SelfAugment: Automatic Augmentation Policies for Self-Supervised Learning

Colorado 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

Abstract

A common practice in unsupervised representation learning is to use labeled data to evaluate the quality of the learned representations. This supervised evaluation is then used to guide critical aspects of the training process such as selecting the data augmentation policy. However, guiding an unsupervised training process through supervised evaluations is not possible for real-world data that does not actually contain labels (which may be the case, for example, in privacy sensitive fields such as medical imaging). Therefore, in this work we show that evaluating the learned representations with a self-supervised image rotation task is highly correlated with a standard set of supervised evaluations (rank correlation > 0.94). We establish this correlation across hundreds of augmentation policies, training settings, and network architectures and provide an algorithm (SelfAugment) to automatically and efficiently select augmentation policies without using supervised evaluations. Despite not using any labeled data, the learned augmentation policies perform comparably with augmentation policies that were determined using exhaustive supervised evaluations.

1. Introduction

Self-supervised learning, a type of unsupervised learning that creates target objectives without human annotation, has led to a dramatic increase in the ability to capture salient feature representations from unlabeled visual data. So much so, that in an increasing number of cases these representations outperform representations learned from the same data with labels [1, 2, 3]. At the center of these advances is a form of *instance contrastive learning* where a single image is augmented using two separate data augmentations, and then a network is trained to distinguish which augmented images originated from the same image when contrasted with other randomly sampled augmented images, see [1, 3, 4, 5].

As illustrated in Figure 1, recent works [1, 4, 6] have used extensive *supervised* evaluations to determine which

_____

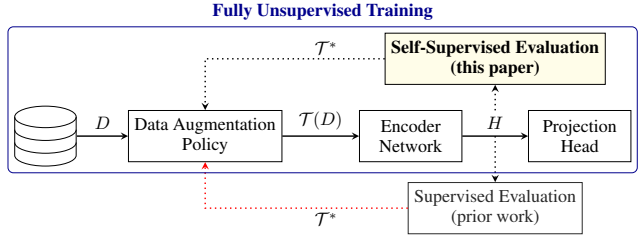[*]equal contribution; correspondence to `cjrd@cs.berkeley.edu`

Figure 1: The blue box highlights our fully unsupervised training pipeline for instance contrastive representation learning: data D are augmented with policy $\mathcal{T}$, then encoded into representations, H, which are fed into a projection head yielding features that determine the InfoNCE loss (Eq. 1). As shown by the red arrow, prior work uses supervised evaluations of the representations, F to inform the training process, e.g. to update the augmentation policy $\mathcal{T} \to \mathcal{T}^*$. In this paper, we show that self-supervised evaluation can be used in lieu of supervised evaluation and show how to use this evaluation for automatic and efficient augmentation selection.

augmentation policies to use for training. The best policies obtain a *sweet spot*, where the augmentations make it difficult for the contrastive task to determine the corresponding image pairs while retaining salient image features; finding this sweet spot can be the difference between state-of-the-art performance or poor performance for various tasks [6].

However, it is often difficult or impossible to obtain accurately labeled data in privacy sensitive fields (e.g. medical imaging [7]), applications with highly ambiguous label definitions (e.g. fashion or retail categorization [8]), or not practical when one set of representations is used for a diverse set of downstream tasks (e.g. in autonomous driving systems [9]). This leads to the open question: *How can we evaluate self-supervised models, especially to efficiently select augmentation policies, when labeled data is not available?* We address this question via the following contributions:

1. We show that a linear, image-rotation-prediction evaluation task is highly correlated with the downstream supervised performance (rank correlation $\rho > 0.94$) on six stan-

dard recognition datasets (CIFAR-10 [10], SVHN [11], ImageNet [12], PASCAL [13], COCO [14], Places-205 [15]) and tasks (image classification, object detection, and few-shot variants) across hundreds of learned representations, spanning three types of common evaluation techniques: linear separability performance, semi-supervised performance, and transfer learning performance.

2. Using self-supervised evaluation, we adapt two automatic data augmentation algorithms for instance contrastive learning. Without using labeled evaluations, these algorithms discover augmentation policies that match or outperform policies obtained using supervised feedback and only use a fraction of the compute.

3. We further show that using linear image rotation prediction to evaluate the representations works across network architectures, and that image rotation prediction has a stronger correlation with supervised performance than a jigsaw [16] or color prediction [17] evaluation task.

Based on these contributions and experiments, we conclude that image rotation prediction is a strong, unsupervised evaluation criteria for evaluating and selecting data augmentations for instance contrastive learning.

2. Background and Related Work

In this paper, we study evaluations for self-supervised representations, particularly through the lens of learning data augmentation policies. We discuss these topics next.

Self-supervised representation learning: The general goal of representation learning is to pre-train a network and then either fine-tune it for a particular task or transfer it to a related model, e.g. see [2, 3, 18, 19, 20, 21, 22, 23, 24]. Recently, [1] and [4] demonstrated substantial improvements by using similar forms of instance contrastive learning whereby one image is augmented using two separate data augmentations and then a network is trained to identify this positive pair contrasted with a large set of distractor images. A common loss function for contrastive methods is the InfoNCE loss, where given two images originating from the same image i and K_d distractor images we have:

$$\mathcal{L}_{NCE} = -\mathbb{E}\left[\log \frac{\exp(sim(\mathbf{z_{1,i}}, \mathbf{z_{2,i}}))}{\sum_{j=1}^{K_d} \exp(sim(\mathbf{z_{1,i}}, \mathbf{z_{2,j}}))}\right] \quad (1)$$

where $\mathbf{z_{1,i}}, \mathbf{z_{2,i}}$ are the two different image representations from image i following the encoder network and projection head as shown in Figure 1, and $sim(\cdot, \cdot)$ is a similarity function such as a weighted dot product.

The InfoNCE loss [25, 2] has been shown to maximize a lower bound on the mutual information $I(\mathbf{h_1}; \mathbf{h_2})$. The Sim-CLR framework [1] relies on large batch sizes to contrast

the image pairs, while the MoCo framework [4] maintains a large queue of contrasting images. Given the increased adoption, broad application, and strong performance of instance contrastive learning [2, 1, 4], we focus our work on this type of self-supervised learning, specifically using the MoCo algorithm and training procedure for experimentation [4].

Self-supervised model evaluation is typically done via:

- *separability*: the network is frozen and the training data trains a supervised linear model (the justification is that good representations will reveal linear separability in the data) [26, 27, 28, 16, 29, 30]

- *transferability*: the network is either frozen or jointly fine-tuned, with a transfer task model that is fine-tuned using a different, labeled dataset (the justification is that good representations will generalize to a number of downstream tasks) [2, 3, 23, 24]

- *semi-supervised*, in which the network is either frozen or jointly fine-tuned using a fraction of labeled data with either the *separability* or *transferability* tasks mentioned above (the justification is that a small set of labeled data will benefit good representations) [2, 1].

While these evaluations characterize the unsupervised model in several ways, they have limited use for making training decisions because: **(i)** accessing labels as a part of the training process is not possible for unlabeled datasets, and **(ii)** evaluating the model on a different, labeled dataset requires an integrated understanding of the relationship between the transfer task, datasets, and models (see [9, 31, 32]). We seek a label-free, task-agnostic evaluation.

Learning data augmentation policies: Data augmentation has played a fundamental role in visual learning, and indeed, has a large body of research supporting its use [33]. In this work, we use a self-supervised evaluation to automatically learn an augmentation policy for instance contrastive models. To formulate our automatic data augmentation framework, we draw on several, equally competitive recent works in the supervised learning space [34, 35, 36, 37], where [34, 35, 36] use a separate search phase to determine the augmentation policy and [37] use a simplified augmentation space and sample from it via a grid search. For instance contrastive learning, we adapt a search-based automatic augmentation framework, Fast AutoAugment (FAA) [36], and a sampling-based approach, RandAugment [37]. We discuss these two algorithms in greater detail in the next section.

3. Self-Supervised Evaluation and Data Augmentation

Our central goals are to **(i)** establish a strong correlation between a self-supervised evaluation task and a supervised evaluation task commonly used to evaluate self-supervised models, and **(ii)** develop a practical algorithm for

self-supervised data augmentation selection. The following subsections defines these goals in more detail.

3.1. Self-supervised evaluation

With labeled data, augmentation policy selection can directly optimize the supervised task performance [35, 36]. With unlabeled data, we seek an evaluation criteria that is highly correlated with the supervised performance without requiring labels. Inspired by [38], where the authors show that self-supervised tasks can be used to evaluate network architectures, we investigate the following self-supervised tasks to evaluate representations:

rotation [27]: the input image undergoes one of four preset rotations $\{0^o, 90^o, 180^o, 270^o\}$, and the evaluation metric is the 4-way rotation prediction classification accuracy

jigsaw [16]: the four quadrants of the input image are randomly shuffled into one of $4! = 24$ permutations, and the evaluation metric is the 24-way classification accuracy

colorization [17]: the input is a grayscale image, and the evaluation metric is formulated as a pixel-wise classification accuracy on pre-defined color classes (313, from [17])

A key point to emphasize is that these self-supervised tasks are used to *evaluate* the representations learned from instance contrastive algorithms, e.g. MoCo. This is worth stressing as these self-supervised tasks were originally used to learn representations themselves, but in this work, we train these tasks using a linear layer in order to evaluate the representations from instance contrastive learning. In §4, we compute the correlation of each of these evaluations with a supervised, top-1 linear evaluation on a frozen backbone trained using a cross entropy loss on the training data.

3.2. Self supervised data augmentation policies

We study and adapt two approaches for augmentation policy selection from the supervised domain: a sampling-based strategy, RandAugment [34] and a search-based strategy, Fast AutoAugment (FAA) [36]. Using the notation from [36], let $\mathbb{O}$ represent the set of image transformations operations $\mathcal{O} : \mathcal{X} \to \mathcal{X}$ on input image $\mathcal{X}$. Following [36, 37], we define $\mathbb{O}$ as the set: {cutout, autoContrast, equalize, rotate, solarize, color, posterize, contrast, brightness, sharpnes, shear-x, shear-y, translate-x, translate-y, invert} (see Appendix B for more details).

Each transformation $\mathcal{O}$ has two parameters: **(i)** the magnitude λ that determines the strength of the transformation and **(ii)** the probability of applying the transformation, p. Let $\mathcal{S}$ represent the set of augmentation sub-policies, where a sub-policy $\tau \in \mathcal{S}$ is defined as the sequential application of N_τ consecutive transformation $\{\bar{\mathcal{O}}_n^{(\tau)}(x; p_n^{(\tau)}, \lambda_n^{(\tau)}) : n = 1, \ldots, N_\tau\}$

Algorithm 1: SelfAugment takes as input a dataset, D_{train}, parameters for policy optimization (see Appendix A for definitions), and loss function $\mathcal{L}$ and returns an augmentation policy.

Input : $(D_{\text{train}}, K, T, B, P, \mathcal{L})$

1 Split D_{train} into K-folds: $D_{\text{train}}^{(k)} = \{(D_{\mathcal{M}}^{(k)}, D_{\mathcal{A}}^{(k)})\}$
2 **for** $a \in \mathbb{O}$ **do**
3   Train θ_{moco} on single aug policy $\mathcal{T}_a(D_{\mathcal{M}}^{(1)})$
4   Train ϕ_{ss} on $D_{\mathcal{M}}^{(1)}$ on top of frozen θ_{moco}
5      $a^* \leftarrow \text{argmin}_{\hat{a} \in \{a, a^*\}} \mathcal{L}(\theta_{moco}, \phi_{ss} | \mathcal{T}_{\hat{a}}(D_{\mathcal{M}}^{(1)}))$
6 **for** $k \in \{1, \ldots, K\}$ **do**
7      $\mathcal{T}^{*(k)} \leftarrow \emptyset, (D_{\mathcal{M}}, D_{\mathcal{A}}) \leftarrow (D_{\mathcal{M}}^{(k)}, D_{\mathcal{A}}^{(k)})$
8   Train θ_{moco} on base aug policy $\mathcal{T}_{a^*}(D_{\mathcal{M}})$
9   Train ϕ_{ss} on $D_{\mathcal{M}}$ on top of frozen θ_{moco}
10   **for** $t \in \{0, \ldots, T-1\}$ **do**
11          $\mathcal{B} \leftarrow \text{BayesOpt}(\mathcal{T}, \mathcal{L}(\theta_{moco}, \phi_{ss} | \mathcal{T}(D_{\mathcal{A}})), B)$
12          $\mathcal{T}_t \leftarrow$ Select top-P policies in $\mathcal{B}$
13   Merge policies via $\mathcal{T}^{*(k)} \leftarrow \mathcal{T}^{*(k)} \cup \mathcal{T}_t$
14 **return** $\mathcal{T}^* = \bigcup_k \mathcal{T}^{*(k)}$

---

where each operation is applied to an input image sequentially with probability p. Applying sub-policy $\tau(x)$ is then a composition of transformation $\tilde{x}_{(n)} = \bar{\mathcal{O}}_n^{(\tau)}(\tilde{x}_{(n-1)})$ for $n = 1, \ldots, N_\tau$, where the full sub-policy application has shorthand $\tilde{x}_{(N_\tau)} = \tau(x)$ and the first application is $\tilde{x}_{(0)} = x$. The full policy, $\mathcal{T}$, is a collection of $N_{\mathcal{T}}$ sub-policies, and $\mathcal{T}(D)$ represents the set of images from D obtained by applying $\mathcal{T}$.

SelfRandAugment: RandAugment makes the following simplifying assumptions: **(i)** all transformations share a single, discrete magnitude, $\lambda \in [1, 30]$ **(ii)** all sub-policies apply the same number of transformations, N_τ **(iii)** all transformations are applied with uniform probability, $p = K_T^{-1}$ for the $K_T = |\mathbb{O}|$ transformations. RandAugment selects the best result from a grid search over (N_τ, λ). To adapt this algorithm for instance contrastive learning, we simply evaluate the searched (N_τ, λ) states using a self-supervised evaluation from §3.1 and refer to this as *SelfRandAugment*.

SelfAugment: We adapt the search-based FAA algorithm to the self-supervised setting; we call this adaptation *SelfAugment*. Formally, let $\mathcal{D}$ be a distribution on the data $\mathcal{X}$. For model $\mathcal{M}(\cdot | \theta) : \mathcal{X}$ with parameters θ, define the supervised loss as $\mathcal{L}(\theta | D)$ on model $\mathcal{M}(\cdot | \theta)$ with data $D \sim \mathcal{D}$. For any given pair of D_{train} and D_{valid}, FAA selects augmentation policies that approximately align the density of D_{train} with the density of the augmented $\mathcal{T}(D_{\text{valid}})$. This means that the transformations should help the model bolster meaningful features and become invariant to unimportant features after retraining with the augmented dataset. In practice, FAA splits D_{train} into $D_{\mathcal{M}}$ and $D_{\mathcal{A}}$, where $D_{\mathcal{M}}$ is used to train the

model and $D_\mathcal{A}$ is used to determine the policy via:

$$\mathcal{T}^* = \operatorname*{argmin}_{\mathcal{T}} \mathcal{L}(\theta_\mathcal{M}|\mathcal{T}(D_\mathcal{A})) \qquad (2)$$

where $\theta_\mathcal{M}$ is trained using $D_\mathcal{M}$. This approximates minimizing the distance between the density of $D_\mathcal{M}$ and $\mathcal{T}(D_\mathcal{A})$ by using augmentations to improve predictions with shared model parameters, $\theta_\mathcal{M}$; see [36] for derivations. FAA obtains the final policy, $\mathcal{T}^*$, by exploring B candidate policies $\mathcal{B} = \{\mathcal{T}_1, \ldots, \mathcal{T}_B\}$ with a Bayesian optimization method that samples a sequence of sub-policies from $\mathcal{S}$ and adjusts the probabilities $\{p_1, \ldots, p_{N_\mathcal{T}}\}$ and magnitudes $\{\lambda_1, \ldots, \lambda_{N_\mathcal{T}}\}$ to minimize $\mathcal{L}(\theta|\cdot)$ on $\mathcal{T}(D_\mathcal{A})$ (see Appendix F for details). The top P policies from each data split are merged into $\mathcal{T}^*$. The network is then retrained using this policy on all training data, $\mathcal{T}^*(D_{\text{train}})$, to obtain the final network parameters θ^*. SelfAugment has three main differences from Fast AutoAugment that we discuss next.

Select the base policy: A *base augmentation policy* is required to perform the first pass of training to determine $\theta_\mathcal{M}$. SelfAugment determines this policy by training a MoCo network [4] for a short period of time on each of the individual transformations in $\mathbb{O}$ as well as `random-resize-crop` (the top performing single transformation in [1]). Each transformation is applied at every iteration, with $p = 1$ and a magnitude parameter λ stochastically set at each iteration to be within the ranges from [36]. Each network is trained until the loss curves separate – we found this to be around 10% of the total training epochs typically used for pre-training. The backbone is then frozen and a linear self-supervised evaluation task, ϕ_{ss}, is trained for each network and evaluated using held out training data. The base policy is then the transformation with the best evaluation.

Search augmentation policies: Given the base augmentations, we split the training data into k-folds. For each fold, we train a MoCo network, θ_{moco}, using the base augmentation, freeze the network, and train a self-supervised evaluation layer, ϕ_{ss}. We use the same Bayesian optimization search strategy as FAA to determine the policies. However, as the loss function $\mathcal{L}(\theta|D)$ cannot use supervised accuracy as in FAA, we explore four variants of a self-supervised loss function. Appendix G discusses and compares each of these loss functions in more detail and §5 compares these loss functions with a supervised variant:

Min. eval error: $\mathcal{T}^{SS} = \operatorname{argmin}_\mathcal{T} \mathcal{L}_{ss}(\theta_\mathcal{M}, \phi_{ss}|\mathcal{T}(D_\mathcal{A}))$ where $\mathcal{L}_{ss}$ is the self-supervised evaluation loss, which yields policies that would directly result in improved performance on the evaluation task if the linear layer was retrained on top of the same base network. Minimizing the self-supervised error encourages augmentation policies that bolster distinguishable image features.

Min. InfoNCE: $\mathcal{T}^{\text{I-min}} = \operatorname{argmin}_\mathcal{T} \mathcal{L}_{\text{NCE}}(\theta_\mathcal{M}|\mathcal{T}(D_\mathcal{A}))$ 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. It is worth noting that a trivial way to distinguish image pairs is to use weak augmentations so paired images have high similarity.

Max InfoNCE: $\mathcal{T}^{\text{I-max}} = \operatorname{argmin}_\mathcal{T} - \mathcal{L}_{\text{NCE}}(\theta_\mathcal{M}|\mathcal{T}(D_\mathcal{A}))$ negates the previous loss function, yielding policies that make it difficult to distinguish image pairs in the feature space. Optimizing this loss function encourages a challenging augmentation policy, which may be overly challenging for training a network to learn meaningful representations.

Min $\mathcal{L}_{ss}$ max $\mathcal{L}_{\text{NCE}}$: $\mathcal{T}^{\text{minmax}} = \operatorname{argmin}_\mathcal{T} \mathcal{L}_{ss} - \mathcal{L}_{\text{NCE}}$ yields policies that simultaneously maximize InfoNCE, encouraging challenging augmentation policies, and minimize $\mathcal{L}_{ss}$, encouraging distinguishable image features.

Retrain MoCo using the full training dataset and augmentation policy: SelfAugment uses the selected policy from the loss functions and then retrains from scratch on the full dataset $\mathcal{D}_{train}$. It is worth noting that because the augmentation policy learned from SelfAugment is used for an instance contrastive task, and not for the evaluation task, the augmentations that minimize the self-supervised evaluation loss are not necessarily the best augmentations for instance contrastive pre-training. Rather, this method provides the set of augmentations that would lead to high evaluation performance if the linear layer were directly retrained on top of the backbone used during augmentation selection [36]. Hence, incorporating the InfoNCE loss balances the instance contrastive task with the downstream task; we observe strong empirical evidence for this in §4 and Appendix G.

4. Experiments

Through the following experiments, we aim to establish that **(i)** a self-supervised evaluation task is highly correlated with the supervised performance of standard visual recognition tasks (image classification, object detection, and few-shot variants) on common datasets (CIFAR-10 [10], SVHN [11], ImageNet [12], PASCAL [13], COCO [14], Places-205 [15]) and **(ii)** SelfAugment provides a competitive approach to augmentation selection, despite being fully unsupervised. All experiments used MoCo training [4] with the standard ResNet-50 backbone [39] on 4 GPUs, using default training parameters from [3], see Appendix C.

4.1. Self-supervised evaluation correlation

As evaluating all possible data augmentations for every dataset, training schedule, and downstream task is intractable, we evaluate a diverse sampling of RandAugment, SelfAugment, MoCoV2, and single-transform policies and training schedules. For CIFAR-10 [10], SVHN [11], and ImageNet [12] we use augmentation policies: **(i)** random horizontal flip and random resize crop, **(ii)** RandAugment on top
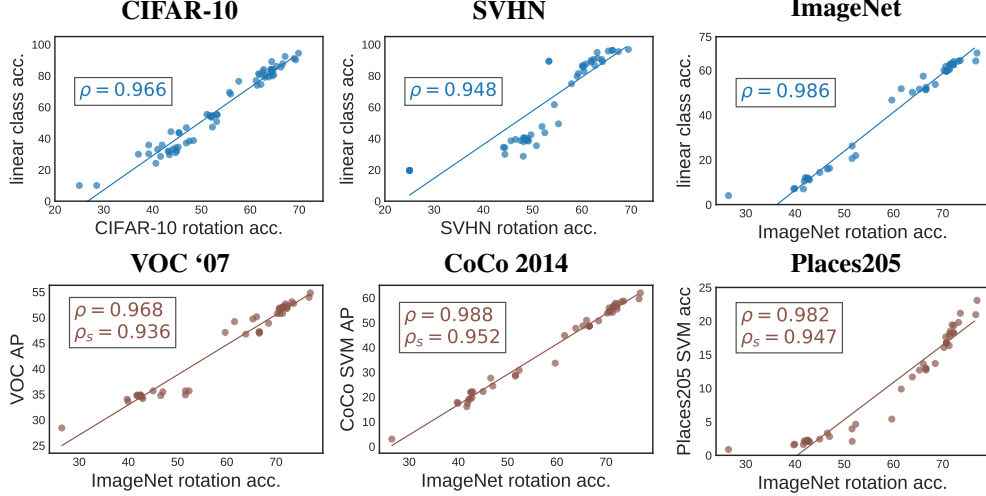
Figure 2: Top row: Correlation between supervised linear classification accuracy and linear image rotation prediction accuracy for three datasets. Bottom row: Correlation between rotation prediction and three transfer tasks from the ImageNet pretraining. ρ is the rank correlation between rotation prediction and the supervised task. ρ_s is the rank correlation between ImageNet supervised linear classification and the transfer task performance.

of (i) grid searched over parameters $\lambda = \{4, 5, 7, 9, 11\}, N_\tau = \{1, 2, 3\}$ at $\{100, 500\}$ epochs for CIFAR-10/SVHN and $\lambda = \{5, 7, 9, 11, 13\}$, $N_\tau = 2$ at $\{20, 60, 100\}$ epochs for ImageNet, **(iii)** each individual RandAugment transformations at 100 and 10 epochs for CIFAR-10 and ImageNet, respectively, **(iv)** scaling the magnitude λ from its min to max value for each N_τ at 500 epochs for CIFAR-10/SVHN and $\{20, 60, 100\}$ epochs for ImageNet, **(v)** for CIFAR-10/SVHN we also performed RandAugment using $K_T = \{3, 6, 9\}$ transformations at each of $\lambda = \{4, 7\}$ with $N_\tau = 2$, at 500 epochs, **(vi)** MoCoV2 augmentations at 100, 200 epochs for ImageNet, **(vii)** the five SelfAugment policies at 750 epochs for CIFAR-10/SVHN and 100 epochs for ImageNet. In total, this yields a diverse set of 61 models for CIFAR-10/SVHN and 43 models for ImageNet.

Linear Evaluation: We first compare the rotation, jigsaw, and colorization evaluation tasks on the models obtained by MoCo pre-training with the above augmentation policies (full details in Appendix C). We compute the Spearman rank correlation [40] between the top-1 supervised linear classification accuracy and each evaluation task, where a higher correlation indicates a better evaluation task. As shown below, the rotation prediction task has a uniformly higher correlation with the supervised linear classification compared to the jigsaw and colorization tasks:

Evaluation	CIFAR-10 (ρ)	SVHN (ρ)	ImageNet (ρ)
Rotation	**0.966**	**0.948**	**0.986**
Jigsaw	0.919	0.904	0.881
Colorization	0.612	0.806	0.627

The rotation evaluation correlations indicate a very strong relationship with the supervised evaluations, and based on its improvement over the jigsaw and colorization evaluations, we focus our main experiments, ablations, and SelfAugment/SelfRandAugment implementation of the automatic augmentation algorithms on this evaluation task. Appendix D and D.4 contain further details and analyses between the self-supervised evaluation tasks, where for instance, we also observe that the network activations from the rotation layer are more similar to the activations from the supervised classification layer compared to the other evaluations. We note that a rotation-based evaluation will not work for rotation invariant images (similarly, a color-based evaluation will not work for black-and-white images), and discuss this direction for future work in Appendix D.

The top row of Figure 2 shows scatterplots of the correlation between the supervised and self-supervised evaluations for the rotation evaluation task, where the poorer performing models come from single transformation augmentation policies and early evaluation schedules, and the better performing models come from the RandAugment, SelfAugment, and MoCoV2 augmentation policies. We observe that the rank correlation is maintained for both the poor and strong performing models, indicating that the rotation evaluation can be used across a wide range of model performance.

Transfer Learning: A central goal of representation learning is to learn transferable features. We therefore study the correlation between rotation evaluation and the ImageNet transfer performance for the following datasets/tasks:

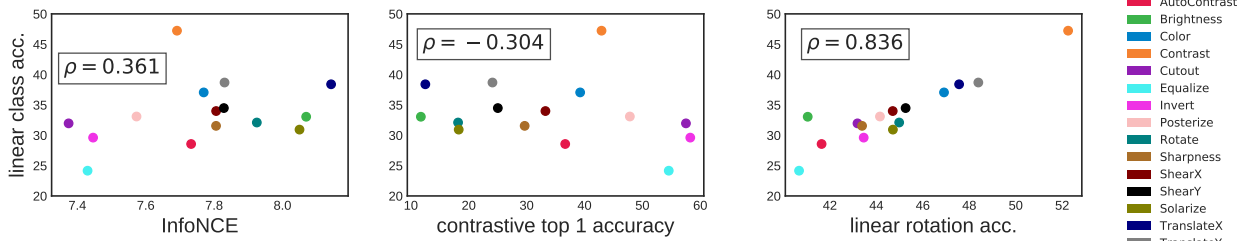**VOC07** [13] **object detection**: Following the specifics

Figure 3: For SVHN, 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.

from [3], we transfer the pre-trained models to a Faster R-CNN R50-C4 model and fine-tune all layers. Over three runs, we evaluate the mean results on VOC07 using the challenging COCO metric, $AP_{50:95}$, and report these results in Table 1. Further, we report the AP_{50}/AP_{75} in Appendix D. Fine-tuning is performed on the `train2007+2012` set and evaluation is on the `test2007` set.

COCO2014 [14] multi-class image classification Following [41], we train linear SVMs [42] on the frozen network and evaluate the accuracy over three end-to-end runs.

Places205 [15] low shot scene classification: Following [41], we train linear SVMs on the frozen network using $k = \{1, 4, 8, 16, 32, 64\}$ labeled examples and evaluate the accuracy over five runs, with the average across all k used as the evaluation criteria, see Appendix D for a breakdown.

For VOC07, COCO2014, and Places205, the bottom row of Figure 2 shows the ImageNet rotation performance vs the transfer task performance, yielding strong rank correlations of $\rho = \{0.968, 0.988, 0.982\}$, respectively. For comparison, the rank correlation of the supervised linear classification on ImageNet is $\rho_s = \{0.936, 0.952, 0.947\}$. For each transfer task, the rotation correlation is stronger than the supervised correlation. In [4], the authors observe that "linear classification accuracy is not monotonically related to transfer performance in detection," an observation that we find further evidence for across more transfer tasks. Furthermore, we observe that rotation prediction has a stronger transfer correlation than linear classification.

Finding the best individual image transformations Similar to the exhaustive evaluation of single-transform augmentation policies performed in [1], Figure 3 shows the performance of single-transform policies for SVHN (additional datasets in Appendix D). 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.

Selecting augmentation policies across architectures In [30], the authors showed that when training an entire network to classify image rotations, rather than just a linear layer, the rotation prediction performance did not correlate across architectures. We study this same problem but using only a linear evaluation layer. Specifically, for CIFAR-10 we study the rotation and supervised evaluation correlation for ResNet18, ResNet50, Wide-ResNet-50-2, using RandAugment with a grid search over the number of transformations $N_\tau = \{1, 2, 3\}$ and magnitudes $\lambda = \{4, 5, 7, 9, 11\}$ evaluated after 100 epochs. Figure 4 shows the results for each architecture: the overall Spearman rank correlation across all architectures is $\rho = 0.921$, between the Wide-ResNet-50-2 and ResNet18 Spearman correlations of 0.924 and 0.914 and less than the ResNet50 correlation of 0.957.
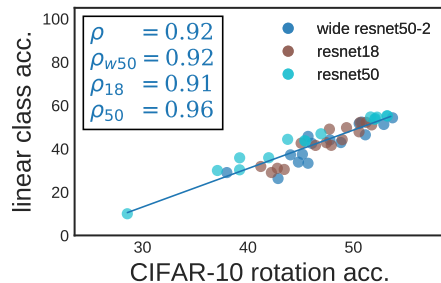

Figure 4: The Spearman rank correlation for CIFAR-10 RandAugment grid search for ResNet18, ResNet50, and Wide-ResNet-50-2, as well as the combined rank correlation.

Overall, these strong correlations indicate that a linear ro-

Table 1: Top-1 accuracy of SelfAug on CIFAR-10, SVHN, ImageNet, as well as ImageNet transfer tasks and semi-supervised experiments. An "R" superscript denotes best rotation accuracy. Bold results are greater than one standard deviation better across multiple runs, see Appendix C. Without using labels or hand-tuning hyperparameters, SelfAugment results in better performance than MoCoV2 for 2 out of 3 benchmark datasets it was directly trained on and comparable transfer performance.

	Self-Supervised			Transfer			Semi-Supervised	
unsup. feedback	C10	SVHN	IN	VOC	COCO	Places	IN-1%	IN-10%
Base Aug	89.1	89.2	46.7	47.1	33.6	5.4	16.0	33.6
SelfRandAug	90.3	**96.8**^{R}	64.1	53.1	58.4	19.8	36.1	53.4
SelfAug (min rot)	91.0	94.9	57.4	50.2	50.9	13.6	26.4	45.1
SelfAug (min Info)	87.5	86.0	51.7	49.2	44.80	9.9	20.5	38.6
SelfAug (max Info)	90.1	96.2	63.3	52.6	57.8	19.4	34.4	52.7
SelfAug (minimax)	**92.6**^{R}	95.8	**64.4**	53.0	58.7	**21.2**	36.2	54.1
supervised feedback								
MoCoV2 [4]	92.3	96.4	64.2^R	**54.0**	**59.6**	20.8	**37.9**^{R}	**54.9**^{R}

tation prediction evaluation is effective across architectures. In Appendix D.2, we show that a drop in correlation occurs when using a two-layer MLP for rotation evaluation instead of a linear layer. Combined with the lack of correlation discovered when using a full network in [30], we surmise that using a *linear evaluation layer* to classify rotations is important as it disentangles the learned representations from the ability of the network to learn its own rotation features.

4.2. Performance benchmarks

Here, we evaluate the SelfAugment and SelfRandAugment augmentation policies with the goal of establishing comparable results to the state-of-the art policies obtained through supervised feedback.

Setup: We evaluate: **(i)** linear classification performance on top of the frozen network using CIFAR-10, SVHN, and ImageNet, **(ii)** transfer performance of ImageNet models on PASCAL VOC07 object detection, COCO2014 multi-class image classification, and Places205 few-show scene classification as described in the previous subsection, **(iii)** semi-supervised ImageNet top-1 accuracy with using only 1% or 10% of labels for linear training. All methods were evaluated with the same number of epochs and hyperparameters: 750 pre-train and 150 linear epochs for CIFAR-10/SVHN, 100 pre-train and 50 linear epochs for ImageNet, 24k fine-tuning iterations on VOC07, and the exact training parameters/schedules from [41] for COCO2014 and Places205. For SelfAugment, we used the settings from [36]: $P = 10$ policies, $T = 2$ transformations, and $K = 5$ folds of training. For SelfRandAugment, we used the grid search described in the previous subsection. See Appendix C for all details.

Linear classification: Table 1 compares all versions of SelfAugment to the MoCoV2 augmentation policies [4] that were based on the extensive study of supervised policy evaluations in [1]. For linear classification with CIFAR-10, SVHN, and ImageNet, SelfAugment policies outperform MoCoV2's policy. Where the largest gain occurs in the

SVHN dataset. SVHN, consisting of images of house numbers, is the most distinct from ImageNet's diverse, object centric images. Since MoCoV2's policy is the result of extensive, supervised study on ImageNet, it does not transfer as well to a distributionally distinct dataset such as SVHN. These results indicate that SelfAugment is a stronger approach to obtaining quality representations for datasets that are distributionally distinct from ImageNet.

Transfer and semi-supervised: For the VOC07 object detection and COCO2014 image classification transfer tasks, the MoCoV2 policy performed best, while SelfAugment had a stronger transfer performance for few-shot scene classification on the Places205 dataset at each k value (see Appendix C for all details). Like ImageNet, VOC07 and COCO2014 are natural images containing objects, while Places205 is a scene classification benchmark, consisting of complex scenes and diverse settings. SelfAugment's policies, as with the linear classification, perform better for the dataset and task that substantially differs from ImageNet. While the SelfAugment policy outperformed MoCoV2 for the linear classification with 100% of the labels used for training, MoCoV2 performed better when using only 1% and 10% of the labeled data for training the linear classifier. These results indicate that using supervised evaluation to select a policy can lead to strong semi-supervised performance on the same dataset, but as indicated by the Places205 results, this policy may not transfer to other semi-supervised tasks.

Rotation prediction: The mean rank correlation for the supervised linear classification and rotation prediction for all results in this subsection is $\rho = 0.978$, indicating that the strong correlation holds when removing the poorer performing models from the previous subsection. For every linear classification result except ImageNet, the top performing augmentation policy also corresponds to the top performing rotation prediction performance, as indicated with an "R" superscript. For ImageNet, MoCoV2's rotation prediction performance was significantly better than all other policies:

$76.7 \pm 0.2\%$ compared to $73.6 \pm 0.2\%$ for SelfAugment's minimax, over three linear evaluations. While for a similar analysis, the linear classification performance for MoCoV2 performed worse: $64.2 \pm 0.1\%$ compared to $64.4 \pm 0.1\%$.

As shown in the previous subsection, however, the rotation prediction has a stronger correlation to transfer performance than the supervised linear classification, and indeed, that is the case here: on the VOC07 and COCO2014 transfer tasks and the ImageNet 1% and 10% semi-supervised evaluations, the MoCoV2 policy significantly outperformed all other policies, while the SelfAugment minimax policy had the best transfer performance for the Places205 task. In other words, *across the transfer and semi-supervised evaluations, rotation prediction was more indicative of performance than supervised linear classification.*

5. Discussion

We have shown that a self-supervised rotation evaluation has a strong correlation with supervised evaluation (outperforming jigsaw/colorization tasks) and that this evaluation can be incorporated into an effective loss function for efficient augmentation selection (§4). Here, we further reflect on the utility of the self-supervised rotation evaluation task.

If rotation prediction is highly correlated with supervised evaluations, why not directly train on it? As discussed in Appendix D.2, the authors of [30] found that rotation accuracy from training a full network on rotation prediction was only weakly correlated with supervised performance. Furthermore, as discussed in Appendix D.2, using a 2-layer MLP for rotation prediction drops the correlation from $\rho = .966$ to $.904$ even though the prediction accuracy improves. This indicates that while rotation prediction using a linear combination of the learned representations is an effective *evaluation*, actually learning the representations via rotation prediction is not as strongly correlated.

Shouldn't minimizing rotation error yield the best policies? We evaluate policies only *after* contrastive training, so minimizing rotation error yields policies that improve rotation prediction if we were to retrain the linear classifier. However, as indicated by the poor performance of using the rotation-error as the loss function, it is important to also take the contrastive task into consideration when retraining the entire network. To explore this idea further, we minimize a supervised classification loss function for SelfAugment. Due to the strong correlation between the rotation task and supervised evaluation, this results in similar performance to minimizing rotation loss with SelfAugment: we observe an accuracy of 90.7 on CIFAR-10 and 94.9 on SVHN using supervised feedback (rotation evaluation yields an accuracy of 91.0 on CIFAR-10 and 93.7 on SVHN). This performance is worse than the loss functions that simultaneously maximize the InfoNCE loss, which encourage difficult augmentation policies (see Appendix G).

Which SelfAugment loss function should be used on a new, unlabeled dataset? When training with a new unlabeled dataset, we recommend starting with the minimax SelfAugment loss function for augmentation policy selection. This recommendation stems from its superior performance across all datasets and tasks. For comparison, the SelfAugment loss function minimizing the InfoNCE produced results that were often worse than the baseline policy, while both maximizing the InfoNCE or minimizing rotation prediction exceeded the baseline, but generally did not surpass MoCoV2's policy. In Appendix G, we show the effective magnitude of each individual transformation for each loss functions. We see that, as expected, the minimax loss function produces policies with intermediate magnitudes across all datasets. Its strong performance indicate that these intermediate magnitudes have found a *sweet spot* where the augmentations strike a balance between creating difficult instance contrastive tasks and retaining salient image features.

It is worth noting that across both SVHN and CIFAR-10, SelfAugment with min InfoNCE loss function results in *worse* results than simply using the baseline augmentation. We believe this is because the selected augmentations make it easier to tell samples apart using weak augmentations that do not actually add meaningful contrastive tasks (see Appendix G). These augmentations minimize InfoNCE but do not appear to be helpful in improving the model's ability to generalize to new data. Accordingly, the model performs poorly when retrained with these augmentations.

Computational efficiency: Importantly, SelfAugment and RandAugment provide an efficient framework for finding augmentations for instance contrastive learning without using labels. Prior work [1] performed grid searches over various augmentation policies, requires training many models to completion. Training a single instance of MoCoV2 on ImageNet using 4 V100 GPUs for 100 epochs takes 52 hours (208 GPU hours). SelfAugment finds useful, and typically improved, augmentation policies in 730 GPU hours, and can learn additional augmentation policies in an additional 98 GPU Hours. (see Appendix H for details). This is less than the computational cost of training just four total instances. For comparison, [1] reported at least 49 full training runs.

6. Conclusion

In this paper, we identified the problem that self-supervised representations are evaluated using labeled data, oftentimes using the labels from the "unlabeled" training dataset itself. We established that a self-supervised image rotation task is strongly correlated with the supervised performance for standard computer vision recognition tasks, and as a result, can be used to evaluate learned representations. Using this evaluation, we establish two unsupervised data augmentation policy selection algorithms and show that they can outperform or perform comparably to policies obtained

using supervised feedback.

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 1, 2, 4, 6, 7, 8, 14, 15

[2] Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019. 1, 2

[3] 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. 1, 2, 4, 6, 11, 15

[4] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 1, 2, 4, 6, 7, 11, 15, 19

[5] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018. 1

[6] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *arXiv preprint arXiv:2005.10243*, 2020. 1

[7] H. Shin, M. Orton, D. J. Collins, S. Doran, and M. O. Leach. Autoencoder in time-series analysis for unsupervised tissues characterisation in a large unlabelled medical image dataset. In *2011 10th International Conference on Machine Learning and Applications and Workshops*, volume 1, pages 259–264, 2011. 1

[8] X Zhang, Y Cui, Y Song, H Adam, and S Belongie. The imaterialist challenge 2017 dataset. In *FGVC Workshop at CVPR*, volume 2, page 3, 2017. 1

[9] Zhengyuan Yang, Yixuan Zhang, Jerry Yu, Junjie Cai, and Jiebo Luo. End-to-end multi-modal multi-task vehicle control for self-driving cars with visual perceptions. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 2289–2294. IEEE, 2018. 1, 2

[10] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). 2, 4

[11] Ian J Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082*, 2013. 2, 4

[12] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 2, 4

[13] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html. 2, 4, 5

[14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2, 4, 6

[15] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014. 2, 4, 6

[16] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016. 2, 3

[17] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016. 2, 3

[18] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014. 2

[19] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010. 2

[20] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 2

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. 2

[22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. 2

[23] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2

[24] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2

[25] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 2

[26] Adam Coates and Andrew Y Ng. Learning feature representations with k-means. In *Neural networks: Tricks of the trade*, pages 561–580. Springer, 2012. 2

[27] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *CoRR*, abs/1803.07728, 2018. 2, 3

[28] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015. 2

[29] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations, 2018. 2

[30] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1920–1929, 2019. 2, 6, 7, 8

[31] Joan Puigcerver, Carlos Riquelme, Basil Mustafa, Cedric Renggli, André Susano Pinto, Sylvain Gelly, Daniel Keysers, and Neil Houlsby. Scalable transfer learning with expert models. *arXiv preprint arXiv:2009.13239*, 2020. 2

[32] Cedric Renggli, André Susano Pinto, Luka Rimanic, Joan Puigcerver, Carlos Riquelme, Ce Zhang, and Mario Lucic. Which model to transfer? finding the needle in the growing haystack. *arXiv preprint arXiv:2010.06402*, 2020. 2

[33] Connor Shorten and Taghi M. Khoshgoftaar. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1):60, July 2019. 2

[34] 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. 2, 3, 11, 12

[35] Daniel Ho, Eric Liang, Ion Stoica, Pieter Abbeel, and Xi Chen. Population based augmentation: Efficient learning of augmentation policy schedules. In *ICML*, 2019. 2, 3

[36] 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, 3, 4, 7, 11, 18

[37] 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. 2, 3, 11, 14

[38] Chenxi Liu, Piotr Dollár, Kaiming He, Ross Girshick, Alan Yuille, and Saining Xie. Are labels necessary for neural architecture search? *arXiv preprint arXiv:2003.12056*, 2020. 3

[39] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017. 4

[40] C Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, page 72, 1904. 5

[41] 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. 6, 7, 11

[42] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992. 6

[43] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com. 9

[44] 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. 18, 19

A. Notation and definitions

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 [36]
InfoNCE	See Eq 1
MoCo	Momentum Contrast, from [3]

B. Augmentation transformation details

Following [36, 37], we define the set of transformations used for SelfAugment and RandAugment, $\mathbb{O}$, as the PIL-based image transformations in Table 2. Each transformation has a minimum and maximum magnitude, λ, where for RandAugment, the entire range is discretized over 30 integers. See [37, 34] for further details and descriptions of each transformation.

C. Additional training and experiment details

CIFAR-10, SVHN, ImageNet: Table 5 lists the training and experimental parameters for CIFAR-10, SVHN, and ImageNet. The training parameters were taken from [3] and adjusted for 4 GPUs, i.e. the learning rate and batch size were scaled by 0.5 since [3] experiments were conducted on 8 GPUs. Consult [3] and [4] 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 [3], 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 [3], 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 4 displays the $AP/AP_{50}/AP_{75}$ breakdown for three fine-tunings.

COCO2014/Places205: Following [41], 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) 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 hyper-parameter 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 3 displays the Places205 results across five linears SVM trainings for all k values.

---

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

Table 2: PIL image transformations used for SelfAugment and RandAugment. The min and max magnitude values are taken and from [34]

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

Table 3: 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 4: 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 5: 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 [37], 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 (transformations listed in §3.2). In addition, we include the random resize crop transformation, as it was shown to be the best performing individual transformation in [1]. 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 [37], 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 [1]. 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 5 shows the the ImageNet rotation prediction correlations with transfer performance to VOC07 for all AP, AP_{50}, and AP_{75} evaluations. Figure 6 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.

---

[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.

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.

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 7 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 8 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 8. 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 [1], 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, [4] released a follow-up paper showing that the MoCo framework [3] 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 [4]:

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.
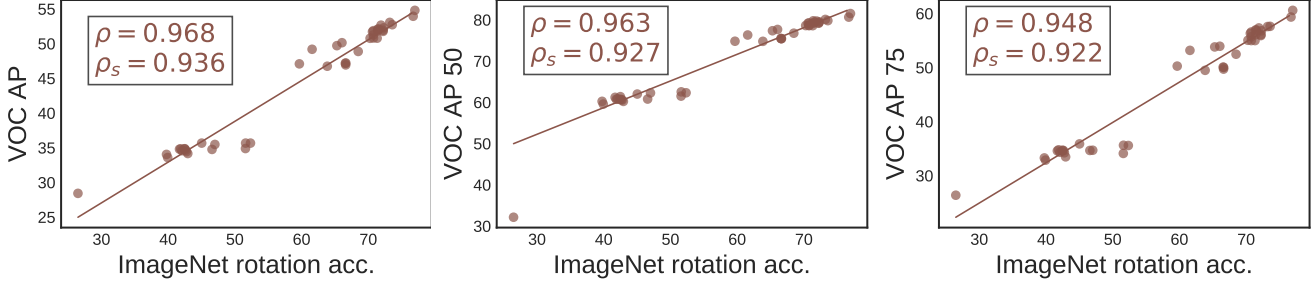
Figure 5: The ImageNet rotation prediction correlations with transfer performance to VOC07 for all AP, AP_{50}, and AP_{75} evaluations. The Spearman rank correlation with rotation prediction is indicated with ρ, while the rank correlation with the supervised ImageNet classification performance is indicated with ρ_s.
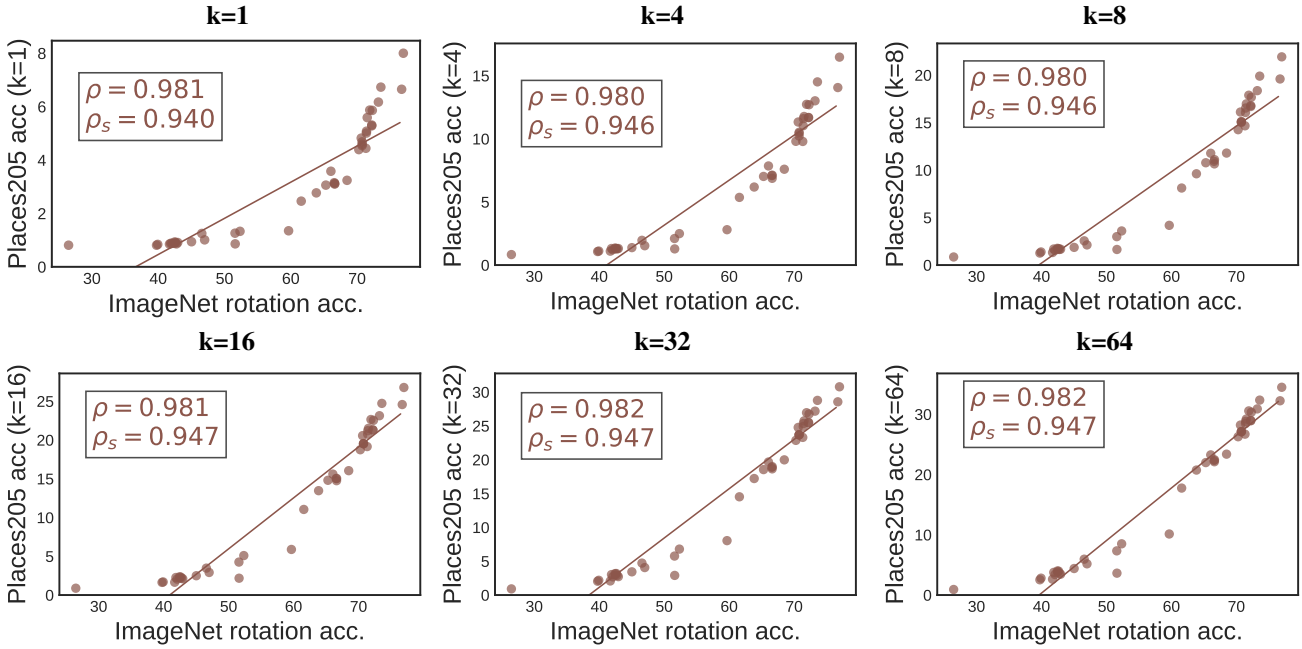

Figure 6: The ImageNet rotation prediction correlations with transfer performance to Places205 few label scene classification using $k = \{1, 4, 8, 16, 32, 64\}$ labels per scene class. The Spearman rank correlation with rotation prediction is indicated with ρ, while the rank correlation with the supervised ImageNet classification performance is indicated with ρ_s.
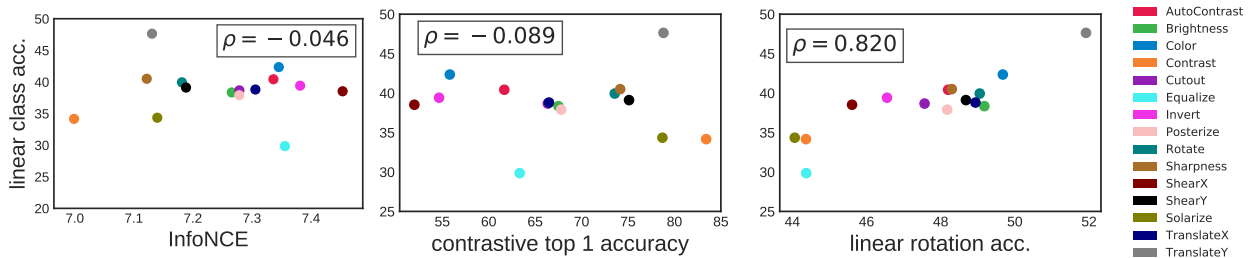

Figure 7: Similar to Figure 3, for CIFAR-10, we plot the supervised classification accuracy (y-axis) vs the InfoNCE loss function (left), contrastive top-1 accuracy (middle), and self-supervised linear rotation accuracy (right), for a self-supervised model trained using one of each transformation used 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.
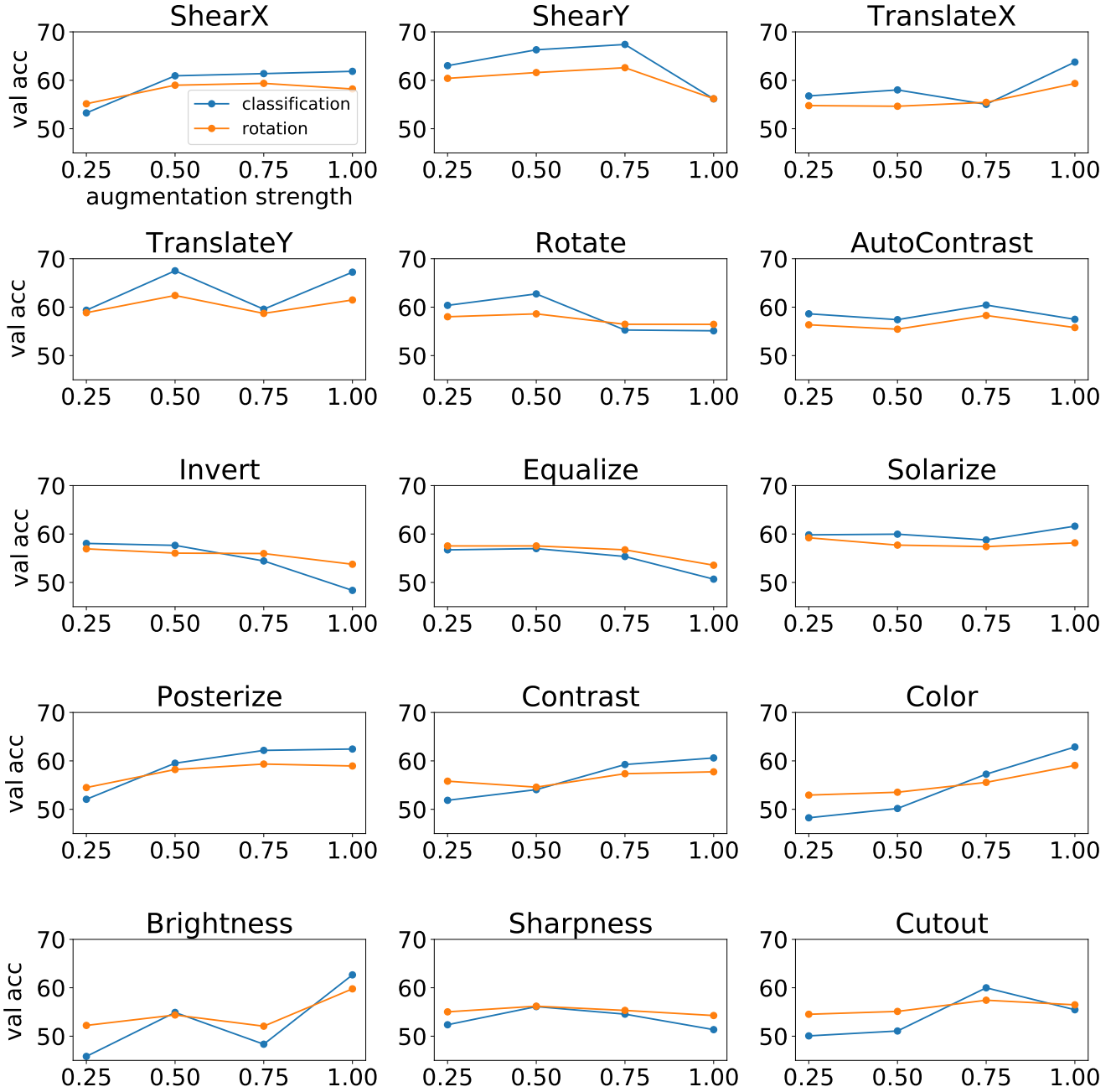
16

Figure 8: 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 [44]) 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:

$$RV(X,Y) = \frac{tr(XX'YY')}{\sqrt{tr[(XX')^2]tr[(YY')^2]}} \quad (3)$$

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.

$$RV_2(X,Y) = \frac{Vec(X\tilde{X}')'Vec(Y\tilde{Y}')}{\sqrt{Vec(X\tilde{X}')'Vec(X\tilde{X}') \times Vec(Y\tilde{Y}')'Vec(Y\tilde{Y}')}} \quad (4)$$

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 [44], 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 9, 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 [36], 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 [36], 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 [36], 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 2.

G. SelfAugment Loss Functions

In this section we discuss the loss functions used for Self-Augment in greater detail and their resulting augmentation policies which are summarized in Figure 10.

- **Min. eval error**:

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

where $\mathcal{L}_{SS}$ 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 10), resulting in relatively poor downstream performance (see Table 1).
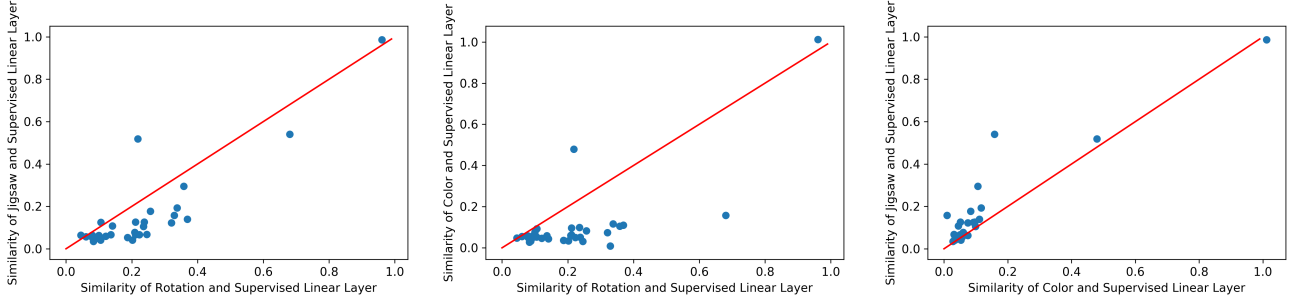

Figure 9: Comparisons of RV_2 Similarity [44] between each self supervised task studied and supervised linear layers. Rotation prediction layers have stronger similarity with supervised linear layers than either of the jigsaw and colorization self supervised tasks (p $< 2.3 \times 10^{-5}$, Wilcoxon 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 10. 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 (see Table 1).

- **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 (§4). 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 [4].

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. 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
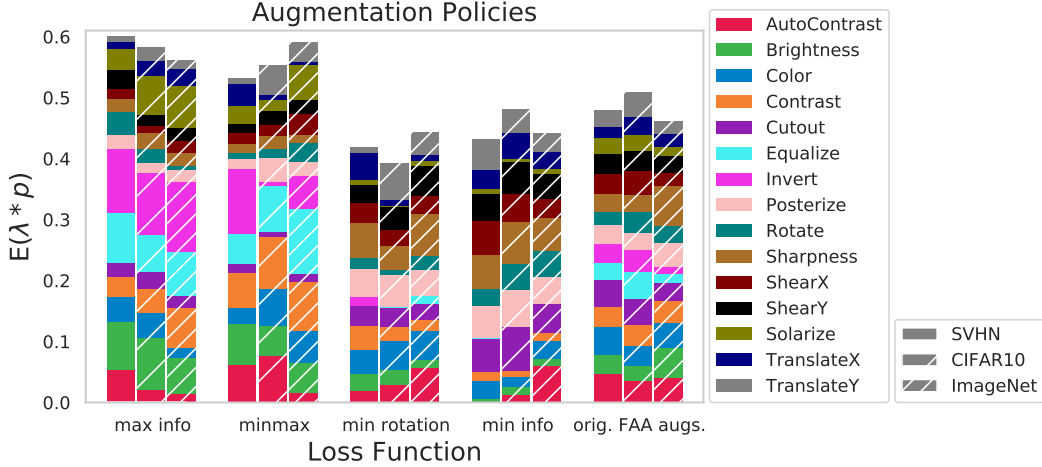
Figure 10: Visualization of the augmentation policies found with SelfAugment as well as Fast AutoAugment for supervised learning. This figure shows expected augmentation strength (mean magnitude×normalized probability) of each augmentation, evaluated across all datasets and loss functions. As expected, 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.

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 eval-

uation 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**