Supplementary Material for Automatic Augmentation Policies for Self-Supervised Learning

Colorado J Reed*†, Sean Metzger*‡, Aravind Srinivas†, Trevor Darrell†, Kurt Keutzer†

†BAIR, Department of Computer Science, UC Berkeley

‡Graduate Group in Bioengineering (Berkeley/UCSF), Weill Neurosciences Institute & UCSF Neurological Surgery

Notation	Definition
ρ	Spearman rank correlation
θ_{moco}	MoCo encoder
ϕ_{ss}	Linear self-supervised evaluation head
D	Dataset
K	Number of folds used for training SelfAugment
T	Number of augmentations to apply to each image at each iteration
B	Number of policies to consider during Bayesian optimization search
P	Top number of policies to select from each fold in SelfAugment
$\mathbb{O}$	The set of candidate image transformations for an augmentation policy
$\mathcal{L}$	A loss function
$\mathcal{O}$	An image transformations in $\mathbb{O}$
$\mathcal{S}$	Set of sub-policies
$\mathcal{M}$	A model (e.g. a neural network)
τ	$\tau \in \mathcal{S}$ is the sequential application of N_τ consecutive transformations
N_τ	The number of consecutive transformations to apply in a sub-policy
λ	The magnitude of an image transformation
FAA	Fast AutoAugment, from [1]
InfoNCE	See Eq. 1
MoCo	Momentum Contrast, from [2]

A. Notation and definitions

B. Augmentation transformation details

Following [1, 4], we define the set of transformations used for SelfAugment and RandAugment, $\mathbb{O}$, as the PIL-based image transformations in Table 1. Each transformation has a minimum and maximum magnitude, λ, where for RandAugment, the entire range is discretized over 30 integers. See [4, 3] for further details and descriptions of each transformation.

C. Additional training and experiment details

CIFAR-10, SVHN, ImageNet: Table 4 lists the training and experimental parameters for CIFAR-10, SVHN, and ImageNet. The training parameters were taken from [2] and adjusted for 4 GPUs, i.e. the learning rate and batch size were scaled by 0.5 since [2] experiments were conducted on 8 GPUs. Consult [2] and [5] for MoCo parameter information. For the linear classifier, we used 50 training iterations where the learning rate was 10× reduced at 30 and 40 epochs for ImageNet and 20 and 30 epochs for CIFAR-10 and SVHN. In [2], the linear layer was trained over 150 iterations with a reduction at 80 and 100 iterations. In early experiments, we found the performance converged much earlier, and to reduce computational resources, we reduced all linear training iterations.

VOC07 Following [2], we transfer the ImageNet ResNet-50 weights to perform object detection using a Faster R-CNN R50-C4, with BN tuned. We fine-tuned all layers end-to-end. The image scale is [480, 800] pixels during training and 800 at inference. Training was on the VOC `trainval07+12` set and evaluation was on the `test2007` set. The R50-C4 backbones is similar to those available in Detectron2[1], where the backbone stops at the conv4 stage, and the box prediction head consists of the conv5 stage followed by a BN layer. Table 3 displays the AP/AP_{50}/AP_{75} breakdown for three fine-tunings.

COCO2014/Places205: Following [6], we train Linear SVMs on frozen feature representations. We train a linear SVM per class for (80 for COCO2014, 205 for Places205) for the cost values $C \in 2^{[-19,-4]} \cup 10^{[-7,2]}$ We used 3-fold cross-validation to select the cost parameter per class and then further calculate the mean average precision. The features are first normalized in a (N, 9k)

*equal contribution; correspondence to `cjrd@cs.berkeley.edu`

[1] https://github.com/facebookresearch/detectron2

Table 1: PIL image transformations used for SelfAugment and RandAugment. The min and max magnitude values are taken and from [3]

Name	Description	Min ($\lambda = 0.0$)	Max ($\lambda = 1.0$)
ShearX	shear the image along the horizontal axis with magnitude rate	-0.3	0.3
ShearY	shear the image along the vertical axis with magnitude rate	-0.3	0.3
TranslateX	translate the image in the horizontal direction by magnitude percentage	-0.45	0.45
TranslateY	translate the image in the vertical direction by magnitude percentage	-0.45	0.45
Rotate	rotate the image by magnitude degrees	-30	30
AutoContrast	adjust contrast so darkest pixel is black and lightest is white	0	1
Invert	invert the pixels of the image	0	1
Solarize	invert the pixels above a magnitude threshold	0	256
Posterize	reduce the number of bits for each color to magnitude	4	8
Contrast	adjust image contrast, where magnitude 0 is grey and magnitude 1 is original image	0.1	1.9
Color	adjust color of image such that magnitude 0 is black and white and magnitude 1 is original image	0.1	1.9
Brightness	brightness adjustment such that magnitude 0 is black image and 1 is original image	0.1	1.9
Sharpness	magnitude 0 is a blurred image and 1 is original image	0.1	1.9
Cutout	cutout a random square from the image with side length equal to the magnitude percentage of pixels	0	0.2
Equalize	equalize the image histogram	0	1

matrix, where N is number of samples in data and 9k is the resized feature dimension, to have norm=1 along each feature dimension. This normalization step is applied on evaluation data too. We use the following hyperparameter setting for training using `LinearSVC` sklearn class: `class_weight` ratio of 2:1 for positive:negative classes, `penalty`=l2, `loss`=squared_hinge, `tol`=1e-4, `dual`=True and `max_iter`=2000. Table 2 displays the Places205 results across five linears SVM trainings for all k values.

Table 2: Transfer results: Places205 top-1 accuracy for frozen ResNet50 encoder after pre-training, with a linear SVM trained for scene classification, with $k = \{1, 4, 8, 16, 32, 64\}$ labeled training images for each class, averaged over five SVM trainings, where the errors indicate the standard deviation (see text for details).

Labeled samples	1	4	8	16	32	64
Base Aug	$1.35 \pm .03$	2.8 ± 0.05	$4.18 \pm .11$	$5.88 \pm .12$	$8.04 \pm .16$	$10.13 \pm .15$
SelfRandAug	$6.17 \pm .05$	$13.01 \pm .15$	$18.36 \pm .12$	$23.13 \pm .11$	$27.18 \pm .11$	$30.89 \pm .09$
SelfAug (min rot)	$3.58 \pm .06$	7.85 ± 0.09	$11.77 \pm .10$	$15.60 \pm .17$	$19.66 \pm .10$	$23.26 \pm .05$
SelfAug (min Info)	$2.46 \pm .04$	5.36 ± 0.07	$8.11 \pm .11$	$11.04 \pm .07$	$14.52 \pm .09$	$17.75 \pm .29$
SelfAug (max Info)	$5.87 \pm .06$	$12.73 \pm .14$	$17.88 \pm .16$	$22.64 \pm .21$	$26.95 \pm .12$	$30.56 \pm .17$
SelfAug (minimax)	$6.73 \pm .08$	$14.51 \pm .20$	$19.89 \pm .18$	$24.72 \pm .10$	$28.77 \pm .13$	$32.35 \pm .09$
MoCoV2	$6.65 \pm .11$	$14.06 \pm .13$	$19.59 \pm .14$	$24.57 \pm .13$	$28.56 \pm .08$	$32.24 \pm .10$

Table 3: Transfer Result: For `VOC07 test`, this table reports the average AP50 and COCO-style AP/AP_{75} over three runs of fine-tuning the ResNet50 encoder from ImageNet pre-training, where the errors indicate the standard deviation (see text for details).

Evaluation	AP	AP_{50}	AP_{75}
Base Aug	47.08 ± 0.17	74.80 ± 0.17	50.26 ± 0.24
SelfRandAug	53.06 ± 0.21	80.09 ± 0.14	57.58 ± 0.29
SelfAug (min rot)	50.13 ± 0.20	77.66 ± 0.18	53.94 ± 0.21
SelfAug (min Info)	49.18 ± 0.21	76.31 ± 0.17	53.17 ± 0.18
SelfAug (max Info)	52.66 ± 0.18	79.69 ± 0.15	57.33 ± 0.25
SelfAug (minimax)	52.72 ± 0.22	79.79 ± 0.21	57.62 ± 0.27
MoCoV2	53.94 ± 0.23	80.64 ± 0.18	59.34 ± 0.31

Table 4: Detailed training parameters for CIFAR-10, SVHN, and ImageNet experiments carried out in this paper.

-MoCo Params	**CIFAR 10/SVHN**	**ImageNet**
Batch Size	512	128
moco-dim	128	128
moco-k	65536	65536
moco-m	0.999	0.99
moco-t	0.2	0.2
num-gpus	4	4
lr	0.4	0.015
schedule	120, 160	60, 80
momentum	0.9	0.9
weight decay	1e-4	1e-4
Classifier Params		
lr	15	30
batch size	256	256
momentum	0.9	0.9
weight decay	0.0	0.0
schedule	20, 30	30, 40
epochs	50	50

D. Correlation study

In this section, we detail the full experimental setup, investigation, and results from our study of the correlation between rotation prediction performance with a linear network and supervised downstream task performance. We also include additional preliminary and ablation studies.

D.1. Correlation study details

We study RandAugment, SelfAugment, and MoCoV2 augmentation policies, and then evaluate the performance using self-supervised rotation prediction with a linear layer. For CIFAR-10 and SVHN, we evaluate:

- A *base augmentation* of random left-right horizontal flips with $p = 0.5$ of being applied and random resize and crop transformation with magnitude range $(0.2, 1.0)$, trained for 750 epochs.
- On top of the base augmentation[2], we performed a RandAugment grid search with magnitude $\lambda = \{4,5,7,9,11\}$ and number of transformations applied $N_\tau = \{1,2,3\}$, evaluated at $\{100,500\}$ epochs. We initially included $N_\tau = 4$ in the grid search, but this always led to a degenerate solution, whereby the training loss would not minimize the objective function and the evaluation would yield a chance result (25% self-supervised rotation prediction and 10% supervised classification for CIFAR-10).
- Following [4], we also experimented with scaling the magnitude parameter λ from $[4,11]$ linearly throughout the training. We did this for each of the three N_τ values and evaluated the results at 500 epochs.
- As part of the SelfAugment algorithm, we trained an augmentation policy consisting of each of the fifteen individual transformations in RandAugment. In addition, we include the random resize crop transformation, as it was shown to be the best performing individual transformation in [7]. The magnitude for each transformation was stochastically selected from its magnitude range defined in §B at each iteration. Each of these 16 transformations were evaluated after a short training cycle of 100 epochs. Each of these transformations were applied on top of a random left-right horizontal flip applied with $p = 0.5$.
- Using the rotation prediction results from each of the individual transformation policies, we selected the top $K_T = \{3,6,9\}$ augmentations, and trained RandAugment using only these transformations. We performed this training for $\lambda = \{4,7\}$ with $N_\tau = 2$, evaluated at 500 epochs.
- We further include the four SelfAugment policies from each of its loss functions, evaluated at 750 epochs.

In total, this yields 61 different models for each of CIFAR-10/SVHN. For ImageNet we evaluate:

- A *base augmentation* of random left-right horizontal flips with $p = 0.5$ of being applied and random resize and crop transformation with magnitude range $(0.2, 1.0)$, trained for 100 epochs.
- On top of the base augmentation[3], we performed a RandAugment grid search with magnitude $\lambda = \{5,7,9,11,13\}$ and number of transformations applied $N_\tau = 2$, evaluated at $\{20,60,100\}$ epochs.
- Following [4], we also experimented with scaling the magnitude parameter λ from $[5,13]$ linearly throughout the training. We did this for $N_\tau = 2$ and evaluated the results at 100 epochs.
- As part of the SelfAugment algorithm, we trained an augmentation policy consisting of each of the fifteen individual transformations in RandAugment (transformations listed in §). In addition, we include the random resize crop transformation, as it was shown to be the best performing individual transformation in [7]. The magnitude for each transformation was stochastically selected from its magnitude range defined in §B at each iteration. Each of these 16 transformations were evaluated after a short training cycle of 100 epochs. Each of these transformations were applied on top of a random left-right horizontal flip applied with $p = 0.5$.
- We further included the five SelfAugment policies from each of its loss functions, evaluated at 100 epochs.
- To further compare with state-of-the-art models, we compare with the MoCoV2 augmentation policy evaluated at $100, 200$ epochs

In total, this yields 43 different models for ImageNet.

Supplementing the main correlation results shown in the experiments section, Figure 1 shows the the ImageNet rotation prediction correlations with transfer performance to VOC07 for all AP, AP_{50}, and AP_{75} evaluations. Figure 2 shows the ImageNet rotation prediction correlations with transfer performance to Places205 few label scene classification using $k = \{1,4,8,16,32,64\}$ labels per scene class. For both VOC07 and Places205, the Spearman rank correlation with rotation prediction is indicated with ρ, while the rank correlation with the supervised ImageNet classification performance is indicated with ρ_s. Across all evaluation metrics, the rotation correlation is higher than the supervised correlation.

[2]Using this base augmentation systematically improved all RandAugment results over not using a base augmentation, both in terms of its rotation prediction performance and supervised linear classification performance.

[3]Using this base augmentation systematically improved all RandAugment results over not using a base augmentation, both in terms of its rotation prediction performance and supervised linear classification performance.

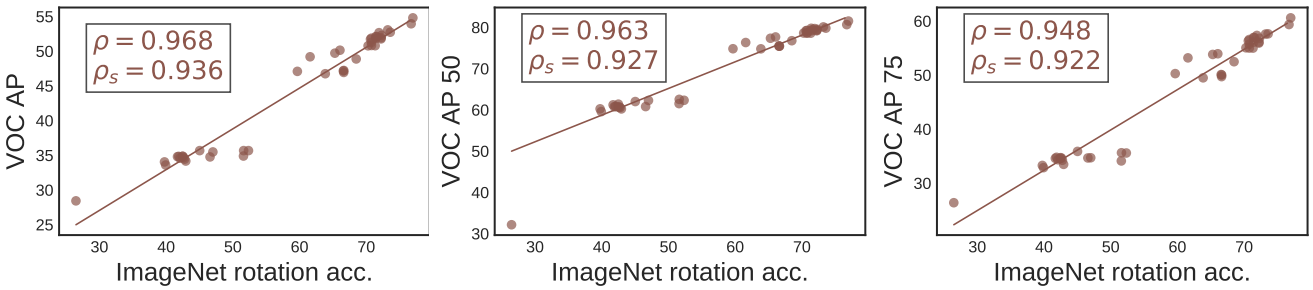

Figure 1: The ImageNet rotation prediction correlations with transfer performance to VOC07 for all AP, AP_{50}, and AP_{75} evaluations. The Spearman rank correlation with rotation prediction is indicated with ρ, while the rank correlation with the supervised ImageNet classification performance is indicated with ρ_s.

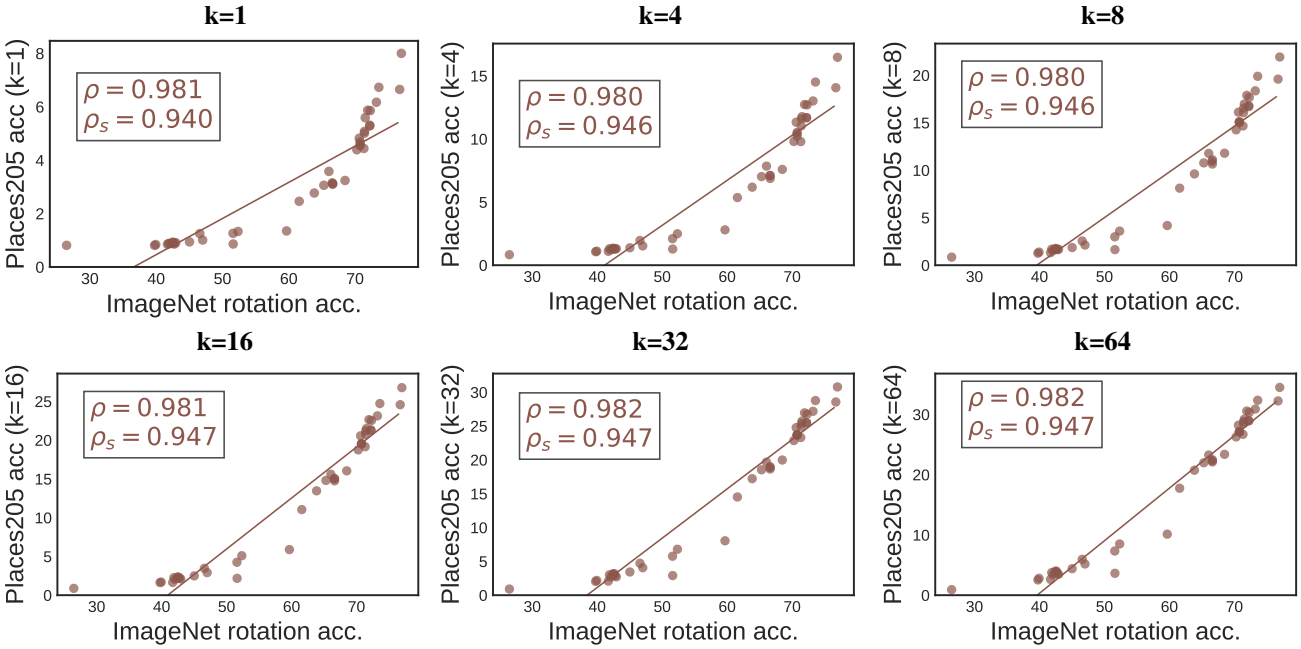

Figure 2: The ImageNet rotation prediction correlations with transfer performance to Places205 few label scene classification using $k = \{1, 4, 8, 16, 32, 64\}$ labels per scene class. The Spearman rank correlation with rotation prediction is indicated with ρ, while the rank correlation with the supervised ImageNet classification performance is indicated with ρ_s.

D.2. Correlation with an MLP rotation prediction

When following the same evaluation protocol, except using a 2 layer MLP instead of a linear layer for rotation prediction, we find that the CIFAR-10 Spearman rank correlation with the supervised linear classification drops from 0.966 to 0.904 while the rotation prediction performance increases by $3.4 \pm 1.4\%$ across all evaluations. This correlation drop indicates that using a simple linear layer, rather than a more complicated network, is ideal for evaluation of the learned representations. A more complicated network can learn its own representation, which distances the evaluation from the learned representations.

D.3. Correlation demonstration: finding and tuning transformation parameters

Figure 3 shows the performance of single-transform policies for CIFAR-10. The left and middle plots show the supervised classification accuracy compared with the InfoNCE loss and top-1 contrastive accuracy (how well the instance contrastive model predicts the augmented image pairs), while the right plot shows the rotation prediction accuracy for the image transformations in $\mathbb{O}$ evaluated after 100 training epochs. Using high or low values of InfoNCE or contrastive accuracy to select the best transformations would select a mixture of mediocre transformations, missing

the top performing transformation in the middle. By using the rotation prediction, each transformation has a clear linear relationship with the supervised performance, enabling the unsupervised selection of the best transformations.

Next, Figure 4 demonstrates that the individual image transformation *parameters* can also be determined through rotation prediction. Specifically, we pre-trained CIFAR-10 using the default MoCoV2 augmentation policy. Then, for each of the fifteen transformation in $\mathbb{O}$, we applied the transformation with probability 1.0 and a magnitude λ randomly selected between 0 and $\{0.25, 0.5, 0.75, 1.0\}$. We evaluate the individual transformation's magnitude parameter using the supervised top-1 linear accuracy (classification) and self-supervised top-1 rotation prediction (rotation) as shown in Figure 4. Overall, the supervised linear classification and self-supervised rotation prediction select the the same magnitude parameter for 13 of the 15 transformations and have a strong Spearman rank correlation of 0.929. Taken together, these results indicate that rotation prediction can be used to both select individual transformations as well as the parameters of the transformations for an augmentation policy.

D.4. Rotation correlation shows the benefit of Gaussian blur on ImageNet

In [7], the authors conducted a thorough, supervised investigation of a diverse set of image transformations that could be used in an augmentation policy for instance contrastive learning with ImageNet. The `Gaussian blur` augmentation was shown to be one of the most effective image transformations for ImageNet, and indeed, [5] released a follow-up paper showing that the MoCo framework [2] significantly benefits from its use. We find that our rotation-based evaluation similarly indicates that Gaussian blur is an effective image transformation for ImageNet under the same conditions studied in [5]:

Augmentation Policy	Top-1 Supervised Acc.	Rotate Acc.
MoCoV1	60.6	72.1
MoCoV2 no G-Blur	63.6	74.1
MoCoV2	67.7	77.0

D.5. Rotation invariant and black-and-white images

We note that certain types of images are not amenable to certain self-supervised evaluations. For instance, rotation evaluation will not work for rotation invariant images (such as images of textures) as the self-supervised evaluation task will not be able to discern the rotations. Similarly, a jigsaw task will not be able to discern images with a interchangeable quadrants (such as centered images of flowers), and a color prediction tasks will not work on black and white images (such as x-ray images). Therefore, for each image dataset, we recommend using a self-supervised evaluation that does not evaluate an invariant factor of the image dataset, e.g. use rotation prediction if the images are black-and-white. We leave an investigation of the trade-offs between image invariances and self-supervised evaluations to future work.

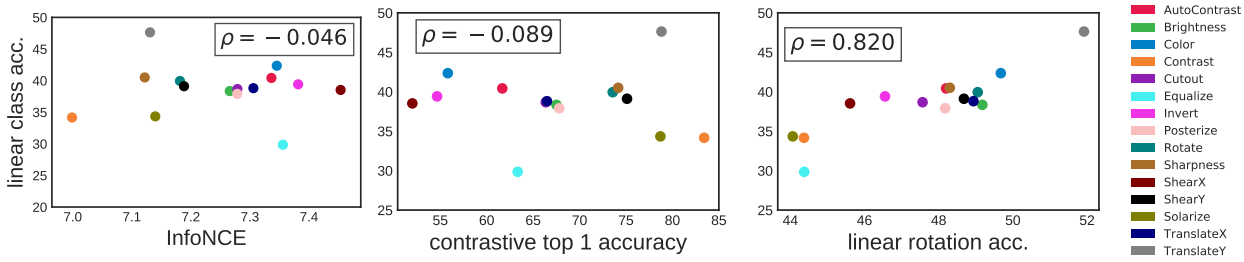

Figure 3: For CIFAR-10, we plot the supervised classification accuracy (y-axis) vs the InfoNCE loss function (left), contrastive top-1 accuracy (middle), and self-supervised linear rotation accuracy (right), for a self-supervised model trained using one of each transformation used by SelfAugment. Neither of the left two training metrics are a consistent measure of the quality of the representations, while the rotation prediction accuracy provides a strong linear relationship.

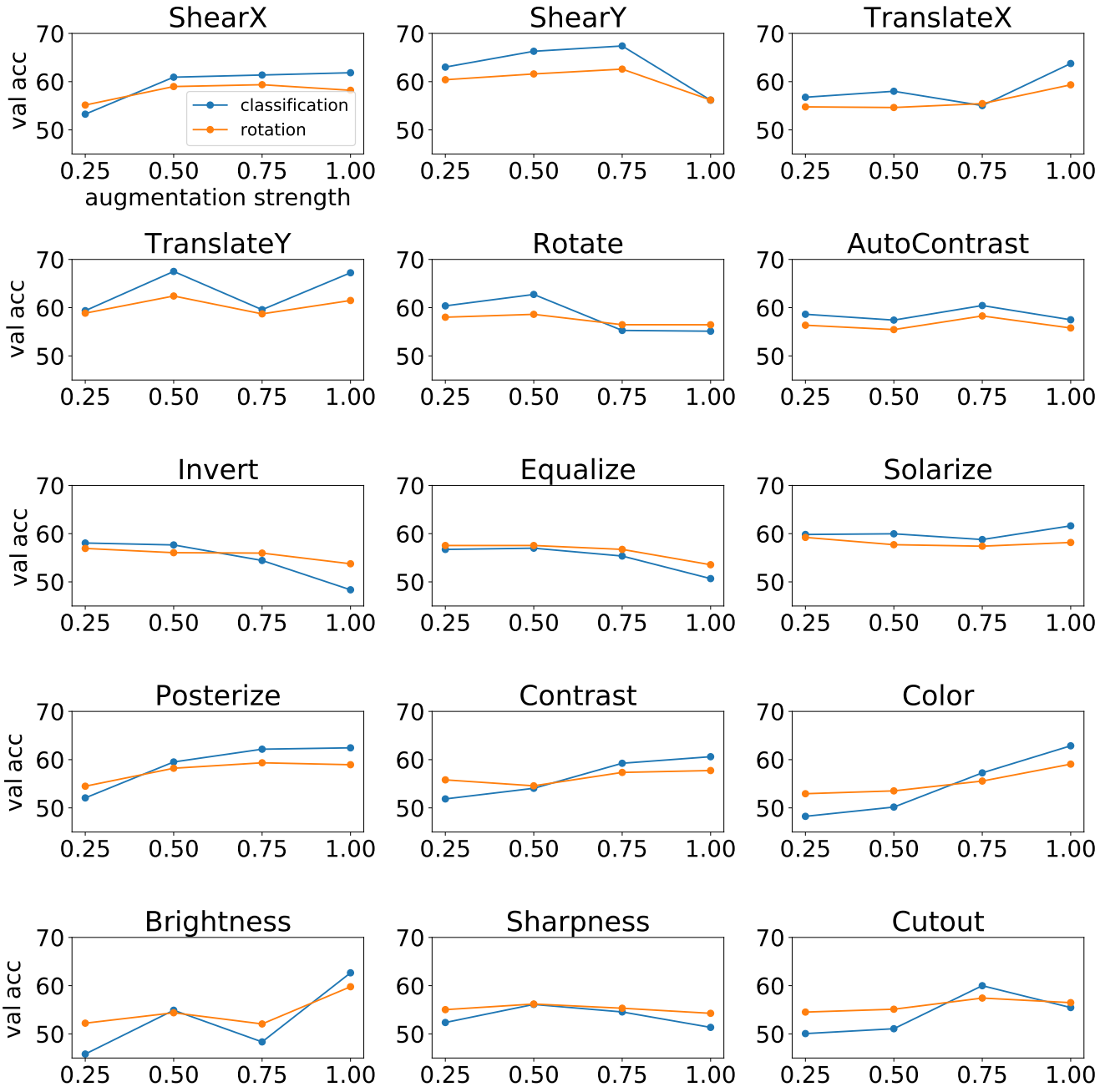

Figure 4: Result of magnitude sweeps on CIFAR-10 for the 15 transformations optimized in SelfAugment. For each augmentation, we train MoCoV2 for 250 epochs using the usual MoCoV2 augmentations, and then added the single augmentation on top. We then vary the range of possible augmentation strength λ parameter to be between 0 and the value on the x-axis, and randomly select a value in that range, and apply the augmentation with 100 % probability. Rotation accuracy and classification accuracy have Spearman rank correlation $\rho = .929$, demonstrating how the relationship between rotation accuracy and classification accuracy can be used to select the parameters of individual transformations.

E. Modified RV similarity analysis

To obtain a better understanding of why the the rotation evaluation has the best correlation with the supervised evaluation performance, we conduct a similarity analysis of the *activations* from the linear evaluation layers. Specifically, we use a modified RV coefficient (as in [8]) to measure the similarity between the activations from the rotation, jigsaw, and colorization evaluation layers on top of the frozen encoder network. The RV coefficient is a matrix correlation method that compares paired comparisons X and Y with different number of columns, and is defined as:

$$RV(X,Y) = \frac{tr(XX'YY')}{\sqrt{tr[(XX')^2]tr[(YY')^2]}} \quad (1)$$

The RV coefficient approaches 1 when datasets are small, even for random and unrelated matricies. To fix this issue, the modified RV coefficient (RV_2) ignores the diagonal elements of XX' and YY', which pushes the numerator to zero when X and Y are random matricies. Hence, the RV_2 similarity metric is less sensitive to dataset size.

$$RV_2(X,Y) = \frac{Vec(X\tilde{X}')'Vec(Y\tilde{Y}')}{\sqrt{Vec(X\tilde{X}')'Vec(X\tilde{X}') \times Vec(Y\tilde{Y}')'Vec(Y\tilde{Y}')}} \quad (2)$$

Where $X\tilde{X}' = XX' - diag(XX')$ and similarly for $Y\tilde{Y}'$. This metric is invariant to orthogonal transformations and isomorphic scaling, but critically not invariant to arbitrary linear invertible linear transformations between representations (e.g. batch normalization). In [8], the authors show that the RV_2 metric could recover expected similarity patterns in neural networks and that it could be used to suggest hypotheses about intermediate representations in deep neural networks.

To study the similarity between linear layers trained using the rotation prediction, jigsaw, and colorization tasks and linear layers trained using supervised learning, we evaluated the RV_2 coefficient of 32 different linear layers trained on top of frozen encoders using the CIFAR-10 dataset. Each of the 32 encoders used a different augmentation policy during training. We evaluated the activations at the final linear layer across the entire validation set of CIFAR-10. We used the same set of image transformations before feeding each image into the the network (center crop to 28x28, and normalization across each channel by it's mean and std deviation across the dataset) across all self supervised tasks. As demonstrated in Figure 5, activations from rotation prediction layers had significantly stronger similarities with activations from supervised layers than other self supervised tasks. This demonstrates that the rotation prediction task uses the learned representation in a significantly more similar way for evaluation compared to the other evaluation tasks, and provides evidence that rotation evaluation performance not only correlates, but so does the activations from the evaluation layer.

F. Augmentation policy exploration via Bayesian optimization

As in [1], we used policy exploration search to automate the augmentation search. Since there are an infinite number of possible policies, we applied Bayesian optimization to explore augmentation strategies. In line 11 in Algorithm 1, we employed the Expected Improvement (EI) criterion as an acquisition function to explore $\mathcal{B}$ candidate policies efficiently: $EI(\mathcal{T}) = \mathbb{E}[min(\mathcal{L}(\theta,\phi|\mathcal{T}(\mathcal{D}_\mathcal{A}) - \mathcal{L}^\dagger, 0)]$. Here $L^\dagger$ represents a constant threshold determined by the quantile of observations amongst previously explored policies. As in [1], we used variable kernel density estimation on a graph-structured search space to approximate the criterion. Since this method is already implemented in the tree-structured Parzen estimator algorithm we used `Ray`[4] and `Hyperopt` to implement this in parallel.

In [1], the authors try to align distributions of data using supervised loss, then retrain the network using supervised loss. Since we do not directly retrain with the loss functions used to find augmentations, our method can instead be thought of as finding distributions of the data - via augmentation policies - that minimize (or maximize) alignment as defined by our loss functions.

The full list of augmentations and range of magnitudes explored during augmentation policy exploration are detailed in Table 1.

G. SelfAugment Loss Functions

In this section we discuss the loss functions used for SelfAugment in greater detail and their resulting augmentation policies which are summarized in Figure 6.

- **Min. eval error**:

  $$\mathcal{T}^{SS} = \text{argmin}_\mathcal{T} \mathcal{L}_{\text{SS}}(\theta_\mathcal{M}, \phi_{\text{ss}}|\mathcal{T}(D_\mathcal{A}))$$

  where $\mathcal{L}_{\text{SS}}$ is the self-supervised evaluation loss, which yields policies that should result in improved performance of the evaluation if we were to retrain the linear classifier with the selected augmentations. However, since the augmentations are instead used to create a contrastive learning task, it is important that we find a different set of augmentations that take the contrastive task into consideration. Indeed, using this loss function results in weak expected augmentation strengths (see Figure 6), resulting in relatively poor downstream performance.

[4] https://github.com/ray-project/ray

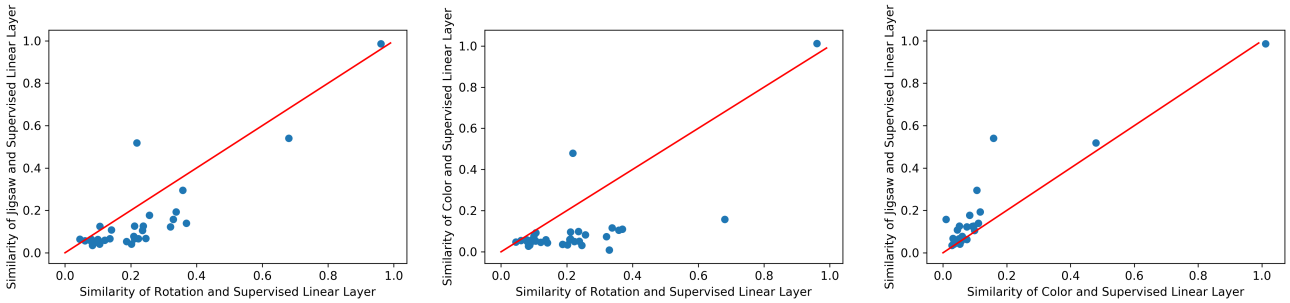

Figure 5: Comparisons of RV_2 Similarity [8] between each self supervised task studied and supervised linear layers. Rotation prediction layers have stronger similarity with supervised linear layers than either of the jigsaw and colorization self supervised tasks ($p < 2.3\text{x}10^{-5}$, Wilcoxon one-tailed signed rank test). The datapoint with highest similarity was using the strongest settings of SelfRandAugment, which learned poor representations with near chance performance on the supervised downstream task, explaining why similarity was so high across tasks. The strong similarity between rotation prediction and supervised linear layers suggests that they use the underlying representations in a similar way for evaluation, explaining why rotation prediction makes for a strong evaluation metric that correlates more strongly with supervised evaluation than the jigsaw and colorization evaluations.

- **Min. InfoNCE**:

$$\mathcal{T}^{\text{I-min}} = \text{argmin}_{\mathcal{T}} \mathcal{L}_{\text{NCE}}(\theta_{\mathcal{M}} | \mathcal{T}(D_{\mathcal{A}}))$$

  where $\mathcal{L}_{\text{NCE}}$ is the InfoNCE loss from Eq. 1, which yields policies that make it easier to distinguish image pairs in the contrastive feature space. This loss function should results in small magnitude augmentations, since a trivial way to minimize InfoNCE is to apply no transforms - resulting in trivial minimization of the InfoNCE loss. Indeed, the loss function yields (i) lower expected augmentation strengths and (ii) emphasizes transformations like `Sharpness`, Figure 6. We did not expect this loss function to perform well, and was primarily included as a sanity check that minimizing InfoNCE would result in light augmentations.
- **Max InfoNCE**:

$$\mathcal{T}^{\text{I-max}} = \text{argmin}_{\mathcal{T}} - \mathcal{L}_{\text{NCE}}(\theta_{\mathcal{M}} | \mathcal{T}(D_{\mathcal{A}}))$$

  negates the previous loss function, yielding policies that make it difficult to distinguish image pairs in the feature space. In practice, this results in high magnitude augmentations and emphasizes augmentations like `Invert`. We hypothesized that maximizing InfoNCE could result in strong augmentations that could create a challenging contrastive task. However, the augmentations do not have any regularizing that would ensure the image maintains important features, leading to relatively suboptimal performance.
- **Min** $\mathcal{L}_{\text{ss}}$ **max** $\mathcal{L}_{\text{NCE}}$: $\mathcal{T}^{\text{minmax}} = \text{argmin}_{\mathcal{T}} \mathcal{L}_{\text{ss}} - \mathcal{L}_{\text{NCE}}$ yields policies with difficult transformations that maximize InfoNCE, while maintaining salient object features that minimize the self-supervised evaluation. When using a linear rotation evaluation prediction, we found this loss function to have the strongest performance for contrastive learning. In practice, we normalized $\mathcal{L}_{\text{rot}}$ and $\mathcal{L}_{\text{NCE}}$ by their expected value for the training data across the K-folds when training with the base augmentation, since $\mathcal{L}_{\text{NCE}}$ was usually higher than $\mathcal{L}_{\text{rot}}$, but had each loss contribute equally for augmentation policy selection. Policies that optimized this loss emphasized transformations like `Equalize`, `AutoContrast` and `Contrast` more than others. For ImageNet, these transformations proved to be challenging yet useful. Notably, they closely resemble the effects of MoCov2 and SimCLR's Color Jitter [5].

Finally, while **min** $\mathcal{L}_{\text{rot}}$ **max** $\mathcal{L}_{\text{NCE}}$ works well, one could potentially gain performance by weighting the importance of minimizing $\mathcal{L}_{\text{rot}}$ vs maximizing $\mathcal{L}_{\text{NCE}}$. Hence we propose experimenting with different values of λ_{NCE} and λ_{rot} when optimizing the following objective: **min** $\lambda_{\text{rot}}\mathcal{L}_{\text{rot}}$ **max** $\lambda_{\text{NCE}}\mathcal{L}_{\text{NCE}}$.

H. Using the Original MoCoV2 [5] policy as the base policy

We replicated our SelfAugment results in Table 1 for the CIFAR10 dataset, where instead of using a single augmentation as the base policy, we used the full augmentation policy from [5] as the base policy. Results are shown in Table 5. This improved performance when we used the augmentation policies learned using min $\mathcal{L}_{\text{rot}}$ as feedback. Minimizing rotation may be the best approach for learning augmentation policies on top of established augmentation policies, because it produces augmentations that would improve performance if we retrained the linear classifier with

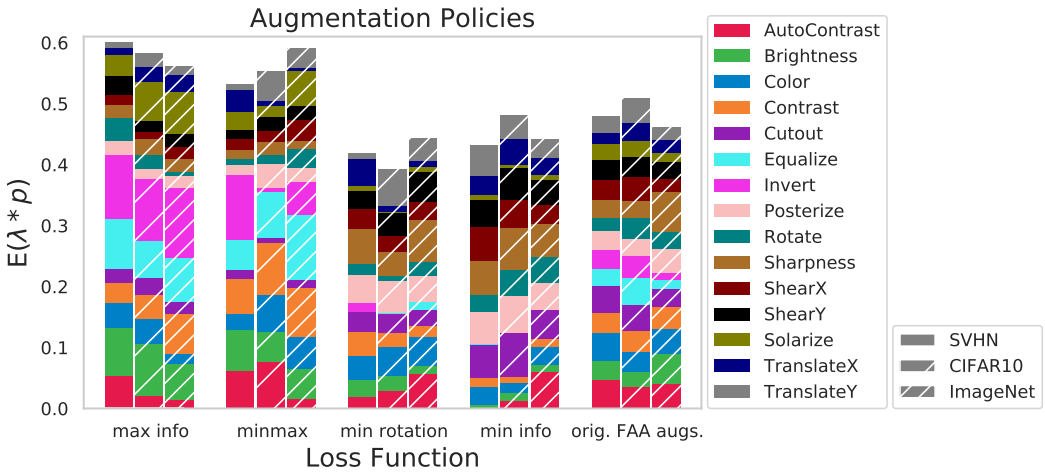

Figure 6: Visualization of the augmentation policies found with SelfAugment as well as Fast AutoAugment for supervised learning. This figure shows expected augmentation strength (mean magnitude×normalized probability) of each augmentation, evaluated across all datasets and loss functions. As expected, minimizing InfoNCE results in augmentations with smaller magnitude, and emphasizes augmentations that do not alter the image much (e.g. `Sharpness`). Maximizing InfoNCE results in augmentations with a larger magnitude and emphasizes augmentations (e.g. `Invert, Solarize`) which heavily alter the image. The minimax loss function yields an augmentation policy that strikes a middle ground between the two, with strong augmentations, but reduced emphasis on heavy augmentations like `Invert`. Comparison of the policies that worked best with contrastive learning relative to the original FAA policies reveals that our policies are (i) stronger, and (ii) have more variability in a single augmentation's E($\lambda * p$) relative to policies that were used for supervised learning.

the selected augmentations. Since the augmentation policy for contrastive learning is already quite strong when we use the MoCov2 augmentations as the base policy, this likely allows us to focus on improving generalization [1], leading to improved performance. Interestingly, the other SelfAugment Loss functions (see Appendix G for more detail) all deteriorated performance. This is likely because trying to change the contrastive performance of such a finely tuned set of augmentations requires more careful tuning. It is possible the minimax approach may result in improved performance with more careful tuning of the weighting of the rotation and contrastive losses.

It is also worth noting that using the best policy using the MoCov2 augmentations as the base augmentation set, then using SelfAugment with min $\mathcal{L}_{\text{rot}}$ as feedback slightly outperformed the full SelfAugment pipeline with min $\mathcal{L}_{\text{rot}}$ max $\mathcal{L}_{\text{ICL}}$ as feedback. For datasets where researchers are confident that the augmentations from [5] are a strong base set of augmentations, this approach is worth exploring, and can be easily done using our pipeline. However, it is not possible when a good base augmentation policy is unknown. We leave further exploration of this approach as future research, as we chose to focus on the case where no existing augmentation policy is known.

unsup. feedback	C10
SelfAug (min rot)	**92.8**
SelfAug (min Info)	92.2
SelfAug (max Info)	90.5
SelfAug (minimax)	90.9
supervised feedback	
MoCov2[5]	92.3

Table 5: Results on CIFAR10 when using SelfAugment but using the augmentation policy from [5] as the base policy. Because the base augmentation policy is already strong, minimizing rotation loss during the augmentation policy search produces the best results. The results with the MoCoV2 augmentations as the base policy and minimization of rotation prediction slightly outperform the best results using SelfAugment and the minimax loss as feedback (92.6, see Table 1).

I. Computational Efficiency

SelfAugment can be broken up into three steps:

1. **Finding the base augmentation**. This entails training 16 instances of MoCoV2 for 10-15% of the total epochs typically used for pre-training, using all of the training data available.

2. **Training on K-Folds using the base augmentation**.

For CIFAR-10/SVHN we used the entire training dataset for this step, and evaluated for the same number of epochs as we use to train the final models, using all the training data from each fold. For ImageNet, we used a subset of 50,000 images and trained for 5x longer, to make up for the smaller amount of images. This was done to improve the computational efficiency of step 3.

3. **Finding augmentation policies**. Because we only need to complete forward passes of the network, this is relatively efficient. We use the held out data from each of the K-folds to evaluate the model with different augmentation policies applied.

It is important to note that steps 1 and 2 are *shared* across all augmentation policies found with SelfAug, meaning that they do not need to be repeated to find a new augmentation policy using a different loss function. This could allow for rapid experimentation with new loss functions for SelfAugment in the future.

Meanwhile, SelfRandAug's computational time is completely determined by the user-defined search space over λ and N_τ. However, because it requires training multiple models to completion on the entire dataset, it is more computationally intensive than SelfAug. SelfRandAug's computation time could be reduced by training on a subset of Images in a large training set.

Finally, it is worth noting that evaluating a *single* augmentation's various hyperparameters for MoCoV2 can be extremely computationally expensive. Because pre-training MoCoV2 for 100 epochs then training a linear head for evaluation takes a total of 244 GPU Hours on a Tesla V100 GPU, searching over various augmentations and their strength and probability parameters can easily require thousands of GPU hours. Hence, an additional contribution of this work is providing a fast, automatic method for selecting augmentations for instance contrastive learning.

Table 6: Computational time, measured in one hour on one NVIDIA Tesla V100 GPU for SelfAugment and SelfRandAug on ImageNet. SelfAug times are evaluated when using the minimax loss function, which takes the most time because we evaluate both InfoNCE and rotation loss. For SelfRandAug, computational time reflects our grid search over $\lambda = \{5,7,9,11,13\}$, $N_\tau = 2$, plus the time to train a rotation head on top of each representation. This time could be larger or smaller depending on the parameters λ and N_τ a user wants to search over.

Algorithm	Find Base Aug	Train	Find Aug	**Total**
SelfAug	476.6	155.0	97.7	**729.4**
SelfRandAug				**1220.0**

References

[1] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. In *Advances in Neural Information Processing Systems*, pages 6662–6672, 2019.

[2] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[3] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123, 2019.

[4] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical data augmentation with no separate search. *arXiv preprint arXiv:1909.13719*, 2019.

[5] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[6] Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6391–6400, 2019.

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[8] Jessica A. F. Thompson, Yoshua Bengio, and Marc Schönwiesner. The effect of task and training on intermediate representations in convolutional neural networks revealed with modified RV similarity analysis. *CoRR*, abs/1912.02260, 2019.