Efficient Model for Image Classification With Regularization Tricks

Taehyeon Kim potter32@kaist.ac.kr
Graduate School of AI, KAIST, Daejeon, Republic of Korea

Jonghyup Kim leenams2@kaist.ac.kr
Department of Industrial and Systems Engineering, KAIST, Daejeon, Republic of Korea

Seyoung Yun yunseyoung@gmail.com
Graduate School of AI, KAIST, Daejeon, Republic of Korea

Editors: Hugo Jair Escalante and Raia Hadsell

Abstract

In the MicroNet Challenge 2019, competitors attempted to design the neural network architecture with fewer resource budgets, e.g., the number of parameters and FLOPS. In this study, we describe the approaches of team KAIST, using which they won the second and third places, respectively, in the CIFAR-100 classification task in the contest. We solve the task into four steps. First, we design a novel baseline network appropriate for the CIFAR-100 dataset. Second, we train this network using our novel structural regularization methods, which penalize the orthogonality of weights and replace the ground-truth label of each data with a noise vector that has class-wise similarity information from the representative feature vectors of each class in the course of training. Third, we seek the most potent data-augmentation methods for significant improvements in accuracy. At last, we perform the sparse training via a pruning technique. Our final score is 0.0054, which represents **370x** improvements over the baseline for the CIFAR-100 dataset. This is the only work that finished in the top 10% of both parameter storage and computation over the CIFAR-100 classification task. The source code is at https://github.com/Kthyeon/micronet_neurips_challenge.

Keywords: Image Classification, FLOPS, Orthogonality Regularization, Pruning, efficient model, data augmentation.

1. Introduction

Despite the massive success of deep neural networks (DNN) in various tasks of machine learning, it is still challenging to deploy them in real-world applications, owing to requiring large resource budgets, e.g., the number of parameters and FLOPS. With the rapid growth of interest in DNNs, there has been a lot of recent literatures such as designing resource-efficient architectures (Howard et al., 2017a; Sandler et al., 2018; Tan and Le, 2019; Howard et al., 2019), quantizing the weight parameters (Banner et al., 2018; Zhao et al., 2019), and training a DNN with sparsity (Frankle and Carbin, 2018; Lee et al., 2018), among others. On the other side, still, only a few studies are based on how to utilize the above-mentioned techniques together.

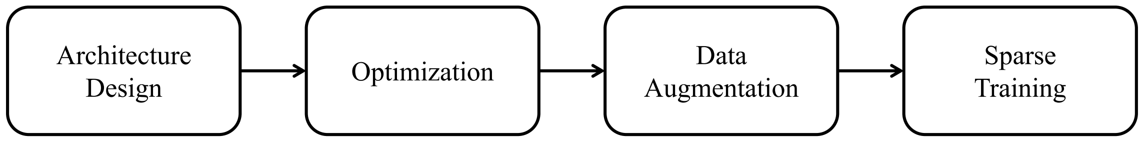


Figure 1: An overview of our solution in the task of the CIFAR-100 Classification.

The MicroNet challenge 2019 focused on this issue. It aimed to build the most efficient model that could solve a target task while achieving a specified quality level. In the task of the CIFAR-100 classification in which we participated, the participants had to design a model whose top-1 accuracy exceeded 80% on the CIFAR-100 dataset.

Our team achieved the target performance using significantly few resources via the following four stages : *(1)* architecture design, *(2)* optimization, *(3)* data augmentation, and *(4)* sparse training (see Figure 1). First, we designed a resource-efficient CNN architecture via extensive comparison experiments. Secondly, to improve the accuracy before the compression stage, we applied our novel structural regularizations, orthogonality regularization (ORN), and adaptive label smoothing (ALS) methods. ORN is not entirely new, but we changed places to apply the ORN. More precisely, we penalized only the convolutional layers whose kernel size was equal to 1, while the orthogonalities of all the filters were considered under the conventional ORN (Bansal et al., 2018). ALS is an enhanced label smoothing approach that uses a label-noise vector instead of the one-hot encoded vector, and it exploits the class-wise similarity to generate the label-noise vectors. We then deployed both FastAutoaugmentation (Lim et al., 2019a) and Cutmix (Yun et al., 2019a) algorithms in the CIFAR-100 dataset. Finally, we pruned the model via a modified Lottery ticket Hypothesis strategy (Frankle and Carbin, 2018) that prunes the network using both posterior information and initial-stage information. Our final score was 0.0054, which won us the 2nd place in this contest.

1.1. Challenge Overview

The challenge hosted at NeurIPS 2019 comprised three different tasks: ImageNet classification, CIFAR-100 classification, and WikiText-103 language modeling. Our team participated in the CIFAR-100 classification task, whose dataset consists of 50,000 training images and 10,000 development images, and the participants had to achieve at least 80% top-1 accuracy. The evaluation metric (score) was defined as follows:

$$Score = \frac{NP}{3.65 \times 10^7} + \frac{MO}{1.049 \times 10^{10}} \tag{1}$$

where NP denotes the number of parameters required to perform inference on the test set, and MO denotes the mean number of arithmetic operations per example in inference time. Both NP and MO can be reduced using quantization and sparsification methods. The detailed scoring policy can be found at the following link[1]. The main objective of this challenge was to obtain the lowest possible score.

1. https://micronet-challenge.github.io/scoring_and_submission.html

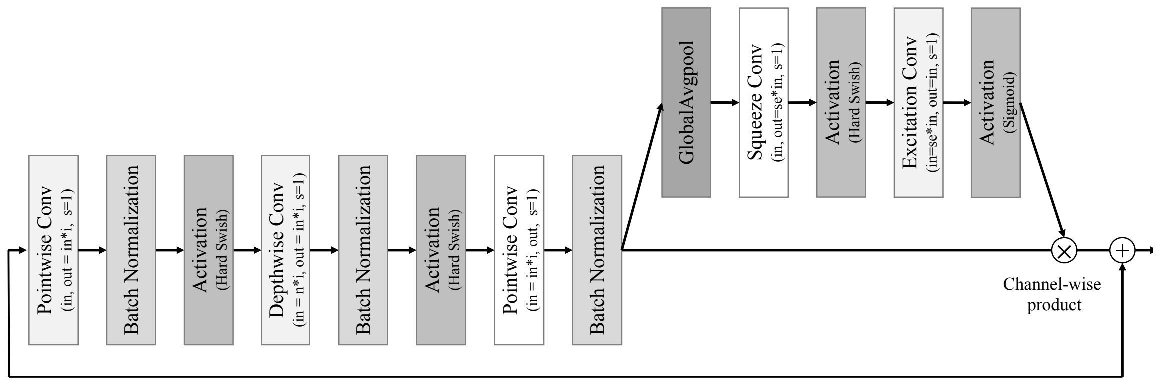

Pointwise Conv (in, out = in*i, s=1)
Batch Normalization
Activation (Hard Swish)
Depthwise Conv (in = n*i, out = in*i, s=1)
Batch Normalization
Activation (Hard Swish)
Pointwise Conv (in = in*i, out, s=1)
Batch Normalization
GlobalAvgpool
Squeeze Conv (in, out=se*in, s=1)
Activation (Hard Swish)
Excitation Conv (in=se*in, out=in, s=1)
Activation (Sigmoid)
Channel-wise product

Figure 2: An overview of the main block (i.e., MBConv) used in a family of C-EfficientNet-i where 'in' is the number of input channels, 'out' is the number of output channels, 'se' is a se ratio, and 's' indicates the stride size.

1.2. Paper Outline

In this paper, we describe how to build a model to reduce both the number of parameters and operations. We first design our baseline network in Section 2, and then search for the most potent optimization methods over our baseline network in Section 3. In Section 4, we apply various data-augmentation strategies on our model. Finally, in Section 5, we study training methods to sparsify the network to the maximum possible extent while maintaining the top-1 accuracy at 80%.

2. Architecture Design

In this section, we first introduce our baseline network, coined as C-EfficientNet, and then find the best training setups such as batch size, weight decay, and learning rate.

2.1. Efficient Architecture

We created our baseline network, coined as C-EfficientNet-i, by changing the internal activation function, resolution, and the number of channels from EfficientNet (Tan and Le, 2019). Table 1 presents the C-EfficientNet-i structure.

Main block. It is the mobile inverted bottleneck MBConv i (Sandler et al., 2018; Tan and Le, 2019) where i denotes the expansion ratio, i.e., the number of input channels to that of output channels. At the end of every main block, we added a squeeze-and-excitation block (Hu et al., 2018) with the reduction ratio of 0.25. Figure 2 depicts an overview of the main block of C-EffientNet families.

Activation function. For training efficiency in terms of the computation speed, we replaced Swish activation functions (Ramachandran et al., 2017) with HardSwish ones (Howard et al., 2019). The Swish function has the disadvantage of high computational cost, and HardSwish resolves this disadvantage by replacing the exponential part of the Swish function with polynomials.

Resolution and the number of channels. We set not only the location of the layer that reduces resolution, but also the number of channels via the grid search. To define the search space, we consider the ratio between the numbers of classes of CIFAR-100 and ImageNet datasets, as EfficientNet is befitted for ImageNet. We configured all the blocks to have the same expansion ratio, i.e., i.

Stage	Operator	Resolution	Channels	Layers
1	Conv3x3	32x32	32	1
2	MBConv i, k3x3	32x32	20	2
3	MBConv i, k3x3	16x16	36	3
4	MBConv i, k3x3	16x16	56	3
5	MBConv i, k3x3	8x8	80	5
6	MBConv i, k3x3	4x4	88	1
7	MBConv i, k3x3	4x4	96	2
8	MBConv i, k3x3	4x4	114	1
9	Pooling & FC	4x4	114	1

Table 1: **C-EfficientNet-i.** Each row describes a stage with layers.

2.2. Training Procedure

We trained all C-EfficientNet-i using the standard pytorch SGD optimizer with the Nesterov momentum of 0.9 for 600 epochs (Sutskever et al., 2013). A cosine annealing scheduler was used to control the learning rate. Its initial and minimum learning rates were 0.1 and 0.0005, respectively, with the batch size of 128 (Loshchilov and Hutter, 2016). We used a dropout of 0.3, and l2 weight decay of 1e-5. Each of our convolutional layers used a batch-normalization layer with the average decay of 0.99.

No-bias decay. We applied the weight decay to neither biases, nor γ, nor β in BN layers. This follows from the recent studies (Jia et al., 2018; He et al., 2019), which claimed that unregularizing all the biases, γ, and β in BN facilitates generalization. Table 2 summarizes the performance results of C-EfficientNet-i. In this experiment, we applied the data-augmentation procedure listed in (Krizhevsky et al., 2012).

Expansion ratio i	1.0	1.5	2.0	2.5	3.0	3.5	4.0
Vanilla	67.97	69.91	71.74	72.27	72.87	73.26	73.34
+ No bias decay	68.19	70.50	71.95	72.42	73.02	73.42	73.78
NP	0.10 M	0.13 M	0.16M	0.20 M	0.23 M	0.26 M	0.29 M
MO	22 M	32 M	41 M	50 M	60 M	69 M	79 M
Score	0.0048	0.0066	0.0083	0.0102	0.0120	0.0137	0.0155

Table 2: Performance changes according to expansion ratio i in C-EfficientNet-i.

3. Optimization

Here, we introduce the main components of our approach. As evident from Table 2, the performances of all the baseline networks could not achieve the target accuracy, although these networks are fairly resource-efficient. Novel regularization methods, ORN and ALS, improved the accuracy of the baseline network; accordingly, we could prune many parameters in the next step while maintaining the accuracy constraint.

3.1. Orthogonality Regularization

ORN is one of the popular training-refinement strategies that spectrally regularize the weight matrices of a model. For instance, (Bansal et al., 2018) proposed the Spectral Restriced Isometry Property (SRIP) that adds the following regularization term to the main objective function:

$$\frac{\lambda}{|\mathcal{W}|}\sum_{\mathbf{W}\in\mathcal{W}}\sigma(\mathbf{W}\mathbf{W}^\top-\mathbf{I})$$

where $\mathcal{W}$ denotes the set of weight matrices, λ is the regularization coefficient, $\mathbf{I}$ is an identity matrix, $|\mathcal{W}|$ is the cardinality of $\mathcal{W}$, and $\sigma(\mathbf{W}\mathbf{W}^\top-\mathbf{I})$ is the output of the following power method calculated as follows:

$$u\leftarrow(\mathbf{W}\mathbf{W}^\top-\mathbf{I})v,\quad v\leftarrow(\mathbf{W}\mathbf{W}^\top-\mathbf{I})u,\quad \sigma(\mathbf{W}\mathbf{W}^\top-\mathbf{I})\leftarrow\frac{\|v\|}{\|u\|}. \tag{2}$$

where the vector $v\in\mathbf{R}^m$ is randomly initialized via normal distribution.

In (Bansal et al., 2018), however, the gain from ORN was minuscule. We thus attmepted to analyze the effects of SRIP on different categories of layers to find ways to effectively apply SRIP. Accordingly, we categorized layers as a fully-connected layer, a depthwise convolution, and a pointwise convolution. We then applied SRIP to all the different combinations of the aforementioned categories. It was empirically observed that upon applying SRIP only to the 1x1 convolutional kernel (i.e., pointwise convolution) could significantly improve the accuracy compared with existing methods (see Appendix for more details).

Moreover, we applied the orthogonal initialization method (Saxe et al., 2013) that initializes weight matrices to be semi-orthogonal (i.e., $WW^\top=I$) in order to facilitate the training with SRIP. Although SRIP considerably increased the accuracy, the model training with SRIP frequently failed. We believe that SRIP sensitized the training to the initial parameters because of the harsh condition that compelled all the singular values of each matrix toward 1. To resolve this issue, we initialized each weight matrix as semi-orthogonal; i.e., all the singular values were equal to 1 at the beginning. Our method proceeds as follows:

1. Create a matrix $\mathbf{W}$ with the normal distribution whose mean and variance are 0 and 1, respectively.

2. Perform the QR factorization of the matrix and make Q uniform. For the case wherein $\mathbf{W}\in R^{m\times n}(m<n)$, transpose the matrix before computing the QR factorization.

3. Initialize the weight matrix to Q, and set the bias to $b=0$.

3.2. Adaptive Label Smoothing

Label smoothing has been widely used and has achieved excellent results in many networks and applications (Szegedy et al., 2016). To train a CNN for image classification tasks, label smoothing minimizes the cross-entropy between the softmax output of the CNN and a modified target, adding a uniform perturbation to all labels. More precisely, the target is defined as follows:

$$q(k|x) = (1 - \epsilon)\delta_{k,y} + \epsilon u(k)$$

where x denotes an independent of the training examples, k is an label, y is the ground-truth label, and $\delta_{k,y}$ is Dirac delta, which equals 1 when $k = y$ and 0 otherwise. Many use a zero-distribution $u(k) = 0$ for general training and an uniform distribution $u(k) = 1/K$ for label smoothing, where K denotes the number of classes. Label smoothing makes every incorrect class be equally treated (Müller et al., 2019). We believe that the uniform distribution might erase important information.

ALS is our novel regularization method in which we consider the similarities among classes to define the target output. To the best of our knowledge, there is a lack of studies on how to consider the similarities to improve the accuracy. C-EfficientNet can be divided into three parts: a feature extractor, a classifier, and the softmax function. We denote by $f_{feature}$ and $f_{classifier}$ the feature extractor composed of all the convolutional layers and the classifier composed of the last fully-connected layer, respectively. Accordingly, $f_{classifier} = W^{\top} f_{feature}$ with the weight matrix W. When the baseline network is satisfactorily trained, the columns of W represent the correlations among classes. Because the inner products between $f_{feature}$ and the columns of W can accurately decide the output class label, we supposed that the columns of W encode the class information and similarities among classes can be measured using the columns of W. We thus propose a new distribution $u(k)$ as follows:

$$u(k) = \begin{cases} \frac{exp(s(w_k, w_y)/T)}{\sum_{i \neq y} exp(s(w_i, w_y)/T)}, & k \neq y \\ 0, & k = y \end{cases} \tag{3}$$

where w_i denotes the i-th column vector of the weight matrix W. Additionally, $s(x, y) = \frac{x \cdot y}{\|x\| \|y\|}$ denotes the cosine similarity function between two vectors x and y, and T denotes the temperature-scaling factor.

The baseline networks incorporated with ALS shows considerably better results in terms of the accuracy than those of the cases without ALS (see Table 3). For hyperparameters, we stay consistent: lambda $\lambda = 1.0$ for ORN, epsilon $\epsilon = 0.2$, and temperature $T = 0.2$ for ALS.

4. Data Augmentation

We used the SOTA data-augmentation strategies Cutmix and FastAutoaugmentation to further improve the training efficiency and performance.

Fast Autoaugmentation. We applied the FastAutoaugmentation method (Lim et al., 2019a), which automatically searches the augmentation policy with Bayesian optimization in the policy-search phase. For this challenge, we used the Wide-ResNet 28-10 model to search an effective strategy.

Expansion ratio i	1.0	1.5	2.0	2.5	3.0	3.5	4.0
Stacked	68.19	70.50	71.95	72.42	73.02	73.42	73.78
+ ORN	72.21	75.45	77.45	78.12	78.90	79.11	79.33
+ ALS	73.24	75.47	77.61	78.21	79.31	79.42	79.77
NP	0.10 M	0.13 M	0.16M	0.20 M	0.23 M	0.26 M	0.29 M
MO	22 M	32 M	41 M	50 M	60 M	69 M	79 M
Score	0.0048	0.0066	0.0083	0.0102	0.0120	0.0137	0.0155

Table 3: Performance of C-EfficientNet-i for stacking ORN and ALS.

Cutmix. For training, we also used a Cutmix (Yun et al., 2019a) strategy, which enables a network to benefit from both a regional dropout strategy and a Mixup strategy (Zhang et al., 2017). The process is as follows:

$$\tilde{x} = M \odot x_A + (\mathbf{1} - M) \odot x_B$$
$$\tilde{y} = \lambda y_A + (1 - \lambda) y_B,$$

where x and y denote a training image and its label, (x_A, y_A) and (x_B, y_B) are two training samples, M denotes a binary mask that indicates where to drop out and fill in from two images, $\mathbf{1}$ is an all-ones matrix, and $\odot$ indicates element-wise multiplication. The Cutmix strategy creates a binary mask with sampled bounding box with uniform distribution, whose corresponding region is filled with 0. In this challenge, we set the lambda λ to 1.0. Table 4 summarizes the performances with FastAutoaugmentation and Cutmix.

Expansion ratio i	1.0	1.5	2.0	2.5	3.0	3.5	4.0
Stacked	73.24	75.47	77.61	78.21	79.31	79.42	79.77
+ Fast Autoaugment	74.12	75.78	78.13	78.92	80.12	79.64	79.91
+ Cutmix	75.21	77.41	79.81	80.09	**80.92**	81.11	81.33
NP	0.10 M	0.13 M	0.16M	0.20 M	0.23 M	0.26 M	0.29 M
MO	22 M	32 M	41 M	50 M	60 M	69 M	79 M
Score	0.0048	0.0066	0.0083	0.0102	0.0120	0.0137	0.0155

Table 4: Performance of C-EfficientNet-i upon stacking FastAutoaugmentation and Cutmix.

5. Sparse Training

We finally applied the pruning method referred to as the Lottery Ticket Hypothesis (Frankle and Carbin, 2018) to extract a sparse subnetwork over C-EfficientNet-3, as sparser is the network, greater are the benefits obtained from the score. Our modified method proceeds as follows:

1. Train the network with optimization methods and data-augmentation strategies that are provided in the previous sections. Let θ_* be the trained parameters.
2. Repeat the following steps during n rounds until a C-EfficientNet-3 model has $p\%$ sparse parameters.
   (a) Prune $\frac{k}{n} \times p\%$ of the parameters in θ_{k-1}, creating a mask m_k. (in the first round, we set $\theta_0 = \theta_*$.)
   (b) Train the network with the same training recipe in Step 1 and the initial parameters $m_k \odot \theta_{k-1}$. Let θ_k be the trained parameters.

We modified two parts in the existing lottery ticket hypothesis method. Fisrt, we linearly increased the pruning rate for each round. Second, we used the sparse network's parameters learned in the previous round, without re-initializing the remaining parameters after each round.

Iterative pruning via linearly prune-rate. We observed that it was better for our model to iteratively prune using a linear prune-rate than an exponentially prune-rate when performing the iterative sparse training. In previous literature (Frankle and Carbin, 2018), in each round, the $p^{\frac{1}{n}}\%$ of the remaining weights were pruned. However, upon changing the prune-rate to exponential, the number of parameters suddenly reduced, thereby posing an obstacle to generalization.

Removal of re-initialization. We removed the re-initialization stage using the parameter θ_0 to train the network with SRIP. To perform the SRIP training, it requires the condition that the spectral norm of the existing weight matrices should be around 1. In this respect, the re-initialization stage through binary mask of (Frankle and Carbin, 2018) is rather a hindrance to SRIP training. Our experiments show that, without re-intialization, we could sparsify a significant number of the parameters of the existing baseline network.

In this challenge, we split 60% prune-rate by 4 rounds and deployed 15% each.

Prune rate (.%)	0. %	15. %	30. %	45. %	60. %
C-EfficientNet-3	80.92	80.72	80.82	80.75	80.07
NP	0.23 M	0.20 M	0.16 M	0.13 M	0.10M
MO	60 M	51 M	43 M	34 M	26 M
Score	0.0120	0.0103	0.0085	0.0068	**0.0054**

Table 5: Performance of C-EfficientNet-i after pruning.

6. Conclusion

In this paper, we summarize our solution to the MicroNet Challenge 2019. Our solution consists of four stages: (1) architecture design, (2) optimization, (3) data augmentation, and (4) sparse training. We significantly improved the top-1 accuracy of the baseline network with novel regularization methods and then compressed the baseline network. Our results achieved the second place in the CIFAR-100 classification task. In future, we will study how these methods facilitate the generalization of other neural network structures.

Acknowledgments

This work was supported by Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) [No.2018-0-00278, Development of Big Data Edge Analytics SW Technology for Load Balancing and Active Timely Response] and [No.2019-0-00075, Artificial Intelligence Graduate School Program(KAIST)]. We thank Changhwan Kim for insightful discussion about this challenge.

References

Ron Banner, Itay Hubara, Elad Hoffer, and Daniel Soudry. Scalable methods for 8-bit training of neural networks. In *Advances in neural information processing systems*, pages 5145–5153, 2018.

Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can we gain more from orthogonality regularizations in training deep networks? In *Advances in Neural Information Processing Systems*, pages 4261–4271, 2018.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123, 2019.

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

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.

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

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017a. URL http://arxiv.org/abs/1704.04861.

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

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.

Kui Jia, Dacheng Tao, Shenghua Gao, and Xiangmin Xu. Improving training of deep neural networks via singular value bounding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4344–4352, 2017.

Xianyan Jia, Shutao Song, Wei He, Yangzihao Wang, Haidong Rong, Feihu Zhou, Liqiang Xie, Zhenyu Guo, Yuanzhou Yang, Liwei Yu, et al. Highly scalable deep learning training system with mixed-precision: Training imagenet in four minutes. *arXiv preprint arXiv:1807.11205*, 2018.

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.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. *CoRR*, abs/1905.00397, 2019a. URL http://arxiv.org/abs/1905.00397.

Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. In *Advances in Neural Information Processing Systems*, pages 6662–6672, 2019b.

Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983, 2016. URL http://arxiv.org/abs/1608.03983.

Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015.

Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. When does label smoothing help? *CoRR*, abs/1906.02629, 2019. URL http://arxiv.org/abs/1906.02629.

Mete Ozay and Takayuki Okatani. Optimization on submanifolds of convolution kernels in cnns. *arXiv preprint arXiv:1610.07008*, 2016.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *CoRR*, abs/1710.05941, 2017. URL http://arxiv.org/abs/1710.05941.

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

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1139–III–1147. JMLR.org, 2013. URL http://dl.acm.org/citation.cfm?id=3042817.3043064.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016. doi: 10.1109/cvpr.2016.308. URL http://dx.doi.org/10.1109/CVPR.2016.308.

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

Di Xie, Jiang Xiong, and Shiliang Pu. All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6176–6185, 2017.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. *CoRR*, abs/1905.04899, 2019a. URL http://arxiv.org/abs/1905.04899.

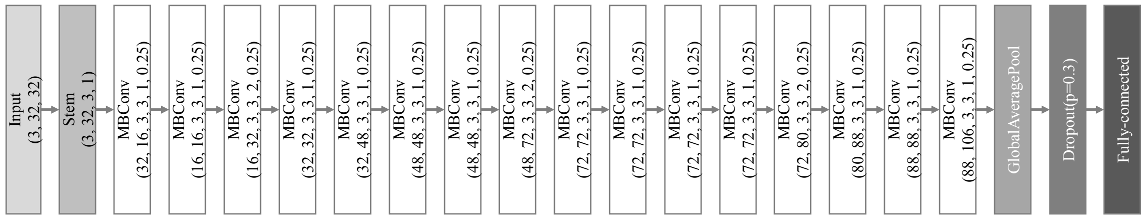
Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6023–6032, 2019b.

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

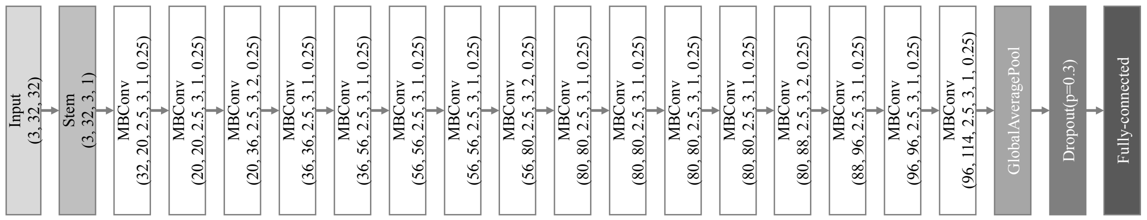
Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Christopher De Sa, and Zhiru Zhang. Improving neural network quantization without retraining using outlier channel splitting. *CoRR*, abs/1901.09504, 2019. URL http://arxiv.org/abs/1901.09504.

Appendix A. Network Overview

We designed two types of architectures that won the second place and third places in the MicroNet Challenge 2019, respectively (see Figure 3).


Input (3, 32, 32)
Stem (3, 32, 3, 1)
MBConv (32, 16, 3, 3, 1, 0.25)
MBConv (16, 16, 3, 3, 1, 0.25)
MBConv (16, 32, 3, 3, 2, 0.25)
MBConv (32, 32, 3, 3, 1, 0.25)
MBConv (32, 48, 3, 3, 1, 0.25)
MBConv (48, 48, 3, 3, 1, 0.25)
MBConv (48, 48, 3, 3, 1, 0.25)
MBConv (48, 72, 3, 3, 2, 0.25)
MBConv (72, 72, 3, 3, 1, 0.25)
MBConv (72, 72, 3, 3, 1, 0.25)
MBConv (72, 72, 3, 3, 1, 0.25)
MBConv (72, 72, 3, 3, 1, 0.25)
MBConv (72, 80, 3, 3, 2, 0.25)
MBConv (80, 88, 3, 3, 1, 0.25)
MBConv (88, 88, 3, 3, 1, 0.25)
MBConv (88, 106, 3, 3, 1, 0.25)
GlobalAveragePool
Dropout(p=0.3)
Fully-connected

(a) Baseline architecture of the 2nd place


Input (3, 32, 32)
Stem (3, 32, 3, 1)
MBConv (32, 20, 2.5, 3, 1, 0.25)
MBConv (20, 20, 2.5, 3, 1, 0.25)
MBConv (20, 36, 2.5, 3, 2, 0.25)
MBConv (36, 36, 2.5, 3, 1, 0.25)
MBConv (36, 56, 2.5, 3, 1, 0.25)
MBConv (56, 56, 2.5, 3, 1, 0.25)
MBConv (56, 56, 2.5, 3, 1, 0.25)
MBConv (56, 80, 2.5, 3, 2, 0.25)
MBConv (80, 80, 2.5, 3, 1, 0.25)
MBConv (80, 80, 2.5, 3, 1, 0.25)
MBConv (80, 80, 2.5, 3, 1, 0.25)
MBConv (80, 80, 2.5, 3, 1, 0.25)
MBConv (80, 88, 2.5, 3, 2, 0.25)
MBConv (88, 96, 2.5, 3, 1, 0.25)
MBConv (96, 96, 2.5, 3, 1, 0.25)
MBConv (96, 114, 2.5, 3, 1, 0.25)
GlobalAveragePool
Dropout(p=0.3)
Fully-connected

(b) Baseline architecture of the 3rd place

Figure 3: (a, b) An overview of the baseline architectures which won the 2nd place and 3rd place, respectively. 'Stem' indicates the convolutional operator used at the beginning of each network. We denote the hyperparameters of each MBConv block as (the number of input channels, the number of output channels, expansion ratio, kernel size, stride, SE ratio).

Appendix B. Related Works

B.1. Orthogonality Regularization

Orthogonal weights have been widely studied in DNNs for vanishing/exploding gradient issue (Saxe et al., 2013; Mishkin and Matas, 2015) while not only reducing the internal covariate shift, but also driving the norm-preserving property of orthogonal matrices. However, this approach can break down easily when in the course of training. Therefore, many recent literatures (Ozay and Okatani, 2016; Xie et al., 2017; Bansal et al., 2018) proposed an orthogonal-constraint technique via penalizing the singular value of each weight matrix during the training procedure. One (Jia et al., 2017) deployed this orthogonality regularization through explicitly bounding the singular values of weight matrices in a narrow band around the value nearby 1.

B.2. Weight Initialization

As a neural network is deepened for improving its generalization, weight initialization becomes more crucial for training highly stacked deep neural networks (DNN). To alleviate this issue, recently, many studies initialized a CNN model via Xavier initialization (Glorot and Bengio, 2010) and Kaiming initialization (He et al., 2015), which are appropriate for ReLU-based networks with residual learning. For newly emerged types of activation functions, such as Swish (Ramachandran et al., 2017) and HardSwish (Howard et al., 2019), no suitable initialization method exists. Furthermore, because SOTA optimization methods, such as orthogonality regularization, change the optimization landscape of a model in a different manner compared with vanilla training, a new initialization technique is required for them.

B.3. Regularization Tricks for Label

DNNs, whose network is over-confident due to the one-hot label vector, faces the challenge and thus many research attempts were made to resolve this issue. One study (Szegedy et al., 2016) mitigated the over-confidence issue by adding uniform noise to one-hot encoded label vector during the training period, and this step is called label smoothing, which slightly increases the performance while it doesn't deal with class-correlation (Müller et al., 2019).

B.4. Pruning

Pruning is one of the most ubiquitous techniques in machine learning to ensure a model has less storage and computational costs. In deep learning, (Han et al., 2015) firstly attempted to apply pruning to a DNN model, which pruned weights by their absolute values. Recently, (Frankle and Carbin, 2018) proposed the lottery ticket hypothesis method that trained a sparse network via posterior information. They empirically observed that performing training using the binary mask extracted from the parameters of the pretrained network over the initialization parameter yielded the same performance as that of the pretrained network. Additionally, they argued that the iterative application of the lottery ticket hypothesis method ensured more network sparsity compared with one-shot pruning under the condition of maintaining the same performance as that of a pretrained network.

B.5. Data Augmentation Strategy

Data augmentation is crucial to improving the performance in a variety of tasks. In image classification, general augmentations include flipping the image horizontally, or cropping the image randomly. With the growth of its importance, several regularization tricks such as random regional dropout method (DeVries and Taylor, 2017; Yun et al., 2019b) and random combination of pairs of image and labels (Zhang et al., 2017; Yun et al., 2019b) have been proposed. However, one attempts to apply AutoML to search the optimal policy (Cubuk et al., 2019) from data despite the significant time-consuming issue. Recently, (Lim et al., 2019b) resolved this computational issue by adopting exploit-explore Bayesian optimization.

B.6. Network Block

Many studies were performed on designing efficient blocks for mobile devices. MobileNetV1 (Howard et al., 2017b) developed the depthwise separable convolutions via factorization to replace the conventional convolution operators. It consists of two parts: depthwise convolution for capturing local information and pointwise convolution for generating features.

MobileNetV2 (Sandler et al., 2018) utilized the linear bottleneck and inverted residual connections for realizing an efficient architecture by considering the topology nature in the latent space of layer-wise feature vectors.

Appendix C. Comparasion Studies for Different Layers

We investigated the places to apply ORN. Here, we categorize the different layers as follows: a fully-connected layer, pointwise convolution, and depthwise convolution. As shown in Table 6 shows, the most significant is attributed to the pointwise convolution.

Method	Conv	Acc. gain(%)
SRIP	fully-connected layer	+ 3.56
SRIP	depthwise convolution	+ 1.33
SRIP	pointwise convolution	**+ 6.05**
SRIP	all layers	+ 3.35

Table 6: **Comparison with orthogonality regularizations.** This experiment was conducted with C-EfficientNet-3 over CIFAR100.