SEP-Nets: Small and Effective Pattern Networks

Zhe Li†,‡, Xiaoyu Wang†, Xutao Lv† and Tianbao Yang‡

†Snap Research, ‡The University of Iowa

Thursday 24th August, 2017
1 Motivation

2 The Proposed Method

3 The Ingredients for SEP-Nets
   • Pattern Residient Block
   • SEP-Net Module
   • Group-wise Convolution

4 The Proposed SEP-Nets

5 Experimental Results

6 Conclusion
1 Motivation

2 The Proposed Method

3 The Ingredients for SEP-Nets
   • Pattern Residient Block
   • SEP-Net Module
   • Group-wise Convolution

4 The Proposed SEP-Nets

5 Experimental Results

6 Conclusion
The success of deep learning

Accuracy on Image Classification

Shallow Learning
Deep learning

Year
2010 2011 2012 2013 2014 2015
Accuracy(%)
71.8 74.2 83.6 88.3 93.3 96.43
Three aspects of deep learning

- Performance (Test accuracy): Almost Done
Three aspects of deep learning

- Performance (Test accuracy): Almost Done
- Computation (Number of floating point operations): Not Yet
Three aspects of deep learning

- Performance (Test accuracy): Almost Done
- Computation (Number of floating point operations): Not Yet
- Memory (Number of parameters): Not Yet
Let’s see performance and memory

![Accuracy vs Model Size for Different Models](image)

- GoogleNet
- ResNet-101
- AlexNet
- VGG-Net
What’s wrong?

Not affordable for large neural network models

- Mobile device

Highly require small and effective neural networks
What’s wrong?

Not affordable for large neural network models

- Mobile device
- Embedded device

Highly require small and effective neural networks
Outline

1 Motivation

2 The Proposed Method

3 The Ingredients for SEP-Nets
   - Pattern Residient Block
   - SEP-Net Module
   - Group-wise Convolution

4 The Proposed SEP-Nets

5 Experimental Results

6 Conclusion
Where to start?

Let’s review the most successful neural network structures:

- **AlexNet**: 2012
- **VGG-Net**: 2013
- **Inception-Net**: 2014
- **Res-Net**: 2015

- Fully connected layers and convolution layers have most parameters in neural network models.

**Focus on convolutional layers**
Where to start?

Let’s review the most successful neural network structures:

- Fully connected layers and convolution layers have most parameters in neural network models.
- Fully connected layers have been removed in modern deep CNN (Inception-Net, ResNets)

**Focus on convolutional layers**
Zoom in convolutional layers
Zoom in convolutional layers
Zoom in convolutional layers
Zoom in convolutional layers

- $7 \times 7$, $5 \times 5$, $3 \times 3$ filters
Zoom in convolutional layers

- $7 \times 7, 5 \times 5, 3 \times 3$ filters
- $1 \times 1$ filters
Pattern Binarization

- \( k \times k (k > 1) \) filters serve as spatial pattern extraction.

A trained 3 \( \times \) 3 filter from GoogleNet (Left) and its binarized version (right)
**Pattern Binarization**

- $k \times k (k > 1)$ filters serve as spatial pattern extraction.
- $1 \times 1$ filters serve as data transformation.

![Pattern Binarization Example](image)

A trained $3 \times 3$ filter from GoogleNet (Left) and its binarized version (right)
Pattern Binarization

- $k \times k (k > 1)$ filters serve as spatial pattern extraction.
- $1 \times 1$ filters serve as data transformation.
- Reduced number of parameters in model dramatically.

A trained $3 \times 3$ filter from GoogleNet (Left) and its binarized version (right).
How to use Pattern Binarization?

Easily adopted to any successful networks structures such as GoogleNet, ResNet including the designed SEP-Nets as following procedure:

- Train a full neural network such as GoogleNet, ResNet and SEP-Net from scratch
How to use Pattern Binarization?

Easily adopted to any successful networks structures such as GoogleNet, ResNet including the designed SEP-Nets as following procedure:

- Train a full neural network such as GoogleNet, ResNet and SEP-Net from scratch
- Binarize $k \times k (k > 1)$ convolutional filters in the well-trained neural network model
How to use Pattern Binarization?

Easily adopted to any successful networks structures such as GoogleNet, ResNet including the designed SEP-Nets as following procedure:

1. Train a full neural network such as GoogleNet, ResNet and SEP-Net from scratch
2. Binarize $k \times k (k > 1)$ convolutional filters in the well-trained neural network model
3. Fine-tune the scaling factors of all binarized $k \times k$ filters and the floating point representation of all $1 \times 1$ filters by back-propagation on the same dataset.
Outline

1. Motivation
2. The Proposed Method
3. The Ingredients for SEP-Nets
   - Pattern Residient Block
   - SEP-Net Module
   - Group-wise Convolution
4. The Proposed SEP-Nets
5. Experimental Results
6. Conclusion
Pattern Residient Block

Consists of $1 \times 1$ and $k \times k$ convolutions, which are executed in parallel and their feature map are added together.

Additive $1 \times 1$ convolutions act the residual between fully $3 \times 3$ filters maps and binarized $3 \times 3$ filtered maps.
Pattern Residient Block

Consists of $1 \times 1$ and $k \times k$ convolutions, which are executed in parallel and their feature map are added together.

Additive $1 \times 1$ convolutions act the residual between fully $3 \times 3$ filters maps and binarized $3 \times 3$ filtered maps.
Pattern Residient Block

- Consists of $1 \times 1$ and $k \times k$ convolutions, which are executed in parallel and their feature map are added together.
Pattern Residient Block

- Consists of $1 \times 1$ and $k \times k$ convolutions, which are executed in parallel and their feature map are added together.

- Additive $1 \times 1$ convolutions act the residual between fully $3 \times 3$ filters maps and binarized $3 \times 3$ filtered maps.
Pattern Resident Block

- Consists of $1 \times 1$ and $k \times k$ convolutions, which are executed in parallel and their feature map are added together.

- Additive $1 \times 1$ convolutions act the residual between fully $3 \times 3$ filters maps and binarized $3 \times 3$ filtered maps.
SEP-Net Module
SEP-Net Module

Pattern Residient Block
SEP-Net Module
Group-wise Convolution

Outline
Motivation
The Proposed Method
The Ingredients for SEP-Nets
The Proposed SEP-Nets
Experimental Results
Conclusion

SEP-Net Module

1 × 1 convolution layer: dimension reduction
2 PRB blocks with different output channels
1 × 1 convolution layer: dimension recovery
Skip connection
1 × 1 convolution layer: dimension reduction
SEP-Net Module

- $1 \times 1$ convolution layer: dimension reduction
- 2 PRB blocks with different output channels
SEP-Net Module

- 1 × 1 convolution layer: dimension reduction
- 2 PRB blocks with different output channels
- 1 × 1 convolution layer: dimension recovery
SEP-Net Module

- $1 \times 1$ convolution layer: dimension reduction
- 2 PRB blocks with different output channels
- $1 \times 1$ convolution layer: dimension recovery
- Skip connection
SEP-Net Module

- $1 \times 1$ convolution layer: dimension reduction
- 2 PRB blocks with different output channels
- $1 \times 1$ convolution layer: dimension recovery
- Skip connection
Group-wise Convolution

Adopt group convolution to reduce the model size. Split the input feature maps into $N$ groups and apply convolution to each group. Set the group number as the number of input channels, which degenerates to depth-wise convolutions (used in Google's MobileNets).
Group-wise Convolution

Adopt group convolution to reduce the model size.
Split the input features maps into \( N \) groups and apply convolution to each group.
Set group number as the number of input channels, it degenerates to depth-wise convolutions (Used in Google's MobileNets).
Group-wise Convolution

- Adopt group convolution to reduce the model size.
Group-wise Convolution

- Adopt group convolution to reduce the model size.
- Split the input features maps into $N$ groups and apply convolution to each group.
Group-wise Convolution

- Adopt group convolution to reduce the model size.
- Split the input features maps into $N$ groups and apply convolution to each group.
- Set group number as the number of input channels, it degenerates to depth-wise convolutions (Used in Google's MobileNets)
Group-wise Convolution

- Adopt group convolution to reduce the model size.
- Split the input features maps into $N$ groups and apply convolution to each group.
- Set group number as the number of input channels, it degenerates to depth-wise convolutions (Used in Google’s MobileNets)
Outline

1 Motivation
2 The Proposed Method
3 The Ingredients for SEP-Nets
   - Pattern Residient Block
   - SEP-Net Module
   - Group-wise Convolution
4 The Proposed SEP-Nets
5 Experimental Results
6 Conclusion
Proposed two small SEP-Nets for mobile/embeded devices.
The Proposed SEP-Net structures

- Proposed two small SEP-Nets for mobile/embeded devices.
- One model has 1.3M parameters while the other 1.7M.
The Proposed SEP-Net structures

- Proposed two small SEP-Nets for mobile/embeded devices.
- One model has 1.3M parameters while the other 1.7M.
- Shared same following structure with slightly difference (group number, output dimension of the last convolution layer)
Outline

1. Motivation
2. The Proposed Method
3. The Ingredients for SEP-Nets
   - Pattern Residient Block
   - SEP-Net Module
   - Group-wise Convolution
4. The Proposed SEP-Nets
5. Experimental Results
6. Conclusion
Experimental Results

- Justify that pattern binarization can reduce number of parameters dramatically. *(Small)*
Experimental Results

- Justify that pattern binarization can reduce number of parameters dramatically. (Small)
  - CIFAR10 with ResNet-20, 34, 44, 50
Justify that pattern binarization can reduce number of parameters dramatically. (Small)

- CIFAR10 with ResNet-20, 34, 44, 50
- ImageNet with GoogleNet, Customize-Inception-Net
Experimental Results

- Justify that pattern binarization can reduce number of parameters dramatically. (Small)
  - CIFAR10 with ResNet-20, 34, 44, 50
  - ImageNet with GoogleNet, Costomize-Inception-Net

- Justify that fine-tuning other parameters of the binarized network with fixed binarized pattern could achieve comparable performance. (Effective)
Justify that pattern binarization can reduce number of parameters dramatically. (Small)

- CIFAR10 with ResNet-20, 34, 44, 50
- ImageNet with GoogleNet, Customized-Inception-Net

Justify that fine-tuning other parameters of the binarized network with fixed binarized pattern could achieve comparable performance. (Effective)

- same as the above setting.
Experimental Results

- Justify that pattern binarization can reduce number of parameters dramatically. (Small)
  - CIFAR10 with ResNet-20, 34, 44, 50
  - ImageNet with GoogleNet, Customize-Inception-Net

- Justify that fine-tuning other parameters of the binarized network with fixed binarized pattern could achieve comparable performance. (Effective)
  - same as the above setting.

- Show that the designed SEP-Net structures could achieve better or comparable performance on ImageNet than using similar sized networks such as MobileNet. (Small & Effective)
Experimental Results–Training strategy

CIFAR10

- Preprocessed by Global Contrast Normalization and ZCA whitening.
CIFAR10

- Preprocessed by Global Contrast Normalization and ZCA whitening.
- $32 \times 32$ crop is randomly sampled from the padded image.
Experimental Results—Training strategy

CIFAR10

- Preprocessed by Global Contrast Normalization and ZCA whitening.
- $32 \times 32$ crop is randomly sampled from the padded image.
- Initial learning rate is 0.1 and divided by 10 at iteration $32K$, $48K$. 
CIFAR10
- Preprocessed by Global Contrast Normalization and ZCA whitening.
- $32 \times 32$ crop is randomly sampled from the padded image.
- Initial learning rate is 0.1 and divided by 10 at iteration $32K$, $48K$.
- Maximum number of iteration is $64K$. 
Experimental Results—Training strategy

CIFAR10

- Preprocessed by Global Contrast Normalization and ZCA whitening.
- $32 \times 32$ crop is randomly sampled from the padded image.
- Initial learning rate is 0.1 and divided by 10 at iteration $32K$, $48K$.
- Maximum number of iteration is $64K$.
- The momentum is 0.9 and the weight decay is 0.0001.
Experimental Results–Training strategy

CIFAR10

- Preprocessed by Global Contrast Normalization and ZCA whitening.
- $32 \times 32$ crop is randomly sampled from the padded image.
- Initial learning rate is 0.1 and divided by 10 at iteration $32K$, $48K$.
- Maximum number of iteration is $64K$.
- The momentum is 0.9 and the weight decay is 0.0001.
- Train on one GPU using mini-batch SGD with a batch size 256.
### Experimental Results on Pattern Binarization

- **Effective** on CIFAR10

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc</th>
<th>Ref</th>
<th>Full</th>
<th>BiPattern</th>
<th>Refined</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-20</td>
<td>Top-1</td>
<td>0.9125</td>
<td>0.9118</td>
<td>0.1546</td>
<td>0.8649</td>
</tr>
<tr>
<td></td>
<td>Top-5</td>
<td>-</td>
<td>0.9974</td>
<td>0.5104</td>
<td>0.9941</td>
</tr>
<tr>
<td>ResNet-32</td>
<td>Top-1</td>
<td>0.9249</td>
<td>0.9276</td>
<td>0.2634</td>
<td>0.9021</td>
</tr>
<tr>
<td></td>
<td>Top-5</td>
<td>-</td>
<td>0.9972</td>
<td>0.6932</td>
<td>0.9962</td>
</tr>
<tr>
<td>ResNet-44</td>
<td>Top-1</td>
<td>0.9283</td>
<td>0.9283</td>
<td>0.4825</td>
<td>0.9145</td>
</tr>
<tr>
<td></td>
<td>Top-5</td>
<td>-</td>
<td>0.9982</td>
<td>0.8765</td>
<td>0.9965</td>
</tr>
<tr>
<td>ResNet-56</td>
<td>Top-1</td>
<td>0.9303</td>
<td>0.9375</td>
<td>0.5382</td>
<td>0.9302</td>
</tr>
<tr>
<td></td>
<td>Top-5</td>
<td>-</td>
<td>0.9977</td>
<td>0.9574</td>
<td>0.9971</td>
</tr>
</tbody>
</table>
Experimental Results on Pattern Binarization

- **Small** on CIFAR10.

<table>
<thead>
<tr>
<th>Model</th>
<th>Full Network</th>
<th>Pattern Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-20</td>
<td>292K</td>
<td>55K</td>
</tr>
<tr>
<td>ResNet-32</td>
<td>487K</td>
<td>78K</td>
</tr>
<tr>
<td>ResNet-44</td>
<td>682K</td>
<td>100K</td>
</tr>
<tr>
<td>ResNet-56</td>
<td>876K</td>
<td>123K</td>
</tr>
</tbody>
</table>
Experimental Results on Pattern Binarization

- **Small** on CIFAR10.

<table>
<thead>
<tr>
<th>Model</th>
<th>Full Network</th>
<th>Pattern Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-20</td>
<td>292K</td>
<td>55K</td>
</tr>
<tr>
<td>ResNet-32</td>
<td>487K</td>
<td>78K</td>
</tr>
<tr>
<td>ResNet-44</td>
<td>682K</td>
<td>100K</td>
</tr>
<tr>
<td>ResNet-56</td>
<td>876K</td>
<td>123K</td>
</tr>
</tbody>
</table>

- Use one number to represent a binarized $3 \times 3$. 
Experimental Results–Training strategy

ImageNet on GoogleNet
- Initial learning rate is 0.01 and follow a polynomial decay.

ImageNet on C-InceptionNet
Experimental Results—Training strategy

ImageNet on GoogleNet
- Initial learning rate is 0.01 and follow a polynomial decay.
- Maximum number of iteration is $600K$.

ImageNet on C-InceptionNet
Experimental Results–Training strategy

ImageNet on GoogleNet

- Initial learning rate is 0.01 and follow a polynomial decay.
- Maximum number of iteration is $600K$.
- The momentum is 0.9 and the weight decay is 0.0001.

ImageNet on C-InceptionNet
Experimental Results–Training strategy

ImageNet on GoogleNet

- Initial learning rate is 0.01 and follow a polynomial decay.
- Maximum number of iteration is 600K.
- The momentum is 0.9 and the weight decay is 0.0001.
- Train on one GPU using mini-batch SGD with a batch size 128.

ImageNet on C-InceptionNet
Experimental Results—Training strategy

ImageNet on GoogleNet
- Initial learning rate is 0.01 and follow a polynomial decay.
- Maximum number of iteration is 600K.
- The momentum is 0.9 and the weight decay is 0.0001.
- Train on one GPU using mini-batch SGD with a batch size 128.

ImageNet on C-InceptionNet
- Initial learning rate is 0.1 and divided learning rate 10 time after every 24 epochs.
Experimental Results—Training strategy

ImageNet on GoogleNet

- Initial learning rate is 0.01 and follow a polynomial decay.
- Maximum number of iteration is 600K.
- The momentum is 0.9 and the weight decay is 0.0001.
- Train on one GPU using mini-batch SGD with a batch size 128.

ImageNet on C-InceptionNet

- Initial learning rate is 0.1 and divided learning rate 10 times after every 24 epochs.
- Train total 90 epochs.
Experimental Results on Pattern Binarization

- **Effective on ImageNet**

<table>
<thead>
<tr>
<th>Model</th>
<th>Acc Top-1</th>
<th>Acc Top-5</th>
<th>Ref</th>
<th>Full</th>
<th>BiPattern</th>
<th>Ref</th>
<th>Full</th>
<th>Ref</th>
<th>Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>GoogLeNet</td>
<td>0.8993</td>
<td>0.8891</td>
<td>0.6865</td>
<td>0.8891</td>
<td>1x1 pattern: 0.0013</td>
<td>0.6117</td>
<td>0.636</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0075</td>
<td>0.8395</td>
<td>0.856</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2x8 3x3 pattern: 0.3706</td>
<td>0.6797</td>
<td>0.6893</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.6290</td>
<td>0.8827</td>
<td>0.8898</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5x5 pattern: 0.5141</td>
<td>0.6917</td>
<td>0.6984</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.7619</td>
<td>0.8904</td>
<td>0.8965</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3x3 &amp; 5x5 pattern: 0.1428</td>
<td>0.6694</td>
<td>0.6812</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.31738</td>
<td>0.8763</td>
<td>0.8844</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-InceptionNet</td>
<td>0.648</td>
<td>0.863</td>
<td>0.0476</td>
<td>0.1464</td>
<td>0.6400</td>
<td>0.6521</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8550</td>
<td>0.8626</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Experimental Results on Pattern Binarization

- **Small** on ImageNet

<table>
<thead>
<tr>
<th>Model</th>
<th>Full Network</th>
<th>Pattern Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>GoogLeNet</td>
<td>6.99M</td>
<td>3 × 3 4.43M</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 × 5 6.43M</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 × 3 and 5 × 5 3.87M</td>
</tr>
<tr>
<td>C-InceptionNet</td>
<td>5.10M</td>
<td>2.43M</td>
</tr>
</tbody>
</table>
Experimental Results on Pattern Binarization

- **Small** on ImageNet

<table>
<thead>
<tr>
<th>Model</th>
<th>Full Network</th>
<th>Pattern Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>GoogLeNet</td>
<td>6.99M</td>
<td>3 x 3 4.43M</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 x 5 6.43M</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 x 3 and 5 x 5 3.87M</td>
</tr>
<tr>
<td>C-InceptionNet</td>
<td>5.10M</td>
<td>2.43M</td>
</tr>
</tbody>
</table>

- Use one number to represent a 3 x 3 or 5 x 5 kernel.
**Experimental Results for the Designed SEP-Nets**

- **Small and Effective** on the designed SEP-Nets

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameter Number</th>
<th>Size (bytes)</th>
<th>Top-1 Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileNet</td>
<td>1.3M</td>
<td>5.2MB</td>
<td>0.637</td>
</tr>
<tr>
<td></td>
<td>2.6M</td>
<td>10.4MB</td>
<td>0.684</td>
</tr>
<tr>
<td>SEP-Net-R</td>
<td>1.3M (small)</td>
<td>5.2MB</td>
<td>0.658</td>
</tr>
<tr>
<td></td>
<td>1.7M (large)</td>
<td>6.7MB</td>
<td>0.667</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SqueezeNet</td>
<td>1.2M</td>
<td>4.8MB</td>
<td>0.604</td>
</tr>
<tr>
<td>MobileNet</td>
<td>1.3M</td>
<td>5.2MB</td>
<td>0.637</td>
</tr>
<tr>
<td>SEP-Net-R (small)</td>
<td><strong>1.3M</strong></td>
<td><strong>5.2MB</strong></td>
<td><strong>0.658</strong></td>
</tr>
<tr>
<td>SEP-Net-B (small)</td>
<td>1.1M</td>
<td>4.2MB</td>
<td>0.637</td>
</tr>
<tr>
<td>SEP-Net-BQ (small)</td>
<td>1.1M</td>
<td>1.3MB</td>
<td>0.635</td>
</tr>
</tbody>
</table>

SEP-Net-R: SEP-Net with raw valued weights  
SEP-Net-B: SEP-Net with pattern binarization  
SEP-Net-BQ: SEP-Net with pattern binarization and other weights quantized using linear quantization with 8 bits
Another Angle to Pattern Binarization

Analyze the effect of binarizing $1 \times 1$ filters and $k \times k$ filter from the view of quantization error:

- Let $W$ denote an $c \times k \times k$ convolutional filter.
Another Angle to Pattern Binarization

Analyse the effect of binarizing $1 \times 1$ filters and $k \times k$ filter from the view of quantization error:

- Let $W$ denote an $c \times k \times k$ convolutional filter.
- Binarization seeks to approximate it by $\alpha B$, where $B$ is a binary filter with entries from $\{1, -1\}$ and $\alpha$ is a scaling factor.
Another Angle to Pattern Binarization

Analyze the effect of binarizing $1 \times 1$ filters and $k \times k$ filter from the view of quantization error:

- Let $W$ denote an $c \times k \times k$ convolutional filter.
- Binarization seeks to approximate it by $\alpha B$, where $B$ is a binary filter with entries from $\{1, -1\}$ and $\alpha$ is a scaling factor.
- From the viewpoint of minimizing the quantization error, $\alpha, B$ can be sought by solving the following problem:

$$
\min_{\alpha \in \mathbb{R}, B \in \{1, -1\}^{c \times k \times k}} E(W, B, \alpha) \triangleq \|W - \alpha B\|^2_F \quad (1)
$$
Another Angle to Pattern Binarization

Analyze the effect of binarizing $1 \times 1$ filters and $k \times k$ filters from the view of quantization error:

- The optimal $B^*$ can be found by thresholding, i.e., $B_{i,j,l}^* = 1$ if $W_{i,j,l} \geq 0$ and $B_{i,j,l}^* = -1$ if $W_{i,j,l} < 0$.  

The optimal $\alpha^*$ can be computed by $\alpha^* = \sum_{i,j,l} |W_{i,j,l}| c \times k \times k$. 

To quantitatively understand the effect of binarizing $1 \times 1$ filters and $k \times k$ filters, we compute the quantization error for all filters in the well-trained GoogleNet and obtain averaged quantization error for different filters:

- $1 \times 1$: $0.0462$
- $3 \times 3$: $0.0029$
- $5 \times 5$: $0.0056$
Another Angle to Pattern Binarization

Analyze the effect of binarizing $1 \times 1$ filters and $k \times k$ filters from the view of quantization error:

- The optimal $B^*$ can be found by thresholding, i.e., $B_{i,j,l}^* = 1$ if $W_{i,j,l} \geq 0$ and $B_{i,j,l}^* = -1$ if $W_{i,j,l} < 0$.

- The optimal $\alpha^*$ can be computed by $\alpha^* = \frac{\sum_{i,j,l} |W_{i,j,l}|}{c \times k \times k}$. 
Another Angle to Pattern Binarization

Analyze the effect of binarizing $1 \times 1$ filters and $k \times k$ filters from the view of quantization error:

- The optimal $B^*$ can be found by thresholding, i.e., $B^*_{i,j,l} = 1$ if $W_{i,j,l} \geq 0$ and $B^*_{i,j,l} = -1$ if $W_{i,j,l} < 0$.

- The optimal $\alpha^*$ can be computed by $\alpha^* = \frac{\sum_{i,j,l} |W_{i,j,l}|}{c \times k \times k}$.

- To quantitatively understand the effect of binarizing $1 \times 1$ filters and $k \times k$ filters, we compute the quantization error for all filters in the well-trained GoogleNet and obtain averaged quantization error for different filters:

<table>
<thead>
<tr>
<th>Filter Size</th>
<th>Quantization Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 \times 1$</td>
<td>0.0462</td>
</tr>
<tr>
<td>$3 \times 3$</td>
<td>0.0029</td>
</tr>
<tr>
<td>$5 \times 5$</td>
<td>0.0056</td>
</tr>
</tbody>
</table>
Outline

1. Motivation
2. The Proposed Method
3. The Ingredients for SEP-Nets
   - Pattern Residient Block
   - SEP-Net Module
   - Group-wise Convolution
4. The Proposed SEP-Nets
5. Experimental Results
6. Conclusion
Conclusion

- Proposed pattern binarization method.
Conclusion

- Proposed pattern binarization method.
- Designed a new pattern residual block.
Conclusion

- Proposed pattern binarization method.
- Designed a new pattern residual block.
- Designed a novel SEP-Net Module.
Conclusion

- Proposed pattern binarization method.
- Designed a new pattern residual block.
- Designed a novel SEP-Net Module.
- Proposed Small and Effective Pattern Networks.
Conclusion

- Proposed pattern binarization method.
- Designed a new pattern residual block.
- Designed a novel SEP-Net Module.
- Proposed Small and Effective Pattern Networks.
- Achieved the-state-of-art performance.
future future following this work

- Train the pattern network with binarized $k \times k$ filters from scratch?
Future future following this work

- Train the pattern network with binarized $k \times k$ filters from scratch?
- Reduce computation cost (number of floating point operation)?
Question?