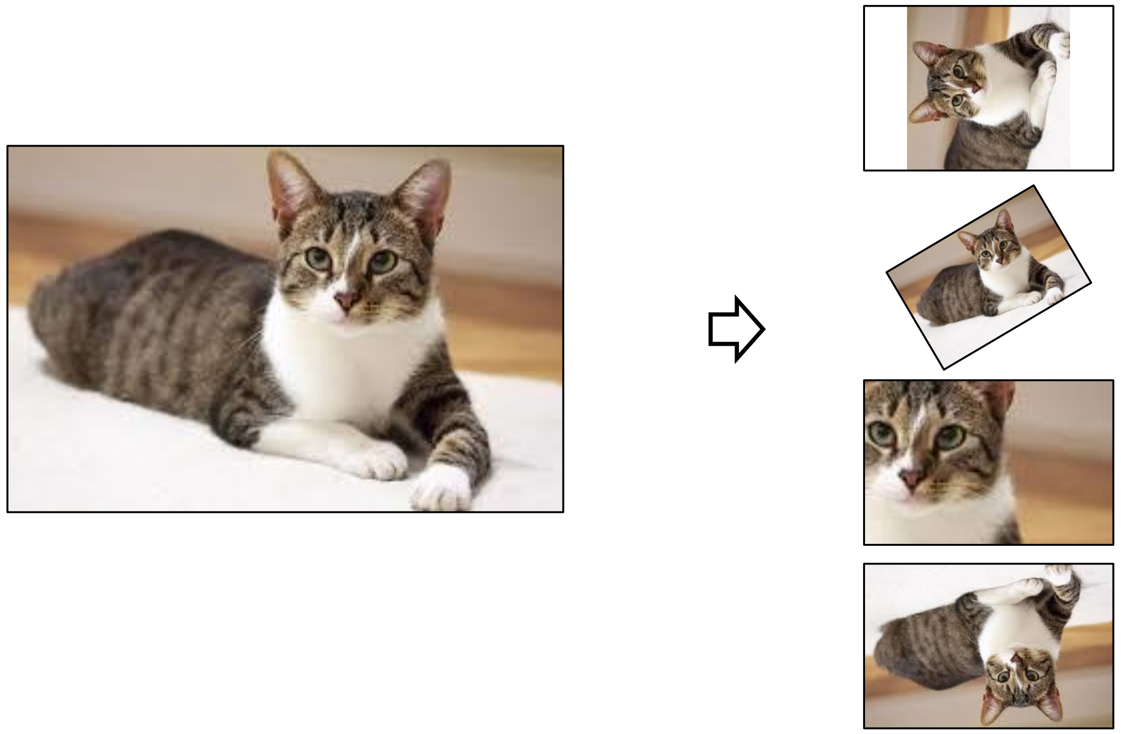
Chapter 08. 좋은 딥러닝 구조를 찾아내는 딥러닝 (Neural Architecture Search)

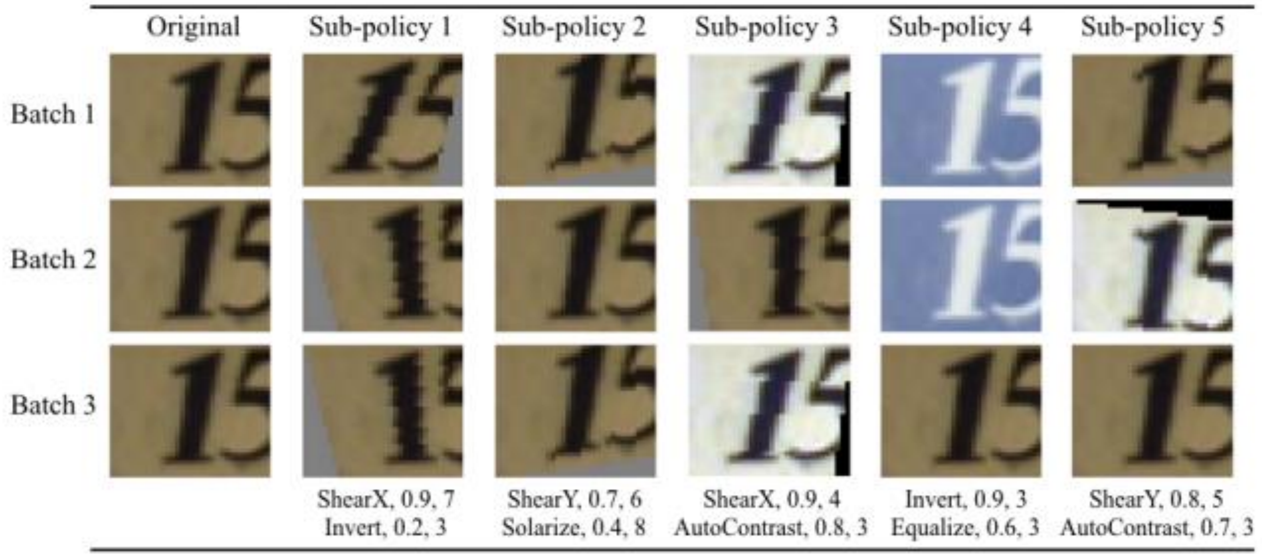
Auto Augmentation

Data Augmentation



일반적으로 많이 사용되는 Data Augmentation 기법들 (Flip, Rotate, Crop)

AutoAugment



NAS의 영향을 받아서 Google Brain에서 발표한 Automatic Data Augmentation 기법

Overall Framework

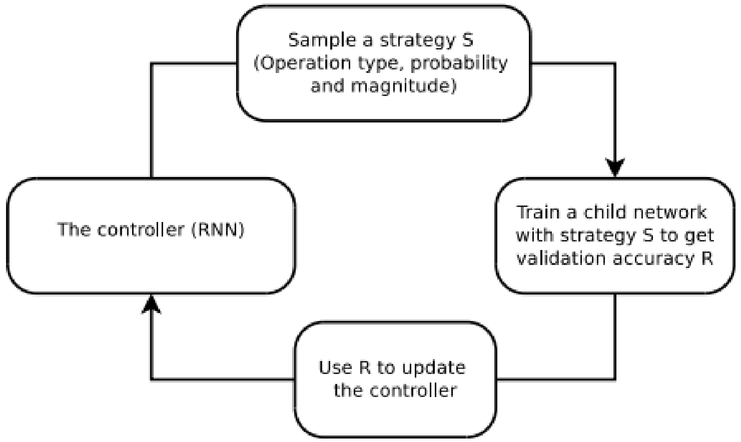
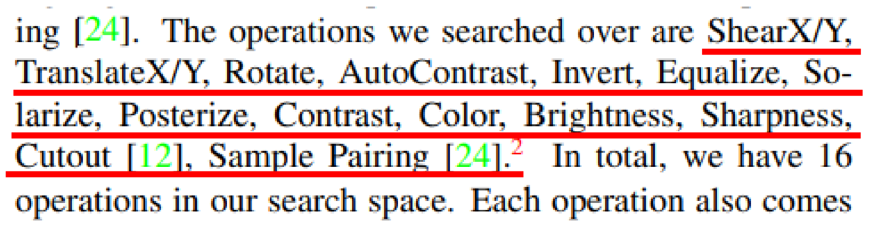


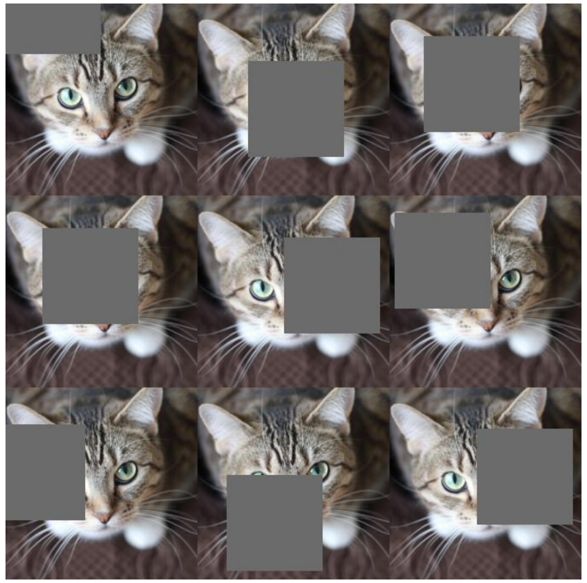
Figure 1. Overview of our framework of using a search method (e.g., Reinforcement Learning) to search for better data augmentation policies. A controller RNN predicts an augmentation policy from the search space. A child network with a fixed architecture is trained to convergence achieving accuracy R. The reward R will be used with the policy gradient method to update the controller so that it can generate better policies over time.

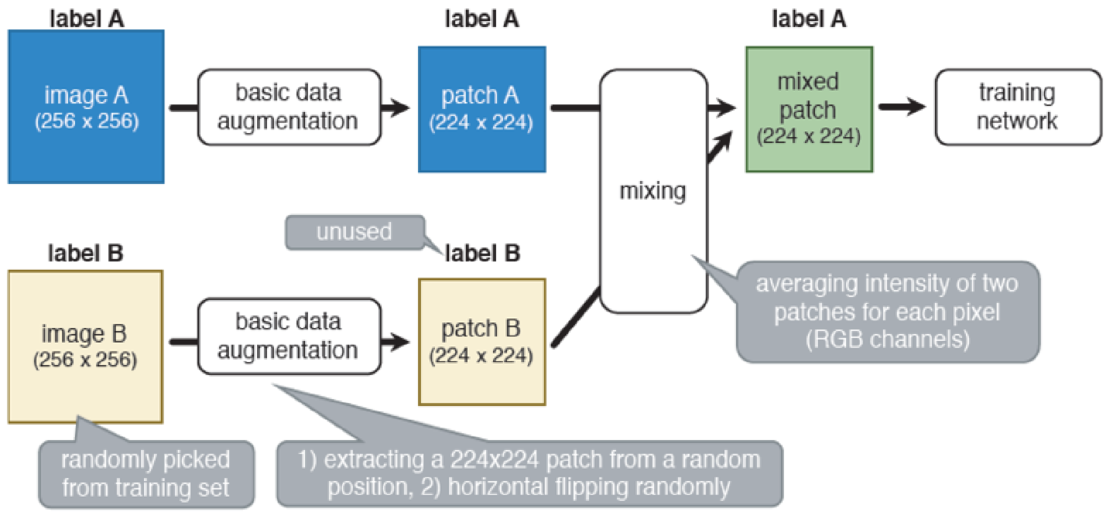
RNN으로 구성된 Controller로 Strategy S를 추정하여 샘플링한다.

Strategy S를 이용해 Child Network를 학습한 결과로 정확도 R을 구하고, Controller를 update한다.

Search Space


ing [24]. The operations we searched over are ShearX/Y, TranslateX/Y, Rotate, AutoContrast, Invert, Equalize, Solarize, Posterize, Contrast, Color, Brightness, Sharpness, Cutout [12], Sample Pairing [24].[2] In total, we have 16 operations in our search space. Each operation also comes

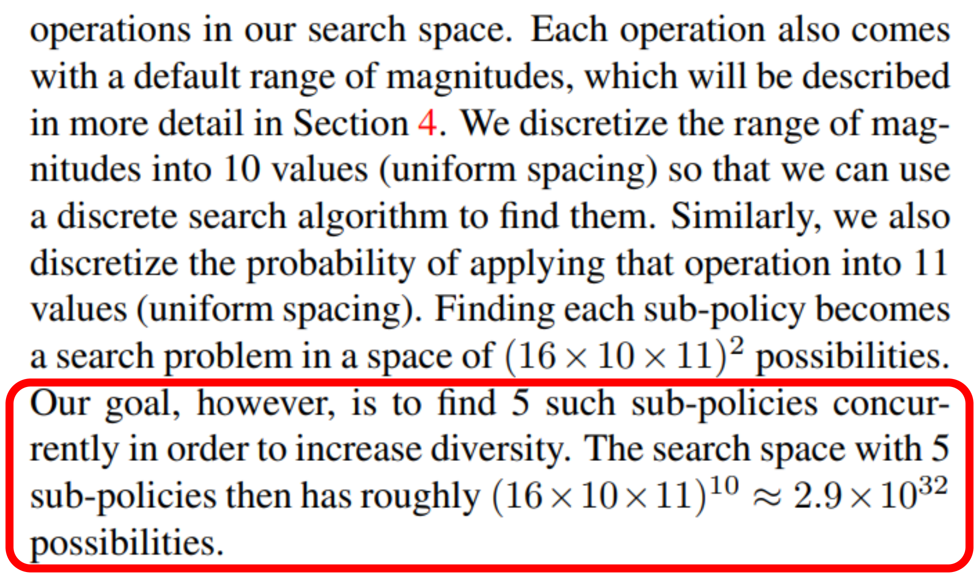

Cutout



Sample Pairing

Augmentation 방법으로는 PIL(Pillow Image Library)에서 제공하는 Function을 사용
추가로 Cutout, Sample Pairing 방법을 적용해 총 16개의 방법 채용

Search Space



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.

Reinforcement Learning 방법인 PPO를 이용해 최적화를 하였다.

Results on ImageNet

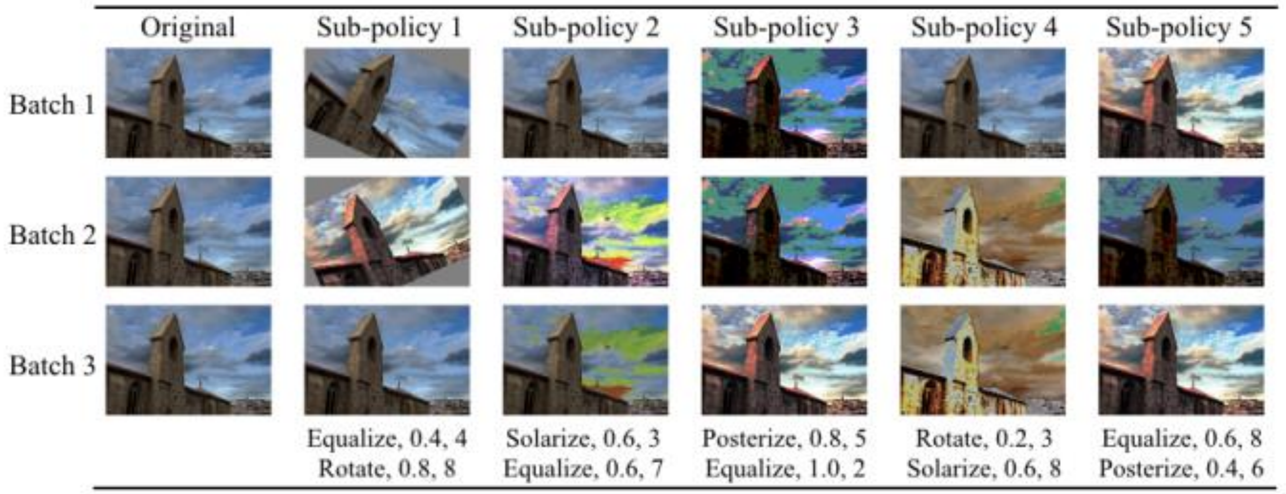
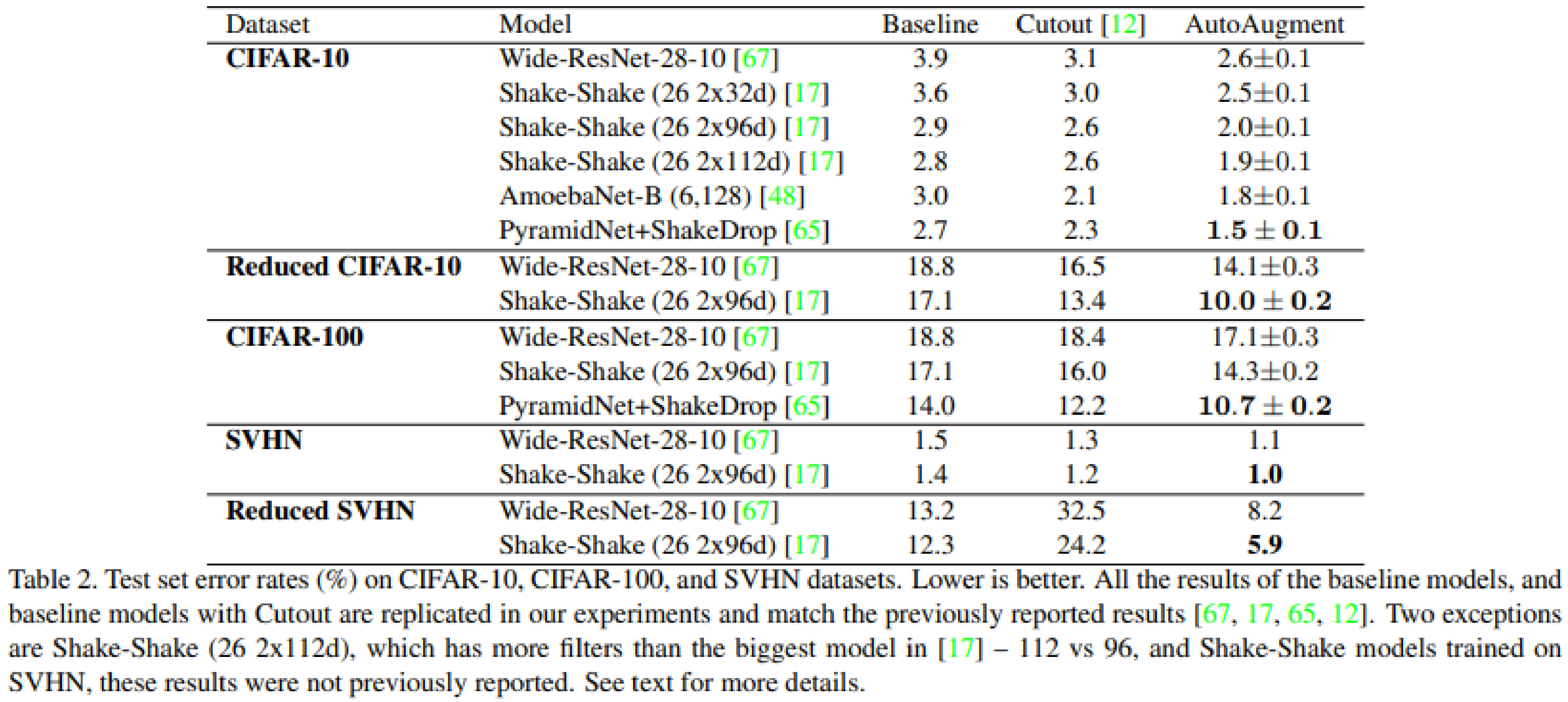


Figure 3. One of the successful policies on ImageNet. As described in the text, most of the policies found on ImageNet used color-based transformations.

Results



Dataset	Model	Baseline	Cutout [12]	AutoAugment
CIFAR-10	Wide-ResNet-28-10 [67]	3.9	3.1	2.6±0.1
	Shake-Shake (26 2x32d) [17]	3.6	3.0	2.5±0.1
	Shake-Shake (26 2x96d) [17]	2.9	2.6	2.0±0.1
	Shake-Shake (26 2x112d) [17]	2.8	2.6	1.9±0.1
	AmoebaNet-B (6,128) [48]	3.0	2.1	1.8±0.1
	PyramidNet+ShakeDrop [65]	2.7	2.3	**1.5 ± 0.1**
Reduced CIFAR-10	Wide-ResNet-28-10 [67]	18.8	16.5	14.1±0.3
	Shake-Shake (26 2x96d) [17]	17.1	13.4	**10.0 ± 0.2**
CIFAR-100	Wide-ResNet-28-10 [67]	18.8	18.4	17.1±0.3
	Shake-Shake (26 2x96d) [17]	17.1	16.0	14.3±0.2
	PyramidNet+ShakeDrop [65]	14.0	12.2	**10.7 ± 0.2**
SVHN	Wide-ResNet-28-10 [67]	1.5	1.3	1.1
	Shake-Shake (26 2x96d) [17]	1.4	1.2	**1.0**
Reduced SVHN	Wide-ResNet-28-10 [67]	13.2	32.5	8.2
	Shake-Shake (26 2x96d) [17]	12.3	24.2	**5.9**

Table 2. Test set error rates (%) on CIFAR-10, CIFAR-100, and SVHN datasets. 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 [67, 17, 65, 12]. Two exceptions are Shake-Shake (26 2x112d), which has more filters than the biggest model in [17] – 112 vs 96, and Shake-Shake models trained on SVHN, these results were not previously reported. See text for more details.

SOTA라고 볼 수 있는 Cutout 기법 대비 좋은 성능을 보인다.

특히 Dataset의 크기가 작은 Reduced SVHN에서 성능 개선이 두드러진다.