AsymmNet: Towards ultralight convolution neural networks using asymmetrical bottlenecks

Haojin Yang*
Hasso Plattner Institute
haojin.yang@hpi.de

Zhen Shen*
Alibaba Cloud
zackary.sz@alibaba-inc.com

Yucheng Zhao*
ByteDance
zhaoyucheng.joe@bytedance.com

Abstract

Deep convolutional neural networks (CNN) have achieved astonishing results in a large variety of applications. However, using these models on mobile or embedded devices is difficult due to the limited memory and computation resources. Recently, the inverted residual block becomes the dominating solution for the architecture design of compact CNNs. In this work, we comprehensively investigated the existing design concepts, rethink the functional characteristics of two pointwise convolutions in the inverted residuals. We propose a novel design, called asymmetrical bottlenecks. Precisely, we adjust the first pointwise convolution dimension, enrich the information flow by feature reuse, and migrate saved computations to the second pointwise convolution. Doing so we can further improve the accuracy without increasing the computation overhead. The asymmetrical bottlenecks can be adopted as a drop-in replacement for the existing CNN blocks. We can thus create AsymmNet by easily stack those blocks according to proper depth and width conditions. Extensive experiments demonstrate that our proposed block design is more beneficial than the original inverted residual bottlenecks for mobile networks, especially useful for those ultralight CNNs within the regime of <220M MAdds. Code is available at https://github.com/Spark001/AsymmNet

1. Introduction

The recent success of deep *Convolution Neural Networks* (CNN) is like the jewel in the crown of modern AI waves [11, 17, 37]. However, the current CNN models are heavily relying on high-performance computation hardware, such as GPU and TPU, which are normally deployed in a cloud computing environment. Thus, the client applications have to transmit user data to the cloud to gain deep CNN models' benefits. This constraint strongly limits such models' applicability on resource-constrained devices, e.g., mobile phones, IoT devices, and embedded devices. Moreover, sending user data to a remote server increases the risk of privacy leakage. Therefore, in recent years, various works aim to solve this problem by reducing memory footprints and accelerating inference. We roughly categorize those works into following directions: network pruning [15, 16], knowledge distillation [7, 33], low-bit quantization [6, 35], and compact network designs [21, 20, 38, 46, 31, 41]. The latter has been recognized as the most popular approach that has a massive impact on industrial applications. The compact networks achieved promising accuracy with generally fewer parameters and less computation. Although significant progress has been made, there is still much room for improvement, especially in the ultralight regime with multiply adds (MAdds) <220M.

In this paper, our efforts focus on improving ultralight CNNs by the handcrafted design of basic building blocks. We first made a thorough investigation on existing block designs of mobile CNNs, and we argue that the two pointwise (PW) convolutions contribute differently in the original inverted residual bottleneck block. [5] first proposes decoupling spatial correlation and channel correlation using the combination of a depthwise (DW) convolution and a PW convolution. [38] further emphasises that the second PW convolution has essential characteristics in the inverted residual bottlenecks since it is responsible for learning new features from different channels, which is especially crucial for expressiveness. The first PW convolution is responsible for channel expansion, on the other hand. Therefore, we propose to partially reduce or migrate the first PW computations to the second one. By following this idea, we introduce a novel *Asymmetrical Bottleneck Block*, as shown in Figure 2(c). Furthermore, we can create *AsymmNet* by easily stack a sequence of asymmetrical blocks according to proper depth and width conditions. We only consider the handcrafted design of the network architecture in this work. However, the proposed CNN block is orthogonal to the recent approaches based on *Neural Architecture Search* (NAS) [41, 20, 42, 34]. Experimental results show

*Equal contribution. This work is done when Haojin Yang and Yuncheng Zhao are with Alibaba Cloud.

that, AsymmNet is especially superior in the ultralight CNN regime, and we have achieved promising results in image classification and other four downstream vision tasks. Summarized our core contributions in this paper are:

- We thoroughly investigated the existing mobile CNN designs and further proposed a novel asymmetrical bottleneck block.
- We propose AsymmNet based on asymmetrical bottlenecks, which achieves promising performance under the ultralight CNN regime (<220M MAdds) on ImageNet classification and multiple downstream vision tasks.

The rest of the paper is organized as follows: Section 2 briefly review the related work. Subsequently, we present the proposed asymmetrical bottleneck design and *AsymmNet* in Section 3, followed by experimental results and discussions (Section 4). Finally, Section 5 concludes the paper and provides an outlook on future work.

2. Related Work

This section provides a thorough overview of the recent efforts in the research domain of model compression and compact network design.

In the model compression area, knowledge distillation [7, 33] aims to generate small "student" networks trained by using distilled supervision signals derived from a cumbersome "teacher" network. The student network is expected to be more compact and as accurate as of the teacher. Connection pruning [15, 16] and channel pruning [27, 19] respectively remove low-rank connections between neurons or weakly weighted channels for model shrinking and acceleration. Low-bit quantization [6, 35, 30, 29, 1] is another crucial complementary approach to improve network efficiency through reduced precision arithmetic or even bitwise (binary) operators. Among them, Bethge et al. [1] introduced a novel block design that suggests applying *DenseNet* [22] style concatenation for preserving a rich information flow and a subsequent improvement block to update the newly added features, which can reduce the computation overhead as well. Our approach is also partially inspired by this concept.

The compact network methods use full precision floating point numbers as weights but reduce the total number of parameters and operations through compact architecture design while minimizing accuracy loss. The commonly used techniques include replacing a large portion of 3×3 filters with smaller 1×1 filters [24]; Using depthwise separable convolution to reduce operations [5]; Utilizing channel shuffling and group convolutions in addition to depthwise convolution [46]. Among those approaches, the MobileNet series (V1-V3) [21, 38, 20] are so far the most successful lightweight CNN models based on depthwise separable convolution and intelligent architecture design. Specifically, MobileNetV3 combines handcrafted block design and architecture search techniques. GhostNet [14] adopts the network architecture of MobileNetV3 but proposed to use a computationally cheaper block design replacing the inverted bottleneck block. Zhou et al. [49] proposed a sandglass block to replace the commonly used inverted bottleneck block, whilst better accuracy can be achieved compared to MobileNetV2 without increasing parameters and computation.

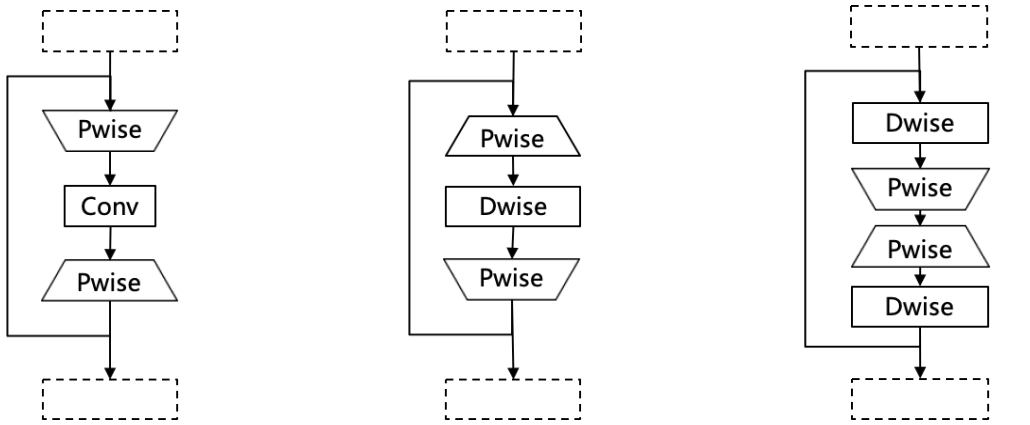
NAS techniques aim to automatically search efficient network architectures [41, 20, 42, 34]. However, the most efficient basic building block design still requires human expertise [20, 49, 14]. Furthermore, such methods need to repeat the network design process and retrain the network from scratch for each setting, which will result in excessive energy consumption and CO_2 emission. E.g., a *Transformer* language model [43] with NAS will cause CO_2 emission as much as 5 cars' lifetime [39].

3. Methodology

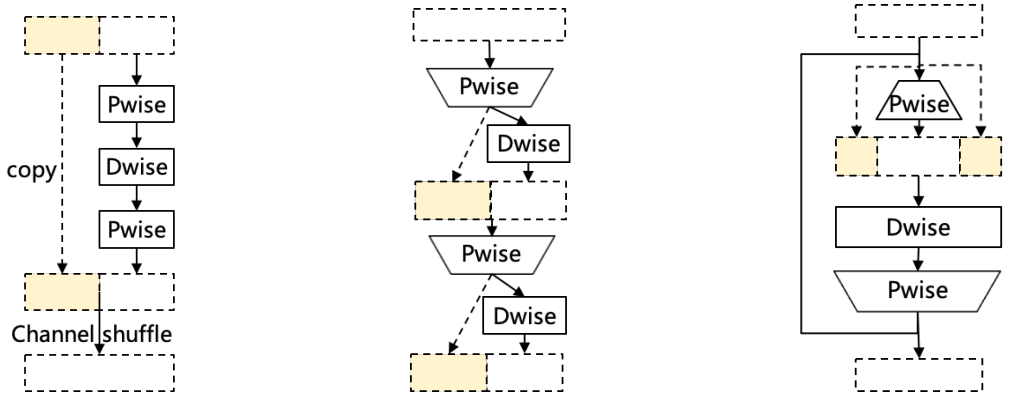
In this section, we first revisit the existing building block designs for lightweight CNN models. We then introduce the proposed *asymmetrical bottleneck block* and *AsymmNet*, discuss the design concept and main differences compared to the existing approaches.

3.1. Preliminaries

Depthwise separable convolution is proposed by Chollet in the *Xception* network [5], which is way more efficient than other CNN networks at that time. Subsequently, this design has been applied in many lightweight CNN architectures such as MobileNet series [21, 38, 20] and ShuffleNet series [31, 46]. It assumes that separately learning spatial and channel correlations should be more efficient and easier for a CNN learner. Specifically, it replaces a standard convolutional operator by splitting convolution into two separate operators (layers). The first one is called depthwise convolution, which adopts single channel filters to learn spatial correlations among locations within each channel separately. The second operator is a 1×1 convolution, also served as pointwise convolution, which is utilized for learning new features through computing linear combinations across all the input channels. According to [5, 21], depthwise separable convolution can reduce computation overhead by a factor of around $kernel\_size^2$ compared to a standard convolution operator with the same kernel size. We also apply depthwise separable convolution in the proposed approach due to its computational efficiency.



(a) Bottleneck block [17] (b) Inverted residual block [38] (c) Sandglass block [49]



(d) ShuffleBlock v2 [31] (e) Ghost module [14] (f) Asymmetrical bottleneck

Figure 1. Different types of basic convolution blocks. "Pwise" denotes 1×1 pointwise convolution, "Dwise" denotes 3×3 depthwise convolution. The dotted rectangles and arrows represent feature maps and feature reuse, respectively.

Inverted bottleneck is proposed by Sandler et al. for MobileNetV2 [38]. Unlike the original bottleneck design [40] (see Figure 1(a)), the inverted bottleneck block adopts a low-dimensional (the number of input channels) input tensor and expands it to a higher dimensional tensor using a pointwise convolution. The expanded high-dimensional tensor will then be fed into a depthwise separable convolution, by which the corresponding pointwise convolution generates low-dimensional new features by linearly combining channels after a depthwise convolution. It can be seen that the first pointwise convolution expands the information flow, which increases the capacity, and the subsequent convolution operators are responsible for the expressiveness of the proper layer. This speculation is derived based on the analysis of the block's capacity and expressiveness in [38]. Figure 1(b) shows the design idea of an inverted bottleneck block.

Cheap operations for more features Table 1 shows the complexity evaluation results of different types of convolutions of MobileNet series. We observe that the computation

Network	DW (%)	PW (%)	Vanilla (%)
MobileNetV1	3.1	95	1.9
MobileNetV2	6.2	84.4	9.4
MobileNetV3	8.9	88.5	2.6

Table 1. Computational complexity distribution of MobileNet V1-V3. We calculate the proportion of MAdds of different types of operators. DW and PW respectively represent the depthwise and pointwise convolution in the corresponding MobileNet blocks. Vanilla denotes the computation of the remaining standard convolutions.

overhead is mainly concentrated on the pointwise convolution part, as e.g., 95% of MobileNetV1, 84.4% of MobileNetV2 and 88.5% of MobileNetV3. If we want to reduce the computational complexity further, the optimization of this part of the network is the first choice. Han et al. proposed to use the Ghost module (see Section 3.2) to replace the pointwise convolution layers and partially remove the depthwise convolution layers (only preserve those for downsampling). The core idea of the Ghost module is to

generate more features using computationally cheaper operators. Our proposed design is also partially inspired by this concept, and we assume that the two pointwise convolutions contribute differently in the structural point of view. We thus shift the amount of calculation to the more important one.

3.2. Revisit existing design

In this section, we review several commonly used design principles: original bottleneck block [17], inverted residual block [38], shuffleblock v2 [31], Ghost module [14], and Sandglass block [49]. Figure 1 demonstrates the specific properties of each design.

Bottleneck (Figure 1(a)) is the fundamental building block of ResNet [17], where a pointwise convolution reduces the feature dimension with the factor t, apply a 3×3 convolution to the narrowed features, and then utilize another pointwise convolution to restore the feature dimension to be equal to the input size. The key difference between Inverted Residual block (also called MMBlock, see Figure 1(b)) and original Bottleneck is that the latter applies standard convolution on narrowed features, while MMBlock uses the first pointwise convolution to expand the feature dimension, and applies depthwise convolution on expanded features. It is so because standard 3×3 convolutions are highly computational intensive in Bottlenecks. However, depthwise convolutions in MMBlock can significantly reduce the computational complexity. Thus, increasing the feature dimension will be beneficial for improving the representative capacity of the block.

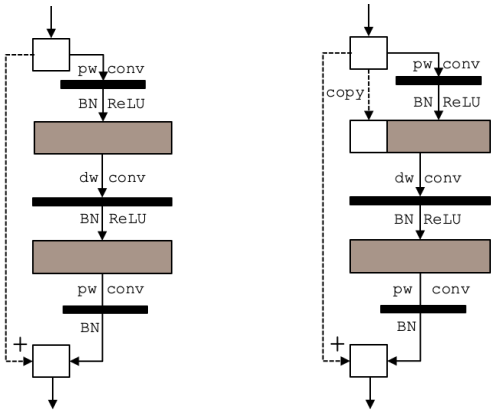
Shuffleblock v2 is the following work of Shuffleblock v1 [46], in which the group convolution is removed for practical efficiency. Furthermore, the input feature map in Shuffleblock v2 is split into two equal channels of narrowed feature maps (Figure 1(d)). One is transformed with a special Bottleneck block without internal dimension changes (the solid arrow path on the right); The other (the dashed arrow path on the left) keeps unchanged until concatenated and shuffled with the transformed feature map. This design reveals that partially reusing the input features doesn't impair the expressiveness of the convolution blocks, but can effectively reduce computational complexity.

Ghost module (Figure 1(e)) is proposed to reduce redundancy of feature maps generated by pointwise convolutions in an MMBlock. Specifically, the amount of output channels of pointwise convolutions is reduced to make more room for integrating cheaper intrinsic features. To keep the output dimension consistent, a series of linear transformation such as depthwise convolutions is used for generating intrinsic features, which will be concatenated with the output of the pointwise convolution to form the final feature vector.

[49] propposed Sandglass block (see Figure 1(c)), which suggests keeping the standard bottleneck structure. It prefers to perform identity mapping and spatial transformation at a higher dimension to alleviate information loss and gradient confusion. Therefore, the sandglass block flips the position of depthwise and pointwise convolutions, which aims to preserve dense information flow and suppress the computation cost.

Based on the previous findings, we argue that improving the capacity by using cheaper intrinsic features or even directly feature reuse is beneficial. We thus rethink the functional characteristics of two pointwise convolutions in the inverted residuals and propose a novel asymmetrical bottleneck described in the next section.

3.3. Asymmetrical bottlenecks



(a) Inverted residual block (b) Pruned block

(c) Asymmetrical block

Figure 2. Detailed illustration of the inverted residual block, pruned block, and asymmetrical bottleneck block. Brown fillings represent the feature maps generated by convolutions, while white fillings denote feature map reuse.

As demonstrated in Table 1, pointwise (PW) convolution is the most computationally intensive part in inverted residual bottlenecks (see Figure 2(a)). The first PW convo-

Input	Operator	Output
$h \times w \times c$	1×1,conv2d,non-linear	$h \times w \times (t-r)c$
$h \times w \times (t-r)c$	Concat	$h \times w \times (t+r)c$
$h \times w \times (t+r)c$	DW(k) s=s,non-linear	$\frac{h}{s} \times \frac{w}{s} \times (t+r)c$
$\frac{h}{s} \times \frac{w}{s} \times (t+r)c$	1×1,conv2d,linear	$\frac{h}{s} \times \frac{w}{s} \times c$

Table 2. Asymmetrical bottleneck block with stride s, asymmetry rate r, and expansion factor t.

lution is adopted to expand the feature tensor's dimension and the second one is significant for learning feature correlations from different channels after the depthwise convolution (DW). We can figure out that the first PW expands the information flow, which increases the capacity, and the second PW convolution is mainly responsible for the expressiveness. We argue that cheaper transformations or even feature reuse can enhance the information flow, but learning channel correlations should not be simplified in relative terms. Therefore, we infer that the second PW has more essential characteristics in the structure. To verify our speculation, we first designed a pruned version (referred to as pruned block subsequently) based on inverted residual bottlenecks, as shown by Figure 2(b). The output of the first PW is expressed by Eq. 1:

$$Y_{pw1} = Concat(X, Y_{t-1}(X)) \tag{1}$$

where $X \in \mathbb{R}^{h \times w \times c}$ denotes the input tensor, while h, w, and c denote the height, width, and channel dimension, respectively. $Y_{t-1} \in \mathbb{R}^{h \times w \times (t-1) \cdot c}$ is the output of the first PW, where t is an expansion factor introduced in [38]. In the pruned block, we reduce the output channels of the first PW by c. Thus, the pruned block can be formulated as:

$$Y_p = X + PW(DW(Concat(X, Y_{t-1}(X)))) \tag{2}$$

where we omit ReLU and BatchNorm for simplicity in the formulation.

Pruned block can save computation, but at the same time, it also brings a small amount of accuracy loss. Therefore, we consider migrating the saved computation of the first PW to the second PW to construct an asymmetrical structure, as shown in Figure 2(c). Experimental results show that the performance can be improved with this asymmetrical structure while the computation amount is basically unchanged. Mathematically, the asymmetrical bottleneck block can be expressed by Eq. 3:

$$Y' = X + PW(DW(Concat(2r \cdot X, Y_{t-r}(X)))) \tag{3}$$

where $Y_{t-r} \in \mathbb{R}^{h \times w \times (t-r) \cdot c}$ is the output of the first PW, t denotes the expansion factor and $r \in [0, t)$ is a new parameter which controls the asymmetry rate. To achieve a good trade-off between accuracy and efficiency, we set r to 1 in all experiments. If $r = 0$ it degenerates into an inverted residual bottleneck.

No.	**Input**	**Operator**	***k***	***p***	***c***	***s***
1	$224^2 \times 3$	conv2d	3	-	16	2
2	$112^2 \times 16$	asymm-bneck	3	16	16	1
3			3	64	24	2
4	$56^2 \times 24$	asymm-bneck	3	72	24	1
5			5	72	40	2
6	$28^2 \times 40$	asymm-bneck	5	120	40	1
7			5	120	40	1
8			3	240	80	2
9	$14^2 \times 80$	asymm-bneck	3	200	80	1
10			3	184	80	1
11			3	184	80	1
12			3	480	112	1
13	$14^2 \times 112$	asymm-bneck	3	672	112	1
14			5	672	160	2
15	$7^2 \times 160$	asymm-bneck	5	960	160	1
16			5	960	160	1
17	$7^2 \times 160$	conv2d	1	-	960	1
18	$7^2 \times 960$	avgpool	7	-	-	1
19	1×960	conv2d	1	-	1280	1
20	1×1280	conv2d	1	-	1000	1

Table 3. Specification for AsymmNet-L using MobileNetV3-large base. Each row shows a conv2d layer or an asymmetrical bottleneck block. c denotes the output channel size, k denotes the kernel size, and s is the stride number of the convolution layer. "Input" and "Operator" indicate the shape of the input tensor and the operator type. p denotes the expanded channel size of the corresponding asymmetrical bottleneck blocks.

3.3.1 Computational complexity

Similar to MMBlock, the theoretical computation complexity of AsymmBlock is $C = C_{pw1} + C_{dw} + C_{pw2}$. For simplicity, we only calculate the blocks whose $stride = 1$. So the theoretical complexity ratio of AsymmBlock and MMBlock can be calculated as

$$\begin{aligned} R_c &= \frac{hwc(tc - rc) + k^2 hw(tc + rc) + hw(tc + rc)c}{hwc(tc) + k^2 hw(tc) + hw(tc)c} \\ &= \frac{2hwtc^2 + k^2 hwtc + hwrck^2}{2hwtc^2 + k^2 hwtc} \\ &= 1 + \frac{rk^2}{2tc + k^2 t} \approx 1. \end{aligned} \tag{4}$$

where t denotes the expansion factor, r represents the asymmetry rate, and k indicates the kernel size of DW convolution. We set $r = 1$ in our experiments and $k^2 \ll c$. $R_c \approx 1$ means that the AsymmBlock can transfer the computation cost from the first PW to the second and keep total complexity roughly unchanged.

3.4. AsymmNet

We further develop several efficient CNN architectures based on the proposed asymmetrical bottleneck design. To gain the best practical benefits, we follow the basic network architecture of MobileNetV3-large and MobileNetV3-small [20], as shown in Table 3 and Table 4. The main reason for

No.	Input	Operator	*k*	*p*	*c*	*s*
1	$224^2 \times 3$	conv2d	3	-	16	2
2	$112^2 \times 16$	asymm-bneck	3	16	16	2
3	$56^2 \times 16$	asymm-bneck	3	72	24	2
4	$28^2 \times 24$	asymm-bneck	3	88	24	1
5			5	96	40	2
6	$14^2 \times 40$	asymm-bneck	5	240	40	1
7			5	240	40	1
8			5	120	48	1
9	$14^2 \times 48$	asymm-bneck	5	144	48	1
10			5	288	96	2
11	$7^2 \times 96$	asymm-bneck	5	576	96	1
12						
13	$7^2 \times 96$	conv2d	1	-	576	1
14	$7^2 \times 576$	avgpool	7	-	-	1
15	1×576	conv2d	1	-	1024	1
16	1×1024	conv2d	1	-	1000	1

Table 4. Specification for AsymmNet-S using MobileNetV3-small base.

our choice is that the core hyper-parameters such as kernel size, expand size, and network depth of MobileNetV3 are determined through a NAS algorithm and exhaustive search process. Our efforts thus mainly focus on the manual improvement of the basic block design. For a fair comparison, we keep those automatically selected hyper-parameters unchanged.

Therefore, the main building blocks of *AsymmNet* consists of a sequence of stacked asymmetrical bottleneck blocks, which gradually downsample the feature map resolution and increase the channel number to maintain the whole network's information capacity. We consider the presented architecture in this work as a basic design, while we believe that the automatic architecture and hyper-parameter search methods can further boost the performance.

4. Experiments

This section presents detailed experimental results. We first evaluate the model performance on the ImageNet classification task under various complexity (MAdds) settings. We further validate the generalization ability and effectiveness of the proposed approach on four downstream tasks, including face recognition, action recognition, pose estimation, and object detection.

4.1. Experiment setup

We utilize the deep learning framework MXNet [4] and the off-the-shelf toolbox Gluon-CV [18] to implement our models. We use the standard SGD optimizer for model training with both decay and momentum of 0.9 and the weight decay is 3e-5. We use the cosine learning rate scheduler with the initial learning rate of 2.6 for eight GPUs. The corresponding batch size was set to 256. Without special declaration, we train all the models for 360 epochs, in which five epochs are conducted for a warm-up phase. Detailed configurations can be found in our open-source codes [1].

4.2. Image classification

4.2.1 Compare to MobileNetV3

This section extensively studies the advantages of the proposed AsymmNet and pruned model over MobileNetV3 (MBV3) on the ImageNet ILSVRC 2012 dataset [8]. As shown in Table 5, we compare their performance under multiple complexity settings by tuning the weight multiplier. We consider both V3-large and V3-small architecture as references and comprehensively evaluated the classification accuracy, computation complexity (MAdds), and inference latency. Doing so can help reveal the performance advantage of the full spectrum of model architecture configurations. We applied the DNN inference toolkit MNN [25] for the latency evaluation since MNN is specifically optimized for mobile devices. All the inference tests have been done on an Android phone equipped with a Qualcomm snapdragon-855 CPU with 8G RAM in the single thread modus. We average the latency results of 1000 runs for each model. To conduct a fair comparison, we replace the corresponding convolution blocks and keep all the hyper-parameters unchanged.

We can figure out that, AsymmNet outperforms MobileNetV3 on classification accuracy in almost all the complexity settings, while the pruned model demonstrates better efficiency with slight accuracy drops. However, the accuracy loss becomes negligible when the MAdds getting smaller or even reversed, e.g., pruned model outperforms MobileNetV3 by 3.4% at the level of 0.35-Small. Specifically, when the model gets smaller and less complex, the accuracy advantage of AsymmNet becomes more apparent. This phenomenon effectively reveals the proposed asymmetrical bottleneck block's superiority in the spectrum of extremely lightweight models (a regime <220M MAdds).

4.2.2 Ablation study on asymmetry rate

We evaluate the asymmetry rate r on the ImageNet dataset to obtain the best choice in terms of accuracy and complexity. Table 6 shows the result, where $r = 1$ demonstrates the best trade-off.

4.3. Face recognition

Face recognition is a crucial identity authentication technology used in many mobile or embedded applications such as mobile payment and device unlock. In this subsection, we employ AsymmNet-L and the proposed AsymmNet-s as network backbone for face recognition. Following MobileFaceNet [3], we use a global depthwise convolution layer

[1] https://github.com/Spark001/AsymmNet

Multiplier	Model Scale	Networks	Top-1 Acc (%)	MAdds (M)	Params (M)	Latency (ms)
0.35	Large	AsymmNet	**65.4**	43	2.2	7.2
		Pruned	63.3	36.9	2.1	**5.2**
		MBV3	64.2	40	2.2	6.3
	Small	AsymmNet	**55**	15	1.7	3.3
		Pruned	53.2	13.6	1.7	**2.9**
		MBV3	49.8	12	1.4	3
0.5	Large	AsymmNet	**69.2**	67.2	2.8	10.2
		Pruned	68.3	59	2.6	**7.1**
		MBV3	68.8	69	2.6	8.8
	Small	AsymmNet	**58.9**	20.6	1.9	4.2
		Pruned	57.3	18.6	1.9	**3.6**
		MBV3	58	21	1.6	3.7
0.75	Large	AsymmNet	**73.5**	142.1	4.2	19.4
		Pruned	72.6	125.3	3.8	**13.6**
		MBV3	73.3	155	4	16.2
	Small	AsymmNet	**65.6**	40.8	2.5	6.9
		Pruned	64	36.9	2.3	**6.1**
		MBV3	65.4	44	2	6.3
1.0	Large	AsymmNet	**75.4**	216.9	5.99	27.1
		Pruned	74.9	193.6	5.3	**19.5**
		MBV3	75.2	216.5	5.4	23.3
	Small	AsymmNet	**68.4**	57.7	3.1	8.9
		Pruned	67	52.5	2.6	**7.9**
		MBV3	67.5	57	2.5	8.17
1.25	Large	AsymmNet	76.4	349.8	8.3	38.8
		Pruned	76.1	311	7.2	**29.2**
		MBV3	**76.6**	356	7.5	34.9
	Small	AsymmNet	**70.6**	91.7	3.9	12.7
		Pruned	69.8	83.1	3.5	**11.2**
		MBV3	70.4	91	3.6	11.7

Table 5. Performance comparison between AsymmNet, Pruned model, and MobileNetV3 across a large variety of scale levels. We consider both V3-Large and V3-small architecture as references and comprehensively evaluated the accuracy, computation complexity (MAdds), and inference efficiency. Top-1 accuracy is on the ImageNet dataset, and all latency are obtained by averaging the inference time of 1000 executions on a Qualcomm snapdragon-855 CPU with 8G RAM in the single thread modus. To conduct a fair comparison, we replace the corresponding convolution blocks and keep all the hyper-parameters unchanged.

Scale	r	Top-1 Acc (%)	MAdds (M)	Params (M)
Large	0	75.2	216.6	5.4
	1	**75.4**	216.9	5.9
	2	74.8	217.3	6.6
Small	0	67.4	56.9	2.9
	1	**68.4**	57.7	3.1
	2	68.0	58.5	3.3

Table 6. Ablation study on asymmetry rate r using ImageNet dataset. We apply both large and small AsymmNet for this evaluation. $r = 1$ shows the best performance.

rather than a global average pooling layer to output discriminative feature vectors. Both models are trained on MS-Celeb-1M [13] dataset from scratch by ArcFace [9] loss, for a fair comparison between them. The input image size is 112×112. We report result on different dataset including LFW [23], CALFW [48], CPLFW [47], AgeDB-30 [32], and VGGFace2 [2] as in Table 7. As shown in the table, our face recognition models with AsymmNet-s/L as backbone outperform MBV3-s/L consistently, especially in the CPLFW dataset, our models outperform by a margin of 1.5% and 1.9%, respectively. The computational complexity of our face recognition model uses AsymmNet-s as the backbone is about 21 times less than MobileFaceNet [3].

4.4. Action recognition

Action recognition has drawn a significant amount of attention from the academic community, owing to its applications in many areas like security and behavior analysis. We evaluate and compare AsymmNet and MobileNetV3

Backbone	LFW (%)	CALFW (%)	CPLFW (%)	AgeDB-30 (%)	VGGFace2 (%)	MAdds(M)	Params(M)
AsymmNet-s	**97.9**	**90.1**	**80.2**	**86.6**	**83.6**	10.3	0.3
MBV3-s	97.2	**90.1**	78.7	85.7	83.5	10.2	0.3
AsymmNet-L	**99.1**	93.8	**86.6**	**93.2**	**89.6**	41.6	1.1
MBV3-L	**99.1**	**94.0**	84.7	93.0	88.4	41.2	1.0
MobileFaceNet[3]	99.6	-	-	-	-	221	1.0

Table 7. Performance comparison among different face recognition datasets.

Backbone	Top1-Acc (%)	MAdds(M)	Params(M)
AsymmNet-s	**47.2**	56.5	1.9
MBV3-s	46.7	55.7	1.7
AsymmNet-L	**52.8**	215.7	4.8
MBV3-L	51.3	215.4	4.2
$ResNet50_{v1b}$[12]	55.2	4087.2	16.1

Table 8. Action recognition result on HMDB51 dataset.

Backbone	AP	AP_{50}	AP_{75}	MAdds(M)	Params(M)
AsymmNet-s	**54.4**	**84.4**	**59.7**	310.7	1.7
MBV3-s[12]	54.3	83.7	59.4	309.7	1.6
AsymmNet-L	63.5	**88.9**	**70.9**	523.9	4.3
MBV3-L[12]	**63.7**	**88.9**	70.8	523.6	3.7

Table 9. Pose estimation results on COCO human pose dataset using SimplePose method. All the AP results are in percentage.

as feature extractors for action recognition following [44] on the HMDB51 dataset [26]. The input image size is 224×224. As summarized in Table 8, AsymmNet achieves superior results compared to MobileNetV3 at both large and small scales. Furthermore, it reaches an accuracy close to ResNet50$_{v1b}$ [12], while its MAdds is about 19 times smaller.

4.5. Pose estimation

There has been significant progress in pose estimation and increasing interest in pose tracking in recent years. Thus, we evaluated AsymmNet as the backbone on this task by using the challenging COCO human pose benchmark [28]. Our approach is based on the *SimplePose* model [45], which estimates heat maps from deep and low-resolution feature maps. We replace the backbone with mobile CNN models and evaluate the accuracy for a fair comparison. The input image size is 256×192. The test set results are given in Table 9. Our pose estimation results with AsymmNet-s backbone surpass MBV3-s in all three metrics.

4.6. Object detection

We apply AsymmNet as a drop-in replacement for the original backbone in YOLO-v3 detector [36]. We compare our results to MobileNetV3 on the PASCAL VOC dataset [10] on object detection. We change the base model of the adopted YOLO-v3 architecture and train our models on the combination of VOC2007 trainval and VOC2012 trainval, test on VOC2007 test set. The input image size is 416×416. Table 10 illustrates the results compared to other models based on MobileNet backbones. AsymmNet-s outperforms MBV3-s on this task.

Backbone	mAP (%)	MAdds(G)	Params(M)
AsymmNet-s	**69.80**	7.99	9.33
MBV3-s	68.98	7.99	9.16
AsymmNet-L	76.18	8.58	11.97
MBV3-L	**76.64**	8.58	11.39
MobileNetV1[12]	75.8	9.92	11.83

Table 10. Object detection results on the PASCAL VOC dataset using YOLO-v3 detector.

5. Conclusion and Discussion

In this paper, we introduced a novel design for ultralight CNN models. We investigated important design choices, redesigned two pointwise convolutions of the inverted residual block, and developed a novel asymmetrical bottleneck. We can see that AsymmNet has a consistent accuracy advantage over MobileNetV3 in the ultralight model regime through our experiments on a series of downstream tasks. It thus can be used as a practical complementary approach to existing state-of-the-art CNN models in the regime of $<$220M MAdds.

However, we observed that AsymmBlock also has its limitations. For instance, with the increase of MAdds, it can not continue to show the advantage of accuracy, as the comparison result with MobileNetV3-1.25. Also, the AsymmNet-L model does not demonstrate benefits in the object detection task. One possible explanation is that the current AsymmNet architecture is based on MBV3, which is searched using MMBlock that not necessarily the most suitable architecture for AsymmNet. Thus, we will continue to optimize it in future work. As the next step, we will combine automatic search techniques with asymmetrical bottlenecks.

6. Acknowledgment

We would like to thank Chao Qian for his valuable technical support on inference speed evaluation.

References

[1] Joseph Bethge, Christian Bartz, Haojin Yang, and Christoph Meinel. Meliusnet: Can binary neural networks achieve mobilenet-level accuracy? *arXiv preprint arXiv:2001.05936*, 2020. 2

[2] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 67–74. IEEE, 2018. 7

[3] Sheng Chen, Yang Liu, Xiang Gao, and Zhen Han. Mobilefacenets: Efficient cnns for accurate real-time face verification on mobile devices. In *Chinese Conference on Biometric Recognition*, pages 428–438. Springer, 2018. 6, 7, 8

[4] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015. 6

[5] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017. 1, 2

[6] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015. 1, 2

[7] Elliot J Crowley, Gavin Gray, and Amos J Storkey. Moonshine: Distilling with cheap convolutions. In *Advances in Neural Information Processing Systems*, pages 2888–2898, 2018. 1, 2

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[9] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019. 7

[10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 8

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 1

[12] Jian Guo, He He, Tong He, Leonard Lausen, Mu Li, Haibin Lin, Xingjian Shi, Chenguang Wang, Junyuan Xie, Sheng Zha, et al. Gluoncv and gluonnlp: Deep learning in computer vision and natural language processing. *Journal of Machine Learning Research*, 21(23):1–7, 2020. 8

[13] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *European conference on computer vision*, pages 87–102. Springer, 2016. 7

[14] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1580–1589, 2020. 2, 3, 4

[15] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 1, 2

[16] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015. 1, 2

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 3, 4

[18] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019. 6

[19] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017. 2

[20] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1314–1324, 2019. 1, 2, 5

[21] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1, 2

[22] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 2

[23] Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. 2008. 7

[24] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 2

[25] Xiaotang Jiang, Huan Wang, Yiliu Chen, Ziqi Wu, Lichuan Wang, Bin Zou, Yafeng Yang, Zongyang Cui, Yu Cai, Tianhang Yu, Chengfei Lv, and Zhihua Wu. Mnn: A universal and efficient inference engine. In *MLSys*, 2020. 6

[26] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: A large video database for human motion recognition. In *IEEE International Conference on Computer Vision*, 2011. 8

[27] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016. 2

[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 8

[29] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. *arXiv preprint arXiv:2003.03488*, 2020. 2

[30] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proceedings of the European conference on computer vision (ECCV)*, pages 722–737, 2018. 2

[31] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018. 1, 2, 3, 4

[32] Stylianos Moschoglou, Athanasios Papaioannou, Christos Sagonas, Jiankang Deng, Irene Kotsia, and Stefanos Zafeiriou. Agedb: the first manually collected, in-the-wild age database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 51–59, 2017. 7

[33] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018. 1, 2

[34] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020. 1, 2

[35] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016. 1, 2

[36] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 8

[37] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1

[38] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 1, 2, 3, 4, 5

[39] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019. 2

[40] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 3

[41] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019. 1, 2

[42] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 1, 2

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 2

[44] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, pages 20–36. Springer, 2016. 8

[45] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 466–481, 2018. 8

[46] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018. 1, 2, 4

[47] Tianyue Zheng and Weihong Deng. Cross-pose lfw: A database for studying cross-pose face recognition in unconstrained environments. *Beijing University of Posts and Telecommunications, Tech. Rep*, 5, 2018. 7

[48] Tianyue Zheng, Weihong Deng, and Jiani Hu. Cross-age lfw: A database for studying cross-age face recognition in unconstrained environments. *arXiv preprint arXiv:1708.08197*, 2017. 7

[49] Daquan Zhou, Qibin Hou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. Rethinking bottleneck structure for efficient mobile network design. *ECCV, August*, 2020. 2, 3, 4