EfficientNetV2: Smaller Models and Faster Training

Mingxing Tan [1] **Quoc V. Le** [1]

Abstract

This paper introduces EfficientNetV2, a new family of convolutional networks that have faster training speed and better parameter efficiency than previous models. To develop these models, we use a combination of training-aware neural architecture search and scaling, to jointly optimize training speed and parameter efficiency. The models were searched from the search space enriched with new ops such as Fused-MBConv. Our experiments show that EfficientNetV2 models train much faster than state-of-the-art models while being up to 6.8x smaller.

Our training can be further sped up by progressively increasing the image size during training, but it often causes a drop in accuracy. To compensate for this accuracy drop, we propose an improved method of progressive learning, which adaptively adjusts regularization (e.g. data augmentation) along with image size.

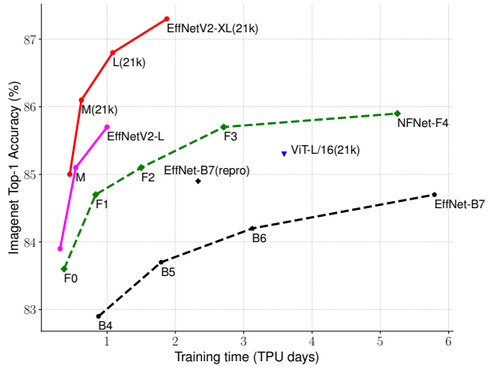
With progressive learning, our EfficientNetV2 significantly outperforms previous models on ImageNet and CIFAR/Cars/Flowers datasets. By pretraining on the same ImageNet21k, our EfficientNetV2 achieves 87.3% top-1 accuracy on ImageNet ILSVRC2012, outperforming the recent ViT by 2.0% accuracy while training 5x-11x faster using the same computing resources. Code is available at https://github.com/google/automl/tree/master/efficientnetv2.

1. Introduction

Training efficiency is important to deep learning as model size and training data size are increasingly larger. For example, GPT-3 (Brown et al., 2020), with much a larger model and more training data, demonstrates the remarkable capability in few shot learning, but it requires weeks of training with thousands of GPUs, making it difficult to retrain or improve.

Training efficiency has gained significant interests recently. For instance, NFNets (Brock et al., 2021) aim to improve training efficiency by removing the expensive batch normalization; Several recent works (Srinivas et al., 2021) focus on improving training speed by adding attention layers into convolutional networks (ConvNets); Vision Transformers (Dosovitskiy et al., 2021) improves training efficiency on large-scale datasets by using Transformer blocks. However, these methods often come with expensive overhead on large parameter size, as shown in Figure 1(b).

In this paper, we use an combination of training-aware neural architecture search (NAS) and scaling to improve both training speed and parameter efficiency. Given the parame-



(a) Training efficiency.

	EfficientNet (2019)	ResNet-RS (2021)	DeiT/ViT (2021)	EfficientNetV2 (ours)
Top-1 Acc.	84.3%	84.0%	83.1%	83.9%
Parameters	43M	164M	86M	24M

(b) Parameter efficiency.

Figure 1. **ImageNet ILSVRC2012 top-1 Accuracy vs. Training Time and Parameters** – Models tagged with 21k are pretrained on ImageNet21k, and others are directly trained on ImageNet ILSVRC2012. Training time is measured with 32 TPU cores. All EfficientNetV2 models are trained with progressive learning. Our EfficientNetV2 trains 5x - 11x faster than others, while using up to 6.8x fewer parameters. Details are in Table 7 and Figure 5.

[1]Google Research, Brain Team. Correspondence to: Mingxing Tan <tanmingxing@google.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

arXiv:2104.00298v3 [cs.CV] 23 Jun 2021

EfficientNetV2：更小模型与更快训练

谭明星 [1] 黎国伟 [1]

摘要

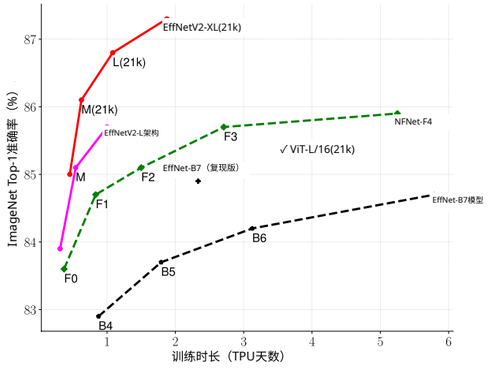
本文提出EfficientNetV2——一种训练速度更快、参数效率更优的新型卷积网络家族。我们通过训练感知的神经架构搜索与缩放策略协同优化训练速度与参数效率，并在搜索空间中引入Fused-MBConv等新型算子。实验表明EfficientNetV2模型训练速度显著优于现有技术，同时模型体积缩小最高达 $6.8\times$ 。

采用训练过程中逐步增大图像尺寸的方法可进一步提升训练速度，但常导致精度下降。为此我们提出改进型渐进学习策略，通过自适应调整正则化方法（如数据增强）与图像尺寸协同优化，有效补偿精度损失。

通过渐进式学习，我们的EfficientNetV2在ImageNet和CIFAR/Cars/Flowers数据集上显著超越先前模型。在相同ImageNet 21k 预训练条件下，EfficientNetV2在ImageNet ILSVRC2012上达到 87.3% 的top-1准确率，以 2.0% 的优势超越最新ViT模型，同时使用相同计算资源训练速度加快 $5\text{x}-11\text{x}$ 。代码详见 https://github.com/google/automl/tree/master/efficientnetv2。

1. 引言

训练效率对深度学习至关重要，因为模型规模和训练数据量正变得越来越大。例如GPT-3（Brown等人，2020）凭借更大的模型和更多训练数据，展现了小样本学习的卓越能力，但其训练耗时数周



(a) 训练效率对比

	高效网络 (2019)	残差网络-RS (2021)	(2021)	高效网络V2(本模型)
Top-1准确率	84.3%	84.0%	83.1%	83.9%
参数量	43M	164M	86M	24M

(b) 参数效率

图1. ImageNet ILSVRC2012 Top-1准确率 vs. 训练时间与参数量 - 带 21k 标记的模型经过ImageNet21k预训练，其余直接在ImageNet ILSVRC2012上训练。训练时间基于32个TPU核心测量。所有EfficientNetV2模型均采用渐进式学习。我们的EfficientNetV2训练速度比同类快 $5\text{x}-11\text{x}$ 倍，同时减少高达 6.8x 的参数量。详见表7和图5。

需动用数千块GPU，导致模型难以重新训练或改进。

训练效率近来备受关注。例如NFNets（Brock等人，2021）通过移除昂贵的批量归一化来提升训练效率；近期多项研究（Srinivas等人，2021）专注于通过向卷积网络（ConvNets）添加注意力层来加速训练；视觉Transformer（Dosovitskiy等人，2021）则采用Transformer模块提升大规模数据集上的训练效率。然而这些方法往往伴随着参数量激增的高昂代价，如图1(b)所示。

本文结合训练感知的神经架构搜索（NAS）与缩放技术，同步提升训练速度与参数效率。给定参

[1] Google Research,Brain Team. 通讯作者：谭明星 <tanmingxing@google.com>。

Proceedings of the 38^{th} 国际机器学习大会，PMLR 139，2021。

ter efficiency of EfficientNets (Tan & Le, 2019a), we start by systematically studying the training bottlenecks in EfficientNets. Our study shows in EfficientNets: (1) training with very large image sizes is slow; (2) depthwise convolutions are slow in early layers. (3) equally scaling up every stage is sub-optimal. Based on these observations, we design a search space enriched with additional ops such as Fused-MBConv, and apply training-aware NAS and scaling to jointly optimize model accuracy, training speed, and parameter size. Our found networks, named *EfficientNetV2*, train up to 4x faster than prior models (Figure 3), while being up to 6.8x smaller in parameter size.

Our training can be further sped up by progressively increasing image size during training. Many previous works, such as progressive resizing (Howard, 2018), FixRes (Touvron et al., 2019), and Mix&Match (Hoffer et al., 2019), have used smaller image sizes in training; however, they usually keep the same regularization for all image sizes, causing a drop in accuracy. We argue that keeping the same regularization for different image sizes is not ideal: for the same network, small image size leads to small network capacity and thus requires weak regularization; vice versa, large image size requires stronger regularization to combat overfitting (see Section 4.1). Based on this insight, we propose an improved method of *progressive learning*: in the early training epochs, we train the network with small image size and weak regularization (e.g., dropout and data augmentation), then we gradually increase image size and add stronger regularization. Built upon progressive resizing (Howard, 2018), but by dynamically adjusting regularization, our approach can speed up the training without causing accuracy drop.

With the improved progressive learning, our EfficientNetV2 achieves strong results on ImageNet, CIFAR-10, CIFAR-100, Cars, and Flowers dataset. On ImageNet, we achieve 85.7% top-1 accuracy while training 3x - 9x faster and being up to 6.8x smaller than previous models (Figure 1). Our EfficientNetV2 and progressive learning also make it easier to train models on larger datasets. For example, ImageNet21k (Russakovsky et al., 2015) is about 10x larger than ImageNet ILSVRC2012, but our EfficientNetV2 can finish the training within two days using moderate computing resources of 32 TPUv3 cores. By pretraining on the public ImageNet21k, our EfficientNetV2 achieves 87.3% top-1 accuracy on ImageNet ILSVRC2012, outperforming the recent ViT-L/16 by 2.0% accuracy while training 5x-11x faster (Figure 1).

Our contributions are threefold:

- We introduce EfficientNetV2, a new family of smaller and faster models. Found by our training-aware NAS and scaling, EfficientNetV2 outperform previous models in both training speed and parameter efficiency.
- We propose an improved method of progressive learning, which adaptively adjusts regularization along with image size. We show that it speeds up training, and simultaneously improves accuracy.
- We demonstrate up to 11x faster training speed and up to 6.8x better parameter efficiency on ImageNet, CIFAR, Cars, and Flowers dataset, than prior art.

2. Related work

Training and parameter efficiency: Many works, such as DenseNet (Huang et al., 2017) and EfficientNet (Tan & Le, 2019a), focus on parameter efficiency, aiming to achieve better accuracy with less parameters. Some more recent works aim to improve training or inference speed instead of parameter efficiency. For example, RegNet (Radosavovic et al., 2020), ResNeSt (Zhang et al., 2020), TResNet (Ridnik et al., 2020), and EfficientNet-X (Li et al., 2021) focus on GPU and/or TPU inference speed; NFNets (Brock et al., 2021) and BoTNets (Srinivas et al., 2021) focus on improving training speed. However, their training or inference speed often comes with the cost of more parameters. This paper aims to significantly improve both training speed and parameter efficiency than prior art.

Progressive training: Previous works have proposed different kinds of progressive training, which dynamically change the training settings or networks, for GANs (Karras et al., 2018), transfer learning (Karras et al., 2018), adversarial learning (Yu et al., 2019), and language models (Press et al., 2021). Progressive resizing (Howard, 2018) is mostly related to our approach, which aims to improve training speed. However, it usually comes with the cost of accuracy drop. Another closely related work is Mix&Match (Hoffer et al., 2019), which randomly sample different image size for each batch. Both progressive resizing and Mix&Match use the same regularization for all image sizes, causing a drop in accuracy. In this paper, our main difference is to adaptively adjust regularization as well so that we can improve both training speed and accuracy. Our approach is also partially inspired by curriculum learning (Bengio et al., 2009), which schedules training examples from easy to hard. Our approach also gradually increases learning difficulty by adding more regularization, but we don't selectively pick training examples.

Neural architecture search (NAS): By automating the network design process, NAS has been used to optimize the network architecture for image classification (Zoph et al., 2018), object detection (Chen et al., 2019; Tan et al., 2020), segmentation (Liu et al., 2019), hyperparameters (Dong et al., 2020), and other applications (Elsken et al., 2019). Previous NAS works mostly focus on improving FLOPs efficiency (Tan & Le, 2019b;a) or inference efficiency (Tan et al., 2019; Cai et al., 2019; Wu et al., 2019; Li et al., 2021).

为提升EfficientNets（Tan & Le，2019a）的训练效率，我们首先系统研究了该模型的训练瓶颈。研究发现：(1) 超大图像尺寸会导致训练速度显著下降；(2) 深度可分离卷积在浅层网络中存在计算效率问题；(3) 各阶段均匀缩放并非最优方案。基于这些发现，我们构建了融合Fused-MBConv等新型算子的搜索空间，并采用训练感知的神经架构搜索与缩放策略，联合优化模型精度、训练速度与参数量级。新提出的EfficientNetV2系列模型训练速度较前代提升 4x 倍（图3），同时参数量最多减少 6.8x 。

通过在训练过程中逐步增大图像尺寸，可进一步加速我们的训练流程。先前诸多研究（如渐进式尺寸调整（Howard, 2018）、FixRes（Touvron等人, 2019）和Mix&Match（Hoffer等人, 2019））虽采用较小训练图像尺寸，但通常对所有尺寸保持相同正则化强度，导致精度下降。我们认为：对不同图像尺寸采用相同正则化并非最优方案——同一网络在小尺寸下会降低模型容量，因而需要较弱正则化；反之大尺寸需更强正则化以抑制过拟合（见第4.1节）。基于此，我们提出改进的渐进式学习方法：在训练初期使用小尺寸图像配合弱正则化（如dropout和数据增强），随后逐步增大图像尺寸并增强正则化强度。该方法以渐进式尺寸调整（Howard, 2018）为基础，通过动态调整正则化策略，能在加速训练的同时避免精度损失。

通过改进的渐进式学习，我们的EfficientNetV2在ImageNet、CIFAR-10、CIFAR-100、Cars和Flowers数据集上取得了优异成果。在ImageNet上，我们实现了 85.7% 的top-1准确率，同时训练速度比先前模型快 $3\mathrm{x}-9\mathrm{x}$ ，模型体积缩小最高达 $6.8\mathrm{x}$ （图1）。EfficientNetV2与渐进式学习还显著提升了大规模数据集训练效率。例如ImageNet21k（Russakovsky等人，2015）规模约为ImageNet ILSVRC2012的 $10\mathrm{x}$ 倍，但我们的EfficientNetV2仅需32个TPUv3核心的中等计算资源，即可在两天内完成训练。通过在公开的ImageNet21k上进行预训练，EfficientNetV2在ImageNet ILSVRC2012上达到87.3%的top-1准确率，以 $5\mathrm{x}-11\mathrm{x}$ 倍的训练速度超越近期ViT-L/16模型2.0个百分点（图1）。

我们的贡献体现在三个方面：

- 提出EfficientNetV2——一个更小更快的模型家族。通过训练感知的神经架构搜索和缩放技术，该模型在训练速度和参数效率上均超越先前模型。
- 提出改进的渐进式学习方法，能自适应地根据图像尺寸调整正则化强度。实验表明该方法既可加速训练，又能提升准确率。
- 在ImageNet、CIFAR、Cars和Flowers数据集上，相比现有技术实现了 $11\mathrm{x}$ 倍的训练加速和 $6.8\mathrm{x}$ 倍的参数效率提升。

2. 相关工作

训练与参数效率：诸多研究如DenseNet（黄等人，2017）和EfficientNet（谭与乐， $2019\mathrm{a}$ ）聚焦参数效率，旨在以更少参数获得更高精度。近期研究则转向提升训练或推理速度而非参数效率，例如RegNet（拉多萨维奇等人，2020）、ResNeSt（张等人，2020）、TResNet（里德尼克等人，2020）和EfficientNet-X（李等人，2021）专注于GPU/TPU推理速度；NFNets（布罗克等人，2021）与BoTNets（斯里尼瓦斯等人，2021）着力提升训练速度。但这些加速往往以增加参数为代价。本文旨在显著超越现有技术，同时提升训练速度与参数效率。

渐进式训练：先前研究针对GAN（Karras等人，2018）、迁移学习（Karras等人，2018）、对抗学习（Yu等人，2019）和语言模型（Press等人，2021）提出了多种动态调整训练设置或网络的渐进训练方法。渐进式缩放（Howard，2018）与我们的方法最为相关，其目标是提升训练速度，但通常以精度下降为代价。另一密切相关的工作是Mix&Match（Hoffer等人，2019），该方法为每批次随机采样不同图像尺寸。渐进式缩放和Mix&Match对所有图像尺寸采用相同的正则化，导致精度降低。本文的核心差异在于同步自适应调整正则化策略，从而兼顾训练速度与精度提升。我们的方法也部分受课程学习（Bengio等人，2009）启发——该理论主张从易到难安排训练样本。我们通过逐步增强正则化来提高学习难度，但无需选择性筛选训练样本。

神经架构搜索(NAS)：通过自动化网络设计流程，该技术已成功应用于图像分类(Zoph等，2018)、目标检测(Chen等，2019；Tan等，2020)、分割(Liu等，2019)、超参数优化(Dong等，2020)及其他领域(Elsken等，2019)。现有研究主要聚焦于提升浮点运算效率(Tan & Le，2019b;a)或推理效率(Tan等，2019；Cai等，2019；Wu等，2019；Li等，2021)。

Unlike prior works, this paper uses NAS to optimize training and parameter efficiency.

3. EfficientNetV2 Architecture Design

In this section, we study the training bottlenecks of EfficientNet (Tan & Le, 2019a), and introduce our training-aware NAS and scaling, as well as EfficientNetV2 models.

3.1. Review of EfficientNet

EfficientNet (Tan & Le, 2019a) is a family of models that are optimized for FLOPs and parameter efficiency. It leverages NAS to search for the baseline EfficientNet-B0 that has better trade-off on accuracy and FLOPs. The baseline model is then scaled up with a compound scaling strategy to obtain a family of models B1-B7. While recent works have claimed large gains on training or inference speed, they are often worse than EfficientNet in terms of parameters and FLOPs efficiency (Table 1). In this paper, we aim to improve the training speed while maintaining the parameter efficiency.

Table 1. EfficientNets have good parameter and FLOPs efficiency.

	Top-1 Acc.	Params	FLOPs
EfficientNet-B6 (Tan & Le, 2019a)	84.6%	**43M**	**19B**
ResNet-RS-420 (Bello et al., 2021)	84.4%	192M	64B
NFNet-F1 (Brock et al., 2021)	84.7%	133M	36B

3.2. Understanding Training Efficiency

We study the training bottlenecks of EfficientNet (Tan & Le, 2019a), henceforth is also called EfficientNetV1, and a few simple techniques to improve training speed.

Training with very large image sizes is slow: As pointed out by previous works (Radosavovic et al., 2020), EfficientNet's large image size results in significant memory usage. Since the total memory on GPU/TPU is fixed, we have to train these models with smaller batch size, which drastically slows down the training. A simple improvement is to apply FixRes (Touvron et al., 2019), by using a smaller image size for training than for inference. As shown in Table 2, smaller image size leads to less computations and enables large batch size, and thus improves training speed by up to 2.2x. Notably, as pointed out in (Touvron et al., 2020; Brock et al., 2021), using smaller image size for training also leads to slightly better accuracy. But unlike (Touvron et al., 2019), we do not finetune any layers after training.

Table 2. EfficientNet-B6 accuracy and training throughput for different batch sizes and image size.

		TPUv3 imgs/sec/core		V100 imgs/sec/gpu	
	Top-1 Acc.	batch=32	batch=128	batch=12	batch=24
train size=512	84.3%	42	OOM	29	OOM
train size=380	84.6%	76	93	37	52

In Section 4, we will explore a more advanced training approach, by progressively adjusting image size and regularization during training.

Depthwise convolutions are slow in early layers but effective in later stages: Another training bottleneck of EfficientNet comes from the extensive depthwise convolutions (Sifre, 2014). Depthwise convolutions have fewer parameters and FLOPs than regular convolutions, but they often cannot fully utilize modern accelerators. Recently, Fused-MBConv is proposed in (Gupta & Tan, 2019) and later used in (Gupta & Akin, 2020; Xiong et al., 2020; Li et al., 2021) to better utilize mobile or server accelerators. It replaces the depthwise conv3x3 and expansion conv1x1 in MBConv (Sandler et al., 2018; Tan & Le, 2019a) with a single regular conv3x3, as shown in Figure 2. To systematically compares these two building blocks, we gradually replace the original MBConv in EfficientNet-B4 with Fused-MBConv (Table 3). When applied in early stage 1-3, Fused-MBConv can improve training speed with a small overhead on parameters and FLOPs, but if we replace all blocks with Fused-MBConv (stage 1-7), then it significantly increases parameters and FLOPs while also slowing down the training. Finding the right combination of these two building blocks, MBConv and Fused-MBConv, is non-trivial, which motivates us to leverage neural architecture search to automatically search for the best combination.

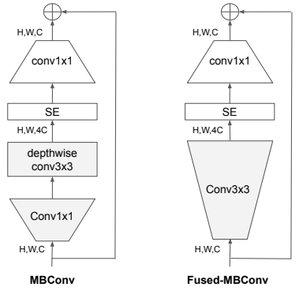


Figure 2. Structure of MBConv and Fused-MBConv.

Table 3. Replacing MBConv with Fused-MBConv. `No fused` denotes all stages use MBConv, `Fused stage1-3` denotes replacing MBConv with Fused-MBConv in stage $\{2, 3, 4\}$.

	Params (M)	FLOPs (B)	Top-1 Acc.	TPU imgs/sec/core	V100 imgs/sec/gpu
No fused	19.3	4.5	82.8%	262	155
Fused stage1-3	20.0	7.5	83.1%	362	216
Fused stage1-5	43.4	21.3	83.1%	327	223
Fused stage1-7	132.0	34.4	81.7%	254	206

Equally scaling up every stage is sub-optimal: EfficientNet equally scales up all stages using a simple compound scaling rule. For example, when depth coefficient is 2, then all stages in the networks would double the number of layers. However, these stages are not equally contributed to

与先前研究不同，本文采用神经架构搜索（NAS）来优化训练与参数效率。

3. EfficientNetV2架构设计

本节分析EfficientNet（Tan & Le, 2019a）的训练瓶颈，提出训练感知的NAS与缩放方法，并介绍EfficientNetV2模型。

3.1. EfficientNet综述

EfficientNet（Tan & Le, 2019a）是一类针对FLOPs与参数效率优化的模型家族。其通过NAS搜索获得在精度与FLOPs间平衡更优的基线模型EfficientNet-B0，再采用复合缩放策略扩展出B1-B7系列模型。尽管近期研究宣称在训练/推理速度上有显著提升，但其参数与FLOPs效率仍逊于EfficientNet（表1）。本文旨在提升训练速度的同时保持参数效率。

表1. EfficientNet系列具有优异的参数与FLOPs效率

	Top-1准确率	参数量	浮点运算量
高效网络B6（Tan & Le，2019a）	84.6%	**43M**	**19B**
残差网络RS-420（Bello等，2021）	84.4%	192M	64B
归一化自由网络F1（Brock等，2021）	84.7%	133M	36B

3.2 训练效率解析

本研究分析了EfficientNet（Tan & Le，2019a，后称EfficientNetV1）的训练瓶颈，并探讨几种提升训练速度的简易技术。

超大图像尺寸导致训练缓慢：如前人研究（Radosavovic等，2020）指出，EfficientNet的大尺寸图像会显著增加内存占用。由于GPU/TPU总内存固定，必须采用更小的批量大小进行训练，这极大拖慢了训练速度。改进方案是应用FixRes（Touvron等，2019），即训练时使用比推理更小的图像尺寸。如表2所示，较小图像尺寸既能减少计算量又可增大批量大小，从而使训练速度最高提升2.2倍。值得注意的是，（Touvron等，2020；Brock等，2021）指出训练采用较小图像尺寸还能略微提高准确率。但与（Touvron等，2019）不同，我们在训练后未对任何层进行微调。

表2. EfficientNet-B6在不同批量和图像尺寸下的准确率与训练吞吐量

		TPUv3每核心图像处理速度(张/秒)		V100每GPU图像处理速度(张/秒)	
	Top-1准确率	批量=32	批量=128	批次=12	批次=24
训练规模=512	84.3%	42	OOM	29	OOM
训练规模=380	84.6%	76	93	37	52

在第4节中，我们将探讨一种更先进的训练方法

通过在训练过程中逐步调整图像尺寸和正则化参数来实现。

深度卷积在早期层速度较慢但在后期效果显著：EfficientNet的另一个训练瓶颈源于大量使用的深度卷积(Sifre, 2014)。相比常规卷积，深度卷积虽参数量和计算量(FLOPs)更少，但往往难以充分利用现代加速器。近期研究(Gupta & Tan, 2019)提出融合MBConv结构，并在后续工作(Gupta & Akin, 2020; Xiong等, 2020; Li等, 2021)中用于优化移动端或服务器加速器性能。该结构将MBConv(Sandler等, 2018; Tan & Le, 2019a)中的深度conv3x3和扩展conv1x1替换为单个常规conv3x3（见图2）。为系统比较这两种基础模块，我们逐步将EfficientNet-B4中的原始MBConv替换为融合MBConv（表3）。当应用于早期1-3阶段时，融合MBConv能以微小参数量和计算量为代价提升训练速度；但若全部替换为融合MBConv（阶段1-7），则会显著增加参数量与计算量并降低训练速度。寻找MBConv与融合MBConv的最佳组合具有挑战性，这促使我们采用神经架构搜索来自动寻找最优配置。

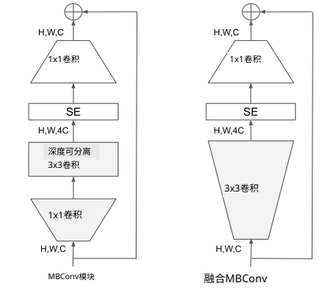


图2. MBConv与Fused-MBConv结构对比

表3. Fused-MBConv替换方案。No fused表示全部阶段使用MBConv，Fused stage 1-3表示在第 $\{2, 3, 4\}$ 阶段用Fused-MBConv替换MBConv

	参数量 (M)	浮点运算量 (B)	Top-1准确率	TPU 每核图像数/秒	V100 每GPU图像数/秒
未融合	19.3	4.5	82.8%	262	155
融合阶段1-3	20.0	7.5	83.1%	362	216
融合阶段1-5	43.4	21.3	83.1%	327	223
融合阶段1-7	132.0	34.4	81.7%	254	206

等比例缩放所有阶段并非最优解：Efficient-Net采用简单的复合缩放规则对全部阶段进行均等扩展。例如当深度系数为2时，网络中所有阶段的层数都会翻倍。然而这些阶段对网络的贡献度并不均等

the training speed and parameter efficiency. In this paper, we will use a non-uniform scaling strategy to gradually add more layers to later stages. In addition, EfficientNets aggressively scale up image size, leading to large memory consumption and slow training. To address this issue, we slightly modify the scaling rule and restrict the maximum image size to a smaller value.

3.3. Training-Aware NAS and Scaling

To this end, we have learned multiple design choices for improving training speed. To search for the best combinations of those choices, we now propose a training-aware NAS.

NAS Search: Our training-aware NAS framework is largely based on previous NAS works (Tan et al., 2019; Tan & Le, 2019a), but aims to jointly optimize accuracy, parameter efficiency, and training efficiency on modern accelerators. Specifically, we use EfficientNet as our backbone. Our search space is a stage-based factorized space similar to (Tan et al., 2019), which consists of the design choices for convolutional operation types {MBConv, Fused-MBConv}, number of layers, kernel size {3x3, 5x5}, expansion ratio $\{1, 4, 6\}$. On the other hand, we reduce the search space size by (1) removing unnecessary search options such as pooling skip ops, since they are never used in the original EfficientNets; (2) reusing the same channel sizes from the backbone as they are already searched in (Tan & Le, 2019a). Since the search space is smaller, we can apply reinforcement learning (Tan et al., 2019) or simply random search on much larger networks that have comparable size as EfficientNet-B4. Specifically, we sample up to 1000 models and train each model about 10 epochs with reduced image size for training. Our search reward combines the model accuracy A, the normalized training step time S, and the parameter size P, using a simple weighted product $A \cdot S^w \cdot P^v$, where w = -0.07 and v = -0.05 are empirically determined to balance the trade-offs similar to (Tan et al., 2019).

EfficientNetV2 Architecture: Table 4 shows the architecture for our searched model EfficientNetV2-S. Compared to the EfficientNet backbone, our searched EfficientNetV2 has several major distinctions: (1) The first difference is EfficientNetV2 extensively uses both MBConv (Sandler et al., 2018; Tan & Le, 2019a) and the newly added fused-MBConv (Gupta & Tan, 2019) in the early layers. (2) Secondly, EfficientNetV2 prefers smaller expansion ratio for MBConv since smaller expansion ratios tend to have less memory access overhead. (3) Thirdly, EfficientNetV2 prefers smaller 3x3 kernel sizes, but it adds more layers to compensate the reduced receptive field resulted from the smaller kernel size. (4) Lastly, EfficientNetV2 completely removes the last stride-1 stage in the original EfficientNet, perhaps due to its large parameter size and memory access overhead.

Table 4. EfficientNetV2-S architecture – MBConv and Fused-MBConv blocks are described in Figure 2.

Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6
5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	256	15
7	Conv1x1 & Pooling & FC	-	1280	1

EfficientNetV2 Scaling: We scale up EfficientNetV2-S to obtain EfficientNetV2-M/L using similar compound scaling as (Tan & Le, 2019a), with a few additional optimizations: (1) we restrict the maximum inference image size to 480, as very large images often lead to expensive memory and training speed overhead; (2) as a heuristic, we also gradually add more layers to later stages (e.g., stage 5 and 6 in Table 4) in order to increase the network capacity without adding much runtime overhead.

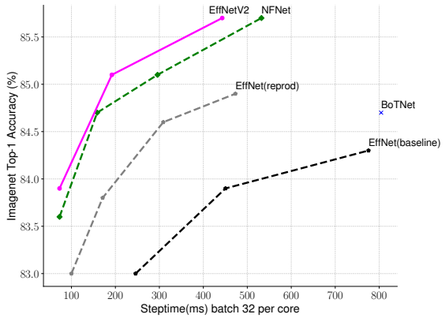


Figure 3. ImageNet accuracy and training step time on TPUv3 – Lower step time is better; all models are trained with fixed image size without progressive learning.

Training Speed Comparison: Figure 3 compares the training step time for our new EfficientNetV2, where all models are trained with fixed image size without progressive learning. For EfficientNet (Tan & Le, 2019a), we show two curves: one is trained with the original inference size, and the other is trained with about 30% smaller image size, same as EfficientNetV2 and NFNet (Touvron et al., 2019; Brock et al., 2021). All models are trained with 350 epochs, except NFNets are trained with 360 epochs, so all models have a similar number of training steps. Interestingly, we observe that when trained properly, EfficientNets still achieve pretty strong performance trade-off. More importantly, with our training-aware NAS and scaling, our proposed EfficientNetV2 model train much faster than the other recent models. These results also align with our inference results as shown in Table 7 and Figure 5.

训练速度与参数效率。本文采用非均匀缩放策略，逐步在后期阶段增加更多网络层。此外，EfficientNets大幅提升图像尺寸的策略会导致显存占用过高且训练速度缓慢。为此，我们微调缩放规则，将最大图像尺寸限制为较小值。

3.3 训练感知的神经架构搜索与缩放

基于此，我们总结出多项提升训练速度的设计方案。为寻找这些方案的最佳组合，现提出训练感知的神经架构搜索方法。

NAS搜索方法：我们的训练感知型神经架构搜索框架主要基于先前NAS研究（Tan等人，2019；Tan & Le，2019a），但致力于在现代加速器上联合优化准确率、参数效率和训练效率。具体采用EfficientNet作为主干网络，搜索空间采用与（Tan等人，2019）类似的阶段式分解空间，包含卷积操作类型{MBConv, Fused-MBConv}、层数、核尺寸 $\{3x3, 5x5\}$ 和扩展率 $\{1, 4, 6\}$ 等设计选项。我们通过以下方式缩小搜索空间：(1)移除原始EfficientNet未使用的冗余选项（如池化跳跃操作）；(2)复用主干网通道维度（该维度已在Tan & Le 2019a中完成搜索）。由于搜索空间更小，我们可在EfficientNet-B4规模的网络上应用强化学习（Tan等人，2019）或简单随机搜索。具体采样1000个模型，每个模型用小尺寸图像训练约10个周期。搜索奖励函数综合模型准确率 A、归一化训练步长时间 S 和参数量 P，采用加权乘积 $A \cdot S^w \cdot P^v$ 进行平衡，其中权重系数 $w = -0.07$ 和 $v = -0.05$ 根据经验设定以保持与（Tan等人，2019）相似的权衡关系。

表4. EfficientNetV2-S架构 - MBConv和Fused-MBConv模块结构详见图2。

阶段	操作	步长	通道数	层级结构
0	3x3卷积	2	24	1
1	融合MBConv1，3x3核	1	24	2
2	融合MBConv4，3x3核	2	48	4
3	融合MBConv4，3x3核	2	64	4
4	MBConv4，3x3核，SE0.25	2	128	6
5	MBConv6模块，3x3卷积核，SE压缩比0.25	1	160	9
6	MBConv6模块，3x3卷积核，SE压缩比0.25	2	256	15
7	1x1卷积层 & 池化层 & 全连接层	-	1280	1

EfficientNetV2缩放策略：我们采用与(Tan & Le, 2019a)类似的复合缩放方法将EfficientNetV2-S扩展为V2-M/L版本，并进行了以下优化：(1)将最大推理图像尺寸限制为480像素，因超大图像会导致显存占用和训练速度显著下降；(2)基于启发式规则，逐步在后期阶段（如表4中第5、6阶段）增加网络层数，这样可在不显著增加计算开销的前提下提升网络容量。

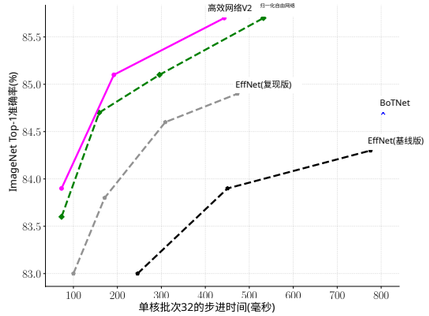


图3. TPUv3上的ImageNet准确率与训练步进时间 - 步进时间越短越好；所有模型均采用固定图像尺寸训练，未使用渐进式学习。

EfficientNetV2架构：表4展示了我们搜索得到的EfficientNetV2-S模型架构。相较于EfficientNet主干网络，我们搜索得到的EfficientNetV2具有以下主要差异：(1) 首要区别在于EfficientNetV2在浅层广泛使用了MBConv（Sandler等人，2018；Tan & Le，2019a）和新加入的fused-MBConv（Gupta & Tan，2019）模块；(2) 其次，EfficientNetV2倾向于采用更小的MBConv扩展系数，因为较小扩展系数通常具有更低的内存访问开销；(3) 第三，EfficientNetV2偏好较小的3x3卷积核尺寸，但通过增加网络层数来补偿因小卷积核导致的感受野缩小；(4) 最后，EfficientNetV2完全移除了原始EfficientNet中最后的步幅1阶段，这可能是由于其参数量过大和内存访问开销较高。

训练速度对比：图3比较了我们新型EfficientNetV2的训练步骤耗时，所有模型均采用固定图像尺寸训练且未使用渐进学习。对于EfficientNet（Tan & Le, 2019a），我们展示两条曲线：一条使用原始推理尺寸训练，另一条采用与EfficientNetV2和NFNet（Touvron等, 2019；Brock等, 2021）相同的 30% 较小图像尺寸训练。除NFNet采用360轮训练外，所有模型均训练350轮以保证训练步数相近。值得注意的是，我们发现当采用适当训练方式时，EfficientNet仍能实现优异的性能权衡。更重要的是，通过我们具有训练感知能力的神经架构搜索和缩放策略，提出的EfficientNetV2模型训练速度显著快于其他最新模型。这些结果与表7和图5所示的推理性能结果一致。

4. Progressive Learning

4.1. Motivation

As discussed in Section 3, image size plays an important role in training efficiency. In addition to FixRes (Touvron et al., 2019), many other works dynamically change image sizes during training (Howard, 2018; Hoffer et al., 2019), but they often cause a drop in accuracy.

We hypothesize the accuracy drop comes from the unbalanced regularization: when training with different image sizes, we should also adjust the regularization strength accordingly (instead of using a fixed regularization as in previous works). In fact, it is common that large models require stronger regularization to combat overfitting: for example, EfficientNet-B7 uses larger dropout and stronger data augmentation than the B0. In this paper, we argue that even for the same network, smaller image size leads to smaller network capacity and thus needs weaker regularization; vice versa, larger image size leads to more computations with larger capacity, and thus more vulnerable to overfitting.

To validate our hypothesis, we train a model, sampled from our search space, with different image sizes and data augmentations (Table 5). When image size is small, it has the best accuracy with weak augmentation; but for larger images, it performs better with stronger augmentation. This insight motivates us to adaptively adjust regularization along with image size during training, leading to our improved method of progressive learning.

Table 5. ImageNet top-1 accuracy. We use RandAug (Cubuk et al., 2020), and report mean and stdev for 3 runs.

	Size=128	Size=192	Size=300
RandAug magnitude=5	**78.3** ±0.16	81.2 ±0.06	82.5 ±0.05
RandAug magnitude=10	78.0 ±0.08	**81.6** ±0.08	82.7 ±0.08
RandAug magnitude=15	77.7 ±0.15	81.5 ±0.05	**83.2** ±0.09

4.2. Progressive Learning with adaptive Regularization

Figure 4 illustrates the training process of our improved progressive learning: in the early training epochs, we train the network with smaller images and weak regularization, such that the network can learn simple representations easily and fast. Then, we gradually increase image size but also making learning more difficult by adding stronger regularization. Our approach is built upon (Howard, 2018) that progressively changes image size, but here we adaptively adjust regularization as well.

Formally, suppose the whole training has N total steps, the target image size is S_e, with a list of regularization magnitude $\Phi_e = \{\phi_e^k\}$, where k represents a type of regularization such as dropout rate or mixup rate value. We divide the training into M stages: for each stage $1 \le i \le M$, the model is trained with image size S_i and regularization magnitude

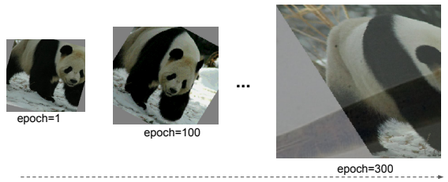


Figure 4. Training process in our improved progressive learning – It starts with small image size and weak regularization (epoch=1), and then gradually increase the learning difficulty with larger image sizes and stronger regularization: larger dropout rate, RandAugment magnitude, and mixup ratio (e.g., epoch=300).

$\Phi_i = \{\phi_i^k\}$. The last stage M would use the targeted image size S_e and regularization Φ_e. For simplicity, we heuristically pick the initial image size S_0 and regularization Φ_0, and then use a linear interpolation to determine the value for each stage. Algorithm 1 summarizes the procedure. At the beginning of each stage, the network will inherit all weights from the previous stage. Unlike transformers, whose weights (e.g., position embedding) may depend on input length, ConvNet weights are independent to image sizes and thus can be inherited easily.

Algorithm 1 Progressive learning with adaptive regularization.

Input: Initial image size S_0 and regularization $\{\phi_0^k\}$.
Input: Final image size S_e and regularization $\{\phi_e^k\}$.
Input: Number of total training steps N and stages M.
for $i = 0$ **to** $M-1$ **do**
  Image size: $S_i \leftarrow S_0 + (S_e - S_0) \cdot \frac{i}{M-1}$
  Regularization: $R_i \leftarrow \{\phi_i^k = \phi_0^k + (\phi_e^k - \phi_0^k) \cdot \frac{i}{M-1}\}$
  Train the model for $\frac{N}{M}$ steps with S_i and R_i.
end for

Our improved progressive learning is generally compatible to existing regularization. For simplicity, this paper mainly studies the following three types of regularization:

- **Dropout** (Srivastava et al., 2014): a network-level regularization, which reduces co-adaptation by randomly dropping channels. We will adjust the dropout rate γ.
- **RandAugment** (Cubuk et al., 2020): a per-image data augmentation, with adjustable magnitude ϵ.
- **Mixup** (Zhang et al., 2018): a cross-image data augmentation. Given two images with labels (x_i, y_i) and (x_j, y_j), it combines them with mixup ratio λ: $\tilde{x_i} = \lambda x_j + (1-\lambda)x_i$ and $\tilde{y_i} = \lambda y_j + (1-\lambda)y_i$. We would adjust mixup ratio λ during training.

5. Main Results

This section presents our experimental setups, the main results on ImageNet, and the transfer learning results on CIFAR-10, CIFAR-100, Cars, and Flowers.

4. 渐进式学习

4.1 研究动机

如第3节所述，图像尺寸对训练效率具有重要影响。除FixRes方法(Touvron等人，2019)外，许多研究(Howard, 2018; Hoffer等人，2019)在训练过程中动态调整图像尺寸，但这往往导致准确率下降。

我们推测准确率下降源于正则化失衡：当采用不同尺寸图像训练时，应相应调整正则化强度（而非如先前研究采用固定值）。事实上，大型模型通常需要更强正则化来防止过拟合，例如EfficientNet-B7就比B0版本采用更大的dropout和更强数据增强。本文提出：即使同一网络，较小图像尺寸会降低网络容量，因此需要较弱正则化；反之较大图像尺寸带来更多计算量和更强容量，因而更容易过拟合。

为验证假设，我们从搜索空间中采样一个模型，在不同图像尺寸和数据增强条件下进行训练（表5）。当图像尺寸较小时，弱增强能获得最佳精度；而对较大尺寸图像，强增强表现更优。这一发现促使我们在训练过程中根据图像尺寸自适应调整正则化强度，从而提出了改进的渐进式学习方法。

表5. ImageNet top-1准确率。采用RandAug（Cubuk等人，2020）方法，报告3次运行的平均值及标准差。

	尺寸=128	尺寸 = 192	尺寸=300
RandAug强度 = 5	**78.3** ±0.16	81.2 ±0.06	82.5 ±0.05
RandAug强度 = 10	78.0 ±0.08	**81.6** ±0.08	82.7 ±0.08
RandAug幅度 = 15	77.7 ±0.15	81.5 ±0.05	**83.2** ±0.09

4.2 基于自适应正则化的渐进式学习

图4展示了改进版渐进学习的训练过程：在训练初期，我们使用较小图像和弱正则化训练网络，使网络能快速掌握简单表征。随后逐步增大图像尺寸，同时通过增强正则化提升学习难度。该方法基于Howard(2018)的图像尺寸渐进变化框架，但额外引入了自适应正则化调整机制。

形式化地说，假设整个训练过程共包含 N 个步骤，目标图像尺寸为 S_e，并给定正则化强度列表 $\Phi_e = \{\phi_e^k\}$，其中 k 代表某类正则化参数（如dropout率或mixup比率值）。我们将训练划分为 M 个阶段：每个阶段 $1 \le i \le M$ 采用图像尺寸 S_i 和对应的正则化强度进行训练

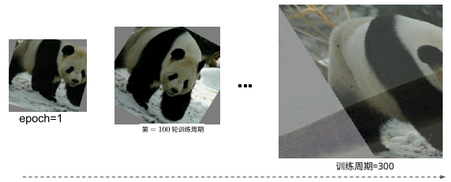


图4.改进版渐进式学习训练流程——初始阶段采用小尺寸图像和弱正则化（周期=1），随后逐步通过增大图像尺寸和增强正则化强度来提升学习难度：包括更高的dropout率、RandAugment强度及mixup混合比例（例如周期=300时）。

$\Phi_i = \{\phi_i^k\}$。最终阶段 M 将采用目标图像尺寸 S_e 和正则化参数 Φ_e。为简化流程，我们通过启发式方法选取初始图像尺寸 S_0 与正则化参数 Φ_0，随后采用线性插值确定各阶段对应数值。算法1概括了该流程。每个阶段开始时，网络将继承前一阶段所有权重。与权重可能依赖输入长度（如位置编码）的Transformer不同，卷积网络权重与图像尺寸无关，因而可轻松继承。

算法1 采用自适应正则化的渐进式学习

输入：初始图像尺寸 S_0 及正则化参数 $\{\phi_0^k\}$
输入：最终图像尺寸 S_e 及正则化参数 $\{\phi_e^k\}$
输入：总训练步数 N 与阶段数 M
循环从 $i = 0$ 到 $M-1$ 执行
  图像尺寸：$S_i \leftarrow S_0 + (S_e - S_0) \cdot \frac{i}{M-1}$
  正则化：$R_i \leftarrow \{\phi_i^k = \phi_0^k + (\phi_e^k - \phi_0^k) \cdot \frac{i}{M-1}\}$
  使用 S_i 和 R_i 训练模型 $\frac{N}{M}$ 步
循环结束

我们改进的渐进式学习方法普遍兼容现有正则化技术。为简化研究，本文主要探讨以下三类正则化方法：

- Dropout（Srivastava等人，2014）：网络级正则化技术，通过随机丢弃通道减少协同适应。我们将调整丢弃率 γ。
- RandAugment（Cubuk等人，2020）：基于单图像的数据增强方法，其增强强度 ϵ 可调。
- Mixup（张等人，2018年）：一种跨图像数据增强方法。给定两幅带标签 (x_i, y_i) 和 (x_j, y_j) 的图像，按混合比例 λ 进行合成：$\tilde{x_i} = \lambda x_j + (1-\lambda)x_i$ 与 $\tilde{y_i} = \lambda y_j + (1-\lambda)y_i$。我们会在训练过程中动态调整混合比例 λ。

5. 主要结果

本节展示实验设置、ImageNet上的主要结果，以及在CIFAR-10、CIFAR-100、Cars和Flowers数据集上的迁移学习效果。

5.1. ImageNet ILSVRC2012

Setup: ImageNet ILSVRC2012 (Russakovsky et al., 2015) contains about 1.28M training images and 50,000 validation images with 1000 classes. During architecture search or hyperparameter tuning, we reserve 25,000 images (about 2%) from the training set as `minival` for accuracy evaluation. We also use `minival` to perform early stopping. Our ImageNet training settings largely follow EfficientNets (Tan & Le, 2019a): RMSProp optimizer with decay 0.9 and momentum 0.9; batch norm momentum 0.99; weight decay 1e-5. Each model is trained for 350 epochs with total batch size 4096. Learning rate is first warmed up from 0 to 0.256, and then decayed by 0.97 every 2.4 epochs. We use exponential moving average with 0.9999 decay rate, RandAugment (Cubuk et al., 2020), Mixup (Zhang et al., 2018), Dropout (Srivastava et al., 2014), and stochastic depth (Huang et al., 2016) with 0.8 survival probability.

Table 6. Progressive training settings for EfficientNetV2.

	S		M		L	
	min	max	min	max	min	max
Image Size	128	300	128	380	128	380
RandAugment	5	15	5	20	5	25
Mixup alpha	0	0	0	0.2	0	0.4
Dropout rate	0.1	0.3	0.1	0.4	0.1	0.5

For progressive learning, we divide the training process into four stages with about 87 epochs per stage: the early stage uses a small image size with weak regularization, while the later stages use larger image sizes with stronger regularization, as described in Algorithm 1. Table 6 shows the minimum (for the first stage) and maximum (for the last stage) values of image size and regularization. For simplicity, all models use the same minimum values of size and regularization, but they adopt different maximum values, as larger models generally require more regularization to combat overfitting. Following (Touvron et al., 2020), our maximum image size for training is about 20% smaller than inference, but we don't finetune any layers after training.

Results: As shown in Table 7, our EfficientNetV2 models are significantly faster and achieves better accuracy and parameter efficiency than previous ConvNets and Transformers on ImageNet. In particular, our EfficientNetV2-M achieves comparable accuracy to EfficientNet-B7 while training 11x faster using the same computing resources. Our EfficientNetV2 models also significantly outperform all recent RegNet and ResNeSt, in both accuracy and inference speed. Figure 1 further visualizes the comparison on training speed and parameter efficiency. Notably, this speedup is a combination of progressive training and better networks, and we will study the individual impact for each of them in our ablation studies.

Recently, Vision Transformers have demonstrated impressive results on ImageNet accuracy and training speed. However, here we show that properly designed ConvNets with improved training method can still largely outperform vision transformers in both accuracy and training efficiency. In particular, our EfficientNetV2-L achieves 85.7% top-1 accuracy, surpassing ViT-L/16(21k), a much larger transformer model pretrained on a larger ImageNet21k dataset. Here, ViTs are not well tuned on ImageNet ILSVRC2012; DeiTs use the same architectures as ViTs, but achieve better results by adding more regularization.

Although our EfficientNetV2 models are optimized for training, they also perform well for inference, because training speed often correlates with inference speed. Figure 5 visualizes the model size, FLOPs, and inference latency based on Table 7. Since latency often depends on hardware and software, here we use the same PyTorch Image Models codebase (Wightman, 2021) and run all models on the same machine using the batch size 16. In general, our models have slightly better parameters/FLOPs efficiency than EfficientNets, but our inference latency is up to 3x faster than EfficientNets. Compared to the recent ResNeSt that are specially optimized for GPUs, our EfficientNetV2-M achieves 0.6% better accuracy with 2.8x faster inference speed.

5.2. ImageNet21k

Setup: ImageNet21k (Russakovsky et al., 2015) contains about 13M training images with 21,841 classes. The original ImageNet21k doesn't have train/eval split, so we reserve randomly picked 100,000 images as validation set and use the remaining as training set. We largely reuse the same training settings as ImageNet ILSVRC2012 with a few changes: (1) we change the training epochs to 60 or 30 to reduce training time, and use cosine learning rate decay that can adapt to different steps without extra tuning; (2) since each image has multiple labels, we normalize the labels to have sum of 1 before computing softmax loss. After pretrained on ImageNet21k, each model is finetuned on ILSVRC2012 for 15 epochs using cosine learning rate decay.

Results: Table 7 shows the performance comparison, where models tagged with `21k` are pretrained on ImageNet21k and finetuned on ImageNet ILSVRC2012. Compared to the recent ViT-L/16(21k), our EfficientNetV2-L(21k) improves the top-1 accuracy by 1.5% (85.3% vs. 86.8%), using 2.5x fewer parameters and 3.6x fewer FLOPs, while running 6x - 7x faster in training and inference.

We would like to highlight a few interesting observations:

- *Scaling up data size is more effective than simply scaling up model size in high-accuracy regime*: when the top-1 accuracy is beyond 85%, it is very difficult to further improve it by simply increasing model size

5.1. ImageNet ILSVRC2012图像分类竞赛

配置说明：ImageNet ILSVRC2012数据集（Russakovsky等人，2015）包含约 1.28M 张训练图像和5万张验证图像，涵盖1000个类别。在架构搜索或超参数调优阶段，我们从训练集中保留25,000张图像（约 2% ）作为微型验证集用于精度评估，并据此实施早停策略。训练参数主要遵循EfficientNets（Tan & Le，2019a）：采用衰减系数0.9、动量0.9的RMSProp优化器；批归一化动量0.99；权重衰减1e-5。每个模型训练350个周期，总批次规模4096。学习率从0线性预热至0.256，之后每2.4个周期衰减0.97倍。同时应用了衰减率0.9999的指数移动平均、RandAugment（Cubuk等人，2020）、Mixup（Zhang等人，2018）、Dropout（Srivastava等人，2014）以及生存概率0.8的随机深度（Huang等人，2016）。

表6. EfficientNetV2的渐进式训练设置

	S		M		L	
	最小值	max	最小	max	分钟	最大值
图像尺寸	128	300	128	380	128	380
随机增强	5	15	5	20	5	25
混合系数	0	0	0	0.2	0	0.4
丢弃率	0.1	0.3	0.1	0.4	0.1	0.5

渐进式学习将训练过程分为四个阶段，每阶段约87个周期：早期阶段使用较小图像尺寸和较弱正则化，后期阶段则采用更大图像尺寸和更强正则化，如算法1所述。表6展示了图像尺寸和正则化的最小值（第一阶段）与最大值（最后阶段）。为简化流程，所有模型使用相同的最小尺寸和正则化值，但采用不同的最大值——因为较大模型通常需要更强正则化来防止过拟合。参照(Touvron等人, 2020)，我们训练时的最大图像尺寸比推理时小 20% ，但训练后不对任何层进行微调。

结果：如表7所示，我们的EfficientNetV2模型在ImageNet上相比传统卷积网络和Transformer具有显著的速度优势，同时实现了更高的准确率与参数效率。其中EfficientNetV2-M在保持与EfficientNet-B7相当精度的前提下，训练速度提升 $11\mathrm{x}$ 倍（使用相同计算资源）。我们的模型在精度和推理速度上也显著优于最新的RegNet与ResNeSt系列。图1进一步可视化展示了训练速度与参数效率的对比。值得注意的是，这种加速效果是渐进式训练与优化网络架构共同作用的结果，我们将在消融实验中分别研究二者的独立影响。

近期，视觉Transformer已展现出令人瞩目的

在ImageNet准确率和训练速度方面取得显著成果。然而本文证明，通过改进训练方法精心设计的卷积网络，仍能在精度和训练效率上大幅超越视觉Transformer模型。具体而言，我们的EfficientNetV2-L模型实现了85.7%的Top-1准确率，超越了在更大规模ImageNet21k数据集上预训练的ViT-L/16(21k) Transformer模型。需注意的是，ViT模型未在ImageNet ILSVRC2012数据集上充分调优；而DeiT虽然采用与ViT相同的架构，但通过增强正则化获得了更优性能。

尽管我们的EfficientNetV2模型针对训练进行了优化，但由于训练速度常与推理速度相关，它们在推理任务中同样表现优异。图5基于表7直观展示了模型参数量、浮点运算次数及推理延迟。由于延迟通常受硬件和软件环境影响，此处我们采用相同的PyTorch Image Models代码库(Wightman, 2021)，在批量大小为16的同一台机器上运行所有模型。总体而言，我们的模型在参数量/浮点运算效率上略优于EfficientNets，且推理延迟最高可比EfficientNets快 $3\mathrm{x}$ 。与专为GPU优化的最新ResNeSt相比，我们的EfficientNetV2-M在准确率上提升 0.6% 的同时，推理速度加快 $2.8\mathrm{x}$ 。

5.2. ImageNet21k数据集

实验设置：ImageNet21k数据集(Russakovsky等，2015)包含约 $13\mathrm{M}$ 张训练图像，涵盖21,841个类别。原始数据集未划分训练/验证集，因此我们随机保留100,000张图像作为验证集，其余用作训练集。基本沿用ImageNet ILSVRC2012的训练参数，主要调整包括：(1)将训练周期缩短至60或30轮以降低耗时，并采用余弦学习率衰减策略，无需额外调参即可适应不同训练步长；(2)由于图像存在多标签特性，在计算softmax损失前对标签进行归一化处理使其总和为1。在ImageNet21k预训练后，所有模型均在ILSVRC2012上采用余弦学习率衰减进行15轮微调。

结果：表7展示了性能对比，其中标注 $21\mathrm{k}$ 的模型均在ImageNet21k上预训练并在ImageNet ILSVRC2012上微调。相较于最新的ViT-L/16(21k)，我们的EfficientNetV2-L(21k)将top-1准确率从 86.8% 提升至 $1.5\%(85.3\%$ ，同时减少 $2.5\mathrm{x}$ 参数量与 $3.6\mathrm{x}$ 计算量，训练和推理速度加快 $6\mathrm{x}-7\mathrm{x}$ 。

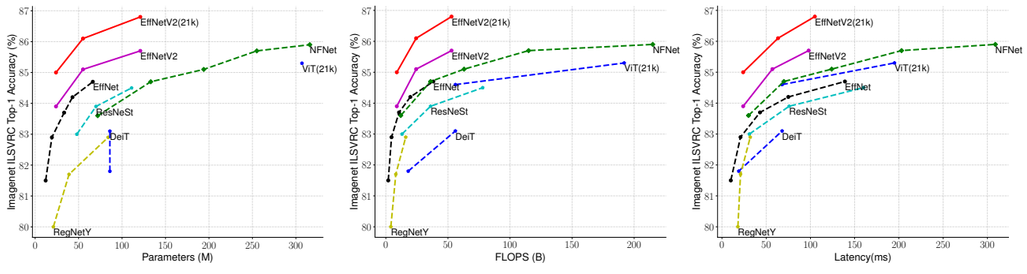
我们特别指出几点重要发现：

•在高精度领域，扩大数据规模比单纯增加模型规模更有效：当top-1准确率超过85%后，仅通过增大模型规模难以进一步提升精度

Table 7. **EfficientNetV2 Performance Results on ImageNet** (Russakovsky et al., 2015) – `Infer-time` is measured on V100 GPU FP16 with batch size 16 using the same codebase (Wightman, 2021); `Train-time` is the total training time normalized for 32 TPU cores. Models marked with 21k are pretrained on ImageNet21k with 13M images, and others are directly trained on ImageNet ILSVRC2012 with 1.28M images from scratch. All EfficientNetV2 models are trained with our improved method of progressive learning.

	Model	Top-1 Acc.	Params	FLOPs	Infer-time(ms)	Train-time (hours)
ConvNets & Hybrid	EfficientNet-B3 (Tan & Le, 2019a)	81.5%	12M	1.9B	19	10
	EfficientNet-B4 (Tan & Le, 2019a)	82.9%	19M	4.2B	30	21
	EfficientNet-B5 (Tan & Le, 2019a)	83.7%	30M	10B	60	43
	EfficientNet-B6 (Tan & Le, 2019a)	84.3%	43M	19B	97	75
	EfficientNet-B7 (Tan & Le, 2019a)	84.7%	66M	38B	170	139
	RegNetY-8GF (Radosavovic et al., 2020)	81.7%	39M	8B	21	-
	RegNetY-16GF (Radosavovic et al., 2020)	82.9%	84M	16B	32	-
	ResNeSt-101 (Zhang et al., 2020)	83.0%	48M	13B	31	-
	ResNeSt-200 (Zhang et al., 2020)	83.9%	70M	36B	76	-
	ResNeSt-269 (Zhang et al., 2020)	84.5%	111M	78B	160	-
	TResNet-L (Ridnik et al., 2020)	83.8%	56M	-	45	-
	TResNet-XL (Ridnik et al., 2020)	84.3%	78M	-	66	-
	EfficientNet-X (Li et al., 2021)	84.7%	73M	91B	-	-
	NFNet-F0 (Brock et al., 2021)	83.6%	72M	12B	30	8.9
	NFNet-F1 (Brock et al., 2021)	84.7%	133M	36B	70	20
	NFNet-F2 (Brock et al., 2021)	85.1%	194M	63B	124	36
	NFNet-F3 (Brock et al., 2021)	85.7%	255M	115B	203	65
	NFNet-F4 (Brock et al., 2021)	85.9%	316M	215B	309	126
	LambdaResNet-420-hybrid (Bello, 2021)	84.9%	125M	-	-	67
	BotNet-T7-hybrid (Srinivas et al., 2021)	84.7%	75M	46B	-	95
	BiT-M-R152x2 (21k) (Kolesnikov et al., 2020)	85.2%	236M	135B	500	-
Vision Transformers	ViT-B/32 (Dosovitskiy et al., 2021)	73.4%	88M	13B	13	-
	ViT-B/16 (Dosovitskiy et al., 2021)	74.9%	87M	56B	68	-
	DeiT-B (ViT+reg) (Touvron et al., 2021)	81.8%	86M	18B	19	-
	DeiT-B-384 (ViT+reg) (Touvron et al., 2021)	83.1%	86M	56B	68	-
	T2T-ViT-19 (Yuan et al., 2021)	81.4%	39M	8.4B	-	-
	T2T-ViT-24 (Yuan et al., 2021)	82.2%	64M	13B	-	-
	ViT-B/16 (21k) (Dosovitskiy et al., 2021)	84.6%	87M	56B	68	-
	ViT-L/16 (21k) (Dosovitskiy et al., 2021)	85.3%	304M	192B	195	172
ConvNets (ours)	**EfficientNetV2-S**	83.9%	22M	8.8B	24	7.1
	EfficientNetV2-M	85.1%	54M	24B	57	13
	EfficientNetV2-L	85.7%	120M	53B	98	24
	EfficientNetV2-S (21k)	84.9%	22M	8.8B	24	9.0
	EfficientNetV2-M (21k)	86.2%	54M	24B	57	15
	EfficientNetV2-L (21k)	86.8%	120M	53B	98	26
	EfficientNetV2-XL (21k)	87.3%	208M	94B	-	45

We do not include models pretrained on non-public Instagram/JFT images, or models with extra distillation or ensemble.



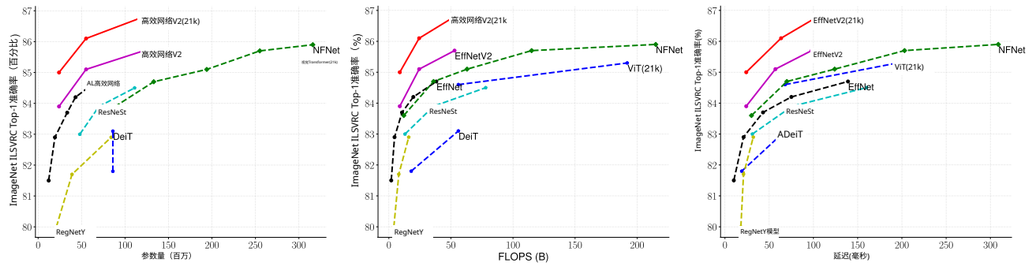
(a) Parameters (b) FLOPs (c) GPU V100 Latency (batch 16)

Figure 5. **Model Size, FLOPs, and Inference Latency** – Latency is measured with batch size 16 on V100 GPU. `21k` denotes pretrained on ImageNet21k images, others are just trained on ImageNet ILSVRC2012. Our EfficientNetV2 has slightly better parameter efficiency with EfficientNet, but runs 3x faster for inference.

表7. EfficientNetV2在ImageNet上的性能结果（Russakovsky等人，2015）——推理时间在V100 GPUFP16上测量，批大小为16，使用相同代码库（Wightman，2021）；训练时间是针对32个TPU核心归一化的总训练时间。标有 21k 的模型是在ImageNet 21k 上使用 13M 张图像预训练的，其他模型则直接在ImageNet ILSVRC2012上从头开始训练，使用128万张图像。所有EfficientNetV2模型均采用我们改进的渐进式学习方法进行训练。

	模型	Top-1准确率	参数量	浮点运算量	推理时间(毫秒)	训练时长（小时）
卷积网络与混合架构	高效网络-B3（Tan & Le，2019a）	81.5%	12M	1.9B	19	10
	高效网络-B4（Tan & Le，2019a）	82.9%	19M	4.2B	30	21
	高效网络-B5（Tan & Le，2019a）	83.7%	30M	10B	60	43
	高效网络-B6（Tan & Le，2019a）	84.3%	43M	19B	97	75
	高效网络B7（Tan & Le，2019a）	84.7%	66M	38B	170	139
	规则网络Y-8GF（Radosavovic等，2020）	81.7%	39M	8B	21	-
	规则网络Y-16GF（Radosavovic等，2020）	82.9%	84M	16B	32	-
	残差嵌套网络101（Zhang等，2020）	83.0%	48M	13B	31	-
	残差嵌套网络200（Zhang等，2020）	83.9%	70M	36B	76	-
	残差嵌套网络269（Zhang等，2020）	84.5%	111M	78B	160	-
	TResNet-L模型（Ridnik等，2020）	83.8%	56M	-	45	-
	TResNet-XL模型（Ridnik等，2020）	84.3%	78M	-	66	-
	高效网络X版（Li等，2021）	84.7%	73M	91B	-	-
	NFNet-F0架构（Brock等，2021）	83.6%	72M	12B	30	8.9
	NFNet-F1架构（Brock等，2021）	84.7%	133M	36B	70	20
	NFNet-F2架构（Brock等，2021）	85.1%	194M	63B	124	36
	NFNet-F3模型（Brock等，2021）	85.7%	255M	115B	203	65
	NFNet-F4模型（Brock等，2021）	85.9%	316M	215B	309	126
	LambdaResNet-420混合架构（Bello，2021）	84.9%	125M	-	-	67
	BotNet-T7混合架构（Srinivas等，2021）	84.7%	75M	46B	-	95
	BiT-M-R152x2模型（21k数据集）（Kolesnikov等，2020）	85.2%	236M	135B	500	-
视觉Transformer模型	ViT-B/32架构（Dosovitskiy等人，2021）	73.4%	88M	13B	13	-
	ViT-B/16架构（Dosovitskiy等人，2021）	74.9%	87M	56B	68	-
	DeiT-B模型（ViT+正则化）（Touvron等人，2021）	81.8%	86M	18B	19	-
	DeiT-B-384模型（ViT+正则化）（Touvron等人，2021）	83.1%	86M	56B	68	-
	T2T-ViT-19架构（Yuan等人，2021）	81.4%	39M	8.4B	-	-
	T2T-ViT-24（袁等人，2021）	82.2%	64M	13B	-	-
	ViT-B/16（21k数据集）（Dosovitskiy等人，2021）	84.6%	87M	56B	68	-
	ViT-L/16（21k数据集）（Dosovitskiy等人，2021）	85.3%	304M	192B	195	172
卷积网络（本研究）	高效网络V2-S型	83.9%	22M	8.8B	24	7.1
	高效网络V2-M型	85.1%	54M	24B	57	13
	高效网络V2-L型	85.7%	120M	53B	98	24
	高效网络V2-S型（21k）	84.9%	22M	8.8B	24	9.0
	高效网络V2-M型（21k）	86.2%	54M	24B	57	15
	高效网络V2-L型（21k）	86.8%	120M	53B	98	26
	高效网络V2-XL型（21k）	87.3%	208M	94B	-	45

本列表不包含基于非公开Instagram/JFT图像预训练的模型，也不包含采用额外蒸馏或集成技术的模型。



(a) 参数量 (b) 浮点运算量 (c) V100显卡延迟(批次16)

图5. 模型尺寸、浮点运算量与推理延迟 - 延迟数据基于V100 GPU批量16测得。21k表示使用ImageNet21k图像预训练，其余模型仅使用ImageNet ILSVRC2012训练。我们的EfficientNetV2在参数效率上略优于EfficientNet，但推理速度 3x 更快。

Table 8. **Transfer Learning Performance Comparison** – All models are pretrained on ImageNet ILSVRC2012 and finetuned on downstream datasets. Transfer learning accuracy is averaged over five runs.

	Model	Params	ImageNet Acc.	CIFAR-10	CIFAR-100	Flowers	Cars
ConvNets	GPipe (Huang et al., 2019)	556M	84.4	99.0	91.3	98.8	94.7
	EfficientNet-B7 (Tan & Le, 2019a)	66M	84.7	98.9	91.7	98.8	94.7
Vision Transformers	ViT-B/32 (Dosovitskiy et al., 2021)	88M	73.4	97.8	86.3	85.4	-
	ViT-B/16 (Dosovitskiy et al., 2021)	87M	74.9	98.1	87.1	89.5	-
	ViT-L/32 (Dosovitskiy et al., 2021)	306M	71.2	97.9	87.1	86.4	-
	ViT-L/16 (Dosovitskiy et al., 2021)	306M	76.5	97.9	86.4	89.7	-
	DeiT-B (ViT+regularization) (Touvron et al., 2021)	86M	81.8	99.1	90.8	98.4	92.1
	DeiT-B-384 (ViT+regularization) (Touvron et al., 2021)	86M	83.1	99.1	90.8	98.5	93.3
ConvNets (ours)	EfficientNetV2-S	24M	83.2	98.7±0.04	91.5±0.11	97.9±0.13	93.8±0.11
	EfficientNetV2-M	55M	85.1	99.0±0.08	92.2±0.08	98.5±0.08	94.6±0.10
	EfficientNetV2-L	121M	85.7	**99.1**±0.03	**92.3**±0.13	**98.8**±0.05	**95.1**±0.10

due to the severe overfitting. However, the extra ImageNet21K pretraining can significantly improve accuracy. The effectiveness of large datasets is also observed in previous works (Mahajan et al., 2018; Xie et al., 2020; Dosovitskiy et al., 2021).

- *Pretraining on ImageNet21k could be quite efficient.* Although ImageNet21k has 10x more data, our training approach enables us to finish the pretraining of EfficientNetV2 within two days using 32 TPU cores (instead of weeks for ViT (Dosovitskiy et al., 2021)). This is more effective than training larger models on ImageNet. We suggest future research on large-scale models use the public ImageNet21k as a default dataset.

5.3. Transfer Learning Datasets

Setup: We evaluate our models on four transfer learning datasets: CIFAR-10, CIFAR-100, Flowers and Cars. Table 9 includes the statistics of these datasets.

Table 9. Transfer learning datasets.

	Train images	Eval images	Classes
CIFAR-10 (Krizhevsky & Hinton, 2009)	50,000	10,000	10
CIFAR-100 (Krizhevsky & Hinton, 2009)	50,000	10,000	100
Flowers (Nilsback & Zisserman, 2008)	2,040	6,149	102
Cars (Krause et al., 2013)	8,144	8,041	196

For this experiment, we use the checkpoints trained on ImageNet ILSVRC2012. For fair comparison, no ImageNet21k images are used here. Our finetuning settings are mostly the same as ImageNet training with a few modifications similar to (Dosovitskiy et al., 2021; Touvron et al., 2021): We use smaller batch size 512, smaller initial learning rate 0.001 with cosine decay. For all datasets, we train each model for fixed 10,000 steps. Since each model is finetuned with very few steps, we disable weight decay and use a simple cutout data augmentation.

Results: Table 8 compares the transfer learning performance. In general, our models outperform previous ConvNets and Vision Transformers for all these datasets, sometimes by a non-trivial margin: for example, on CIFAR-100, EfficientNetV2-L achieves 0.6% better accuracy than prior GPipe/EfficientNets and 1.5% better accuracy than prior ViT/DeiT models. These results suggest that our models also generalize well beyond ImageNet.

6. Ablation Studies

6.1. Comparison to EfficientNet

In this section, we will compare our EfficientNetV2 (V2 for short) with EfficientNets (Tan & Le, 2019a) (V1 for short) under the same training and inference settings.

Performance with the same training: Table 10 shows the performance comparison using the same progressive learning settings. As we apply the same progressive learning to EfficientNet, its training speed (reduced from 139h to 54h) and accuracy (improved from 84.7% to 85.0%) are better than the original paper (Tan & Le, 2019a). However, as shown in Table 10, our EfficientNetV2 models still outperform EfficientNets by a large margin: EfficientNetV2-M reduces parameters by 17% and FLOPs by 37%, while running 4.1x faster in training and 3.1x faster in inference than EfficientNet-B7. Since we are using the same training settings here, we attribute the gains to the EfficientNetV2 architecture.

Table 10. Comparison with the same training settings – Our new EfficientNetV2-M runs faster with less parameters.

	Acc. (%)	Params (M)	FLOPs (B)	TrainTime (h)	InferTime (ms)
V1-B7	85.0	66	38	54	170
V2-M (ours)	85.1	55 (-17%)	24 (-37%)	13 (-76%)	57 (-66%)

Scaling Down: Previous sections mostly focus on large-scale models. Here we compare smaller models by scaling down our EfficientNetV2-S using EfficientNet compound scaling. For easy comparison, all models are trained without progressive learning. Compared to small-size EfficientNets (V1), our new EfficientNetV2 (V2) models are generally faster while maintaining comparable parameter efficiency.

表8. 迁移学习性能对比 - 所有模型均在ImageNet ILSVRC2012上预训练，并在下游数据集微调。迁移学习准确率为五次运行的平均值。

	模型	参数量	ImageNet准确率	CIFAR-10	CIFAR-100	花卉数据集	汽车数据集
卷积神经网络	GPipe（黄等，2019）	556M	84.4	99.0	91.3	98.8	94.7
	高效网络-B7（谭与乐，2019a）	66M	84.7	98.9	91.7	98.8	94.7
视觉变换器	视觉变换器-B/32（多索维茨基等，2021）	88M	73.4	97.8	86.3	85.4	-
	ViT-B/16架构（Dosovitskiy等人，2021年）	87M	74.9	98.1	87.1	89.5	-
	ViT-L/32架构（Dosovitskiy等人，2021年）	306M	71.2	97.9	87.1	86.4	-
	ViT-L/16架构（Dosovitskiy等人，2021年）	306M	76.5	97.9	86.4	89.7	-
	DeiT-B模型（ViT+正则化技术）（Touvron等人，2021年）	86M	81.8	99.1	90.8	98.4	92.1
	DeiT-B-384模型（ViT+正则化技术）（Touvron等人，2021年）	86M	83.1	99.1	90.8	98.5	93.3
卷积神经网络（本研究）	高效网络V2-S型	24M	83.2	98.7±0.04	91.5±0.11	97.9±0.13	93.8±0.11
	高效网络V2-M型	55M	85.1	99.0±0.08	92.2±0.08	98.5±0.08	94.6±0.10
	高效网络V2-L型	121M	85.7	**99.1**±0.03	**92.3**±0.13	**98.8**±0.05	**95.1**±0.10

由于严重的过拟合问题。但额外的ImageNet 21 K预训练能显著提升准确率。大数据集的有效性在先前研究中也有体现（Mahajan等人，2018；Xie等人，2020；Dosovitskiy等人，2021）。

•ImageNet21k预训练效率极高。尽管ImageNet 21k包含 10x 更多数据，我们的训练方法仅需32个TPU核心两天即可完成高效网络V2预训练（相比ViT需数周（Dosovitskiy等人，2021））。这比在ImageNet上训练更大模型更高效。建议未来大规模模型研究将公开的ImageNet 21k 作为默认数据集。

5.3 迁移学习数据集

实验设置：我们在四个迁移学习数据集（CIFAR-10、CIFAR-100、Flowers和Cars）上评估模型性能。表9列出了这些数据集的统计信息。

表9. 迁移学习数据集

	训练图像	验证图像	类别数
CIFAR-10数据集（Krizhevsky & Hinton，2009）	50,000	10,000	10
CIFAR-100数据集（Krizhevsky & Hinton，2009）	50,000	10,000	100
花卉数据集（Nilsback & Zisserman，2008）	2,040	6,149	102
汽车数据集（Krause等，2013）	8,144	8,041	196

本实验采用在ImageNet ILSVRC2012上训练的检查点。为公平对比，未使用ImageNet21k图像。微调设置基本与ImageNet训练一致，仅作少量调整（参考Dosovitskiy等，2021；Touvron等，2021）：采用较小批量512、初始学习率0.001配合余弦衰减。所有数据集均固定训练10,000步。由于微调步数极少，我们禁用权重衰减并采用简单裁剪数据增强。

结果：表8对比了迁移学习性能。总体而言，我们的模型在所有数据集上均优于传统卷积网络和视觉Transformer，部分场景优势显著：例如在CIFAR-100上，

EfficientNetV2-L模型比先前的GPipe/EfficientNets精度提升0.6%，较之前的ViT/DeiT模型提升1.5%。这些结果表明我们的模型在ImageNet之外也具备良好的泛化能力。

6. 消融实验

6.1. 与EfficientNet的对比

本节将在相同的训练和推理设置下，对比我们的EfficientNetV2（简称V2）与EfficientNets（Tan & Le, 2019a）（简称V1）。

相同训练条件下的性能表现：表10展示了采用相同渐进式学习设置的性能对比。当我们将相同的渐进式学习应用于EfficientNet时，其训练速度（从 139 h 缩短至 54 h ）和准确率（从84.7%提升至85.0%）均优于原论文（Tan & Le, 2019a）。但如表10所示，我们的EfficientNetV2模型仍大幅领先EfficientNet系列：EfficientNetV2-M参数量减少17%，计算量降低37%，训练速度比EfficientNet-B7快 4.1x 倍，推理速度快 3.1x 倍。由于采用完全相同的训练配置，我们认为这些提升应归功于EfficientNetV2的架构设计。

表10. 相同训练配置下的对比——我们的新型EfficientNetV2-M以更少参数实现更快运行

	准确率 (%)	参数量 (M)	计算量 (B)	训练时长 (h)	推理时间 (ms)
V1-B7	85.0	66	38	54	170
V2-M（我们的模型）	85.1	55 (-17%)	24 (-37%)	13 (-76%)	57 (-66%)

模型缩小：前文主要关注大规模模型。本节我们通过复合缩放方法将EfficientNetV2-S缩小，与其他小型模型对比。为便于比较，所有模型均未采用渐进式学习。与小型EfficientNet（V1）相比，我们的新版EfficientNetV2（V2）模型在保持参数量效率相当的同时，通常具有更快的速度。

Table 11. Scaling down model size – We measure the inference throughput (images/sec) on V100 FP16 GPU with batch size 128.

	Top-1 Acc.	Parameters	FLOPs	Throughput
V1-B1	79.0%	7.8M	0.7B	2675
V2-B0	78.7%	7.4M	0.7B	(2.1x) 5739
V1-B2	79.8%	9.1M	1.0B	2003
V2-B1	79.8%	8.1M	1.2B	(2.0x) 3983
V1-B4	82.9%	19M	4.2B	628
V2-B3	82.1%	14M	3.0B	(2.7x) 1693
V1-B5	83.7%	30M	9.9B	291
V2-S	83.6%	24M	8.8B	(3.1x) 901

6.2. Progressive Learning for Different Networks

We ablate the performance of our progressive learning for different networks. Table 12 shows the performance comparison between our progressive training and the baseline training, using the same ResNet and EfficientNet models. Here, the baseline ResNets have higher accuracy than the original paper (He et al., 2016) because they are trained with our improved training settings (see Section 5) using more epochs and better optimizers. We also increase the image size from 224 to 380 for ResNets to further increase the network capacity and accuracy.

Table 12. Progressive learning for ResNets and EfficientNets – (224) and (380) denote inference image size. Our progressive training improves both accuracy and training time for all networks.

	Baseline		Progressive	
	Acc.(%)	TrainTime	Acc.(%)	TrainTime
ResNet50 (224)	78.1	4.9h	78.4	3.5h (-29%)
ResNet50 (380)	80.0	14.3h	80.3	5.8h (-59%)
ResNet152 (380)	82.4	15.5h	82.9	7.2h (-54%)
EfficientNet-B4	82.9	20.8h	83.1	9.4h (-55%)
EfficientNet-B5	83.7	42.9h	84.0	15.2h (-65%)

As shown in Table 12, our progressive learning generally reduces the training time and meanwhile improves the accuracy for all different networks. Not surprisingly, when the default image size is very small, such as ResNet50(224) with 224x224 size, the training speedup is limited (1.4x speedup); however, when the default image size is larger and the model is more complex, our approach achieves larger gains on accuracy and training efficiency: for ResNet152(380), our approach improves speed up the training by 2.1x with slightly better accuracy; for EfficientNet-B4, our approach improves speed up the training by 2.2x.

6.3. Importance of Adaptive Regularization

A key insight from our training approach is the adaptive regularization, which dynamically adjusts regularization according to image size. This paper chooses a simple progressive approach for its simplicity, but it is also a general method that can be combined with other approaches.

Table 13 studies our adaptive regularization on two training settings: one is to progressively increase image size from small to large (Howard, 2018), and the other is to randomly sample a different image size for each batch (Hoffer et al., 2019). Because TPU needs to recompile the graph for each new size, here we randomly sample a image size every eight epochs instead of every batch. Compared to the vanilla approaches of progressive or random resizing that use the same regularization for all image sizes, our adaptive regularization improves the accuracy by 0.7%. Figure 6 further compares the training curve for the progressive approach. Our adaptive regularization uses much smaller regularization for small images at the early training epochs, allowing models to converge faster and achieve better final accuracy.

Table 13. Adaptive regularization – We compare ImageNet top-1 accuracy based on the average of three runs.

	Vanilla	+our adaptive reg
Progressive resize (Howard, 2018)	84.3±0.14	85.1±0.07 (+0.8)
Random resize (Hoffer et al., 2019)	83.5±0.11	84.2±0.10 (+0.7)

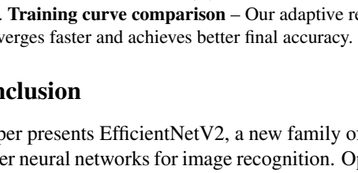


Figure 6. **Training curve comparison** – Our adaptive regularization converges faster and achieves better final accuracy.

7. Conclusion

This paper presents EfficientNetV2, a new family of smaller and faster neural networks for image recognition. Optimized with training-aware NAS and model scaling, our EfficientNetV2 significantly outperforms previous models, while being much faster and more efficient in parameters. To further speed up the training, we propose an improved method of progressive learning, that jointly increases image size and regularization during training. Extensive experiments show our EfficientNetV2 achieves strong results on ImageNet, and CIFAR/Flowers/Cars. Compared to EfficientNet and more recent works, our EfficientNetV2 trains up to 11x faster while being up to 6.8x smaller.

Acknowledgements

Special thanks to Lucas Sloan for helping open sourcing. We thank Ruoming Pang, Sheng Li, Andrew Li, Hanxiao Liu, Zihang Dai, Neil Houlsby, Ross Wightman, Jeremy Howard, Thang Luong, Daiyi Peng, Yifeng Lu, Da Huang, Chen Liang, Aravind Srinivas, Irwan Bello, Max Moroz, Futang Peng for their feedback.

表11. 模型尺寸缩减实验 - 我们在V100 FP16 GPU上以128的批量大小测量推理吞吐量（图像/秒）。

	Top-1准确率	参数量	浮点运算量	吞吐量
V1-B1	79.0%	7.8M	0.7B	2675
V2-B0	78.7%	7.4M	0.7B	(2.1x) 5739
V1-B2	79.8%	9.1M	1.0B	2003
V2-B1	79.8%	8.1M	1.2B	(2.0x) 3983
V1-B4	82.9%	19M	4.2B	628
V2-B3	82.1%	14M	3.0B	(2.7x) 1693
V1-B5	83.7%	30M	9.9B	291
V2-S	83.6%	24M	8.8B	(3.1x) 901

6.2. 不同网络的渐进式学习

我们针对不同网络进行了渐进式学习的性能消融实验。表12展示了在相同ResNet和EfficientNet模型下，渐进式训练与基线训练的性能对比。需要注意的是，基线ResNet的准确率高于原论文(He et al., 2016)报告值，这是因为采用了改进的训练设置（参见第5节），包括更多训练轮次和更优优化器。此外，我们将ResNet的输入图像尺寸从224提升至380，以进一步增强网络容量和准确率。

表12. ResNet与EfficientNet的渐进式学习效果 -(224)和(380)表示推理图像尺寸。我们的渐进式训练在所有网络上都实现了准确率和训练时间的双重提升。

	基线		渐进式	
	准确率(%)	训练耗时	准确率(%)	训练时间
ResNet50 (224)	78.1	4.9h	78.4	3.5h (-29%)
ResNet50 (380)	80.0	14.3h	80.3	5.8h (-59%)
ResNet152 (380)	82.4	15.5h	82.9	7.2h (-54%)
EfficientNet-B4	82.9	20.8h	83.1	9.4h (-55%)
高效网络B5	83.7	42.9h	84.0	15.2h (-65%)

如表12所示，我们的渐进式学习普遍缩短了训练时间，同时提升了所有不同网络的准确率。当默认图像尺寸较小时（如ResNet50(224)采用 224×224 尺寸），训练加速效果有限（1.4倍加速）；但当默认图像尺寸较大且模型更复杂时，我们的方法在精度和训练效率上获得更大提升：对于ResNet152(380)，训练速度提升 2.1x 倍且精度略有提高；对于高效网络B4，训练速度提升 2.2x 倍。

6.3. 自适应正则化的重要性

本训练方法的核心在于自适应正则化机制，它能根据图像尺寸动态调整正则化强度。本文选择简单的渐进式方法以实现简洁性，但这也是一种通用方案，可与其他方法结合使用。

表13研究了两种训练中的自适应正则化

设置项：一种是从小到大逐步增加图像尺寸（Howard，2018），另一种是为每批次随机采样不同尺寸（Hoffer等，2019）。由于TPU需为每个新尺寸重新编译计算图，此处改为每八个周期而非每批次随机采样尺寸。相比传统渐进式或随机尺寸调整对所有尺寸采用相同正则化，我们的自适应正则化将准确率提升了 0.7% 。图6进一步对比了渐进式方法的训练曲线。自适应正则化在训练早期对小尺寸图像采用更弱正则化，使模型收敛更快且最终精度更高。

表13. 自适应正则化 - 基于三次运行平均值对比ImageNet top-1准确率

	传统方法	+我们的自适应正则
渐进式尺寸调整（Howard，2018）	84.3±0.14	85.1±0.07 (+0.8)
随机尺寸调整（Hoffer等，2019）	83.5±0.11	84.2±0.10 (+0.7)

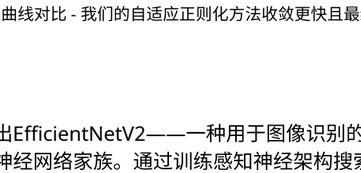


图6. 训练曲线对比 - 我们的自适应正则化方法收敛更快且最终精度更高。

7. 结论

本文提出EfficientNetV2——一种用于图像识别的新型轻量高效神经网络家族。通过训练感知神经架构搜索和模型缩放优化，我们的EfficientNetV2在显著超越先前模型的同时，参数量更少、训练速度更快。为加速训练过程，我们提出改进的渐进式学习方法，在训练过程中同步调整图像尺寸与正则化强度。大量实验表明，EfficientNetV2在ImageNet及CIFAR/Flowers/Cars数据集上表现优异。相较于EfficientNet及最新研究成果，EfficientNetV2训练速度提升最高达11倍，模型体积缩小 6.8x 。

致谢

特别感谢Lucas Sloan协助开源工作。感谢Ruoming Pang、Sheng Li、Andrew Li、Hanxiao Liu、Zihang Dai、Neil Houlsby、Ross Wightman、Jeremy Howard、Thang Luong、Daiyi Peng、Yifeng Lu、Da Huang、Chen Liang、Aravind Srinivas、Irwan Bello、Max Moroz、Futang Peng提供的宝贵建议。

References

Bello, I. Lambdanetworks: Modeling long-range interactions without attention. *ICLR*, 2021.

Bello, I., Fedus, W., Du, X., Cubuk, E. D., Srinivas, A., Lin, T.-Y., Shlens, J., and Zoph, B. Revisiting resnets: Improved training and scaling strategies. *arXiv preprint arXiv:2103.07579*, 2021.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. *ICML*, 2009.

Brock, A., De, S., Smith, S. L., and Simonyan, K. High-performance large-scale image recognition without normalization. *arXiv preprint arXiv:2102.06171*, 2021.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. *NeurIPS*, 2020.

Cai, H., Zhu, L., and Han, S. Proxylessnas: Direct neural architecture search on target task and hardware. *ICLR*, 2019.

Chen, Y., Yang, T., Zhang, X., Meng, G., Pan, C., and Sun, J. Detnas: Neural architecture search on object detection. *NeurIPS*, 2019.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. Randaugment: Practical automated data augmentation with a reduced search space. *ECCV*, 2020.

Dong, X., Tan, M., Yu, A. W., Peng, D., Gabrys, B., and Le, Q. V. Autohas: Efficient hyperparameter and architecture search. *arXiv preprint arXiv:2006.03656*, 2020.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

Elsken, T., Metzen, J. H., and Hutter, F. Neural architecture search: A survey. *Journal of Machine Learning Research*, 2019.

Gupta, S. and Akin, B. Accelerator-aware neural network design using automl. *On-device Intelligence Workshop in SysML*, 2020.

Gupta, S. and Tan, M. Efficientnet-edgetpu: Creating accelerator-optimized neural networks with automl. *https://ai.googleblog.com/2019/08/efficientnet-edgetpu-creating.html*, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CVPR*, pp. 770–778, 2016.

Hoffer, E., Weinstein, B., Hubara, I., Ben-Nun, T., Hoefler, T., and Soudry, D. Mix & match: training convnets with mixed image sizes for improved accuracy, speed and scale resiliency. *arXiv preprint arXiv:1908.08986*, 2019.

Howard, J. Training imagenet in 3 hours for 25 minutes. *https://www.fast.ai/2018/04/30/dawnbench-fastai/*, 2018.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. Deep networks with stochastic depth. *ECCV*, pp. 646–661, 2016.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. *CVPR*, 2017.

Huang, Y., Cheng, Y., Chen, D., Lee, H., Ngiam, J., Le, Q. V., and Chen, Z. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *NeurIPS*, 2019.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. *ICLR*, 2018.

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. Big transfer (bit): General visual representation learning. *ECCV*, 2020.

Krause, J., Deng, J., Stark, M., and Fei-Fei, L. Collecting a large-scale dataset of fine-grained cars. *Second Workshop on Fine-Grained Visual Categorizatio*, 2013.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

Li, S., Tan, M., Pang, R., Li, A., Cheng, L., Le, Q., and Jouppi, N. Searching for fast model families on datacenter accelerators. *CVPR*, 2021.

Liu, C., Chen, L.-C., Schroff, F., Adam, H., Hua, W., Yuille, A., and Fei-Fei, L. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. *CVPR*, 2019.

Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., and van der Maaten, L. Exploring the limits of weakly supervised pretraining. *arXiv preprint arXiv:1805.00932*, 2018.

Nilsback, M.-E. and Zisserman, A. Automated flower classification over a large number of classes. *ICVGIP*, pp. 722–729, 2008.

EfficientNetV2: Smaller Models and Faster Training

参考文献

Bello, I. Lambdanetworks: Modeling long-range interactions without attention. *ICLR*, 2021.

Bello, I., Fedus, W., Du, X., Cubuk, E. D., Srinivas, A., Lin, T.-Y., Shlens, J., and Zoph, B. Revisiting resnets: Improved training and scaling strategies. *arXiv preprint arXiv:2103.07579*, 2021.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. *ICML*, 2009.

Brock, A., De, S., Smith, S. L., and Simonyan, K. High-performance large-scale image recognition without normalization. *arXiv preprint arXiv:2102.06171*, 2021.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. *NeurIPS*, 2020.

Cai, H., Zhu, L., and Han, S. Proxylessnas: Direct neural architecture search on target task and hardware. *ICLR*, 2019.

Chen, Y., Yang, T., Zhang, X., Meng, G., Pan, C., and Sun, J. Detnas: Neural architecture search on object detection. *NeurIPS*, 2019.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. Randaugment: Practical automated data augmentation with a reduced search space. *ECCV*, 2020.

Dong, X., Tan, M., Yu, A. W., Peng, D., Gabrys, B., and Le, Q. V. Autohas: Efficient hyperparameter and architecture search. *arXiv preprint arXiv:2006.03656*, 2020.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

Elsken, T., Metzen, J. H., and Hutter, F. Neural architecture search: A survey. *Journal of Machine Learning Research*, 2019.

Gupta, S. and Akin, B. Accelerator-aware neural network design using automl. *On-device Intelligence Workshop in SysML*, 2020.

Gupta, S. and Tan, M. Efficientnet-edgetpu: Creating accelerator-optimized neural networks with automl. *https://ai.googleblog.com/2019/08/efficientnet-edgetpu-creating.html*, 2019.

何凯明，张祥雨，任少卿，孙剑。深度残差 learning for image recognition. *CVPR*, pp. 770–778, 2016.

Hoffer, E., Weinstein, B., Hubara, I., Ben-Nun, T., Hoefler, T., and Soudry, D. Mix & match: training convnets with mixed image sizes for improved accuracy, speed and scale resiliency. *arXiv preprint arXiv:1908.08986*, 2019.

Howard, J. Training imagenet in 3 hours for 25 minutes. *https://www.fast.ai/2018/04/30/dawnbench-fastai/*, 2018.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. Deep networks with stochastic depth. *ECCV*, pp. 646–661, 2016.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. *CVPR*, 2017.

Huang, Y., Cheng, Y., Chen, D., Lee, H., Ngiam, J., Le, Q. V., and Chen, Z. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *NeurIPS*, 2019.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. *ICLR*, 2018.

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. Big transfer (bit): General visual representation learning. *ECCV*, 2020.

Krause, J., Deng, J., Stark, M., and Fei-Fei, L. Collecting a large-scale dataset of fine-grained cars. *Second Workshop on Fine-Grained Visual Categorizatio*, 2013.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

Li, S., Tan, M., Pang, R., Li, A., Cheng, L., Le, Q., and Jouppi, N. Searching for fast model families on datacenter accelerators. *CVPR*, 2021.

Liu, C., Chen, L.-C., Schroff, F., Adam, H., Hua, W., Yuille, A., and Fei-Fei, L. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. *CVPR*, 2019.

Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., and van der Maaten, L. Exploring the limits of weakly supervised pretraining. *arXiv preprint arXiv:1805.00932*, 2018.

Nilsback, M.-E. and Zisserman, A. Automated flower classification over a large number of classes. *ICVGIP*, pp. 722–729, 2008.

Press, O., Smith, N. A., and Lewis, M. Shortformer: Better language modeling using shorter inputs. *arXiv preprint arXiv:2012.15832*, 2021.

Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., and Dollár, P. Designing network design spaces. *CVPR*, 2020.

Ridnik, T., Lawen, H., Noy, A., Baruch, E. B., Sharir, G., and Friedman, I. Tresnet: High performance gpu-dedicated architecture. *arXiv preprint arXiv:2003.13630*, 2020.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3): 211–252, 2015.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. *CVPR*, 2018.

Sifre, L. Rigid-motion scattering for image classification. *Ph.D. thesis section 6.2*, 2014.

Srinivas, A., Lin, T.-Y., Parmar, N., Shlens, J., Abbeel, P., and Vaswani, A. Bottleneck transformers for visual recognition. *arXiv preprint arXiv:2101.11605*, 2021.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. *ICML*, 2019a.

Tan, M. and Le, Q. V. Mixconv: Mixed depthwise convolutional kernels. *BMVC*, 2019b.

Tan, M., Chen, B., Pang, R., Vasudevan, V., and Le, Q. V. Mnasnet: Platform-aware neural architecture search for mobile. *CVPR*, 2019.

Tan, M., Pang, R., and Le, Q. V. Efficientdet: Scalable and efficient object detection. *CVPR*, 2020.

Touvron, H., Vedaldi, A., Douze, M., and Jégou, H. Fixing the train-test resolution discrepancy. *arXiv preprint arXiv:1906.06423*, 2019.

Touvron, H., Vedaldi, A., Douze, M., and Jégou, H. Fixing the train-test resolution discrepancy: Fixefficientnet. *arXiv preprint arXiv:2003.08237*, 2020.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2021.

Wightman, R. Pytorch image model. *https://github.com/rwightman/pytorch-image-models, Accessed on Feb.18, 2021*, 2021.

Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., and Keutzer, K. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. *CVPR*, 2019.

Xie, Q., Luong, M.-T., Hovy, E., and Le, Q. V. Self-training with noisy student improves imagenet classification. *CVPR*, 2020.

Xiong, Y., Liu, H., Gupta, S., Akin, B., Bender, G., Kindermans, P.-J., Tan, M., Singh, V., and Chen, B. Mobiledets: Searching for object detection architectures for mobile accelerators. *arXiv preprint arXiv:2004.14525*, 2020.

Yu, H., Liu, A., Liu, X., Li, G., Luo, P., Cheng, R., Yang, J., and Zhang, C. Pda: Progressive data augmentation for general robustness of deep neural networks. *arXiv preprint arXiv:1909.04839*, 2019.

Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Tay, F. E., Feng, J., and Yan, S. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. Mixup: Beyond empirical risk minimization. *ICLR*, 2018.

Zhang, H., Wu, C., Zhang, Z., Zhu, Y., Lin, H., Zhang, Z., Sun, Y., He, T., Mueller, J., Manmatha, R., Li, M., and Smola, A. Resnest: Split-attention networks. *arXiv preprint arXiv:2012.12877*, 2020.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. *CVPR*, 2018.

EfficientNetV2: Smaller Models and Faster Training

普雷斯，O.，史密斯，N. A.，与刘易斯，M. 短文本模型：更优 language modeling using shorter inputs. *arXiv preprint arXiv:2012.15832*, 2021.

Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., and Dollár, P. Designing network design spaces. *CVPR*, 2020.

Ridnik, T., Lawen, H., Noy, A., Baruch, E. B., Sharir, G., and Friedman, I. Tresnet: High performance gpu-dedicated architecture. *arXiv preprint arXiv:2003.13630*, 2020.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3): 211–252, 2015.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. *CVPR*, 2018.

Sifre, L. Rigid-motion scattering for image classification. *Ph.D. thesis section 6.2*, 2014.

Srinivas, A., Lin, T.-Y., Parmar, N., Shlens, J., Abbeel, P., and Vaswani, A. Bottleneck transformers for visual recognition. *arXiv preprint arXiv:2101.11605*, 2021.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. *ICML*, 2019a.

Tan, M. and Le, Q. V. Mixconv: Mixed depthwise convolutional kernels. *BMVC*, 2019b.

Tan, M., Chen, B., Pang, R., Vasudevan, V., and Le, Q. V. Mnasnet: Platform-aware neural architecture search for mobile. *CVPR*, 2019.

Tan, M., Pang, R., and Le, Q. V. Efficientdet: Scalable and efficient object detection. *CVPR*, 2020.

Touvron, H., Vedaldi, A., Douze, M., and Jégou, H. Fixing the train-test resolution discrepancy. *arXiv preprint arXiv:1906.06423*, 2019.

Touvron, H., Vedaldi, A., Douze, M., and Jégou, H. Fixing the train-test resolution discrepancy: Fixefficientnet. *arXiv preprint arXiv:2003.08237*, 2020.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2021.

Wightman, R. Pytorch image model. *https://github.com/rwightman/pytorch-image-models, Accessed on Feb.18, 2021*, 2021.

Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., and Keutzer, K. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. *CVPR*, 2019.

Xie, Q., Luong, M.-T., Hovy, E., and Le, Q. V. Self-training with noisy student improves imagenet classification. *CVPR*, 2020.

Xiong, Y., Liu, H., Gupta, S., Akin, B., Bender, G., Kindermans, P.-J., Tan, M., Singh, V., and Chen, B. Mobiledets: Searching for object detection architectures for mobile accelerators. *arXiv preprint arXiv:2004.14525*, 2020.

Yu, H., Liu, A., Liu, X., Li, G., Luo, P., Cheng, R., Yang, J., and Zhang, C. Pda: Progressive data augmentation for general robustness of deep neural networks. *arXiv preprint arXiv:1909.04839*, 2019.

Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Tay, F. E., Feng, J., and Yan, S. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. Mixup: Beyond empirical risk minimization. *ICLR*, 2018.

Zhang, H., Wu, C., Zhang, Z., Zhu, Y., Lin, H., Zhang, Z., Sun, Y., He, T., Mueller, J., Manmatha, R., Li, M., and Smola, A. Resnest: Split-attention networks. *arXiv preprint arXiv:2012.12877*, 2020.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. *CVPR*, 2018.