Neural Processing Letters (2024) 56:140
https://doi.org/10.1007/s11063-024-11593-1

Efficient Bayesian CNN Model Compression using Bayes by Backprop and L1-Norm Regularization

Ali Muhammad Shaikh[1] · Yun-bo Zhao[1] · Aakash Kumar[2] · Munawar Ali[1] · Yu Kang[1]

Accepted: 12 March 2024 / Published online: 4 April 2024

Abstract
The swift advancement of convolutional neural networks (CNNs) in numerous real-world utilizations urges an elevation in computational cost along with the size of the model. In this context, many researchers steered their focus to eradicate these specific issues by compressing the original CNN models by pruning weights and filters, respectively. As filter pruning has an upper hand over the weight pruning method because filter pruning methods don't impact sparse connectivity patterns. In this work, we suggested a Bayesian Convolutional Neural Network (BayesCNN) with Variational Inference, which prefaces probability distribution over weights. For the pruning task of Bayesian CNN, we utilized a combined version of L1-norm with capped L1-norm to help epitomize the amount of information that can be extracted through filter and control regularization. In this formation, we pruned unimportant filters directly without any test accuracy loss and achieved a slimmer model with comparative accuracy. The whole process of pruning is iterative and to validate the performance of our proposed work, we utilized several different CNN architectures on the standard classification dataset available. We have compared our results with non-Bayesian CNN models particularly, datasets such as CIFAR-10 on VGG-16, and pruned 75.8% parameters with float-point-operations (FLOPs) reduction of 51.3% without loss of accuracy and has achieved advancement in state-of-art.

Keywords Bayesian CNNs · Bayes by backprop · Capped L1-norm · L1-norm regularization · Model compression · Uncertainty estimation

✉ Yun-bo Zhao
ybzhao@ustc.edu.cn

Ali Muhammad Shaikh
alims@mail.ustc.edu.cn

Aakash Kumar
aakash@cust.edu.cn

Munawar Ali
alimunawar@mail.ustc.edu.cn

Yu Kang
kangduyu@ustc.edu.cn

1 University of Science and Technology of China, Hefei, China

2 Zhongshan Institute of Changchun University of Science and Technology, Zhongshan, China

1 Introduction

To achieve more efficient Convolutional Neural Networks (CNNs), researchers have utilized numerous state-of-the-art methods in a diverse range of machine learning practices, namely object detection [1, 2], image recognition [3, 4], and web search [5]. However, achieving outstanding outcomes also add up immense challenges to cope with, such as cumbrous architectures that are not quite efficient in terms of memory and have heavy computational requirements, specifically for mobile and embedded devices. Additionally, they incur significant inference costs. Hence, model compression acquired noteworthy consideration amongst researchers to decrease the size issue of CNN architecture. Even so, by nature, CNNs are computationally intensive, along with memory footprint, the inevitability of the floating-point operations (FLOPS) dramatically increased [6]. The overall growth is because of trainable parameters quantity is in millions along with convolution operations.

Besides, the increase in parameter count, the runtime memory also plays a crucial role because even with a single image, the activation layer of CNN possibly uses up additional memory footprint rather than saving parameters throughout the inference period. To contend with this sort of challenge more robust Graphical Processing Units GPUs are presented as the solution which is not so affordable for numerous implementations. Thus, to alleviate the incompatibility of elevated resource necessity of CNNs numerous approaches have been offered in terms of compressing CNNs in a diverse range of models without taking any evident accuracy loss. Network pruning is quite a prevailing technique amongst researchers for the compression of networks. Pruning can help to lower computation costs through dropping the number of feature maps. The earlier pruning approaches are merely utilized for networks that are fully connected namely second-order derivatives [7] and optimal brain damage [8]. The obvious drawback of these approaches is that parameter pruning does not offer a substantial reduction in computation time since eliminated parameters are mostly from fully connected layers.

There has been considerable research available in the area of compressing CNN networks instead of taking rather efficient CNN models directly. To alleviate the conflict of the high resource necessity of the CNNs, the literature comprises a variety of approaches to aid in compressing along with accelerating CNNs in a diverse range of models with no noticeable accuracy loss. For instance, approaches based on quantization have been suggested to make CNNs more appropriate for devices with limited resources [9, 10]. However, these approaches have a common issue of reduced accuracy. However, there are multiple ways to address the accuracy reduction issue. For instance, perform only parameter quantization rather on activations, by increasing the size of network, and performing fine-tuning. Conversely, the extensively utilized approach amongst researchers to compress networks is pruning. However, there are two subclasses of pruning namely filter pruning [11–13] and weight pruning [14–16]. In weight pruning parameters are eliminated directly in the filter and that produces unstructured sparsity causes an impact on efficiency of the network. Recently, inspiring outcomes attained through Bayesian-based techniques that utilize weight pruning [17–19]. This class of pruning scheme is still not quite impactful to ease up computational power along with memory footprint, and in the meantime dedicated Basic Linear Algebra Subprograms (BLAS) libraries are also compulsory. Also, there has been a noteworthy amount of research available for filter-level pruning [20–22]. This class of pruning does not take in additional hardware and allows a structured model. Filter pruning has the upper hand regarding eliminating redundant filters along with reducing model size without harming the structure of model.

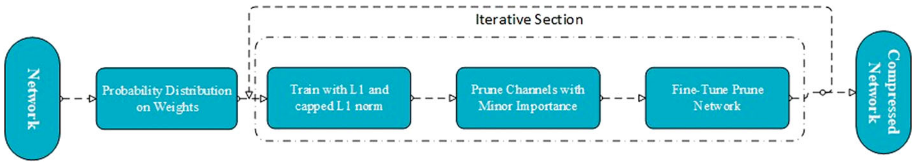
Iterative Section
Network
Probability Distribution on Weights
Train with L1 and capped L1 norm
Prune Channels with Minor Importance
Fine-Tune Prune Network
Compressed Network

Fig. 1 Structure of iterative algorithm

As another option, a Bayesian perception channel pruning alongside minimizing the bit precision for weights is associated with accomplishing higher-level accuracy, since the Bayesian approaches pursue the optimum structure of model. In this study, we have managed to create a Bayesian-based filter level pruning. In this work, Bayesian methods are applied for CNNs to evaluate uncertainty along with regularization within their predictions concerning their training. With the help of this scheme, the network can determine uncertainty through parameters as probability distributions. This provides the benefit of the regularization effect for the network that results in avoiding overfitting.

Further, In this work, we proposed a process of network compression by utilizing Bayes by Backprop [23] with approximate intractable true posterior probability distributions $p(w \mid \mathcal{D})$ along with variational probability distributions $q_\theta(w \mid \mathcal{D})$. This however has the properties of the Gaussian distribution denoted as $\mu \in \mathbb{R}^d$ and $\sigma \in \mathbb{R}^d$, represented as $\mathcal{N}\big(\theta \mid \mu, \sigma^2\big)$, here number of parameters are denoted as d which define probability distribution. Besides, variance σ^2 defines the form of Gaussian variational posterior probability distributions. Moreover, to prune the network we utilized L1 and capped L1 norm to regulate the tradeoff amongst regularization and selection of filter. The filters of nearly all layers with small L1-norm are picked and set to zero. This arrangement helps channel level pruning in the subsequent step and the overall performance will barely be affected by the parameter regularization. However, pruning redundant channels might degrade the performance momentarily but it can be eased through fine-tuning the pruned network. This proposition offers a slimmer network along with a compact structure regarding model size, runtime memory, and computational cost when compared to other techniques. The proposed pruning approach is iterative as demonstrated in Fig. 1. We evaluate our proposed method on prevailing CNN architecture through several standard datasets.

2 Literature Work

Using CNN on embedded devices is a modern trend that has a strong influence on mobile computing applications. However, utilization of CNNs in numerous applications added impressive computational costs increase along with immense size, and because of advancements CNNs transformed into a wider architecture. Hence, gradually increased parameter size and therefore greatly affects the applicability of models on embedded devices. However, the substantial redundancy in parameterization turns out to be an extensively recognized quality [24]. The redundant nature and over-parameterization of neural networks attract an increase in memory requirements and computational costs. For instance, VGG-16 [25] needs almost 30 billion float point operations (FLOPs) along with 138 million parameters and with 500 MB required storage. This incurs a substantial problem and also confines several CNN applications.

Moreover, the Bayesian comparable neural networks are termed as Bayesian neural networks (BNNs), where values of parameters of a particular network are generally represented via probability distributions. The BNN models have a handful of perks as compared to the non-Bayesian neural networks for instance BNN models not only acknowledge for integrating prior knowledge but offer robustness to overfitting along with simple continual learning [26]. There is not much research available regarding Bayesian networks [27], the authors in [28] presented a variational Bayes (VB) deterministic approximation for moments of activations of neural networks along with a straightforward empirical Bayes hyper-parameter update. Thus, their work achieved robust and efficient results through a combination of these two approaches. Besides, in another work [29] ReLU nonlinearities are decomposed into a product of an identity along with a Heaviside step function. Further, they familiarized another path that helps in decomposing neural network expectations from variance. Their methodology contains distinct latent binary variables for activations which causes neural network likelihood to behave as a chain of linear operations. This formulation is more robust than the Monte Carlo [30] sampling approaches because it allows computation that is sampling free of evidence lower bound.

Recently, considerable advancement in devices with limited power resources have crafted outstanding prospects for researchers to cope with issues of deploying deep learning systems on mobile devices with inadequate resources namely memory [31]. Accomplishing these objectives necessitates the computational costs reduction and memory requirements which aids in broadening the deep learning models applicability and being able to employ in a wide variety of applications namely embedded systems, real-time applications, and mobile devices. Although, several approaches available in the literature regarding coping with compressing CNNs have been introduced recently [32, 33], pruning in this field emerges as a famous solution which is eliminating redundant weights out of initial networks. While techniques related to pruning were conceived in early era of the 1980s-1990s, and were able to be utilized in deep learning networks [32]. However, [7, 8] are pioneers of technique of channel pruning, they demonstrated that by eliminating redundant weights from a trained network along with negligible loss in accuracy. Subsequently, in [14, 34] they presented that weights with small magnitude have less information and can be pruned. But then these sophisticated approaches are unconstructed along with hold format weight matrix. This will restrain the acceleration effect except by adopting the Compressed Sparse Column (CSC).

On the assumption that [35] presented that before the step of retraining, there is a possibility to override the retraining phase by a random initialization. Further, in [35] presented, swapped fully connected layers with sparsely connected layers through utilizing initial topology based upon Erdȍs–Rényi random graph. Through the training of the network, portions of the smallest weights are discarded iteratively and then swapped with new random weights. To find a sparse architecture prior training step is accomplished by employing initial topology. Nonetheless, the disadvantage also lies in the random iterative initialization because these all steps are quite expensive. This approach also causes jumping memory access along with poor cache locality, which tremendously impacts and confines practical acceleration [36].

Subsequently, [37] suggested that for the deeper architectures pruning networks with initialization values fail to perform better. Their solution was to set up weights for those that are acquired at early epochs of training the network. In [38] sparsity is adapted in the model parameters alongside they necessitate the aid of sparse libraries as well to accomplish intended results. Likewise, the mentioned approach gives a deficient compression rate on total run memory (TRM) along with FLOPs. Nonetheless, these specific methods provide a finer compression rate concerning weight storage, along with insubstantial FLOPs. However, in [39] proposed filter importance holds particular limits as required that are not usually accumulated. They proposed a methodology based upon meta-attribute-based filter pruning (MFP). They have broadened the current magnitude information which is based on pruning criterion and they also familiarized a new standard to contemplate the geometric distance of filters. Consequently, models endure redundancy because of these methodologies. In the meantime, these methodologies cease to anticipate filter redundancy during pruning. To remove the redundant feature maps in [33] proposed an approach based on the correlation between feature maps that are generated out of corresponding filter. This technique eliminates the redundant feature maps to aid in reducing the size of a model along with reduced computational cost plus being able to save many FLOPs too.

Likewise, in [40] presented Distinguishing Layer Pruning based on RFC (DLRFC). They presented a novel filter criterion that employs network interpretability to aid in constructing a filter peak feedback set, subsequently estimating redundancy based upon the uniformity of the filter's feedback toward the class. In this approach, they pruned filters in dissimilar layers discriminately. This helps in avoiding measuring filters amongst individual layers directly contrary to the filter criteria. Additionally, we have suggested how Bayes by Backprop can be applied to different CNN Models without trimming the network to half to make it similar to non-Bayesian CNN models because in this work we have utilized two convolutional operations for mean and variance which make the model double in the size as compare to non-Bayesian. We also inspected the aleatoric and epistemic uncertainties and made the network to become more deterministic. Further, we have applied L1-norm with capped L1-norm to help train the parameters of the model, prune the unimportant filters, and further fine-tune the model to reduce the accuracy loss that occurred during the pruning process.

In this work, we perform proposed approach on different datasets along with CNN models. For VGG16 on CIFAR100, the approach obtained 59.6% of parameter pruning along with 46.4% in FLOPs reduction and 0.17% accuracy loss. For VGG16 on CIFAR10, we achieved 44.6% parameter pruning without loss of accuracy. Further details of our proposed approach are briefly detailed in the coming sections of the article.

3 Proposed Methodology

3.1 Variational Inference

As we state function $y = f(x)$ which approximates the inputs $\{x_1, \ldots, x_N\}$ with relative outputs $\{y_1, \ldots, y_N\}$, which generates an estimated output, respectively. A prior distribution is applied through a span of functions $p(f)$ with utilization of Bayesian Inference. So, the distribution implies our prior beliefs regarding which exact functions are created in our data. Further, a likelihood can be stated as $p(Y|f, X)$ to obtain the procedure where a function observation is formed. In that case, the Bayes rule is utilized to locate the posterior distribution considering our dataset $p(f|X, Y)$.

So, through incorporating overall probable functions f a new output can be estimated for a new input point x^*,

$$p\big(y^* \mid x^*, X, Y\big) = \int p\big(y^* \mid f^*\big)p\big(f^* \mid x^*, X, Y\big)df^* \tag{1}$$

However, because of the integration symbol, Eq. 1 is intractable. But it can be approximated through utilizing a finite set of random variables such as w then condition model over it. This however makes the model dependent upon variables alone which results in sufficient statistics in our model. Hence, the distribution for a new input point x^* can be written as follows

$$p\big(y^* \mid x^*, X, Y\big) = \iint p\big(y^* \mid f^*\big)p\big(f^* \mid x^*, w\big)p(w|X, Y)df^*dw \tag{2}$$

Nonetheless, $p(w|X, Y)$ distribution still be intractable. A variational distribution $q(w)$ is required to approximate it, though is computable. The approximating distribution should be closer to posterior distribution that is acquired from original model. We then have to minimalize Kullback–Leibler (KL) divergence, intuitively a similarity measure amongst two diverse distributions; $KL(q(w)\|p(w \mid X, Y))$. This formation leads to approximate predictive distribution and becomes

$$q\left(y^* \mid x^*\right) = \iint p\left(y^* \mid f^*\right) p\left(f^* \mid x^*, w\right) q(w) df^* dw \tag{3}$$

The process of minimizing KL divergence is similar to maximizing log-evidence lower bound

$$KL_{VI} := \int q(w) p(\mathrm{F} \mid \mathrm{X}, \mathrm{w}) logp(\mathrm{Y} \mid \mathrm{F}) dFdw - KL(q(w)\|p(w))$$

referring to variational parameters determining $q(w)$, which is fundamentally a variational inference. Further, maximizing Kullback–Leibler (KL) divergence amongst both posterior and prior with w might provide a variational distribution which exactly learns a finer description out of data, i.e., achieved from log-likelihood, this makes it near to prior distribution.

3.2 Bayes by backprop utilization

With the aim of computing intractable true posterior probability distribution, we utilized variational inference such as Bayes by Backprop in our case. In order to build a CNN along with probability distributions above its weights in every filter. Besides, it is not possible to achieve a fully Bayesian perspective on a CNN through placing probability distributions over weights in convolutional layers, but also needs probability distributions over weights [41].

For a Bayesian neural network to learn the posterior distribution over weights $w \sim q_\theta(w \mid \mathcal{D})$ where weights w is sampled in backpropagation. Bayes by backprop familiarized by [23] (basically implies a more empirical solution for the challenge of intractability can be solved sufficiently. Additionally, this particular distribution through minimizing the compression cost can be able to regularize weights. This termed as variational free energy. However, true posterior is intractable which leads to approximate distribution $q_\theta(w \mid \mathcal{D})$ which is then intended to mimic true posterior $p(w \mid \mathcal{D})$ and this calculated through the KL-divergence [42]. Thus, this formation determines optimal parameters θ^{opt} as follows:

$$\begin{aligned} \theta^{\mathrm{opt}} &= \underset{\theta}{\arg min} KL[q_\theta(w \mid \mathcal{D})\|p(w \mid \mathcal{D})] \\ &= \underset{\theta}{\arg min} KL[q_\theta(w \mid \mathcal{D})\|p(w)] \\ &\quad -\mathbb{E}_{q(w|\theta)}[logp(D \mid w)] + logp(D) \end{aligned} \tag{4}$$

now KL-divergence is defined as

$$\mathrm{KL}[q_\theta(w \mid \mathcal{D})\|p(w)] = \int q_\theta(w \mid \mathcal{D}) \log \frac{q_\theta(w \mid \mathcal{D})}{p(w)} dw. \tag{5}$$

The outcome of the above derivation is an optimization issue including a resulting cost function termed as "variational free energy" [43, 44]. This cost function carried out by two terms i.e. former, $\mathrm{KL}[q_\theta(w \mid \mathcal{D})\|p(w \mid \mathcal{D})]$ which basically rely on prior $p(w)$, termed as complexity cost, while the later term, $\mathbb{E}_{q(w|\theta)}[\log p(\mathcal{D} \mid w)]$ rely on data $p(\mathcal{D} \mid w)$, termed as likelihood cost. However, in the optimization, we can exclude $\log p(\mathcal{D})$ since it is constant. Furthermore, we cannot exactly compute KL-divergence due to its intractable nature so we have to utilize another term such as stochastic variational scheme [23]. The weights w sampled from $q_\theta(w \mid \mathcal{D})$ variational distribution, considering the much easier approach to extract the samples that are relevant to numerical approaches from variational posterior $q_\theta(w \mid \mathcal{D})$ compared with true posterior $p(\mathcal{D} \mid w)$ in this case. Hence a tractable cost function is formulated in Eq. 6 which is intended to be minimized along with optimized regarding θ, throughout training session.

$$\mathcal{F}(\mathcal{D},\theta) \approx \sum_{i=1}^{n} \log q_\theta\left(w^{(i)} \mid \mathcal{D}\right) - \log p\left(w^{(i)}\right) - \log p\left(\mathcal{D} \mid w^{(i)}\right) \tag{6}$$

In the above equation, n represents the number of extractions performed. Moreover, $w^{(i)}$ is sampled out of $q_\theta(w \mid \mathcal{D})$.

3.3 Local Reparameterization Trick

In this work, local reparameterization trick [45] is utilized for CNNs. Further, in our work layer activation b is sampled instead of weights w because of resulting computational acceleration of layer activation b. Moreover, $q_\theta\left(w_{ijhw}\mathcal{D}\right) = \mathcal{N}\left(\mu_{ijhw}, \alpha_{ijhw}\mu^2_{ijhw}\right)$ represents variational posterior probability distribution and for any layer, i and j are input along with output layers, where h and w denoted as height and width, respectively. This formation within convolutional layer can permit to apply local reparameterization trick. The convolutional layer activation b can be written as follows:

$$b_j = A_i * \mu_i + \epsilon_j \odot \sqrt{A_i^2 * \left(\alpha_i \odot \mu_i^2\right)} \tag{7}$$

here receptive field is represented as $\epsilon_j \mathcal{N}(0,1)$,$A_i$ along with $\odot$ which is component-wise multiplication, and signalizes convolutional operation denoted as $*$.

We deployed a different estimator for which $Cov\left[L_i, L_j\right] = 0$, so the variance of the stochastic gradients scales as $1/M$. After that new estimator is constructed computationally efficient through sampling intermediate variables along with not directly sampling ϵ though $f(\epsilon)$ with which ϵ impacts $L_D^{SGVB}(\varnothing)$. With this source of global noise can interpreted into local noise ($\epsilon \to f(\epsilon)$). Hence, to achieve an effective gradient estimator local reparameterization is applied. For instance, an input (X) is stated as a random uniform function with values between (-1 to + 1) with an output of (Y) denoted as a random normal distribution over mean X along with standard deviation δ, and $(Y - X)^2$ stated as Mean Squared Loss. During backpropagation from random normal distribution leads to this issue and due to the reason of propagating over a stochastic node, then we reparametrize it with adding X in order to random normal distribution output and then multiply it with standard deviation.

3.4 Utilizing Sequential Convolutional Operations

The crucial point of CNN which has probability distributions over weights rather than single point estimates which is also capable of updating variational posterior probability distribution $q_\theta(w \mid \mathcal{D})$ through backpropagation exists in utilizing more than one convolutional operation but filters with a single point estimates utilize only one operation. Moreover, this work utilized local parameterization trick along with sample out of output b. Given that b is considered as a function of mean μ_{ijwh} and variance $\alpha_{ijhw}\mu^2_{ijhw}$, this help to separately calculate two variables defining a Gaussian probability distribution such as of mean μ_{ijwh} and variance $\alpha_{ijhw}\mu^2_{ijhw}$. The detailed explanation is defined as follows:

- Output b is treated as an output of CNN which is updated through frequent inference. Adam optimizer [46] is utilized in order to obtain a single-point estimate. This single point-estimate is interpreted as mean μ_{ijwh} .
- In the second convolutional operation, we familiarize variance $\alpha_{ijhw}\mu^2_{ijhw}$, here variance comprises mean μ_{ijwh} so requires to learn α_{ijhw} section in second convolutional operation [18]. This formulation ascertains that one parameter is updated for every convolutional operation.

3.5 Predictive Uncertainty for Convolutional Neural Networks (CNNs)

To estimate the uncertainties, we modeled Epistemic uncertainty through putting prior distributions over weights of the model and attempting to apprehend how often the weights deviate provided with some data. Besides, we modelled Aleatoric uncertainty though placing distribution over the model output.

The predictive distribution $p_{\mathcal{D}}(y^* \mid x^*)$, is the main concern in classification tasks, where y^* is considered a predictive class and x^* represents unseen data example. As for Bayesian neural network concern, predictive distribution can be written as:

$$p_{\mathcal{D}}\big(y^* \mid x^*\big) = \int p_w\big(y^* \mid x^*\big)p_{\mathcal{D}}(w)dw \tag{8}$$

The finite and discrete characteristics of majority of classification tasks lead to assuming predictive distribution to be categorical which results in;

$$\begin{aligned} p_{\mathcal{D}}(y^* \mid x^*) &= C \textstyle\int (y^* \mid f_w(x^*))\mathcal{N}\big(w \mid \mu, \sigma^2\big)dw \\ &= \textstyle\int \prod_{c=1}^{C} f\big(x_c^* \mid w\big)^{y_c^*} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(w-\mu)^2}{2\sigma^2}} dw \end{aligned} \tag{9}$$

In Eq. 9 C represents an overall number of classes along with $\sum_c f\big(x_c^* \mid w\big) = 1$. Moreover, an unbiased estimator of expectation can be created via sampling from $q_\theta(w \mid \mathcal{D})$. This is a must because of insufficient conjugacy amongst Gaussian distribution and categorical. Now we can formulate as follows:

$$\begin{aligned} \mathbb{E}_q\big[p_{\mathcal{D}}(y^* \mid x^*)\big] &= \textstyle\int q_\theta(w \mid D)p_w(y \mid x)dw \\ &\approx \tfrac{1}{T}\textstyle\sum_{t=1}^{T} p_{w_t}(y^* \mid x^*) \end{aligned} \tag{10}$$

In above equation T represents pre-defined samples. Now this estimator is helpful to measure our uncertainty through definition of variance. This formation is termed as "predictive variance" represented as Var_q and defined as follows:

$$Var_q\big(p\big(y^* \mid x^*\big)\big) = \mathbb{E}_q\Big(yy^T\Big) - \mathbb{E}_q[y]\mathbb{E}_q[y]^T \tag{11}$$

Now we can fetch the epistemic uncertainty and aleatoric uncertainty from Eq. 11 and given as follows:

$$\mathrm{Var}_q\big(p\big(y^*|x^*\big)\big) = \underbrace{\frac{1}{T}\sum_{t=1}^{T} \mathrm{diag}\big(\hat{p}_t\big) - \hat{p}_t\hat{p}_t^T}_{\text{aleatoric}} + \underbrace{\frac{1}{T}\sum_{t=1}^{T}\big(\hat{p}_t - \overline{p}\big)\big(\hat{p}_t - \overline{p}\big)^T}_{\text{epistemic}} \tag{12}$$

$$\text{where } \overline{p} = \frac{1}{T}\sum_{t=1}^{T}\widehat{p_t} \text{ and } \widehat{p_t} = \text{Softmax}\big(f_{w_t}(x^*)\big)$$

Moreover, to weigh how much a model can be improved, it is necessary to discriminate between two types of uncertainty: aleatoric and epistemic. Aleatoric uncertainty (or statistical uncertainty) reveals the variability of the data, which might be incomplete or noisy. while Epistemic uncertainty arises from the model, which might be incomplete or inaccurate. As a result of separating these two causes of uncertainty, a modeler can detect whether the quality of data is poor (high aleatoric uncertainty) or the quality of the model is poor (high epistemic uncertainty).

3.6 Model Pruning

3.6.1 Filter Pruning

The CNN is parametrized as $\{W^{(i)} \in \mathbb{R}^{M_i \times N_i \times K \times K}, 1 \le i \le L\}$. The matrix of connection weights within the i-th layer is represented as $W^{(i)}$ whereas L in denotes the total number of layers. Further, a number of input channels are represented by M_i within the i-th convolutional layer, and output channels are represented as N_i along with height x and $K \times K$ denoted as width. While $w_i \times h_i$ is input feature map size and output feature map size is denoted as $w_{i+1} \times h_{i+1}$. Furthermore, the i-th layer comprises M_i, N_i kernel, and regarding $k \times k$ kernel the convolutional layer computations are represented as $M_i \times N_i \times k^2 \times w_{i+1} \times h_{i+1}$. As we can see in Fig. 2 which shows the step of pruning filter parameter, where its correlated feature map is also eliminated, which then shrinks $M_i \times k^2 \times w_{i+1} \times h_{i+1}$ within the i-th layer. Hence, in the subsequent convolutional layer filters are also removed considering kernels have been applied in the eliminated feature maps in preceding layer. This formation saves further $N_i \times k^2 \times w_{i+2} \times h_{i+2}$ processes within $(i+1)$th layer, respectively. Moreover, the filters are pruned according to the importance valuations at each end of an epoch. In Fig. 2 the filters are in blue and orange horizontal bars, the importance measured through their L1 norm along with ones that contain smaller values are then chosen to be pruned.

Moreover, N indicates the images that are randomly selected which are input to the model, the greater value of N leads to more consumption of memory of the system. However, value of N is the same for each considered dataset. The minimum achieved scores of final accuracy are out of N of 16, 32, 50, and 64. Conversely, the maximum achieved scores of final accuracy are shown by N of 256 and 512 occurrences. Further, the N of 100, 128, 150, and 200 cases display the average results of the final accuracy. Consequently, the higher the value of N, the better final accuracy is achieved, also copious values can further affect the final accuracy as shown in Fig. 3. (Fig. 3 only emphasizes CIFAR-10 dataset for the VGG16 model). For instance: in Fig. 3 we can see that the loss in accuracy is higher if the N value is 2^{16} and at 2^{14} we have gained accuracy and as the value of N changes the accuracy can be gained accordingly. Thus, the Value of N highly affects the model accuracy.

3.6.2 Relevance of Weights of Filters

Fusing the lasso with L1-regularization and linear classification in Eq. 13 [47] we get:

$$V = \min_{W} \sum_{(x_i, y_i)} l(x_i, y_i, W) + \lambda |W|_1 \tag{13}$$

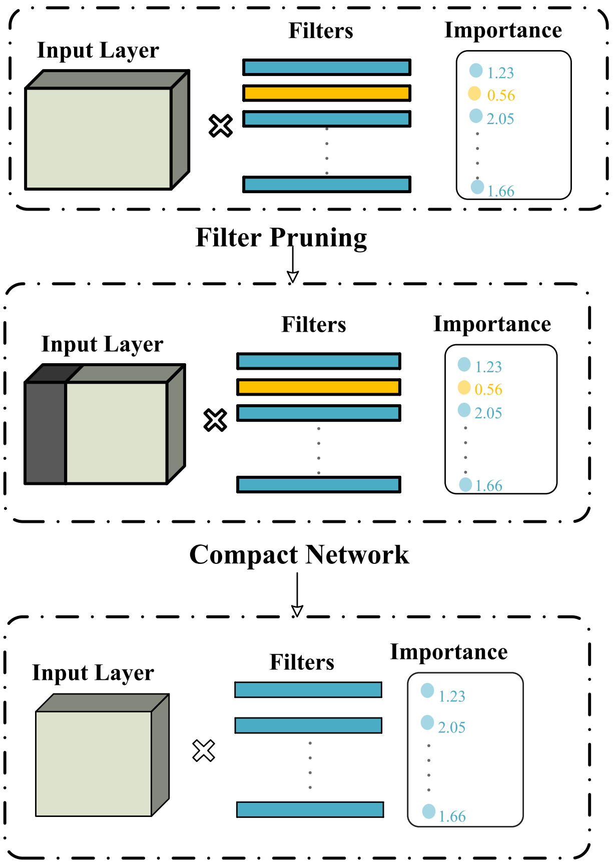

Fig. 2 Filter pruning process

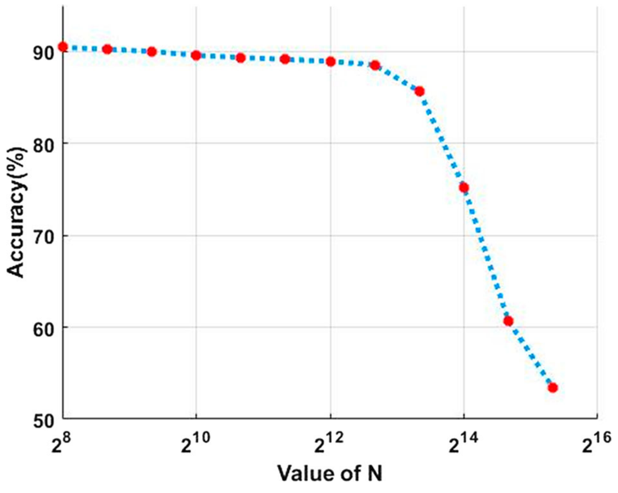

Fig. 3 Impact of N on VGG16 accuracy with CIFAR-10 dataset

where the square loss is denoted as $l(\bullet)$, $l(x_i, y_i, W) = \left(W^T x_i - y_i\right)^2$. However, there is a possibility of other sort of loss function owing to feature selection. In this work, to accurately determine the significance of convolutional kernels we fused both L1-norm and capped L1-norm. The Capped L1-norm has an upper hand over L1-norm given that further penalizing cannot be utilized after a feature is extracted. Because definitively not going to provide small weights further.

$$q_\epsilon(w_i) = min(|w_i|, \epsilon) \tag{14}$$

Equation 14, the Capped L1-norm is indicated as an element-wise function, where w_i is stated as weights of filters, and ϵ stated as a constant. The more accurate approximation of the L1-norm is that it just penalizes a feature when utilized without any interference, scaling the magnitude of weights. Once ϵ is substantially small as an instance, $\epsilon \leq \min_i |w_i|$, then we might be able to calculate an exact number of features extracted through $q_\epsilon(w)/\epsilon$. Simply put, penalizing $q_\epsilon(w)$ is a very close proxy to penalizing number of extracted features. However, the capped L1-norm is not convex which makes it quite bothersome to optimize. So, to ease up the complexity by fusing the norms such as L1-norm or L2-norm with capped L1-norm to help in controlling the trade-off among filter selection and regularization through adjusting the desired parameters, $\mu, \lambda \geq 0$,

$$V = \underset{W}{min} \sum_{(x_i, y_i)} l(x_i, y_i, W) + \lambda|W|_1 + \mu q_\epsilon, (w) \tag{15}$$

Here W is training weights, (x_i), (y_i) are train input and output. The first section of Eq. 15 corresponds to normal training loss. The second section is a non-structured regularization which is applied to each filter $q_\epsilon(w)$ that denotes capped L1-norm that is applied to each layer. Further, in Eq. 15 there are two penalty terms namely ordinary L1- norm and capped L1-norm. The standard L1-norm drops the overfitting while second penalty which is capped L1-norm selects filters. However, the contemporary form of capped L1-norm selects features rather than selecting filters. Hence, above equation needs modification so that it directly penalizes feature extraction. To model a total number of features that are extracted through a set of filters we state a binary matrix $F \in \{0, 1\}$ which has the dimension of $\times T$, whereas an entry $F_{ft} = 1$, assuming the filter h_t utilizes feature f. Now we can determine total weight assigned for filter extracted with feature f:

$$W = \sum_{t=1}^{T} \left|F_{ft}\beta_t\right| \tag{16}$$

where β represents the sparse linear vector, by modifying $q_\epsilon(w)$ to make it penalize actual weights allocated to features. Equation 17 shows the final optimization;

$$L = \&\underset{W}{min} \sum_{(x_i, y_i)} l(x_i, y_i, W) + \lambda|W|_1 + \mu \sum_{f=1}^{d} q_\varepsilon \left(\sum_{t=1}^{T} \left|F_{ft}\beta_t\right| \right) \tag{17}$$

Furthermore, if ϵ is quite small previously $\left(\epsilon \leq \underset{\mathrm{f}}{min}\left|\sum_{\mathrm{t}=1}^{\mathrm{T}} \mathrm{F}_{\mathrm{ft}}\beta_{\mathrm{t}}\right|\right)$, then $\mu = 1/\epsilon$ can be set along with choosing the feature penalty relates accurately with utilized features. The capped L1-norm in Eq. 17 is for estimating the relevance of every filter. Generally, convolutional result of a filter having a smaller L1-norm value leans reasonably small compared to activation values; which results in inconsiderable numerical impact over final estimation of deep CNN-based models. We prune filters throughout several layers of CNN with the help of adjusting the pruning rate. For instance, prune 70% of the entire CNNs. However, network defined the number of filters needed to prune in every layer. This formation avoids hurdle of adjusting the pruning rate layer by layer. We used values of L1-norm and capped-L1 norm as a criterion for setting the filter choice for pruning. At this time, pruning might be able to direct towards some loss in accuracy but it occurs only when the percentage of pruning is high enough. Although, through fine-tuning this issue can be rectified. Our testing suggests that the fine-tuning procedure on pruned network might achieve better accuracy as compared to original unpruned network. The scheme is also capable of saving the training time efficiently. Algorithm 1 shows the steps based on our introduced method.

Algorithm 1 Prunning algorithm illustration

```
Data: data for training: $\mathbf{X}$, pruning ratio: 30% to 70% the model with
      weights
$\mathbf{W} = [W^{(i)}, 1 \leq i \leq L]$.
Result: The slimmer Bayesian network and its weights $W^*$
1  Probability Distribution on weights (Bayes by Backprop);
2  for layer $i$ = 1; $i \leq L; i++$ do
3      according to Bayes Backprop get probability distribution on each layer
        weights ;
4  end
5  Initialization the model probablity distributed weights $\mathbf{W}$;
6  for iteration = 1; iteration $\leq$ totaliterations; iteration ++ do
7      Based on $\mathbf{X}$, Update the model weights $\mathbf{W}$ ;
8  end
9  Train With $L_1$Regularization
10 for i=1; $i \leq L; i++$ do
11     compute L1-norm and capped L1-norm for each filter $\|\mathcal{F}_{i,j}\|_1, \leq j \leq N_{i+1}$;
12     sparsify $N_{i+1}$ x pruning − ratio, filters through L1-norm filter selection;
13 end
14 Prune Channels with Small $L_1$ − normimportance
15 Fine-tune Process, j==4
16 for i=1;$i \leq j; i++$ do
17     Compensate the Accuracy loss after fine-tune;
18 end
19 Achieved the Slimmer network with probablity distributed weights $W^*$
```

The proposed technique can perform smoothly on different architectures namely AlexNet, and VGGNet, respectively. Moreover, in this work, a Bayesian Convolutional Network learns two weights, for instance, the mean and the variance both are compared to point estimate learning one single weight. Thus, the parameters of a Bayesian Network doubled as compared to the parameters of a point estimate similar architecture. Further, we take the weights of all the layers, apply the L1 norm, and capped L1 norm over it, and for weights values with zero or below with a defined threshold are removed and the model is pruned. After that to compensate for the accuracy loss of the model due to pruning we fine-tune the model. Since Bayesian CNNs have twice the parameters we did not trim the network to half to make it similar to non-Bayesian models.

4 Experimental Framework

4.1 Experiment Settings

We evaluated our proposed framework on a few of the most extensively utilized networks and datasets. The LeNet-5 architecture utilized on MNIST, AlexNet architecture utilized on the ImageNet dataset, and VGG-16 architecture was utilized for CIFAR-10 and CIFAR-100 datasets, respectively. During training sessions, data augmentation is applied which helps in cropping images into 32×32 with padding of four along with a horizontal flip is performed also. Further, the mini-batch size for training is set to 100 and for the test images, the mini-batch size is set to 1000. During the process of fine-tuning and training, initial learning rate is set to 0.1 and separated through 10 at 50% and 75% of the total 150 epochs. Furthermore, while training with L1- norm and capped L1 norm, we also adjust the hyper-parameter λ. This helps to manage the trade-off between empirical loss and sparsity. This is picked through a grid search over 10^{-3}, 10^{-4}, and 10^{-5} over CIFAR-10 test set. However, we go for $\lambda = 10^{-4}$ and $\lambda = 10^{-5}$ for VGGNet. Lastly, all further settings are kept same as in standard training. Furthermore, The NVIDIA GTX TITAN Xp GPU is used for experiments with a Python framework known as Pytroch.

4.2 Learning the Objective Function

Bayes by Backprop is utilized for learning objective functions. This framework regularizes weights w through minimizing a compression cost which is another term for variational free energy. The intractability issue is explained in the previous section which concluded the tractable cost function as follows:

$$\mathcal{F}(\mathcal{D},\theta) \approx \sum_{i=1}^{n} \log q_\theta\left(w^{(i)} \mid \mathcal{D}\right) - \log p\left(w^{(i)}\right) - \log p\left(\mathcal{D} \mid w^{(i)}\right) \tag{18}$$

In Eq. 18 n is the number of draws. Moreover, in Eq. 18 the left side term is variational posterior $q_\theta\left(w^{(i)} \mid \mathcal{D}\right) = \prod_i \mathcal{N}\left(w_i \mid \mu, \sigma^2\right)$, where variational posterior taken in a form of Gaussian distribution which is centered around mean μ and variance σ^2, respectively. Now log and log posterior are outlined as follows:

$$\log\left(q_\theta\left(w^{(i)} \mid \mathcal{D}\right)\right) = \sum_i \log \mathcal{N}\left(w_i \mid \mu, \sigma^2\right) \tag{19}$$

Furthermore, in Eq. 18 the other term on is prior over the weights which stated as a product of individual Gaussians and defined as follows:

$$p\left(w^{(i)}\right) = \prod \mathcal{N}\left(w_i \mid 0, \sigma_p^2\right) \tag{20}$$

whereas log and the log prior are outlined as:

$$\log\left(p\left(w^{(i)}\right)\right) = \sum_i \log \mathcal{N}\left(w_i \mid 0, \sigma_p^2\right) \tag{21}$$

The last section of Eq. 18 $\log p\left(\mathcal{D} \mid w^{(i)}\right)$ is likelihood.

4.3 Model Pruning Results

For the compression work, we took all the weights of every layer of the network then L1 and capped L1 norm applied along with value of every weight set to zero or under a determined threshold are pruned.

As for the Bayesian-AlexNet on ImageNet dataset, $\lambda = 5 \times 10^{-6}$ and $\lambda = 10^{-4}$ and then trained the model for 100 iterations. Our proposed method pruned 65.9% of parameters along with achieving the model accuracy of 51.33% and a 39.6% reduction in FLOP. While, for LeNet-5 architecture on MNIST dataset, where 5 represents the number of layers in the network, which have an input layer containing two convolutional layers along with two fully connected (FC) layers and 431 K parameters in total. Furthermore, initial learning rate is set to 0.001–0.1 for an entire number of iterations. Our proposed method pruned 75.9% of parameters without losing much of the actuary in the process.

For VGG-16 model, we analyzed it with CIFAR-10 and CIFAR-100 datasets, respectively. Considering the VGGNet convolutional layers are diverse in robustness and information concentration [12]. As for CIFAR-10, during pruning of channels that are trained with L1-norm and capped L1-norm, a threshold of pruning is required to be calculated on filters. Further, in our method after pruning 50% of the parameters the accuracy of model gets some improvements in accuracy. The parameters pruning was able to reach 75.8% after the fourth iteration along with a 51.3% reduction in FLOP and having 0.01% enhancement in accuracy. The results are tabulated in Table 1a. Nonetheless, the process of pruning is accomplished through creating a slimmer model along with parallel weights are then copied from a model that is trained with L1-norm and capped L1-norm.

For the CIFAR-100 dataset on VGG-16, we used similar settings. As shown in Table 1b, the proposed method saved 35.6% of FLOPs right after the second iteration. In this formation, we choose a less pruning ratio contrary to CIFAR-10 since CIFAR-100 contains more classes and requires extra information to classify the images. Moreover, Note, that in general, we can maintain the original accuracy on VGG without sampling by simply fine-tuning with a small learning rate, as done at [50]. This will still induce (less) sparsity but unfortunately, it does not lead to good compression as the bit precision remains very high due to not appropriately increasing the marginal variances of the weights. We compared outcomes of CIFAR-10 and

Table 1 Pruning filters for our proposed Bayesian model compared with non-Bayesian models

Approach	Baseline accuracy (%)	Parameters pruned (%)	FLOPs saved (%)	Accuracy (%)
(a)VGG-16 on CIFAR-10 datasets (Prune results)				
[33]	93.73	89.7	52.4	93.82
[48]	93.75	90.5	65.56	93.80
[49]	99.47	62.38	42.56	99.25
Ours	89.35	75.8	51.3	90.33
(b) VGG-16 on CIFAR-100 datasets (Prune results)				
[33]	73.72%	59.6%	46.4%	73.55%
[22]	73.44%	56.54%	44.2%	73.61%
[49]	90.38%	35.62%	33.38%	90.30%
Ours	70.23%	40.4%	35.6%	70.33%

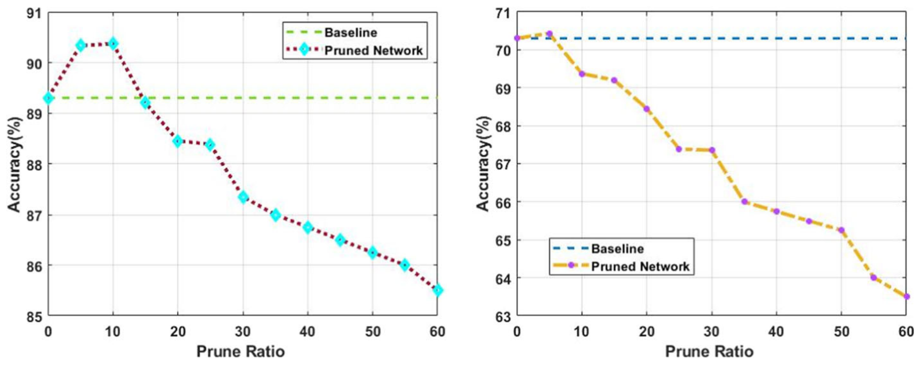

Fig. 4 Pruning results of VGG-16 on CIFAR-10(L) and on CIFAR-10(W) regarding pruning ratio

Table 2 MNIST and CIFAR-10 uncertainties comparison based on the proposed method by [50]

Datasets	Epistemic uncertainty	Aleatoric uncertainty
CIFAR-10	0.0402	0.1918
MNIST	0.0024	0.0094

CIFAR-100 dataset on VGG-16 pruning filters with baseline accuracy of the Bayesian CNN model with the proposed technique Fig. 4 displays the prune ratio results.

Furthermore, Table 1 displays the comparison between frequent networks with our proposed Bayesian CNNs network. In our proposed work, a Bayesian Convolutional Network acquires two weights, such as the mean and the variance. If we compare it to point estimate learning one single weight is there to learn. This formation makes the overall number of parameters of a Bayesian Network twice in contrast to the parameters of a non-Bayesian architecture.

For every parameter for a frequentist inference network, Bayesian CNNs have two parameters (μ, σ). Instead of trimming the parameters in half to ensure the number of parameters is comparable to non-Bayesian models. Thus, having double the amount of parameters, the proposed model has achieved considerable accuracy in the process.

Thus, our proposed method has combined both L1-norm and capped L1-norm providing control over trade-offs amongst filter selection and regularization. With the help of the threshold make the weights to be zero if the value falls below the threshold and only keep the non-zero weights. Besides, we also highlighted the filter importance in every convolutional layer of the neural network along with results implying that within many layers' maximum number of filters possesses a minor effect on architectures' performance in general.

4.4 Uncertainty Estimation

We utilized two small datasets such as CIFAR-10 and MNIST to compare both aleatoric and epistemic uncertainties for the Bayesian LeNet-5 having variational inference. For CIFAR-10 dataset the aleatoric uncertainty is twenty times greater than the MNIST dataset. This is due to aleatoric uncertainty calculates irreducible variability besides this it relies upon predictive values. In contrast for CIFAR-10 dataset, epistemic uncertainty is approximately fifteen times greater than the MNIST dataset. As this likely happens because epistemic uncertainty most likely shrinks the proportionally to validation accuracy. Table 2 shows the comparison of both uncertainties for CIFAR-10 and MNIST datasets, respectively.

5 Conclusion

In this work, we suggested Bayesian CNNs with Bayes by Backprop as a variational inference approach for the CNN. We then estimated the model's uncertainties such as aleatoric and epistemic and at the end we utilized a capped L1-norm combined with a regular L1-norm to observe the filter importance that measures the effectiveness of filter weights. We inspect uncertainties both aleatoric and epistemic and suggest that both be able to compute for the proposed work along with how epistemic uncertainties possibly compact through more training data. we applied channel pruning on Bayesian CNN which performs better and is equally decent comparable to a frequentist method.

The combined utilization of L1-norm and capped L1-norm provided control over trade-offs amongst filter selection and regularization. Besides, we also emphasized the filter importance in each convolutional layer of neural networks along with results implying that within many layers' maximum number of filters possesses an inconsiderable impact on the architecture' performance in general. Moreover, Considerable research illustrates the benefit of our proposed approach with a comparison of the non-Bayesian approaches. Particularly, in the VGG-16 model on CIFAR-10 datasets, our presented approach can prune 75.8% of parameters along with yield 51.3% FLOPs drop with slight accuracy improvement.

Besides, normal distribution utilization as prior for the purpose of estimating uncertainty was similarly done in [51] which showed that standard normal prior drives function posterior to simplify in unanticipated means on inputs outside of training distribution. Hence, adding noise into a normal distribution as prior is fruitful for superior uncertainty estimation. But in our experiments, we did not encounter such which can be explored in future work. Further, current work can be extended by utilizing models of Super Resolution (SR) which is the recovery of a High-Resolution (HR) image from a certain Low-Resolution (LR) image. Besides SR, another extension of the proposed work is Generative Adversarial Networks [52]. Also, a possible improvement can be made in future work to utilize trimmed versions of models such as halving the parameters in Bayesian CNNs because of two parameters instead of one similar to non-Bayesian networks to build a custom network to improve the overall accuracy.

Author contributions AMS conceptualized the research, collected the data, and performed data analysis. Y-Z assisted with formal analysis and investigation and also served as the corresponding author. AKB and MA reviewed and edited the draft. YK and Y-Z supervised the entire research during various phases of the research and provided meaningful guidelines. AMS wrote the first draft of the manuscript and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Data availability The datasets generated during and/or analyzed during the current study are available from the corresponding author upon reasonable request.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

Ethical approval I certify that there is no actual or potential conflict of interest about this article. This research does not involve human participants and/or animals and also does not require informed consent.

References

1. Zhang L, Sheng Z, Li Y, Sun Q, Zhao Y, Feng D (2020) Image object detection and semantic segmentation based on convolutional neural network. Neural Comput Appl 32(7):1949–1958. https://doi.org/10.1007/s00521-019-04491-4
2. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit, pp 580–587, https://doi.org/10.1109/CVPR.2014.81
3. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit vol 2016, pp 770–778, https://doi.org/10.1109/CVPR.2016.90
4. Fang W, Zhang F, Sheng VS, Ding Y (2018) A method for improving CNN-based image recognition using DCGAN. Comput Mater Contin 57(1):167–178. https://doi.org/10.32604/cmc.2018.02356
5. Nguyen PQ, Do T, Nguyen-Thi AT, Ngo TD, Le DD, Nguyen TAH (2016) Clustering web video search results with convolutional neural networks. In: NICS 2016—Proc 2016 3rd Natl Found Sci Technol Dev Conf Inf Comput Sci pp 135–140, https://doi.org/10.1109/NICS.2016.7725638
6. Kumar A et al (2022) Structure level pruning of efficient convolutional neural networks with sparse group LASSO. Int J Mach Learn Comput. https://doi.org/10.18178/ijmlc.2022.12.5.1111
7. Babak Hassibi DGS (2014) Second order derivatives for network pruning: optimal brain surgeon. pp 1–8, 2014, [Online]. Available: https://authors.library.caltech.edu/54983/3/647-second-order-derivatives-for-network-pruning-optimal-brain-surgeon(1).pdf
8. Le Cun Y, Denker JS, Solla S (1990) Optimal brain damage. Adv Neural Inf Process Syst 2(1):598–605
9. Goncharenko A, Denisov A, Alyamkin S (2022) Fast adjustable threshold for uniform neural network quantization. Low-Power Comput Vis. https://doi.org/10.1201/9781003162810-6
10. Choukroun Y, Kravchik E, Yang F, Kisilev P (2019) Low-bit quantization of neural networks for efficient inference. In: Proceedings—2019 international conference on computer vision workshop, ICCVW 2019, https://doi.org/10.1109/ICCVW.2019.00363
11. Yu R et al (2018) NISP: pruning networks using neuron importance score propagation. In: Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit, pp 9194–9203, https://doi.org/10.1109/CVPR.2018.00958
12. Li H, Samet H, Kadav A, Durdanovic I, Graf HP (2016) Pruning filters for efficient convnets. In: 5th Int Conf Learn Represent ICLR 2017—Conf Track Proc, pp 1–13
13. He Y, Ding Y, Liu P, Zhu L, Zhang H, Yang Y (2020) Learning filter pruning criteria for deep convolutional neural networks acceleration. In: Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit vol 2, pp 2006–2015, https://doi.org/10.1109/CVPR42600.2020.00208
14. Han S, Pool J, Tran J, Dally WJ (2015) Learning both weights and connections for efficient. Neural Netw, pp 1–9, https://doi.org/10.1016/S0140-6736(95)92525-2
15. Carreira-Perpiñán MA, Idelbayev Y (2018) Learning-compression' algorithms for neural net pruning. In: Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit, pp 8532–8541, https://doi.org/10.1109/CVPR.2018.00890
16. Liu B, Wang M, Foroosh H, Tappen M, Penksy M (2015) Sparse convolutional neural networks. In: Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit, vol 07–12-June, pp 806–814, 2015, https://doi.org/10.1109/CVPR.2015.7298681
17. Guo Y, Yao A, Chen Y (2016) Dynamic network surgery for efficient DNNs. [Online]. Available: http://arxiv.org/abs/1608.04493
18. Molchanov D, Ashukha A, Vetrov D (2017) Variational dropout sparsifies deep neural networks. 2017, [Online]. Available: http://arxiv.org/abs/1701.05369

19. van Baalen M et al (2020) Bayesian bits: unifying quantization and pruning. Adv Neural Inf Process Syst, vol 2020, no. NeurIPS
20. Wang W, Fu C, Guo J, Cai D, He X (2019) COP: customized deep model compression via regularized correlation-based filter-level pruning. In: IJCAI Int Jt Conf Artif Intell, vol 2019, pp 3785–3791, https://doi.org/10.24963/ijcai.2019/525
21. He Y, Kang G, Dong X, Fu Y, Yang Y (2018) Soft filter pruning for accelerating deep convolutional neural networks. In: IJCAI Int Jt Conf Artif Intell vol 2018, pp 2234–2240, https://doi.org/10.24963/ijcai.2018/309
22. Kumar A, Shaikh AM, Li Y, Bilal H, Yin B (2021) Pruning filters with L1-norm and capped L1-norm for CNN compression. Appl Intell. https://doi.org/10.1007/s10489-020-01894-y
23. Blundell C, Cornebise J, Kavukcuoglu K, Wierstra D (2015) Weight uncertainty in neural networks. In: 32nd International conference on machine learning, ICML 2015
24. Denton E, Zaremba W, Bruna J, LeCun Y, Fergus R (2014) Exploiting linear structure within convolutional networks for efficient evaluation. Adv Neural Inf Process Syst 2(January):1269–1277
25. Guerra E, de Lara J, Malizia A, Díaz P (2009) Supporting user-oriented analysis for multi-view domain-specific visual languages. Inf Softw Technol 51(4):769–784. https://doi.org/10.1016/j.infsof.2008.09.005
26. Jospin LV, Buntine W, Boussaid F, Laga H, Bennamoun M (2020) Hands-on Bayesian neural networks—a tutorial for deep learning users. IEEE Comput Intell Mag 17(2):29–48. https://doi.org/10.1109/MCI.2022.3155327
27. Beckers J, Van Erp B, Zhao Z, Kondrashov K, De Vries B (2023) Principled pruning of bayesian neural networks through variational free energy minimization. IEEE Open J Signal Process. https://doi.org/10.1109/OJSP.2023.3337718
28. Wu A, Nowozin S, Meeds E, Turner RE, Miguel Hernández-Lobato J, Gaunt AL, Deterministic variational inference for robust Bayesian neural networks
29. Haußmann M, Hamprecht FA, Kandemir M, Sampling-free variational inference of bayesian neural networks by variance backpropagation
30. Neal RM (1996) Bayesian learning for neural networks. Springer, New York, NY
31. Wu J, Leng C, Wang Y, Hu Q, Cheng J (2016) Quantized convolutional neural networks for mobile devices. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit 2016:4820–4828. https://doi.org/10.1109/CVPR.2016.521
32. Alqahtani A, Xie X, Jones MW (2021) Literature review of deep network compression. Informatics 8(4):1–12. https://doi.org/10.3390/informatics8040077
33. Kumar A, Yin B, Shaikh AM, Ali M, Wei W (2022) CorrNet: pearson correlation based pruning for efficient convolutional neural networks. Int J Mach Learn Cybern 13(12):3773–3783. https://doi.org/10.1007/s13042-022-01624-5
34. Han S, Mao H, Dally WJ (2015) Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding. pp 1–14, 2015, abs/1510.00149/1510.00149
35. Liu Z, Sun M, Zhou T, Huang G, Darrell T (2019) Rethinking the value of network pruning. In: 7th Int Conf Learn Represent ICLR 2019, pp 1–21, 2019
36. Wen W, Wu C, Wang Y, Chen Y, Li H (2016) Learning structured sparsity in deep neural networks. 2016, [Online]. Available: http://arxiv.org/abs/1608.03665
37. Frankle J, Dziugaite GK, Roy DM, Carbin M (2019) Stabilizing the lottery ticket hypothesis. [Online]. Available: http://arxiv.org/abs/1903.01611
38. Chen W, Wilson JT, Tyree S, Weinberger KQ, Chen Y (2015) Compressing neural networks with the hashing trick. [Online]. Available: http://arxiv.org/abs/1504.04788
39. He Y, Liu P, Wang Z, Hu Z, Yang Y (2019) Filter pruning via geometric median for deep convolutional neural networks acceleration. Proc IEEE Comput Soc Conf Comput Vis Pattern Recogn. https://doi.org/10.1109/CVPR.2019.00447
40. He Z, Qian Y, Wang Y, Wang B., Guan X, Gu Z, Zhou W (2022) Filter pruning via feature discrimination in deep neural networks. In: European conference on computer vision (pp 245–261). Cham: Springer Nature Switzerland
41. Shridhar K, Laumann F, Maurin AL, Olsen M, Liwicki M (2018) Bayesian convolutional neural networks with variational inference. arXiv:1806.05978 [cs.LG]
42. Kullback S, Leibler RA (1951) On information and sufficiency. Ann Math Stat. https://doi.org/10.1214/aoms/1177729694
43. Yedidia JS, Freeman WT, Weiss Y (2005) Constructing free-energy approximations and generalized belief propagation algorithms. IEEE Trans Inf Theory. https://doi.org/10.1109/TIT.2005.850085
44. Neal RM, Hinton GE (1998) A view of the Em algorithm that justifies incremental, sparse, and other variants. Learn Graph Models. https://doi.org/10.1007/978-94-011-5014-9_12

45. Kingma DP, Salimans T, Welling M (2015) Variational dropout and the local reparameterization trick. In: Advances in neural information processing systems
46. Kingma DP, Ba JL (2015) Adam: a method for stochastic optimization. In: 3rd international conference on learning representations, ICLR 2015—conference track proceedings
47. Tuv E, Borisov A, Runger G, Torkkola K (2009) Feature selection with ensembles, artificial variables, and redundancy elimination. J Mach Learn Res
48. Aketi SA, Roy S, Raghunathan A, Roy K (2020) Gradual channel pruning while training using feature relevance scores for convolutional neural networks. IEEE Access 8:171924–171932. https://doi.org/10.1109/ACCESS.2020.3024992
49. Yan Z, Xing P, Wang Y, Tian Y (2020) Prune it yourself: automated pruning by multiple level sensitivity. In: 2020 IEEE Conference Multimedia Information Processing Retrievel pp 73–78, 2020
50. Kwon Y, Won JH, Kim BJ, Paik MC (2020) Uncertainty quantification using Bayesian neural networks in classification: application to biomedical image segmentation. Comput Stat Data Anal 142:106816. https://doi.org/10.1016/j.csda.2019.106816
51. Hafner D, Tran D, Lillicrap T, Irpan A, Davidson J (2018) Noise contrastive priors for functional uncertainty. 2018, [Online]. Available: http://arxiv.org/abs/1807.09289
52. Goodfellow IJ et al (2024) Generative adversarial networks. Sci Robot 3:2672–2680