Tied-Augment: Controlling Representation Similarity Improves Data Augmentation

Emirhan Kurtulus [1 2] **Zichao Li** [3] **Yann Dauphin** [3] **Ekin D. Cubuk** [3]

Abstract

Data augmentation methods have played an important role in the recent advance of deep learning models, and have become an indispensable component of state-of-the-art models in semi-supervised, self-supervised, and supervised training for vision. Despite incurring no additional latency at test time, data augmentation often requires more epochs of training to be effective. For example, even the simple flips-and-crops augmentation requires training for more than 5 epochs to improve performance, whereas RandAugment requires more than 90 epochs. We propose a general framework called Tied-Augment, which improves the efficacy of data augmentation in a wide range of applications by adding a simple term to the loss that can control the similarity of representations under distortions. Tied-Augment can improve state-of-the-art methods from data augmentation (e.g. RandAugment, mixup), optimization (e.g. SAM), and semi-supervised learning (e.g. FixMatch). For example, Tied-RandAugment can outperform RandAugment by 2.0% on ImageNet. Notably, using Tied-Augment, data augmentation can be made to improve generalization even when training for a few epochs and when fine-tuning. We open source our code at `https://github.com/ekurtulus/tied-augment/tree/main`

1. Introduction

Data augmentation is an integral part of training deep neural networks to improve their performance by modulating the diversity and affinity of data (Gontijo-Lopes et al., 2020). Although data augmentation offers significant benefits (Simard et al., 2003; Krizhevsky et al., 2017; Shorten & Khoshgoftaar, 2019; Szegedy et al., 2015), as the complexity of the augmentation increases, so does the minimum number of epochs required for its effectiveness (Cubuk et al., 2019). As neural networks and datasets get larger, machine learning models get trained for fewer epochs (for example, Dosovitskiy et al. (2020) pretrained for 7 epochs), typically due to computational limitations. In such cases, conventional data augmentation methods lose their effectiveness. Additionally, data augmentation is not as effective when finetuning pretrained models as it is when training from scratch.

In this work, we present a general framework that mitigates these problems and is applicable to a range of problems from supervised training to semi-supervised learning by amplifying the effectiveness of data augmentation through feature similarity modulation. Our framework, Tied-Augment, makes forward passes on two augmented views of the data with tied (shared) weights. In addition to the classification loss, we add a similarity term to enforce invariance between the features of the augmented views. We find that our framework can be used to improve the effectiveness of both simple flips-and-crops (Crop-Flip) and aggressive augmentations even for few-epoch training. As the effect of data augmentation is amplified, the sample efficiency of the data increases. Therefore, our framework works well even with small amounts of data, as shown by our experiments on CIFAR-4K (4k samples from CIFAR-10), Oxford-Flowers102, and Oxford-IIT Pets.

Despite the simplicity of our framework, Tied-Augment empowers augmentation methods such as Crop-Flip and RandAugment (Cubuk et al., 2020) to improve generalization even when trained for a few epochs, which we demonstrate for a diverse set of datasets. For longer training, Tied-Augment leads to significant improvements over already-strong baselines such as RandAugment and mixup (Zhang et al., 2017). For example, Tied-RandAugment achieves a 2% improvement over RandAugment when training ResNet-50 for 360 epochs on ImageNet, without any architectural modifications or additional regularization.

[1]Stanford University [2]Cagaloglu Anadolu Lisesi [3]Google Research, Brain Team. Correspondence to: Emirhan Kurtulus <emirhank@stanford.edu>, Ekin D. Cubuk <cubuk@google.com>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

Our contributions can be summarized as follows:

- We show that adding a simple loss term to modulate feature similarity can significantly improve the effectiveness of data augmentation, which we show for a diverse set of data augmentations such as Crop-Flip, RandAugment, and mixup.
- Unlike conventional methods of data augmentation, with our framework, data augmentation can improve performance even when training for only a single epoch for finetuning pretrained networks or training from scratch on a wide range of datasets with different architectures.
- We compare Tied-Augment to multi-stage self-supervised learning methods (first pretraining, then finetuning on ImageNet). Our proposed framework is designed to be as straightforward as traditional data augmentation techniques, while avoiding the need for additional components such as a memory bank, large batch sizes, contrastive data instances, extended training periods, or large model sizes. Despite this simplicity, Tied-Augment can outperform more complex self-supervised learning methods on ImageNet validation accuracy.

2. Background / Related Work

2.1. Data Augmentation

Data augmentation has been a critical component of recent advances in deep vision models (He et al., 2022; Bai et al., 2022; Liu et al., 2021). We can divide data augmentation works into two categories: individual operations and optimal combinations of individual operations. In general, data augmentation operations are performed to expand the distribution of the input space and improve performance.

Random cropping and horizontal flips are widely used operations in image processing problems. This set of operations is usually extended by color operations (Szegedy et al., 2016; 2017). mixup (Zhang et al., 2017) is a method that uses a convex sum of images and their labels. This operation provides better generalization and robustness even in the presence of corrupted labels. Other operations include Cutout (DeVries & Taylor, 2017), a method that randomly masks out square regions within the image; Patch Gaussian (Lopes et al., 2019), an operation that combines Cutout with the addition of Gaussian noise by randomly adding noise to selected square regions; (Liu et al., 2016), a cropping strategy for object detection that generates smaller training samples by taking crops from an upscaled version of the original image; Copy-Paste (Ghiasi et al., 2021), an augmentation method that inserts random objects onto the selected training sample.

2.2. Self-supervised Learning

Self-supervised learning is a form of representation learning that usually makes use of pretext tasks to learn general representations (Ericsson et al., 2022). Generally, self-supervised learning methods follow a two-step paradigm. They first pretrain the network on a large dataset, then use it for finetuning on downstream tasks.

Clustering is the paradigm of mapping non-linear augmented views projections into a unit sphere of K classes (Bautista et al., 2016). This paradigm is notably widespread in image understanding problems (Caron et al., 2018; Asano et al., 2019; Caron et al., 2019; Gidaris et al., 2020). SwAV (Caron et al., 2020) is particularly noteworthy in this set of works. They cluster the data by enforcing consistency between the assigned clusters of the augmented views. Additionally, they propose *multi-crop* strategy, a random cropping strategy that not only two standard resolution crops but also N low resolution crops to take the features of varying resolutions into account.

Contrastive instance discrimination learns representations by pushing the features of positive instances, meaning augmented views of the same image or images with same classes, closer and pushing features of negative instances away (Hadsell et al., 2006). Currently, this is one of the most widely used paradigms.

MoCo (He et al., 2020) maintains a dictionary of encodings and views the problem as query matching. SimSiam (Chen & He, 2021) proposes to encode two augmented views of the same image, one with an MLP (multi-layer perceptron) head, and increase feature similarity. BYOL (Grill et al., 2020) follows the same method, but uses a network and another network following it by exponential moving average. SimCLR (Chen et al., 2020a) uses a network with an MLP head to encode two augmented views and maximizes similarity through contrastive loss (Hadsell et al., 2006). NNCLR (Dwibedi et al., 2021) improves on this approach by using clustering to maximize the number of correct negative instances. SupCon (Khosla et al., 2020) adapts this paradigm to supervised learning by following SimCLR and using contrastive loss, but selecting the correct positive and negative labels by using labels. SupCon showed that augmentation methods such as RandAugment with a supervised-contrastive loss can outperform the same data augmentation methods with a cross-entropy loss.

3. Tied-Augment

Tied-Augment framework combines supervised and representation learning in a simple way. We propose to enforce pairwise feature similarity between given augmented views of the same image while using a supervised loss signal. As shown in Figure 2, our framework consists of three compo-

```
# model: a neural network that returns
    features and logits
# tw: tied-weight
# augment1: a stochastic data augmentation
    module
# augment2: a stochastic data augmentation
    module
# note that augment1 can be the same as
    augment2

# ce = cross entropy loss
# mse = mean squared error loss

for x,y in loader:
  # generate two augmented views of the
      same image
  x1 = augment1(x)
  x2 = augment2(x)

  # extract features and logits
  f1, l1 = model(x1)
  f2, l2 = model(x2)

  # calculate loss
  ce_loss = (ce(l1, y) + ce(l2, y)) / 2
  feature_loss = mse(f1, f2)
  loss = ce_loss + tw * feature_loss
```

Figure 1. Python code for Tied-Augment based on NumPy.

nents:

- **Two stochastic data augmentation modules** (can be identical) produce two augmented views of the same image. These transformations can be chosen arbitrarily as long as they improve the performance of the baseline supervised model. However, in this work, we use the same augmentation for both branches for simplicity. Given two augmentations, we name the case after the more complex augmentation. For example, if RandAugment is used with Crop-Flip on the other branch, we name the case Tied-RandAugment. In Section 4 we provide a thorough analysis of the effects of the chosen data augmentation modules.

- **A neural network** generates features (pre-logits) and logits based on given an image. There are no architectural constraints as our framework is based on the pre-logit feature vector, which is used in all classification networks.

- **Pairwise feature similarity and supervised loss functions** enforce pairwise feature similarity/dissimilarity and supervised loss signal, respectively. In this work, we use L2 loss as the pairwise feature similarity function (we ablate this decision in 5) and, for simplicity, cross entropy loss as the supervised loss. The contribution of the feature-similarity term to the loss is controllable by the hyperparameter Tied-weight.

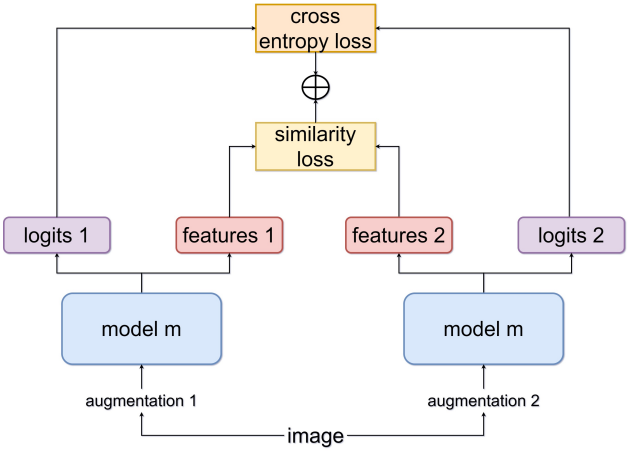

Figure 2. Tied-Augment framework.

The training of Tied-Augment works as follows. In each training iteration, we sample a batch of images of size N, and generate two augmented views, resulting in $2N$ images. For each image pair, we compute the feature similarity with L2 loss and for each image we calculate cross entropy loss. For given input x, logits $f(x)$, labels y, features of the first augmented views $v_1 = v_1(x)$, features of the second augmented views $v_2 = v_2(x)$, supervised loss ℓ, and the feature similarity loss weight w the loss function of Tied-Augment is:

$$\mathcal{L}_{\text{Tied-Aug}} = \sum_i \ell(f(v_i(x)), y) + w\|v_1(x) - v_2(x)\|^2 \quad (1)$$

In Algorithm 1, we provide an overview of our framework. In general, the views correspond directly to the feature representations $v_i = h_i = h(\text{aug}_i(x))$ where h is the function that produces the feature representation and $\text{aug}_i(\cdot)$ is the i-th augmentation function. However, we will also examine cases that require more elaborate views such as Tied-mixup.

3.1. Tied-FixMatch

In this section, we apply the Tied-Augment framework to FixMatch (Sohn et al., 2020) as a case study to demonstrate the easy adaptability of our framework. We refer to this version as Tied-FixMatch. FixMatch is a semi-supervised learning algorithm that combines consistency regularization and pseudo-labeling. For the labeled portion of the dataset, FixMatch uses standard cross-entropy loss. For the unlabeled images, FixMatch generates two augmented views of the same image using a weak (Crop-Flip) and a strong (RandAugment) transformation. Then, the model's predictions for the weakly augmented image are used as pseudo-labels

for the strongly-augmented image. However, predictions whose confidence is below a threshold are masked out and not used to calculate the unsupervised loss. FixMatch uses a standard cross-entropy loss denoted ℓ_s on the labeled images.

Considering that FixMatch already has a two-branch strategy for learning from unlabeled images, we can introduce Tied-Augment without any additional computational cost. In Tied-FixMatch, we change the objective as not only maximizing consistency and minimizing pseudo-labeling loss, but also minimizing the pairwise feature distance between augmented views of the same unlabeled images. In doing so, we also mask the instances with a confidence threshold and do not apply the pairwise feature similarity loss. Therefore, given features of the weakly-augmented unlabeled images h_1, features of the strongly-augmented unlabeled images h_2, and a similarity loss weight w, the loss minimized by Tied-FixMatch is simply:

$$\ell_s + \lambda_u \ell_u + w\|h_1 - h_2\|^2 \tag{2}$$

3.2. Tied-mixup

Here, we consider the application of Tied-Augment to mixup (Zhang et al., 2017). mixup is a popular data augmentation technique that produces augmented examples by convex combination of pairs of training points

$$\hat{x} = \lambda x_1 + (1-\lambda)x_2$$
$$\hat{y} = \lambda y_1 + (1-\lambda)y_2$$

where $\lambda \sim \text{Beta}(\alpha, \alpha)$ is a mixing coefficient sampled from a Beta distribution with parameter α.

Unlike the previously considered augmentations, different mixup augmented views have different labels in general. Applying Tied-Augment to mixup requires defining a better correspondence between the two augmented views. We propose the following

$$\Omega_M(h) = w\|\lambda h(x_1) + (1-\lambda)h(x_2) - h(\hat{x})\|^2. \tag{3}$$

In order to produce features that are in the same space as the first view of mixed examples $v_1 = h(\hat{x})$, this approach mixes the features of the clean examples to produce the second view $v_2 = \lambda h(x_1) + (1-\lambda)h(x_2)$. In effect, this is encouraging the features of the model to be linear in-between training points.

3.3. Tied-SAM (Sharpness Aware Minimization)

Sharpness-Aware Minimization (SAM) (Foret et al., 2020) is a widely-used training strategy that consists of two steps. At the first step, SAM applies an adversarial perturbation to first place the weights at the highest point in the loss landscape. Then, in the second step, this results in a move to a wider minimum. Tied-SAM augments this algorithm by boosting the adversarial move through pushing features of the augmented views apart (negating the Tied-weight) in the first step. In doing so, enable SAM to find a better adversarial loss landscape location. For the second step, we apply standard Tied-Augment to move to an even wider minimum.

3.4. Understanding Tied-Augment

We can gain some insight into TiedAugment by considering its application to Gaussian input noise augmentation. The additive regularization for Tied-GaussianNoise is given by

$$\Omega_G(f) = wE_\epsilon[\|h(\mathbf{x}) - h(\mathbf{x}+\epsilon)\|^2]$$

where h produces the features of the network, $\mathbf{x} \in \mathbb{R}^n$, and $\epsilon \sim \mathcal{N}(0, \sigma)^n$. Consider the approximation of this term using the first-order Taylor expansion of h at $\mathbf{x}$, $\Omega_G(h) \approx w\sigma^2\|\nabla h(\mathbf{x})\|_F^2$. This additive regularization is part of the well known class of Tikhonov regularizers (Tikhonov & Arsenin, 1977; Bishop, 1995) that include weight decay. It encourages the feature mapping function to become more invariant to small corruptions in the input, which can be beneficial for generalization. For a more detailed analysis of Tied-Augment, please refer to Appendix 8.7.

4. Experiments

To show the effectiveness of Tied-Augment, we experiment with training from-scratch on CIFAR-10, CIFAR-100, and ImageNet. We extend these tests with finetuning and few-epoch / low-data regimes to simulate more realistic scenarios, where the amount of domain-specific data or available compute is limited. Lastly, we show that Tied-Augment significantly improves the performance of state-of-the-art methods (e.g. mixup and SAM) and can be used for semi-supervised learning (e.g. FixMatch). For all models that use RandAugment, we show its configuration as "RandAugment(N=number of layers, M=magnitude, P=probability)". If probability is not given, it is set to the default of 1.0.

4.1. CIFAR-10 and CIFAR-100, CIFAR-4K

CIFAR-10 and CIFAR-100 are widely studied datasets, and CIFAR-4K is a benchmark intended to simulate the low-data regime. All baselines and Tied-Augment models include random pad-and-crops and flips (CF). RandAugment baselines and Tied-RandAugment models also include Cutout (DeVries & Taylor, 2017). For RandAugment experiments, we copy the reported optimal number of layers (N) and magnitude (M) for both augmentation branches, decoupling the hyperparameter search space from augmentation selection. We did not find an additional improvement from tuning the RandAugment on the second branch (e.g. RandAugment(N=2, M=14) for one branch, and RandAugment(N=2,

	CF	Tied-CF	RA	Tied-RA
CIFAR-10				
WRN-28-2	94.9	**95.5**	95.8	**96.9**
WRN-28-10	96.1	**96.5**	97.3	**98.1**
CIFAR-100				
WRN-28-2	75.4	**76.9**	79.3	**80.4**
WRN-28-10	81.2	**81.6**	83.3	**85.0**
CIFAR-4K				
WRN28-2	82.0	**82.5**	85.3	**87.8**
WRN28-10	83.5	**84.5**	86.8	**90.2**

Table 1. **Test accuracy (%) on CIFAR-10, CIFAR-100, CIFAR-4K, and ImageNet datasets.** We compare Tied-Augment to Crop-Flip (CF) and RandAugment (RA) baselines. Reported results are the average of 5 independent runs. Standard deviation of the results for each experiment is smaller than or equal to 0.1%.

	#epochs	Identity	Baseline (CF)	Tied-CF	Baseline (RA)	Tied-RA
CIFAR-10	1	72.6	70.3	**73.3**	72.8	**73.4**
	2	76.1	75.1	**82.8**	81.4	**82.4**
	5	89.2	88.5	**89.5**	88.6	**89.5**
	10	91.8	91.8	**92.2**	91.0	**92.5**
CIFAR-100	1	26.9	24.6	**28.1**	22.8	**29.4**
	2	41.4	39.4	**42.6**	41.7	**43.9**
	5	62.4	60.6	**62.8**	61.7	**62.2**
	10	70.8	70.4	**71.2**	71.2	**71.7**

Table 2. Test accuracy (%) for few-epoch training on CIFAR datasets. Reported results are the average of 10 independent runs. For 1, 2, 5, 10 epochs, standard deviations are below 0.5, 0.3, 0.2, and 0.1 respectively.

#epochs	Identity	Baseline CF	Tied-CF	Baseline (RA)	Tied-RA
Cars					
2	69.0	59.9	**69.5**	58.7	**69.4**
5	80.9	81.6	**84.7**	81.4	**84.6**
10	82.0	86.7	**88.3**	87.1	**89.2**
25	82.0	88.9	**89.4**	90.4	**91.5**
50	82.3	89.6	**90.0**	91.5	**92.2**
Flowers					
2	56.6	47.1	**56.8**	47.2	**56.5**
5	88.3	86.4	**88.7**	84.7	**88.7**
10	90.7	91.6	**93.3**	92.1	**93.5**
25	91.8	93.9	**94.1**	93.5	**94.3**
50	92.2	93.6	**94.5**	94.1	**95.1**
Pets					
2	91.4	91.4	**92.1**	91.4	**92.0**
5	92.4	92.8	**93.1**	92.1	**93.0**
10	92.5	93.1	**93.3**	92.9	**93.2**
25	92.9	93.4	**93.7**	93.4	**93.6**
50	92.8	93.5	**93.8**	93.5	**93.7**
Aircraft					
2	44.2	34.1	**41.8**	31.6	**40.8**
5	58.2	51.1	**58.3**	50.6	**58.1**
10	59.3	60.6	**61.9**	60.7	**61.5**
25	61.2	68.8	**69.9**	72.3	**74.6**
50	62.3	71.6	**72.3**	74.2	**76.1**
CIFAR-10					
2	95.7	95.2	**95.9**	95.1	**95.9**
5	96.4	96.3	**96.8**	96.3	**96.8**
10	96.5	96.8	**97.1**	96.8	**97.2**
25	96.6	97.2	**97.4**	97.3	**97.6**
50	96.6	97.2	**97.4**	97.6	**97.8**

Table 3. **Finetuning experiments on Stanford-cars, Oxford-Flowers102 (Flowers), Oxford-IIIT Pets (Pets), FGVC Aircraft (Aircraft), CIFAR-10 datasets.** Reported results for the 2, 5 and 10 epoch experiments are the average of 10 independent runs, while the rest is the average of 5 independent runs. Baseline results are the maximum of standard training and double augmentation branch with no similarity loss. pretrained model is a standard ResNet-50, Tied-Augment is only used for finetuning. Best CF (CF vs. Tied-CF) and RA (RA vs. Tied-RA) results are bolded. The standard deviations of the accuracies are smaller than or equal to 0.5%, 0.4%, 0.2%, 0.1%, and 0.1% for 2, 5, 10, 25, and 50 epochs respectively.

M=19) for the second). We also experimented with Stacked-RandAugment (Tian et al., 2020) and SimAugment (Chen et al., 2020a) on the second branch but saw no performance improvement over standard RandAugment. On both CIFAR-10 and CIFAR-100, we use the same data augmentation pairs for Wide-ResNet-28-10 and Wide-ResNet-28-2. All models are trained using the hyperparameters from RandAugment (Cubuk et al., 2020).

Additionally, we measure the efficacy of Tied-Augment on CIFAR-4K. We randomly sample 400 images from each class for training and leave the test set as is. We use the same hyperparameters as Cubuk et al. (2020) including training for 500 epochs. We use the same optimal setting of RandAugment(N=2, M=14) on both branches. As shown in Table 1, Tied-Augment improves both Crop-Flip and RandAugment by a significant amount, on all CIFAR datasets considered. We report all the hyperparameters in Appendix 8.2.

4.2. Few-epoch training

Previous work has shown that data augmentation is only able to improve generalization when the model is trained for more than a certain number of epochs. Usually, more complex data augmentation protocols require more epochs. For example, Cubuk et al. (2019) reported that more than 90 epochs was required to be able to search and apply AutoAugment policies. Similarly, Lopes et al. (2019) reported that none of the tested augmentation transformations was helpful when trained for only 1 epoch, even for simple augmentations such as flips or crops. To test how much of this problem can be mitigated by Tied-Augment, we evaluate our method on CIFAR-10 and CIFAR-100 for $\{1, 2, 5, 10\}$ epochs. For runs with epoch=$\{1, 2, 5\}$, the learning rate and weight-decay were tuned to maximize the validation accuracy of the identity baseline (since in this regime identity baseline outperforms the Crop-Flip baseline). The learning rate and weight-decay hyperparameters for the 10 epoch models were tuned to maximize the validation set performance of the Crop-Flip baseline.

To ensure fairness by eliminating the possibility of doubled epochs introduced by the two forward passes of our framework, in all reported tasks, the baselines performances are the maximum of standard training (no similarity loss and single augmentation branch) and double augmentation branch (with variable augmentation methods) with no similarity loss. Unlike our 200 epochs CIFAR experiments, we do not use the same augmentation for both branches but allow both the baseline and the Tied-Augment model to combine any one of the following augmentation methods: RandAugment(N=1, M=2), RandAugment(N=2, M=14), Crop-Flip, and identity. If one of the branches use RandAugment, for instance RandAugment for one branch and identity for the other, then it is only compared to RandAugment runs.

In Table 2, we show that Tied-Augment can outperform identity transformation in the epoch regimes as small as 2. Unconventionally, Tied-Augment is capable of pushing RandAugment to the level of Crop-Flip and identity, and even outperforming them in the $\{2, 5, 10\}$ epochs regimes. For all the epoch regimes, Tied-Augment outperforms its baseline significantly, up to 6.7%.

In addition to training networks from scratch for a limited number of epochs, finetuning for a few epochs is also an important problem given the ever-growing trend to go deeper for neural networks. Therefore, we test our framework on finetuning tasks where data augmentation is considerably less effective than from-scratch training. For this purpose, we train a ImageNet-pretrained ResNet-50 (He et al., 2016) model on Stanford-Cars (Krause et al., 2013), Oxford Flowers (Nilsback & Zisserman, 2008), Oxford Pets (Parkhi et al., 2012), FGVC Aircraft (Maji et al., 2013), and CIFAR-10 (Krizhevsky et al., 2009). Table 3 compares the performance of our framework to the baselines models. It is evident that, like our from-scratch experiments, Tied-Augment is able to outperform identity with not only a weak augmentation like Crop-Flip but with RandAugment. On all the finetuning datasets we experimented with, Tied-Augment consistently and significantly improves the baseline, up to 10.7%.

4.3. Image classification on ImageNet

We train ResNet-50 and ResNet-200 architectures (He et al., 2016) on the ImageNet dataset (Deng et al., 2009) with RandAugment. Previous work had shown that aggressive data augmentation strategies such as AutoAugment or RandAugment do not improve generalization if trained only for 90 epochs. To see if Tied-Augment can fix this issue, we train with Tied-RandAugment on ResNet-50 for 90 epochs. To see the full benefit of aggressive data augmentation, we also train Tied-RandAugment models for 360 epochs. Finally, to see the impact our approach on simple augmentations, we train the standard ResNet-50 with the standard Crop-Flip baseline and our Tied-CropFlip. Finally, to test the impact of Tied-Augment on a larger model, we train ResNet-200 for 180 epochs. We train ResNet-200 for fewer epochs to compensate for its larger compute requirement. We do not observe an improvement on the baseline ResNet-200 models when training for longer. All ResNet models use the standard training hyperparameters for ResNet, listed in Appendix Section 8.1.

ImageNet	#epochs	CF	Tied-CF	RA	Tied-RA
ResNet-50	90	76.3	**77.0**	76.3	**78.2**
ResNet-50	360	76.3	**76.9**	77.6	**79.6**
ResNet-200	180	78.5	**79.7**	80.0	**81.8**

Table 4. **ImageNet results.** CF and RA refer to Crop-Flip and RandAugment, respectively. ResNet-200 baselines do not improve when trained for more than 180 epochs. Standard deviations for the reported results are smaller than or equal to 0.2%.

In Table 4, we find that Tied-RandAugment is able to improve top-1 accuracy by almost 2% when trained for 90 epochs, and significantly reduces the number of epochs required for RandAugment to be effective, whereas regular RandAugment requires more than 90 epochs to improve generalization. When trained for 360 epochs, Tied-RandAugment still improves on RandAugment by 2%, totalling a 3.3% improvement over simple Crop-Flip. We also observe that Tied-CropFlip outperforms regular Crop-Flip in every setting.

To evaluate Tied-Augment on a different data augmentation method, we trained Resnet-50 networks with the same setup with mixup. We cross-validate the mixup coefficient α within the values $\{0.2, 0.3, 0.4\}$, and the similarity loss weight within $\{1, 50, 100\}$. Our mixup baseline achieves an top-1 accuracy of 77.9%. When we apply our simple Tied-Augment framework to mixup, Tied-mixup achieves 78.8%, an almost 1% improvement over an already strong baseline.

Since the Tied-Augment loss has a supervised and an unsupervised term, we compare Tied-Augment to relevant self-supervised methods that utilize all the training labels of ImageNet in addition to self-supervised training on ImageNet samples. We find that even though Tied-RandAugment is trained for fewer epochs without the need for multiple stages of training, Tied-RandAugment outperforms other methods for both ResNet-50 and ResNet-200 (Table 5).

4.4. Transferability of Tied-Augment Features

We finetune a Tied-ResNet50 to downstream datasets to measure the transferability of its features and compare it to BYOL (Grill et al., 2020), SimCLR-v2 (Chen et al., 2020b), and SwAV (Caron et al., 2020). We follow the SSL-Transfer (Ericsson et al., 2021) framework for our finetuning experiments. Namely, we finetune for 5000 step using a batch

	Epochs	Multi-stage	Top-1
ResNet-50			
SimCLR	1000	✓	76.0
SimCLR v2	800	✓	76.3
BYOL	1000	✓	77.7
SupCon	350	✓	78.7
Tied-RandAugment	360	✗	**79.6**
ResNet-200			
SupCon	700	✓	81.4
Tied-RandAugment	360	✗	**81.8**

Table 5. **Comparison of our method to self-supervised models.** Multi-stage denotes the need for separate pretraining and finetuning stages. Note that Tied-Augment methods do not require a pretraining stage. Performances of the self-supervised models are their 100% ImageNet finetuned results. Results reported are the average of 5 independent runs. The standard deviations are smaller than or equal to 0.2% for all reported results.

size of 64, SGD with Nesterov momentum (Sutskever et al., 2013), doing a grid search over learning rate and weight decay. We choose the learning rate from 4 logarithmically spaced values between 0.0001 and 0.1. Weight decay is chosen from 4 logarithmically spaced values between 10^{-6} and 10^{-3} as well as 0.

In Table 6, we compare the performance of Tied-Augment to self-supervised models and the supervised baseline. Our model outperforms SwAV (Caron et al., 2020) by 0.8% and the supervised baseline by 1.6%. This shows that the features learned by Tied-Augment are more transferable than its self-supervised and supervised counterparts. It is worth noting that a Tied-RandAugment model finetuned using Tied-CropFlip significantly improves an already strong performacee (0.9%).

4.5. Tied-FixMatch

To back up our claim that we offer a framework that can be used for a wide-range of problems, we apply Tied-Augment to a semi-supervised learning algorithm: FixMatch (Sohn et al., 2020). We compare the performance of our framework to the baseline exactly following the hyperparameters of the original work, without changing the augmentation pair of the unsupervised branch or adding the similarity term to the supervised branch. We use Wide-ResNet-28-2 and Wide-ResNet-28-8 configurations for CIFAR-10 and CIFAR-100 respectively. For the unsupervised branch, we use Crop-Flip for the weak branch and RandAugment(N=2, M=10, probability=0.5) for the strong branch, while supervised branch uses Crop-Flip. For our CIFAR-10 and CIFAR-100 experiments, we use 4000 and 10000 labeled examples preserving the class balance respectively. In Table 7, it is shown that Tied-FixMatch not only outperforms the baseline FixMatch but also outperforms its supervised counterpart which uses all of the 50000 labeled images. All hyperparameters are listed in Appendix 8.3.

4.6. Composability of Tied-Augment

It is crucial for a framework to be composable with other methods while retaining their performance improvements. To show that Tied-Augment has this property, we experiment with Sharpness-Aware Minimization (Foret et al., 2020). For SAM experiments, we train a Wide-ResNet-28-10 following the hyperparameters of the original work for 200 epochs which are listed in Appendix 8.4. We replicate their results with RandAugment(N=2, M=14). In Table 8, we show that Tied-SAM outperforms the baseline SAM.

5. Ablations and Analysis

In this section, we analyze the components of Tied-Augment framework and show their effectiveness. Additionally, we ablate our design choices.

5.1. Deconstructing Tied-Augment

In Table 9, we deconstruct Tied-Augment framework and show the improvement from each component. For each task considered, we create the highest-performing Tied-Augment method by first starting with the simplest baseline (standard crop-flip). Then, we apply RandAugment. Even though RandAugment provides noteworthy performance benefits (e.g. 1.3% on ImageNet), it is not effective and even harmful for finetuning and few-epoch training. Since Tied-Augment requires two differently augmented views of a sample, some of its improvement comes from "augmenting the batch" (Hoffer et al., 2020; Fort et al., 2021) (row (3)). We find additional benefits from diversifying the augmentation policies used for the different views (row (4)). Finally, the largest improvement comes from "tying" the representations coming from the two branches, which gives us Tied-RandAugment (row (5)), which adds an additional 1.1%, 0.4%, and 15.2% accuracy on ImageNet, CIFAR-10, and Stanford-Cars (epochs), respectively, in addition to our improved diversely augmented batch approach.

We find that for few-epoch from-scratch and fning experiments, generally 2 or 5 epochs, supervised signal from only one branch shows a better performance. In other cases, however, we found that cross entropy loss on both batches b_1 and b_2 improves the results more.

We, then, discuss the computational cost Tied-Augment below. Tied-Augment requires a single forward pass and a single backward pass. If there is no I/O bottleneck and a high-end accelerator (e.g Nvidia A100), the runtime of a forward pass on b_1 is roughly equal to a forward pass on b_1 and b_2. However, from a number of computational operations perspective, the required computation is double

	Aircraft	Cal-101	Cars	CIFAR-10	CIFAR-100	DTD	Flowers	Food	Pets	SUN397	Avg
SimCLR v2	78.7	82.9	79.8	96.2	79.1	70.2	94.3	82.2	83.2	61.1	80.8
BYOL	79.5	89.4	84.6	97.0	84.0	73.6	94.5	85.5	89.6	64.0	84.2
SwAV	83.1	89.9	86.8	96.8	84.4	75.2	95.5	**87.2**	89.1	**66.2**	85.4
Supervised	83.5	91.0	82.6	96.4	82.9	73.30	95.5	84.6	92.4	63.6	84.6
Tied-RA	84.7	92.6	89.9	96.9	83.9	75.8	96.7	84.3	93.5	63.9	86.2
Tied-RA + Tied-CF finetune	**88.1**	**93.3**	**90.2**	**97.2**	**85.2**	**76.2**	**97.3**	86.4	**93.9**	64.5	**87.1**

Table 6. Finetuning experiments on downstream datasets comparing self-supervised learning to Tied-Augment pretrained model. All reported models are ResNet50. Supervised baseline is pretrained using only RandAugment. SimCLR-v2, BYOL, SwAV, and supervised baseline are from (Ericsson et al., 2021). Tied-RA stands for Tied-RandAugment. Tied-RA + Tied-CF finetune is the case where a Tied-RA pretrained ResNet50 is finetuned using Tied-CropFlip. All models are finetuned using crop-flip data augmentation.

	FixMatch baseline	Supervised baseline	Tied-FixMatch
#labels	4k	50k	4k
CIFAR-10	95.7 $\pm$ 0.05	95.8 $\pm$ 0.02	**96.1** $\pm$ 0.04
CIFAR-100	77.4 $\pm$ 0.12	77.6 $\pm$ 0.04	**77.9** $\pm$ 0.08

Table 7. **Application of Tied-Augment framework to FixMatch.** Similarity function is applied to the features between the unsupervised branches. The reported FixMatch baseline results are from (Sohn et al., 2020), supervised baseline results are from (Cubuk et al., 2020) and include RandAugment, and our results are the average of 5 runs.

	Supervised baseline	SAM baseline	Tied-SAM
CIFAR-10	97.3 $\pm$ 0.03	97.9 $\pm$ 0.1	**98.3** $\pm$ 0.1
CIFAR-100	83.3 $\pm$ 0.05	86.2 $\pm$ 0.1	**86.5** $\pm$ 0.1

Table 8. **Sharpness-Aware minimization (SAM) experiments.** Baselines are replicated. Supervised baseline and SAM baseline both include RandAugment. The reported results are the average of 5 independent runs.

the forward pass of standard training. The cost of the backward pass on b_1 and b_2 size is approximately the same as a backward pass on b_1 on modern accelerators. Therefore, Tied-Augment only increases the computational cost by the additional forward pass; however, it is still computationally cheaper than double-step methods like SAM because it does not require two separate backward passes. For example, instead of a 100% increase in computational cost (as would be the case for SAM), we empirically observe an increase of roughly 30% increase on an Nvidia A100 on CIFAR-10.

5.2. Similarity Function

One of the critical components of Tied-Augment is the similarity term. In Table 10, we report the results of L1, L2 and cosine similarity functions. Here, it is worth noting that in the reported results, the weight of the cosine function is negative unlike L1 and L2 in the sense that for maximizing feature similarity L1 and L2 need to be minimized while cosine angle between the representations needs to be maximized. It is a known phenomenon that data augmentation can improve the model output invariance to distortions (Gontijo-Lopes et al., 2020). Therefore, it is intuitive to also encourage representation invariance. Interestingly, we find that the opposite can also be true. Enforcing feature dissimilarity can also improve performance on highly overparametrized CIFAR datasets considered; however, this is not the case for ImageNet for L2 similarity function. For simplicity (halving the search space for Tied-weight) and maximum performance improvement on all considered datasets, we choose only to consider increasing invariance. It is worth noting that negative Tied-weights for L1 and L2 (minimizing feature similarity) on CIFAR datasets also outperforms the baseline (Tied-weight=0). For cosine similarity, positive Tied-weight can outperform baseline for all datasets considered. We provide an analysis of the stability of tied-weight in Appendix 8.6.

5.3. General Design Choices

In Tied-Augment framework, there are many design choices that are of interest. For example, given that we double the batch size, there are two ways of doing the forward pass: separate forward passes on the batches or a single forward pass on both of the batches concatenated. These two approaches are not functionally equivalent for networks with BatchNorm (BN) (Ioffe & Szegedy, 2015) due to the running statistics. We find that the performance difference between these cases is generally equal to or less than 0.1%. We consistently report the results of double separate forward passes.

Another design choice to consider is the use of BN layers. For our experiments where we use two different RandAugment configurations (one weak, one stronger), we evaluated Split BatchNorms (Xie et al., 2020; Merchant et al., 2020) but did not find significant performance improvements. Thus we only report experiments that use standard BN layers.

Being invariant to different crops is a desirable property

Different components of Tied-Augment

	ImageNet	CIFAR-10	Stanford-Cars (2 epochs)
(1) Baseline (Flips and Crops)	76.3	96.1	59.9
(2) RandAugment	77.6	97.3	58.7
(3) Two views with same RandAugment policy	78.0	97.6	52.4
(4) Two views with different RandAugment policies	78.5	97.7	54.2
(5) Tied-Augment	**79.6**	**98.1**	**69.4**

Table 9. **Ablation study for the improvements coming from Tied-Augment on ImageNet, CIFAR-10, and CIFAR-100.** Relative to a baseline model, addition of two augmented views of the same image improves performance (3). Creating two augmented views by two distinct augmentation methods (generally one more aggressive RandAugment, and one less aggressive RandAugment) further boosts performance (4). Finally, adding a feature similarity objective yields a significant performance increase (5).

	similarity function	Tied-Crop-Flip	Tied-RA
CIFAR-10	L1	96.3	97.8
	Cosine	**96.5**	98.0
	L2	**96.5**	**98.1**
CIFAR-100	L1	81.3	84.8
	Cosine	81.5	**85.0**
	L2	**81.6**	**85.0**
ImageNet	L1	**76.9**	78.7
	Cosine	76.7	78.8
	L2	**76.9**	**79.2**

Table 10. **Ablation on the similarity function.** Tied-weights of all considered similarity functions have the signs so that they increase the feature similarity. Reported results are the average of 5 distinct runs. Imagenet Tied-RA models use (N=2, M=9) on both branches.

when targeting occlusion-invariance (Purushwalkam & Gupta, 2020). We also try using the same crop for both branches in our Tied-RandAugment experiments. This means taking a random (resized for ImageNet) crop from the image once and feeding the same crop into RandAugment on both augmentation branches. Surprisingly, this has little to no effect on performance. Therefore, for simplicity, we use different crops on both augmentation branches for CIFAR and finetuning experiments, same crop for ImageNet experiments.

6. Conclusion

As dataset and model sizes increase, machine learning models are trained for fewer and fewer epochs. Traditionally this has made data augmentation less useful. We introduce Tied-Augment, a simple method for combining self-supervised learning and regular supervised learning to strengthen state-of-the-art methods such as mixup, SAM, FixMatch, and RandAugment by up to 2% on Imagenet. Tied-Augment can be implemented with only a few lines of additional code.

Tied-Augment can improve the effectiveness of standard data augmentation approaches such as Crop-Flip even when training for a few epochs. When training for longer, Tied-Augment achieves significant improvements over state-of-the-art augmentation methods.

Tied-Augment shows the promise of combining self-supervised approaches with regular supervised learning. An exciting future direction would be to evaluate Tied-Augment for large language model training which tends to be for a few epochs.

7. Acknowledgments

We thank Omer Faruk Ursavas for his contributions to this project. The numerical calculations reported in this paper were partially performed at TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA resources). We acknowledge the support of CURe program from Google AI, Jonathan Caton, and the Google Cloud team. We acknowledge Johannes Gasteiger for his feedback on the manuscript, Jascha Sohl-dickstein for helpful discussions. .

References

Asano, Y. M., Rupprecht, C., and Vedaldi, A. Self-labelling via simultaneous clustering and representation learning. *arXiv preprint arXiv:1911.05371*, 2019.

Bai, J., Yuan, L., Xia, S.-T., Yan, S., Li, Z., and Liu, W. Improving vision transformers by revisiting high-frequency components. *arXiv preprint arXiv:2204.00993*, 2022.

Bautista, M. A., Sanakoyeu, A., Tikhoncheva, E., and Ommer, B. Cliquecnn: Deep unsupervised exemplar learning. *Advances in Neural Information Processing Systems*, 29, 2016.

Bishop, C. M. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

Caron, M., Bojanowski, P., Joulin, A., and Douze, M. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 132–149, 2018.

Caron, M., Bojanowski, P., Mairal, J., and Joulin, A. Unsupervised pre-training of image features on non-curated data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2959–2968, 2019.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924, 2020.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020a.

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. E. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020b.

Chen, X. and He, K. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15750–15758, 2021.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 113–123, 2019.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 702–703, 2020.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Dwibedi, D., Aytar, Y., Tompson, J., Sermanet, P., and Zisserman, A. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9588–9597, 2021.

Ericsson, L., Gouk, H., and Hospedales, T. M. How well do self-supervised models transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5414–5423, 2021.

Ericsson, L., Gouk, H., Loy, C. C., and Hospedales, T. M. Self-supervised representation learning: Introduction, advances, and challenges. *IEEE Signal Processing Magazine*, 39(3):42–62, 2022.

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

Fort, S., Brock, A., Pascanu, R., De, S., and Smith, S. L. Drawing multiple augmentation samples per image during training efficiently decreases test error. *arXiv preprint arXiv:2105.13343*, 2021.

Ghiasi, G., Cui, Y., Srinivas, A., Qian, R., Lin, T.-Y., Cubuk, E. D., Le, Q. V., and Zoph, B. Simple copy-paste is a strong data augmentation method for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2918–2928, 2021.

Gidaris, S., Bursuc, A., Komodakis, N., Pérez, P., and Cord, M. Learning representations by predicting bags of visual words. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6928–6938, 2020.

Gontijo-Lopes, R., Smullin, S., Cubuk, E. D., and Dyer, E. Tradeoffs in data augmentation: An empirical study. In *International Conference on Learning Representations*, 2020.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Hadsell, R., Chopra, S., and LeCun, Y. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pp. 1735–1742. IEEE, 2006.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.

Hoffer, E., Ben-Nun, T., Hubara, I., Giladi, N., Hoefler, T., and Soudry, D. Augment your batch: Improving generalization through instance repetition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8129–8138, 2020.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673, 2020.

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. Ssd: Single shot multibox detector. In *European conference on computer vision*, pp. 21–37. Springer, 2016.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.

Lopes, R. G., Yin, D., Poole, B., Gilmer, J., and Cubuk, E. D. Improving robustness without sacrificing accuracy with patch gaussian augmentation. *arXiv preprint arXiv:1906.02611*, 2019.

Maji, S., Kannala, J., Rahtu, E., Blaschko, M., and Vedaldi, A. Fine-grained visual classification of aircraft. Technical report, 2013.

Merchant, A., Zoph, B., and Cubuk, E. D. Does data augmentation benefit from split batchnorms. *arXiv preprint arXiv:2010.07810*, 2020.

Nilsback, M.-E. and Zisserman, A. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pp. 722–729. IEEE, 2008.

Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3498–3505. IEEE, 2012.

Purushwalkam, S. and Gupta, A. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. *Advances in Neural Information Processing Systems*, 33:3407–3418, 2020.

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

Shorten, C. and Khoshgoftaar, T. M. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

Simard, P. Y., Steinkraus, D., Platt, J. C., et al. Best practices for convolutional neural networks applied to visual document analysis. In *Icdar*, volume 3. Edinburgh, 2003.

Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C. A., Cubuk, E. D., Kurakin, A., and Li, C.-L. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in neural information processing systems*, 33:596–608, 2020.

Sun, C., Shrivastava, A., Singh, S., and Gupta, A. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pp. 843–852, 2017.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147. PMLR, 2013.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. What makes for good views for contrastive learning? *arXiv preprint arXiv:2005.10243*, 2020.

Tikhonov, A. N. and Arsenin, V. Y. Solutions of ill-posed problems. 1977.

Xie, C., Tan, M., Gong, B., Wang, J., Yuille, A. L., and Le, Q. V. Adversarial examples improve image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 819–828, 2020.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

8. Appendix

8.1. ImageNet

All ImageNet models use a learning rate of 0.4 with a batch size of 1024, weight-decay rate of 1e-4. The Tied-RandAugment model that was trained for 90 epochs used Crop-Flip on first branch, and RandAugment(N=2, M=9) on the other branch, with a Tied-weight of 4. The Tied-RandAugment ResNet-50 model that was trained for 360 epochs used RandAugment(N=2, M=13) for the first branch and RandAugment(N=2, M=9, P=0.5) for the second branch, with a Tied-weight of 12.0. The Tied-RandAugment ResNet-200 model used RandAugment(N=2, M=13) for both branches with a Tied-weight of 12.0.

All Tied-Augment ImageNet models trained for 90 epochs used a Tied-weight of 4, and models trained for longer used a Tied-weight of 12. The optimal Tied-weight for Tied-mixup on Imagenet was 50.

8.2. CIFAR-10, CIFAR-100 and CIFAR-4K

For CIFAR-4K, we use a learning rate of 0.1 with a batch size of 128 and weight-decay rate of 5e-4. All the baselines and Tied-Augment models were trained for 500 epochs. The reported models for Wide-ResNet-28-10 and Wide-ResNet-28-2 configurations use RandAugment(N=2, M=14) for both branches with a Tied-weight of 10 and 24 respectively.

CIFAR-10 and CIFAR-100 models use a learning rate of 0.1 with a batch size of 128 and weight-decay rate of 5e-4 and were trained for 200 epochs. For both Wide-ResNet-28-2 and Wide-ResNet-10 configurations, we use RandAugment(N=2, M=14) for the first branch and RandAugment(N=2, M=19) for the second branch. Tied-weights for the reported results for Wide-ResNet-28-2 and Wide-ResNet-28-10 are 16 and 20 respectively.

8.3. Tied-FixMatch

For our Tied-FixMatch experiments on CIFAR-10 and CIFAR-100, we set the FixMatch parameters as follows: τ=0.95, λ_u=1, μ=7, batch size=64, learning rate=0.03 using SGD with Nesterov momentum. We set weight decay to 0.0005 and 0.001 for CIFAR-10 and CIFAR-100 respectively.

8.4. Tied-SAM

On CIFAR-10 and CIFAR-100, we use learning rate of 0.1, batch size of 256, weight decay of 0.0005. SAM hyperparameter ρ to 0.05 and 0.1 for CIFAR-10 and CIFAR-100 respectively.

8.5. Comparison between Self-Supervised Learning and Tied-Augment

Given that Tied-Augment combines self-supervised learning with supervised learning, it is important to understand the intuition behind this framework. One intuitive observation is that purely self-supervised methods can sometimes suffer from representation-collapse which is converging to the trivial solution of outputting zeros for all inputs, which would make sure representations are the same for differently augmented samples. This could also happen if we only trained with our similarity loss.

This intuition seems to be relevant to why self-supervised training can be unstable and papers have focused on removing this instability. For example, SimCLR has added an additional layer for contrastive learning that gets discarded during finetuning. Other self-supervised methods such as MoCo had to use momentum and large batch sizes to stabilize training. In our case, since we use supervision from the beginning, the representation-collapse or other instabilities do not occur. It is of course possible to increase the tied-weight sufficiently to cause collapse, but in practice a simple hyperparameter search over values of $\{1, 5, 10, 50\}$ is sufficient and tied-weight up to 10 is stable for most experimental setups. In addition to increasing performance in a stable way without the need for search over many hyperparameters or other tricks required self-supervised training, Tied-Augment provides a significant improvement in performance even with the very same hyperparameters with its supervised baseline. Therefore, in the presence of labels, Tied-Augment is more favorable than supervised or self-supervised learning.

One important advantage of self-supervised learning is its promise of not requiring labels in which case Tied-Augment cannot be used. We would like to draw attention to the fact that, as evinced by Tied-FixMatch, Tied-Augment is beneficial even if there are only few labels available. Additionally, existence of methods like CLIP and large language models and datasets such as LAION-5B (Schuhmann et al., 2022) and JFT-300 (Sun et al., 2017) show us that (1) supervision is possible without real class labels (2) curating large datasets that are noisly and weakly labeled is possible and such datasets are extremely effective. This shows us that the combination of supervision and self-supervision will be a crucial paradigm in the future which we propose in this paper.

8.6. Stability of tied-weight

In Figure 3, we present the stability of the introduced tied-weight hyperparameter. It is shown that even for a large range of values, tied-weight is able to improve the performance of the model on ImageNet dataset, indicating that Tied-Augment offers significant performance improvements

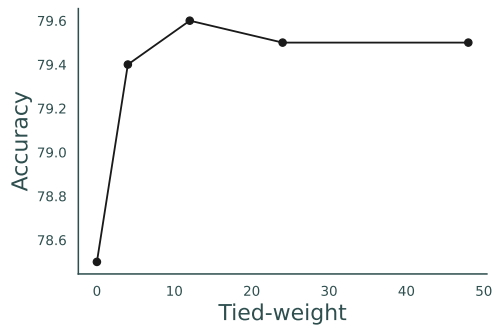

Figure 3. Tied-RandAugment accuracy with ResNet-50 on ImageNet as a function of tied-weight.

without the need for extensive hyperparameter search.

8.7. More Detailed Tied-Augment Analysis

The following is the analysis sketch for various approaches We are tying together two different augmentation distributions $P(\tilde{\mathbf{x}}_1, \tilde{\mathbf{y}}_1|\mathbf{x}, \mathbf{y}), P(\tilde{\mathbf{x}}_2, \tilde{\mathbf{y}}_2|\mathbf{x}, \mathbf{y})$ as follows

$$R_{\text{TiedAug}}(f, h) = \frac{1}{N} \sum_{(\mathbf{x},\mathbf{y})} E[\ell(f(h(\tilde{\mathbf{x}}_1)), \tilde{\mathbf{y}}_1) + \tag{4}$$

$$\|h(\tilde{\mathbf{x}}_1) - m(h(\tilde{\mathbf{x}}_2))\|^2] \tag{5}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the input, $\mathbf{y} \in \mathbb{R}^m$ are the labels, f is the final classifier on the features provdided by h. m is a function that ensures that the hidden features from both transformations correspond to the same class

$$P(\mathbf{y}|h(\tilde{\mathbf{x}}_1)) = P(\mathbf{y}|m(h(\tilde{\mathbf{x}}_2))) \tag{6}$$

In the case of augmentations that do not change the class such as additive gaussian noise, the identity function will suffice $m(x) = x$.

8.7.1. Tied-GaussianNoise with L2 distance

We can gain some insight into TiedAugment by considering its application to Gaussian input noise augmentation. The additive regularization for Tied-GaussianNoise is given by

$$\Omega_G(f) = wE_\epsilon[\|h(\mathbf{x}) - h(\mathbf{x} + \epsilon)\|^2]$$

where h produces the features of the network, $\mathbf{x} \in \mathbb{R}^n$, and $\epsilon \sim \mathcal{N}(0, \sigma)^n$. Consider the approximation of this term using the first-order Taylor expansion of h at $\mathbf{x}$, $\Omega_G(h) \approx w\sigma^2\|\nabla h(\mathbf{x})\|_F^2$.

$$\begin{aligned}\Omega_G(h) &\approx wE_{\mathbf{x},\epsilon}[\|\nabla_{\mathbf{x}} f(\mathbf{x})^T \epsilon\|^2] \\ &= wE_{\mathbf{x}}[\nabla_{\mathbf{x}}^T f(\mathbf{x}) E_\epsilon[\epsilon\epsilon^T] \nabla_{\mathbf{x}} h(\mathbf{x})] \\ &= w\sigma^2 E_{\mathbf{x}}[\|\nabla_{\mathbf{x}} f(\mathbf{x})\|_F^2]\end{aligned} \tag{7}$$

This additive regularization is part of the well known class of Tikhonov regularizers (Tikhonov & Arsenin, 1977; Bishop, 1995) that include weight decay. It encourages the feature mapping function to become more invariant to small corruptions in the input, which can be beneficial for generalization.

8.7.2. (worse approximation) TiedGaussianNoise with cosine similarity

The regularization for TiedGaussianNoise with cosine similarity is given by

$$\Omega_{TGNCS}(h) = \lambda E\left[\frac{h(\mathbf{x})^T h(\mathbf{x} + \epsilon)}{\|h(\mathbf{x})\|\|h(\mathbf{x} + \epsilon)\|}\right] \tag{8}$$

Consider the second-order Taylor expansion of h around $\mathbf{x}$

$$\approx \lambda E\left[\frac{h(\mathbf{x})^T (h(\mathbf{x}) + \nabla_{\mathbf{x}} h(\mathbf{x})\epsilon + \epsilon^T \nabla_{\mathbf{x}}^2 h(\mathbf{x})\epsilon)}{\|h(\mathbf{x})\|\|h(\mathbf{x} + \epsilon)\|}\right] \tag{9}$$

Now, consider the first-order Taylor expansion of the norm

$$E[\|h(\mathbf{x} + \epsilon)\|] \approx \|h(\mathbf{x})\| + \nabla_{\mathbf{x}}(\|h(\mathbf{x})\|)E[\epsilon] \tag{10}$$

Given that the noise is zero-mean the second term disappears

$$E[\|h(\mathbf{x} + \epsilon)\|] \approx E[\|h(\mathbf{x})\|] \tag{11}$$

Putting this together we have

$$\approx \lambda E\left[\frac{h(\mathbf{x})^T (h(\mathbf{x}) + \nabla_{\mathbf{x}} h(\mathbf{x})\epsilon + \epsilon^T \nabla_{\mathbf{x}}^2 h(\mathbf{x})\epsilon)}{\|h(\mathbf{x})\|^2}\right] \tag{12}$$

which simplifies to

$$= \lambda + 0 + \lambda E\left[\frac{h(\mathbf{x})^T \epsilon^T \nabla_{\mathbf{x}}^2 h(\mathbf{x})\epsilon}{\|h(\mathbf{x})\|^2}\right] \tag{13}$$

dropping the constant term and simplifying further we have

$$\lambda \sum_i \frac{h_i(\mathbf{x})}{\|h(\mathbf{x})\|} \frac{\text{Tr}(\nabla_{\mathbf{x}}^2 h_i(\mathbf{x}))}{\|h(\mathbf{x})\|} \tag{14}$$