BasisNet: Two-stage Model Synthesis for Efficient Inference

Mingda Zhang[1*] Chun-Te Chu[2] Andrey Zhmoginov[2] Andrew Howard[2] Brendan Jou[2]
Yukun Zhu[2] Li Zhang[2] Rebecca Hwa[1] Adriana Kovashka[1]

[1]Department of Computer Science, University of Pittsburgh [2]Google Research
{mzhang,hwa,kovashka}@cs.pitt.edu {ctchu,azhmogin,howarda,bjou,yukun,zhl}@google.com

Abstract

In this work, we present BasisNet which combines recent advancements in efficient neural network architectures, conditional computation, and early termination in a simple new form. Our approach incorporates a lightweight model to preview the input and generate input-dependent combination coefficients, which later controls the synthesis of a more accurate specialist model to make final prediction. The two-stage model synthesis strategy can be applied to any network architectures and both stages are jointly trained. We also show that proper training recipes are critical for increasing generalizability for such high capacity neural networks. On ImageNet classification benchmark, our BasisNet with MobileNets as backbone demonstrated clear advantage on accuracy-efficiency trade-off over several strong baselines. Specifically, BasisNet-MobileNetV3 obtained 80.3% top-1 accuracy with only 290M Multiply-Add operations, halving the computational cost of previous state-of-the-art without sacrificing accuracy. With early termination, the average cost can be further reduced to 198M MAdds while maintaining accuracy of 80.0% on ImageNet.

1. Introduction

High-accuracy yet low-latency convolutional neural networks enable opportunities for on-device machine learning, and are playing increasingly important roles in various mobile applications, including but not limited to intelligent personal assistants, AR/VR, and real-time speech translations. Therefore designing efficient convolutional neural networks especially for edge devices has received significant research attention. Prior research attempted to tackle this challenge from different perspectives, such as novel network architectures [19, 35, 12], better incorporation with hardware accelerators [15], or conditional computation and adaptive inference algorithms [1, 8, 30]. However, focusing on one perspective *in isolation* may have side effects. For example, novel network architectures may introduce custom operators that are not well-supported by hardware accelerators, thus a promising new model may have limited practical improvements on real devices due to a lack of hardware support. We believe that these perspectives should be better integrated to form a more holistic general approach for broader applicability.

	MAdds (FLOPs)	Top-1 Acc. (%)
MobileNetV2 1.0x [19]	300M	72.0
CondConv-MobileNetV2 1.0x [33]♠	329M	74.6
DY-MobileNetV2 1.0x [4]♠	313M	75.2
MobileNetV3-Large [12]	219M	75.2
Dy-MobileNetV3-Large [36]	228M	77.1
ShuffleNetV2 1.5x [16]	299M	72.6
EfficientNet-B0 [22]	390M	77.1
EfficientNet-B0 (Noisy Stdt.) [32]♦♥	390M	78.1
EfficientNet-B0 (AA + KD) [31]♦♠	390M	78.0
CondConv-EfficientNet-B0 [33]♠	413M	78.3
ProxylessNas [3]	320M	74.6
FBNetV2-L1 [28]	325M	77.2
FBNetV3-A [7]♣	343M	78.0
MnasNet-A1 [21]	312M	75.2
CondConv-MnasNet-A1 [33]♠	325M	76.2
EfficientNet-B2 [22]	1.0B	80.1
EfficientNet-B1 (Noisy Stdt.) [32]♦♥	700M	80.2
FBNetV3-E [7]♣	752M	80.4
OFA [2]♦	595M	80.0
BasisNet-MV3 (Ours)♦♠♡	**290M**	**80.3**
+ Early Termination (Ours)♦♠♡	**~198M**	**~80.0**

Table 1. Comparison with other efficient networks on ImageNet. Statistics on referenced baselines are cited from original papers. Different training strategies are applied, *e.g.*, ♦ knowledge distillation; ♥ training with extra data; ♠ custom data augmentation; ♣ AutoML-based learned training recipes.

In this paper, we present BasisNet, which takes advantage of progress in all these perspectives and combines several key ideas in a simple new form. The core idea behind BasisNet is *dynamic model synthesis*, which aims at efficiently generating input-dependent specialist model from a

[*] This work was done when Mingda Zhang was an intern at Google.

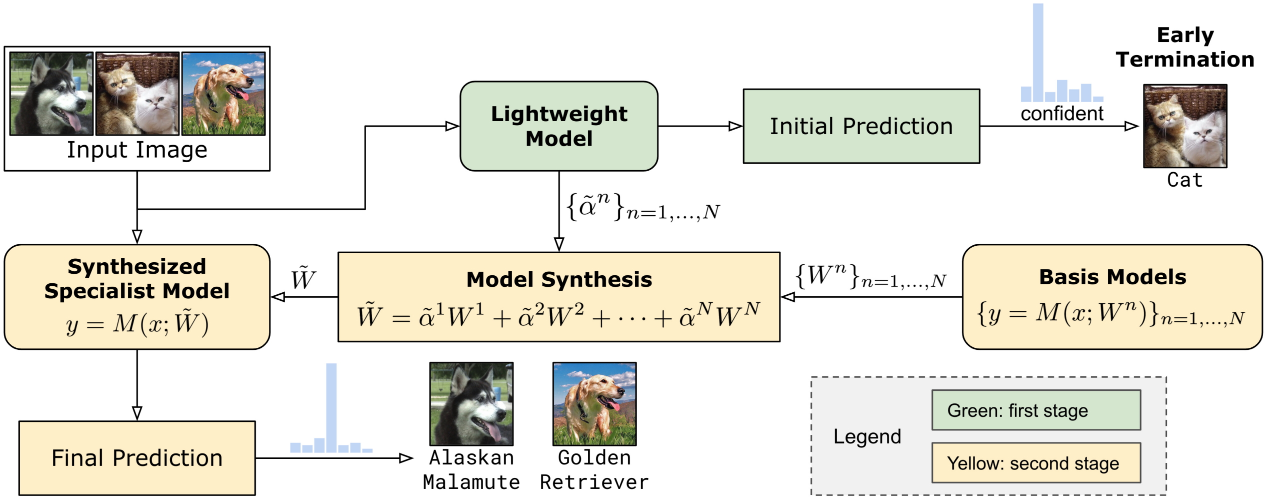


Figure 1. An overview of the BasisNet and more details can be found in Sec. 3.2. For easy images (*e.g.* distinguishing cats from dogs), lightweight model can give sufficiently accurate predictions thus the second stage could be skipped. For more difficult images (*e.g.*, distinguishing different breeds of dogs), a specialist model is synthesized following guidance from lightweight model, which is good at recognizing subtle differences to make more accurate predictions about the given images.

collection of bases, so the resultant model is specialized at handling the given input and can give more accurate predictions. This concept is flexible and can be applied to any *novel network architectures*. On the *hardware side*, the two-stage model synthesis strategy allows the execution of the lightweight and synthesized specialist model on different processing units (*e.g.*, CPU, mobile GPUs, dedicated accelerators, *etc*.) in parallel to better handle streaming data. The BasisNet design is naturally compatible with *early termination*, and can easily balance between computation budget and accuracy. With extensive experiments, we also show that a proper training recipe is critical to mitigate overfitting and improve generalizability.

An overview of the BasisNet is shown in Fig. 1. Using image classification as an example, our BasisNet has two stages: the first stage relies on a *lightweight model* to preview the input image and produce both an initial prediction and a group of combination coefficients. In the second stage, the coefficients are used to combine a set of models, which we call *basis models*, into a single one to process the image and generate the final prediction. The second stage could be skipped if the initial prediction is sufficiently confident. The basis models share the same architecture but differ in some weight parameters, while other weights are shared to avoid overfitting and reduce the total model size.

We validated BasisNet with different generations and sizes of MobileNets and observed significant improvements in inference efficiency. In Table 1 we show comparisons with selected efficient networks on ImageNet classification benchmark. Notably, *without* using early termination, our BasisNet with 16 basis models of MobileNetV3-large only requires 290M Multiply-Adds (MAdds) to achieve 80.3% top-1 accuracy, halving the computation cost of previous state-of-the-art [2] without sacrificing accuracy. If we enable early termination, the *average* cost can be further reduced to 198M MAdds with the top-1 accuracy remaining 80.0% on ImageNet.[1]

In summary, our main contribution is two-fold:

- We propose a two-stage model synthesis strategy that combines efficient neural networks, conditional computation, and early termination in a simple new form. Our BasisNet achieves state-of-the-art performance of accuracy with respect to computation budget on ImageNet even *without* early termination; if enabling early termination, the average computation cost can be further reduced with only marginal accuracy drop.
- We propose an accompanying training recipe for the new BasisNet, which is critical to improve generalizability for high capacity dynamic neural networks, and can also improve the performance of other models.

2. Related Work

Efficient neural networks. Different approaches for building efficient networks have been studied. Early effort includes knowledge distillation [11], post-training pruning [10] and quantization [14]. Later work distinguishes *model complexity* (size) and *run-time latency* (speed), and optimizes for them either with human expertise [19, 35] and/or neural architecture search [12, 22]. All these approaches aim at producing a *static* model that is generally efficient but agnostic to inputs. On the contrary, our Basis-

[1] Average cost is reduced since easy inputs are only handled by lightweight model; max remains 290M MAdds.

Net is built on efficient network architectures, and is *dynamically* adaptive based on inputs. In this work we optimize for inference speed rather than model size.

Conditional computation. Several prior work have explored accelerating inference by skipping part of computation graph based on input-dependent signals. For example, [8] propose a ResNet extension that dynamically adjusts the number of executed layers based on image regions. [24] propose HydraNet which creates multiple parallel branches across the network, and adopts a soft gating module to selectively activate few branches to reduce inference cost. [20] use mixture of experts with a gating network to choose from thousands of candidates. Recently, [33] propose conditionally parameterized convolution (CondConv), which applies weighted combinations of convolution kernels. This idea is adopted by several later work [34, 4], because it has equivalent expressive power as linear mixture of experts, but requires much fewer computations than combining feature maps. However, one common characteristic of these approaches is that their conditioning modules are inserted before each configurable component (e.g., layer or branch), thus these dynamic adjustments only rely on local information. This concept is defined by [5], and according to them lacking global knowledge may be less than optimal because shallower layers cannot benefit from semantic knowledge which is only available from deeper layers. Some other work also identified similar issues and have attempted to leverage global knowledge in dynamic modulation. For example, in SkipNet [30] a gating network is built to conditionally skip certain layers in the backbone, and the authors report that the best performance comes from a RNN-based gating network because it can access feature maps across multiple layers. [5] introduce GaterNet where a dedicated deep neural network is used to analyze the inputs before generating input-dependent masks for the filters in backbone network. BasisNet use a lightweight but fully-fledged model to process the inputs and produce combination coefficients, thus the model synthesis is relying on semantic-aware global knowledge. Different from SkipNet and GaterNet, our lightweight model can synthesize new kernels that do not exist beforehand via linear combination. Another distinction is that by separating conditioning model from backbone, our BasisNet is more flexible and easier to adapt to different architectures and hardware constraints.

Cascading networks and early exiting. Since input samples are naturally of varying difficulty, using a single model to equally process all inputs with a fixed computation budget is wasteful. This observation has been leveraged by prior work, e.g., the famous Viola-Jones face detector [27] built a cascade of increasingly more complex classifiers to achieve real-time execution. Similar ideas were also used in deep learning, e.g., reducing unnecessary inference computations for easy cases in a cascaded system [26], attaching multiple classification heads on different layers [23, 13], or cascading multiple models [1]. One common limitation in previous work is that only the exit point adapts to the samples but the underlying models remain static. Instead, our BasisNet dynamically adjusts the convolution kernel weights based on the guidance from lightweight model, thus the synthesized specialist can better handle the more difficult cases.

3. Approach

In general, our BasisNet has two stages: the first stage lightweight model, and the second stage model synthesis from a set of basis models. Given a specific input, the lightweight model generates two outputs, an initial prediction and a group of basis *combination coefficients*. If the initial prediction is of high confidence, the input is presumably easy and BasisNet could directly return the initial prediction and terminate early. But if the initial prediction is less confident (implying the input is difficult, *e.g.* identifying dogs by breed), the *coefficients* will be used to guide the synthesis of a specialist model in the second stage. The synthesized specialist will handle final prediction.

3.1. Lightweight model

The lightweight model is a fully-fledged network handling two tasks: generating initial category prediction and generating combination coefficients for second stage model synthesis. The first is a standard classification task thus we only elaborate on the second below. Assuming there are N basis models and each has K layers, the lightweight model will predict combination coefficients $\alpha \in \mathcal{R}^{K \times N}$

$$\alpha = \phi(\text{LM}(f(x))) \tag{1}$$

where LM stands for *lightweight model* and ϕ represents a non-linear activation function. We use softmax by default because it enforces convexity, which promotes sparsity and can lead to more efficient executions. $f(x)$ represents a transformation of the input image, and we typically use $f(x) = x$ or $f(x) = \text{DownSampling}(x)$.

3.2. Basis model synthesis

Our basis models are a collection of model candidates, which share the same architecture but differ in model parameters. By combining basis models with different weights, a specialist network can be synthesized. Various strategies can be used for building basis models, such as mixture of experts [20] or using multiple parameter-efficient patches [17]. We explored a few options and found that the recently proposed CondConv [33] best fits our needs for building a low-latency but high-capacity model.

Specifically, consider a *regular* deep network with image input x. Assume the output of the k-th convolutional layer is $O_k(x)$, which could be obtained by

$$O_k(x) = \begin{cases} \phi(W_0 * x), & \text{if } k = 0 \\ \phi(W_k * O_{k-1}(x)), & \text{if } k > 0 \end{cases} \tag{2}$$

where W_k represents the convolution kernel at the k-th layer and $*$ represents a convolution operation. For simplicity some operations like batch normalization and squeeze-and-excitation are omitted from the notation. In BasisNet, different inputs will be processed by different, input-dependent kernel $\tilde{W}_k$ at k-th layer, which is obtained by linearly combining the kernels from N basis models at k-th layer, denoted by $\{W_k^n\}_{n=1,\ldots,N}$:

$$\tilde{W}_k = \tilde{\alpha}_k^1 \cdot W_k^1 + \cdots + \tilde{\alpha}_k^N \cdot W_k^N \tag{3}$$

where $\tilde{\alpha}_k^n$ represents the weight for the k-th layer of the n-th basis. We use $\tilde{W}$ and $\tilde{\alpha}$ to emphasize their dependency on x. This design allows us to increase model capacity effectively but retain the same number of convolution operations. Besides, since the number of parameters is much less than number of MAdds in a single basis architecture, the combination only marginally increase the computation cost.

Besides, using sparse convex coefficients further reduces the overhead. Thus we generally consider convex coefficients, but also studied two special cases:

- α_k is the same for all layers. In this case, the combination is *per-model* instead of *per-layer*.
- α_k as an N-dimension vector is one-hot encoded. In this case, model synthesis becomes *model selection*.

Key difference from CondConv. Our model synthesis mechanism is inspired by CondConv [33] but there exists many distinctions. In CondConv the combination coefficients for k-th layer are computed following

$$\alpha_k = \phi(\text{FC}(\text{GAP}(O_{k-1}(x)))) \tag{4}$$

where FC stands for *fully connected layer* and GAP stands for *global average pooling*. This formulation shows the dynamic kernels in CondConv can only be synthesized layer by layer, because the combination coefficients for next layer depend on output of previous layer. This complicates scheduling of computation thus is not hardware friendly [34]. In BasisNet, the issue is addressed by the lightweight model, which generates the combination coefficients for all layers simultaneously as shown in Equation 1. Therefore the entire specialist model can be synthesized all at once. Separating kernel combination from execution also enables BasisNet to be easily deployed *to* (or even *across*) different hardware accelerators on edge devices if needed. Besides, early termination is naturally supported by BasisNet, but is much harder to be incorporated for CondConv. Arguably one can try attaching additional prediction heads like [23] to enable "layer-level early termination" for CondConv. However, this change requires non-trivial efforts for designing the proper exit points in CondConv, let alone introducing extra computational cost. More specifically, since the backbone efficient network is already highly compact (*e.g.* MobileNets), it is unlikely that early layers can offer signals sufficient for prediction which is required for early termination. For BasisNet, the signal comes from fully-fledged lightweight model, which generates the prediction as a side product thus offers early termination for free. Lastly, BasisNet is complementary to CondConv, as we find (in Sec. 4.5) that combining CondConv and BasisNet can further boost prediction accuracy.

3.3. Training BasisNet properly

BasisNet significantly increases model capacity, but the risk of overfitting also increases. We found the standard training procedures used to train MobileNets lead to severe overfitting on BasisNet. Here we describe a few regularization techniques that are crucial for training BasisNet successfully. This is also a key contribution of our work, as previously there is no good practice on how to effectively train such high capacity dynamic neural networks.

- **Basis model dropout (BMD)** Inspired by [9], we experimented with randomly shutting down certain basis model candidates during training. It is similar to applying DropConnect [29] on the predicted coefficient matrix from the lightweight model. We find this approach is extremely effective against "experts degeneration" [20] where the controlling model always picks the same few candidates and never activates the rest.
- **AutoAugment (AA)** AutoAugment [6] is a search-based procedure for finding specific data augmentation policy towards a target dataset. We find that replacing the original data augmentation in MobileNets [19] with the ImageNet policy in AutoAugment can significantly improve the model generalizability.
- **Knowledge distillation** [11] showed that using soft targets from a well-trained teacher network can effectively prevent a student model from overfitting. We observe that knowledge distillation is also effective on training BasisNet, and find EfficientNet-B2 with noisy student training [32] can be a good teacher.

In addition to stronger regularization, we applied a few other tricks in order to properly train BasisNet. Since the lightweight model directly controls how the specialist model is synthesized, any slight changes in the combination coefficients will propagate to the parameter of the synthesized model and finally affect the final prediction. Since we train the two stages from scratch, this is especially troublesome at the early phase when the lightweight model is still ill-trained. To deal with the unstable training, we introduced

$\epsilon \in [0,1]$ to balance between a uniform combination and a predicted combination coefficients from the lightweight model,

$$\alpha' = \epsilon \cdot \frac{1}{N} \cdot \mathbf{1}^{K \times N} + (1-\epsilon) \cdot \alpha \tag{5}$$

When $\epsilon = 1$ all bases are combined equally while when $\epsilon = 0$ the synthesis is following the combination coefficients. In practice ϵ linearly decays from 1 to 0 in the early phase of training then remains at 0, thus the lightweight model can gradually take over the control of model synthesis. This approach effectively stabilizes training and accelerates convergence. A recent work [4] proposed temperature-controlled softmax to achieve similar goal.

All models in both stages are trained together in an end-to-end manner via back-propagation. In other words, all basis models are trained from scratch by gradients from the synthesized model. The total loss includes two cross-entropy losses for the synthesized model and the lightweight model, respectively, and L2 regularization,

$$\begin{aligned} L = &-\log P(y|x;\tilde{W}) + \lambda(-\log P(y|f(x);W_{\text{LM}})) \\ &+ \Omega(\{W^n\}_{n=1,\ldots,N}, W_{\text{LM}}) \end{aligned} \tag{6}$$

where λ is the weight for cross-entropy loss from lightweight model ($\lambda = 1$ in our experiments), and $\Omega(\cdot)$ is a L2 regularizer applied to all model parameters. The lightweight model receives all gradients, while basis models are only updated by the first term and regularization.

4. Experiments

4.1. Dataset and model architecture setup

We demonstrate the effectiveness of BasisNet on both MobileNetV2 and MobileNetV3 architectures, and evaluate on the ImageNet ILSVRC 2012 classification dataset [18] consisting of 1.28M images for training and 50K for validation. We did not explicitly use extra data, but one teacher model we used for knowledge distillation, *i.e.*, EfficientNet-b2 with noisy student training [32], is obtained with extra data. For BasisNet-MV2, the basis models follow the architecture described in Table 2 of [19]. For simplicity in notation, we sequentially number all the layers starting from L0, *e.g.* the first `conv2d` layer is L0 and the `avgpool 7x7` layer is L19. For BasisNet-MV3, the basis models follow the MobileNetV3-large architecture described in Table 1 of [12]. We also sequentially number all the layers, *e.g.* the `pool,7x7` layer is L17.

For fair comparison, we *retrained all models including BasisNet and all the baselines using the same training recipe*, and reported the performance *without* early termination except for Sec. 4.6. Note that the lightweight model introduces computation overhead for BasisNet, but our reported MAdds statistics for BasisNet always *include* the lightweight model. More details about our model as well as training recipes can be found in supplementary materials.

4.2. Comparison with MobileNets

For both BasisNet-MV2 and BasisNet-MV3, we compute the accuracy-MAdds curves by varying the input image resolution to the synthesized model from `{128, 160, 192, 224}`. We compute the curves for the MobileNets in the same way. As shown in Fig. 2, even with the computation overhead of the lightweight model, our BasisNets consistently outperform the MobileNets with large margins.

4.3. The effect of regularization for proper training

In Fig. 3 we show the performance improvements when different regularizations (basis model dropout, AutoAugment, and knowledge distillation) discussed in Sec. 3.3 are individually applied to BasisNet-MV2 training, as well as combined altogether. Each regularization helps generalization, and the most effective single regularization is the knowledge distillation. By combining all strategies the validation accuracy increases the most. In fact, we observed that the proposed training recipe also helps improving performance of other models like original MobileNets, as shown in Fig. 5. However, applying the regularization is more crucial for BasisNet training, as the top-1 accuracy of BasisNet-MV2 (1.0x224) improves by +3.4 percentage points (74.7% → 78.1%), while for MobileNetV2 the improvement is +2.0 percentage points (72.9% → 74.9%).

4.4. Number of bases in basis models

We varied the number of bases to investigate their effect on the model size, inference cost and final accuracy. Intuitively, the more bases in the candidate pool, the more diverse domains the final synthesized model can adapt to. We chose a fix-sized MV3-small (1.0x224) as our lightweight model, and use different numbers of MV3-large (1.0x224) for basis. As shown in Fig. 4, the top-1 accuracy improves monotonically with increased number of bases. With 16 bases, our BasisNet-MV3 achieved 80.3% accuracy with 290M MAdds. The shaded area represents the relative model size (#Params). Note that we explicitly trained a regular MobileNetV3-large with large multiplier and low image resolution (2.5x128), so it has similar model size with BasisNet. We show that BasisNet requires only 2/3 of computations (290M vs 435M) to achieve the comparable accuracy with the MobileNetV3 counterpart (80.3% vs 80.4%).

4.5. Comparison with CondConv

We re-implemented CondConv[2] to directly compare with our BasisNet. We choose MobileNetV3 as backbone,

[2] Our re-implementation of CondConv-MV2 achieved 76.2% accuracy, better than the reported 74.6% from [33].

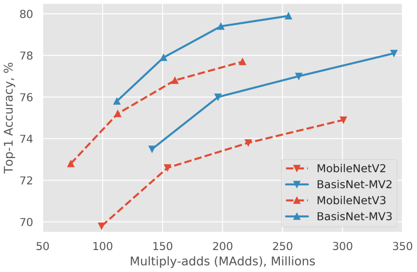


Figure 2. Accuracy-MAdds trade-off comparison of the proposed BasisNet and MobileNet on ImageNet validation set.

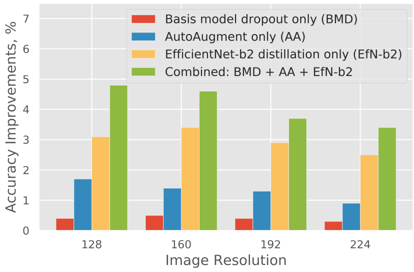


Figure 3. Performance boost with various regularizations on BasisNet-MV2. All combined gives the largest improvement.

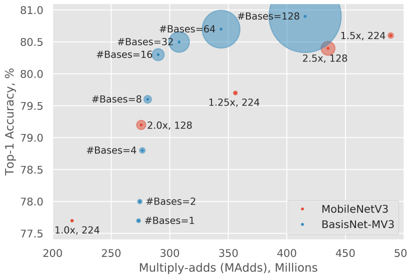


Figure 4. Prediction accuracy monotonically increases when more bases are added to the basis models.

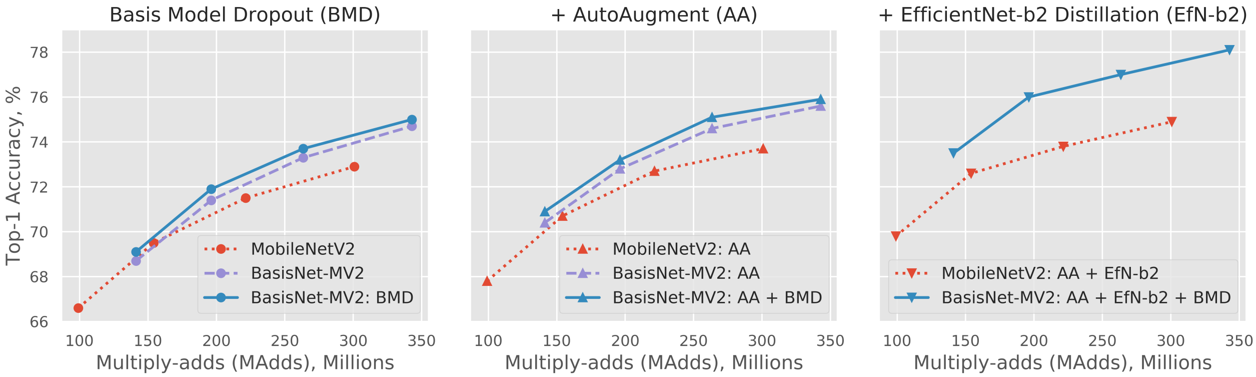


Figure 5. MobileNet and BasisNet training using different regularizations. BasisNet uses MV2-0.5x as its lightweight model and 8 MV2-1.0x for basis models. Input image resolutions vary from `{128, 160, 192, 224}`. Note that basis model dropout (BMD) is not applicable to MobileNet because it has only one model.

Model	Activation	MAdds	Top-1 Acc.
CondConv-MV3	Sigmoid	253M	79.9%
BasisNet-MV3	Softmax	290M	80.3%
BasisNet-MV3	Sigmoid	290M	80.0%
(BasisNet+CC)-MV3	Softmax	290M	80.5%

Table 2. Comparison of BasisNet with CondConv.

and selected $N = 16$ for both BasisNet and CondConv from layers 11 to 15. We chose MV3-small as the lightweight model for BasisNet, and disabled early termination for fair comparison. All models including CondConv baselines are re-trained using the same recipe as in Sec. 3.3.

The top-1 accuracy for CondConv-MV3 and BasisNet-MV3 is 79.9% and 80.3% respectively, although BasisNet has relatively larger overhead due to the lightweight model. However, we find that BasisNet is more flexible than CondConv. CondConv reports that simultaneously activating multiple routes is essential for any single input, therefore sigmoid activation has to be used. For BasisNet, we find both sigmoid and softmax work fine (80.0% and 80.3% accuracy respectively). In fact, using softmax can lead to sparse and even one-hot combination coefficients (see Sec. 4.9), which may help reducing latency from model loading I/O perspective. We also experiment to combine CondConv with BasisNet, and the accuracy can be further boosted to 80.5%, showing the performance gain from BasisNet is complementary to CondConv.

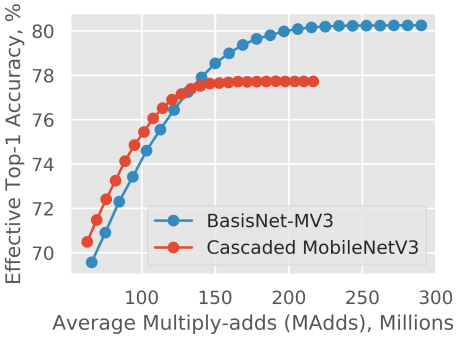


Figure 6. Simulated accuracy comparison of BasisNet-MV3 and cascaded MobileNets with early exiting under varying thresholds.

4.6. Early stop to reduce average inference cost

The two stage design of BasisNet naturally supports early termination, as the lightweight model can make an initial prediction. We chose the maximum value of softmax probability [13] of initial prediction as the criterion to measure the confidence. Specifically, for each input image if the initial prediction confidence is higher than a predefined threshold then second stage could be skipped, otherwise the second stage specialist needs be synthesized to make the final prediction.

We verified the early termination strategy on ImageNet validation set with a well-trained BasisNet-MV3 (1.0x224,16 basis) model. In Fig. 6, we alter thresholds of initial prediction confidence and plot the average cost and accuracy of BasisNet. For fair comparison, we cascade two well-trained MobileNets of the same size as the lightweight model and basis model respectively. In general the figure shows that BasisNet achieves better results for the same cost, except when the computation budget is very limited. Particularly for BasisNet, with a threshold of 0.7, 39.3% of images will skip the second stage thus the average computation cost reduces to 198M MAdds while the overall accuracy remains 80.0% on ImageNet validation set.

4.7. Convex combination: special cases

Per-model model synthesis. When lightweight model predicts a single vector of combination coefficients *for all layers*, *i.e.* $\alpha_1 = \alpha_2 = \cdots = \alpha_K \in \mathcal{R}^N$, it can be seen as a per-model synthesis. Note that per-model synthesis of BasisNet is still different from HydraNets [24], as the branches in HydraNets span across multiple layers and do not fuse in the middle; instead, in BasisNet the convolution kernels are obtained from linear combination for each layer.

We use BasisNet-MV3 with 8 bases and a lightweight model of MV3-small (1.0x224), and share all layers in basis models except for L11-15. Interestingly both per-model BasisNet and per-layer BasisNet have the same performance, 79.6% top-1 accuracy on ImageNet validation set, implying the combination coefficients across layers may have high correlations for BasisNet-MV3. We also experiment with BasisNet-MV2 in a similar setting, but it turns out training per-model BasisNet-MV2 is more challenging because the model easily collapses after roughly 30K steps in our multiple attempts. We suspect that training per-model model synthesis is generally more difficult as it has stronger constraints on the basis models, and it may depend on the base architectures (MobileNetV2 or MobileNetV3).

Model selection instead of model synthesis. When the predicted combination coefficients are one-hot encoded, the model synthesis can be simplified as *model selection*, as only one base will be selected for a particular layer. We experimented with BasisNet-MV3 with 8 bases, and the lightweight model is MV3-small (1.0x128). Basis models share all layers except for L8-15, and the original BasisNet-MV3 has an accuracy of 79.8% under this setting. After training for 100K steps we froze the lightweight model and transformed the predicted combination coefficients into one-hot embedding, then continued training the basis models. The resulting BasisNet finally achieved 78.5% accuracy. This is +0.7% better than post-processing a well-trained BasisNet (77.8%) implying the potential for training model selection end-to-end. We leave more careful finetuning for the model selection as future work, but emphasize that model selection has potential to further reduce latency in practice from a model loading I/O perspective.

Models	Top-1 Acc.	MAdds (M)	Latency (ms)
BasisNet-MV3 8-routes	79.6%	281	60.6
BasisNet-MV3 16-routes	80.3%	290	62.9
– With early termination	80.0%	198 (avg.)	43.6 (avg.)
MobileNetV3 (1.25x224)	79.7%	356	66.3
MobileNetV3 (1.5x224)	80.6%	489	86.2
CondConv-MobileNetV3	79.9%	253	53.1

Table 3. Latency measurements on Google Pixel 3XL.

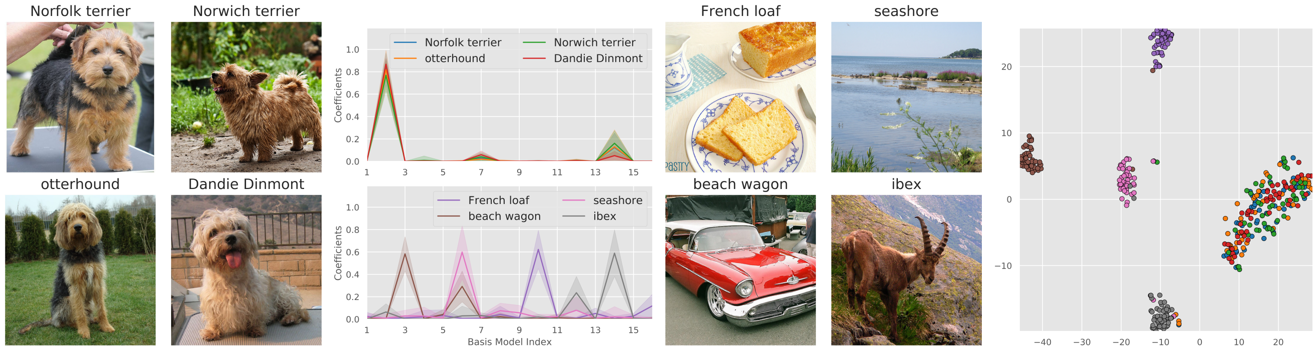
4.8. On-device latency measurements

To validate the practical applicability, we measured the latency of the proposed BasisNet and other baselines on physical mobile device. We choose Google Pixel 3XL and run floating-point models on the big core of the phone's CPU. In Table 3 we show that BasisNet can run efficiently on existing mobile device. Our efficiency conclusion drawn from MAdds also applies to real latency. Specifically, MobileNetV3 with 1.25x and 1.5x multipliers have similar accuracy as BasisNet-MV3 with 8 and 16 routes, while the BasisNet has lower latency. We also measured the latency for CondConv. Primarily because of the first stage lightweight model, BasisNet without early termination has higher latency than CondConv (62.9ms vs 53.1ms). However, we emphasize that the first stage lightweight model generates better combination coefficients thus improves the top-1 accuracy (80.3% vs 79.9%). Besides, the lightweight model generates initial prediction to enable early termination. When early termination is enabled, the average latency for BasisNet reduced significantly to 43.6ms[3], which is much lower than CondConv (53.1ms) while retaining slightly superior accuracy (80.0% vs 79.9%). As we described in Sec. 3.2, deploying early termination for CondConv is much more challenging.

4.9. Understanding the learned BasisNet models

Visualizing the specialization of basis models. We visualized the combination coefficient vectors on ImageNet validation set to better understand the effectiveness of model synthesis. In Fig. 7 we show visually similar and distinct categories, as well as the combination coefficients of L15. From (B) top, we can see that the lightweight model chooses the same specialist for distinguishing dif-

[3] With threshold of 0.7 on ImageNet, 39.3% of images can skip second stage thus the estimated average latency is reduced to $0.393 \times 13.7\text{ms} + (1 - 0.393) \times 62.9\text{ms} = 43.6\text{ms}$.



(A) (B) (C) (D)

Figure 7. (A,C) Sample images from visually similar or distinct categories. (B) Mean coefficient weights at L15 layer for selected categories. (D) t-SNE visualization of combination coefficients.

Disturbance	BasisNet-MV2	BasisNet-MV3
CORRECT	78.2	79.8
TOP-1	73.9 (-4.3)	77.8 (-2.0)
MEAN	67.2 (-11.0)	69.5 (-10.3)
UNIFORM	67.2 (-11.0)	69.7 (-10.1)
SHUFFLED	56.5 (-21.7)	58.1 (-21.7)

Table 4. Apply various disturbance to combination coefficients.

ferent dogs, in order to better handle the subtleties between dog breeds. But for visually distinct categories, the synthesized models are very different evidenced by the non-overlapping curves in (B) bottom. In Fig. 7 (D) we show the coefficients for all images using t-SNE [25]. The dog categories form a single cluster while the others reside in very different clusters. More interestingly, we find that even fine-grained visual patterns can be distinguished as different base models are activated, *e.g.*, fluffy dogs mainly activate 2nd base but short-haired dogs use 14th base. More qualitative results are provided in supplementary materials.

The importance of optimal basis model synthesis. To verify the importance of model synthesis, we apply disturbances to the predicted combination coefficients. The specialist should be most effective for the corresponding image, and a disturbed synthesis signal is expected to hurt performance. We train BasisNet-MV2 (Accuracy 78.2%) and BasisNet-MV3 (Accuracy 79.8%), and share only the first 7 layers in the basis, then disturb the coefficients α as follows: (1) preserving the highest probable basis model only (TOP-1), (2) uniformly combining all basis models (UNIFORM), (3) using mean weights over entire validation set (MEAN), or (4) randomly shuffling the coefficients within each layer (SHUFFLED). As shown in Table 4, all disturbances lead to inferior performance validating that basis models have varied expertise. SHUFFLED leads to a totally mismatched specialist thus performance drops over 20 percentage points.

Effect of model synthesis at different layers We also apply disturbances on each individual layer to investigate the sensitivity within the model. As shown in Table 5, we find the layers closer to the final classification layer have more impacts, as the accuracy drop is more significant. Interestingly, the regular convolutional layer right after the *residual bottleneck layers* [19, 12] (e.g. L18 of MobileNetV2 and L16 of MobileNetV3) seems less sensitive towards inputs.

Disturbed Layer	BasisNet-MV2	BasisNet-MV3
8	78.1 (-0.1)	79.6 (-0.2)
9	78.1 (-0.1)	79.6 (-0.2)
10	78.0 (-0.2)	79.6 (-0.2)
11	78.0 (-0.2)	79.4 (-0.4)
12	77.7 (-0.5)	79.0 (-0.8)
13	77.4 (-0.8)	78.3 (-1.5)
14	77.2 (-1.0)	77.9 (-1.9)
15	76.0 (-2.2)	76.4 (-3.4)
16	76.0 (-2.2)	79.1 (-0.7)
17	76.1 (-2.1)	N/A
18	77.2 (-1.0)	76.6 (-3.2)
Reference	78.2	79.8

Table 5. Top-1 accuracy drops when SHUFFLED disturbance was applied at different layer. The last row shows the reference model that uses undisturbed predicted coefficients.

5. Conclusion

We present BasisNet, which combines the recent advancements in multiple perspectives such as efficient model design and dynamic inference. With a standalone lightweight model, the unnecessary computation on easy examples can be saved and the information extracted by the lightweight model help synthesizing a specialist network for better prediction. With extensive experiments on ImageNet we show the proposed BasisNet is particularly effective on efficient inference, and BasisNet-MV3 achieves 80.3% top-1 accuracy with only 290M MAdds even without early termination.

References

[1] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for efficient inference. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017. 1, 3

[2] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *International Conference on Learning Representations (ICLR)*, 2020. 1, 2

[3] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *International Conference on Learning Representations (ICLR)*, 2019. 1

[4] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 3, 5

[5] Zhourong Chen, Yang Li, Samy Bengio, and Si Si. You look twice: Gaternet for dynamic filter selection in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[6] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 4

[7] Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Bichen Wu, Zijian He, Zhen Wei, Kan Chen, Yuandong Tian, Matthew Yu, Peter Vajda, et al. Fbnetv3: Joint architecture-recipe search using neural acquisition function. *arXiv preprint arXiv:2006.02049*, 2020. 1

[8] Michael Figurnov, Maxwell D. Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 3

[9] Xavier Gastaldi. Shake-shake regularization. *arXiv preprint arXiv:1705.07485*, 2017. 4

[10] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*, 2016. 2

[11] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. 2, 4

[12] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 1, 2, 5, 8

[13] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Weinberger. Multi-scale dense networks for resource efficient image classification. *International Conference on Learning Representations (ICLR)*, 2018. 3, 6

[14] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 2

[15] Juhyun Lee, Nikolay Chirkov, Ekaterina Ignasheva, Yury Pisarchyk, Mogan Shieh, Fabio Riccardi, Raman Sarokin, Andrei Kulik, and Matthias Grundmann. On-device neural net inference with mobile gpus. In *Efficient Deep Learning for Computer Vision CVPR 2019 (ECV2019)*, 2019. 1

[16] 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)*, 2018. 1

[17] Pramod Kaushik Mudrakarta, Mark Sandler, Andrey Zhmoginov, and Andrew Howard. K for the price of 1: Parameter-efficient multi-task and transfer learning. *International Conference on Learning Representations (ICLR)*, 2018. 3

[18] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 2015. 5

[19] 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 (CVPR)*, 2018. 1, 2, 4, 5, 8

[20] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *International Conference on Learning Representations (ICLR)*, 2017. 3, 4

[21] 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/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[22] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019. 1, 2

[23] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, 2016. 3, 4

[24] Ravi Teja Mullapudi, William R Mark, Noam Shazeer, and Kayvon Fatahalian. Hydranets: Specialized dynamic architectures for efficient inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3, 7

[25] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 2008. 8

[26] Swagath Venkataramani, Anand Raghunathan, Jie Liu, and Mohammed Shoaib. Scalable-effort classifiers for energy-efficient machine learning. In *Proceedings of the Annual Design Automation Conference (DAC)*, 2015. 3

[27] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001. 3

[28] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, Peter Vajda, and Joseph E. Gonzalez. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[29] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2013. 4

[30] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 1, 3

[31] Longhui Wei, An Xiao, Lingxi Xie, Xin Chen, Xiaopeng Zhang, and Qi Tian. Circumventing outliers of autoaugment with knowledge distillation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 1

[32] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 4, 5

[33] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 1, 3, 4, 5

[34] Tianyuan Zhang, Bichen Wu, Xin Wang, Joseph Gonzalez, and Kurt Keutzer. Domain-aware dynamic networks. *arXiv preprint arXiv:1911.13237*, 2019. 3, 4

[35] 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 (CVPR)*, 2018. 1, 2

[36] Yikang Zhang, Jian Zhang, Qiang Wang, and Zhao Zhong. Dynet: Dynamic convolution for accelerating convolutional neural networks. *arXiv preprint arXiv:2004.10694*, 2020. 1