AutoAugment: Learning Augmentation Policies from Data

Authors: Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, Quoc V. Le

Maryam Tayyab, Gorisha Agarwal, Wen Ruochen

Outline

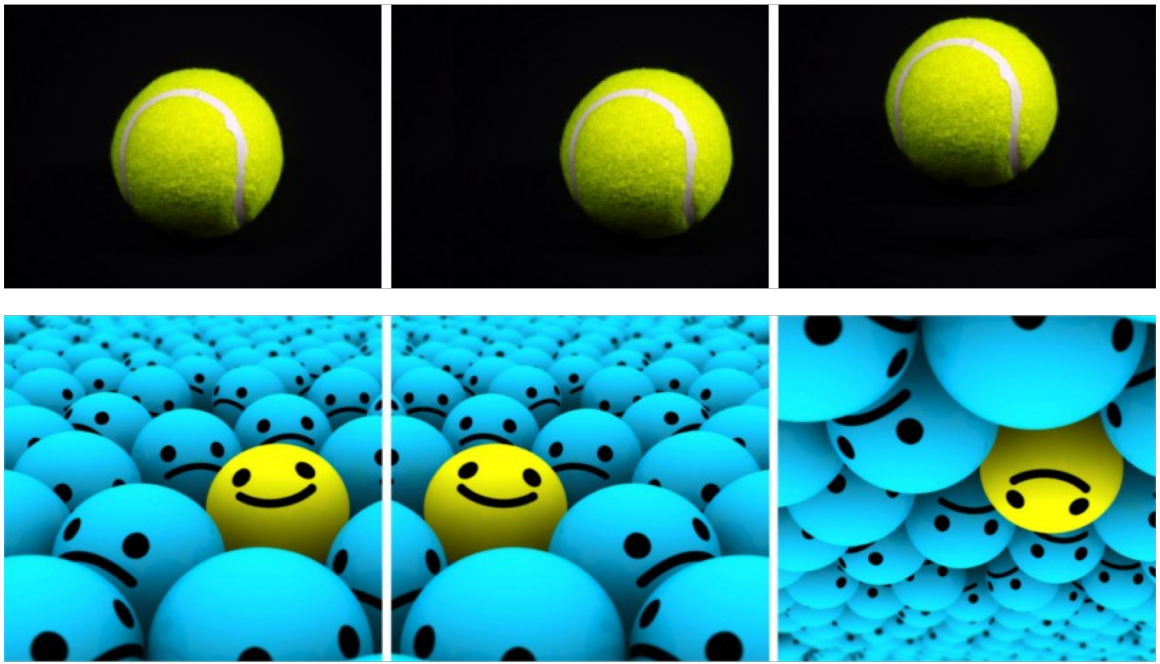
- Introduction
- Motivation
- Goal
- Related Work
- AutoAugment -- detailed overview
- Experiments and results
- Discussion
- Strengths
- Weaknesses
- Extension

Introduction

- What is data augmentation?

Introduction

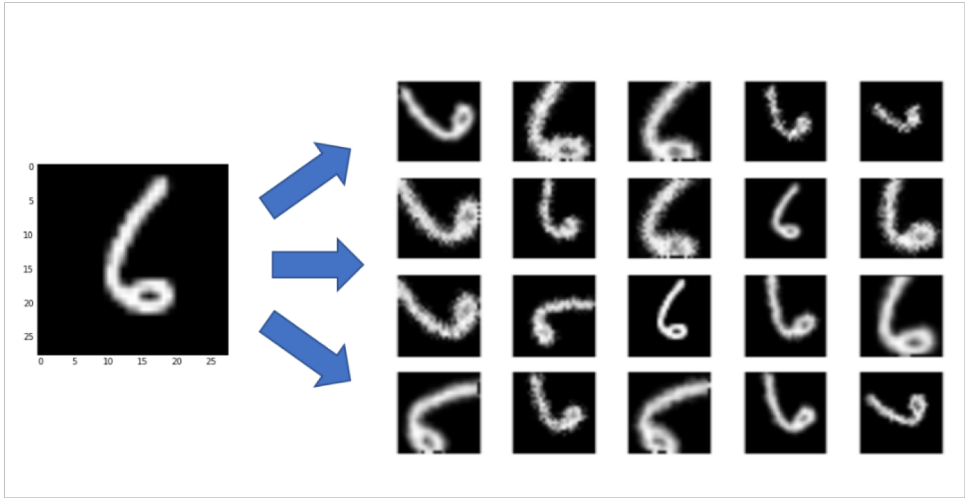
- What is data augmentation?



https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced

Introduction

- What is data augmentation?



https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced

Introduction

- What is data augmentation?
- Purpose of data augmentation

https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced

Introduction

- What is data augmentation?
- Purpose of data augmentation
- Network Architectures can have Invariance baked in them
  - CNN
  - Physics Models

Motivation

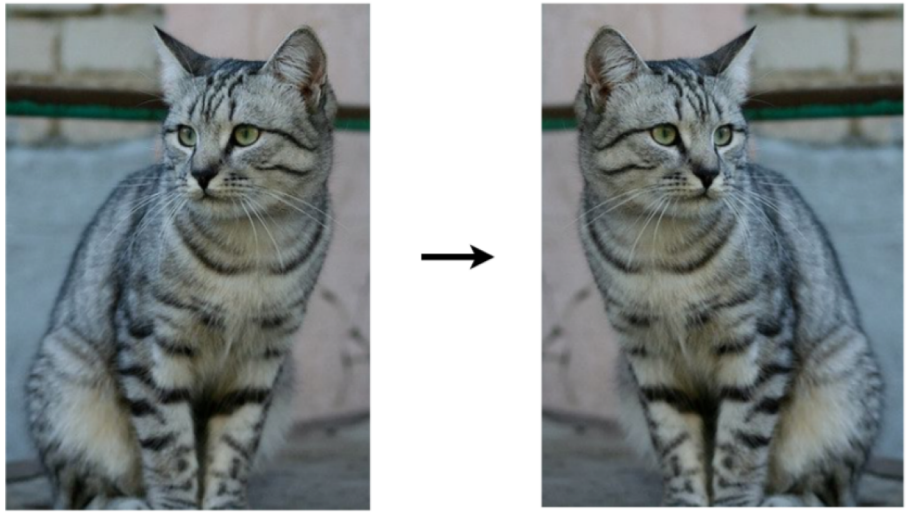
- Data augmentation is very hand engineered

Motivation

- Data augmentation is very hand engineered
- Augmentation techniques by Krizhevsky et al, (2012)

Data Augmentation:

Horizontal flips Random crops & scales Color Jitter



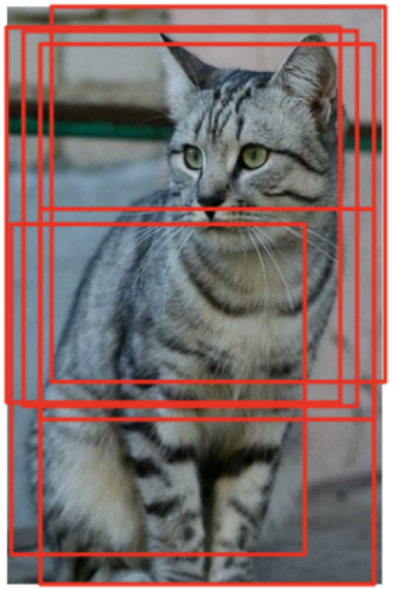
Adapted from Leonid Sigal **532S-W2 (2018)

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Data Augmentation

Horizontal flips **Random crops & scales** Color Jitter

Training: sample random crops and scales



Testing: average a fix set of crops

- Predictions are averaged

Adapted from Leonid Sigal **532S-W2 (2018)

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

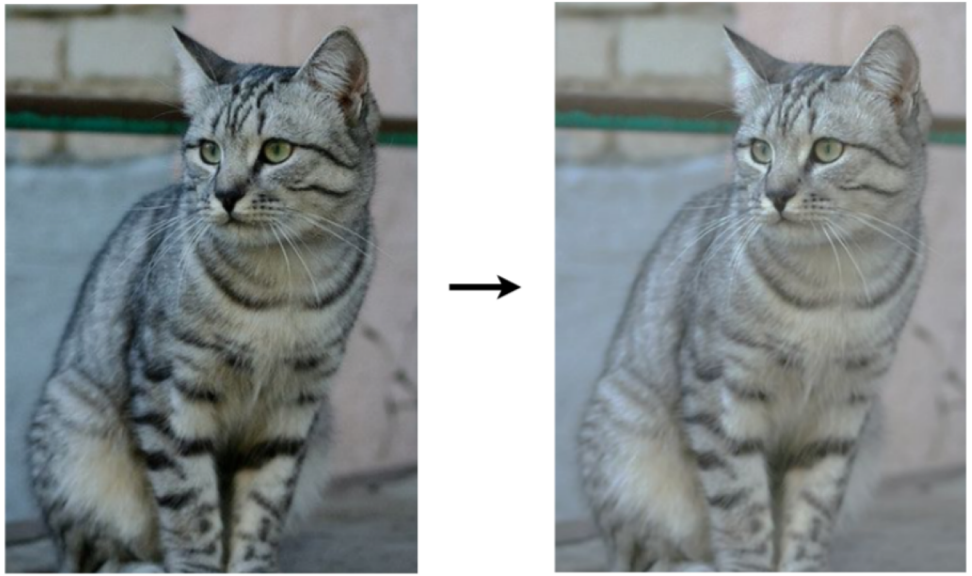
Data Augmentation

Horizontal flips

Random crops & scales

Color Jitter

Random perturbations in contrast and brightness



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Adapted from Leonid Sigal **532S-W2 (2018)

Motivation

- Data augmentation is very hand engineered
- Translations & Horizontal flipping, RGB Intensities (Krizhevsky et al, 2012)
- Augmentation techniques do not inherently translate
- Need for automatically learned optimal data augmentation approach

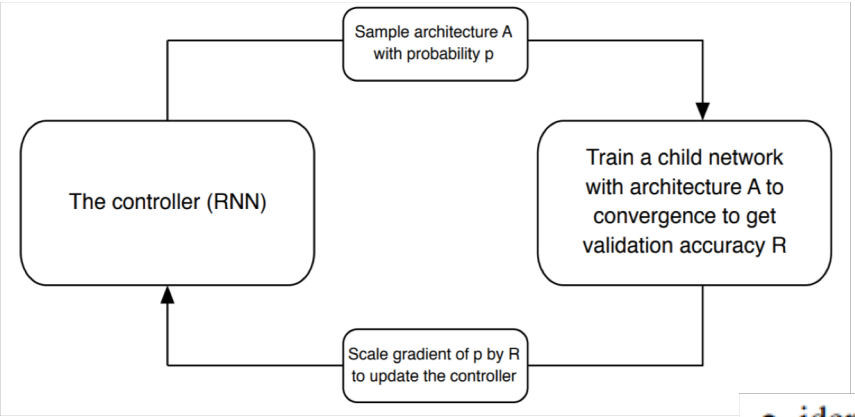
Goal

Automate the process of finding an effective data augmentation policy, that is transferable across datasets and architectures

Related Work

- Inspired by recent advances in the realm of AutoML using Reinforcement Learning e.g. NASNet

Neural Architecture Search (NAS)



- identity
- 1x7 then 7x1 convolution
- 3x3 average pooling
- 5x5 max pooling
- 1x1 convolution
- 3x3 depthwise-separable conv
- 7x7 depthwise-separable conv
- 1x3 then 3x1 convolution
- 3x3 dilated convolution
- 3x3 max pooling
- 7x7 max pooling
- 3x3 convolution
- 5x5 depthwise-seperable conv

Fig: Commonly used blocks for an image recognition network

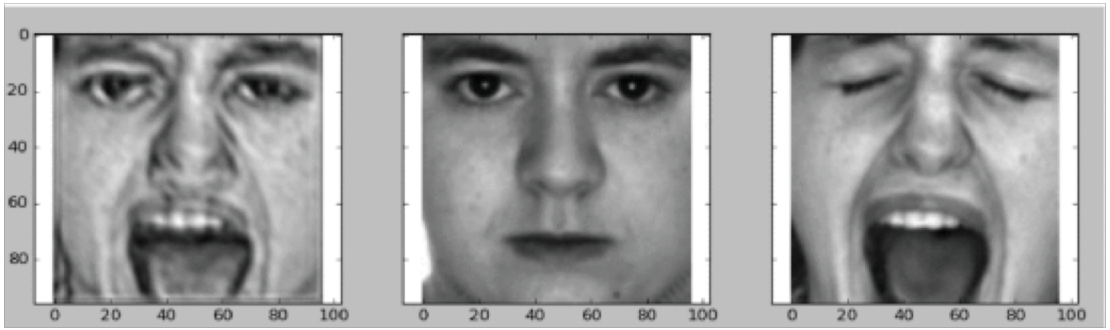
Zoph et al. (2018)

GANs for Data Augmentations

- Class dependant Vs. Class independent data generation
- Style based Augmentations
- Using GANs to train augmentation neural net

Smart Augmentation

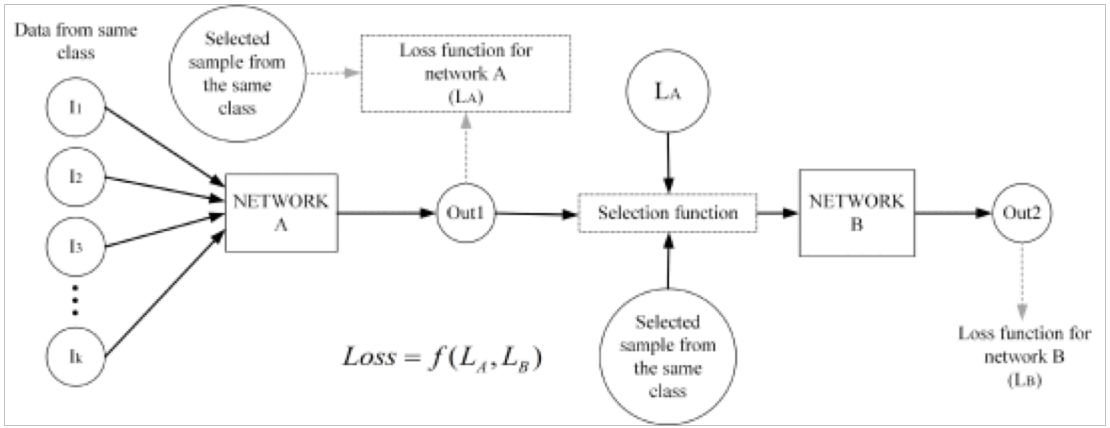
- Learning to merge two or more samples in one class.



Lemly et al. (2017)

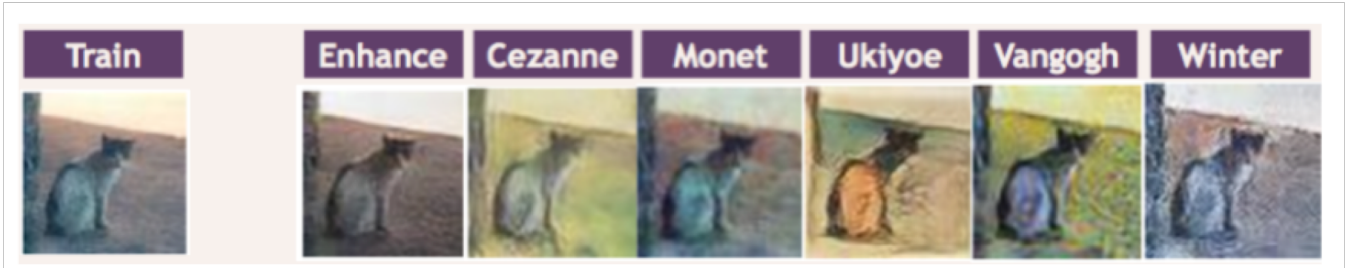
Smart Augmentation

- Use network A to learn the best data augmentation to train network B.
- End-to-end training
- Error propagated back to network A



Lemly et al. (2017)

Deep Learned Augmentation



Output from a style transfer network

Peres et al. (2017)

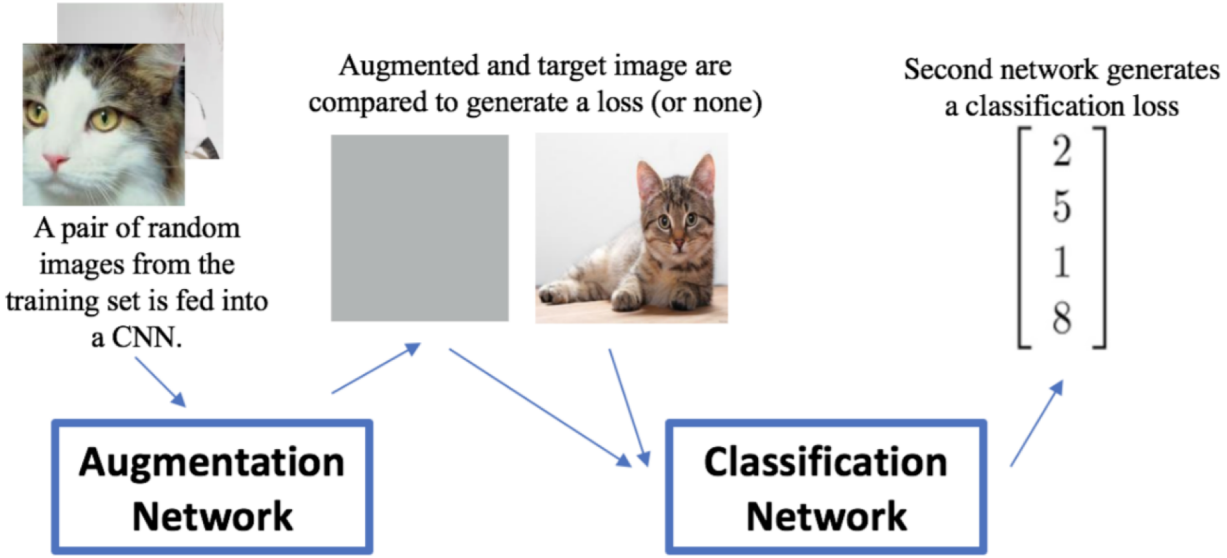
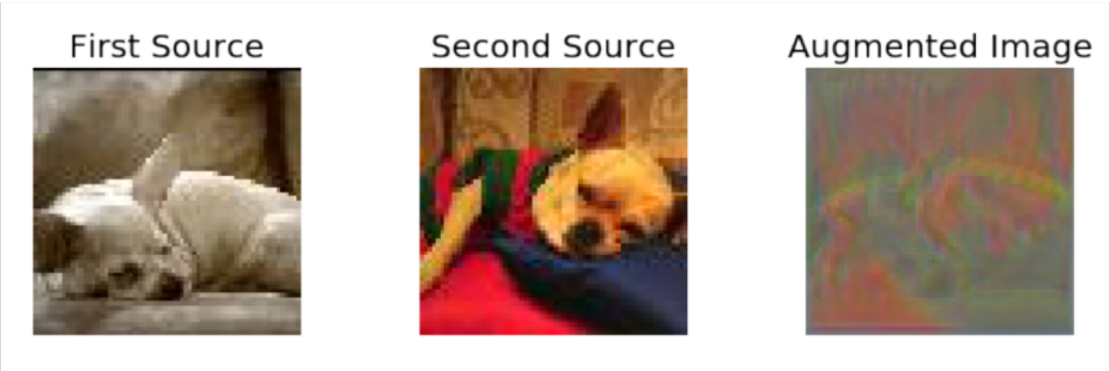


Figure III: Training model

Peres et al. (2017)

Deep Learned Augmentation

Output from a style transfer network



Output from a style transfer network

Peres et al. (2017)

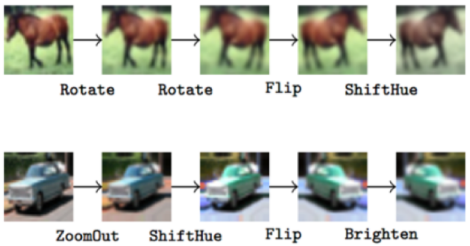
Domain-Specific Transformations

- Augmentative transformations are sequences of incremental black-box operations

Domain-Specific Transformations

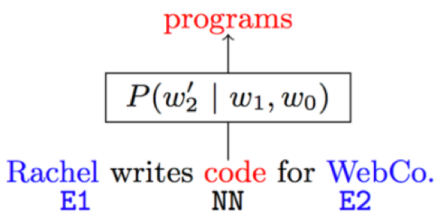
- Augmentative transformations are sequences of incremental black-box operations

Images



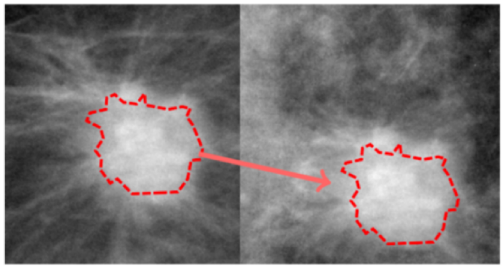
- Rotations
- Scaling / Zooms
- Brightness
- Color Shifts
- Etc…

Text



- Synonymy
- Positional Swaps
- Etc…

Medical



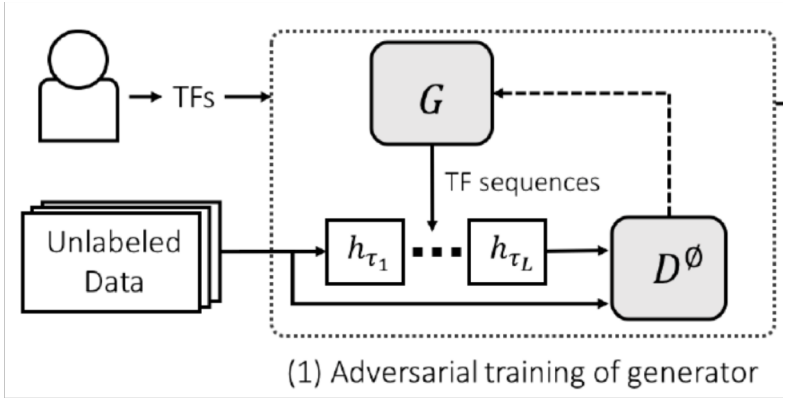
Domain-specific transformations.
Ex:

1. *Segment tumor mass*
2. *Move*
3. *Resample background tissue*
4. *Blend*

Ratner et al. (2017)

Domain-Specific Transformations

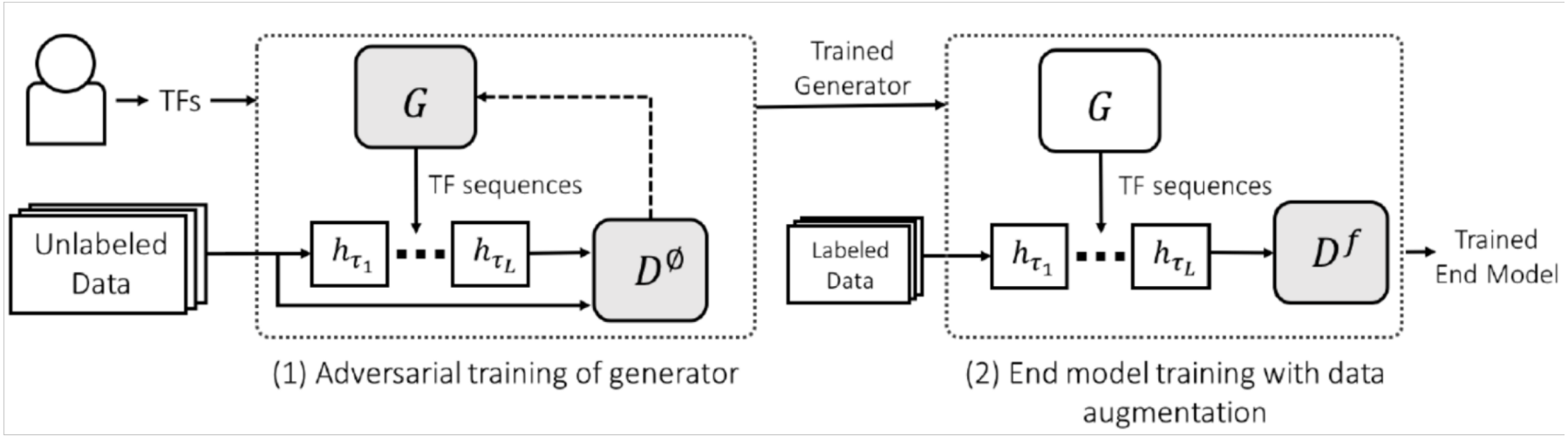
- Learn a generative sequence model that produces realistic, class-preserving augmentations



Ratner et al. (2017)

Domain-Specific Transformations

- Learn a generative sequence model that produces realistic, class-preserving augmentations



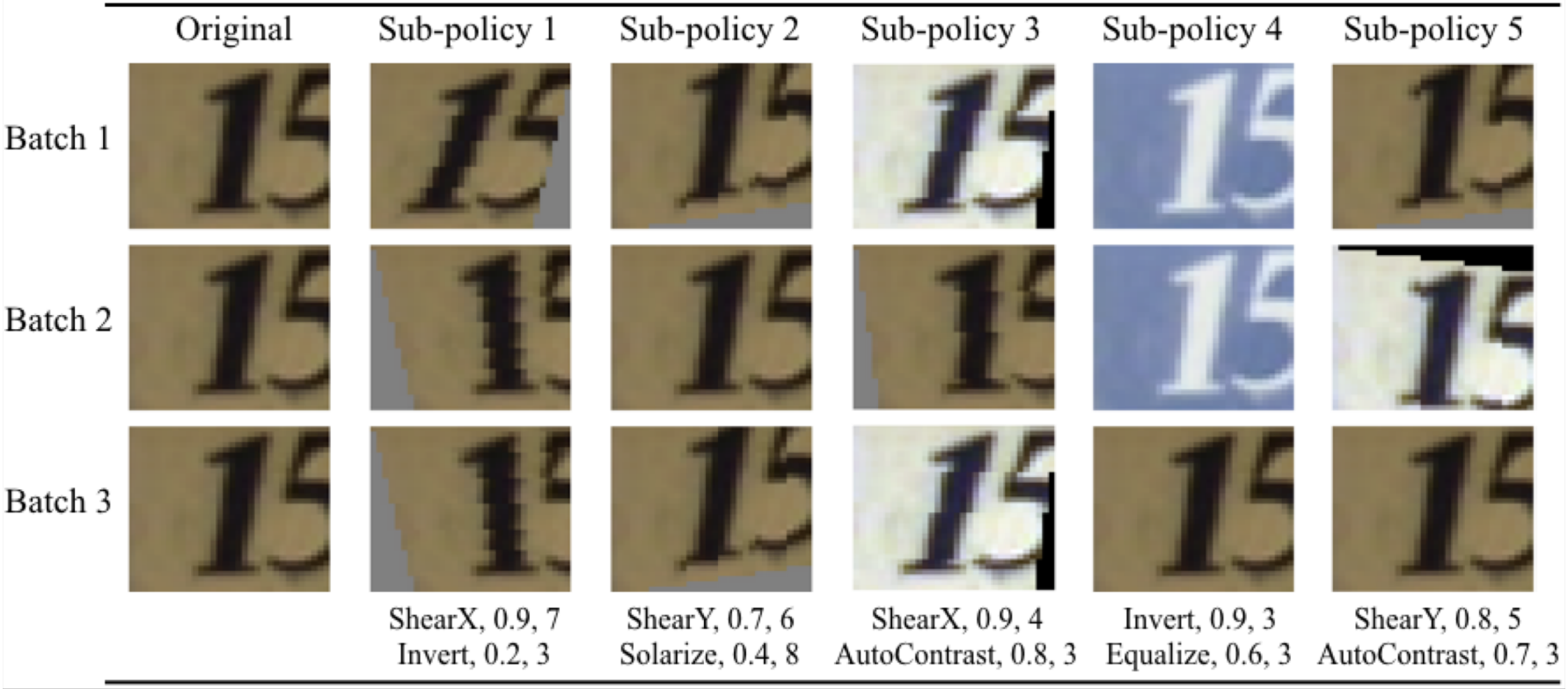
Ratner et al. (2017)

AutoAugment

- Discrete search problem.
- 5 sub-policies, 2 image operations.
- 2 hyperparameters: probability, magnitude.

Search Space

- Operations from PIL.
- 2 other augmentation techniques: Cutout, SamplePairing.
- Discretize range of magnitudes (uniform spacing).
- Discretize the probability (uniform spacing).
- Each sub-policy: $(16 \times 10 \times 11)^2$
- Goal: 5 sub-policies to increase diversity.
- Total possibilities: $(16 \times 10 \times 11)^{10} \approx 2.9 \times 10^{32}$


Original
Sub-policy 1
Sub-policy 2
Sub-policy 3
Sub-policy 4
Sub-policy 5
Batch 1
Batch 2
Batch 3
ShearX, 0.9, 7
Invert, 0.2, 3
ShearY, 0.7, 6
Solarize, 0.4, 8
ShearX, 0.9, 4
AutoContrast, 0.8, 3
Invert, 0.9, 3
Equalize, 0.6, 3
ShearY, 0.8, 5
AutoContrast, 0.7, 3

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, 73]	Set a random square patch of side-length *magnitude* pixels to gray.	[0,60]
Sample Pairing [51, 74]	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).

Search Algorithm

- RL to search for a policy; similar to NASNet.
- 2 components:

Search Algorithm

- RL to search for a policy; similar to NASNet.
- 2 components:
- **Controller**: RNN
  - Predicts a decision produced by softmax; 30 softmax predictions.
  - The decision is fed to controller's next step as input.

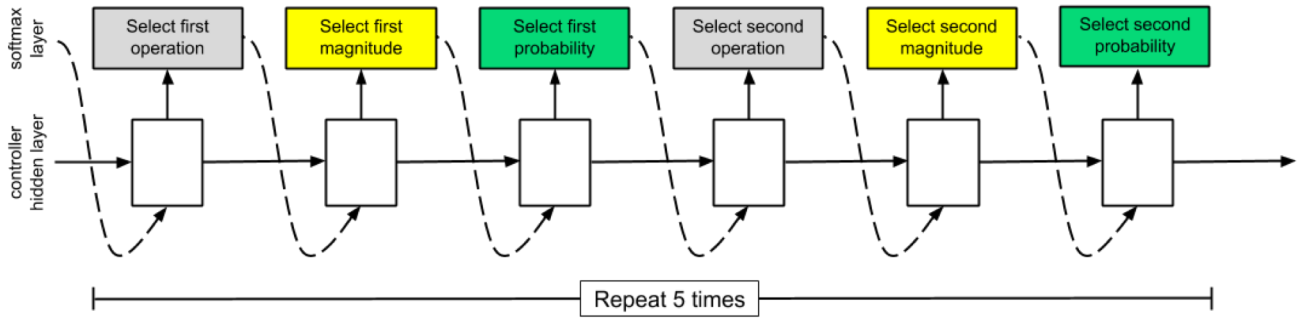


Image taken from https://arxiv.org/abs/1707.07012

Search Algorithm

- RL to search for a policy; similar to NASNet.
- 2 components:
- **Controller**: RNN
  - Predicts a decision produced by softmax over decision.
  - The decision is fed to controller's next step as input.
- **Training**: PPO algorithm
  - "Child model" is trained with the current data augmentation policy.
  - For each image in the mini-batch, one of the 5 sub-policies chosen randomly to augment the image.
  - Validation set to measure generalization of child model.
  - Accuracy → reward signal and the weights of RNN controller are updated using PPO.

Proximal Policy Optimization

- For the vanilla policy gradients:

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t\left[\log \pi_\theta(a_t \mid s_t)\hat{A}_t\right].$$

- But.. high variance.
- TRPO methods:

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right].$$

- But.. if r(θ) is very high? KL divergence constraints.
- Constraints are difficult to model, modify objective function? PPO!

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

PPO Algorithm

Algorithm 1 PPO, Actor-Critic Style

for iteration=$1, 2, \ldots$ **do**
  for actor=$1, 2, \ldots, N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
end for

Source: https://arxiv.org/pdf/1707.06347.pdf

Search Algorithm

- The controller samples about 15,000 policies for each dataset.
- Concatenate 5 best policies into 1; 25 sub-policies.
- This final policy is used to train the models for each dataset.

Experiments

Experiments

Benchmark Data Augmentation techniques:

- Baseline pre-processing:
  - Standardizing data, horizontal flips with 0.5 probability, zero padding and random crops.

Experiments

Benchmark Data Augmentation Techniques:

- Baseline pre-processing:
  - Standardizing data, horizontal flips with 0.5 probability, zero padding and random crops.

Image taken from http://nealan.vettivelu.com/wp-content/uploads/2017/07/Report-1.pdf

Experiments

Benchmark Data Augmentation Techniques:

- Baseline pre-processing:
  - Standardizing data, horizontal flips with 0.5 probability, zero padding and random crops.
- Cutout:
  - Randomly masking 16 x 16 regions in input during training.

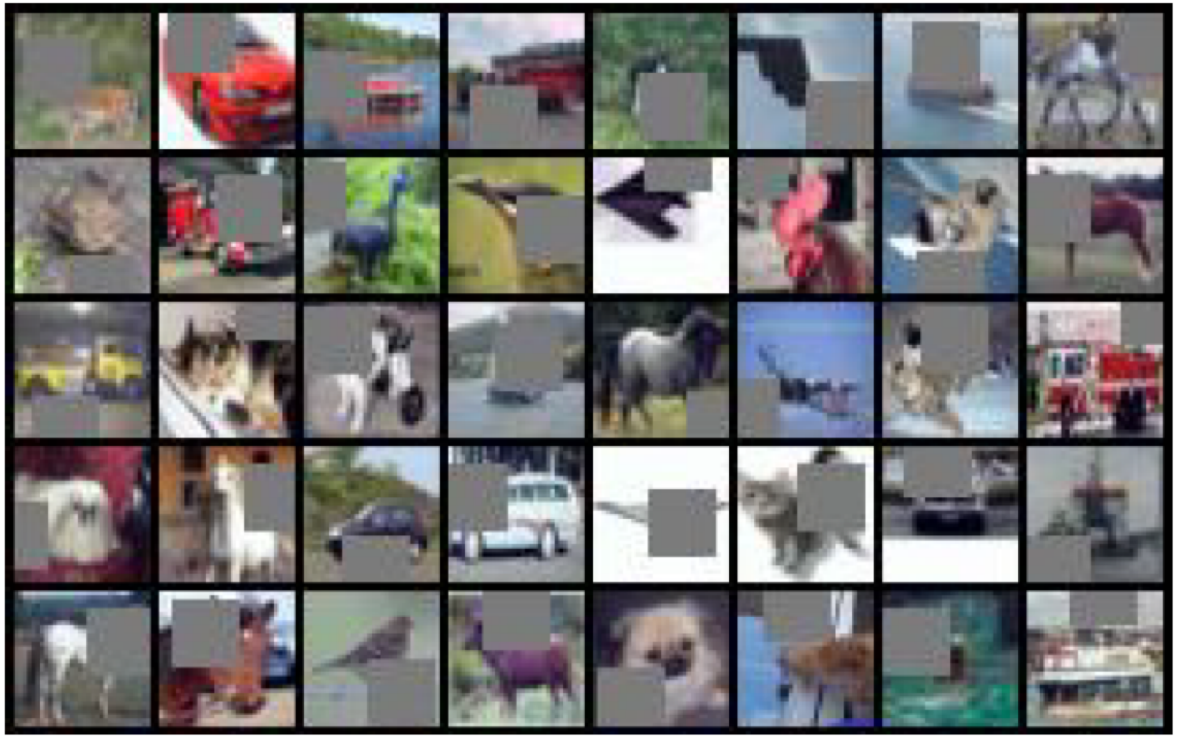


Figure 1: Cutout applied to images from the CIFAR-10 dataset.

Image taken from
https://arxiv.org/pdf/1708.04552.pdf

Experiments

- Dataset: **CIFAR-10** and **CIFAR-100**

	#Training images	#Test images	Resolution	#Classes
CIFAR-10	50k	10k	32 x 32	10
CIFAR-100	50k	10k	32 x 32	100

- Best policy search on reduced CIFAR-10 -- 4k random samples.
- Train child model on Wide-ResNet-40-2, 120 epochs.
- Small Wide-ResNet for computational efficiency.

Experiments

- Concatenate 5 best policies.
- Train final models on CIFAR-10 and CIFAR-100 using the best policy.
- Process of AutoAugment:
  - Baseline methods
  - AutoAugment Policy
  - Cutout with 16 x 16 pixels.

Results

CIFAR-10:

- Mostly color-based transformations.

	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.

Results

CIFAR-10:

- Mostly color-based transformations.
- Test error:

Model	Baseline	Cutout [25]	AutoAugment
Wide-ResNet-28-10 [57]	3.87	3.08	2.68
Shake-Shake (26 2x32d) [59]	3.55	3.02	2.47
Shake-Shake (26 2x96d) [59]	2.86	2.56	1.99
Shake-Shake (26 2x112d) [59]	2.82	2.57	1.89
AmoebaNet-B (6,128) [21]	2.98	2.13	1.75
PyramidNet+ShakeDrop [60]	2.67	2.31	**1.48**

Results

- Error of 1.48% with ShakeDrop model, 0.65% better than state-of-the-art.
- Replicated results match the previously reported; except one.

Model	Baseline	Cutout [25]	AutoAugment
Wide-ResNet-28-10 [57]	3.87	3.08	2.68
Shake-Shake (26 2x32d) [59]	3.55	3.02	2.47
Shake-Shake (26 2x96d) [59]	2.86	2.56	1.99
Shake-Shake (26 2x112d) [59]	2.82	2.57	1.89
AmoebaNet-B (6,128) [21]	2.98	2.13	1.75
PyramidNet+ShakeDrop [60]	2.67	2.31	**1.48**

Results

- Evaluated on recently prepared CIFAR-10 test set.

Model	Error Rate	Error Rate relative to original test set
Shake-Shake(26 2x64d) + Cutout	7.0%	4.1%
PyramidNet + ShakeDrop	7.7%	4.6%
PyramidNet + ShakeDrop + AutoAugment	4.4%	2.9%

Experiments and Results

CIFAR-100:

- Train on same AutoAugment policy found on reduced CIFAR-10.
- Test Error:

Model	Baseline	Cutout [25]	AutoAugment
Wide-ResNet-28-10 [57]	18.80	18.41	17.09
Shake-Shake (26 2x96d) [59]	17.05	16.00	14.28
PyramidNet+ShakeDrop [60]	13.99	12.19	**10.67**

- Results are replicated, matched.

Experiments and Results

Reduced CIFAR-10:

- Same dataset used for training as was used for finding the best policy.
- Test Error:

Model	Baseline	Cutout [25]	AutoAugment
Wide-ResNet-28-10 [57]	18.84	16.50	14.13
Shake-Shake (26 2x96d) [59]	17.05	13.40	**10.04**

- The improvement is more significant on reduced dataset than full dataset.

Experiments

Dataset: **SVHN**

SVHN Dataset

The image is taken from
http://ufldl.stanford.edu/housenumbers/

Experiments:

Dataset: **SVHN**

	#Training images	#Test images	Resolution	#Classes
SVHN	73,257 + 531,131	26032	32 x 32	10
Reduced SVHN	1k	--	32 x 32	10

- AutoAugment on reduced SVHN to find best policy.
- The policies picked are different from the transformations in 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.

Experiments

SVHN:

- Concatenate the best 5 policies into 1.
- For full training, Wide-ResNet architecture using core and the extra data.
- Validation set to tune the hyperparameters.
- Trained Shake-Shake model for 160 epochs.
- Baseline, cutout - 20 x 20 pixels.
- AutoAugment:
    - Baseline pre-processing
    - AutoAugment policy
    - ~~Cutout~~

Results

Model	Reduced SVHN Dataset			SVHN Dataset		
	Baseline	Cutout [25]	AA	Baseline	Cutout	AA
Wide-ResNet-28-10 [57]	13.21	32.5	8.15	1.50	1.30	1.07
Shake-Shake (26 2x96d) [59]	12.32	24.22	**5.92**	1.40	1.20	**1.02**

- Results replicated on Wide-Resnet replicated and matched.
- No previous results reported on Shake-Shake model for SVHN.

Experiment

ImageNet

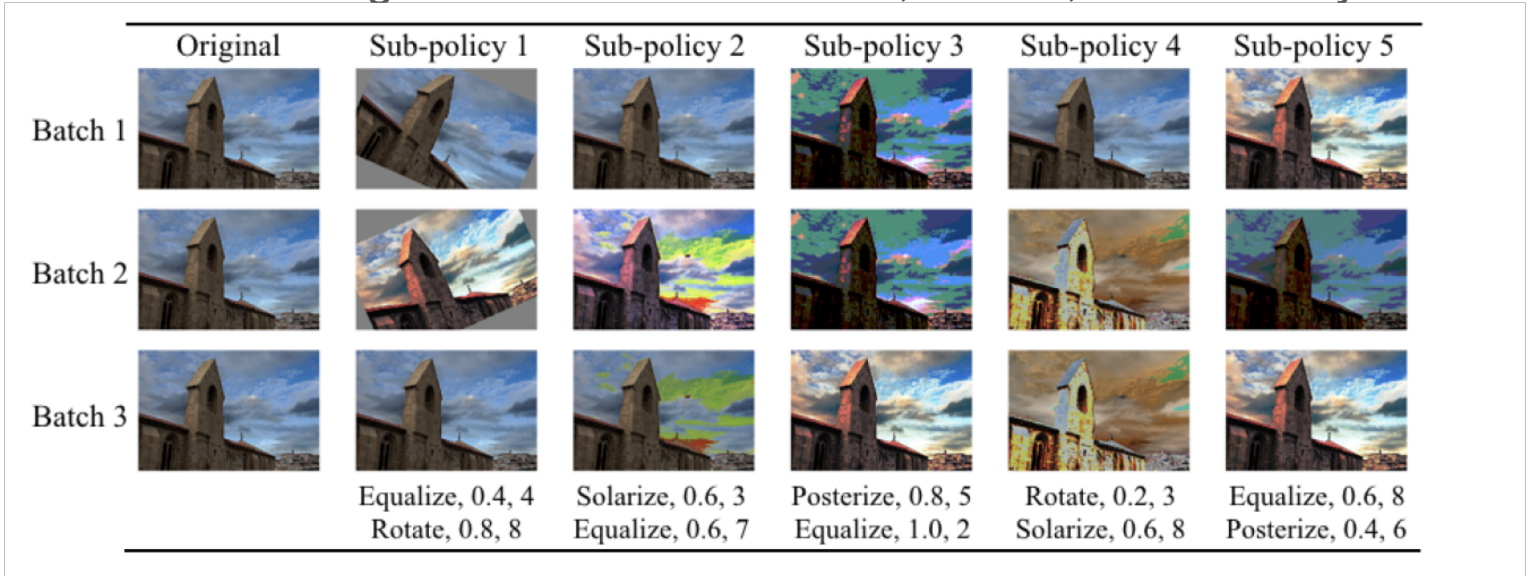
	Data	Augmentation	Model	Epoch	Learning Rate
Reduced ImageNet	6000 samples with 120 classes	Baseline augmentation	Wide-ResNet 40-2	200	0.1 with weight decay of 1e-4
Full ImageNet	Full	Baseline augmentation + policies learned	ResNet 50 ResNet 200	270	1.6 with weight decay by 10-fold at epochs 90, 180, 240

Baseline augmentation: standard Inception-style pre-processing

- ➔ scaling pixel values to [-1,1]
- ➔ horizontal flips with 50% probability
- ➔ random distortions of colors

Results

- The best policies found on imagenet are **similar** to those found on **CIFAR-10**
  - Color-based transformations
- One difference: a geometric transformation, Rotate, is commonly used



Results

Validation set Top-1 / Top-5 error rates (%) on ImageNet

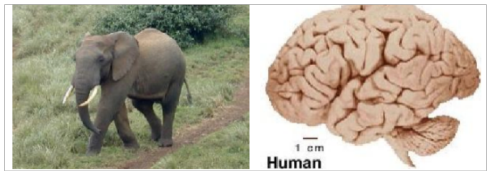
Model	Baseline	Inception Pre-processing [14]	AutoAugment
ResNet-50 [15]	24.70 / 7.80	23.69 / 6.92	22.37 / 6.18
ResNet-200 [15]	-	21.52 / 5.85	20.00 / 4.99
AmoebaNet-B (6,190) [21]	-	17.80 / 3.97	17.25 / 3.78
AmoebaNet-C (6,228) [21]	-	16.90 / 3.90	**16.46 / 3.52**

Experiment

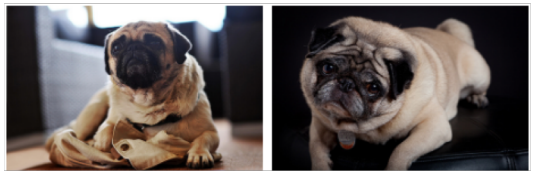
Fine Grained Visual Classification Datasets

Oxford 102 Flowers

Caltech-101



Oxford-IIIT Pets



FGVC Aircraft

Stanford Cars

Small set of training examples with
Large amount of classes

Experiment

Fine Grained Visual Classification Datasets

- Use the same policy learned on ImageNet

Dataset	Image size	model	epoch	Learning rate
5 FGVC datasets	448*448	InceptionV4	1000	Cosine learning rate decay

Result

Fine Grained Visual Classification Datasets

- The policies found on imagenet improve the generalization accuracy of all 5 FGVCD datasets significantly
- The lowest error rate achieved on Stanford Cars dataset although training from the scratch

Dataset	Train Size	Classes	Inception Pre-processing [14]	AutoAugment
Oxford 102 Flowers [69]	2,040	102	6.69	4.64
Caltech-101 [70]	3,060	102	19.35	13.07
Oxford-IIIT Pets [71]	3,680	37	13.46	11.02
FGVC Aircraft [29]	6,667	100	9.09	7.33
Stanford Cars [28]	8,144	196	6.35	5.19

Discussion

Discussion

How many iterations are required before the model can fully benefit from all of the sub-policies?

Discussion

Requirement for stochastic application of sub-policies to be effective

- Search place
  - Each sub-policy needs a certain number of epochs
  - Child model with 5 sub-policies: 80 - 100 epochs
    - 120 epochs are chosen
- The full model is trained for longer
  - 270 epochs for ResNet-50 on ImageNet
  - 1800 epochs for Shake-shake on CIFAR-10

Discussion

How would the number of sub-policies affect the generalization accuracy?

Discussion

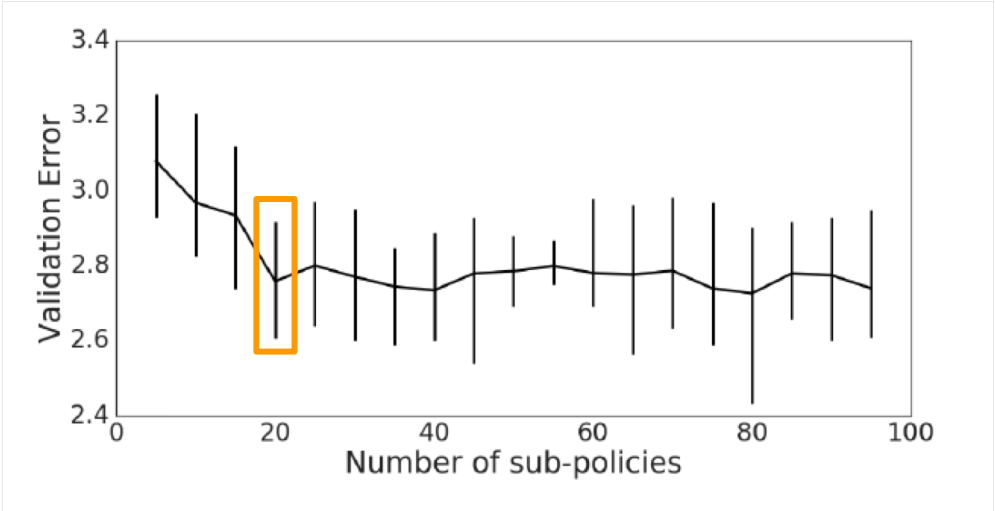
How would the number of sub-policies affect the generalization accuracy?

- Hypothesis:
  - number of sub-policies → diversity → generalization accuracy
- Test:
  - randomly select sub-policies sets from 500 good sub-policies
  - train the Wide-ResNet-28-10 model for 200 epochs on CIFAR-10

Discussion

How would the number of sub-policies affect the generalization accuracy?

- Hypothesis:
  - number of sub-policies → diversity → generalization accuracy
- Test:



Strength and Weakness

Strength

- ★ Automatically learned data-augmentation
- ★ Transferability across datasets and architectures

Strength and Weakness

Strength

- ★ Automatically learned data-augmentation
- ★ Transferability across datasets and architectures

Weakness

- ❖ Computationally expensive and time consuming
- ❖ Not use a different discrete search algorithm

Extension

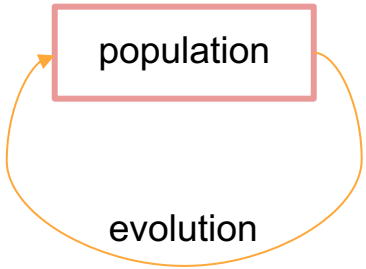
Algorithm and architectures in Neural architecture search(NAS)

Extension

“Regularized Evolution for Image Classifier Architecture Search”(2018)

Regularized(aging) Evolution Algorithm

Initialized with models with random architectures



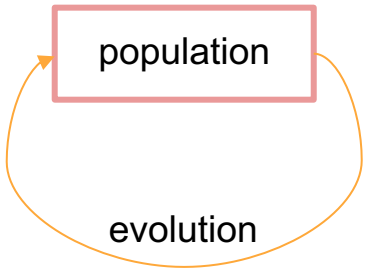
Extension

“Regularized Evolution for Image Classifier Architecture Search”(2018)

Regularized(aging) Evolution Algorithm

parent

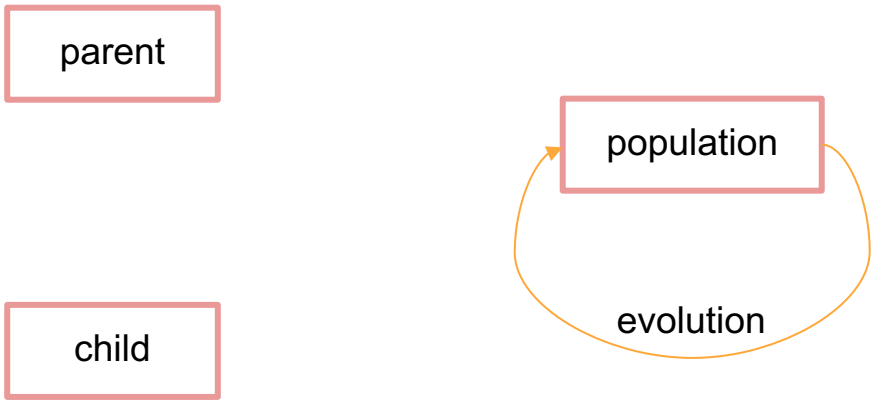
Model with the highest validation fitness at each cycle



Extension

“Regularized Evolution for Image Classifier Architecture Search”(2018)

Regularized(aging) Evolution Algorithm



A new architecture with mutation from parent

Extension

“Regularized Evolution for Image Classifier Architecture Search”(2018)

Regularized(aging) Evolution Algorithm

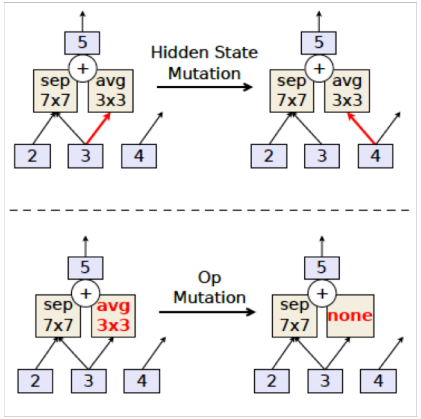
parent

child

population

evolution

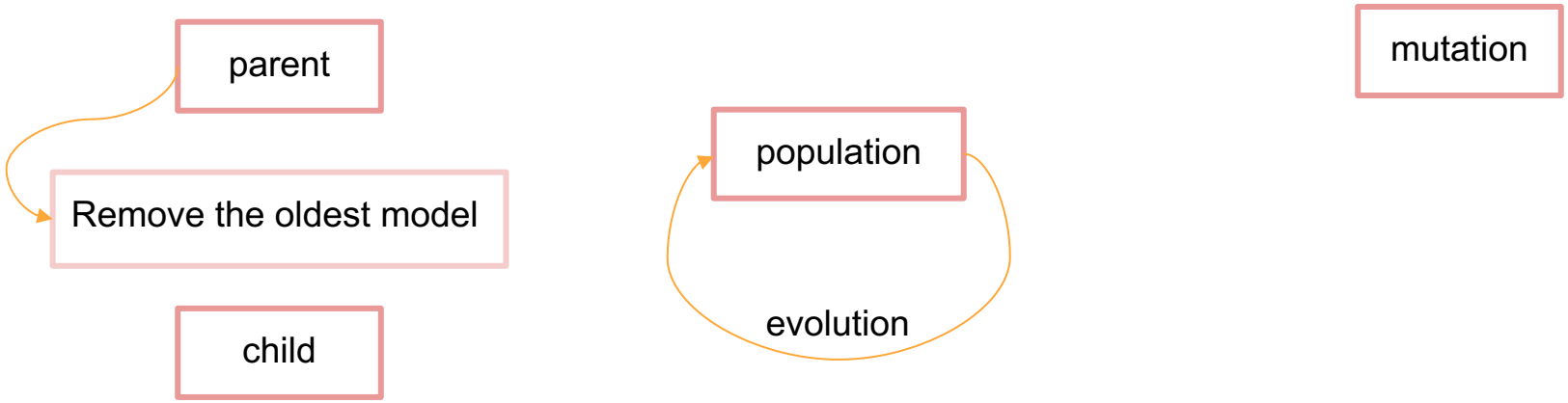
mutation



Extension

“Regularized Evolution for Image Classifier Architecture Search”(2018)

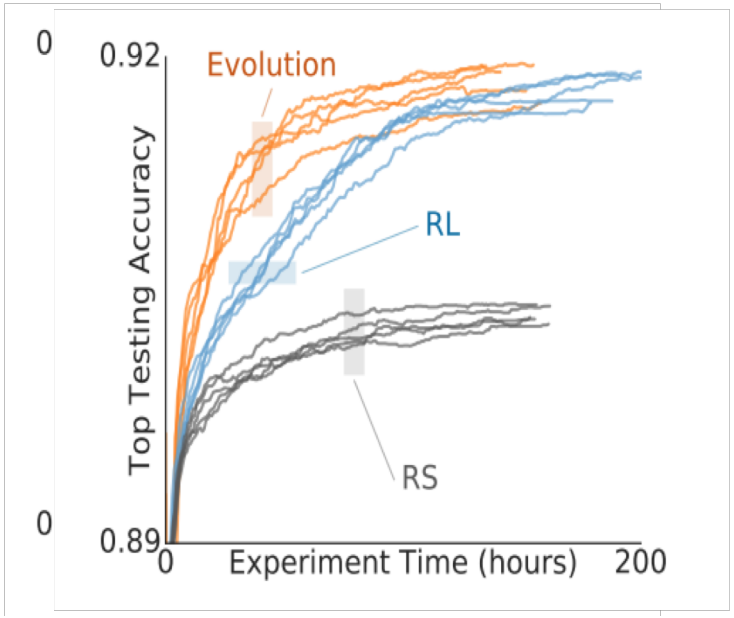
Regularized(aging) Evolution Algorithm

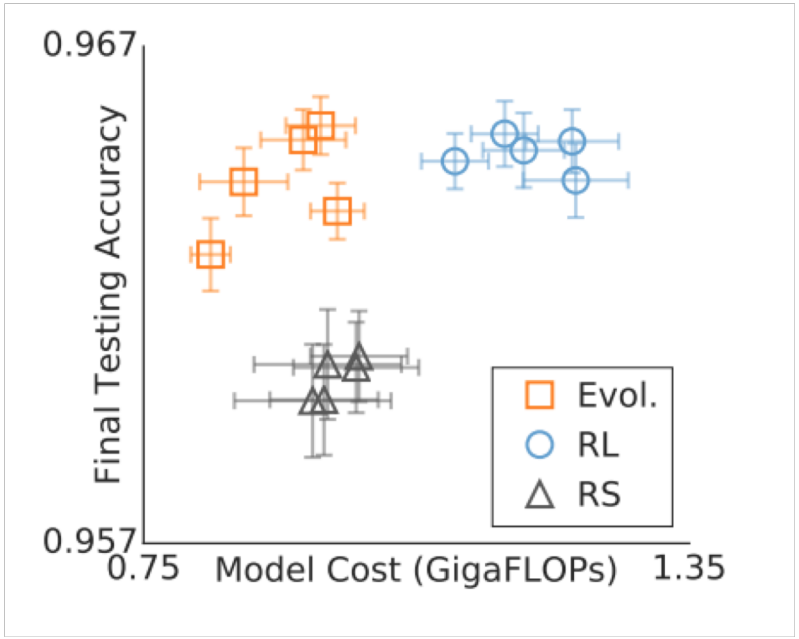


Extension

“Regularized Evolution for Image Classifier Architecture Search”(2018)

Regularized(aging) Evolution Algorithm



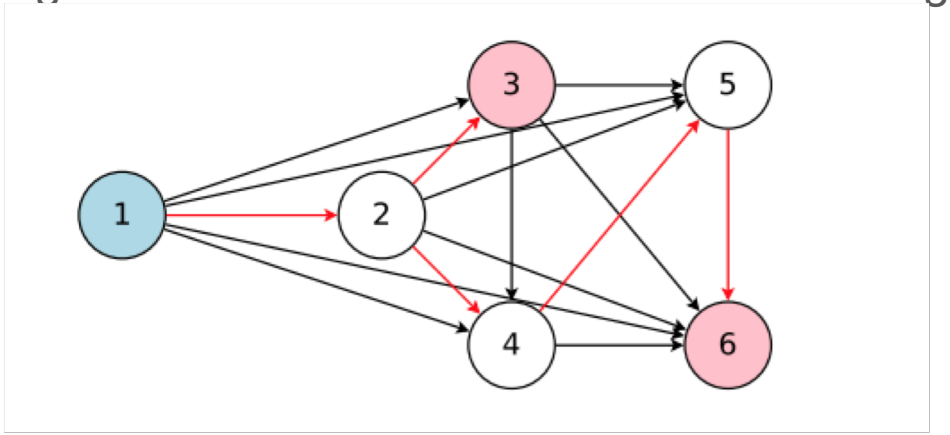


Extension

“Efficient Neural Architecture Search via Parameter Sharing”(2018)

Efficient Neural Architecture Search

- forcing all child models to share weights
- eschew training each child model from scratch to convergence.



Extension

“Efficient Neural Architecture Search via Parameter Sharing”(2018)

Efficient Neural Architecture Search

Method	**GPUs**	**Times** (days)	**Params** (million)	**Error** (%)
Hierarchical NAS (Liu et al., 2018)	200	1.5	61.3	3.63
Micro NAS + Q-Learning (Zhong et al., 2018)	32	3	–	3.60
Progressive NAS (Liu et al., 2017)	100	1.5	3.2	3.63
NASNet-A (Zoph et al., 2018)	450	3-4	3.3	3.41
NASNet-A + CutOut (Zoph et al., 2018)	450	3-4	3.3	**2.65**
ENAS + micro search space	1	0.45	4.6	3.54
ENAS + micro search space + CutOut	1	0.45	4.6	**2.89**

THANK YOU FOR YOUR TIME

ANY QUESTIONS?