AutoAugment: Learning Augmentation Policies from Data

Ekin D. Cubuk[*†], **Barret Zoph**[†], **Dandelion Mané, Vijay Vasudevan, Quoc V. Le**

Abstract

In this paper, we take a closer look at data augmentation for images, and describe a simple procedure called **AutoAugment** to search for improved data augmentation policies. Our key insight is to create a search space of data augmentation policies, evaluating the quality of a particular policy directly on the dataset of interest. In our implementation, we have designed a search space where a policy consists of many sub-policies, one of which is randomly chosen for each image in each mini-batch. A sub-policy consists of two operations, each operation being an image processing function such as translation, rotation, or shearing, and the probabilities and magnitudes with which the functions are applied. We use a search algorithm to find the best policy such that the neural network yields the highest validation accuracy on a target dataset. Our method achieves state-of-the-art accuracy on CIFAR-10, CIFAR-100, SVHN, and ImageNet (without additional data). On ImageNet, we attain a Top-1 accuracy of 83.54%. On CIFAR-10, we achieve an error rate of 1.48%, which is 0.65% better than the previous state-of-the-art. On reduced data settings, AutoAugment performs comparably to semi-supervised methods without using any unlabeled examples. Finally, policies learned from one dataset can be transferred to work well on other similar datasets. For example, the policy learned on ImageNet allows us to achieve state-of-the-art accuracy on the fine grained visual classification dataset Stanford Cars, without fine-tuning weights pre-trained on additional data.

1 Introduction

Deep neural nets are powerful machine learning systems that tend to work well when trained on massive amounts of data. Data augmentation is a strategy which increases both the amount and diversity of data by randomly "augmenting" it [1–3]; in the image domain, common augmentations include translating the image by a few pixels, or flipping the image horizontally. Intuitively, data augmentation is used to teach a model about invariances in the data domain: classifying an object is often insensitive to horizontal flips or translation. Network architectures can also be used to hardcode invariances: convolutional networks bake in translation invariance [3–6], whereas physics models bake in invariance to translations, rotations, and permutations of atoms [7–12]. However, using data augmentation to demonstrate potential invariances can be easier than hardcoding invariances into the model architecture directly.

Yet a large focus of the machine learning and computer vision community has been to engineer better network architectures (e.g., [13–21]). Less attention has been paid to finding better data augmentation methods that incorporate more invariances. For instance, on ImageNet, the data augmentation approach by [3], introduced in 2012, remains the standard with small changes. Even when augmentation improvements have been found for a particular dataset, they often do not transfer

[*]Work performed as a member of the Google Brain Residency Program.
[†]Equal contribution.
Correspondence to: {cubuk,barretzoph,vrv,qvl}@google.com

Preprint. Work in progress.

to other datasets as effectively. For example, horizontal flipping of images during training is an effective data augmentation method on CIFAR-10, but not on MNIST, due to the different symmetries present in these datasets. The need for automatically learned data-augmentation has been raised recently as an important unsolved problem [22].

In this paper, we aim to automate the process of finding an effective data augmentation policy for a target dataset. In our implementation (§3), each policy expresses several choices and orders of possible augmentation operations, where each operation is an image processing function (e.g., translation, rotation, or color normalization), the probabilities of applying the function, and the magnitudes with which they are applied. We use a search algorithm to find the best choices and orders of these operations such that training a neural network yields the best validation accuracy. In our experiments, we use Reinforcement Learning [23] as the search algorithm, but we believe the results can be further improved if better algorithms are used [21, 24].

利用增强学习搜索空间

Our method achieves state-of-the-art accuracy on datasets such as CIFAR-10, reduced CIFAR-10, CIFAR-100, SVHN, reduced SVHN, and ImageNet (without additional data). On CIFAR-10, we achieve an error rate of 1.48%, which is 0.65% better than the previous state-of-the-art [21]. On SVHN, we improve the state-of-the-art error rate from 1.30% [25] to 1.02%. On reduced datasets, our method achieves performance comparable to semi-supervised methods without using any unlabeled data. On ImageNet, we achieve a Top-1 accuracy of 83.54%. Finally, we show that policies found on one task can generalize well across different models and datasets. For example, the policy found on ImageNet leads to significant improvements on a variety of FGVC datasets. Even on datasets for which fine-tuning weights pre-trained on ImageNet does not help significantly [26], e.g. Stanford Cars [27] and FGVC Aircraft [28], training with the ImageNet policy reduces test set error by 1.16% and 1.76%, respectively. This result suggests that transferring data augmentation policies offers an alternative method for transfer learning.

2 Related Work

Common data augmentation methods for image recognition have been designed manually and the best augmentation policies are dataset-specific. For example, on MNIST, most top-ranked models use elastic distortions, scale, translation, and rotation [2, 29–31]. On natural image datasets, such as CIFAR-10 and ImageNet, random cropping, image mirroring and color shifting / whitening are more common [3]. As these methods are designed manually, they require expert knowledge and time. Our approach of learning data augmentation policies from data in principle can be used for any dataset, not just one.

传统数据增强：随机裁剪，颜色转换，白化，都需要一定的专家只是和时间。

This paper introduces an automated approach to find data augmentation policies from data. Our approach is inspired by recent advances in architecture search, where reinforcement learning and evolution have been used to discover model architectures from data [19, 21, 23, 32–40]. Although these methods have improved upon human-designed architectures, it has not been possible to beat the 2% error-rate barrier on CIFAR-10 using architecture search alone.

本文提出了一种自动的方法从数据中寻找数据增强策略

Previous attempts at learned data augmentations include Smart Augmentation, which proposed a network that automatically generates augmented data by merging two or more samples from the same class [41]. Tran et al. used a Bayesian approach to generate data based on the distribution learned from the training set [42]. DeVries and Taylor used simple transformations in the learned feature space to augment data [43].

Generative adversarial networks have also been used for the purpose of generating additional data (e.g., [44–48]). The key difference between our method and generative models is that our method generates symbolic transformation operations, whereas generative models, such as GANs, generate the augmented data directly. An exception is work by Ratner et al., who used GANs to generate sequences that describe data augmentation strategies [49].

我们的方法是生成符号转换操作，GAN是直接生成增强数据

3 AutoAugment

We formulate the problem of finding the best augmentation policy as a discrete search problem. In our search space, a policy consists of 5 sub-policies, each sub-policy consisting of two image

我们将寻找最优的增强策略的问题转化为离散的搜索问题，在搜索空间中，一个策略由5个子策略组成，每个子策略包含两个图像操作，每个操作联系两个参数：
1. 施加这个操作的可能性
2. 这个操作的幅度。

operations to be applied in sequence, each operation is also associated with two hyperparameters: 1) the probability of applying the operation, and 2) the magnitude of the operation.

Figure 1 shows an example of a policy with 5-sub-policies in our search space. The first sub-policy specifies a sequential application of ShearX followed by Invert. The probability of applying ShearX is 0.9, and when applied, has a magnitude of 7 out of 10. We then apply Invert with probability of 0.8. The Invert operation does not use the magnitude information. We emphasize that these operations are applied in the specified order.

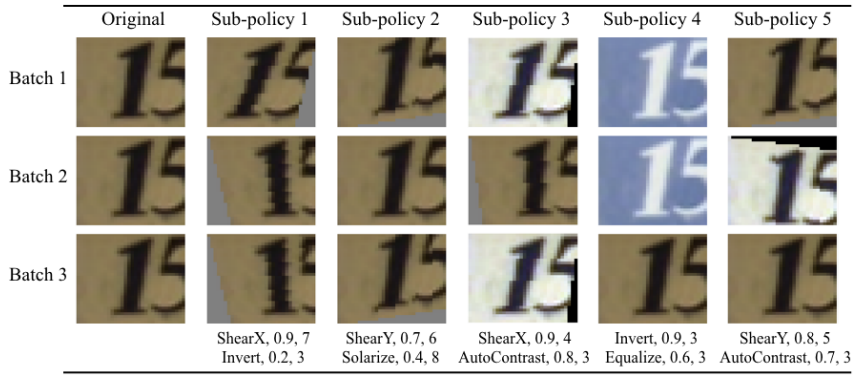

Figure 1: One of the policies found on SVHN, and how it can be used to generate augmented data given an original image used to train a neural network. The policy has 5 sub-policies. For every image in a mini-batch, we choose a sub-policy uniformly at random to generate a transformed image to train the neural network. Each sub-policy consists of 2 operations, each operation is associated with two numerical values: the probability of calling the operation, and the magnitude of the operation. There is a probability of calling an operation, so the operation may not be applied in that mini-batch. However, if applied, it is applied with the fixed magnitude. We highlight the stochasticity in applying the sub-policies by showing how one example image can be transformed differently in different mini-batches, even with the same sub-policy. As explained in the text, on SVHN, geometric transformations are picked more often by AutoAugment. It can be seen why Invert is a commonly selected operation on SVHN, since the numbers in the image are invariant to that transformation.

搜索文件的策略：
1. 16种操作
2. 10种增强的强度
3. 使用该操作的可能性
11中可能性；

在这样的搜索空间查找信息。

Search space of operations: The operations we used in our experiments are from PIL, a popular Python image library.[3] For generality, we considered all functions in PIL that accept an image as input and output an image. We additionally used two other promising augmentation techniques: Cutout [25] and SamplePairing [50]. The operations we searched over are ShearX/Y, TranslateX/Y, Rotate, AutoContrast, Invert, Equalize, Solarize, Posterize, Contrast, Color, Brightness, Sharpness, Cutout [25], Sample Pairing [50][4]. In total, we have 16 operations in our search space. Each operation also comes with a default range of magnitudes, which will be described in more detail in Section 4. We discretize the range of magnitudes into 10 values (uniform spacing) so that we can use a discrete search algorithm to find them. Similarly, we also discretize the probability of applying that operation into 11 values (uniform spacing). Finding each sub-policy becomes a search problem in a space of $(16 \times 10 \times 11)^2$ possibilities. Our goal, however, is to find 5 such sub-policies concurrently in order to increase diversity. The search space with 5 sub-policies then has roughly $(16 \times 10 \times 11)^{10} \approx 2.9 \times 10^{32}$ possibilities.

The 16 operations we used and their default range of values are shown in Table 1 in the Appendix. Notice that there is no explicit "Identity" operation in our search space; this operation is implicit, and can be achieved by calling an operation with probability set to be 0.

算法细节：
1. 搜索算法采用增强学习
2. 算法包含控制器，可以递归网络， 一个训练算法。
3. 每一步，控制器预测通过softmax预测一个决定，

Search algorithm details: The search algorithm that we used in our experiment uses Reinforcement Learning, inspired by [19, 23, 32, 51]. The search algorithm has two components: a controller, which is a recurrent neural network, and the training algorithm, which is the Proximal Policy Optimization algorithm [52]. At each step, the controller predicts a decision produced by a softmax;

[3] `https://pillow.readthedocs.io/en/5.1.x/`

[4] Details about these operations are listed in Table 1 in the Appendix.

预测在下一步作为嵌入输入，控制器由30个softmax预测，为了预测5个子策略，每个子策略包含2个操作，每个操作包含操作类型，强度和概率。

the prediction is then fed into the next step as an embedding. In total the controller has 30 softmax predictions in order to predict 5 sub-policies, each with 2 operations, and each operation requiring an operation type, magnitude and probability.

控制器通过奖励信号训练，为了提升生成一个好的子模型，好的策略是什么，

The controller is trained with a reward signal, which is how good the policy is in improving the generalization of a "child model" (a neural network trained as part of the search process). In our experiments, we set aside a validation set to measure the generalization of a child model. A child model is trained with augmented data generated by applying the 5 sub-policies on the training set (that does not contain the validation set). For each example in the mini-batch, one of the 5 sub-policies is chosen randomly to augment the image. The child model is then evaluated on the validation set to measure the accuracy, which is used as the reward signal to train the recurrent network controller. On each dataset, the controller samples about 15,000 policies. We follow the training procedure and hyperparameters from [19] for training the controller. At the end of the search, we concatenate the sub-policies from the best 5 policies into a single policy (with 25 sub-policies). This final policy with 25 sub-policies is used to train the models for each dataset.

The above search algorithm is one of many possible search algorithms we can use to find the best policies. It might be possible to use a different discrete search algorithm such as genetic programming [21] or even random search [24] to improve the results in this paper.

4 Experiments and Results

In this section, we empirically investigate the performance of our approach on the CIFAR-10 [53], CIFAR-100 [53], SVHN [54], and ImageNet [55] datasets. We describe the experiments we used to find the augmentation policies. We also explain how those policies are used to fully train models and achieve state-of-the-art results on these datasets

4.1 CIFAR-10 and CIFAR-100 Results

Although CIFAR-10 has 50,000 training examples, we perform the search for the best policies on a smaller dataset we call "reduced CIFAR-10", which consists of 4,000 randomly chosen examples, to save time for training child models during the augmentation search process. We find that for a fixed amount of training time, it is more useful to allow child models to train for more epochs rather than train for fewer epochs with more training data. For the child model architecture we use small Wide-ResNet-40-2 (40 layers - widening factor of 2) model [56], and train for 120 epochs. The use of a small Wide-ResNet is for computational efficiency as each child model is trained from scratch to compute the gradient update for the controller. We use a weight decay of 10^{-4}, learning rate of 0.01, and a cosine learning decay with one annealing cycle [57].

The policies found during the search on reduced CIFAR-10 are later used to train final models on CIFAR-10, reduced CIFAR-10, and CIFAR-100. As mentioned above, we concatenate sub-policies from the best 5 policies to form a single policy with 25 sub-policies, which is used for all of AutoAugment experiments on the CIFAR datasets.

The baseline pre-processing follows the convention for state-of-the-art CIFAR-10 models: standardizing the data, using horizontal flips with 50% probability, zero-padding and random crops, and finally Cutout with 16x16 pixels [19, 21, 58, 59]. The AutoAugment policy is applied in addition to the standard baseline pre-processing: on one image, we first apply the baseline augmentation provided by the existing baseline methods, then apply the AutoAugment policy, then apply Cutout. We did not optimize the Cutout region size, and use the suggested value of 16 pixels [25]. Note that since Cutout is an operation in the search space, Cutout may be used twice on the same image: the first time with learned region size, and the second time with fixed region size. In practice, as the probability of the Cutout operation in the first application is small, Cutout is often used once on a given image.

在一张图片上，我们首先使用基本的增强技术，然后使用AutoAugment策略，然后使用Cutout。我们不选择Cutout优选的Cutout区域尺寸，而是使用建议的值：16个像素。

我们使用两次Cutout操作，在搜索空间。在一张图片可能要用两次，
1. 使用学习区域尺寸
2. 使用固定区域尺寸

On CIFAR-10, AutoAugment picks mostly color-based transformations. For example, the most commonly picked transformations on CIFAR-10 are Equalize, AutoContrast, Color, and Brightness (refer to Table 1 in the Appendix for their descriptions). Geometric transformations like ShearX and ShearY are rarely found in good policies. Furthermore, the transformation Invert is almost never applied in a successful policy. The policy found on CIFAR-10 is included in the Appendix. Below, we describe our results on the CIFAR datasets using the policy found on reduced CIFAR-10. All of the reported results are averaged over 5 runs.

CIFAR-10 Results. In Table 1, we show the test set accuracy on different neural network architectures. We implement the Wide-ResNet-28-10 [56], Shake-Shake [58] and ShakeDrop [59] models in TensorFlow[60], and find the weight decay and learning rate hyperparameters that give the best validation set accuracy for regular training with baseline augmentation. Other hyperparameters are the same as reported in the papers introducing the models [56, 58, 59]. We then use the same model and hyperparameters to evaluate the test set accuracy of AutoAugment. For AmoebaNets, we use the same hyperparameters that were used in [21] for both baseline augmentation and AutoAugment. As can be seen from the table, we achieve an error rate of 1.48% with the ShakeDrop [59] model, which is 0.65% better than the state-of-the-art [21]. Notice that this gain is much larger than the previous gains obtained by AmoebaNet-B against ShakeDrop (+0.18%), and by ShakeDrop against Shake-Shake (+0.25%). Ours is also the first method that beats a long-standing 2% error rate barrier.

Model	Baseline	Cutout [25]	AutoAugment
Wide-ResNet-28-10 [56]	3.87	3.08	2.68
Shake-Shake (26 2x32d) [58]	3.55	3.02	2.47
Shake-Shake (26 2x96d) [58]	2.86	2.56	1.99
Shake-Shake (26 2x112d) [58]	2.82	2.57	1.89
AmoebaNet-B (6,128) [21]	2.98	2.13	1.75
PyramidNet+ShakeDrop [59]	2.67	2.31	**1.48**

Table 1: Test set error rates (%) on CIFAR-10. Lower is better. All the results of the baseline models, and baseline models with Cutout are replicated in our experiments and match the previously reported results [25, 56, 58, 59]. One exception is Shake-Shake (26 2x112d), which has more filters than the biggest model in [58] – 112 vs 96, and the results were not previously reported. Note that the best policies are found on reduced CIFAR-10. See text for more details.

CIFAR-100 Results. We also train models on CIFAR-100 with the same AutoAugment policy found on reduced-CIFAR-10; results are shown in Table 2. Again, we achieve the state-of-art result on this dataset, beating the previous record of 12.19% error rate by ShakeDrop regularization [59].

Model	Baseline	Cutout [25]	AutoAugment
Wide-ResNet-28-10 [56]	18.80	18.41	17.09
Shake-Shake (26 2x96d) [58]	17.05	16.00	14.28
PyramidNet+ShakeDrop [59]	13.99	12.19	**10.67**

Table 2: Test set error rates (%) on CIFAR-100. Baseline and Cutout results for Wide-ResNet and ShakeDrop are replicated in our experiments and match the previously reported results [25, 59]. Note that the best policies are found on reduced CIFAR-10.

A comparison against Semi-Supervised Learning. Finally, we apply the same AutoAugment policy to train models on reduced CIFAR-10, and the same 4,000 example training set that we use to find the best policy. This experimental setup is similar to the experimental convention suggested by semi-supervised learning community [61–65]. Note that the reported state-of-art error rates in this area range from 12.36% by consistency regularization (Π-M) [64] to 10.55% by virtual adversarial training and entropy minimization [62], all of which make use of an additional 46,000 unlabeled examples. Our results shown in Table 3 with Wide-ResNet-28-10 [56] are only slightly worse than the state-of-the-art error rates in semi-supervised methods (as summarized by [65]), which use an additional 46,000 unlabeled samples in their training. When we use a different model, Shake-Shake, we outperform all of the reported semi-supervised results without using any of the unlabeled samples.

We note that the improvement in accuracy due to AutoAugment is more significant on the reduced dataset compared to the full dataset. As the size of the training set grows, we expect that the effect of data-augmentation will be reduced. However, in the next sections we show that even for larger datasets like SVHN and ImageNet, AutoAugment can still lead to improvements in generalization accuracy.

4.2 SVHN Results

We experimented with the SVHN dataset [54], which has 73,257 training examples (also called "core training set"), and 531,131 additional training examples. The test set has 26,032 examples. To save

Model	Baseline	Cutout [25]	AutoAugment
Wide-ResNet-28-10 [56]	18.84	16.50	14.13
Shake-Shake (26 2x96d) [58]	17.05	13.40	**10.04**

Table 3: Test set error rates (%) on reduced CIFAR-10, which has 4,000 training instances. Lower error rates are better. Note that this is the same training set that we use to find the best policy. Our result is comparable to state-of-the-art in semi-supervised learning of 10.55% without using additional data.

time during the search, we created a reduced SVHN dataset of 1,000 examples sampled randomly from the core training set. We use AutoAugment to find the best policies. The model architecture and training procedure of the child models are identical to the above experiments with CIFAR-10.

The policies picked on SVHN are different than the transformations picked on CIFAR-10. For example, the most commonly picked transformations on SVHN are Invert, Equalize, ShearX/Y, and Rotate. As mentioned above, the transformation Invert is almost never used on CIFAR-10, yet it is very common in successful SVHN policies. Intuitively, this makes sense since the specific color of numbers is not as important as the relative color of the number and its background. Furthermore, geometric transformations ShearX/Y are two of the most popular transformations on SVHN. This also can be understood by general properties of images in SVHN: house numbers are often naturally sheared and skewed in the dataset, so it is helpful to learn the invariance to such transformations via data augmentation. Five successful sub-policies are visualized on SVHN examples in Figure 1.

After the end of the search, we concatenate the 5 best policies and apply them to train architectures that already perform well on SVHN using standard augmentation policies. For full training, we follow the common procedure mentioned in the Wide-ResNet paper [56] of using the core training set and the extra data. The validation set is constructed by setting aside the last 7325 samples of the training set. We tune the weight decay and learning rate on the validation set performance. Other hyperparameters and training details are identical to the those in the papers introducing the models [56, 58]. One exception is that we trained the Shake-Shake model only for 160 epochs (as opposed to 1,800), due to the large size of the full SVHN dataset. Baseline pre-processing involves standardizing the data and applying Cutout with a region size of 20x20 pixels, following the procedure outlined in [25]. AutoAugment results combine the baseline pre-processing with the policy learned on SVHN. One exception is that we do not use Cutout on reduced SVHN as it lowers the accuracy significantly. The summary of the results in this experiment are shown in Table 4. As can be seen from the table, we achieve state-of-the-art accuracy using both models.

Model	**Reduced SVHN Dataset**			**SVHN Dataset**		
	Baseline	Cutout [25]	AA	Baseline	Cutout	AA
Wide-ResNet-28-10 [56]	13.21	32.5	8.15	1.50	1.30	1.07
Shake-Shake (26 2x96d) [58]	12.32	24.22	**5.92**	1.40	1.20	**1.02**

Table 4: Test set error rates (%). Lower error rates are better. AA refers to AutoAugment. **Reduced SVHN Dataset**: Results on the same training set that we use to find the best policy. Wide-ResNet baseline results on reduced SVHN are replicated in our experiments and roughly match the previously reported results (ours is higher by 0.38%, probably due to hyperparameter tuning) [65]. **SVHN Dataset**: Wide-ResNet results with baseline augmentation [56] and Cutout [25] are replicated in our experiments and match the previously reported results. Shake-Shake results on SVHN have not been reported before. Note that the AutoAugment policy is found on reduced SVHN.

We also test the best policies on reduced SVHN (the same 1,000 example training set where the best policies are found). AutoAugment results on the reduced set are again comparable to the leading semi-supervised methods, which range from 5.42% to 3.86% [62]. (see Table 4). We see again that AutoAugment leads to more significant improvements on the reduced dataset than the full dataset.

4.3 ImageNet Results

Similar to above experiments, we use a reduced subset of the ImageNet training set, with 120 classes (randomly chosen) and 6,000 samples, to search for policies. We train a Wide-ResNet 40-2 using cosine decay for 200 epochs. A weight decay of 10^{-5} was used along with a learning rate of 0.1. The best policies found on ImageNet are similar to those found on CIFAR-10, focusing on color-based

transformations. One difference is that a geometric transformation, Rotate, is commonly used on ImageNet policies. One of the best policies is visualized in Figure 2.

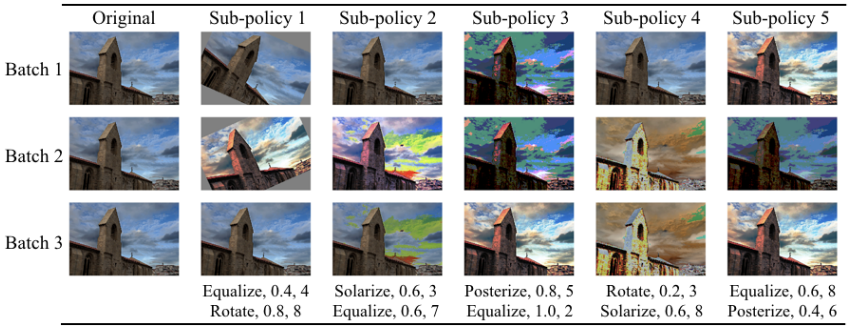

Figure 2: One of the successful policies on ImageNet. As described in the text, most of the policies found on ImageNet used color-based transformations.

Again, we combine the 5 best policies for a total of 25 sub-policies to create the final policy for ImageNet training. We then train on the full ImageNet from scratch with this policy using the ResNet-50 and ResNet-200 models for 270 epochs. We use a batch size of 4096 and a learning rate of 1.6. We decay the learning rate by 10-fold at epochs 90, 180, and 240. For baseline augmentation, we use the standard Inception-style pre-processing which involves scaling pixel values to [-1,1], horizontal flips with 50% probability, and random distortions of colors [14, 66]. For models trained with AutoAugment, we use the baseline pre-processing and the policy learned on ImageNet. We find that removing the random distortions of color does not change the results for AutoAugment.

Model	Baseline	Inception Pre-processing [14]	AutoAugment
ResNet-50 [15]	24.70 / 7.80	23.69 / 6.92	22.37 / 6.18
ResNet-200 [15]	-	21.52 / 5.85	20.00 / 4.99
AmoebaNet-B (6,190) [21]	-	17.80 / 3.97	17.25 / 3.78
AmoebaNet-C (6,228) [21]	-	16.90 / 3.90	**16.46 / 3.52**

Table 5: Validation set Top-1 / Top-5 error rates (%) on ImageNet. Lower is better. ResNet-50 with baseline augmentation result is taken from [15]. AmoebaNet-B,C results with Inception-style preprocessing are replicated in our experiments and match the previously reported result by [21]. There exists a better result of 14.6% Top-1 error rate [67] but their method makes use of a large amount of weakly labeled extra data.

4.4 Fine Grained Visual Classification Datasets

To evaluate the transferability of the policy found on ImageNet, we use the same policy that is learned on ImageNet (and used for the results on Table 5) on five FGVC datasets with image size similar to ImageNet. These datasets are challenging as they have relatively small sets of training examples while having a large number of classes.

Dataset	Train Size	Classes	Inception Pre-processing [14]	AutoAugment
Oxford 102 Flowers [68]	2,040	102	6.69	4.64
Caltech-101 [69]	3,060	102	19.35	13.07
Oxford-IIIT Pets [70]	3,680	37	13.46	11.02
FGVC Aircraft [28]	6,667	100	9.09	7.33
Stanford Cars [27]	8,144	196	6.35	5.19

Table 6: Test set Top-1 error rates (%) on FGVC datasets. Lower rates are better. AutoAugment results use the policy found on ImageNet to train an Inception v4 from scratch.

For all of the datasets listed in Table 6, we train a Inception v4 [16] for 1,000 epochs, using a cosine learning rate decay with one annealing cycle. The learning rate and weight decay are chosen based on the validation set performance. We then combine the training set and the validation set and train again with the chosen hyperparameters. The image size is set to 448x448 pixels. The policies found

on ImageNet improve the generalization accuracy of all of the FGVC datasets significantly. To the best of our knowledge, our result on the Stanford Cars dataset is the lowest error rate achieved on this dataset although we train the network weights from scratch. Previous state-of-the-art fine-tuned pre-trained weights on ImageNet and used deep layer aggregation to attain a 5.9% error rate [71].

5 Discussion

Relation between training steps and number of sub-policies: An important aspect of our work is the stochastic application of sub-policies during training. Every image is only augmented by one of the many sub-policies available in each mini-batch, which itself has further stochasticity since each transformation has a probability of application associated with it. We find that this stochasticity requires a certain number of epochs per sub-policy for AutoAugment to be effective. Since the child models are each trained with 5 sub-policies, they need to be trained for more than 80-100 epochs before the model can fully benefit from all of the sub-policies. This is the reason we choose to train our child models for 120 epochs. Each sub-policy needs to be applied a certain number of times before the model benefits from it. After the policy is learned, the full model is trained for longer (e.g. 1800 epochs for Shake-Shake on CIFAR-10, and 270 epochs for ResNet-50 on ImageNet), which allows us to use more sub-policies.

Importance of Diversity in AutoAugment Policies: Our hypothesis is that as we increase the number of sub-policies, the neural network is trained on the same points with a greater diversity of augmentation, which should increase the generalization accuracy. To test this hypothesis, we investigate the average validation accuracy of fully-trained Wide-ResNet-28-10 models on CIFAR-10 as a function of the number of sub-policies used in training. We randomly select sub-policy sets from a pool of 500 good sub-policies, and train the Wide-ResNet-28-10 model for 200 epochs with each of these sub-policy sets. For each set size, we sampled sub-policies five different times for better statistics. The training details of the model are the same as above for Wide-ResNet-28-10 trained on CIFAR-10. Figure 3 shows the average validation set accuracy as a function of the number of sub-policies used in training, confirming that the validation accuracy improves with more sub-policies up to about 20 sub-policies.

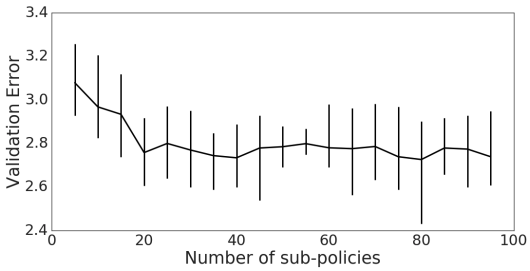

Figure 3: Validation error (averaged over 5 runs) of Wide-ResNet-28-10 trained on CIFAR-10 as a function of number of *randomly selected* sub-policies (out of a pool of 500 good sub-policies) used in training with AutoAugment. Bars represent the range of validation errors for each number.

Transferability across datasets and architectures: It is important to note that the policies described above transfer well to many model architectures and datasets. For example, the policy learned on Wide-ResNet-40-2 and reduced CIFAR-10 leads to the improvements described on all of the other model architectures trained on full CIFAR-10 and CIFAR-100. Similarly, a policy learned on Wide-ResNet-40-2 and reduced ImageNet leads to significant improvements on Inception v4 trained on FGVC datasets that have different data and class distributions. AutoAugment policies are never found to hurt the performance of models even if they are learned on a different dataset, which is not the case for example for Cutout on reduced SVHN (Table 4). We present the best policy on ImageNet and SVHN in the Appendix, which can hopefully help researchers improve their generalization accuracy on relevant image classification tasks.

Despite the observed transferability, we find that policies learned on data distributions closest to the target yield the best performance: when training on SVHN, using the best policy learned on reduced

CIFAR-10 does slightly improve generalization accuracy compared to the baseline augmentation, but not as significantly as applying the SVHN-learned policy.

6 Acknowledgments

We thank Alok Aggarwal, Gabriel Bender, Yanping Huang, Pieter-Jan Kindermans, Simon Kornblith, Augustus Odena, Avital Oliver, and Colin Raffel for helpful discussions.

References

[1] Henry S Baird. Document image defect models. In *Structured Document Image Analysis*, pages 546–556. Springer, 1992.

[2] Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of International Conference on Document Analysis and Recognition*, 2003.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[4] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[5] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[6] Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. What is the best multi-stage architecture for object recognition? In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 2146–2153. IEEE, 2009.

[7] Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Physical review letters*, 98(14):146401, 2007.

[8] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Advances in Neural Information Processing Systems*, pages 992–1002, 2017.

[9] Ekin D Cubuk, Brad D Malone, Berk Onat, Amos Waterland, and Efthimios Kaxiras. Representations in neural network based empirical potentials. *The Journal of chemical physics*, 147(2):024104, 2017.

[10] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

[11] Samuel S Schoenholz, Ekin D Cubuk, Efthimios Kaxiras, and Andrea J Liu. Relationship between local structure and relaxation in out-of-equilibrium glassy systems. *Proceedings of the National Academy of Sciences*, 114(2):263–267, 2017.

[12] Ekin D Cubuk, Samuel S Schoenholz, Efthimios Kaxiras, and Andrea J Liu. Structural properties of defects in glassy liquids. *The Journal of Physical Chemistry B*, 120(26):6139–6146, 2016.

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Advances in Neural Information Processing Systems*, 2015.

[14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[15] 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 (CVPR)*, pages 770–778, 2016.

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

[17] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995, 2017.

[18] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6307–6315. IEEE, 2017.

[19] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[20] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 2017.

[21] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. *arXiv preprint arXiv:1802.01548*, 2018.

[22] Ilya Sutskever, John Schulman, Tim Salimans, and Durk Kingma. Requests For Research 2.0. `https://blog.openai.com/requests-for-research-2`, 2018.

[23] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017.

[24] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*, 2018.

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

[26] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? *arXiv preprint arXiv:1805.08974*, 2018.

[27] Jonathan Krause, Jia Deng, Michael Stark, and Li Fei-Fei. Collecting a large-scale dataset of fine-grained cars. In *Second Workshop on Fine-Grained Visual Categorization*, 2013.

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

[29] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649. IEEE, 2012.

[30] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066, 2013.

[31] Ikuro Sato, Hiroki Nishimura, and Kensuke Yokoi. Apac: Augmented pattern classification with neural networks. *arXiv preprint arXiv:1505.03229*, 2015.

[32] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *International Conference on Learning Representations*, 2017.

[33] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Smash: one-shot model architecture search through hypernetworks. In *International Conference on Learning Representations*, 2017.

[34] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. In *International Conference on Learning Representations*, 2018.

[35] Thomas Elsken, Jan-Hendrik Metzen, and Frank Hutter. Simple and efficient architecture search for convolutional neural networks. *arXiv preprint arXiv:1711.04528*, 2017.

[36] Chenxi Liu, Barret Zoph, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. *arXiv preprint arXiv:1712.00559*, 2017.

[37] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *International Conference on Machine Learning*, 2018.

[38] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc Le, and Alex Kurakin. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, 2017.

[39] Lingxi Xie and Alan Yuille. Genetic CNN. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[40] Ekin D Cubuk, Barret Zoph, Samuel S Schoenholz, and Quoc V Le. Intriguing properties of adversarial examples. *arXiv preprint arXiv:1711.02846*, 2017.

[41] Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. Smart augmentation learning an optimal data augmentation strategy. *IEEE Access*, 5:5858–5869, 2017.

[42] Toan Tran, Trung Pham, Gustavo Carneiro, Lyle Palmer, and Ian Reid. A bayesian data augmentation approach for learning deep models. In *Advances in Neural Information Processing Systems*, pages 2794–2803, 2017.

[43] Terrance DeVries and Graham W Taylor. Dataset augmentation in feature space. *arXiv preprint arXiv:1702.05538*, 2017.

[44] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[45] Seongkyu Mun, Sangwook Park, David K Han, and Hanseok Ko. Generative adversarial network based acoustic scene training set augmentation and selection using svm hyper-plane. In *Detection and Classification of Acoustic Scenes and Events Workshop*, 2017.

[46] Xinyue Zhu, Yifan Liu, Zengchang Qin, and Jiahong Li. Data augmentation in emotion classification using generative adversarial networks. *arXiv preprint arXiv:1711.00648*, 2017.

[47] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

[48] Leon Sixt, Benjamin Wild, and Tim Landgraf. Rendergan: Generating realistic labeled data. *arXiv preprint arXiv:1611.01331*, 2016.

[49] Alexander J Ratner, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. Learning to compose domain-specific transformations for data augmentation. In *Advances in Neural Information Processing Systems*, pages 3239–3249, 2017.

[50] Hiroshi Inoue. Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*, 2018.

[51] Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc V Le. Neural optimizer search with reinforcement learning. In *International Conference on Machine Learning*, 2017.

[52] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[53] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[54] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[55] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[56] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference*, 2016.

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

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

[59] Yoshihiro Yamada, Masakazu Iwamura, and Koichi Kise. Shakedrop regularization. *arXiv preprint arXiv:1802.02375*, 2018.

[60] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, Berkeley, CA, USA, 2016. USENIX Association.

[61] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017.

[62] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. In *International Conference on Learning Representations*, 2016.

[63] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 1163–1171, 2016.

[64] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

[65] Avital Oliver, Augustus Odena, Colin Raffel, Ekin D Cubuk, and Ian J Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. *arXiv preprint arXiv:1804.09170*, 2018.

[66] Andrew G Howard. Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*, 2013.

[67] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. *arXiv preprint arXiv:1805.00932*, 2018.

[68] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Computer Vision, Graphics & Image Processing, 2008. ICVGIP'08. Sixth Indian Conference on*, pages 722–729. IEEE, 2008.

[69] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer vision and Image understanding*, 106(1):59–70, 2007.

[70] Yan Em, Feng Gag, Yihang Lou, Shiqi Wang, Tiejun Huang, and Ling-Yu Duan. Incorporating intra-class variance to fine-grained visual recognition. In *Multimedia and Expo (ICME), 2017 IEEE International Conference on*, pages 1452–1457. IEEE, 2017.

[71] Fisher Yu, Dequan Wang, and Trevor Darrell. Deep layer aggregation. *arXiv preprint arXiv:1707.06484*, 2017.

[72] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. *arXiv preprint arXiv:1708.04896*, 2017.

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

7 Appendix

Operation Name	Description	Range of magnitudes
ShearX(Y)	Shear the image along the horizontal (vertical) axis with rate *magnitude*.	[-0.3,0.3]
TranslateX(Y)	Translate the image in the horizontal (vertical) direction by *magnitude* number of pixels.	[-150,150]
Rotate	Rotate the image *magnitude* degrees.	[-30,30]
AutoContrast	Maximize the the image contrast, by making the darkest pixel black and lightest pixel white.	
Invert	Invert the pixels of the image.	
Equalize	Equalize the image histogram.	
Solarize	Invert all pixels above a threshold value of *magnitude*.	[0,256]
Posterize	Reduce the number of bits for each pixel to *magnitude* bits.	[4,8]
Contrast	Control the contrast of the image. A *magnitude*=0 gives a gray image, whereas *magnitude*=1 gives the original image.	[0.1,1.9]
Color	Adjust the color balance of the image, in a manner similar to the controls on a colour TV set. A *magnitude*=0 gives a black & white image, whereas *magnitude*=1 gives the original image.	[0.1,1.9]
Brightness	Adjust the brightness of the image. A *magnitude*=0 gives a black image, whereas *magnitude*=1 gives the original image.	[0.1,1.9]
Sharpness	Adjust the sharpness of the image. A *magnitude*=0 gives a blurred image, whereas *magnitude*=1 gives the original image.	[0.1,1.9]
Cutout [25, 72]	Set a random square patch of side-length *magnitude* pixels to gray.	[0,60]
Sample Pairing [50, 73]	Linearly add the image with another image (selected at random from the same mini-batch) with weight *magnitude*, without changing the label.	[0, 0.4]

Table 7: List of all image transformations that the controller could choose from during the search. Additionally, the values of magnitude that can be predicted by the controller during the search for each operation at shown in the third column (for image size 331x331). Some transformations do not use the magnitude information (e.g. Invert and Equalize).

	Operation 1	Operation 2
Sub-policy 0	(Invert,0.1,7)	(Contrast,0.2,6)
Sub-policy 1	(Rotate,0.7,2)	(TranslateX,0.3,9)
Sub-policy 2	(Sharpness,0.8,1)	(Sharpness,0.9,3)
Sub-policy 3	(ShearY,0.5,8)	(TranslateY,0.7,9)
Sub-policy 4	(AutoContrast,0.5,8)	(Equalize,0.9,2)
Sub-policy 5	(ShearY,0.2,7)	(Posterize,0.3,7)
Sub-policy 6	(Color,0.4,3)	(Brightness,0.6,7)
Sub-policy 7	(Sharpness,0.3,9)	(Brightness,0.7,9)
Sub-policy 8	(Equalize,0.6,5)	(Equalize,0.5,1)
Sub-policy 9	(Contrast,0.6,7)	(Sharpness,0.6,5)
Sub-policy 10	(Color,0.7,7)	(TranslateX,0.5,8)
Sub-policy 11	(Equalize,0.3,7)	(AutoContrast,0.4,8)
Sub-policy 12	(TranslateY,0.4,3)	(Sharpness,0.2,6)
Sub-policy 13	(Brightness,0.9,6)	(Color,0.2,8)
Sub-policy 14	(Solarize,0.5,2)	(Invert,0.0,3)
Sub-policy 15	(Equalize,0.2,0)	(AutoContrast,0.6,0)
Sub-policy 16	(Equalize,0.2,8)	(Equalize,0.6,4)
Sub-policy 17	(Color,0.9,9)	(Equalize,0.6,6)
Sub-policy 18	(AutoContrast,0.8,4)	(Solarize,0.2,8)
Sub-policy 19	(Brightness,0.1,3)	(Color,0.7,0)
Sub-policy 20	(Solarize,0.4,5)	(AutoContrast,0.9,3)
Sub-policy 21	(TranslateY,0.9,9)	(TranslateY,0.7,9)
Sub-policy 22	(AutoContrast,0.9,2)	(Solarize,0.8,3)
Sub-policy 23	(Equalize,0.8,8)	(Invert,0.1,3)
Sub-policy 24	(TranslateY,0.7,9)	(AutoContrast,0.9,1)

Table 8: AutoAugment policy found on reduced CIFAR-10.

	Operation 1	Operation 2
Sub-policy 0	(ShearX,0.9,4)	(Invert,0.2,3)
Sub-policy 1	(ShearY,0.9,8)	(Invert,0.7,5)
Sub-policy 2	(Equalize,0.6,5)	(Solarize,0.6,6)
Sub-policy 3	(Invert,0.9,3)	(Equalize,0.6,3)
Sub-policy 4	(Equalize,0.6,1)	(Rotate,0.9,3)
Sub-policy 5	(ShearX,0.9,4)	(AutoContrast,0.8,3)
Sub-policy 6	(ShearY,0.9,8)	(Invert,0.4,5)
Sub-policy 7	(ShearY,0.9,5)	(Solarize,0.2,6)
Sub-policy 8	(Invert,0.9,6)	(AutoContrast,0.8,1)
Sub-policy 9	(Equalize,0.6,3)	(Rotate,0.9,3)
Sub-policy 10	(ShearX,0.9,4)	(Solarize,0.3,3)
Sub-policy 11	(ShearY,0.8,8)	(Invert,0.7,4)
Sub-policy 12	(Equalize,0.9,5)	(TranslateY,0.6,6)
Sub-policy 13	(Invert,0.9,4)	(Equalize,0.6,7)
Sub-policy 14	(Contrast,0.3,3)	(Rotate,0.8,4)
Sub-policy 15	(Invert,0.8,5)	(TranslateY,0.0,2)
Sub-policy 16	(ShearY,0.7,6)	(Solarize,0.4,8)
Sub-policy 17	(Invert,0.6,4)	(Rotate,0.8,4)
Sub-policy 18	(ShearY,0.3,7)	(TranslateX,0.9,3)
Sub-policy 19	(ShearX,0.1,6)	(Invert,0.6,5)
Sub-policy 20	(Solarize,0.7,2)	(TranslateY,0.6,7)
Sub-policy 21	(ShearY,0.8,4)	(Invert,0.8,8)
Sub-policy 22	(ShearX,0.7,9)	(TranslateY,0.8,3)
Sub-policy 23	(ShearY,0.8,5)	(AutoContrast,0.7,3)
Sub-policy 24	(ShearX,0.7,2)	(Invert,0.1,5)

Table 9: AutoAugment policy found on reduced SVHN.

	Operation 1	Operation 2
Sub-policy 0	(Posterize,0.4,8)	(Rotate,0.6,9)
Sub-policy 1	(Solarize,0.6,5)	(AutoContrast,0.6,5)
Sub-policy 2	(Equalize,0.8,8)	(Equalize,0.6,3)
Sub-policy 3	(Posterize,0.6,7)	(Posterize,0.6,6)
Sub-policy 4	(Equalize,0.4,7)	(Solarize,0.2,4)
Sub-policy 5	(Equalize,0.4,4)	(Rotate,0.8,8)
Sub-policy 6	(Solarize,0.6,3)	(Equalize,0.6,7)
Sub-policy 7	(Posterize,0.8,5)	(Equalize,1.0,2)
Sub-policy 8	(Rotate,0.2,3)	(Solarize,0.6,8)
Sub-policy 9	(Equalize,0.6,8)	(Posterize,0.4,6)
Sub-policy 10	(Rotate,0.8,8)	(Color,0.4,0)
Sub-policy 11	(Rotate,0.4,9)	(Equalize,0.6,2)
Sub-policy 12	(Equalize,0.0,7)	(Equalize,0.8,8)
Sub-policy 13	(Invert,0.6,4)	(Equalize,1.0,8)
Sub-policy 14	(Color,0.6,4)	(Contrast,1.0,8)
Sub-policy 15	(Rotate,0.8,8)	(Color,1.0,2)
Sub-policy 16	(Color,0.8,8)	(Solarize,0.8,7)
Sub-policy 17	(Sharpness,0.4,7)	(Invert,0.6,8)
Sub-policy 18	(ShearX,0.6,5)	(Equalize,1.0,9)
Sub-policy 19	(Color,0.4,0)	(Equalize,0.6,3)
Sub-policy 20	(Equalize,0.4,7)	(Solarize,0.2,4)
Sub-policy 21	(Solarize,0.6,5)	(AutoContrast,0.6,5)
Sub-policy 22	(Invert,0.6,4)	(Equalize,1.0,8)
Sub-policy 23	(Color,0.6,4)	(Contrast,1.0,8)

Table 10: AutoAugment policy found on reduced ImageNet.