RandAugment

Practical automated data augmentation with a reduced search space

Cubuk, Zoph, Shlens, Le

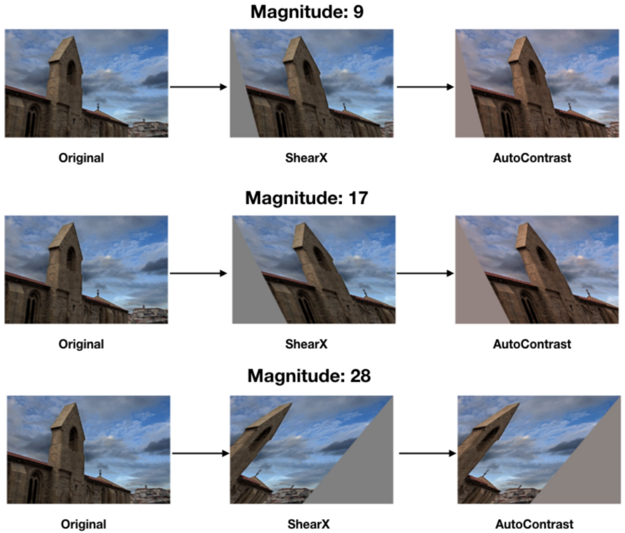
Googe Research, Brain Team

Presented by William Guimont-Martin

2024-04-03

Data Augmentations

- Increase the diversity of training data
- Act as regularization
- Require expertise and manual work to design
- Depend on the network and dataset



https://pytorch.org/vision/stable/_images/sphx_glr_plot_transforms_illustrations_thumb.png

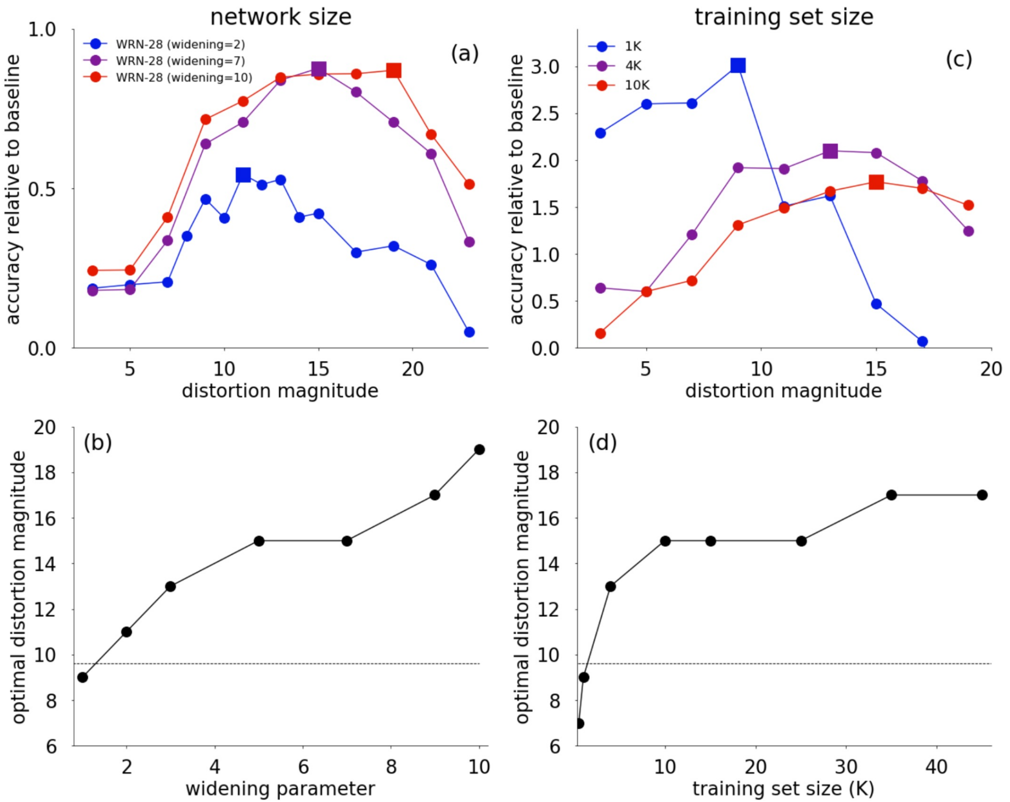
Automatic Data Augmentation

- Learning data augmentation policies
    - AutoAugment (Cubuk et al., 2019)
    - Population Based Augmentation (Ho et al., 2019)
- High computational cost and complicated procedures
- Often use proxy tasks

	search space	CIFAR-10 PyramidNet	SVHN WRN	ImageNet ResNet	ImageNet E. Net-B7
Baseline	0	97.3	98.5	76.3	84.0
AA	10^{32}	98.5	98.9	77.6	84.4
Fast AA	10^{32}	98.3	98.8	77.6	-
PBA	10^{61}	98.5	98.9	-	-
RA (ours)	10^{2}	98.5	99.0	77.6	85.0

Challenging the proxy task assumption

Network size and dataset subsets



RandAugment

- Does not rely on proxy tasks
- Small search space
- No complicated optimization procedure

	search space	CIFAR-10 PyramidNet	SVHN WRN	ImageNet ResNet	ImageNet E. Net-B7
Baseline	0	97.3	98.5	76.3	84.0
AA	10^{32}	98.5	98.9	77.6	84.4
Fast AA	10^{32}	98.3	98.8	77.6	-
PBA	10^{61}	98.5	98.9	-	-
RA (ours)	10^{2}	98.5	99.0	77.6	85.0

RandAugment

```python
transforms = [
    'Identity', 'AutoContrast', 'Equalize',
    'Rotate', 'Solarize', 'Color', 'Posterize',
    'Contrast', 'Brightness', 'Sharpness',
    'ShearX', 'ShearY', 'TranslateX', 'TranslateY']


def randaugment(N, M):
    """Generate a set of distortions.

    Args:
        N: Number of augmentation transformations to
           apply sequentially.
        M: Magnitude for all the transformations.
    """

    sampled_ops = np.random.choice(transforms, N)
    return [(op, M) for op in sampled_ops]
```

K Augmentations

Two hyperparameters

Sample uniformly with replacement (K^N policies)

Same magnitude

Results

Comparison with other approaches

	baseline	PBA	Fast AA	AA	RA
CIFAR-10					
Wide-ResNet-28-2	94.9	-	-	**95.9**	95.8
Wide-ResNet-28-10	96.1	**97.4**	97.3	**97.4**	97.3
Shake-Shake	97.1	**98.0**	**98.0**	**98.0**	**98.0**
PyramidNet	97.3	**98.5**	98.3	**98.5**	**98.5**
CIFAR-100					
Wide-ResNet-28-2	75.4	-	-	**78.5**	78.3
Wide-ResNet-28-10	81.2	**83.3**	82.7	82.9	**83.3**
SVHN (core set)					
Wide-ResNet-28-2	96.7	-	-	98.0	**98.3**
Wide-ResNet-28-10	96.9	-	-	98.1	**98.3**
SVHN					
Wide-ResNet-28-2	98.2	-	-	**98.7**	**98.7**
Wide-ResNet-28-10	98.5	98.9	98.8	98.9	**99.0**

Results

Top-1 and Top-5 on ImageNet

- AutoAugment does not help for larger models
  - Policy found on 10% of the dataset
  - Proxy not representative of the full dataset
- SOTA accuracy in top-1

	baseline	Fast AA	AA	RA
ResNet-50	76.3 / 93.1	**77.6** / 93.7	**77.6 / 93.8**	**77.6 / 93.8**
EfficientNet-B5	83.2 / 96.7	-	83.3 / 96.7	**83.9 / 96.8**
EfficientNet-B7	84.0 / 96.9	-	84.4 / 97.1	**85.0 / 97.2**

Results

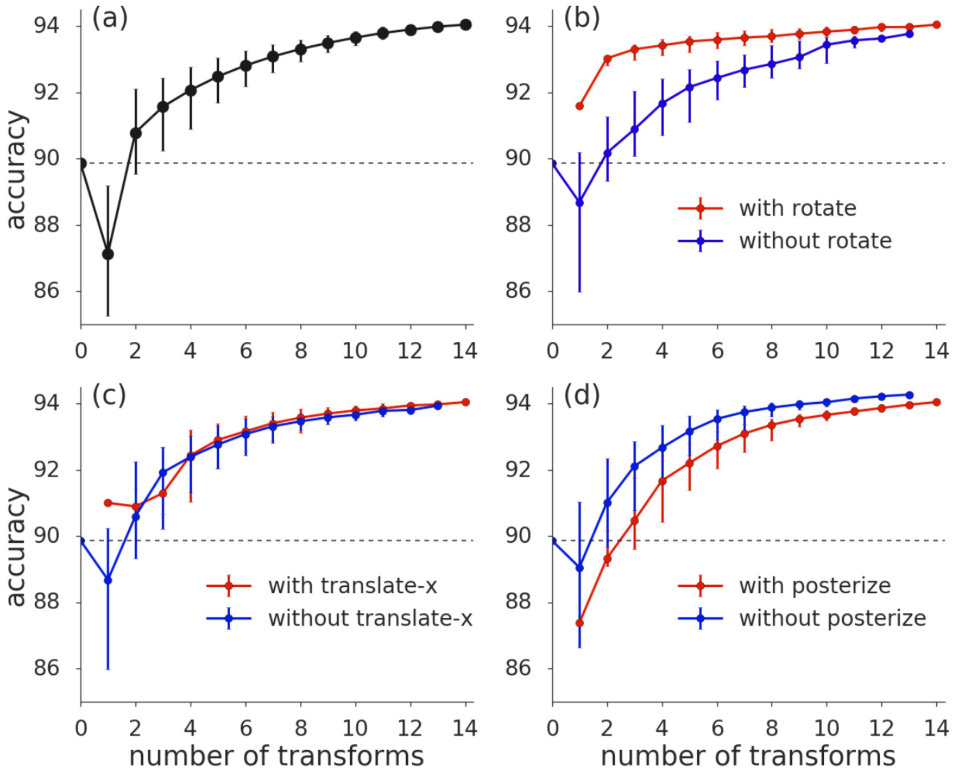
COCO dataset for object detection

- AutoAugment
  - Uses specialized augmentations for bounding boxes
  - Took 15k GPU hours
- RandAugment is competitive without it
  - Searched over 6 values (N = 1, M = {4, 5, 6, 7, 8, 9})

model	augmentation	mAP	search space
ResNet-101	Baseline	38.8	0
	AutoAugment	**40.4**	10^{34}
	RandAugment	40.1	10^{2}
ResNet-200	Baseline	39.9	0
	AutoAugment	**42.1**	10^{34}
	RandAugment	41.9	10^{2}

Results

Impact of included transformations



Results

Impact of included transformations



transformation	Δ (%)	transformation	Δ (%)
rotate	**1.3**	shear-x	0.9
shear-y	0.9	translate-y	0.4
translate-x	0.4	autoContrast	0.1
sharpness	0.1	identity	0.1
contrast	0.0	color	0.0
brightness	0.0	equalize	-0.0
solarize	-0.1	posterize	-0.3

https://pytorch.org/vision/main/auto_examples/transforms/plot_transforms_illustrations.html

Results

Learning the probability per augmentation

- RandAugment samples uniformly N transforms
- Can also set a different probability per transform
- Most transformations are differentiable
    - backprop to learn the probability per transformation
- Learn to maximize the results on validation images

Results

Learning the probability per augmentation

- Improves performance
- Higher computational demands
  - Requires to apply all K transformations N times to each image

	baseline	AA	RA	+ 1^{st}
Reduced CIFAR-10				
Wide-ResNet-28-2	82.0	**85.6**	85.3	85.5
Wide-ResNet-28-10	83.5	**87.7**	86.8	87.4
CIFAR-10				
Wide-ResNet-28-2	94.9	95.9	95.8	**96.1**
Wide-ResNet-28-10	96.1	**97.4**	97.3	**97.4**

Results

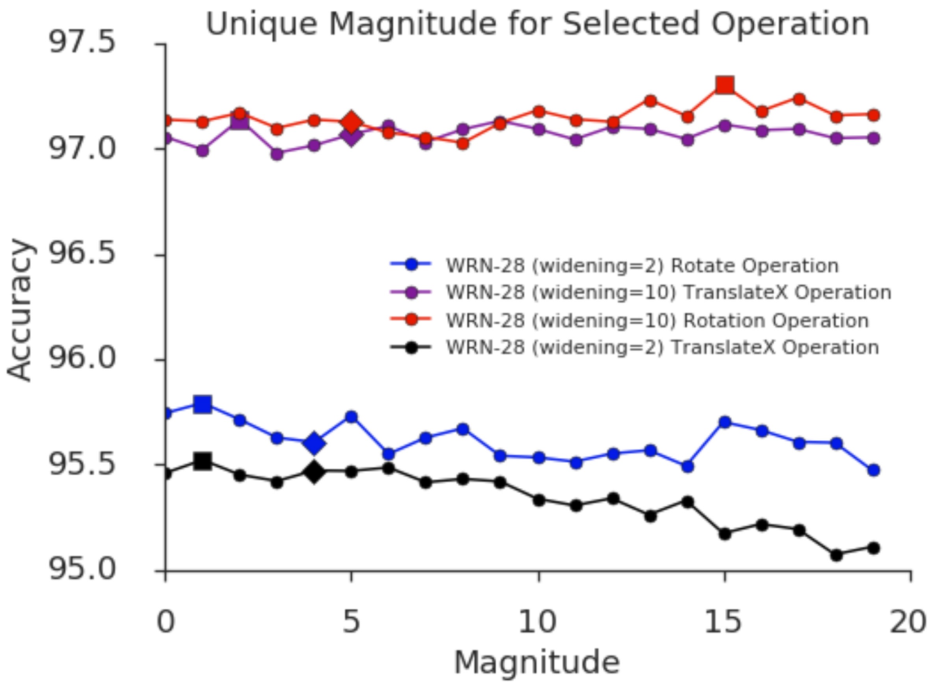
Magnitude methods

- Ways to choose the magnitude
- Similar performances

Magnitude Method	Accuracy
Random Magnitude	97.3
Constant Magnitude	97.2
Linearly Increasing Magnitude	97.2
Random Magnitude with Increasing Upper Bound	97.3

Results

Individual magnitude



RangAugment

```python
transforms = [
    'Identity', 'AutoContrast', 'Equalize',
    'Rotate', 'Solarize', 'Color', 'Posterize',
    'Contrast', 'Brightness', 'Sharpness',
    'ShearX', 'ShearY', 'TranslateX', 'TranslateY']


def randaugment(N, M):
    """Generate a set of distortions.

    Args:
        N: Number of augmentation transformations to
           apply sequentially.
        M: Magnitude for all the transformations.
    """

    sampled_ops = np.random.choice(transforms, N)
    return [(op, M) for op in sampled_ops]
```

Questions?