RandAugment

After AutoAugment and Fast AutoAugment

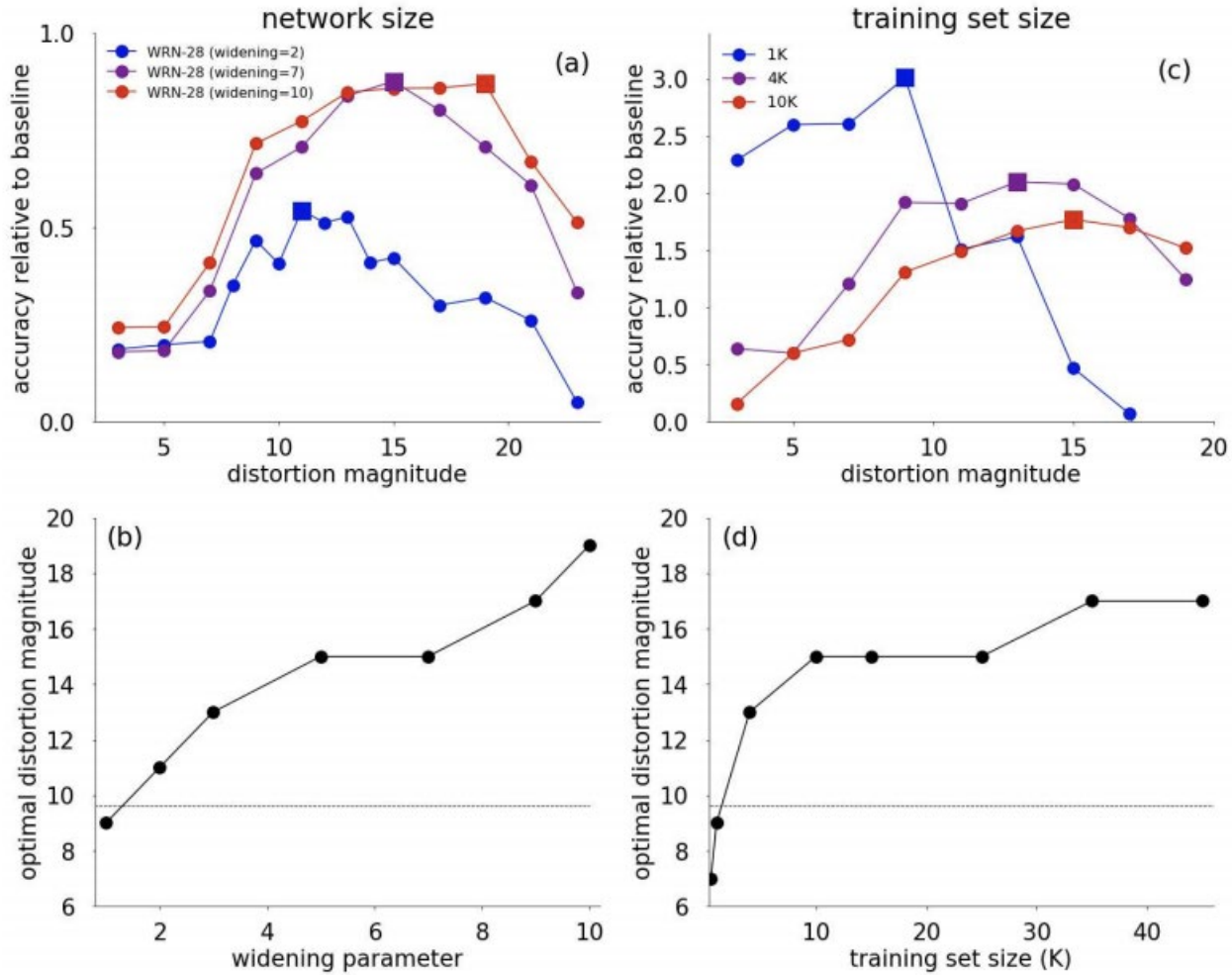
Algorithm

Grid Search(N, M)

```
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]
```

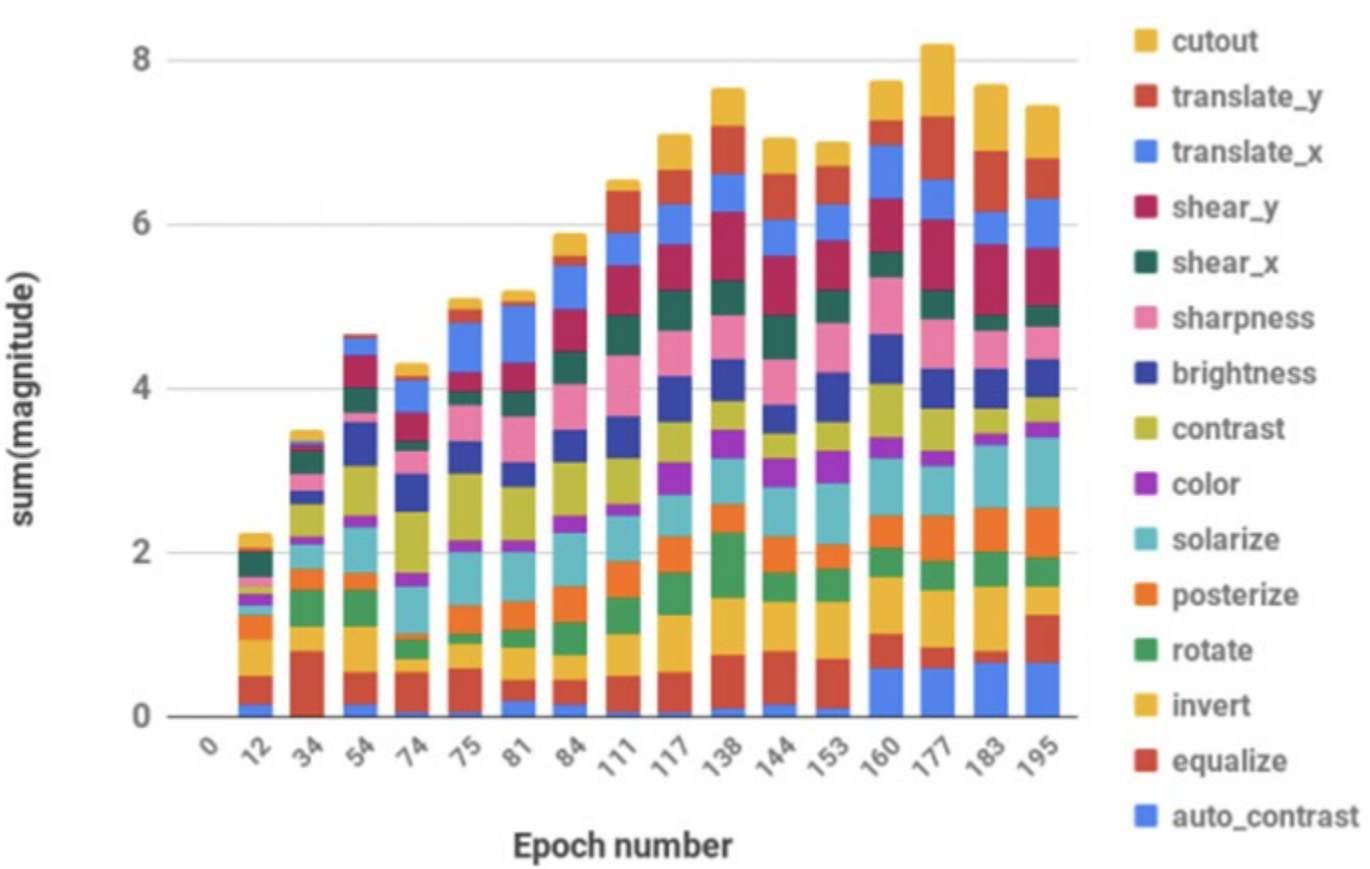
Figure 3. Python code for RandAugment based on numpy.

M: constant magnitude for all transformations



For width of model and training data size, there seems to be a relationship with optimal unified transformation magnitude

M: constant magnitude for all transformations



From PBA: the magnitudes for the transformations tend to be at similar levels

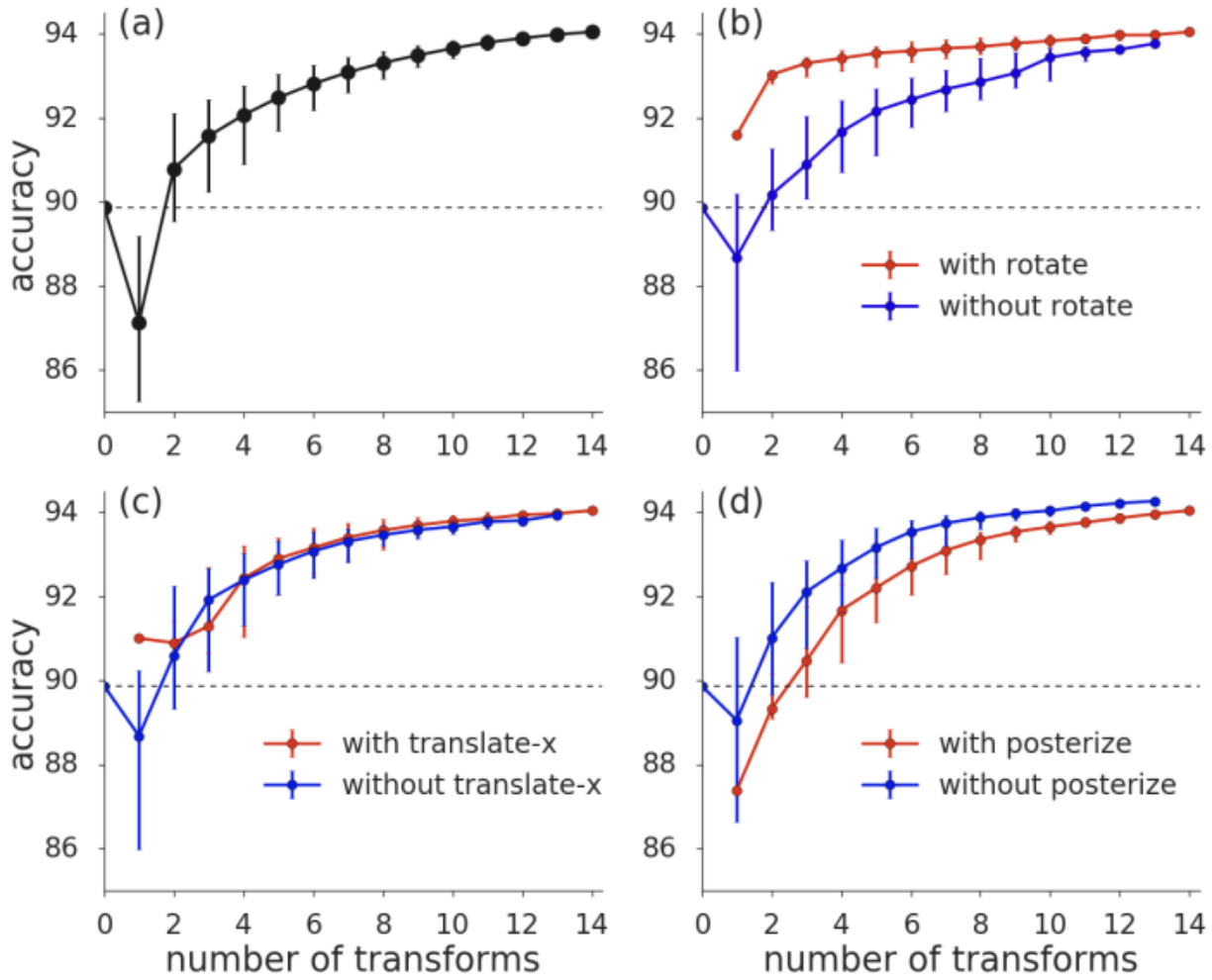
M: constant magnitude for all transformations

	baseline	AA	RA	+ 1st
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**

1st-order density matching operation: fine tuning magnitudes for individual transformations

Can be expensive as K transformations needs to be applied N times to each image independently

N: number of transformations



Experiment excluded flip/pad-n-crop/cutout which is default augmentation for better comparison

Including all transformations show good results

However, could think of finding and removing less helpful operations

N: number of 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

However, could think of finding and removing less helpful operations

Also, would rotate be as helpful if the default augmentations were also included here?

But could think of removing the last 6 for increased efficiency – if under computational restrictions