AutoAugment: Learning Augmentation Strategies from Data

Abstract

automatically search for improved data augmentation policies

a search space where a policy consists of many subpolicies, one of which is randomly chosen for each image in each mini-batch

Introduction

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

Even when augmentation improvements have been found for a particular dataset, they often do not transfer to other datasets as effectively.

automate the process of finding an effective data augmentation policy for a target dataset

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

Reinforcement Learning [71] as the search algorithm, but we believe the results can be further improved if better algorithms are used [48, 39].

AutoAugment achieves excellent improvements in two use cases:

1) AutoAugment can be applied directly on the dataset of interest to find the best augmentation policy (AutoAugment-direct)

2) learned policies can be transferred to new datasets (AutoAugment-transfer)

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 [38], training with the ImageNet policy reduces test set error by 1.2% and 1.8%, respectively

Dataset	GPU hours	Best published results	Our results
CIFAR-10	5000	2.1	1.5
CIFAR-100	0	12.2	10.7
SVHN	1000	1.3	1.0
Stanford Cars	0	5.9	5.2
ImageNet	15000	3.9	3.5

Related Work

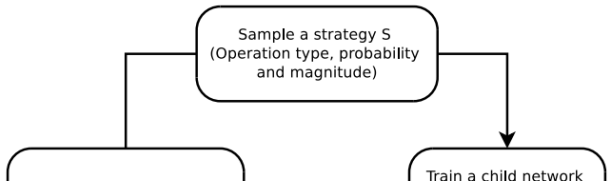
Common data augmentation methods

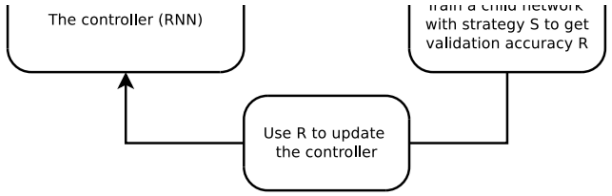
designed manually and the best augmentation strategies are dataset-specific

these methods are designed manually, they require expert knowledge and time

our method generates symbolic transformation operations, whereas generative models, such as GANs, generate the augmented data directly

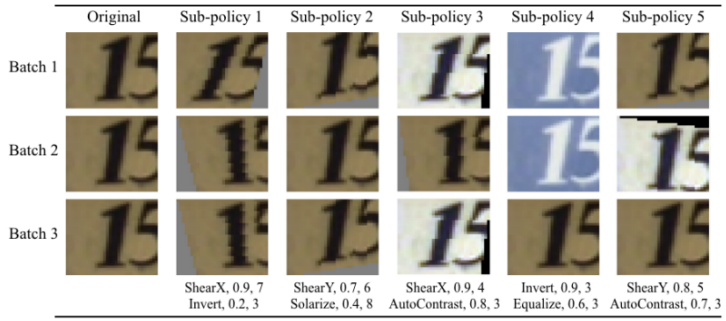
AutoAugment: Searching for best Augmentation policies Directly on the Dataset of Interest





RNN - probability of using the operation in each batch, and the magnitude
policy S will be used to train a neural network with a fixed architecture, whose validation accuracy R will be sent back to update the controller

Search space details
5 sub-policies with each sub-policy consisting of two image operations to be applied in sequence.
two hyperparameters
1) the probability of applying the operation
2) the magnitude of the operation



two other promising augmentation techniques: Cutout [12] and SamplePairing [24]
operations we searched over are ShearX/Y, TranslateX/Y, Rotate, AutoContrast, Invert, Equalize, Solarize, Posterize, Contrast, Color, Brightness, Sharpness, Cutout [12], Sample Pairing

Finding each sub-policy becomes a search problem in a space of (16×10×11)^2 possibilities.
The search space with 5 sub-policies then has roughly (16×10×11)^10 ≈ 2.9×10^32 possibilities

no explicit “Identity” operation

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 [12, 69]	Set a random square patch of side-length *magnitude* pixels to gray.	[0,60]
Sample Pairing [24, 68]	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]

Search algorithm details

Reinforcement Learning

The search algorithm has two components: a controller, which is a recurrent neural network, // the training algorithm, which is the Proximal Policy Optimization algorithm

30 softmax predictions in order to predict 5 sub-policies, each with 2 operations, and each operation requiring an operation type, magnitude and probability

The training of controller RNN

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

Architecture of controller RNN and training hyperparameters

the controller RNN is a one-layer LSTM [21] with 100 hidden units at each layer and 2 × 5B softmax predictions for the two convolutional cells

joint probability is used to compute the gradient for the controller RNN.

might be possible to use a different discrete search algorithm such as genetic programming [48] or even random search [6] to improve the results in this paper.

Experiments and Results

Summary of Experiments

AutoAugment-direct and AutoAugmenttransfer

benchmark AutoAugment with direct search for best augmentation policies on highly competitive datasets

Our results show that a direct application of AutoAugment improves significantly the baseline models and produces state-of-the-art accuracies on these challenging datasets

transferability of augmentation policies between datasets

CIFAR-10, CIFAR-100, SVHN Results

reduced CIFAR-10

4,000 randomly chosen examples, to save time for training child models during the augmentation search process

We find that the resulting policies do not seem to be sensitive to this number

it is more useful to allow child models to train for more epochs rather than train for fewer epochs with more training data

existing baseline methods, then apply the AutoAugment policy, then apply Cutout.

On CIFAR-10, AutoAugment picks mostly color-based transformations

CIFAR-10 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**

As the size of the training set grows, we expect that the effect of data-augmentation will be reduced

SVHN Results

reduced SVHN dataset of 1,000 examples sampled randomly from the core training set

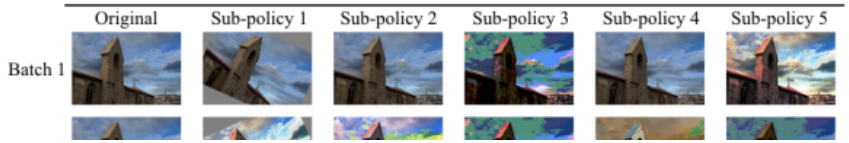
the most commonly picked transformations on SVHN are Invert, Equalize, ShearX/Y, and Rotate

we do not use Cutout on reduced SVHN as it lowers the accuracy significantly.

ImageNet Results

The best policies found on ImageNet are similar to those found on CIFAR-10

One difference is that a geometric transformation, Rotate, is commonly used on ImageNet policies



Model	Inception Pre-processing [59]	AutoAugment ours
ResNet-50	76.3 / 93.1	77.6 / 93.8
ResNet-200	78.5 / 94.2	80.0 / 95.0
AmoebaNet-B (6,190)	82.2 / 96.0	82.8 / 96.2
AmoebaNet-C (6,228)	83.1 / 96.1	**83.5 / 96.5**

best augmentation policy was discovered on 5,000 images
results to be even better when more compute is available

The Transferability of Learned Augmentation policies to Other Datasets

AutoAugment can be resource-intensive

Dataset	Train Size	Classes	Baseline	AutoAugment-transfer
Oxford 102 Flowers [43]	2,040	102	6.7	**4.6**
Caltech-101 [15]	3,060	102	19.4	**13.1**
Oxford-IIIT Pets [14]	3,680	37	13.5	**11.0**
FGVC Aircraft [38]	6,667	100	9.1	**7.3**
Stanford Cars [27]	8,144	196	6.4	**5.2**

Discussion

AutoAugment vs. other automated data augmentation methods

our method tries to optimize classification accuracy directly whereas their method just tries to make sure the augmented images are similar to the current training images

Method	Baseline	Augmented	Improvement Δ
LSTM [47]	7.7	6.0	1.6
MF [47]	7.7	5.6	2.1
AutoAugment (ResNet-32)	7.7	4.5	**3.2**
AutoAugment	6.6	3.6	**3.0**

(ResNet-56)			

Relation between training steps and number of subpolicies

Every image is only augmented by one of the many sub-policies available in each mini-batch

certain number of epochs per sub-policy for AutoAugment to be effective

We find that this stochasticity requires a certain number of epochs per sub-policy for AutoAugment to be effective

Transferability across datasets and architectures

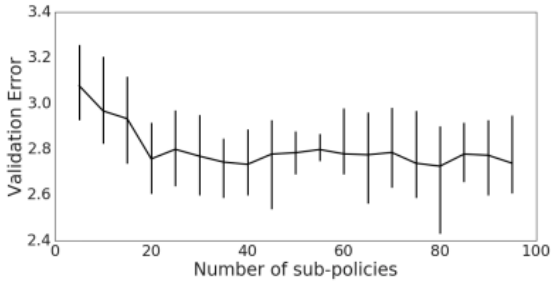
AutoAugment policies are never found to hurt the performance of models

we find that policies learned on data distributions closest to the target yield the best performance

Ablation experiments

Changing the number of sub-policies:

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



Randomizing the probabilities and magnitudes in the augmentation policy

Performance of random policies

slightly worse than randomizing only the probabilities and magnitudes

even AutoAugment with randomly sampled policy leads to appreciable improvements

randomly sampled from our search space can lead to improvements on CIFAR-10 over the baseline augmentation policy

but less than those shown by the AutoAugment policy

probability and magnitude information learned within the AutoAugment policy seem to be important