stated integer or step or group of integers or steps but not the exclusion of any other integer or step or group of integers or steps.

The reference in this specification to any prior publication (or information derived from it), or to any matter which is known, is not, and should not be taken as an acknowledgment or admission or any form of suggestion that that prior publication (or information derived from it) or known matter forms part of the common general knowledge in the field of endeavour to which this specification relates.

Claims

1. A system for image pattern recognition, comprising memory storing a neural network model, the neural network model comprising:
 - an input layer for receiving the image;
 - an output layer for outputting the image pattern; and
 - a context-aware guiding module (CAGM) between the input layer and output layer, the CAGM:
 - receiving feature maps supplied by a layer of the neural network;
 - forming a context guiding vector from a layer-wise integrated channel interdependency of the feature maps; and
 - directing contextually informative channel selection based on the context guiding vector,wherein the output layer outputs the image pattern based on features of channels selected using the context guiding vector.
2. The system of claim 1, wherein the CAGM forms the context guiding vector by computing a channel affinity matrix and forming the context guiding vector from the channel affinity matrix.
3. The system of claim 2, wherein the CAGM forms the context guiding vector by squeezing the channel affinity matrix based on dimensions of a spatial size of the layer supplying the feature maps.
4. The system of any one of claims 1 to 3, wherein the context guiding vector carries an adaptive penalty strength for selectively penalising channels of the neural network.
5. The system of claim 4, wherein the CAGM comprises a normalisation layer and the adaptive penalty strength modifies one or more scaling factors in the normalisation layer.
6. The system of any preceding claim, wherein the CAGM obtains the context guiding vector β_j^i by:

$$\beta_j^i = \frac{\sum_k a_{j,k}^i - \text{Min}(\sum_k a_{:,k}^i)}{\text{Max}(\sum_k a_{:,k}^i) - \text{Min}(\sum_k a_{:,k}^i)} \in [0,1]$$

wherein $\text{Max}()$ and $\text{Min}()$ compute a maximum and minimum value along a channel dimension of channel k , β_j^i is a guiding factor for channel j in layer i , and $a_{j,k}^i$ is an affinity level between channel j and channel k in layer i , and $a_{:,k}^i$ is an affinity level between channel k and remaining channels in layer i .

7. The system of any preceding claim, wherein the CAGM directs contextually informative channel selection based on the context guiding vector by applying a modifying scaling factor to each channel based on the context guiding vector, to perform context-aware guided sparsification (CAGS).
8. The system of claim 7, wherein the CAGM performs CAGS based on a channel-to-channel interdependency as determined from the context guiding vector.
9. The system of claim 7 or 8, further comprising a pruning module for pruning one or more channels of the neural network based on the modified scaling factor for the respective channel.
10. A neural network model for image pattern recognition, comprising:
 - an input layer for receiving the image;
 - an output layer for outputting the image pattern; and
 - a context-aware guiding module (CAGM) between the input layer and output layer, the CAGM:
 - receiving feature maps supplied by a layer of the neural network;
 - forming a context guiding vector from a layer-wise integrated channel interdependency of the feature maps; and
 - directing contextually informative channel selection based on the context guiding vector,

wherein the output layer outputs the image pattern based on features of channels selected using the context guiding vector.

11. The neural network model of claim 10, wherein the CAGM directs contextually informative channel selection based on the context guiding vector by applying a modifying scaling factor to each channel based on the context guiding vector, to perform context-aware guided sparsification (CAGS).
12. A method of neural network pruning, comprising:
 - receiving, at a context-aware guiding module (CAGM), a plurality of feature maps for an image, each feature map corresponding to a channel of the neural network;
 - forming, using the CAGM, a context guiding vector from a layer-wise integrated channel interdependency of the feature maps; and
 - directing contextually informative channel selection using the CAGM, based on the context guiding vector.
13. The method of claim 12, wherein directing contextually informative channel selection comprises pruning one or more channels of the neural network using a pruning module, based on the modified scaling factor for the respective channel.
14. The method of claim 12, wherein directing contextually informative channel selection comprises applying a weight to each channel based on an individual integrated interdependent level of the respective channel relative to one or more other said channels.
15. The method of any one of claims 12 to 14, wherein forming the context guiding vector comprises computing a channel affinity matrix and forming the context guiding vector from the channel affinity matrix.
16. The method of claim 15, wherein forming the context guiding vector comprises squeezing the channel affinity matrix based on dimensions of a spatial size of the layer supplying the feature maps.
17. The method of any one of claims 12 to 16, wherein forming the context guiding vector comprises forming a context guiding vector carrying an adaptive penalty strength for selectively penalising channels of the neural network.

18. The method of claim 17, further comprises modifying one or more scaling factors in a normalisation layer using the adaptive penalty strength.
19. The method of any one of claims 12 to 18, wherein the CAGM obtains the context guiding vector β^i by:

$$\beta_j^i = \frac{\sum_k a_{j,k}^i - \text{Min}(\sum_k a_{:,k}^i)}{\text{Max}(\sum_k a_{:,k}^i) - \text{Min}(\sum_k a_{:,k}^i)} \in [0,1]$$

wherein $\text{Max}()$ and $\text{Min}()$ compute a maximum and minimum value along a channel dimension of channel k , β_j^i is a guiding factor for channel j in layer i , and $a_{j,k}^i$ is an affinity level between channel j and channel k in layer i , and $a_{:,k}^i$ is an affinity level between channel k and remaining channels in layer i .

20. The method of any one of claims 12 to 19, wherein the CAGM directs contextually informative channel selection based on the context guiding vector by applying a modifying scaling factor to each channel based on the context guiding vector, to perform context-aware guided sparsification (CAGS).
21. The method of claim 20, wherein the CAGM performs CAGS based on a channel-to-channel interdependency as determined from the context guiding vector.
22. A method of image pattern recognition, comprising:
- receiving an image at an input layer;
 - formulating a plurality of feature maps for the image, each feature map corresponding to a channel;
 - performing the method of any one of claims 12 to 21; and
 - outputting the image pattern from an output layer, based on features of channels selected using the context guiding vector.