

AutoAugment: Learning Augmentation Strategies from Data

(Cubuk, Zoph, Mane, Vasudevan, Le) Google Brain
CVPR 2019

vkaushal@cse.iitb.ac.in | www.vishalkaushal.in

Motivation 1: Automation

What do ML Scientists do?

- **Identify problems** that can potentially be solved by ML
- **Data** collection, labeling, preprocessing, splitting
- **Model** design/selection, hyper parameter tuning, training, troubleshooting

What do ML Scientists do?

- **Identify problems** that can potentially be solved by ML
- **Data** collection, labeling, preprocessing, splitting
- **Model** design/selection, hyper parameter tuning, training, troubleshooting

These become VERY involving, especially with Deep Learning for Computer Vision

Auto ML



Current:

Solution = ML expertise + data + computation

Can we turn this into:

Solution = data + 100X computation

???

GOOGLE'S LEARNING SOFTWARE LEARNS TO WRITE LEARNING SOFTWARE

AutoML?

- What **CAN** be automated, **SHOULD** be and **WILL** be automated

GOOGLE'S LEARNING SOFTWARE LEARNS TO WRITE LEARNING SOFTWARE

AutoML?

- What **CAN** be automated, **SHOULD** be and **WILL** be automated
- Example: Neural Architecture Search
 - Google's **NAS** "Neural Architecture Search with reinforcement learning" ICLR 2017 (Same group)
 - **NASNet**: "Learning transferable architectures for scalable image recognition." CVPR 2018 (Same group)
 - **ENAS**: "Efficient neural architecture search via parameter sharing" ICML 2018 (Same group)
 - **AmoebaNet**: "Regularized evolution for image classifier architecture search" AAAI 2019 (Same group)
 - **DARTS**: Differentiable Architecture Search, ICLR 2019 (CMU and DeepMind - Almost same group)
- Tons of resources at www.automl.org

Problem with manual data augmentation?

- Best augmentation strategies are **dataset specific**
 - MNIST: elastic distortions, scale, translation and rotation
 - Natural image datasets like CIFAR, ImageNet: random cropping, image mirroring, color shifting/whitening
- **Require expert knowledge and time**

Motivation 2: Generalizability

Manually designed techniques are non-transferrable

- Because of different image characteristics
- Examples:
 - Horizontal flipping is effective on CIFAR-10, but not on MNIST



Motivation 3: Low Hanging Fruit

Two ways of making a model invariant to certain data characteristics

- **Hardcoding in model architecture**
 - Example: CNNs are translation invariant
- **Data augmentation**
 - Effective technique for improving accuracy by **class preserving transformations**
 - Translate, flip, rotate, ...
- **Latter can be easier than former**
 - **Yet primary focus is on engineering better networks**
 - VGGNet, GoogLeNet, ResNet, ResNeXt, PyramidNet, SENet, NASNet, AmoebaNet,
- “It has not been possible to beat the 2% error rate barrier on CIFAR-10 using architecture search alone”

Related Work

- “A **Bayesian** Data Augmentation Approach for Learning Deep Models” NIPS 2017
 - Bayesian formulation where new annotated training points are treated as missing variables and generated based on the distribution learned from the training set
- “[Dataset augmentation in feature space](#)” ICLR 2017 Workshop
 - Perform the transformation not in input space, but in a learned feature space
- “[Smart Augmentation](#) Learning an Optimal Data Augmentation Strategy” 2017
 - Proposed a network that automatically generates augmented data by merging two or more samples from the same class to create new samples
- Various GAN approaches
- All of the above **generate augmented data directly**
- **Exception** - “[Learning to Compose Domain-Specific Transformations for Data Augmentation](#)” NIPS 2017
 - used GANs to generate transformation sequences

Data Augmentation in Learned Feature Space

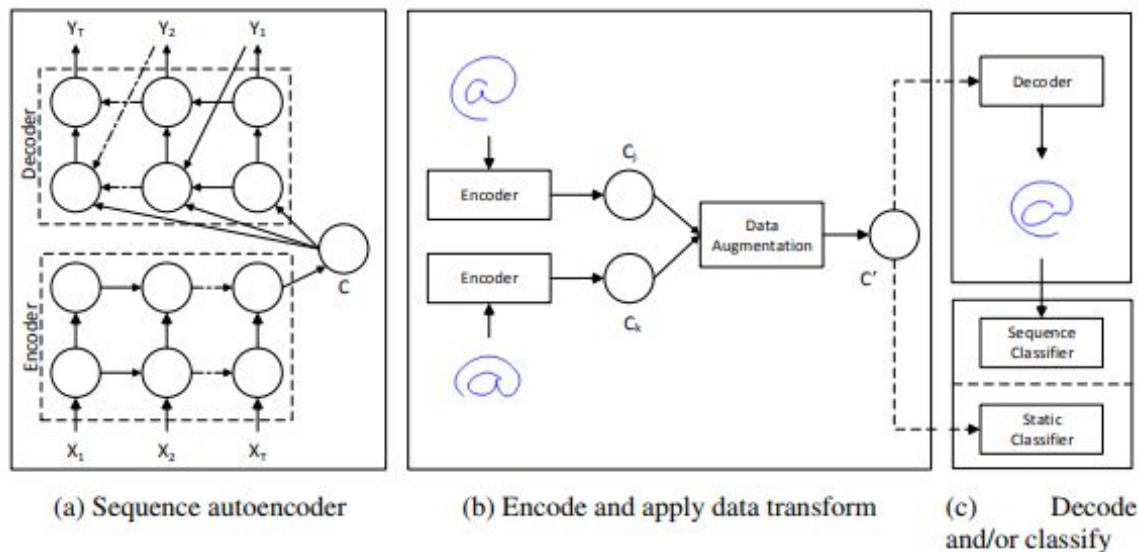


Figure 1: System architecture composed of three steps. (a) A sequence autoencoder learns a feature space from unlabeled data, representing each sequence by a context vector (C). (b) Data is encoded to context vectors and augmented by adding noise, interpolating, or extrapolating (here we depict interpolation). (c) The resulting context vectors can either be used directly as features for supervised learning with a static classifier, or they can be decoded to reconstruct full sequences for training a sequence classifier.

Smart Augmentation

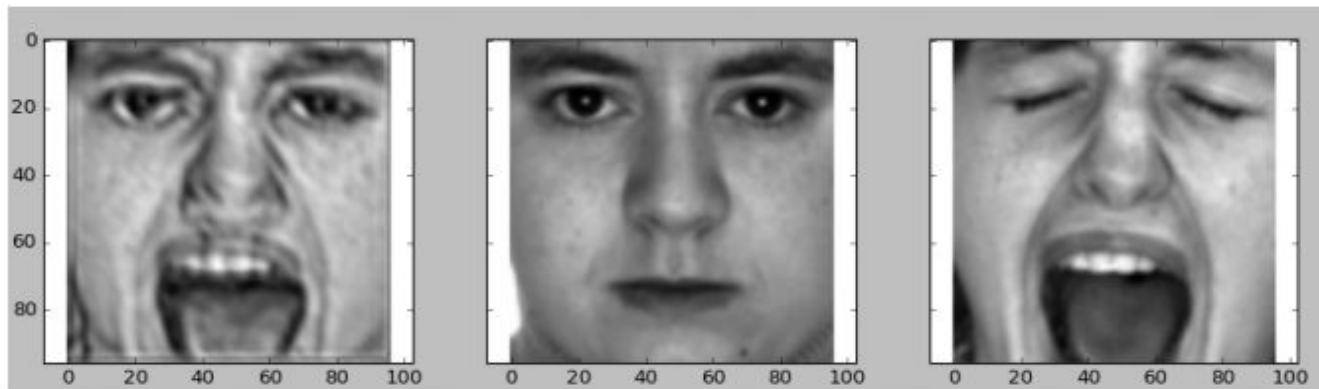


FIGURE 3. The image on the left is created by a learned combination of the two images on the right. This type of image transformation helped increase the accuracy of network B. The image was not produced to be an ideal approximation of a face but instead, contains features that helped network B better generalize the concept of gender which is the task it was trained for.

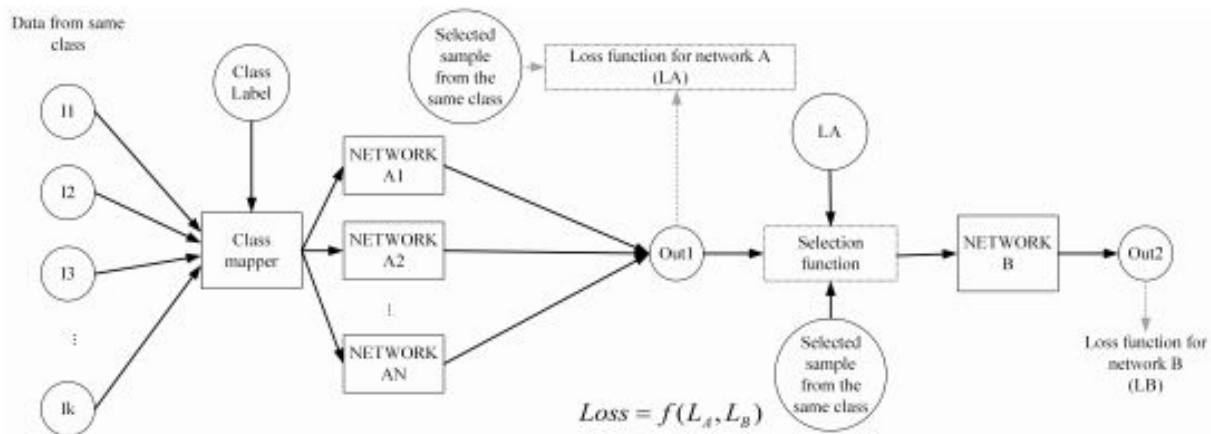


FIGURE 1. Smart augmentation with more than one network A.

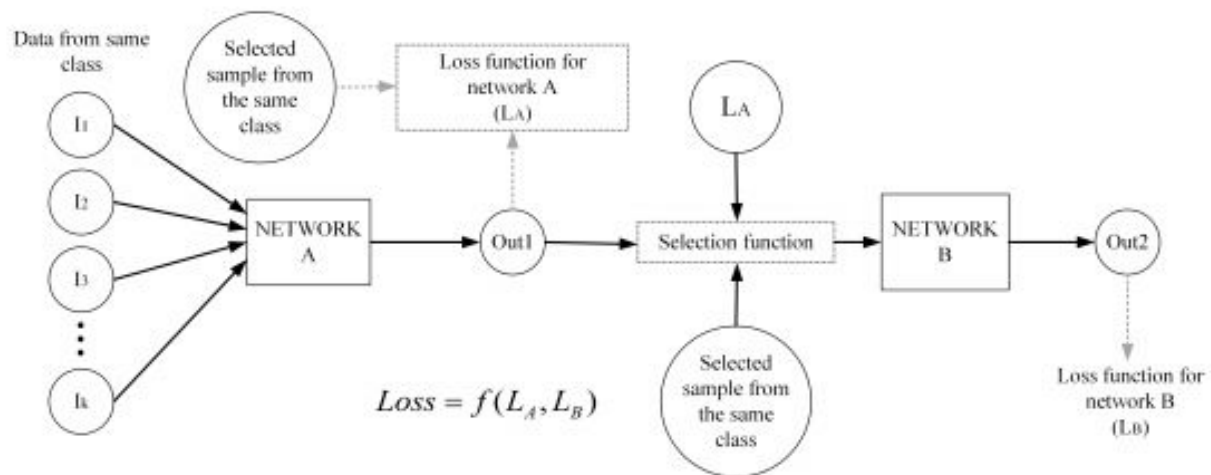


FIGURE 2. Diagram illustrating the reduced smart augmentation concept with just one network A.

Learning Transformations Using GAN

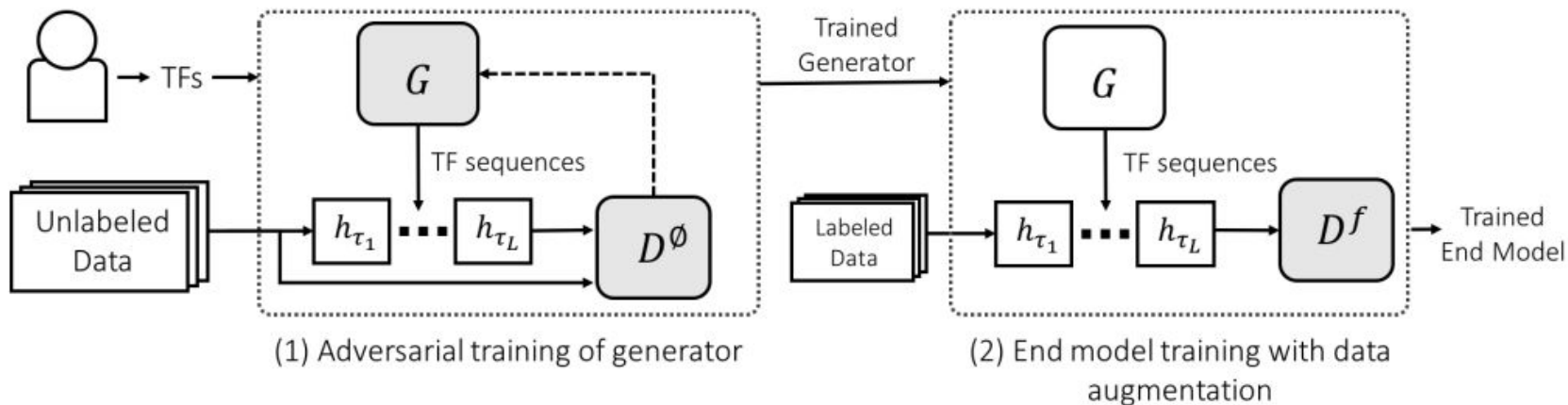


Figure 2: A high-level diagram of our method. Users input a set of transformation functions h_1, \dots, h_K and unlabeled data. A generative adversarial approach is then used to train a *null class* discriminator, D^\emptyset , and a generator, G , which produces TF sequences $h_{\tau_1}, \dots, h_{\tau_L}$. Finally, the trained generator is used to perform data augmentation for an end discriminative model D^f .

Definition of a Data Augmentation “Policy”

- A **policy** consists of many subpolicies
- A **subpolicy** contains
 - Operation 1, Probability, Magnitude
 - Operation2, Probability, Magnitude
- **Operations** to be applied in sequence
- **Operations**
 - Image operations from PIL: ShearX/Y, TranslateX/Y, Rotate, AutoContrast, Invert, Equalize, Solarize, Posterize, Contrast, Color, Brightness, Sharpness
 - Cutout
 - SamplePairing
- No explicit Identity operation



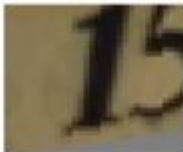















Operation Name	Description	Range of magnitudes
ShearX(Y)	Shear the image along the horizontal (vertical) axis with rate <i>magnitude</i> .	[-0.3,0.3]
TranslateX(Y)	Translate the image in the horizontal (vertical) direction by <i>magnitude</i> number of pixels.	[-150,150]
Rotate	Rotate the image <i>magnitude</i> 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 <i>magnitude</i> .	[0,256]
Posterize	Reduce the number of bits for each pixel to <i>magnitude</i> bits.	[4,8]
Contrast	Control the contrast of the image. A <i>magnitude</i> =0 gives a gray image, whereas <i>magnitude</i> =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 <i>magnitude</i> =0 gives a black & white image, whereas <i>magnitude</i> =1 gives the original image.	[0.1,1.9]
Brightness	Adjust the brightness of the image. A <i>magnitude</i> =0 gives a black image, whereas <i>magnitude</i> =1 gives the original image.	[0.1,1.9]
Sharpness	Adjust the sharpness of the image. A <i>magnitude</i> =0 gives a blurred image, whereas <i>magnitude</i> =1 gives the original image.	[0.1,1.9]
Cutout [12, 69]	Set a random square patch of side-length <i>magnitude</i> 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 <i>magnitude</i> , without changing the label.	[0, 0.4]

Table 6. 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 as shown in the third column (for image size 331x331). Some transformations do not use the magnitude information (e.g. Invert and Equalize).

How is AutoAugment applied during training?

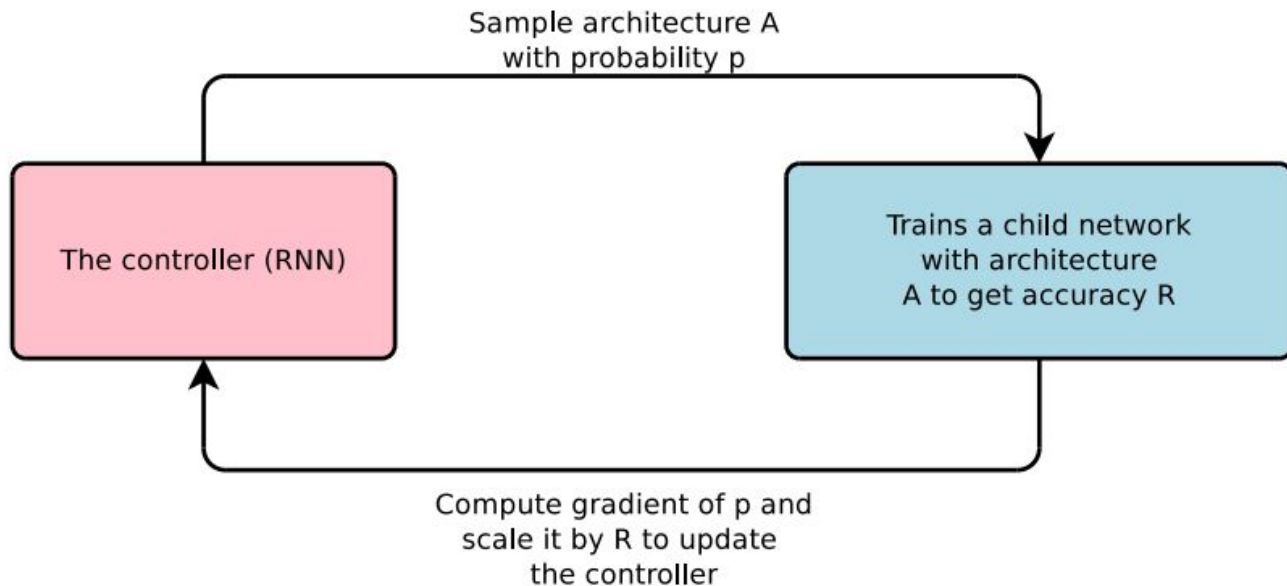
- One subpolicy of the optimal policy is randomly picked up for each image in each mini batch
- Operation 1 is applied with the probability and magnitude
- Operation 2 is applied with the probability and magnitude
- **One image can be transformed differently in different mini-batches even with the same subpolicy**

Example

	Original	Sub-policy 1	Sub-policy 2	Sub-policy 3	Sub-policy 4	Sub-policy 5
Batch 1						
Batch 2						
Batch 3						

How is an optimal policy learnt?

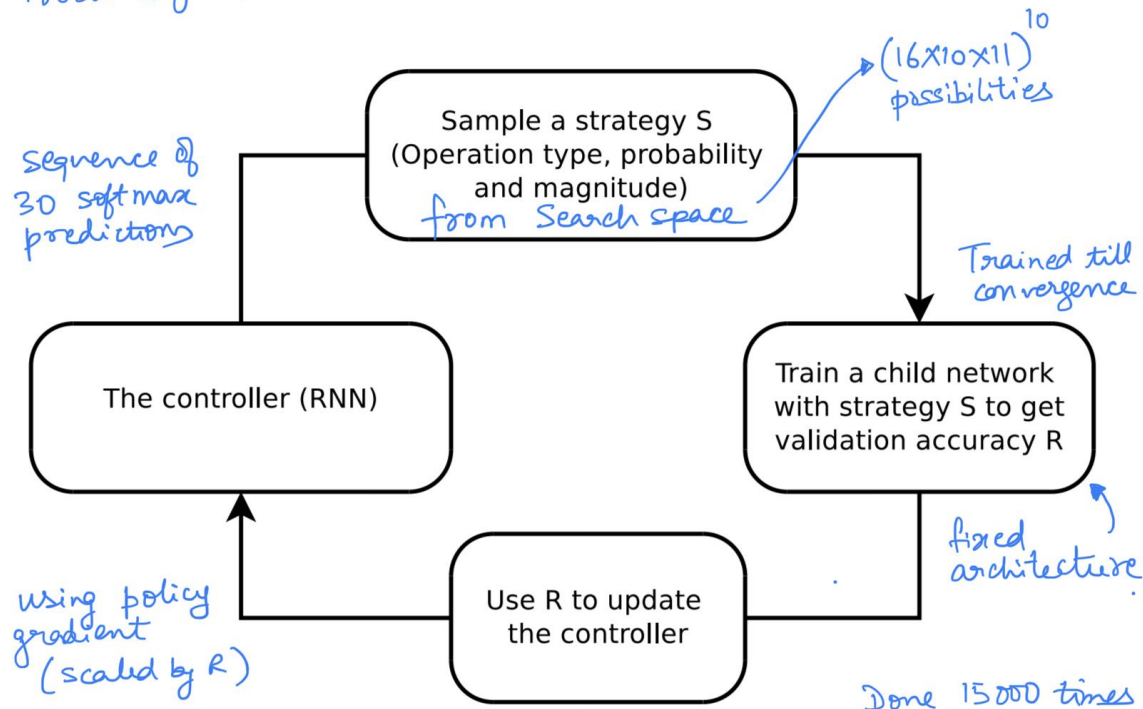
- Inspired by Neural Architecture Search (**NAS**) framework proposed by “Neural Architecture Search with Reinforcement Learning” ICLR 2017



How is an optimal policy learnt?

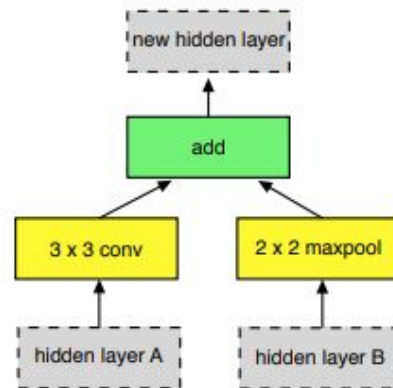
- Discrete search problem - RL as search algorithm
- In search space, every policy has 5 subpolicies

Operations - 16
Magnitude - 10
Probability - 11

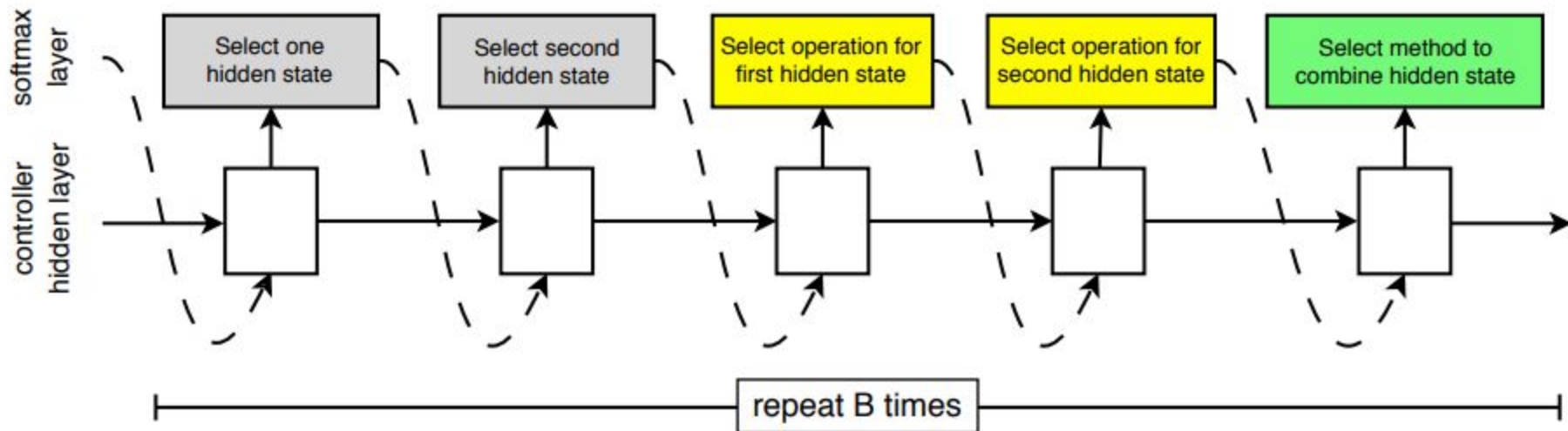


Controller Architecture

- Inspired by “Learning Transferable Architectures for Scalable Image Recognition” Google Brain, CVPR 2018 (**NASNet**)
- Identify **repeating patterns** that are at the core of successful architectures
- Two types of convolutional cells
 - **Normal cell** - convolutional cells that return a feature map of the same dimension
 - **Reduction cell** - convolutional cells that return a feature map where the feature map height and width is reduced by a factor of two
- Search space: structures of normal and reduction cells
- Each convolutional cell contains B such blocks
 - Each block is defined by 5 discrete parameters



Controller Architecture



- 1 layer LSTM with 100 hidden units
- $2 \times 5 \times B$ softmax predictions for one architectural decision
- Joint probability of child network = product of all $10B$ probabilities
- Each child model is trained from scratch to compute the gradient update

REINFORCE Algorithm

$$J(\theta_c) = E_{P(a_{1:T}; \theta_c)}[R] \quad \rightarrow \text{Expected reward. To be maximized.}$$

$$\nabla_{\theta_c} J(\theta_c) = \sum_{t=1}^T E_{P(a_{1:T}; \theta_c)} [\nabla_{\theta_c} \log P(a_t | a_{(t-1):1}; \theta_c) R] \quad \rightarrow \text{Policy gradient method REINFORCE}$$

$$\frac{1}{m} \sum_{k=1}^m \sum_{t=1}^T \nabla_{\theta_c} \log P(a_t | a_{(t-1):1}; \theta_c) R_k$$

\rightarrow Empirical approximation of the above quantity.
 m = no. of samples in one batch
 T = no. of parameters that define child network's architecture

$$\frac{1}{m} \sum_{k=1}^m \sum_{t=1}^T \nabla_{\theta_c} \log P(a_t | a_{(t-1):1}; \theta_c) (R_k - b)$$

\rightarrow baseline function b to reduce the variance
For the update to remain unbiased, b should not depend on the current action
 b = exponential moving average of previous accuracies

Training Algorithm

- Gradient computed from joint probability is scaled by R such that controller assigns low probabilities for bad child networks and high probabilities for good child networks
- PPO with learning rate 0.00035
 - Unlike NAS, which used REINFORCE from “Simple statistical gradient-following algorithms for connectionist reinforcement learning” (1992)
 - Enables multiple epochs of minibatch updates
 - PPO is faster and more stable - other methods may perform better
- Entropy penalty with a weight of 0.00001
 - To encourage exploration
- Baseline function - exponential moving average of previous rewards with a weight of 0.95
- Weights of LSTM are initialized uniformly between -0.1 and 0.1

Reinforcement Learning Algorithms

- **Q-learning** (with function approximation) fails on many simple problems and is poorly understood
- **Vanilla policy gradient methods** have poor data efficiency and robustness
- **Trust region policy optimization (TRPO)** is relatively complicated, and is not compatible with architectures that include noise (such as dropout) or parameter sharing (between the policy and value function, or with auxiliary tasks)
- Hence the authors decide to go with **PPO**

Optimal Policy?

- Concatenation of subpolicies of 5 best policies to have one optimal policy with 25 subpolicies

Learnt Policies

	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	(ShearX,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 7. AutoAugment policy found on reduced CIFAR-10.

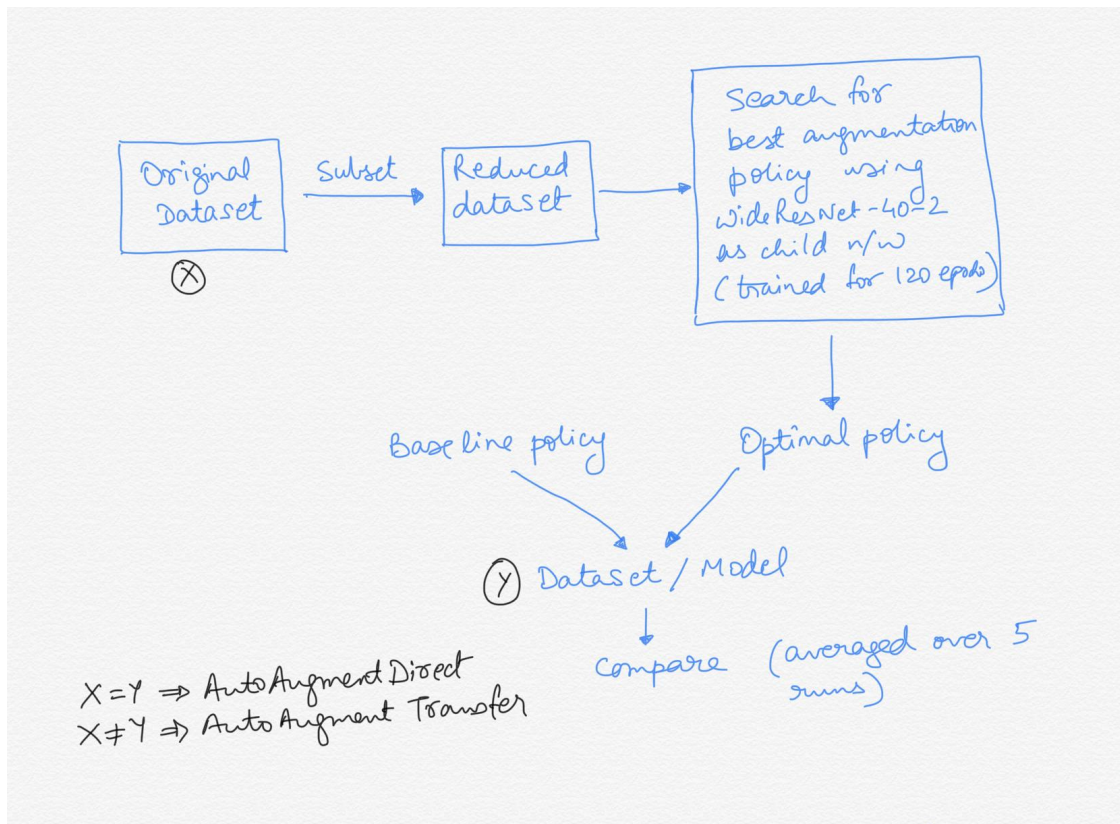
	Operation 1	Operation 2
Sub-policy 0	(ShearX,0.9,4)	(Invert,0.2,3)
Sub-policy 1	(ShearY,0.9,8)	(Invert,0.7,5)
Sub-policy 2	(Equalize,0.6,5)	(Solarize,0.6,6)
Sub-policy 3	(Invert,0.9,3)	(Equalize,0.6,3)
Sub-policy 4	(Equalize,0.6,1)	(Rotate,0.9,3)
Sub-policy 5	(ShearX,0.9,4)	(AutoContrast,0.8,3)
Sub-policy 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 8. 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)
Sub-policy 24	(Equalize,0.8,8)	(Equalize,0.6,3)

Table 9. AutoAugment policy found on reduced ImageNet.

Typical Experimental Setup



CIFAR

- CIFAR 10 - 50000 training examples
- Reduced CIFAR-10 - 4000 randomly chosen training examples
- **Baseline preprocessing**
 - Standardizing → horizontal flips with 50% probability → zero padding → random crops → cutout (16X16)
- **AutoAugment**
 - Baseline preprocessing → optimal autoaugment policy → cutout (16X16)
 - Cutout may potentially be applied twice on the same image

Results

- ⊛ Effect of AutoAugment reduces as the dataset size increases
- ⊛ Results on reduced datasets are comparable to their semi-supervised counterparts

Dataset	Model	Baseline	Cutout [12]	AutoAugment	
CIFAR-10	Wide-ResNet-28-10 [67]	3.9	3.1	2.6±0.1	
	generalizing across models } 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 SOTA	1.8±0.1
		PyramidNet+ShakeDrop [65]	2.7	2.3	1.5 ± 0.1 0.6% ↓
Reduced CIFAR-10 4000 labeled, 0 unlabeled	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 3.4% ↓ ⊛	
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 SOTA	10.7 ± 0.2 1.5% ↓	

New CIFAR-10 Test Set

- Motivated by the fact that **classifiers accuracy may be over-estimated**
 - Same test sets have been used to select these models for multiple years now
- Shake Shake + Cutout - degraded by 4.1%
- PyramidNet + Shake Drop - degraded by 4.6%
- PyramidNet + Shake Drop + AutoAugment - **degraded by only 2.9%**

SVHN

- 73257 core training examples
- 531131 additional training examples
- Test set - 26032 examples
- Reduced SVHN - 1000 examples sampled randomly from core training set
- Validation set - Last 7325 samples of training set
- **Baseline pre-processing**
 - Standardizing → Cutout (20X20) (cutout not used in reduced SVHN)
- **AutoAugment processing**
 - Baseline → AutoAugment Policy

Results

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

Results

- Optimal policy for SVHN
 - **Geometric transformations** are picked more often
 - **ShearX/Y** are most common
 - **Invert** is a commonly selected operation
- Optimal policy for CIFAR 10
 - **Color based transformations** (Equalize, AutoContrast, Color, Brightness)
 - **Geometric transformations** are rarely found
 - **Invert** is almost never applied

ImageNet

- Reduced ImageNet - 120 classes (randomly chosen), 6000 samples
- Child model - WideResNet 40-2 using cosine decay for
- **Baseline augmentation**
 - Inception style preprocessing (scaling pixel values to $[-1,1]$) → horizontal flips with 50% probability → random distortions of colors
- **AutoAugment augmentation**
 - Baseline → AutoAugment policy

ImageNet Results

- Best policies are similar to those found on CIFAR-10 focusing on **color transformations**
- One difference - **Rotate** is commonly seen in ImageNet policies
- Improvement **by just using 5000 images** for learning best policy

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

Isn't this computationally VERY expensive?

- Yes
- Thus, **transferring a data augmentation policy to other datasets / models can be a good alternative**, if works
- It would also establish that **AutoAugment doesn't overfit** to the dataset of interest
- Policy learned using reduced ImageNet
- Applied on five challenging datasets with image size similar to ImageNet
 - Oxford 102 flowers, Caltech 101, Oxford IIIT Pets, FGVC Aircraft, Stanford Cars
 - **Challenging** because relatively small number of training examples as compared to the classes

Results

- Optimal 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 [38], training with the ImageNet policy reduces test set error by 1.2% and 1.8%, respectively
- Transferring data augmentation policies offers an **alternative method for standard weight transfer learning**
- AutoAugment policies are **never found to hurt the performance of models even if they are learned on a different dataset** (closer the better, of course!)

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 SOTA *

Table 4. Test set Top-1 error rates (%) on FGVC datasets for Inception v4 models trained from scratch with and without AutoAugment-transfer. Lower rates are better. AutoAugment-transfer results use the policy found on ImageNet, Baseline models used Inception pre-processing. * Even though trained from scratch

Point to be noted

- Results can be further improved if **better search algorithms** are used
- For example:
 - “Simple random search provides a competitive approach to reinforcement learning”
 - “Regularized evolution for image classifier architecture search”

Comparison with the only other similar paper

- “Learning to Compose Domain-Specific Transformations for Data Augmentation” NIPS 2017
 - Generator learns to propose a sequence of transformations so that augmented images can fool a discriminator
- There: Tries to make sure that augmented images are similar to the current training images
- Here: Tries to optimize classification accuracy directly

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 (ResNet-56)	6.6	3.6	3.0

Ablation Experiments and Results

- **More number of sub-policies** => NN is trained on same points with greater diversity of augmentation => increased generalization accuracy
-
- **Randomizing probabilities and magnitudes of operations** => worse results => right probability and magnitude were actually being learnt
-
- **Randomizing operations, probabilities and magnitudes** => only slightly worse => auto augment with random search also yields good results

Thank You

AutoAugment: Learning Augmentation Strategies from Data

vkaushal@cse.iitb.ac.in | www.vishalkaushal.in