Compounding the Performance Improvements of Assembled Techniques in a Convolutional Neural Network

Jungkyu Lee, Taeryun Won, Kiho Hong

Clova Vision, NAVER Corp.

{jungkyu.lee, lory.tail, kiho.hong}@navercorp.com

Abstract

Recent studies in image classification have demonstrated a variety of techniques for improving the performance of Convolutional Neural Networks (CNNs). However, attempts to combine existing techniques to create a practical model are still uncommon. In this study, we carry out extensive experiments to validate that carefully assembling these techniques and applying them to a basic CNN model in combination can improve the accuracy and robustness of the model while minimizing the loss of throughput. For example, our proposed ResNet-50 shows an improvement in top-1 accuracy from 76.3% to 82.78%, and an mCE improvement from 76.0% to 48.9%, on the ImageNet ILSVRC2012 validation set. With these improvements, inference throughput only decreases from 536 to 312. The resulting model significantly outperforms state-of-the-art models with similar accuracy in terms of mCE and inference throughput. To verify the performance improvement in transfer learning, fine grained classification and image retrieval tasks were tested on several open datasets and showed that the improvement to backbone network performance boosted transfer learning performance significantly. Our approach achieved 1st place in the iFood Competition Fine-Grained Visual Recognition at CVPR 2019 [1], and the source code and trained models will be made publicly available [2].

Model	top-1	mCE	throughput
EfficientNet B4 [34]+AutoAugment [4]	83.0	60.7	95
EfficientNet B6 [34]+AutoAugment [4]	84.2	60.6	28
EfficientNet B7 [34]+AutoAugment [4]	84.5	59.4	16
ResNet-50 [9] (baseline)	76.3	76.0	536
Assemble-ResNet-50 (ours)	82.8	48.9	312
Assemble-ResNet-152 (ours)	84.2	43.3	143

Table 1. **Summary of key results.** *top-1* is ImageNet validation dataset accuracy. *mCE* [11] is mean corruption error (Lower is better). The *throughput* refers to how many images per second the model processes during inference.

1. Introduction

Since the introduction of AlexNet [20], many studies have mainly focused on designing new network architectures for image classification to increase accuracy. For example, new architectures such as Inception [32], ResNet [9], DenseNet [15], NASNet [41], MNASNet [33] and EfficientNet [34] have been proposed. Inception introduced new modules into the network with convolution layers of different kernel sizes. ResNet utilized the concept of skip connection, and DenseNet added dense feature connections to boost the performance of the model. In addition, in the area of AutoML, network design was automatically decided to create models such as NASNet and MNASNet. EfficientNet proposes an efficient network by balancing the resolution, height, and width of the network. The resulting performance of EfficientNet for ImageNet top-1 accuracy was greatly improved relative to AlexNet.

Unlike these studies which focus on designing new network architecture, He *et al*. [10] propose different approaches to improve model performance. They noted that performance can be improved not only through changes in the model structure, but also through other aspects of network training such as data preprocessing, learning rate decay, and parameter initialization. They also demonstrate that these minor "tricks" play a major part in boosting model performance when applied in combination. As a result of using these tricks, ImageNet validation top-1 accuracy of ResNet-50 improved from 75.3% to 79.29%. This improvement is hugely significant, and shows as much performance improvement as improvements to network design. Thus, it is of critical importance to combine these existing techniques.

Inspired by their works, we conducted a more extensive and systematic study of *assembling* several CNN-related techniques into a single network. When considering the

[1] https://www.kaggle.com/c/ifood-2019-fgvc6/leaderboard

[2] https://github.com/clovaai/assembled-cnn

many techniques that have been introduced, we first divided the techniques into two categories: network tweaks and regularization. Network tweaks are methods that modify the CNN architectures to be more efficient. For example, a representative work is SENet [14]. Regularization is a method that prevents overfitting by increasing the training data through data augmentation processes such as AutoAugment [4] and Mixup [39], or by limiting the complexity of the CNN with processes such as Dropout [31], and DropBlock [5]. Furthermore, we systematically analyze the process of assembling these two types of techniques through extensive experiments and demonstrate that our approach leads to significant performance improvements.

In addition to top-1 accuracy, mCE and throughput were used as performance indicators for combining these various techniques. Hendrycks *et al*. [11] proposed mCE (mean corruption error), which is a measure of network robustness against input image corruption. Moreover, we used throughput (images/sec) instead of the commonly used measurement of FLOPS (floating point operations per second) because we observed that FLOPS is not proportional to the inference speed of the actual GPU device. Detailed experiments on the discrepancy between the FLOPS and the throughput of GPU devices are included in the Appendix A.

Our contributions can be summarized as follows:

1. By organizing the existing CNN-related techniques for image classification, we find techniques that can be assembled into a single CNN. We then demonstrate that our resulting model surpasses the state-of-the-art models with similar accuracy in terms of mCE and inference throughput (Table 1).
2. We provide detailed experimental results for the process of assembling CNN techniques and release the code for accessibility and reproducibility.

2. Preliminaries

Before introducing our approach, we describe default experimental settings and evaluation metrics used in Sections 3 and 4.

2.1. Training Procedure

We use the official TensorFlow [1] ResNet [3] as base code. We use the ImageNet ILSVRC-2012 [29] dataset, which has 1.3M training images and 1,000 classes. All models were trained on a single machine with 8 Nvidia Tesla P40 GPUs compatible with the CUDA 10 platform and cuDNN 7.6. TensorFlow version 1.14.0 was used.

The techniques proposed by He *et al*. [10] are basically applied to all our models described in Section 3. We briefly describe the default hyperparameters and training techniques as follows.

[3]https://github.com/tensorflow/models

- **Preprocessing** In the training phase, a rectangular region is randomly cropped using a randomly sampled aspect ratio from 3/4 to 4/3, and the fraction of cropped area over whole image is randomly chosen from 5% to 100%. Then, the cropped region is resized as a 224x224 square image flipped horizontally with a random probability of 0.5. During validation, we first resize the shorter size of each image to 256 pixels while the aspect ratio is maintained. Next, we center crop the image to the 224x224 size and normalize the RGB channels, identically to training.
- **Hyperparameter** We use 1,024 batch sizes for training. In our study, this is close to the maximum size that can be received on a single machine with 8 P40 GPUs. The initial learning rate is 0.4 and weight decay is set to 0.0001. The default number of training epochs is 120, but some techniques require a different number of epochs, which is given explicitly when necessary. Stochastic gradient descent with momentum 0.9 is used as the optimizer.
- **Learning rate warmup** If a large batch size is set, using a high learning rate may result in numerical instability. Goyal *et al*. [8] proposes a warmup strategy that linearly increases the learning rate from 0 to the initial learning rate at warm-up periods set to first 5 epochs.
- **Zero γ** We initialize $\gamma = 0$ for all batch-norm layers that sit at the end of a residual block. Therefore, all the residual blocks return only their shortcut branch result in the early stages of training. It is easier to train by creating an effect that shrinks the entire layer at the initial stage.
- **Mixed-precision floating point** We only use mixed-precision floating point in the training phase because mixed-precision accelerates the overall training speed if the GPU supports it [24]. In our study, training speed for mixed-precision is 1.2 times faster than FP32 on an Nvidia P40 GPU, and is twice as fast as FP32 on an Nvidia V100 GPU. However, this does not result in improvement in top-1 accuracy.
- **Cosine learning rate decay (cosine)** The cosine decay [22] starts at a low rate from the beginning of training, and then drops to a large rate in the middle and again at a small rate in the end.

2.2. Evaluation Metrics

The selection of metrics used to measure the performance of the model is important because it indicates the direction in which the model is developed. We use the following three metrics as key indicators of model performance.

top-1 The top-1 is a measure of classification accuracy on the ImageNet ILSVRC-2012 [29] validation dataset. The validation dataset consists of 50,000 images of 1,000 classes.

throughput Throughput is defined as how many images are processed per second on the GPU device. We measured inference throughput for an Nvidia P40 1 GPU. For comparison with other models, we used FP32 instead of FP16 in our experiments, using a batch size of 64.

mCE The mean corruption error (mCE) was proposed by Hendrycks *et al*. [11] to measure the performance of the classification model on corrupted images.

3. Method

In this section, we introduce various network tweaks and regularization techniques to be assembled, and describe the details of the implementation. We also perform preliminary experiments to study the effect of different parameter choices.

3.1. Network Tweaks

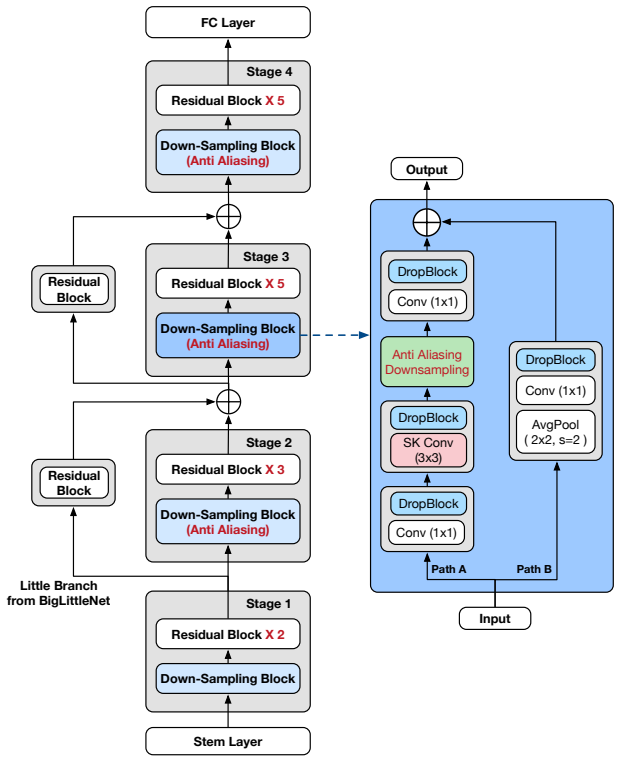
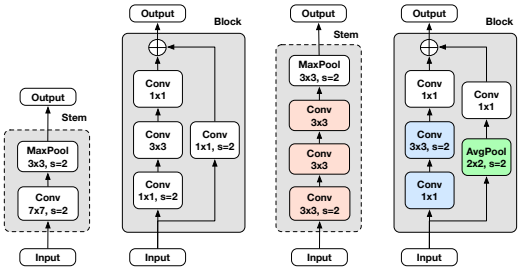


Figure 1. **Assembling techniques into ResNet-50**. We apply network tweaks such as ResNet-D, SK, Anti-alias, DropBlock, and BigLittleNet to ResNet. In more detail, ResNet-D and SK apply to all blocks in all stages. Downsampling with anti-aliasing only applies to the downsampling block from Stage 2 to Stage 4. DropBlock only applies all blocks in Stage 3 and Stage 4. Little-Branch from BigLittleNet uses one residual block with smaller width.

Figure 1 shows the overall flow of our final ResNet-50 model. Various network tweaks were applied to vanilla ResNet. The network tweaks we used are as follows.

ResNet-D ResNet-D is a minor adjustment to the vanilla ResNet network architecture model proposed by He *et al*. [10]. It is known to work well in practice and has little impact to computational cost [10]. Three changes were added to the ResNet model, as illustrated in Figure 2. First, the stride sizes of the first two convolutions have been switched. (Blue in Figure 2(b)). Second, a 2×2 average pooling layer has been added with a stride of 2 before the convolution (Green). Last, a large 7x7 convolution has been replaced with three smaller 3x3 convolutions in Stem layer (Red).



(a) ResNet (b) ResNet-D

Figure 2. Comparison of changes between ResNet and ResNet-D.

Channel Attention (SE, SK) We examine two tweaks in relation to channel attention. First, Squeeze and Excitation (SE) network [14] focuses on enhancing the representational capacity of the network by modeling channel-wise relationships. SE eliminates spatial information by global pooling to get channel information only, and then two fully connected layers in this module learn the correlation between channels. Second, Selective Kernel (SK) [21] is used, is inspired by the fact that the receptive sizes of neurons in the human visual cortex are different from each other. SK unit has multiple branches with different kernel sizes, and all branches are fused using softmax attention.

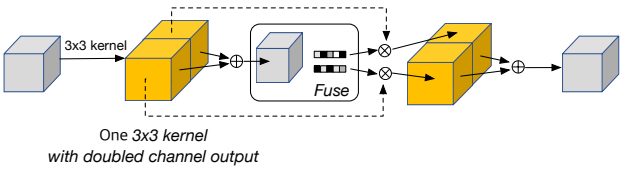


Figure 3. **Modified SK Unit.** We use one 3x3 kernel with doubled output channel size instead of 5x5 and 3x3 kernels.

The original SK generates multiple paths with 3x3 and 5x5 convolutions, but we instead use two 3x3 convolutions to split the given feature map. This is because two convolutions of the same kernel size can be replaced by a convolution with twice as many channels, thereby lowering the

inference cost. Figure 3 shows an SK unit that replaces two branches with one convolution operation.

Exp No.	Model	SK Configuration	SK r	top-1	throughput
C0	R50 (baseline)	-	-	76.30	536
C1	R50+SE	-	-	77.40	466
C2	R50+SK	3x3 + 5x5	2	78.00	326
C3	R50+SK†	3x3, 2x-channel	2	77.92	382
C4	R50+SK	3x3, 2x-channel	16	77.57	402
C5	R50+SK+SE	3x3, 2x-channel	2	77.50	345

Table 2. Result of channel attention with different configurations. R50 is a simple notation for ResNet-50. In these experiments, the learning rate starts from 0.4 and is decayed by 0.1 at 30, 60 and 90 epochs and is trained for 120 epochs. r is the reduction ratio of SK in the *Fuse* operation.

Table 2 shows the results for different configurations of channel attention. Compared with SK, SE has higher throughput but lower accuracy (C1 in Table 2). Comparing R50+SK† (C3) to R50+SK with 3x3 and 5x5 kernels (C2), the top-1 accuracy only differs by 0.08% (78.00 and 77.92), but the throughput is significantly different (326 and 382). Considering the trade-off between accuracy and throughput, we used one 3x3 kernel with doubled channel size instead of 3x3 and 5x5 kernels. Comparing C3 and C4, we see that changing the setting of reduction ratio r for SK units from 2 to 16 yields a large degradation of top-1 accuracy relative to throughput improvement. Applying both SE and SK (C5) not only decreases accuracy by 0.42 (from 77.92 to 77.50), but also decreases inference throughput by 37 (from 382 to 345). For a better trade-off between top-1 accuracy and throughput, R50+SK† is preferred.

Anti-Alias Downsampling (AA) CNN models for image classification are known to be very vulnerable to small amounts of distortion [37]. Zhang *et al.* [40] proposes AA to improve the shift-equivariance of deep networks. The max-pooling is commonly viewed as a competing downsampling strategy, and is inherently composed of two operations. The first operation is to densely evaluate the max operator and second operation is naive subsampling [40]. AA is proposed as a low-pass filter between them to achieve practical anti-aliasing with any existing strided layer such as strided-conv.

In the original paper, AA applies to max-pooling, projection-conv, and strided-conv of ResNet. In addition, the smoothing factor can be adjusted by changing the blur kernel filter size, where a larger filter size results in increased blur. Table 3 shows the experimental results for AA. We observed that reducing the filter size from 5 to 3 maintains the top-1 accuracy while increasing inference throughput (A1,2 in Table 3). However, removing the AA applied to the projection-conv does not affect the accuracy (A3). We also observe that applying AA to max-pooling degrades throughput significantly (A1-3). Finally, we apply AA only to strided-conv in our model (Green in Figure 1).

Exp No.	Filter Size	Max Pooling	Pro-jection	Strided Conv	top-1	throughput
A0	-	X	X	X	76.30	536
A1	5	O	O	O	76.81	422
A2	3	O	O	O	76.83	456
A3	3	O	X	O	76.84	483
A4	3	X	X	O	76.67	519

Table 3. Results for downsampling with anti-aliasing. The performance of the model was tested according to a range of applications for downsampling with anti-aliasing. All experiments were tested based on ResNet-50. We applied downsampling with anti-aliasing only to strided-conv based on the result of the experiment. In these experiments, the learning rate starts from 0.4 and is decayed by 0.1 at 30, 60 and 90 epochs and is trained for 120 epochs.

Big Little Network (BL) BigLittleNet [3] applies multiple branches with different resolutions while aiming at reducing computational cost and increasing accuracy. The Big-Branch has the same structure as the baseline model and operates at a low image resolution, whereas the Little-Branch reduces the convolutional layers and operates at same image resolution as the baseline model.

BigLittleNet has two hyperparameters, α and β, which adjust the width and depth of the Little-Branch, respectively. We use α=2 and β=4 for ResNet-50 and use α=1 and β=2 for ResNet-152. The left small branch in Figure 1 represents the Little-Branch. The Little-Branch has one residual block and is smaller in width than the main Big-Branch.

3.2. Regularization

AutoAugment (Autoaug) AutoAugment [4] is a data augmentation procedure which learns augmentation strategies from data. It uses reinforcement learning to choose a sequence of image augmentation operations with the best accuracy by searching in a discrete search space of their probability of application and magnitude. We borrow the augmentation policy found by Autoaug on ImageNet ILSVRC-2012 [4].

Mixup Mixup [39] creates one example by interpolating two examples of the training set for data augmentation. Neural networks are known to memorize training data rather than generalizing from the data [38]. As a result, the neural network produces unexpected outputs when it encounters data that differs from the distribution of the training set. Mixup mitigates these problems by showing the neural network interpolated examples of filling empty space on the feature space of the training dataset.

[4] https://github.com/tensorflow/models/tree/master/research/autoaugment

Mixup has two types of implementation. The first type uses two mini batches to create a mixed mini batch. this type of implementation is suggested in the original paper [39]. The second type uses a single mini batch to create the mixed mini batch by mixing the mini batch with a shuffled clone of itself. The second type of implementation uses less CPU resources because only one mini batch needs to be preprocessed to create one mixed mini batch. However, experiments show that the second type of implementation reduces top-1 accuracy (Table 4). Therefore, in later experiments, we use the first type of implementation. We set the Mixup hyperparameter α to 0.2.

Model	Configuration	top-1
R50D (E2)	LS	77.37
R50D	+ Mixup (type=2)	78.85
R50D (E3)	+ Mixup (type=1)	**79.10**

Table 4. Result of the change of Mixup implementation type. We choose type=1 for the next experiment. E2 and E3 correspond to the experiment numbers given in Table 7.

DropBlock Dropout [31] is a popular technique for regularizing deep neural networks. It prevents the network from over-fitting the training set by dropping neurons at random. However, Dropout does not work well for extremely deep networks such as ResNet [7]. DropBlock [7] can remove specific semantic information by dropping a continuous region of activations. Thus, it is efficient for the regularization of very deep networks. We borrow the same DropBlock setting used in the original paper. We apply DropBlock to Stages 3 and 4 for ResNet-50 and linearly decay the $keep\_prob$ hyperparameter from 1.0 to 0.9 during training.

Label Smoothing (LS) In the classification problem, class labels are expressed as one hot encoding. If CNN trains to minimize cross entropy with this hard one hot encoding target, the logits of the last fully connected layer of CNN grow to infinity, which leads to over-fitting [10]. Label smoothing [27] suppresses infinite output and prevents over-fitting. We set the label smoothing factor ϵ to 0.1.

Knowledge Distillation (KD) Knowledge Distillation [13] is a technique for transferring knowledge from one neural network (teacher) to another (student). Teacher models are often complex but cumbersome models with high accuracy, and a weak but light student model can improve its own accuracy by mimicking a teacher model. The T hyperparameter of KD was said to be optimal when set to 2 or 3 in the originally paper [13], but we use T=1 for our model. Because our model uses Mixup and KD techniques together, the teacher network should also be applied to Mixup. This leads to better performance at lower temperatures because the teacher's signal itself is already smoothed by the Mixup (Table 5). We used AmoebaNet-A as a teacher with 83.9% of ImageNet validation top-1 accuracy.

Model	Configuration	top-1
R50D+SK (E6)	LS+Mixup+DropBlock	81.40
R50D+SK	+ KD (T=2)	81.47
R50D+SK	+ KD (T=1.5)	81.50
R50D+SK (E7)	+ KD (T=1)	**81.69**

Table 5. Result of the change of KD temperature. On top of the E6 (described in Table 7), we apply KD by varying the temperature T to find the optimal T value. We choose T=1 for next experiment.

4. Experiment Results

4.1. Ablation Study

In this section we will describe ablation experiments for assembling the individual network tweaks covered in Section 3.1 to find better model. The results are shown in Table 6.

Exp. No.	Model	Input Size	top-1	throughput
M0	R50 (baseline)	224	76.87	536
M1	R50**D**	224	77.37	493
M2	R50D+**SK**	224	78.83	359
M3	R50D+SK+**BL**	**256**	79.27	359
M4	R50D+SK+BL+**AA**	256	79.39	312

Table 6. Performance comparison of stacking network tweaks. By stacking the ResNet-D, Selective Kernel (SK), BigLittleNet (BL) and downsampling with anti-aliasing (AA), we have steadily improved the ResNet-50 model with some inference throughput losses. The focus of each experiment is highlighted in boldface. The cosine learning rate decay is used in all experiments.

Adding ResNet-D to the baseline model improves top-1 accuracy by 0.5% (from 76.87 to 77.37) (M1 in Table 6), and adding SK tweaks improves accuracy by 1.46% (from 77.37 to 78.83) (M2). In Table 2, We show that the accuracy is increased by 1.62% when SK is independently applied to ResNet (from 76.30 to 77.92). Stacking ResNet-D and SK increases the top-1 accuracy gains almost in equal measure to the sum of the performance gains of applying ResNet-D and SK separately. The results show that the two tweaks can improve performance independently with little effect on each other. Applying BL to R50D+SK improves top-1 accuracy by 0.44% (from 79.27 to 78.83) (M3). To achieve higher accuracy while maintaining throughput similar to that of the R50D+SK, we use a 256x256 image resolution for inference, whereas we use 224x224 image res-

Exp. No.	Model	Regularization Configuration	Train Epoch	Input Size	top-1	top-1 Δ	mCE	mCE Δ	throughput
	EfficientNet B0 [34]	Autoaug	-	224	77.3	-	70.7	-	510
	EfficientNet B1 [34]	Autoaug	-	240	79.2	-	65.1	-	352
	EfficientNet B2 [34]	Autoaug	-	260	80.3	-	64.1	-	279
	EfficientNet B3 [34]	Autoaug	-	300	81.7	-	62.9	-	182
	EfficientNet B4 [34]	Autoaug	-	380	83.0	-	60.7	-	95
	EfficientNet B5 [34]	Autoaug	-	456	83.7	-	62.3	-	49
	EfficientNet B6 [34]	Autoaug	-	528	84.2	-	60.6	-	28
	EfficientNet B7 [34]	Autoaug	-	600	84.5	-	59.4	-	16
E0	R50 (baseline)		120	224	76.87	0.00	75.55	0.00	536
E1	R50**D**		120	224	77.37	0.50	75.73	-0.18	493
E2	R50D	**LS**	120	224	78.35	0.98	74.27	1.46	493
E3	R50D	LS+**Mixup**	200	224	79.10	0.75	68.19	6.08	493
E4	R50D+**SE**	LS+Mixup	200	224	79.71	0.61	64.48	3.71	420
E5	R50D+SE	LS+Mixup+**DropBlock**	270	224	80.40	0.69	62.64	1.84	420
E6	R50D+**SK**	LS+Mixup+DropBlock	270	224	81.40	1.00	58.34	4.30	359
E7	R50D+SK	LS+Mixup+DropBlock+**KD**	270	224	81.69	0.29	57.08	1.26	359
E8	R50D+SK	LS+Mixup+DropBlock+KD	**600**	224	82.10	0.41	56.48	0.60	359
E9	R50D+**BL**+SK	LS+Mixup+DropBlock+KD	600	**256**	82.44	0.34	55.20	1.28	359
E10	R50D+BL+SK+**AA**	LS+Mixup+DropBlock+KD	600	256	82.69	0.25	54.12	1.08	312
E11	R50D+BL+SK+AA	LS+Mixup+DropBlock+KD+**Autoaug**	600	256	82.78	0.09	48.89	4.14	312
E12	**R152D**+BL+SK+AA	LS+Mixup+DropBlock+**KD**†+Autoaug	600	256	84.19	1.41	43.27	5.62	143

Table 7. Ablation study for assembling the network tweaks and regularization with ResNet-50 on ImageNet ILSVRC2012 dataset. The focus of each experiment is highlighted in boldface. The cosine learning rate decay is used in all experiments. The top-1 accuracy and mCE scores for EfficientNet are borrowed from the official code in [17] and [37] respectively. As with other experiments, the inference throughput measurements of EfficientNet were performed on a single NVIDIA Tesla P40 using official EfficientNet code [17]. For comparison, we also experiment with ResNet-152. KD† uses EfficientNet-B7 instead of AmoebaNet as a teacher model.

olution for training. Applying AA to the R50D+SK+BL improves top-1 accuracy by 0.12% (from 79.27 to 79.39) and decreases throughput by 47 (from 359 to 312) (M4). Because AA is a network structure designed for robustness to image distortion, the top-1 accuracy does not reliably determine the AA effect. The effect on AA is further shown in the next section, wherein mCE is introduced to evaluate models.

The ablation study described in Table 7 shows the impact of assembling the techniques described in Section 3.2. We stack the network tweaks and regularizations alternately to balance the performance effects.

The regularization techniques increase both accuracy and mCE, but the performance improvement effect of mCE is greater than the improvement of accuracy (E2,3,5,7,11). For example, applying Mixup, DropBlock, KD, and Autoaug individually improves top1/mCE 0.75%/6.08%, 0.69%/1.84%, 0.29%/1.26%, and 0.09%/4.14% respectively. It can be seen that regularization help to make CNNs more robust to image distortions.

Adding SE improves top-1 accuracy by 0.61% and improves mCE by 3.71% (E4). SE having a greater effect on mCE enhancement than top-1 accuracy is similar to the result of the regularization techniques. We confirm that channel attention is also helpful for robustness to image distortion.

Replacing SE with SK improves performance by 1.0% and 4.3% for the top-1 and mCE, respectively (E6). In Table 2, when SE is changed to SK without regularization, the accuracy increases by 0.5%. Compared to SK without regularization, replacing SE with SK with regularization leads to nearly double the accuracy improvement. This means that SK is more complementary for regularization techniques than SE.

Changing the epochs from 270 to 600 improves performance (E8). Because data augmentation and regularization are stacked, they have a stronger effect of regularization, so longer epochs seems to yield better generalization performance. BL shows a high performance improvement not only on top-1, but also on mCE, and without inference throughput loss (E9). AA also shows higher performance gain in mCE relative to top-1 (E10), which agrees with AA being used as a network tweak to make the CNN robust for image translations as claimed in [40].

The assembled model of all the techniques described so far has a top-1 accuracy of 82.78% and an mCE of 48.89%. This final model is listed in Table 7 as E11, and we call this model **Assemble-ResNet-50**. We also experiment with ResNet-152 for comparison as E12, we call this model **Assemble-ResNet-152**.

Exp. No.	Backbone Model	Backbone top-1	Regularization	Food-101 top-1	Food-101 mCE
F0	R50 (baseline)	76.87	-	86.99	61.50
F1	R50D	77.37	-	87.63	62.12
F2	R50D+**SK**	78.83	-	89.77	57.20
F3	R50D+SK+**BL**	79.27	-	90.15	57.16
F4	R50D+SK+BL+**AA**	79.39	-	90.37	56.66
F5	R50D+SK+BL+AA	79.39	**DropBlock**	91.25	53.13
F6	R50D+SK+BL+AA	79.39	DropBlock+**Mixup**	91.64	48.53
F7	R50D+SK+BL+AA	79.39	DropBlock+Mixup+**Autoaug**	91.85	41.73
F8	R50D+SK+BL+AA	79.39	DropBlock+Mixup+Autoaug+**LS**	91.76	41.40
F9	R50D+SK+BL+AA+***REG***	82.78	-	90.63	53.98
F10	R50D+SK+BL+AA+*REG*	82.78	**DropBlock**	91.62	51.01
F11	R50D+SK+BL+AA+*REG*	82.78	DropBlock+**Mixup**	92.11	45.73
F12	R50D+SK+BL+AA+*REG*	82.78	DropBlock+Mixup+**Autoaug**	92.21	41.69
F13	R50D+SK+BL+AA+*REG*	82.78	DropBlock+Mixup+Autoaug+**LS**	92.47	41.99

Table 8. Ablation study of transfer learning with the Food-101 dataset. *REG* means that regularization techniques "LS + Mixup + DropBlock + KD + Autoaug" are applied during backbone training. The Food-101 mCE is not normalized by AlexNet's errors. We use the augmentation policy which is found by Autoaug on CIFAR-10 in these experiments [4].

Dataset	The state-of-the-art Models		ResNet-50 -tuned [18]	ResNet-50	Assemble-ResNet-FGVC-50
Food-101	EfficientNet B7 [34]	93.0	87.8	87.0	92.5
Stanford Cars	EfficientNet B7 [34]	94.7	91.7	89.1	94.4
Oxford-Flowers	EfficientNet B7 [34]	98.8	97.5	96.1	98.9
FGVC Aircraft	EfficientNet B7 [34]	92.9	86.6	78.8	92.4
Oxford-IIIT Pets	AmoebaNet-B [16]	95.9	91.5	92.5	94.3

Table 9. Transfer learning for FGVC task with our model and comparison with other state-of-the-art models. ImageNet-based transfer learning results are only compared with a single crop. Assemble-ResNet-FGVC-50 means that the final F13 model in Table 8. ResNet-50-tuned is a model in which the learning rate and weigh decay are tuned as in [18]. Unfortunately, Kornblith *et al*. [18] did not specify optimal hyperparameters for each datasets. We did not tune hyperparameters because we aim to compare between ResNet-50 and Assemble-ResNet-FGVC-50 rather than to achieve the state-of-the-art performance. Assemble-ResNet-FGVC-50 not only boosts the performance of ResNet-50 significantly, but also leads to performance comparable to heavyweight state-of-the-art models for all datasets.

4.2. Transfer Learning: FGVC

In this section, we will investigate whether these improvements discussed so far can help with transfer learning. Before that, we first need to analyze the contribution of transfer learning for each technique. To do this, we performed an ablation study on the Food-101 [2] dataset, which is the largest public fine-grained visual classification (FGVC) dataset. The basic experiment setup and hyperparameters that differ from the backbone training are:

- Initial learning rate reduced from 0.1 to 0.01.
- Weight decay is set to 0.001.
- Momentum for BN is set to $\max(1 - 10/s, 0.9)$.
- Keep probability of DropBlock starts at 0.9 and decreases linearly to 0.7 at the end of training
- The training epoch is set differently for each dataset and is indicated in Appendix B.

As shown in Table 8, stacking network tweaks and regularization techniques have steadily improved both top-1 accuracy and mCE for the transfer learning task on the Food-101 dataset. In particular, comparing the experiments F4-F8 with experiments F9-F13 (in Table 8) shows the effect of regularization on the backbone. We use the same network structure in F4-F13, but for F9-F13, they have regularization such as Mixup, DropBlock, KD and Autoaug on the backbone. This regularization of the backbone gives performance improvements for top-1 accuracy as expected. On the other hand, the aspect of mCE performance differed from the top-1 accuracy. Without regularization during fine-tuning such as in F4 and F9, the backbone with regularization leads to better mCE performance than backbone with-

out regularization. However, adding regularization during fine-tuning narrows the mCE performance gap (F5-8 and F10-13). For convenience, we call the final F13 model in Table 8 as Assemble-ResNet-FGVC-50.

We also have evaluated Assemble-ResNet-FGVC-50 in Table 8 on the following datasets: Stanford Cars [19], CUB-200-2011 [35], Oxford 102 Flowers [25], Oxford-IIIT Pets [26], FGVC-Aircraft [23], and Food-101 [2]. The statistics for each dataset are as shown in Appendix C. We borrow the same training settings from Kornblith *et al*. [18] and fine-tuned new datasets from Assemble-ResNet-50 ImageNet checkpoint.

Table 9 shows the transfer learning performance. Compared to EfficientNet [34] and AmoebaNet-B [16] which are state-of-the-art model for image classification tasks. Our Assemble-ResNet-FGVC-50 model achieves comparable accuracy with 20x faster inference throughput than the existing state-of-the-art models.

4.3. Transfer Learning: Image Retrieval

Exp. No.	Backbone	Regularization	recall@1
S0	R50 (baseline)		82.9
S1	R50D		84.2
S2	R50D+SK		85.4
S3	R50D+SK+BL		85.2
S4	R50D+SK+BL+AA		85.1
S5	R50D+SK	DropBlock	85.9
S6	R50D+SK	DropBlock+Autoaug	83.7
S7	R50D+SK + *REG*		85.2
S8	R50D+SK + *REG*	DropBlock	85.9
S9	R50D+SK + *REG*	DropBlock+Autoaug	84.0

Table 10. Ablation study of transfer learning with SOP dataset. *REG* means "LS + Mixup + DropBlock + KD".

We also conducted an ablation study on three standard fine-grained image retrieval (IR) datasets: Stanford Online Products (SOP) [30], CUB200 [35] and CARS196 [19]. We borrow the zero-shot data split protocol from [30].

The basic experiment setup and hyperparameters are as follows.

- Image preprocessing resizes to 224x224 without maintaining aspect ratio with probability 0.5 and resize to 256x256 and random crop to 224x224 with probability 0.5.
- Data augmentation includes random flip with 0.5 probability.
- Momentum for BN is set to $\max(1 - 10/s, 0.9)$.
- Weight decay is set to 0.0005.
- The training epoch, batch size, learning rate decay and assembling configuration is set differently for each dataset. We will describe the settings in the Appendix D.

On top of that, cosine-softmax based losses were used for image retrieval. In this work, we use ArcFace [6] loss with a margin of 0.3 and use generalized mean-pooling (GeM) [28] for a pooling method without performing downsampling at Stage 4 of backbone networks because it has better performance for the image retrieval task.

In the case of SOP, the degree of the effect was examined by an ablation study with the results listed in Table 10. In contradiction to our results on FGVC, the particular combination of network tweaks and regularization that worked well on the SOP dataset were different from that for FGVC datasets. Comparing S2-4, we see that BL and AA did not work well on the SOP dataset. Of the regularizers, DropBlock works well, but Autoaug do not improve the recall@1 performance (S2 and S5,6). Nevertheless, in the best configuration, there was a significant performance improvement of 3.0% compared to the baseline ResNet-50.

The recall at 1 for image retrieval datasets (recall@1) are reported in Table 11. There is also a significant performance improvement on CUB200 and CARS196 datasets.

Dataset	ResNet-50	Assemble-ResNet-IR-50
SOP	82.9	85.9
CUB200	75.9	80.3
CARS196	92.9	96.1

Table 11. Transfer learning for IR task with our method. Assemble-ResNet-IR-50 means that the best configuration model for each dataset. The best configurations for each dataset are specified in Appendix D. Assemble-ResNet-IR-50 boosts the performance of ResNet-50 significantly for all IR datasets.

5. Conclusion

In this paper, we show that assembling various techniques for CNNs to single convolutional networks leads to improvements of top-1 accuracy and mCE on the ImageNet ILSVRC2012 validation dataset. Synergistic effects have been achieved by using a variety of network tweaks and regularization techniques together in a single network. Our approach has also improved performance consistently on transfer learning such as FGVC and image retrieval tasks. More excitingly, our network is not frozen, but is still evolving, and can be further developed with future research. For example, we already are planning to reassemble various new studies such as AugMix [12] and ECA-net [36], which were recently published. We expect that there will be further improvements if we change the vanilla backbone to a

more powerful backbone such as EfficientNet [34] instead of ResNet, which we leave as future works.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European Conference on Computer Vision*, pages 446–461. Springer, 2014.

[3] Chun-Fu Chen, Quanfu Fan, Neil Mallinar, Tom Sercu, and Rogerio Feris. Big-little net: An efficient multi-scale feature representation for visual and speech recognition. *arXiv preprint arXiv:1807.03848*, 2018.

[4] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[5] Zuozhuo Dai, Mingqiang Chen, Xiaodong Gu, Siyu Zhu, and Ping Tan. Batch dropblock network for person re-identification and beyond. *arXiv preprint arXiv:1811.07130*, 2018.

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

[7] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems*, pages 10727–10737, 2018.

[8] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

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

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

[11] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[12] Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019.

[13] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[14] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

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

[16] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in Neural Information Processing Systems*, pages 103–112, 2019.

[17] Google Inc. Efficientnet official code. https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet, 12 2019.

[18] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2661–2671, 2019.

[19] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[21] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. 2019.

[22] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[23] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[24] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

[25] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008.

[26] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE, 2012.

[27] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

[28] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation. *IEEE transactions on pattern analysis and machine intelligence*, 41(7):1655–1668, 2018.

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[30] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

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

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

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

[35] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[36] Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu. Eca-net: Efficient channel attention for deep convolutional neural networks, 2019.

[37] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan Yuille, and Quoc V Le. Adversarial examples improve image recognition. *arXiv preprint arXiv:1911.09665*, 2019.

[38] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[39] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[40] Richard Zhang. Making convolutional networks shift-invariant again. In *ICML*, 2019.

[41] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

Appendices

A. FLOPS and throughput

We observe in several experiments that FLOPS is not proportional to the inference speed of the actual GPU.

Model	FLOPS	throughput
EfficientNet B0	0.39B	510
EfficientNet B1	0.70B	352
EfficientNet B2	1.0B	279
EfficientNet B3	1.8B	182
EfficientNet B4	4.2B	95
EfficientNet B5	9.9B	49
EfficientNet B6	19B	28
EfficientNet B7	37B	16
R50	4.1B	536
R50D	4.4B	493
R50D+SK	6.2B	359
R50D+SK+BL	5.4B	359
R50D+SK+BL+AA	7.2B	312
R152D+SK+BL+AA	15.8B	143

Table 12. FLOPS change with Model Tweak. We use the TensorFlow official profiler code to measure FLOPS. EfficientNet's FLOPS is borrowed from [34].

B. FGVC Configuration

We use the same hyperparameters for as all datasets as possible for transfer learning. The different parameters settings for each dataset are described in Table 13.

FGVC Dataset	Training Epochs
Food-101	100
Stanford Cars	1,000
Oxford-Flowers	1,000
FGVC Aircraft	800
Oxford-IIIT Pets	1,300

Table 13. Training configuration of FGVC datasets.

C. FGVC Datasets

Dataset	Train Size	Test Size	# Classes
Food-101	75,750	25,250	101
Stanford Cars	8,144	8,041	196
Oxford-Flowers	2,040	6,149	102
FGVC Aircraft	6,667	3,333	100
Oxford-IIIT Pets	3,680	3,669	37

Table 14. Statistics of FGVC datasets.

D. IR Configuration

IR uses a different regularization for each dataset.

Dataset	Backbone	Regularization
SOP	R50D+SK+*REG*	DropBlock
CUB200	R50D+SK+*REG*	DropBlock
CARS196	R50D+SK+*REG*	DropBlock+LS+Autoaug

Table 15. Model configuration for IR tasks. *REG* means "LS + Mixup + DropBlock + KD"

We use the same hyperparameters for as all datasets as possible for transfer learning. The parameters set differently for each dataset are described in Table 16.

Dataset	Loss Fuction	Learning rate	Batch size	Feature size	Training Epochs
SOP	Arcface	0.008	128	1536	60
CUB200	Softmax	0.001	32	1536	100
CARS196	Softmax	0.01	32	1536	100

Table 16. Different hyperparameters setting for IR tasks.