Batch-Shaped Channel Gated Networks

Babak Ehteshami Bejnordi **Tijmen Blankevoort** **Max Welling**
Qualcomm AI Research*
Qualcomm Technologies Netherlands B.V.
{behtesha, tijmen, mwelling}@qti.qualcomm.com

Abstract

We present a method for gating deep-learning architectures on a fine-grained level. Individual convolutional maps are turned on/off conditionally on features in the network. This method allows us to train neural networks with a large capacity, but lower inference time than the full network. To achieve this, we introduce a new residual block architecture that gates convolutional channels in a fine-grained manner. We also introduce a generally applicable tool *batch-shaping* that matches the marginal aggregate posteriors of features in a neural network to a pre-specified prior distribution. We use this novel technique to force gates to be more conditional on the data. We present results on CIFAR-10 and ImageNet datasets for image classification and Cityscapes for semantic segmentation. Our results show that our method can slim down large architectures conditionally, such that the average computational cost on the data is on par with a smaller architecture, but with higher accuracy. In particular, our ResNet34 gated network achieves a performance of 72.55% top-1 accuracy compared to the 69.76% accuracy of the baseline ResNet18 model, for similar complexity. We also show that the resulting networks automatically learn to use more features for difficult examples and fewer features for simple examples.

1 Introduction

Almost all deep neural networks have a prior that seems suboptimal: All features are calculated all the time. Both from a generalization and inference-time perspective, this is superfluous. For example, there is no reason to compute features that help differentiate between several dog breeds, if there is no dog to be seen in the image. We can make use of this natural prior information that the necessity of certain features for classification performance depends on other features, to improve our neural networks. We can also exploit this to spend less computational power on simple and more on complicated examples.

This general idea is commonly encapsulated in the terms *conditional computing* [2] or gating architectures [34]. It is known that models with increased capacity, for example increased model depth [10] or width [41], generally help increase model performance when properly regularized. However, as models increase in size, so do their training and inference times. This often limits practical applications of deep learning. By conditionally turning parts of the architecture on or off we can train networks with very large capacity, while keeping the run-time overhead small. The hypothesis is that when training conditionally gated networks, we can train models with a better accuracy/inference time trade-off than their fully feed-forward counterparts.

Another major advantage of gating networks is that they are more amenable to continual learning settings. Neural networks exhibit catastrophic forgetting [26] when batches of data are fed to the algorithm in specific sequence instead of randomly. Models quickly forget categories that were

*Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

Preprint. Under review.

previously learned if new batches contain no examples of that category. This makes it impossible to have a single network perform multi-task learning, where data is fed continually. There is reason to believe that architectures with gating properties help counteract this behavior, as shown in [36, 25, 32]. This is natural as some features used in tasks that are not currently learned can turn off and prevent updating from happening. In a sense, the parts of the network that are off function as a memory for previous tasks.

There are several works in recent literature that successfully learn conditional architectures, for example ConvNet-AIG [38] and Gao et al. [9]. However, the conditionality is often very coarse as in ConvNet-AIG [38], or the amount of actual conditional features learned is very minimal as in Gaternet [4]. We attempt to solve both. Our contributions are as follows:

- We propose a fine-grained gating architecture that turns individual input and output convolutional maps on or off, leading to features that are individually gated. This allows for a better trade-off between inference time and accuracy than previous work.
- We propose a general tool dubbed *batch-shaping* that matches the marginal aggregated posterior of a feature in the network to a specified prior. Depending on the chosen prior, networks can match activation distributions to e.g. the uniform distribution for better quantization, or the Gaussian to enforce behavior similar to batch-normalization [15]. Specifically in this paper, we apply batch-shaping to help the network learn conditional features. We show that this helps performance by controlling the gates to be more conditional on the input data at the end of training.
- We show state-of-the-art results compared to other conditional computing architectures such as Convnet-AIG [38], SkipNet [39] and soft-guided adaptively-dropped neural network [40].

2 Background and related work

Literature on gating connections for deep neural networks dates back to Hinton [12]. Gating can be seen as a tri-way connection in a neural network [7], where one output can only be 0 and 1. These connections have originally been used to learn transformations between images with gated Restricted Boltzmann Machines as in Memisevic and Hinton [27]. One of the earliest works to apply this to create sparse network architectures is that of a Mixture of Experts as in Jacobs et al. [17].

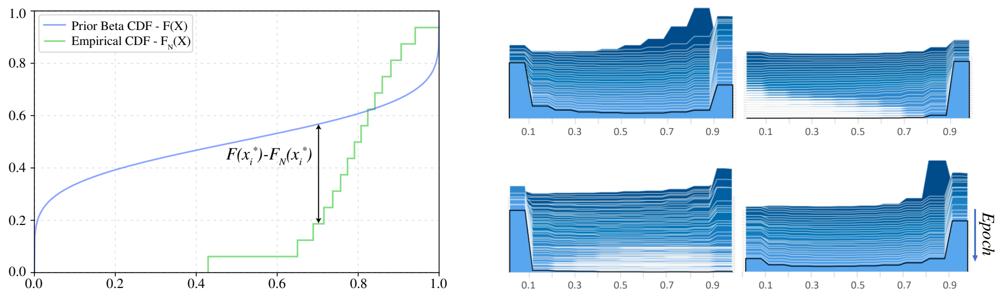
Several compression methods exist that reduce model complexity in a static way. Tensor factorization methods [18, 42] decompose single layers into two more efficient bottleneck layers. Methods such as channel-pruning [11, 28] remove entire input/output channels from the network. Similarly, full channels can be removed during training as in VIBnets [6], Bayesian Compression [22] and L0-regularization [23]. These methods reduce the overall model capacity while keeping the accuracy as high as possible. Our method allows for higher model capacity, while keeping inference times similar to the papers cited above.

Some networks exploit the complexity of each input example to gain performance. Networks such as Branchynet [37] and Multi-scale dense net [14] have early exiting nodes, where complex examples proceed deeper in the network than simpler ones. Our approach also assigns less computation to simpler examples than more complicated examples, but has no early-exiting paths. Both methods of inducing less computation can work in tandem, which we leave for future work.

Other works exploit similar conditional sparsity properties of networks as this work. ConvNet-AIG [38] and SkipNet [39] turn full residual blocks on or off, conditionally dependent on the input. Dynamic Channel Pruning [9] turns individual features on or off similar to our approach, but they choose the top-k features instead, akin to Outrageously large neural networks [33]. This approach loses the benefit of being able to trade off compute for simple and complex examples. Gaternet [4] trains a completely separate network to gate each individual channel of the main network. The overhead of this network is not necessary for learning effective gates, and we show better conditionality of the gates than is achieved by this paper, at almost no overhead.

3 Batch-shaping

First, we introduce batch-shaping, a general method to match the marginal aggregated posterior distribution over any feature in the neural network to a pre-specified prior distribution. In the next



(a) (b)

Figure 1: Batch-shaping loss. (a) Illustration of the computation of the batch-shaping loss. (b) The output distribution for four gates and their shapes over different epochs. We can see that initially gates are firing in a conditional pattern, but after removing the shaping loss and introducing the L_0-loss they may become fully active, stay conditional or turn off completely.

paragraph we will use this to train gates that fire more conditionally, but we see large potential value of this tool for many other applications such as training auto-encoders or quantization. Consider a parameterized feature in a neural network $X(\theta)$. The intention is to have $X(\theta)$ distributed more according to a chosen probability density function (PDF) $f(x)$, while still being able to differentiate with respect to θ. To do this we consider the Cramér-von-Mises criterion [1], which is a statistical distance between the cumulative distribution function $F(x)$ and an empirical cumulative distribution function $F_N(x)$. The Cramér-von-Mises criterion lends itself naturally to this use-case. Other frequently used statistical distance functions, such as KL-divergence, require calculating a histogram of the samples to compare to $f(x)$, which does not allow for gradients to propagate. As we will see, we can derive gradients with respect to each sample x with the proposed loss function. The Cramér-von-Mises criterion is given by:

$$\omega^2 = \int_{-\infty}^{\infty} \left[F_N(x) - F(x)\right]^2 \mathrm{d}F(x) \tag{1}$$

We consider batches of N samples $x_{1:N}$ drawn from $X(\theta)$. In order to optimize this we follow Anderson [1]. Sorting $sort(x_{1:N}) = x^*_{1:N}$, replacing $\mathrm{d}F(x)$ with $\mathrm{d}F_N(x)$ and normalizing with N gives us the batch-shaping loss to minimize

$$S(x^*, \lambda) = \frac{\lambda}{N} \sum_{i=1}^{N} \left[\frac{i}{N+1} - F(x^*_i)\right]^2, \tag{2}$$

where λ is a parameter that controls the strength of the loss function. This approach is shown visually in Figure 1a. We can sum this loss for each considered feature in the network to attain a full network batch-shaping loss. Note that we can differentiate $x^*_{1:N}$ with respect to θ through the sorting operator by keeping the sorted indices. In the backward pass, if a value with index i was sorted to index j, we simply put the error from position j in position i. This makes the whole loss term differentiable as long as the chosen CDF function is differentiable.

We can use this loss to match the marginal aggregated posterior of a feature in the network to any PDF. For example, if we want to encourage our activations to be Gaussian, we could use the CDF of the Gaussian in the loss function. This could be useful for purposes similar to batch-normalization. Or the CDF can be that of a uniform distribution, which could help with fixed-point quantization [16]. We leave this for future work, and only consider a loss function that encourages conditionality of gates in the next paragraph.

4 Channel gated networks

In this section we introduce our gating network. While the basic structure of our gating network could be any kind of CNN structure, we use ResNet [10] as the basic structure. Figure 2 shows an overview of a channel gated ResNet block. Formally, a ResNet building block is defined as:

$$x_{l+1} = r(F(x_l) + x_l), \tag{3}$$

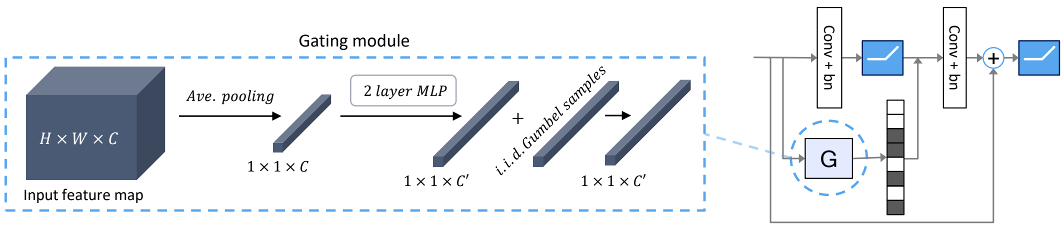


Figure 2: Illustration of our channel gated ResNet block and the gating module.

where $x_l \in R^{c^l \times w^l \times h^l}$, and $x_{l+1} \in R^{c^{l+1} \times w^{l+1} \times h^{l+1}}$ denote the input and output of the residual block, and r is the activation function, in this case a ReLU [29] function. The residual function $F(x_l)$ is the residual mapping to be learned and is defined as $F = W_2 * r(W_1 * x)$. Here $*$ denotes the convolution operator. $W_1 \in R^{c^l \times {c_1}^{l+1} \times k \times k}$ is a set of ${c_1}^{l+1}$ filters, with each filter of size $k \times k$. Similarly, $W_2 \in R^{{c_1}^{l+1} \times c^{l+1} \times k \times k}$. After each convolution layer, batch normalization [15] is used. Our gated residual block is defined as:

$$x_{l+1} = r(W_2 * (G(x_l) \cdot r(W_1 * x_l)) + x_l), \tag{4}$$

where G is a gating module and $G(x_l) = [g_1, g_2, g_3, \cdots, g_{{c_1}^{l+1}}]$ is the output of the gating function, where $g_c \in \{0, 1\}$: 0 denotes skipping the convolution operation for filter c in W_1, and 1 denotes computing the convolution. Here $\cdot$ refers to channel-wise multiplication between the output feature map $r(W_1 * x)$ and the vector $G(x_l)$. The more sparse the output of $G(x_l)$, the more computation we can potentially save.

The position of the gate was chosen for two specific reasons. Firstly, the gate is applied *after* the ReLU activation. This prevents the convolution from updating if the gate is off. Placing the gating function before the ReLU caused unstable training behavior. Secondly, we only gate the representation between the two layers in the residual block. We allow each block to use the full input, and update the full output. The network only gates each individual feature that determines how to update the incoming representation. We tested applying multiple gating setups, including before and after each convolutional layer. The proposed setup performed significantly better.

4.1 Gating module

To enable a light-weight gating module design we squeeze global spatial information in x_l into a channel descriptor as input to our gating module, similar to ConvNet-AIG and Squeeze-and-excitiation nets [38, 13]. This is achieved via channel-wise global average pooling.

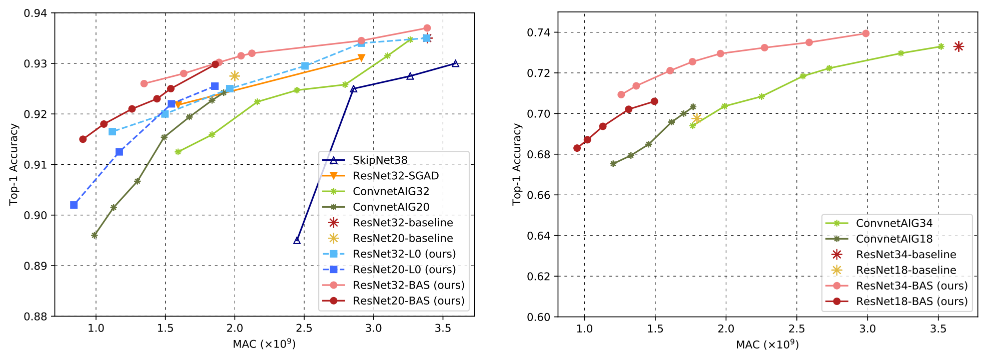
For our gating module, we use a simple feed-forward design comprising of two fully connected layers, with only 16 neurons in the hidden layer. We apply batch normalization and ReLU on the output of the first fully connected layer. The second fully connected layer linearly projects the features to (unnormalized) log-probabilities π_k, $k \in \{1, 2, \cdots, {c_1}^{l+1}\}$.

Our gating module is computationally inexpensive, and has an additional overhead that is between $0.018\% - 0.087\%$ of a ResNet block multiply-accumulate (MAC) usage.

To dynamically select a subset of filters relevant for our current input, we need to map the output of our gating module to a binary vector. The task of training binary valued gates is challenging because we can not directly back-propagate through a non-differentiable gate. In this paper, we leverage a recently proposed approach called Gumbel-Softmax sampling [19, 24] to circumvent this problem. We consider the binary case of the Gumbel-Max trick, the Binary concrete relaxation $BinConcrete(\pi, \tau)$. In the forward pass we use the discrete argmax and for the backward pass we use a sigmoid function with temperature: $\sigma_\tau(x) = \sigma(\frac{x}{\tau})$. We use $\tau = 2/3$ in all of our experiments as suggested by Maddison et al. [24].

4.2 Batch-shaping beta distribution prior for conditional gates

When initially training the channel-wise gating architecture, many features were trained to be only on or off in the first few epochs. Instead, we would like a feature to be sometimes on and sometimes off



(a) Results on CIFAR-10

(b) Results on ImageNet

Figure 3: Comparison of the results of our algorithm and competing methods on CIFAR-10 and ImageNet datasets.

for different data points, to exploit the potential for conditionality. We regulate this by applying the batch-shaping loss with the CDF of a Beta distribution $I_x(a, b)$ as a prior on each of the gates.

We set $a = 0.6$ and $b = 0.4$ in our experiments, initially inducing 40% sparsity. The Beta distribution will regularize gates towards being sometimes on and sometimes off for different data points, pushing the gates towards the desired batch-wise conditionality. We apply this loss at the start of training to encourage activations to be conditional, and gradually anneal λ to 0. Figure 1b presents the output distribution of four gates during training.

4.3 L_0-loss

Our batch-shaping loss encourages the network to learn more conditional features. But, our actual intent is to find a model that has the best trade-off between (conditional) sparsity and our task loss (e.g. cross-entropy loss). For the second part of our training procedure, we add a loss that regularizes the complexity of the full network explicitly. We use a method proposed by Louizos et al. [23], which defines an L_0 regularization process for neural network sparsification by learning a set of gates with a sparsifying regularizer. Our work can be considered as a conditional version of the L_0 gates introduced in this paper, sans stretching parameters. Hence this loss term is a natural choice for sparsifying the activations. We use a modified version of the L_0-loss without stretching, defined as:

$$L_C = \gamma \sum_{i=1}^{k} \sigma(ln(\pi_i)), \tag{5}$$

where k is the total number of gates, σ is the sigmoid function, and γ is a parameter that controls the level of sparsification we want to achieve.

It is important to mention that introducing this loss too early in training can reduce effective network capacity and potentially hurt performance. If the training procedure is not carefully chosen, the procedure often degenerates into training a smaller architecture, as full convolutional maps are turned off prematurely. Thus, in all our experiments, we introduce this L_0-loss after some delay, and we use a warm-up schedule as described in Sønderby et al. [35]

5 Experiments

We evaluate the performance of our method on two image classification benchmarks: CIFAR-10 [20] and ImageNet [31]. We additionally report preliminary results on the Cityscapes semantic segmentation benchmark [5]. For CIFAR-10, we use ResNet20 and ResNet32 architectures [10] as our base model. For ImageNet, we use ResNet18 and ResNet34. We compare our algorithm with competitive conditional computation methods. We additionally perform experiments to understand the learning patterns of the gates and whether they specialize to certain categories. For semantic segmentation, we employ the pyramid scene parsing network (PSPNet) [43] with ResNet-50 backbone.

5.1 CIFAR-10

We trained all models using Nesterov's accelerated gradient descent [30] with a momentum of 0.9 and weight decay factor of $5e^{-4}$. No weight decay was applied on the parameters of the gating modules. We used a standard data-augmentation scheme by randomly cropping and horizontally flipping the images [21]. We trained the models for 500 epochs with a mini-batch of 256. The initial learning rate was 0.1 and it was divided by 10 at epoch 300, 375, and 450. The base networks and the gates were trained together from scratch. We applied the batch-shaping loss with beta distribution prior from the start of training with a coefficient of $\lambda = 0.75$ and linearly annealed it to zero until epoch 100. Next, the L_0-loss was applied to the output of the gates starting from epoch 100 and the coefficient was linearly increased until epoch 300 where it was kept fixed for the rest of the training. For the L_0-loss we used γ values of $\{1, 2, 5, 10, 15, 20\} \cdot 10^{-2}$ to generate different trade-off points.

We compare our batch-shaped channel gated ResNet20 and ResNet32 models, hereafter referred to as ResNet20-BAS and ResNet32-BAS, with other adaptive computation methods: ConvNetAIG [38], SkipNet [39], and SGAD [40]. As shown in Figure 3a, ResNet20-BAS and ResNet32-BAS outperform SkipNet38, SGAD-ResNet32 and ConvNetAIG variants of ResNet18 and ResNet32 by a large margin. Our results show that given a deep network such as ResNet32, we can reduce the average computation (conditioned on the input) to a value equal or lower than that of a ResNet20 architecture and still achieve better performance. Ideally, a gated ResNet32 model should outperform a gated ResNet20 model at the same average computation. This property is evident in our results. However, the competing methods show performance equal to or lower than their smaller sized counterparts, indicating one may instead train a smaller model from scratch.

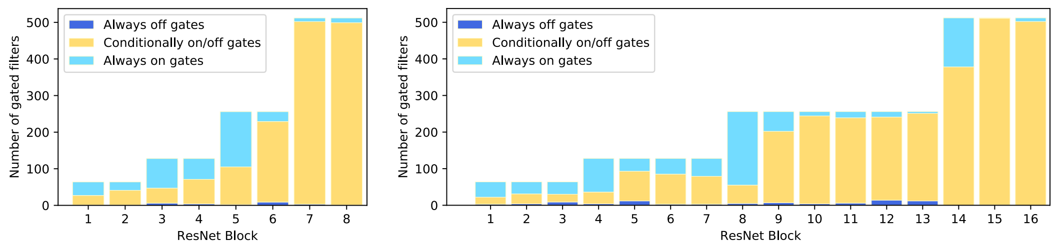
5.2 ImageNet

To evaluate the performance of our models on a larger dataset, we applied our gating networks to ImageNet. We used similar optimization settings to CIFAR-10 with a weight decay factor of $1e^{-4}$. We used a standard data-augmentation scheme adopted from He et al. [10] and trained the model for 150 epochs with a mini-batch size of 256. All models were trained using a single GPU. The initial learning rate of 0.1 was divided by 10 at epoch 60, 90, and 120. Similar to CIFAR classification we introduce the batch-shaping loss from the start of the training with $\lambda = 0.75$ and linearly annealed it to zero until epoch 20. L_0-loss was then applied to the output of the gates starting from epoch 30 and the coefficient was linearly increased until epoch 60 where it was kept fixed for the rest of the training. We used γ values of $\{1, 2, 5, 10, 15, 20, 30, 40\} \cdot 10^{-2}$ to generate different trade-off points.

Compared to CIFAR-10, the performance difference with the baseline is larger for ImageNet, likely because of the larger complexity of the dataset allowing for more conditionality of the features. We also see increased performance for lower 'compression rates', similar to what is frequently seen in compression literature because of extra regularization, e.g. as in Frankle and Carbin [8].

Figure 3ba shows the trade-off between the computation and Top-1 accuracy for our gated network and ConvNetAIG. The results indicate that our ResNet18-BAS and ResNet34-BAS models consistently outperform corresponding ConvNetAIG18 and ConvNetAIG34 models. Similar to the observations on the CIFAR-10 dataset, the performance of ConvNetAIG34 degrades to a level lower than that of ConvNetAIG18 at the same computation cost. Our models, in contrast, make a better use of dynamic allocation of features by learning more conditional features. Subsequently, when the average computation cost is on par with ResNet18, our ResNet34-BAS gated network achieves a performance as high as 72.55% top-1 accuracy compared to the 70.57% best accuracy of ResNet18-BAS and 69.76% accuracy of the baseline ResNet18 model. Our resulting gated networks automatically learn to use more features for difficult examples, and fewer features for simple examples in the dataset.

Cityscapes [5] is a dataset for semantic urban scene understanding including 5,000 images with high quality pixel-level annotations and 20,000 additional images with coarse annotations. There are 19 semantic classes for evaluation of semantic segmentation models in this benchmark. For data augmentation, we adopt random mirror and resize with a factor between 0.5 and 2 and use a crop-size of 448×672 for training. We train and test with only single-scale input and run inference on the whole image. The PSPNet network with the ResNet-50 back-end was trained from scratch with a mini-batch size of 6 for 150k iterations. Momentum and weight decay are set to 0.9 and $1e^{-4}$ respectively. For the learning rate we use similar policy to [3] where the initial learning rate of $2e - 2$ is multiplied by $(1 - {iter_{current}}/{iter_{max}})^{0.9}$. We used the same settings for training our gated PSPNet.



(a) Gated ResNet-18

(b) Gated ResNet-34

Figure 4: The distribution of different gate activation patterns in our ResNet18-BAS and ResNet34-BAS models trained on ImageNet. For 65% and 60% sparsity respectively. Gates are categorized as always on/off, if they are on/off for more than 99% of the inputs.

Weight decay for the layers in the gating units was set to $1e^{-6}$. We only used our batch-shaping loss with a fixed coefficient in this experiment.

The original PSP network achieves an overall IoU (intersection over Union) of 0.706 with a pixel-level accuracy of 0.929 on the validation set. Our gated PSPNet model was able to achieve an IoU of 0.719 and pixel accuracy of 0.935 while using 76.3% of the MAC count ($\lambda = 0.2$) of the original PSP model. We additionally compared the models when started training using ImageNet-pretrained weights to initialize the ResNet-50 base network. In this setting, PSPNet achieved an IoU of 0.739 and pixel accuracy of 0.9446. Our gated-model, in comparison, obtained an IoU of 0.744 and pixel accuracy of 0.946 using 76.5% of the PSPNet MAC count ($\lambda = 0.2$). The performance of our gated network further reaches an IoU of 0.747 and pixel accuracy of 0.948 using 95% of the PSPNet MAC count ($\lambda = 0.05$).

5.3 Effect of the batch-shaping loss

To validate the effectiveness of our proposed batch-shaping loss, we compare the performance of our ResNet20-BAS and ResNet32-BAS networks on the CIFAR10 classification in two settings: 1) using both the batch-shaping loss and the L_0 complexity loss for training the model similar to the experiments above, or 2) only using the L_0 complexity loss.

As can be seen in Figure 3a, the models that additionally use the batch-shaping loss consistently outperform the ones using only the L_0 complexity loss. ResNet20-L0 and ResNet32-L0 appear to have more rapid accuracy degradation than the models trained using the batch-shaping loss. However, our L_0-gated models still outperform ConvNetAIG and SkipNet architectures. This can be attributed to our specific channel gated network architecture which allows for fine-grained dynamic selection of channels or filters in a layer, as compared to these models which are designed to skip whole layers. Very importantly, as shown in Figure 3a, by gating the larger non-BAS ResNet32/38 models, we do not see any improvement over the smaller ResNet20 model at a similar computation cost. This is in sharp contrast to our ResNet34-BAS model. We observe a similar pattern in ImageNet results (See Figure 3b).

5.4 Gate distribution

Analyzing the distribution of the learned gates gives us a better insight into the characteristics of the learned features of the network. Ideally, we expect three main gate distributions to appear in a network with dynamic computations: 1) Gates that are always on. We expect certain filters in a network to be of key importance for all types of inputs. 2) Gates that fire conditionally on/off based on the input. The filters that are more input dependent and hence are more specialized for certain categories can be dynamically selected to be executed based on the input. These type of filters are desirable as they contribute to saving computation at the inference phase and formation of conditional expert sub-networks inside our network. Therefore, we would like to maximize the learning of such filters in our network. 3) Gates that are always off. This introduces complete sparsity.

Figure 4 shows the distribution of gates on the ImageNet validation set for our ResNet18-BAS and ResNet34-BAS models. We see that the majority of the gates are activated conditionally. Despite regularizing with L0-regularization, conditional sparsity is preferred over fully turning gates off.

6 Conclusion

In this paper, we presented a fine-grained gating architecture that enables conditional computation in deep networks. Our gating network achieves state-of-the-art accuracy among competing conditional computation architectures on CIFAR10 and ImageNet datasets. In both datasets, given a model with large capacity, our gating method was the only approach that could reduce the inference computation to a value equal or lower than that of a lower capacity base network, while obtaining a higher accuracy. On ImageNet, our ResNet34-BAS improves the accuracy by more than 2.8% over a ResNet18 at the same computation cost.

We also proposed a novel batch-shaping loss that can match the marginal aggregated posterior of a feature in the network to any prior PDF. We use it to enforce each gate in the network to be more conditionally activated at the start of training and improve performance significantly. We look forward to see many novel applications for this loss in the future, for e.g. autoencoders, better quantized models and as an alternative to batch-normalization. Another important future research direction that can benefit from our fine-grained gating architecture is continual learning. Designing gating mechanisms that can dynamically decide to allow or prevent the flow of gradients through certain parts of the network could potentially mitigate catastrophic forgetting. Finally, with our gating setup we could distill smaller sub-networks that work on only a subset of the trained classes.

References

[1] T. W. Anderson. On the distribution of the two-sample cramér-von mises criterion. *The Annals of Mathematical Statistics*, 33(3):1148–1159, 1962.

[2] Yoshua Bengio. Deep learning of representations: Looking forward. In *Proceedings of the First International Conference on Statistical Language and Speech Processing*, SLSP'13, pages 1–37, Berlin, Heidelberg, 2013. Springer-Verlag.

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[4] Zhourong Chen, Yang Li, Samy Bengio, and Si Si. Gaternet: Dynamic filter selection in convolutional neural network via a dedicated global gating network. *arXiv preprint arXiv:1811.11205*, 2018.

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[6] Bin Dai, Chen Zhu, Baining Guo, and David P. Wipf. Compressing neural networks using the variational information bottleneck. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 1143–1152, 2018.

[7] Alain Droniou, Serena Ivaldi, and Olivier Sigaud. Deep unsupervised network for multimodal perception, representation and classification. *Robot. Auton. Syst.*, 71(C):83–98, September 2015.

[8] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

[9] Xitong Gao, Yiren Zhao, Lukasz Dudziak, Robert Mullins, and Cheng-Zhong Xu. Dynamic channel pruning: Feature boosting and suppression. *arXiv preprint arxiv:1810.05331*, 2018.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778, 2016.

[11] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 1398–1406, 2017.

[12] Geoffrey F. Hinton. A parallel computation that assigns canonical object-based frames of reference. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 683–685, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[13] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[14] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations*, 2018.

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, pages 448–456. JMLR. org, 2015.

[16] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[17] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Comput.*, 3(1):79–87, March 1991.

[18] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *CoRR*, abs/1405.3866, 2014.

[19] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparametrization with gumbel-softmax. In *Proceedings International Conference on Learning Representations 2017*. OpenReviews.net, April 2017.

[20] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[21] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[22] Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3288–3298. Curran Associates, Inc., 2017.

[23] Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l0 regularization. In *International Conference on Learning Representations*, 2018.

[24] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[25] Nicolas Y. Masse, Gregory D. Grant, and David J. Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences*, 115(44):E10467–E10475, 2018.

[26] Michael Mccloskey and Neil J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation*, 24:104–169, 1989.

[27] Roland Memisevic and Geoffrey Hinton. Unsupervised learning of image transformations. pages 1–8, 07 2007.

[28] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *ICLR*, 2017.

[29] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[30] Yurii E Nesterov. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547, 1983.

[31] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[32] Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. *CoRR*, abs/1801.01423, 2018.

[33] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[34] Olivier Sigaud, Clément Masson, David Filliat, and Freek Stulp. Gated networks: an inventory. Unpublished manuscript, 17 pages, May 2016.

[35] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. How to train deep variational autoencoders and probabilistic ladder networks. In *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*, 2016.

[36] Rupesh K Srivastava, Jonathan Masci, Sohrob Kazerounian, Faustino Gomez, and Jürgen Schmidhuber. Compete to compute. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2310–2318. Curran Associates, Inc., 2013.

[37] S. Teerapittayanon, B. McDanel, and H. T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469, Dec 2016.

[38] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. 2018.

[39] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E. Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*, pages 420–436, 2018.

[40] Zhisheng Wang, Fangxuan Sun, Jun Lin, Zhongfeng Wang, and Bo Yuan. Sgad: Soft-guided adaptively-dropped neural network. *arXiv preprint arXiv:1807.01430*, 2018.

[41] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, September 2016.

[42] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(10):1943–1955, 2016.

[43] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.