Multiple Networks are More Efficient than One: Fast and Accurate Models via Ensembles and Cascades

Xiaofang Wang [*]
Carnegie Mellon University
xiaofan2@cs.cmu.edu

Dan Kondratyuk [†]
Google Research
dankondratyuk@google.com

Kris M. Kitani
Carnegie Mellon University
kkitani@cs.cmu.edu

Yair Movshovitz-Attias
Google Research
yairmov@google.com

Elad Eban
Google Research
elade@google.com

Abstract

Recent work on efficient neural network architectures focuses on discovering a solitary network that can achieve superior computational efficiency and accuracy. While this paradigm has yielded impressive results, the search for novel architectures usually requires significant computational resources. In this work, we demonstrate a simple complementary paradigm to obtain efficient and accurate models that requires no architectural tuning. We show that committee-based models, i.e., ensembles or cascades of models, can easily obtain higher accuracy with less computation when compared to a single model. We extensively investigate the benefits of committee-based models on various vision tasks and architecture families. Our results suggest that in the large computation regime, model ensembles are a more cost-effective way to improve accuracy than using a large solitary model. We also find that the computational cost of an ensemble can be significantly reduced by converting them to cascades, while often retaining the original accuracy of the full ensemble.

1. Introduction

Neural networks with higher performance usually incur more computational cost during inference. However, many real-world applications can only afford to use limited resources, thus imposing constraints on the computational cost of network inference. This has motivated the efforts to optimize the speed-accuracy trade-off of neural networks. Recent work in this direction [46, 15, 28, 14, 36, 5] mostly focus on designing novel network architectures such that one *single* network can achieve high accuracy with small computational cost. This paradigm of designing novel architectures has yielded impressive results.

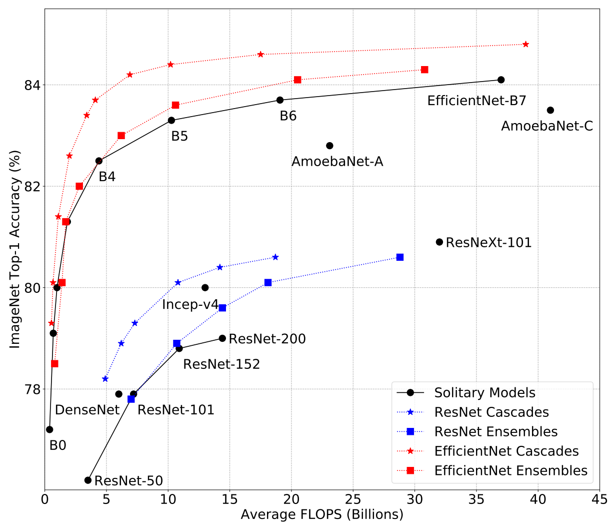


Figure 1: Committee-based models v.s. Solitary models. Committee-based models can achieve better accuracy with fewer average FLOPS than solitary models. For example, while Inception-v4 [32] ('Incep-v4') obtians a higher accuracy than all solitary ResNet [13] models in the figure, the cascade of ResNet models can still outperform Inception-v4 while being more efficient.

However, discovering novel architectures is highly challenging. Manually designing network architectures requires extensive experience and knowledge of downstream tasks. Recent work on neural architecture search (NAS) proposes to automatically design the architecture in a data-driven manner. But state-of-the-art NAS methods usually require significant computational resources. For example, it took 2,000 GPU days to discover NASNet [48] and 288 TPUv2

[*]Work done during an internship at Google.
[†]Work done as part of the Google AI Residency Program.

days to find MnasNet [35].

In this work, we demonstrate that committee-based models, *i.e*., ensembles or cascades of models, which are well-known techniques in machine learning, can be much more efficient than state-of-the-art solitary models. This provides a complementary paradigm to designing or searching for novel networks architectures. Such committee-based models are simple to use and require no architectural tuning.

Our work aims to highlight the practical benefits of committee-based models and does not propose new algorithms for building ensembles or cascades. We demonstrate that one can obtain considerably more efficient models by constructing committees of existing networks with the simplest committee-building techniques. The findings generalize to several vision tasks, including image classification on ImageNet [27], video classification on Kinetics-600 [4], and semantic segmentation on Cityscapes [7]. Committee-based models also work for different architecture families, such as EfficientNet [36], ResNet [13], MobileNetV2 [28], and X3D [9]. The paper is organized as a series of experiments analyzing the efficiency of committee-based models.

First, we study the efficiency of model ensembles, and find that in the large computation regime, ensembles are a more cost-effective way to improve accuracy than using a larger solitary model. For example, an ensemble of two independently trained EfficientNet-B5 models matches B7 accuracy, a state-of-the-art ImageNet model, while having almost 50% less FLOPS (20.5B v.s. 37B FLOPS). In addition to the speedup in inference FLOPS, two B5 models also take much shorter time to train than one B7.

Second, we show that one can considerably speed up ensembles by converting them to cascades, such that we can achieve full ensemble accuracy without paying the full computational cost of ensembles. Cascades demonstrate better efficiency than solitary models in all computation regimes. Continuing with the example above, by cascading two B5 models, we can further reduce average FLOPS to 13.1B while retaining B7 accuracy (a 2.8x speedup).

Third, for best performance, we show that one can design cascades to match the FLOPS or accuracy of a specific solitary model, by selecting models to be used in the cascade. This yields even better speedup over solitary models. For example, we select existing networks from EfficientNet [36] to build a cascade, and target for a similar accuracy to EfficientNet-B7. Our cascade matches B7 accuracy while using on average 5.4x fewer FLOPS. For more practical usefulness, we also show that cascades can provide guarantee for worst-case FLOPS, and can be easily scaled up to respect different FLOPS constraints.

Fourth, we show that one can convert a single model into a cascade by passing the same input image at different resolutions to the model. We refer to such cascades as "self-cascades" since the cascade only contains the model itself. These self-cascades also demonstrate speedup over solitary models. Our results suggest that, if one wants to achieve B7 accuracy, training a B6 model and building a self-cascade can be faster in both training and inference.

Finally, we show that the benefit of committee-based models generalizes beyond image classification, and cascades outperform solitary models in multiple domains, *e.g*., video classification on Kinetics-600 [4] and semantics segmentation on Cityscapes [7]. Our cascade outperforms X3D-XL by 1.2% while using fewer FLOPS, and achieves 83.1% accuracy, a new state-of-the-art on Kinetics-600.

2. Related Work

Efficient Neural Networks. There has been significant progress in designing efficient neural networks. In early work, most efficient networks, such as MobileNet [15, 28] and ShuffleNet [14], are manually designed by human experts. Recent work starts to use neural architectures search (NAS) to automatically learn efficient network designs [48, 3, 35, 36, 5]. These work mostly focuses on discovering one single network with superior computational efficiency and accuracy. Our work demonstrates a complementary paradigm to obtain efficient and accurate models without tuning the architecture.

Ensembles. Ensemble learning has been well studied in machine learning and there have been many seminal works, such as Bagging [2], Boosting [29], and AdaBoost [11]. Ensembles of neural networks have been used to boost the performance on many tasks, such as image classification [33, 18], video classification [25], machine translation [42], and out-of-distribution robustness [21, 10, 43]. But the efficiency of model ensembles has rarely been systematically investigated. Recent work indicates that model ensembles can be more efficient than solitary models on image classification tasks [20, 24]. Our work further substantiates this claim through extensive analysis of modern architectures on large-scale benchmarks.

Cascades. A large family of works have explored using cascades to speed up certain tasks. For example, the seminal work from Viola and Jones [39] builds a cascade of increasingly complex classifiers to speed up face detection. Cascades have also been explored in the context of deep neural networks. Bolukbasi et al. [1] reduces the average test-time cost by learning a policy to allow easy examples to early exit from a network. A similar idea is also explored by Guan et al. [12]. Huang et al. [17] proposes a novel architecture Multi-Scale DenseNet to better incorporate early exits into neural networks. Given a pool of models, Streeter [31] presents an approximation algorithm to produce a cascade such that the cascade uses minimal average inference cost to obtain a certain accuracy. Different from previous work that develops new cascade-building techniques, we show that one can obtain models with superior efficiency and accuracy with the simplest cascade-building techniques.

Conditional Computation. Conditional computation methods allocate computational resources based on the input image, *i.e.*, spending more computation on hard images and less on easy ones. For example, Shazeer et al. [30] trains a gating network to determine what parts in a high-capacity model should be used for each example. Recent work [44, 38, 41] proposes to learn a policy to dynamically select layers or blocks to execute in ResNet based on the input image. Model cascades can be viewed as one particular solution for conditional computation, which are simple to use and demonstrate superior performance.

3. Efficiency of Ensembles

Model ensembles are useful for improving accuracy, but the usage of multiple models in an ensemble also introduces extra computational cost. When holding the total computation is fixed, which one will give higher accuracy: solitary models or model ensembles? The answer is important for real-world applications but this question has rarely been systematically studied on modern network architectures and large-scale benchmarks.

To answer the question, we investigate the efficiency of ensembles on ImageNet [27] with three architecture families: EfficientNet [36], ResNet [13], and MobileNetV2 [28]. Each architecture family contains a series of networks with different levels of speed-accuracy trade-off. Within each family, we train a pool of models, compute the ensemble of different combinations of models, and compare these ensembles with the solitary models in the family.

We denote an ensemble of n image classification models by $\{M_1, \ldots, M_n\}$, where M_i is the i^{th} model. Given an image x, $\alpha_i = M_i(x)$ is a vector representing the logits for each class. To ensemble the n models, we compute the mean of logits[1] $\alpha^{\text{ens}} = \frac{1}{n}\sum_i \alpha_i$ and predicts the class for image x by applying argmax to α^{ens}. The total computation of the ensemble is $\text{FLOPS}^{\text{ens}} = \sum_i \text{FLOPS}(M_i)$, where $\text{FLOPS}(\cdot)$ gives the FLOPS of a model.

We show the top-1 accuracy on ImageNet and FLOPS of solitary models and ensembles in Figure 2. Since there are many possible combinations of models to ensemble, we only show those Pareto optimal ensembles in the figure. We see that ensembles are more cost-effective than large solitary models, *e.g.*, EfficientNet-B5/B6/B7 and ResNet-152/200. But in the small computation regime, solitary models outperform ensembles. For example, the ensemble of 2 B5 models matches B7 accuracy while using about 50% less FLOPS. However, ensembles use more FLOPS than solitary MobileNetV2 models when they have a similar accuracy.

Large models can suffer from overfitting and fixate on the noise or irrelevant information in training data. Therefore, increasing the size of an already large model may only give a small gain in accuracy (*e.g.*, from B5 to B7, the FLOPS increase by 3.6x with only 0.8% accuracy gain). Ensembles take advantage of multiple independently trained models to reduce the variance in prediction [47], which can lead to lower test error.

Our results indicate that instead of using a large solitary model, one can use the ensemble of multiple relatively smaller models, which would give similar performance but with fewer FLOPS. In practice, model ensembles can be easily parallelized (*e.g.*, using multiple accelerators), which may provide further speedup for inference. Ensembles are also highly parallelizable in training. Moreover, often the total training cost of an ensemble is much lower than that of an equally accurate solitary model.

4. From Ensembles to Cascades

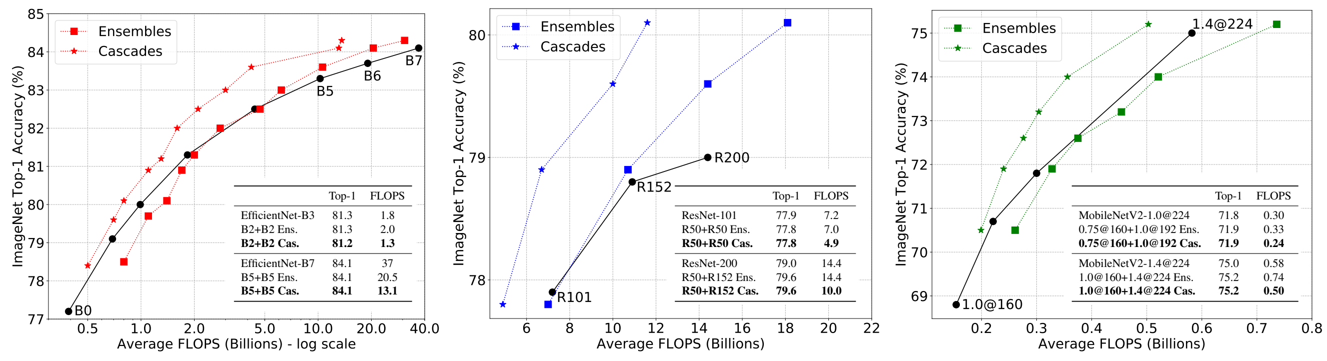
In the above we have identified the scenarios where model ensembles outperform or underperform solitary models. Specifically, ensembles are not an ideal choice when only a small amount of computation is allowed. In this section, we show that by simply converting an ensemble to a cascade, one can significantly reduce the computation without hurting the accuracy and outperform solitary models in all computation regimes.

Given an input example, we need to apply all the models in an ensemble. This can be wasteful for inputs where a subset of models are enough to classify the example correctly. Unlike ensembles, we sequentially apply models in a cascade to the input example. The prediction is progressively aggregated and early exit is allowed once the prediction is determined to be correct with high likelihood. The total computation is substantially reduced if we can accurately determine when to exit from cascades. For this purpose, we need a function to measure how likely a prediction is correct. This function is termed *confidence* and we give more details in Sec. 4.1. A formal procedure of cascades is provided in Algorithm 1.

4.1. Confidence Function

One possible choice for the confidence function is to train a policy to determine whether to move forward or stop after applying a model [12]. Huang et al. [19] use the maximum probability in the predicted distribution as the confidence measure. Streeter [31] shows that the gap between the largest and second largest logits or entropy of the distribution also indicates the prediction confidence. We choose maximum probability for its simplicity.

[1]We note that the mean of probabilities is a more general choice since logits can be arbitrarily scaled. In our experiments, we observe that they yield similar performance with the mean of logits performs marginally better. The findings in our work hold true no matter which choice is used.



	Top-1	FLOPS
EfficientNet-B3	81.3	1.8
B2+B2 Ens.	81.3	2.0
B2+B2 Cas.	**81.2**	**1.3**
EfficientNet-B7	84.1	37
B5+B5 Ens.	84.1	20.5
B5+B5 Cas.	**84.1**	**13.1**

	Top-1	FLOPS
ResNet-101	77.9	7.2
R50+R50 Ens.	77.8	7.0
R50+R50 Cas.	**77.8**	**4.9**
ResNet-200	79.0	14.4
R50+R152 Ens.	79.6	14.4
R50+R152 Cas.	**79.6**	**10.0**

	Top-1	FLOPS
MobileNetV2-1.0@224	71.8	0.30
0.75@160+1.0@192 Ens.	71.9	0.33
0.75@160+1.0@192 Cas.	**71.9**	**0.24**
MobileNetV2-1.4@224	75.0	0.58
1.0@160+1.4@224 Ens.	75.2	0.74
1.0@160+1.4@224 Cas.	**75.2**	**0.50**

(a) EfficientNet (b) ResNet (c) MobileNetV2

Figure 2: Ensembles outperform solitary models in the large computation regime and cascades show benefits in all computation regimes. Black dots represent solitary models in EfficientNet, ResNet or MobileNetV2. **Ensembles:** Ensembles outperform large solitary models, such as EfficientNet-B5/B6/B7 and ResNet-152/200, with similar FLOPS. **Cascades:** Simply converting ensembles to cascades results in a significant reduction in FLOPS while retaining the accuracy of ensembles (each star is on the left of a square). Cascades achieve better efficiency than solitary models in all computation regimes.

Formally, as above, let α be a vector representing the predicted logits. We define the confidence function $g(\cdot)\colon \mathbb{R}^N \to \mathbb{R}$ as $g(\alpha) = \max(\text{softmax}(\alpha))$. The function $g(\cdot)$ maps predicted logits α to a confidence score. The higher the confidence score $g(\alpha)$ is, the more likely the prediction is correct. We provide empirical evidence for this in supplementary materials.

For a cascade of n models $\{M_i\}$, we also need $(n-1)$ thresholds $\{t_i\}$ on the confidence score, where we use t_i to decide whether a prediction is confident enough to exit after applying model M_i (see Algorithm 1). As we define $g(\cdot)$ as the maximum probability, t_i is in $[0, 1]$. A smaller t_i indicates more images will be passed to the next model M_{i+1}. A cascade will reduce to an ensemble if all the thresholds $\{t_i\}$ are set to 1. t_n is unneeded, since the cascade will stop after applying the last model M_n, no matter how confident the prediction is.

We can flexibly control the trade-off between the computation and accuracy of a cascade through thresholds $\{t_i\}$. To understand how the thresholds influence a cascade, we visualize several 2-model cascades in Figure 3. For each cascade, we sweep t_1 from 0 and 1 and plot the results. Note that all the curves in Figure 3 have a plateau, indicating that we can significantly reduce the average FLOPS without hurting the accuracy if t_1 is properly chosen. We select the thresholds $\{t_i\}$ on held-out validation images according to the target FLOPS or validation accuracy.

4.2. Converting Ensembles to Cascades

For each ensemble in Figure 2, we convert it to a cascade that uses the same set of models. During conversion, we set the confidence thresholds such that the cascade performs similar to the ensemble while the FLOPS are minimized.

Algorithm 1 Cascades

```
Input: Models $\{M_i\}$. Thresholds $\{t_i\}$. Test image $x$.
for $k = 1, 2, \dots, n$ do
    $\alpha^{\text{cas}} = \frac{1}{k}\sum_{i=1}^{k} \alpha_i = \frac{1}{k}\sum_{i=1}^{k} M_i(x)$
    $\text{FLOPS}^{\text{cas}} = \sum_{i=1}^{k} \text{FLOPS}(M_i)$
    Early exit if confidence score $g(\alpha^{\text{cas}}) \geq t_k$
end for
Return $\alpha^{\text{cas}}$ and $\text{FLOPS}^{\text{cas}}$
```

By design in cascades some inputs incur more FLOPS than others. To compare models we consider the average FLOPS computed over all images in the test set.

We see that cascades consistently use less computation than the original ensembles and outperform solitary models in all computation regimes and for all architecture families. Taking 2 EfficientNet-B2 as an example (see Figure 2a), the ensemble initially obtains a similar accuracy to B3 but uses more FLOPS. After converting this ensemble to a cascade, we successfully reduce the average FLOPS to 1.3B (1.4x speedup compared to B3) while still maintaining the accuracy of B3. Compared to small MobileNetV2 models (FLOPS $<$ 1B) in Figure 2c, cascades can also achieve higher accuracy with fewer average FLOPS.

5. Model Selection for Building Cascades

The cascades in Figure 2 do not optimize the choice of models, and directly use the set of models in the original ensembles. For best performance, we show that one can design cascades to match a specific target FLOPS or accuracy, by selecting models to be used in the cascade.

Let $\mathcal{M}$ be the set of available models, *e.g.*, models in the

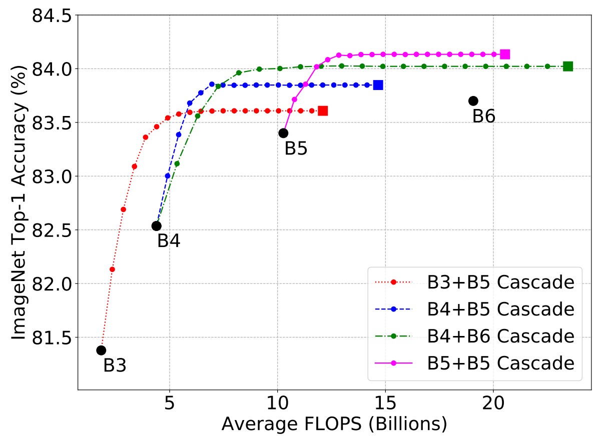


Figure 3: Cascades with different confidence thresholds. Each black dot is a solitary model and each square is an ensemble of models. Each dot represents a cascade with a specific $t_1 (0 \leq t_1 \leq 1)$. As t_1 increases from 0 to 1, the cascade changes from a solitary model (first model in the cascade; $t_1 = 0$) to the ensemble ($t_1 = 1$). The plateau in each curve indicates that the cascade can achieve a similar accuracy to the ensemble with much less computation.

EfficientNet family. Given a target FLOPS target β, we select n models $M = \{M_i \in \mathcal{M}\}$ and confidence thresholds $T = \{t_i\}$ by solving the following problem:

$$\begin{aligned} \max_{\{M_i \in \mathcal{M}\}, \{t_i\}} \quad & \text{Accuracy}\left(\mathcal{C}\left(M, T\right)\right) \\ s.t. \quad & \text{FLOPS}\left(\mathcal{C}\left(M, T\right)\right) \leq \beta, \end{aligned} \tag{1}$$

where $\mathcal{C}\left(M, T\right)$ is the cascade of models $\{M_i\}$ with thresholds $\{t_i\}$, $\text{Accuracy}(\cdot)$ gives the validation accuracy of a cascade, and $\text{FLOPS}(\cdot)$ gives the average FLOPS.

Similarly, given a target validation accuracy γ, we can build a cascade whose validation accuracy is at least γ while minimizing the FLOPS by solving the following problem:

$$\begin{aligned} \min_{\{M_i \in \mathcal{M}\}, \{t_i\}} \quad & \text{FLOPS}\left(\mathcal{C}\left(M, T\right)\right) \\ s.t. \quad & \text{Accuracy}\left(\mathcal{C}\left(M, T\right)\right) \geq \gamma. \end{aligned} \tag{2}$$

Note that this optimization is done after all models in $\mathcal{M}$ were independently trained. The difficulty of solving Eq. 1 or Eq. 2 depends on the size of $\mathcal{M}$ and the number of models in the cascade n. The problem will be challenging if $|\mathcal{M}|$ or n is large. In our experiments, $|\mathcal{M}|$ and n are not prohibitive, *e.g.*, $|\mathcal{M}| = 8$ and $n \leq 4$ for EfficientNet family. We are therefore able to solve the optimization problem with exhaustive search. One can use more efficient procedures such as described in [31]. Please see supplementary materials for more details.

5.1. Targeting for a Specific FLOPS or Accuracy

For each solitary model, we search for a cascade to match its FLOPS (middle column in Table 1) or its accuracy (right column in Table 1). We consider all models in the same family as the set of available models. The same model type is allowed to be used for multiple times in a cascade but they will be different models trained separately. For ImageNet experiments, the search is conducted on a small set of held-out training images and cascades are evaluated on the original validation set. We provide more experimental details in supplementary materials.

As shown in Table 1, cascades outperform solitary models in all three architecture families and in all computation regimes. For small models, we can outperform MobileNetV2-1.0@224 by 1.4% using equivalent FLOPS. For large models, the cascade can match the accuracy of EfficientNet-B7 with only 6.9B FLOPS (a 5.4x speedup). Table 1 further substantiates our finding that cascades can be more accurate and efficient than solitary models.

To better understand how a cascade works, we compute the exit ratio of the cascade, *i.e.*, the percentage of images that exit from the cascade at each stage. The cascade above that matches B7 accuracy contains four models: B3, B5, B5, and B5. The exit ratios for them are 67.3%, 21.6%, 5.6% and 5.5%, respectively. In this cascade, 67.3% images only consume the cost of B3 and only 5.5% images use all four models. This saves a large amount of computation compared to using B7 for all the images. We provide similar information for more cascades in supplementary materials.

5.2. Guarantee for Worst-case FLOPS

Up until now we have been measuring the computation of a cascade using the average FLOPS across all images. But for some images, it is possible that all the models in the cascade need to be applied. In this case, the average FLOPS cannot fully indicate the computational cost of a cascade. Therefore, we now consider worst-case FLOPS of a cascade, the sum of FLOPS of all models in the cascade.

For the three cascades in Table 1 that matches the accuracy of B5, B6 or B7, we report their worst-case FLOPS in Table 2. We notice that the cascade for B7 uses fewer FLOPS than B7 in the worst-case scenario, but the cascades for B5 and B6 have higher worst-case FLOPS than the networks they compare to (see 'w/o' in Table 2).

We can find cascades with guarantee for worst-case FLOPS by adding another constraint to Eq. 1 or Eq. 2: $\sum_i \text{FLOPS}(M_i) \leq \beta^{\text{worst}}$, where β^{worst} is the upper bound on the worst-case FLOPS of the cascade. With the new condition, we re-select models in the cascades for B5 or B6. As shown in Table 2, compared to solitary models, the new cascades achieve a significant speedup in average-case FLOPS and also ensure its worst-case FLOPS are smaller.

	Solitary Models		Cascades - Similar FLOPS			Cascades - Similar Accuracy		
	Top-1 (%)	FLOPS (B)	Top-1 (%)	FLOPS (B)	ΔTop-1	Top-1 (%)	FLOPS (B)	Speedup
EfficientNet								
B1	79.1	0.69	**80.1**	0.67	**1.0**	79.3	**0.54**	**1.3x**
B2	80.0	1.0	**81.2**	1.0	**1.2**	80.1	**0.67**	**1.5x**
B3	81.3	1.8	**82.4**	1.8	**1.1**	81.4	**1.1**	**1.7x**
B4	82.5	4.4	**83.7**	4.1	**1.2**	82.6	**2.0**	**2.2x**
B5	83.3	10.3	**84.4**	10.2	**1.1**	83.4	**3.4**	**3.0x**
B6	83.7	19.1	**84.6**	17.5	**0.9**	83.7	**4.1**	**4.7x**
B7	84.1	37	**84.8**	39.0	**0.7**	84.2	**6.9**	**5.4x**
ResNet								
R101	77.9	7.2	**79.3**	7.3	**1.4**	78.2	**4.9**	**1.5x**
R152	78.8	10.9	**80.1**	10.8	**1.3**	78.9	**6.2**	**1.8x**
R200	79.0	14.4	**80.4**	14.2	**1.3**	79.2	**6.8**	**2.1x**
MobileNetV2								
1.0@160	68.8	0.154	**69.5**	0.153	**0.6**	69.1	**0.146**	**1.1x**
1.0@192	70.7	0.22	**71.8**	0.22	**1.1**	70.8	**0.18**	**1.2x**
1.0@224	71.8	0.30	**73.2**	0.30	**1.4**	71.8	**0.22**	**1.4x**
1.4@224	75.0	0.58	**76.1**	0.56	**1.1**	75.1	**0.43**	**1.4x**

Table 1: Cascades of EfficientNet, ResNet or MobileNetV2 models on ImageNet. The middle column shows that when using similar FLOPS, cascades can obtain better accuracy than solitary models. The right column shows that cascades can match the accuracy of solitary models with significantly fewer FLOPS (*e.g.*, 5.4x fewer for B7). The benefit of cascades generalizes to all three architecture families and all computation regimes.

	Top-1 (%)	Average-case FLOPS (B)	Worst-case FLOPS (B)	Average-case Speedup
B5	83.3	10.3	10.3	
w/o*	83.4	3.4	14.2	3.0x
with	83.3	3.6	**9.8**	2.9x
B6	83.7	19.1	19.1	
w/o*	83.7	4.1	25.9	4.7x
with	83.7	4.2	**15.0**	4.5x
B7	84.1	37	37	
with*	84.2	6.9	**32.6**	5.4x

* Cascades from Table 1. The cascade for B7 that was only optimized to use minimal average-case FLOPS, but it can also uses fewer FLOPS than B7 in the worst-case scenario.

Table 2: Cascades can be built with guarantee for worst-case FLOPS. We use 'with' or 'w/o' to indicate whether a cascade can provide guarantee for worst-case FLOPS. Cascades with such guarantee are assured to use fewer FLOPS than solitary models in the worst-case scenario, and also achieve a considerable speedup in average-case FLOPS.

5.3. Cascades can be Scaled Up

One appealing property of solitary models is that they can be easily scaled up or down based on the available computational resources one has. We show that such property is also applicable to cascades, *i.e.*, we can scale up a base cascade to respect different FLOPS constraints. This avoids the model selection procedure when designing cascades for different FLOPS, which is required for cascades in Table 1.

Specifically, we build a 3-model cascade to match the FLOPS of EfficientNet-B0. We call this cascade C0. Then, simply by scaling up the architectures in C0, we obtain a family of cascades C0 to C7 that have increasing FLOPS and accuracy. The models in C0 are from the EfficientNet family. We include more details of how we build and scale up C0 in supplementary materials. The results of C0 to C7 in Table 3 show that simply scaling up C0 gives us a family of cascades that consistently outperform solitary models in all computation regimes. This finding enhances the practical usefulness of cascades as one can select cascades from this family based on available resources, without worrying about what models should be used in the cascade.

6. Self-cascades

Cascades typically contains multiple models. This requires training multiple models and combining them after training. What about when only one model is available? We demonstrate that one can convert a single model into a cascade by passing the same input image at different resolutions to the model. Here, we leverage the fact that resizing

Model	Top-1 (%)	FLOPS (B)	ΔTop-1	Model	Top-1 (%)	FLOPS (B)	ΔTop-1
C0	**78.1**	**0.41**		**C1**	**80.3**	**0.71**	
EfficientNet-B0 [36]	77.1	0.39	1.0	EfficientNet-B1 [36]	79.1	0.69	1.2
MnasNet-A3 [35]	76.7	0.40	1.4	**C2**	**81.2**	**1.0**	
FiGS [5]	74.0	0.40	4.1	EfficientNet-B2 [36]	80.0	1.0	1.2
NASNet-$A_{4@1056}$ [48]	74.0	0.56	4.1	**C3**	**82.2**	**1.8**	
DARTS [23]	73.3	0.57	4.8	EfficientNet-B3 [36]	81.3	1.8	0.9
MobileNetV1 1.0x [15]	70.9	0.57	7.2	NASNet-$A_{5@1538}$ [48]	78.6	2.4	3.6
MobileNetV2 1.4x [28]	75.0	0.58	3.1	ResNet-50 [13]	76.2	3.5	6.0
MobileNetV3 1.25x [14]	76.6	0.36	1.5	DenseNet-169 [19]	76.2	3.5	6.0
C4	**83.7**	**4.2**		**C5**	**84.3**	**10.2**	
EfficientNet-B4 [36]	82.5	4.4	1.2	EfficientNet-B5 [36]	83.3	10.3	1.0
NASNet-$A_{7@1920}$ [48]	80.8	4.9	2.9	ResNet-152 [13]	78.8	10.9	5.5
Inception-v3 [34]	78.8	5.7	4.9	ResNet-200 [13]	79.0	14.4	5.3
DenseNet-264 [19]	77.9	6.0	5.8	Inception-v4 [32]	80.0	13	4.3
ResNet-101 [13]	77.9	7.2	5.8	Inception-ResNet-v2 [32]	80.1	13	4.2
C6	**84.6**	**18.7**		**C7**	**84.8**	**32.6**	
EfficientNet-B6 [36]	83.7	19.1	0.9	EfficientNet-B7 [36]	84.1	37	0.7
PNASNet [22]	82.9	25.0	1.7	ResNeXt-101 [45]	80.9	32	3.9
AmoebaNet-A [26]	82.8	23.1	1.8	AmoebaNet-C [8]	83.5	41	1.3
NASNet-$A_{6@4032}$ [48]	82.7	23.8	1.9	SENet [16]	82.7	42	2.1

Table 3: A Family of Cascades C0 to C7. C0 to C7 significantly outperform other solitary models in all computation regimes. Within each block in the table, the cascade uses the fewest FLOPS while obtaining the highest accuracy. This shows that the cascades can also be scaled up or down to respect different FLOPS constraints as solitary models do. This is helpful for avoiding the model selection procedure when designing cascades for different FLOPS.

EfficientNet	Top-1 (%)	FLOPS (B)	Self-cascades	Top-1 (%)	FLOPS (B)	Speedup
B2	80.0	1.0	B1-240-300	80.1	**0.85**	**1.2x**
B3	81.3	1.8	B2-260-380	81.3	**1.6**	**1.2x**
B4	82.5	4.4	B3-300-456	82.5	**2.7**	**1.7x**
B5	83.3	10.3	B4-380-600	83.4	**6.0**	**1.7x**
B6	83.7	19.1	B5-456-600	83.8	**12.0**	**1.6x**
B7	84.1	37	B6-528-600	84.1	**22.8**	**1.6x**

Table 4: Self-cascades. In the column of self-cascades, the two numbers represent the two resolutions r_1 and r_2 used in the cascade. Self-cascades easily outperform solitary models, *i.e.*, use fewer FLOPS while obtaining a similar accuracy.

an image to a higher resolution than the model is trained on often yields higher accuracy [37] at the cost of more computation. We call such cascades as "self-cascades" since these cascade only contains the model itself.

Formally, given a model M trained on the resolution r_1, we build a 2-model cascade, where the first model is applying M to an image at resolution r_1 and the second model is applying M to the same image at resolution $r_2 (r_2 > r_1)$. The idea of self-cascades generalize to three or more resolutions, but we find that two resolutions already yield satisfactory results and more resolutions yield diminishing returns.

We build self-cascades using EfficientNet models. Since each EfficientNet model is defined with a specific image resolution (*e.g.*, 240 for B1), we set r_1 to its original resolution and set r_2 to a higher resolution. We set the confidence threshold such that the self-cascade matches the accuracy of a solitary model.

Table 4 shows that self-cascades easily outperform solitary models, *i.e.*, obtaining a similar accuracy with fewer FLOPS. Table 4 also suggests that if we want to obtain B7 accuracy, we can train a B6 model and then build a self-cascade, which not only uses much fewer FLOPS during inference, but also takes much shorter time to train.

Self-cascades provide a way to convert one single model to a cascade, which can be more efficient than solitary models. The conversion is almost free and does not require training any additional models.

	Solitary Models		**Cascades - Similar FLOPS**			**Cascades - Similar Accuracy**		
	Top-1 (%)	FLOPS (B)	Top-1 (%)	FLOPS (B)	ΔTop-1	Top-1 (%)	FLOPS (B)	Speedup
X3D-M	78.8	6.2	**80.3**	5.7	**1.5**	79.1	**3.8**	**1.6x**
X3D-L	80.6	24.8	**82.7**	24.6	**2.1**	80.8	**7.9**	**3.2x**
X3D-XL	81.9	48.4	**83.1**	38.1	**1.2**	81.9	**13.0**	**3.7x**

Table 5: Cascades of X3D models on Kinetics-600. Our cascade achieves state-of-the-art 83.1% accuracy.

	mIoU	FLOPS (B)	Speedup
ResNet-50	77.1	348	-
ResNet-101	78.1	507	-
Cascade - full	78.4	568	0.9x
Cascade - $r = 512$	78.1	439	1.2x
Cascade - $r = 128$	78.2	**398**	**1.3x**

Table 6: Cascades of DeepLabv3 models on Cityscapes.

7. Applicability beyond Image Classification

We apply cascades to vision tasks beyond image classification to further demonstrate its benefits. Specifically, we consider video classification on Kinetics-600 [4] and semantic segmentation on Cityscapes [7].

7.1. Video Classification

Similar to image classification, a video classification model outputs a vector of logits over possible classes. We use the same procedure as image classification to build cascades of video classification models.

We consider the X3D [9] architecture family for video classification, which is the state-of-the-art in terms of both the accuracy and efficiency. The X3D family contains a series of models of different sizes. Specifically, we build cascades of X3D models to match the FLOPS or accuracy of X3D-M, X3D-L or X3D-XL on Kinetics-600 [4].

The results are summarized in Table 5, where cascades significantly outperform the original X3D models. Our cascade outperforms X3D-XL by 1.2% while using fewer average FLOPS, and achieves 83.1% accuracy, a new state-of-the-art on Kinetics-600. Our cascade can also match the accuracy of X3D-XL with 3.7x fewer average FLOPS.

For clarity, the FLOPS in Table 5 are the inference cost for a single clip. Following Feichtenhofer [9], we sample 30 clips from each input video during inference. Please refer to supplementary materials for more experimental details.

7.2. Semantic Segmentation

In semantic segmentation, models predict a vector of logits for each pixel in the image. This differs from image classification, where the model makes a single prediction for the entire image. We therefore revisit the confidence function definition to handle such dense prediction tasks.

Similar to before, we use the maximum probability to measure the confidence of the prediction for a single pixel p, *i.e.*, $g(\alpha_p) = \max(\text{softmax}(\alpha_p))$, where α_p is the predicted logits for pixel p. Next, we need a function $g^{\text{dense}}(\cdot)$ to rate the confidence of the dense prediction for an image, so that we can decide whether to apply the next model to this image based on this confidence score. For this purpose, we define $g^{\text{dense}}(\cdot)$ as the average confidence score of all the pixels in the image: $g^{\text{dense}}(R) = \frac{1}{|R|}\sum_{p \in R} g(\alpha_p)$, where R represents the input image.

We notice that many pixels are unlabeled in semantic segmentation datasets, *e.g.*, Cityscapes [7], and are ignored during training and evaluation. These unlabeled pixels may introduce noise when we average the confidence score of all the pixels. To filter out unlabeled pixels in the image, we only consider pixels whose confidence is higher than a preset threshold t^{unlab}. So we update the definition of $g^{\text{dense}}(\cdot)$ as follows: $g^{\text{dense}}(R) = \frac{1}{|R'|}\sum_{p \in R'} g(\alpha_p)$, where $R' = \{p \mid g(\alpha_p) > t^{\text{unlab}}, p \in R\}$.

In a cascade of segmentation models, we decide whether to pass an image R to the next model based on $g^{\text{dense}}(\cdot)$. Since the difficulty to label different parts in one image varies significantly, *e.g.*, roads are easier to segment than traffic lights, making a single decision for the entire image can be inaccurate and lead to a waste of computation. Therefore, in practice, we divide an image into grids and decide whether to pass each grid to the next model separately.

We conduct experiments on Cityscapes [7] and use mean IoU (mIoU) as the metric. We build a cascade of DeepLabv3-ResNet-50 and DeepLabv3-ResNet-101 [6] and report the reults in Table 6. r is the size of the grid. The full image resolution is 1024×2048, so $r = 512$ means each image is divided into a grid of 8 cells. We observe that if we operate on the full image level ('full'), the cascade will use more FLOPS than ResNet-101. But if operating on the grid level, the cascade can successfully reduce the computation without hurting the performance. For example, the smaller grid size ('$r = 128$') yields 1.3x reduction in FLOPS while matching the mIoU of ResNet-101.

8. Discussion

We demonstrate that committee-based models, *i.e.*, model ensembles or cascades, provide a complementary paradigm to obtain efficient and accurate models with no architectural tuning. Notably, model cascades achieve superior efficiency and accuracy over state-of-the-art solitary models on several vision tasks. Our findings are informative for future work on optimizing network architectures.

The fact that these simple techniques outperform more sophisticated and costly architecture search methods, as well as manually designed architectures, should motivate researchers to include them as strong baselines whenever presenting a new architecture. For ML practitioners, committee-based models outline a simple procedure to improve accuracy while maintaining efficiency.

One practical concern is that the difficulty to batch examples when using model cascades may hurt the utilization of modern accelerators. Cascades are particularly useful for offline processing of large-scale data. For example, if in order to process all frames in a large video dataset, we can first apply a cheap model to all frames and then select a subset of frames based on the prediction confidence to apply more expensive models. Here, all the processing can be batched to fully utilize the accelerators. Cascades are also useful for applications with a fixed batch size, *e.g.*, robots processing one frame at a time, or mobile phone cameras processing a single image [40]. Cascades with guarantee for worst-case FLOPS can be used in applications with strict requirement on the response time.

9. Acknowledgement

We would like to thank Alex Alemi, Eric Christiansen, Sergey Ioffe, Shankar Krishnan, Max Moroz, and Matthew Streeter for their valuable help and feedback during the development of this work.

10. Contributions

Xiaofang wrote most of the code and paper, and ran all of the experiments. Elad and Yair advised on formulating the research question and plan. Dan generated the predictions of X3D on Kinetics-600. Kris helped in writing the paper and provided general guidance.

References

[1] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for efficient inference. In *ICML*, 2017. 2

[2] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996. 2

[3] Shengcao Cao, Xiaofang Wang, and Kris M. Kitani. Learnable embedding space for efficient neural architecture compression. In *ICLR*, 2019. 2

[4] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about kinetics-600. *arXiv:1808.01340*, 2018. 2, 8, 13

[5] Shraman Ray Chaudhuri, Elad Eban, Hanhan Li, Max Moroz, and Yair Movshovitz-Attias. Fine-grained stochastic architecture search. *arXiv:2006.09581*, 2020. 1, 2, 7

[6] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv:1706.05587*, 2017. 8, 13

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 2, 8, 13

[8] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. In *CVPR*, 2019. 7

[9] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *CVPR*, 2020. 2, 8, 13

[10] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv:1912.02757*, 2019. 2

[11] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997. 2

[12] Jiaqi Guan, Yang Liu, Qiang Liu, and Jian Peng. Energy-efficient amortized inference with cascaded deep classifiers. In *IJCAI*, 2018. 2, 3

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2, 3, 7, 11

[14] 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 *ICCV*, 2019. 1, 2, 7

[15] 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:1704.04861*, 2017. 1, 2, 7

[16] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 7

[17] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. In *ICLR*, 2018. 2

[18] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. In *ICLR*, 2017. 2

[19] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 3, 7

[20] Dan Kondratyuk, Mingxing Tan, Matthew Brown, and Boqing Gong. When ensembling smaller models is more efficient than single large models. *arXiv:2005.00570*, 2020. 2

[21] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*, 2017. 2

[22] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *ECCV*, 2018. 7

[23] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *ICLR*, 2019. 7

[24] Ekaterina Lobacheva, Nadezhda Chirkova, Maxim Kodryan, and Dmitry P Vetrov. On power laws in deep ensembles. In *NeurIPS*, 2020. 2

[25] Zhaofan Qiu, Dong Li, Yehao Li, Qi Cai, Yingwei Pan, and Ting Yao. Trimmed action recognition, dense-captioning events in videos, and spatio-temporal action localization with focus on activitynet challenge 2019. *arXiv:1906.07016*, 2019. 2

[26] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI*, 2019. 7

[27] 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. *IJCV*, 2015. 2, 3

[28] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 1, 2, 3, 7, 11

[29] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990. 2

[30] 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. In *ICLR*, 2017. 3

[31] Matthew Streeter. Approximation algorithms for cascading prediction models. In *ICML*, 2018. 2, 3, 5

[32] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017. 1, 7

[33] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 2

[34] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 7

[35] 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 *CVPR*, 2019. 2, 7

[36] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 1, 2, 3, 7, 11, 12

[37] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy. In *NeurIPS*, 2019. 7

[38] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *ECCV*, 2018. 3

[39] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001. 2

[40] Neal Wadhwa, Rahul Garg, David E Jacobs, Bryan E Feldman, Nori Kanazawa, Robert Carroll, Yair Movshovitz-Attias, Jonathan T Barron, Yael Pritch, and Marc Levoy. Synthetic depth-of-field with a single-camera mobile phone. *ACM Transactions on Graphics (TOG)*, 37(4):1–13, 2018. 9

[41] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *ECCV*, 2018. 3

[42] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *ICLR*, 2020. 2

[43] Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. In *NeurIPS*, 2020. 2

[44] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *CVPR*, 2018. 3

[45] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 7

[46] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018. 1

[47] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012. 3

[48] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018. 1, 2, 7

A. Efficiency of Ensembles

A.1. Experimental Details

We conduct experiments on three architecture families: EfficientNet [36], ResNet [13] and MobileNetV2 [28]. The EfficientNet family consists of eight networks: B0 to B7. We consider four networks in the ResNet family: ResNet-50/101/152/200. For MobileNetV2, we consider MobileNetV2-0.75@160, 1.0@160, 1.0@192, 1.0@224, and 1.4@224. Each MobileNetV2 network is represented in the form of $w@r$, where w is the width multiplier and r is the image resolution.

For EfficientNet, we consider ensembles of two to four models of either the same or different architectures. When the same architecture is used multiple times in an ensemble, *e.g.*, B5 and B5, they are different models trained separately from different random initializations. The FLOPS range of ResNet or MobileNetV2 models is relatively narrow compared to EfficientNet, so we only consider ensembles of two models for ResNet and MobileNetV2.

A.2. Training Time of Ensembles

In Section 3, we show that ensembles match the accuracy of large solitary models with fewer inference FLOPS. We now show that the total training cost of an ensemble can also be lower than an equally accurate solitary model.

We show the training time of solitary EfficinetNet models in Table A. We use 32 TPUv3 cores to train B0 to B5, and 128 TPUv3 cores to train B6 or B7. All the models are trained with the public official implementation of EfficientNet. We choose the ensemble that matches the accuracy of B6 or B7 and compute the total training time of the ensemble based on Table A. As shown in Table B, the ensemble of 2 B5 can match the accuracy of B7 while being faster in both training and inference.

B0	B1	B2	B3	B4	B5	B6	B7
9	12	15	24	32	48	128	160

Table A: Training time (TPUv3 days) of EfficientNet.

	Top-1 (%)	FLOPS (B)	Training
B6	83.7	19.1	128
B3+B4+B4	83.6	10.6	88
B7	84.1	37	160
B5+B5	84.1	20.5	96
B5+B5+B5	84.3	30.8	144

Table B: Training time (TPU v3 days) of ensembles. We use the '+' notation to indicate the models in enmsebles.

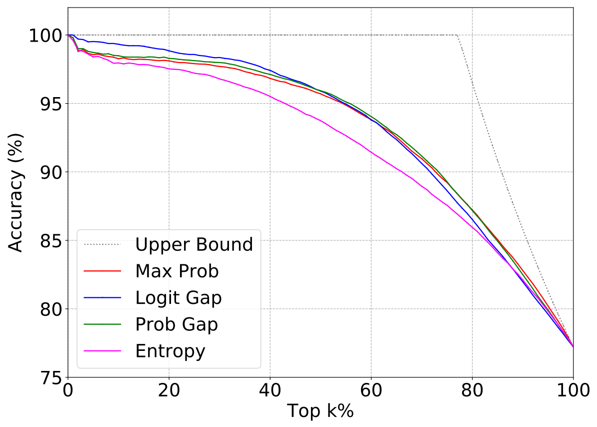


Figure A: Different metrics for the confidence function. When $k = 100$, all the images are selected so the accuracy is exactly the accuracy of EfficientNet-B0 (77.1%). The 'Upper Bound' curve represents the best possible performance for the metric. It has 100% accuracy when $k \leq 77.1$, *i.e.*, all the selected images are correctly classified. The accuracy starts to drop when k becomes larger, since some misclassified images are inevitably chosen.

B. From Ensembles to Cascades

B.1. Confidence Function

We empirically show that the higher the confidence score $g(\alpha)$ is, the more likely the prediction given by α is correct. In Sec 4.1, we define the confidence function as the maximum probability in the predicted distribution. Other than that, the following metrics can also indicate the prediction confidence: the gap between the largest and second largest logits, the gap between the largest and second largest probabilities, and the (negative) entropy of the distribution.

Given a set of images, we select the top-$k\%$ images with highest confidence scores. Then we compute the classification accuracy within the selected images. If a higher confidence score indicates that the prediction is more likely to be correct, the accuracy should drop as as k increases. For a EfficientNet-B0 model trained on ImageNet, we sweep k from 0 to 100 and compute the accuracy within the selected top-$k\%$ images from the ImageNet validation set.

As shown in Figure A, the accuracy drops as more images are selected. This empirically shows that predictions with a higher confidence score are more likely to be correct. All the metrics demonstrate reasonably good performance, where the entropy performs slightly worse than other metrics. The logit gap performs the best when $k < 60$. When k becomes larger, the maximum probability and probability gap performs the best.

B.2. Determine the Thresholds

Given the n models $\{M_i\}$ in a cascade, we also need $(n-1)$ thresholds $\{t_i\}$ on the confidence score. We can flexibly control the trade-off between the computation and accuracy of a cascade through thresholds $\{t_i\}$.

We determine the thresholds $\{t_i\}$ based on the target FLOPS or accuracy on validation images. Note that The thresholds are determined after all models are trained. We only need the logits of validation images to determine $\{t_i\}$, so computing the cascade for a specific choice of thresholds is fast. Therefore we can select the thresholds by enumerating all possible combinations for $\{t_i\}$. As t_i is a real number, we make sure two trials of t_i are sufficiently different by only considering the percentiles of confidence scores as possible values. When $n > 2$, there might be multiple choices of $\{t_i\}$ that can give the target FLOPS or accuracy. In that case, we choose $\{t_i\}$ that gives the higher accuracy or fewer FLOPS. Many choices for $\{t_i\}$ can be easily ruled out as the FLOPS or accuracy of a cascade changes monotonically with respect to any threshold t_i.

In practice, we often want the accuracy of a cascade to match the accuracy of a solitary model. To do that, we determine the thresholds such that the cascade matches the accuracy of the solitary model on *validation* images. Such thresholds usually enable the cascade to have a similar *test* accuracy with the solitary model.

For ImageNet, we randomly select ~25k training images and exclude them from training. We use these held-out training images to determine the thresholds and select models for a cascade (solving Eq. 1 or Eq. 2) in the following experiments. The final accuracy is computed on the original ImageNet validation set.

C. Model Selection for Building Cascades

C.1. Targeting for a Specific FLOPS or Accuracy

We can build a cascade targeting for a specific FLOPS or accuracy by solving Eq. 1 or Eq. 2. Note that this optimization is done after all models in M are trained. The complexity of solving Eq. 1 or Eq. 2 is exponential in $|\mathcal{M}|$ and n, and the problem will be challenging if $|\mathcal{M}|$ and n are large. In our experiments, $|\mathcal{M}|$ and n are not prohibitive. Therefore, we solve the optimization problem with exhaustive search.

Concretely, for EfficientNet, $|\mathcal{M}| = 8$ and $n \leq 4$; for ResNet, $|\mathcal{M}| = 4$ and $n = 2$; for MobileNetV2, $|\mathcal{M}| = 5$ and $n = 2$. The networks in each family are the same as listed in Sec. A.1. We do not consider cascades of more than four EfficientNet models because we observe diminishing returns as n increases. Same as ensembles in Sec A.1, we only consider cascades of two ResNet or MobileNetV2 models due to their relative narrow FLOPS range.

We allow the same network to be used by multiple times in one cascade but they will be different models trained separately. Taking EfficientNet for an example, we have in total 4672 $(= 8^4 + 8^3 + 8^2)$ possible combinations of models. In practice, we first train each EfficientNet model separately for four times and pre-compute their predicted logits. Then for each possible combination of models, we load the logits of models and determine the thresholds according to the target FLOPS or accuracy. Finally, we choose the best cascade among all possibe combinations. As mentioned above, we solve Eq. 1 or Eq. 2 on held-out training images for ImageNet experiments. No images from the ImageNet validation set are used when we select models for a cascade.

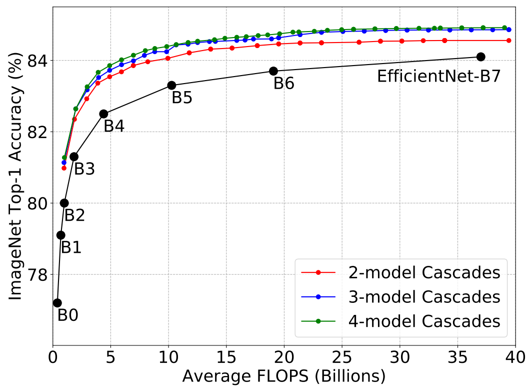


Figure B: Impact of the number of models in cascades.

C.2. Cascades can be Scaled Up

The networks in EfficientNet family are obtained by scaling up the depth, width and resolution of B0. The scaling factors for depth, width and resolution are defined as $d = \alpha^\phi$, $w = \beta^\phi$ and $r = \gamma^\phi$, respectively, where $\alpha = 1.2$, $\beta = 1.1$ and $\gamma = 1.15$, as suggested in Tan et al. [36]. One can control the network size by changing ϕ. For example, $\phi = 0$ gives B0, $\phi = 1$ gives B2, and $\phi = 7$ gives B7.

We build a 3-model cascade C0 that matches the FLOPS of EfficientNet-B0 by solving Eq. 1 on held-out training images from ImageNet. When building C0, we consider 13 networks from EfficientNet family. As we want C0 to use similar FLOPS to B0, we make sure the 13 networks include both networks smaller than B0 and networks larger than B0. Their ϕ are set to -4.0, -3.0, -2.0, -1.0, 0.0, 0.25, 0.5, 0.75, 1.0, 1.25, 1.50, 1.75, 2.0, respectively.

The ϕ of the three models in C0 are -2.0, 0.0 and 0.75. Then simply scaling up the architectures in C0, *i.e.*, increasing the ϕ of each model in C0, gives us a family of cascades C0 to C7 that have increasing FLOPS and accuracy. The thresholds in C0 to C7 are determined such that their FLOPS are similar to B0 to B7.

	Top-1 (%)	FLOPS (B)	Exit Ratio (%) at Each Stage			
			Model 1	Model 2	Model 3	Model 4
B1	79.1	0.69				
B0+B1	79.3	0.54	78.7	21.3		
B2	80.0	1.0				
B0+B1+B3	80.1	0.67	73.2	21.4	5.4	
B3	81.3	1.8				
B0+B3+B3	81.4	1.1	68.0	26.4	5.7	
B4	82.5	4.4				
B1+B3+B4	82.6	2.0	67.9	15.3	16.8	
B5	83.3	10.3				
B2+B4+B4+B4	83.4	3.4	67.6	21.2	0.0	11.2
B2+B4+B4*	83.3	3.6	57.7	26.0	16.3	
B6	83.7	19.1				
B2+B4+B5+B5	83.7	4.1	67.6	21.2	5.9	5.3
B3+B4+B4+B4*	83.7	4.2	67.3	16.2	10.9	5.6
B7	84.1	37				
B3+B5+B5+B5*	84.2	6.9	67.3	21.6	5.6	5.5

* Cascades from Table 2 with guarantee for worstcase FLOPS.

Table C: Exit ratios of cascades. We use the '+' notation to indicate the models in cascades.

C.3. Exit Ratios

To better understand how a cascade works, we compute the exit ratio of the cascade, *i.e.*, the percentage of images that exit from the cascade at each stage. Specifically, we choose the cascades in Table 1&2 that match the accuracy of B1 to B7 and include their exit ratios in Table C. For all the cascades in Table C, most images only consume the cost of the first model in the cascade and only a few images have to use all the models. This shows that cascades are able to allocate fewer resources to easy images and explains the speedup of cascades over solitary models.

C.4. Number of Models in Cascades

We study the influence of the number of models in cascades on the performance. Concretely, we consider the EfficientNet family and follow the same experimental setup as in Sec. C.1. We sweep the target FLOPS from 1 to 40 and find cascades of 2, 3, or 4 models. As shown in Figure B, the performance of cascades keeps improving as the number of models increases. We see a big gap between 2-model cascades and 3-model cascades, but increasing the number of models from 3 to 4 demonstrates a diminishing return.

D. Applicability beyond Image Classification

D.1. Video Classification

For video classification, we conduct experiments on Kinetics-600 [4]. Following Feichtenhofer [9], we sample 30 clips from each input video when evaluating X3D models on Kinetics-600. The 30 clips are the combination of 10 uniformly sampled temporal crops and 3 spatial crops. The final prediction is the mean of all individual predictions. The FLOPS in Table 5 are the inference cost for a single clip, so the total FLOPS for solitary models or cascades need to be multiplied by 30. We omit '$\times 30$' in Table 5 for brevity. This does not change the relative speedup of cascades over solitary models.

D.2. Semantic Segmentation

For semantic segmentation, we conduct experiments on the Cityscapes [7] dataset, where the full image resolution is 1024×2048. We train DeepLabv3 [6] models on the train set of Cityscapes and report the mean IoU (mIoU) over classes on the validation set. The threshold t^{unlab} to filter out unlabeled pixels is set to 0.5. For DeepLabv3-ResNet-50 or DeepLabv3-ResNet-101, we follow the original architecture of ResNet-50 or ResNet-101, except that the first 7x7 convolution is changed to three 3x3 convolutions.