

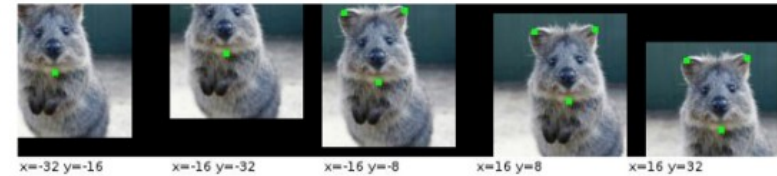
# AutoAugment: Learning Augmentation Strategies from Data

Ahmed Agha  
AutoML Seminar  
22.05.2019

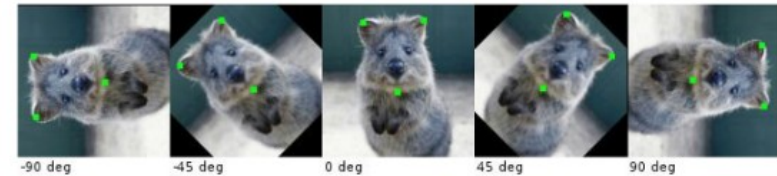
# Data Augmentation

- Effective method to increase diversity and amount of data
- Helps network to learn about invariances in the data domain

Affine: Translate



Affine: Rotate



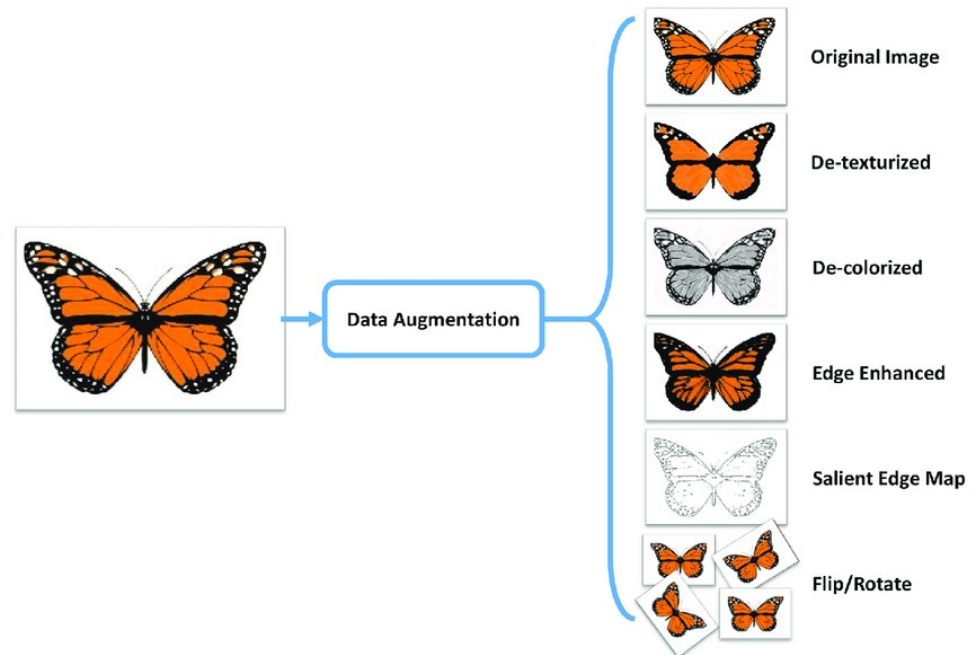
Affine: Shear



# Data Augmentation

- Especially important in small or unbalanced datasets



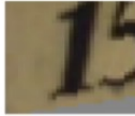











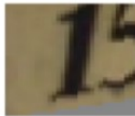



- Data augmentation operations are dataset dependant



# AutoAugment

# Search Space

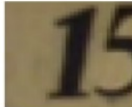

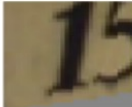
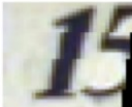

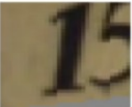
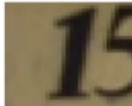

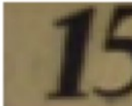
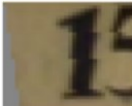
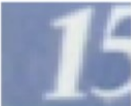

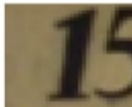

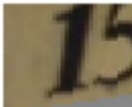
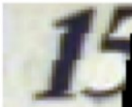
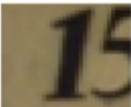
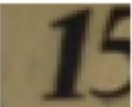
- Search space contains different augmentation policies
- Policy contains of multiple sub-policies
- Subpolicy consists of:
  - Two image processing operations
  - Probability of operations
  - Magnitude

	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

# Search Space

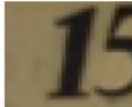

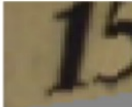
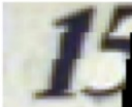

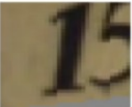
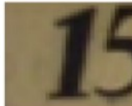

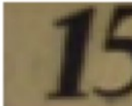
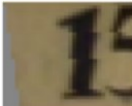
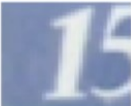

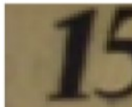

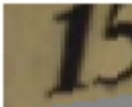
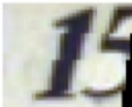
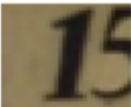
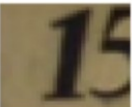
- For every image in mini batch a sub-policy is chosen

- Same sub-policy may produce two different results

	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

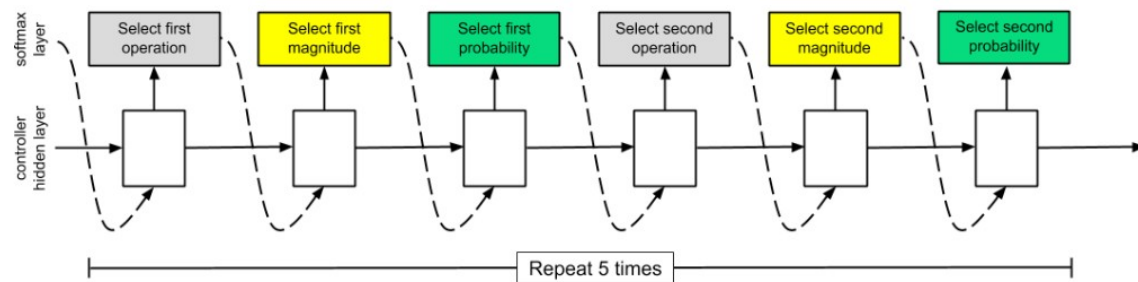
# Search Space

- Discrete search space:
  - 16 operations
  - Probability discretized into 10 Values
  - Magnitude value discretized into 11 values
- Search space of  $(16 \times 10 \times 11)^{10}$

	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

# Search algorithm

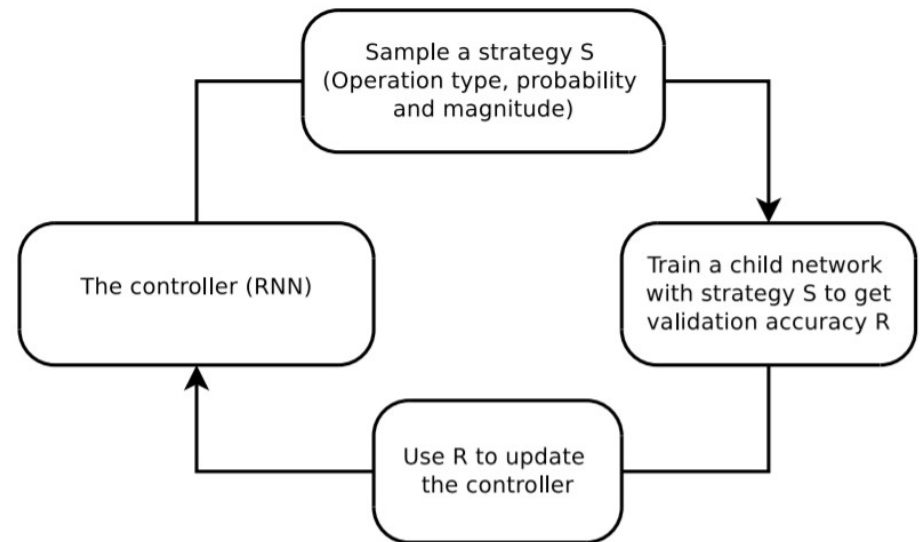
- The search algorithm consists of a controller, which is a recurrent neural network
- Controller RNN trained using Proximal Policy Optimization algorithm





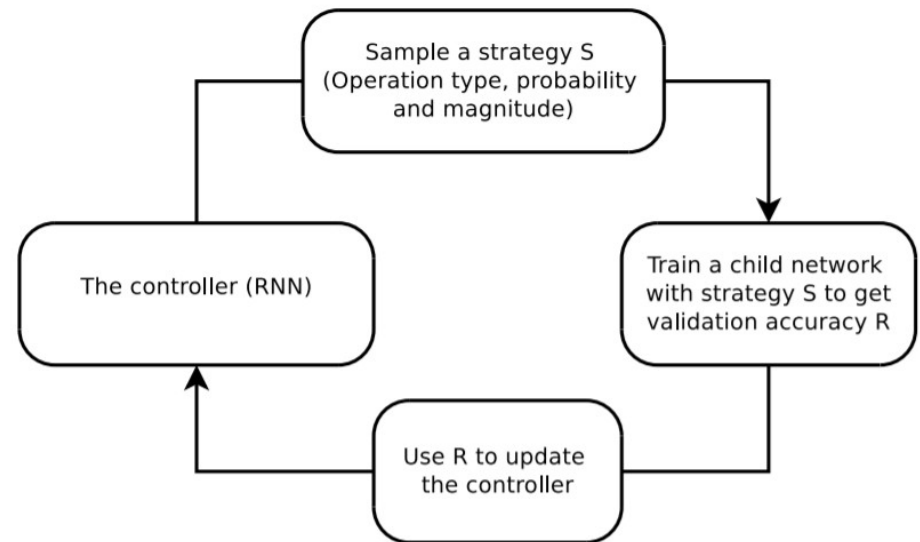
# Training

- A child Network with fixed architecture is used for every policy
- A child neural network is trained for every policy
- Child model evaluated on validation set returning accuracy R



# Training

- Training the child network with augmentation policy  $S$  Returning reward  $R$
- $R$  is fed to the proximal policy optimization
- Top 5 policies are concatenated to form one policy with 25 subpolicies



# AutoAugment in different datasets

- Geometry based policies are learned



















SVHN

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

# AutoAugment in different datasets

- Color based policies are learned

ImageNet

	Original	Sub-policy 1	Sub-policy 2	Sub-policy 3	Sub-policy 4	Sub-policy 5
Batch 1						
Batch 2						
Batch 3						
		Equalize, 0.4, 4 Rotate, 0.8, 8	Solarize, 0.6, 3 Equalize, 0.6, 7	Posterize, 0.8, 5 Equalize, 1.0, 2	Rotate, 0.2, 3 Solarize, 0.6, 8	Equalize, 0.6, 8 Posterize, 0.4, 6

# Results

- CIFAR 10 : Decreasing state of the art error rate by 0.6%

- SVHN : Decreasing state of the art error rate by 0.2%

Dataset	Model	Baseline	Cutout [12]	AutoAugment
<b>CIFAR-10</b>	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	<b>1.5 ± 0.1</b>
<b>Reduced CIFAR-10</b>	Wide-ResNet-28-10 [67]	18.8	16.5	14.1±0.3
	Shake-Shake (26 2x96d) [17]	17.1	13.4	<b>10.0 ± 0.2</b>
<b>CIFAR-100</b>	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	<b>10.7 ± 0.2</b>
<b>SVHN</b>	Wide-ResNet-28-10 [67]	1.5	1.3	1.1
	Shake-Shake (26 2x96d) [17]	1.4	1.2	<b>1.0</b>
<b>Reduced SVHN</b>	Wide-ResNet-28-10 [67]	13.2	32.5	8.2
	Shake-Shake (26 2x96d) [17]	12.3	24.2	<b>5.9</b>

# Experiments

# Policy transferability

- FGFV datasets small number of samples  
high number of classes
- Using AutoAugment could be resource intensive
- Transfer policies from one dataset to another could save lots of compute

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

# AutoAugment vs. Randomly chosen policies

- Randomizing probabilities and magnitudes led to 0.4% higher error rate on CIFAR10
- Using random policies was slightly worse than using random probabilities and magnitudes
- Using policies in search space superior to using baseline augmentations
- Learned probabilities and magnitudes provide better results

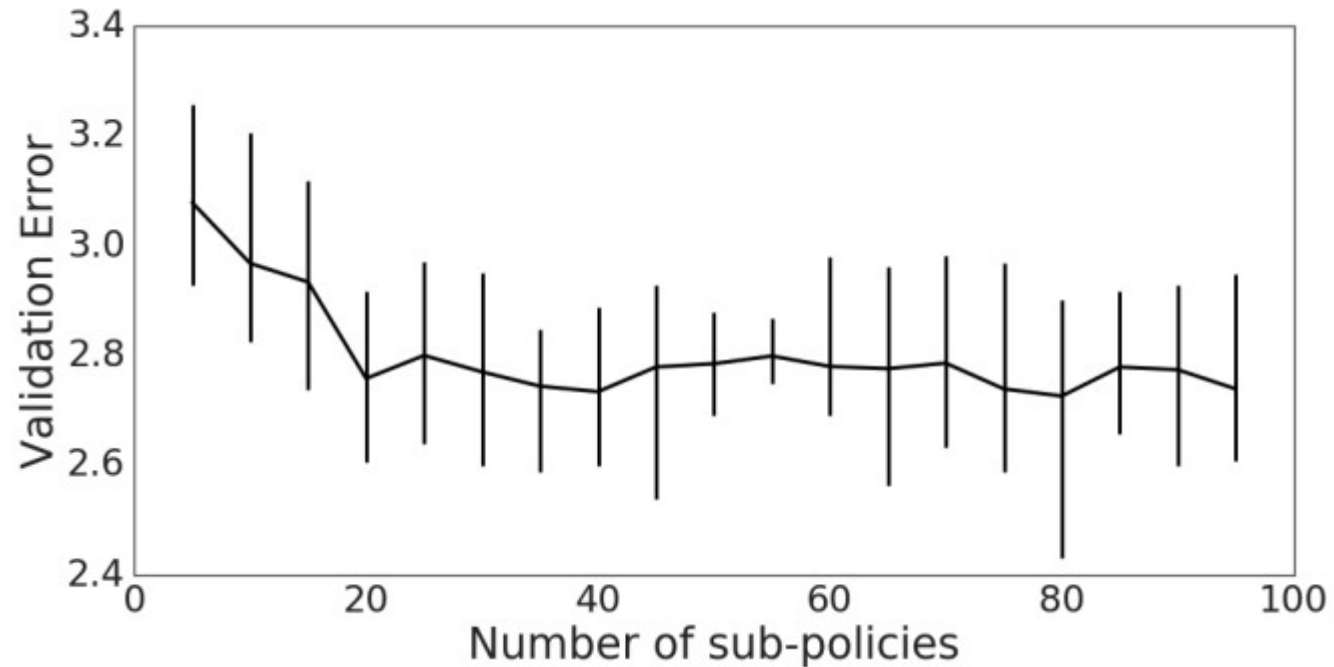


# Influence of number of training steps

- Stochastic application of sub-policies during training
- Each transformation has its own probability
- A certain number of epochs per sub-policy for AutoAugment to be effective.

# Changing number of policies

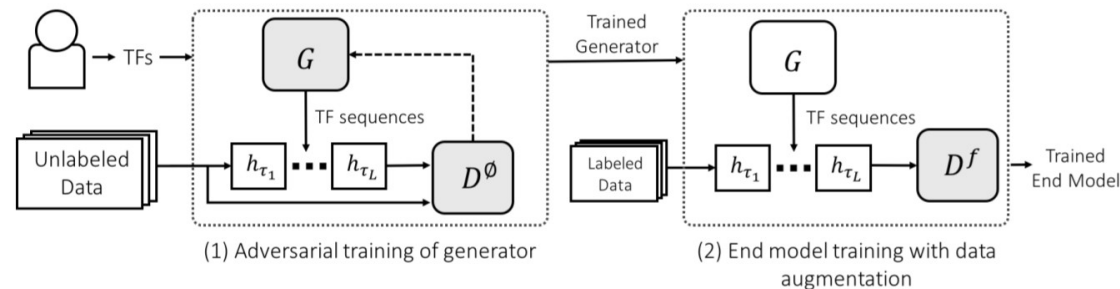
- Increasing number of subpolicies leads to improvements till up to 20 subpolicies



# Data augmentation using GANS

- A generator learns to propose augmentation policy (a sequence of image processing operations) to fool the discriminator

- The method tries to make sure the augmented images are similar to the current training images.



# Comparison with data augmentation using GANs

- AutoAugment tries to optimize the validation accuracy directly leading to bigger improvement

Method	Baseline	Augmented	Improvement $\Delta$
LSTM [47]	7.7	6.0	1.6
MF [47]	7.7	5.6	2.1
AutoAugment (ResNet-32)	7.7	4.5	<b>3.2</b>
AutoAugment (ResNet-56)	6.6	3.6	<b>3.0</b>

Thanks for your Attention !