On the Importance of Asymmetry for Siamese Representation Learning

Xiao Wang*,† Haoqi Fan[1,†] Yuandong Tian[1] Daisuke Kihara[2] Xinlei Chen[1]
[1]Facebook AI Research (FAIR) [2]Purdue University

Code: https://github.com/facebookresearch/asym-siam

Abstract

Many recent self-supervised frameworks for visual representation learning are based on certain forms of Siamese networks. Such networks are conceptually symmetric with two parallel encoders, but often practically asymmetric as numerous mechanisms are devised to break the symmetry. In this work, we conduct a formal study on the importance of asymmetry by explicitly distinguishing the two encoders within the network – one produces source encodings and the other targets. Our key insight is keeping a relatively lower variance in target than source generally benefits learning. This is empirically justified by our results from five case studies covering different variance-oriented designs, and is aligned with our preliminary theoretical analysis on the baseline. Moreover, we find the improvements from asymmetric designs generalize well to longer training schedules, multiple other frameworks and newer backbones. Finally, the combined effect of several asymmetric designs achieves a state-of-the-art accuracy on ImageNet linear probing and competitive results on downstream transfer. We hope our exploration will inspire more research in exploiting asymmetry for Siamese representation learning.

1. Introduction

Despite different motivations and formulations, many recent un-/self-supervised methods for visual representation learning [1, 6–8, 18, 19, 44] are based on certain forms of Siamese networks [4]. Siamese networks are inherently *symmetric*, as the two encoders within such networks share many aspects in design. For example, their model architectures (*e.g.*, ResNet [20]) are usually the same; their network weights are often copied over; their input distributions – typically compositions of multiple data augmentations [8] – are by default identical; and their outputs are encouraged to be similar for the same image. Such a symmetric structure not only enables straightforward adaptation from off-the-shelf, supervised learning architectures to self-supervised learning, but also introduces a minimal inductive bias to learn representations invariant w.r.t. various transformations in computer vision [10].

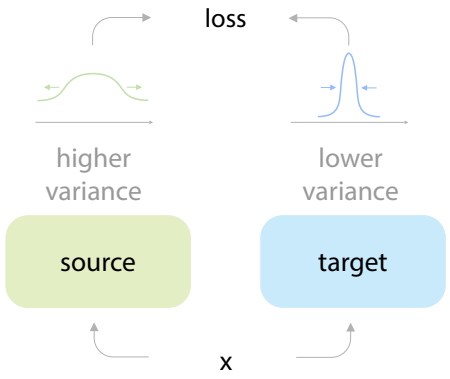


Figure 1. **Asymmetry for Siamese** representation learning. For the two encoders in a Siamese network, we treat one as a source encoder, and the other as a target encoder. We find it generally beneficial to have relatively lower variance in target than source.

However, symmetry is not the only theme in these frameworks. In fact, numerous mechanisms were proposed to break the conceptual symmetry. For example, BYOL [18] and SimSiam [10] place a special predictor head on one of the encoders, so architecture-wise they are no longer symmetric; MoCo [19] introduces momentum encoder, in which the weights are computed with moving-averages instead of directly copied; SwAV [6] and DINO [7] additionally adopt a multi-crop [27] strategy to enhance the augmentation on one side, shifting the data distribution asymmetric between encoders; even the InfoNCE loss [28] treats outputs from two encoders differently – one is positive-only and the other also involves negatives. Among them, some specific asymmetric designs are crucial and well-studied (*e.g.*, stop-gradient to prevent collapse [10]), but the general role of *asymmetry* for Siamese representation learning is yet to be better understood.

In this paper, we conduct a more formal study on the importance of asymmetry for Siamese learning. Deviating from the original meaning of 'Siamese', we explicitly mark the two encoders within the network functionally different: a *source* encoder and a *target* encoder.[1] The

∗: work done during internship at FAIR. †: equal contribution.

[1]Depending on the context, *source* has also been referred as query/online/student; and *target* as key/teacher in the literature [18, 19, 32].

source encoder generates source encodings, and updates its weights via normal gradient-based optimization like in supervised learning. The target encoder updates its weights only with their source counterparts, and outputs target encodings which in turn judge the quality of sources. This asymmetric encoder formulation also covers symmetric encoders (*e.g.*, in SimCLR [8]), where the target weights can be simply viewed as source duplicates.

With this distinction, our key insight is that *keeping a relatively lower variance in target encodings than source can help representation learning* (illustrated in Fig. 1). We systematically study this phenomenon with our MoCo v2 [9] variant beyond existing – but scattered – evidence in the literature [5, 6, 19, 24, 37]. Specifically, given a variance-oriented design, we first quantify its encoding variance with our baseline model, and then apply it to source or target (or both) encoders and examine the influence on learned representations. In total, we have conducted *five* case studies to explore various design spaces, ranging from encoder inputs, to intermediate layers and all the way to network outputs. The results are well-aligned with our insight: designs that increase encoding variance generally help when applied to source encoders, whereas ones that decrease variance favor target. We additionally provide a preliminary theoretical analysis taking MoCo pre-training objective as an example, aimed at revealing the underlying cause.

Our observation generalizes well. First, we show the improvements from asymmetry – lower variance in target than source – can hold with longer pre-training schedules, suggesting they are not simply an outcome of faster convergence. Second, directly applying proper asymmetric designs from MoCo v2 to a variety of other frameworks (*e.g.*, BYOL [18], Barlow Twins [44]) also works well, despite notable changes in objective function (contrastive or non-contrastive), model optimization (large-batch training [43] or not), *etc*. Third, using MoCo v3 [11], we also experimented a more recent backbone – Vision Transformer (ViT) [14] – and find the generalization still holds well. Finally, several asymmetric designs are fairly compositional: their combined effect enables single-node pre-trained MoCo v2 to reach a top-1 linear probing accuracy of *75.6%* on ImageNet, a state-of-the-art with ResNet-50 backbone. This model also demonstrates good transferring ability to other downstream classification tasks [8, 15, 18].

In summary, our study reveals an intriguing correlation between the relative source-target variance and the learned representation quality. We have to note that such correlation has limitations, especially as self-supervised learning follows a staged evaluation paradigm and the final result is inevitably influenced by many *other* factors. Nonetheless, we hope our exploration will raise the awareness of the important role played by asymmetry for Siamese representation learning, and inspire more research in this direction.

2. Related Work

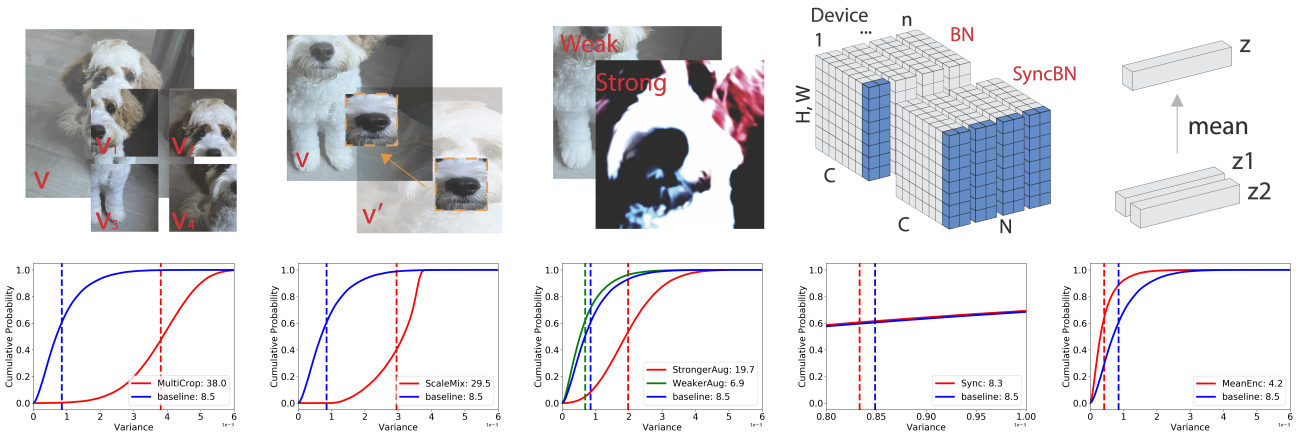
Siamese networks are weight-sharing networks [4] that process multiple inputs and produce multiple outputs in parallel. It has been widely used in computer vision [3,4,31,38] and has recently caught attention in self-supervised learning [8, 10]. This can be explained by the design of Siamese networks, which can conveniently learn *invariance* in a data-driven fashion – a widely acknowledged property for useful visual representations [10]. While a naïve application of Siamese network can incur collapse, various formulations and mechanisms (*e.g.*, contrastive learning [8, 19], online balanced clustering [6, 7], extra predictor [10, 18], variance reduction loss [1, 44]) – many of them asymmetric – have been proposed to maintain healthy learning dynamics. Our focus is *not* on collapse prevention. Instead, we study generic designs that change encoding variance, analyze their effect on the output representations, and show that an asymmetry between source and target helps learning.

Symmetry for Siamese learning. While the theme of the paper is asymmetry, symmetry is also a powerful concept in Siamese learning. One advantage of symmetry is in reducing the computation cost when source and target encoders share the same backbone weights. In such frameworks [8, 10], source features can be *reused* for targets, saving the extra need to compute with a second encoder. Recently, symmetric designs alone are also shown to yield the same level of performance as asymmetric methods [1, 44].

Interestingly, there is often an attempt to *symmetrize* the loss by forwarding image views once as source and once as target [11, 18], even when the encoder weights are *not* shared (*e.g.*, in case of a momentum encoder [19]). Compared to using a single asymmetric loss but training for 2× as long, this practice has the same number of forward/backward passes and we empirically verify it generates similar results across frameworks (see Sec. 6.2) [10]. Therefore, we believe loss symmetrization is *not* essential beyond plausible better performance at the 'same' training epochs.

Asymmetric source-target variance. Asymmetry in variance is already serving self-supervised learning in implicit ways. MoCo [19] itself is a successful example: by smoothing its target encoder, the memory bank stores consistent keys with smaller variance across training iterations. Momentum update has been extended to normalization statistics to further reduce variance [5, 24], again applied on targets. State-of-the-art on ImageNet [37, 41, 47] is held by using high-variance, strong augmentations on source views.

Siamese networks are also popular in *semi*-supervised learning, where some examples are unlabeled. To create more reliable pseudo labels, the common practice is to average predicted labels over augmented views [2,30,36], which effectively reduces variance on target. Such evidences are scattered in the literature, and we analyze it systematically.



(a) **MultiCrop** (Sec. 4.1) (b) **ScaleMix** (Sec. 4.2) (c) **AsymAug** (Sec. 4.3) (d) **SyncBN** (Sec. 4.4) (e) **MeanEnc** (Sec. 4.5)

Figure 2. We present **five case studies** exploring different variance-oriented designs for source and target encoders. For each column, we show the specific design on the top, and its influence on the encoding variance (both the cumulative distribution function and the mean on the validation set as our empirical reference) at the bottom. Each design is then applied to either the source, the target, or both encoders. The resulting representation is evaluated by linear probing on ImageNet. Best viewed on a screen and zoomed in. See Sec. 4 for details.

3. Methodology Overview

In this section we give an overview for our methodology to systematically study variance-oriented encoder designs. First, we specify our variance of interest. While exactly quantifying such variance during training is hard, we provide an approximate *reference* for such variance using our baseline model. Now, for each design we can then compute its variance reference and quantify the *relative* change in comparison to a vanilla encoder. Regardless of the change (higher or lower), we plug-in the design to either the source, the target, or both encoders and see its influence on resulting representations after pre-training. The influence is measured by linear probing on ImageNet [13]. For a particular design, if applying it to both (or neither) encoders is better, then it implies maintaining symmetry is important; if it prefers either source or target, then it means *asymmetry* is beneficial. In such cases, we also check whether the change in variance is correlated with the encoder preference.

In total, we have conducted *five* case studies exploring various design spaces, ranging from encoder inputs (*i.e.*, data augmentations), to intermediate layers (*i.e.*, different batch sizes for Batch Normalization [21]) all the way to network outputs (*i.e.*, averaging multiple encodings to reduce variance). Fig. 2 shows these designs and their variance plots in conjunction with our baseline. We detail our baseline and each case study in Sec. 4, and first motivate our variance of interest and its reference in the following.

Variance of interest. As each encoding is the encoder output of an augmented view from an image, the total variance in encodings mainly comes from three types: *i)* changes to the encoder, *ii)* changes across images, and *iii)* changes within a single image. For type i), MoCo [19] with its momentum encoder is already a major, well-studied asymmetric design that intuitively reduces the target variance across training iterations. For type ii), as Siamese representation learning encourages *uniformity* [10, 35], the cross-image variance quickly converges to a constant dependent only on encoding dimensions (evidenced in Appendix A).[2] Therefore, we focus on type iii), *i.e.*, *intra*-image variance as the main subject of our study. Note that it does not restrict us to design input augmentations as the only means to adjust variance, as will be discussed in Secs. 4.4 and 4.5.

Variance reference. Exactly quantifying intra-image variance requires sampling all possible augmentations of all images and forward all of them to obtain encodings for all training steps. Even if possible, this process is highly expensive and also probably unnecessary. Therefore, we resort to an approximation with the goal of keeping a reference to characterize the encoding variance when changed.

To this end, we simply augment each image in the validation set r times and feed them to a pre-trained baseline encoder. The output encodings are then used to compute the per-image, intra-sample variance, which jointly form a distribution. All variances across the entire set are then averaged to a single value v, the reference variance used to measure different designs. More details are listed in Sec. 7.

4. Case Studies for Source-Target Variance

In this section, we introduce our baseline and perform five empirical case studies exploring the impact of different designs. For each one of them, we record its corresponding variance reference v, and linear-probing accuracies when placed on encoders with different configurations

[2] If encodings are uniformly distributed on the unit hypersphere (due to ℓ_2 normalization), their variance is $1/d$ where d is the encoding dimension.

without preset bias. Since our goal is to analyze the behavior, all models in this section are pre-trained for 100 epochs, with the generalization toward longer schedules deferred to Sec. 6.1 after we draw the connection between variance change and encoder preference in Sec. 4.6.

Baseline. Our baseline is an improved variant of MoCo v2 [9], which itself is an improved baseline over original MoCo [19]. It consists of a gradient-updated source encoder f_s, a momentum-updated target encoder f_t, and an encoding-updated memory bank [40]. Inspired by SimCLR [8], each MoCo v2 encoder further uses a projection head (projector), which is a 2-layer MLP *without* Batch Normalization (BN) [21] in-between. Our baseline adds an additional fully connected layer (2048-*d*, *with* BN) before the 2-layer MLP. Inherited from MoCo v1, all BNs in f_s are performed per GPU device, and all BNs in f_t are shuffled [19]. All the output encodings $\mathbf{z}$ are ℓ_2 normalized to unit-length vectors before InfoNCE loss [28]. We do *not* employ any loss symmetrization [6,18] in this baseline, thus one source/target pair only contributes to the loss *once*.

Compared to vanilla MoCo v2 [9], our baseline is generally better in linear probing on ImageNet [13] (detailed in Sec. 7). The table below summarizes the top-1 accuracy (%) using ResNet-50 [20] and the same evaluation protocol:

	100 ep	200 ep	400 ep	800 ep
MoCo v2 [9]	64.7	67.9	69.6	70.7
MoCo v2, *ours*	65.8	69.0	70.5	71.9

The improvement (~1 percent) is consistent across different number of training epochs. We also notice no degradation in object detection transfer on VOC [16] – *e.g.*, achieving 57.4 mAP at 800 pre-training epochs, same as original [9]. The variance reference for our baseline v_0 is 8.5 ($\times 10^{-4}$).

4.1. Study 1: MultiCrop Augmentation

We begin our study with an existing design in the literature – multi-crop augmentation (or 'MultiCrop') [6, 7, 27]. Besides the two basic views needed for Siamese learning, MultiCrop takes additional views from each image per iteration. To alleviate the added computation cost, a common strategy is to have m low-resolution crops (*e.g.*, 96×96 [6]) instead of standard-resolution crops (224×224) as added views (illustrated in Fig. 2a top for m=4). As a side effect, inputting small crops can potentially increase the variance for an encoder due to the size and crop-distribution changes. This is confirmed in Fig. 2a bottom, where we compare the variance distribution of MultiCrop to our baseline on the ImageNet val set. We show the cumulative distribution function in solid lines with increasing per-image variances from left to right, and the mean variances v and v_0 in dotted vertical lines. MultiCrop has significantly higher variance than our baseline: v=38.0 *vs.* 8.5 ($\times 10^{-4}$).

We plug-in MultiCrop to either the source, the target, or both encoders (detailed in Appendix D). The table below summarizes the corresponding top-1 accuracy and change (Δ) to the baseline in linear probing:

+MultiCrop (↑)	neither	source	target	both
accuracy (%)	65.8	69.9	57.1	61.7
Δ (%)	/	+4.1	-8.7	-4.1

As a design that increases variance (indicated by '↑' in table), MultiCrop improves the accuracy substantially (+4.1%) when applied to the source encoder, and hurts when applied to the target. When applied to both, the performance also degenerates significantly (-4.1%), even with more crops processed per training iteration than to source alone. These results indicate that the source encoder is the preferred place of applying MultiCrop (column shaded in gray) – which also matches the common protocols in the literature when multi-crop augmentation is used [6, 7, 27].

4.2. Study 2: ScaleMix Augmentation

Next, we introduce and study a different type of augmentation called 'ScaleMix', illustrated in Fig. 2b top (more details are found in Appendix B). As the name suggests, it generates new views of an image by mixing two views of potentially different scales together via binary masking. The masking strategy follows CutMix [29], where an entire region – denoted by a box with randomly sampled coordinates – is cropped and pasted. Unlike CutMix, ScaleMix only operates on views from the *same* image, and the output is a *single* view of standard size (224×224). This single view can be regarded as an efficient approximation of multiple crops in MultiCrop, without the need to process small crops separately. Like MultiCrop, ScaleMix also introduces extra variance to the encoding space (as shown in Fig. 2b bottom), with a mean variance of v=29.5 ($\times 10^{-4}$).

Again, we apply ScaleMix augmentation to the source, the target, or both encoders without preset preference. The results for linear probing are summarized in the table below:

+ScaleMix (↑)	neither	source	target	both
accuracy (%)	65.8	67.3	52.8	64.8
Δ (%)	/	+1.5	-13.0	-1.0

We observe a similar trend as the MultiCrop case: ScaleMix benefits source encoders, harms target encoders, and the effect neutralizes when applied to both. This suggests source encoder is again the preferred choice for ScaleMix.

4.3. Study 3: General Asymmetric Augmentations

MultiCrop and ScaleMix are mostly on geometric transformations of images. Next, we study the behavior by varying other ingredients in the MoCo v2 augmentation recipe.

The original v2 recipe is *symmetric*: the same set of augmentations (*e.g.*, random resized cropping, color jittering [40], blurring [8]) is used for both source and target. In this case study, we add or remove augmentations (beyond geometric ones), and present two more recipes: one

deemed stronger ('StrongerAug'), and the other weaker ('WeakerAug') compared to the original one (detailed in Appendix D). Together, they can form general *asymmetric* augmentation recipes for source and target. Complying with the intuition, we find StrongerAug has higher variance 19.7 ($\times 10^{-4}$), and WeakerAug has lower variance 6.9 ($\times 10^{-4}$) w.r.t. to the baseline v_0 (shown in Fig. 2c bottom).

The results are split into three tables for clarity. The influence of WeakerAug is summarized first:

+WeakerAug (↓)	neither	source	target	both
accuracy (%)	65.8	51.0	67.2	46.8
Δ (%)	/	-14.8	+1.4	-19.0

Interestingly, the effect of WeakerAug on source/target encoder is *opposite* compared to the previous studies: it hurts source but helps target (referred as 'AsymAug'). A symmetric WeakerAug on both does not work, suggesting the heavy reliance of Siamese learning on augmentation recipes [8, 18]. On the StrongerAug side:

+StrongerAug (↑)	neither	source	target	both
accuracy (%)	65.8	66.7	62.2	66.2
Δ (%)	/	+0.9	-3.6	+0.4

It helps most when used only on source, but harms accuracy when used only on target. For completeness, we also experimented changing augmentation strength in opposite directions for source and target:

Stronger & Weaker	source ↑ target ↓	source ↓ target ↑
accuracy (%)	67.2	44.3
Δ (%)	+1.4	-21.5

Compared to having WeakerAug on target alone (67.2%), further adding StrongerAug on source does not bring extra gains. In contrast, stronger augmentations on target and weaker augmentations on source results in the worst performance in all the cases we have studied.

4.4. Study 4: Sync BatchNorm

Although input data augmentation is a major source of intra-image variance, it is *not the only* cause of such variance within output encodings. One notable source lies in intermediate BN layers [21], a popular normalization technique in modern vision architectures [20]. During training, the statistics for BN are computed *per-batch*, which means if other images within the batch are replaced, the output will likely change even if the current image stays the same. As a result, the magnitude of this variance is largely controlled by the *batch size*: a sufficiently large size can provide nearly stable statistics, whereas for small batches (*e.g.*, below 16) the estimation is generally less accurate [39]. For MoCo v2, its effective batch size is 32, because the default BN performs normalization only on the same device (256 images/8 GPUs).[3] A natural alternative is to employ SyncBN that normalizes over all devices, so the batch size is 256 (illustrated in Fig. 2d top for 4 devices). From the zoomed-in variance plot (Fig. 2d bottom), SyncBN leads to a slight decrease in variance from 8.5 to 8.3 ($\times 10^{-4}$) in this case – suggesting 32 is already sufficiently stable in our baseline.

For efficiency and generalizability, we replace the *single* BN in our 3-layer projector with SyncBN.[4] As before, we tried different combinations on encoders and the results are:

+SyncBN (↓)	neither	source	target	both
accuracy (%)	65.8	64.7	66.5	66.0
Δ (%)	/	-0.9	+0.7	+0.2

Despite the seemly minor modification, SyncBN still leads to a notable improvement when applied to target (referred as 'AsymBN') and degeneration to source. SyncBN on both encoders is at-par with the baseline per-device BNs.

4.5. Study 5: Mean Encoding

In this last study we focus on the encoder output. According to basic statistics, a direct approach to reduce the variance of a random variable is to perform i.i.d. sampling multiple times and take the mean as the new variable. Specifically for v, we can reduce it by a factor of $\sim n$ if the output encoding $\mathbf{z}$ is averaged from n separate encodings $\{\mathbf{z}^1, \dots, \mathbf{z}^n\}$ (illustrated in Fig. 2e top for n=2).[5] These encodings can be simply generated by running the same encoder on n augmented views of the same image (detailed in Appendix D). For example, we show v is 4.2 ($\times 10^{-4}$), about half of v_0 when two encodings are averaged in Fig. 2e bottom. We name this design 'MeanEnc' for an encoder.

As discussed in our Sec. 2 (also shown in [10]), increasing the number of views per training iteration can lead to better performance *by itself*. To minimize this effect, we conduct our main analysis of MeanEnc by *fixing* the total number of views to 4 per training iteration. The 4 views are split between source (n_s) and target (n_t) encoders, shown in the first 3 result columns below:

+MeanEnc (↓)	n_s =1 n_t =3	n_s =2 n_t =2	n_s =3 n_t =1	n_s =1 n_t =2
accuracy (%)	67.9	67.1	59.9	67.5
Δ (%)	+2.1	+1.3	-5.9	+1.7

With more views in the target encoder (and simultaneously fewer views in source), we observe a trend for better accuracy. Having 2 views in both encoders still keeps symmetry, so its improvement over baseline (65.8%) is an outcome of more views. For simplicity, we also experimented MeanEnc with 2 views in the target encoder alone (last column). The result strikes a better balance between speed and accuracy, so we pick this setting as default for MeanEnc.

[3] MoCo v2 inherits MoCo v1 and uses 'shuffled BN' in f_t. It shuffles the input to avoid cheating but the normalization still happens per-device.

[4] Replacing *all* BNs including ones in ResNet also exhibits the same pattern. Replacing BNs in projector only is noticeably faster, and generalizes to other BN-free backbones such as ViT [14].

[5] Here the reduction is approximate because we jointly forward multiple views which doubles or triples the batch size in BN; and encodings are further ℓ_2 normalized before calculating v.

	MultiCrop (Sec. 4.1)	ScaleMix (Sec. 4.2)	WeakerAug (Sec. 4.3)	StrongerAug (Sec. 4.3)	SyncBN (Sec. 4.4)	MeanEnc (Sec. 4.5)
variance change	↑	↑	↓	↑	↓	↓
encoder preference	source	source	target	source	target	target

Table 1. **Summary** of the 6 designs covered in our case studies. For each design, we list its qualitative change in intra-image variance v, and its preferred encoder. We see a consistent pattern that higher-variance designs prefer source, whilst lower-variance ones prefer target.

4.6. Summary of Studies

In total, we covered 6 variance-oriented designs in the 5 case studies described above. Interestingly, none of them achieves best result when designs are symmetrically applied to both (or neither) encoders. Instead, all of them have a single *preferred* encoder in the Siamese network. This phenomenon directly supports the importance of asymmetry for Siamese representation learning.

Moreover, we observe a consistent pattern: designs that introduce higher encoding variance generally help when placed on source encoders, whereas designs that decrease variance favor target encoders. We summarize the relation between: i) change of variance and ii) encoder preference in Tab. 1. This is well-aligned with our insight: the specific asymmetry of a relatively lower variance in target encodings than source can benefit Siamese representation learning, and not the other way around.

From the results, we do have to note that such a pattern holds within a *reasonable* range of v, and more extreme asymmetry does not always lead to better performance (*e.g.*, when further increasing source augmentation strength while having WeakerAug in target). Moreover, asymmetry is usually not the only factor in play for self-supervised frameworks; *other factors* (*e.g.* the number of views in MeanEnc) can also influence the final outcome of our pipelines.

5. Theoretical Analysis for Variance

Here we aim to provide a preliminary theoretical analysis for MoCo following [33, 34] (More details in Appendix C). Consider the following simplified InfoNCE objective:[6]

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(\mathbf{S}_{ii'}/\tau)}{\sum_{j\neq i}\exp(\mathbf{S}_{ij'}/\tau)}, \tag{1}$$

where N is batch size, τ is temperature, $\mathbf{S}_{ii'}{=}\mathbf{z}_i^\top\mathbf{z}'_i$ and $\mathbf{S}_{ij'}{=}\mathbf{z}_i^\top\mathbf{z}'_j$ are pairwise similarities between source encodings $\mathbf{z}_i$ and targets $\mathbf{z}'_i$ (target weights and encodings all come with prime $'$). For MoCo, gradients are only back-propagated through the source $\mathbf{z}_i$, but *not* $\mathbf{z}'_i$ or $\mathbf{z}'_j$.

Now, let's take the last linear layer immediately before $\mathbf{z}$ as an example for analysis. Let $\mathbf{f}$ be the input features of this layer, W be its weight matrix (so $\mathbf{z}{=}W\mathbf{f}$), and denotes coefficients $\alpha_{ij'}{=}\exp(\mathbf{S}_{ij'}/\tau)/\sum_{k\neq i}\exp(\mathbf{S}_{ik'}/\tau)$, we can write the gradient flow of W as:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}W} = W'\frac{1}{\tau N}\sum_{i=1}^{N}\sum_{j\neq i}\alpha_{ij'}(\mathbf{f}'_j-\mathbf{f}'_i)\mathbf{f}_i^\top. \tag{2}$$

To study the behavior of gradients especially w.r.t. our variance of interest, we can model intra-image variance as an *additive noise* in $\mathbf{f}$ (and $\mathbf{f}'$) that affects training. Specifically, let $\tilde{\mathbf{f}}$ be the feature corresponding to the original image, we can assume:

- Source features $\mathbf{f}_i{=}\tilde{\mathbf{f}}_i{+}\mathbf{e}_i$, with $\mathbb{E}[\mathbf{e}_i]{=}\bar{\mathbf{e}}$ and $\mathbb{V}[\mathbf{e}_i]{=}\Sigma$;
- Target side $\mathbf{f}'_i{=}\tilde{\mathbf{f}}_i{+}\mathbf{e}'_i$, with $\mathbb{E}[\mathbf{e}'_i]{=}\bar{\mathbf{e}}'$ and $\mathbb{V}[\mathbf{e}'_i]{=}\Sigma'$.

$\mathbb{E}[\cdot]$ computes expectation and $\mathbb{V}[\cdot]$ outputs variance. Note that $\tilde{\mathbf{f}}_i$ and $\tilde{\mathbf{f}}_j$ are from different images, while $\mathbf{e}_i$, $\mathbf{e}'_i$ and $\mathbf{e}'_j$ model intra-sample variance that comes from multiple sources, *e.g.*, input augmentations, BNs with different batch sizes (Sec. 4.4), *etc*. Due to the independent augmentation process, these noises are modeled as independent of each other.

Under such setting, we can arrive at the following result (detailed derivations in Appendix C) to better understand our observation from a theoretical perspective:

Higher variance on the target side is not necessary and can be less stable. With higher variance on the target side (*i.e.*, Σ' has larger eigenvalues), the variance of the gradient w.r.t. W, $\mathbb{V}[\mathrm{d}\mathcal{L}/\mathrm{d}W]$, will become larger without affecting its expectation $\mathbb{E}[\mathrm{d}\mathcal{L}/\mathrm{d}W]$. Intuitively, this asymmetry comes from an asymmetric structure in Eq. (2): there is a subtraction term $(\mathbf{f}'_j{-}\mathbf{f}'_i)$ on the target side, but not on the source side ($\mathbf{f}_i$). To make the training dynamics more stable, maintaining a relative lower variance on the target side than source is preferred.

6. Generalization Studies and Results

The keyword of this section is *generalization*, for which we study our insight for Siamese learning under various conditions. Specifically for MoCo v2, we study the behavior of asymmetric designs by training with longer schedules, and by composing multiple designs together. As a by-product, our final model achieves state-of-the-art on ImageNet, and performs well beyond when transferred to other datasets. Besides MoCo v2, we seek generalizations across more frameworks and backbones and find it also holds well. Unless otherwise specified, all the evaluations are top-1 linear probing accuracy on ImageNet [13].

[6] We make two simplifications to InfoNCE [28] by ignoring ℓ_2 normalization and the positive term $\exp(\mathbf{S}_{ii'}/\tau)$ in the denominator [42].

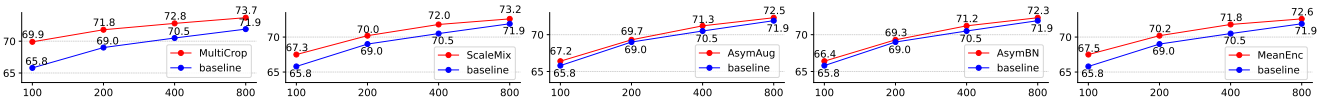


Figure 3. Generalization to **longer pre-training**. Here y-axis is accuracy (%) and x-axis is number of epochs (log-scale). Asymmetric designs consistently outperform the baseline. MultiCrop as the single strongest one reaches 73.7% at 800-ep *without* loss symmetrization.

(%)	baseline	ScaleMix	AsymBN	MeanEnc
MoCo v3 [11]	69.9	*70.7*	*70.1*	*70.6*
asym., 2× / Δ	*69.7*	+1.0	+0.4	+0.9
SimCLR [8]	65.0	*66.3*	*65.8*	*66.4*
asym., 2× / Δ	*65.0*	+1.3	+0.8	+1.4
BYOL [18]	69.5	*70.4*	*69.9*	*69.7*
asym., 2× / Δ	*69.0*	+1.4	+0.9	+0.7
SimSiam [10]	67.8	*68.7*	*68.0*	*68.0*
asym., 2× / Δ	*67.4*	+1.3	+0.6	+0.6
Barlow Twins [44]	66.8	*67.3*	*66.6*	*67.1*
asym. / Δ	*66.4*	+0.9	+0.2	+0.7

Table 2. Generalization to **more frameworks**. We cover *5* of them and convert each to and *asymmetric* one first. In the second column, we show similar results using our asymmetric versions compared to the original ones at 100-ep (in gray), optionally with 2× training schedules.[7] On top of these, we find asymmetric designs help learning across the board: third to fifth columns list accuracies and improvements over the asymmetric baseline.

6.1. Longer Training

The first generalization is to longer training schedules. Most Siamese learning frameworks [6, 8, 18], including our baseline MoCo v2, produce substantially better results in linear probing with more training epochs. Meanwhile, lower variance in target – in the extreme a *fixed* target per image, could result in *faster* convergence closer to supervised learning where longer training is not as helpful [20]. We run our baseline with the five asymmetric setups studied in Sec. 4 for 200, 400 and 800 epochs to check the behaviors, and put the trends in Fig. 3. Overall, *all* the asymmetric models outperform the baseline across different epoch numbers. The maintained gap suggests the gain from asymmetry cannot be simply explained away by faster convergence.

6.2. More Frameworks

Next we examine the generalization to other frameworks. Roughly ranked by its similarity to our baseline MoCo v2 from closest to furthest, they are: i) *MoCo v3* [11], where the memory bank is replaced by large batch sizes [43]; ii) *SimCLR* [8], where no momentum encoder is needed; iii) *BYOL* [18], where the contrastive formulation is challenged by learning only on comparing positive pairs; iv) *SimSiam* [10], where neither momentum encoder nor negative pairs are required; and v) *Barlow Twins* [44], where a fully symmetric pipeline for Siamese learning is discovered. Note that we only outlined major differences above and more subtleties (including detailed setup for each framework in this paper) are found in Appendix D.

(%)	baseline	ScaleMix	AsymBN	MeanEnc
MoCo v3, ViT [11]	69.1	*69.1*	*69.4*	*69.4*
asym., 2× / Δ	*68.7*	+0.4	+0.7	+0.7

Table 3. Generalization to **ViT** [14], a new architecture gaining popularity in vision and is recently studied in MoCo v3 [11]. The procedure and table format follow Tab. 2.

For ease of applying asymmetric designs to these frameworks, we first convert their *symmetrized* components to an asymmetric form following our source-target formulation. A popular one is loss symmetrization, used by all except Barlow Twins. We remove it by only forwarding a pair of views through the network *once* (instead of *twice*) per iteration. Intuitively, training 2× as long can roughly compensate for the symmetrized loss with fair amount of compute, as discussed in Sec. 2 and analyzed in [10]. Moreover, methods without momentum encoders [8, 10, 44] reuse source encoders for targets. In such cases, we explicitly maintain a target encoder by using an online clone of the source one, and *stopping gradients* from flowing into the branch – a choice deviated from SimCLR and Barlow Twins [8, 44]. We show in Tab. 2 (second column) that our asymmetric versions work similarly in accuracy compared to the original ones, despite the above modifications.[7]

We pick ScaleMix, AsymBN and MeanEnc as three representative designs which range from encoder inputs to outputs. MultiCrop is relatively well studied in the literature [6, 7] and we find it non-trivial to train MultiCrop with large batch sizes [8, 11, 18, 44]. More recent frameworks [11, 18, 44] already employ stronger *asymmetric* augmentation recipes [18] like AsymAug. Thus we did not include them in our comparisons listed in Tab. 2 (last three columns). Our asymmetric source-target designs generalize well beyond MoCo v2, showing consistent improvements across the board with same number of pre-training epochs.

6.3. ViT Backbone

With MoCo v3, we also benchmarked a newly proposed backbone: ViT [14]. We follow the same procedure by first building an asymmetric baseline and then applying different designs (detailed in Appendix D). Again, we find asymmetry works well (Tab. 3). The only notable difference is the reduced gap for ScaleMix, which is likely related to patches fed for ViT *not aligned* with ScaleMix masks [22].

[7] We keep all the optimization hyper-parameters the same when running the asymmetric version. The results can be further improved when *e.g.* learning rate is adjusted following the batch size change [17].

	Food-101	CIFAR-10	CIFAR-100	Birdsnap	SUN-397	Cars	Aircraft	VOC-07	DTD	Pets	Caltech-101	Flowers
Supervised	72.3	93.6	78.3	53.7	61.9	66.7	61.0	87.5	74.9	91.5	**94.5**	94.7
SimCLR [8]	68.4	90.6	71.6	37.4	58.8	50.3	50.3	85.5	74.5	83.6	90.3	91.2
BYOL [18]	75.3	91.3	78.4	57.2	62.2	67.8	60.6	82.5	75.5	90.4	94.2	**96.1**
NNCLR [15]	76.7	**93.7**	**79.0**	**61.4**	62.5	67.1	**64.1**	83.0	75.5	**91.8**	91.3	95.1
Ours, 1600-ep	**79.4**	92.8	77.8	58.5	**67.8**	**69.7**	59.3	**93.8**	**80.2**	87.2	93.1	92.5

Table 4. Generalization by **transferring our model** to 12 different downstream datasets with linear probing. We follow the protocol of [15, 18] and report results on the test set. For VOC-07, we cite the improved numbers from [44] for fair comparisons. Our 1600-ep model achieves best results on 5 out of 12, while being less competitive on tasks with iconic images (such as CIFAR [23] and Aircraft [26]).

6.4. Design Compositions

As another aspect for generalization, we compose multiple asymmetric designs together and check their joint effect on representation quality. To this end, we fall back to our MoCo v2 baseline (100-ep) and start from our strongest single asymmetric design, MultiCrop. When pairing it with other two input designs (ScaleMix an AsymAug), we find their added value has mostly diminished so we did not include them. On the target side, we first enabled SyncBN, and then enabled MeanEnc (n_t =2) to reduce variance, and both designs further improved performance:

compositions	none	+MultiCrop	+MultiCrop +AsymBN	+MultiCrop +AsymBN +MeanEnc
accuracy (%)	65.8	69.9	70.4	71.3
Δ (%)	-	+4.1	+4.6	+5.5

While our exploration on this front is preliminary and improvement is not guaranteed (as discussed in Sec. 4.6), it indicates different asymmetric designs can be compositional.

Finally, we pre-train our best composition (shaded column above) for 1600 epochs to check its limit. We arrive at *75.6%* on ImageNet linear probing (more details in Sec. 7). This puts us in the state-of-the-art cohort [37, 41, 47] with single-node training and no other bells or whistles.

6.5. Transfer Learning

In Tab. 4, we show transfer learning results of our final ImageNet 1600-ep model to 12 standard downstream classification tasks for linear probing [8, 15, 18]. For each dataset, we search the learning rate on the validation set and report results on the test set, following the protocol of [15, 18] (see Appendix D). Our model performs competitively against the most recent NNCLR [15]), achieving best on 5 tasks but lags behind on ones with iconic images. We hypothesis it's due to MultiCrop which used local small crops. We further transferred to Places-205 [46], which focuses on scene-level understanding. We find our model indeed achieves state-of-the-art (56.8%), slightly better than SwAV [6] which also used MultiCrop. These results verify our learned representation is effective beyond ImageNet.

7. Implementation Details

We list the most important implementation details for our paper below. Other subtleties are found in Appendix D.

Variance reference. We use ImageNet val set (50k images in total), r=32 views, and the 800-ep pre-trained baseline source encoder for variance calculation.[8] Encodings are ℓ_2 normalized. To fully mimic the pre-training setting, we use online *per-batch* statistics for BN, not recorded moving-average ones from the training set.

Pre-training. By default, we adopt the same MoCo v2 setup (*e.g.*, augmentation recipe, SGD optimizer *etc.*) for experiments on our baseline. A half-cycle cosine learning rate decay schedule [25] is used given the number of pre-training epochs. Mixed-precision is enabled for efficiency.

Linear probing. Linear probing freezes backbone after pre-training, and only trains a linear classifier on top of the global image features to test the representation quality. By default on ImageNet, we use LARS [43] optimizer with batch size 4096, initial learning rate lr=1.6 (linearly scaled [17]), weight decay 0 and train the classifier for 90 epochs with a half-cycle cosine schedule following SimSiam [10]. We choose LARS over SGD as the former shows better adaptation for explorations, without the need to search hyper-parameters (*e.g.* lr) extensively for good performance. For our final model, we switched back to SGD optimizer following MoCo [20], with an initial learning rate of 120 and batch size of 256.

8. Conclusion

Through systematic studies, we have revealed an interesting correlation between the *asymmetry* of source-target variance and the representation quality for Siamese learning methods. While such a correlation is conditioned on other factors and certainly not universal, we find as guideline it is generally applicable to various training schedules, frameworks and backbones. Composing asymmetric designs helps us achieve state-of-the-art with MoCo v2, and the learned representation transfers well to other downstream classification tasks. We hope our work will inspire more research exploiting the importance of asymmetry for Siamese learning, *e.g.* for object detection transfer [19] or speeding up model convergence for carbon neutral training.

[8] A potential concern is the variance reference being biased by out-of-distribution views, since the baseline model has not seen certain data (*e.g.*, small crops) during training. To address this, we also experimented with a model pre-trained with all the asymmetric designs. The trends still hold.

References

[1] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021. 1, 2

[2] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *NeurIPS*, 2019. 2

[3] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *ECCV*, 2016. 2

[4] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "Siamese" time delay neural network. In *NeurIPS*, 1994. 1, 2

[5] Zhaowei Cai, Avinash Ravichandran, Subhransu Maji, Charless Fowlkes, Zhuowen Tu, and Stefano Soatto. Exponential moving average normalization for self-supervised and semi-supervised learning. In *CVPR*, 2021. 2

[6] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020. 1, 2, 4, 7, 8, 11

[7] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2104.14294*, 2021. 1, 2, 4, 7

[8] 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, 5, 7, 8, 11

[9] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 2, 4

[10] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, 2021. 1, 2, 3, 5, 7, 8, 9, 11

[11] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. *arXiv preprint arXiv:2104.02057*, 2021. 2, 7, 11

[12] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPRW*, 2020. 11

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 3, 4, 6

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 2, 5, 7

[15] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. *arXiv preprint arXiv:2104.14548*, 2021. 2, 8

[16] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 4

[17] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv:1706.02677*, 2017. 7, 8

[18] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020. 1, 2, 4, 5, 7, 8, 11

[19] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 1, 2, 3, 4, 8, 11

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 4, 5, 7, 8

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 3, 4, 5

[22] Zihang Jiang, Qibin Hou, Li Yuan, Daquan Zhou, Yujun Shi, Xiaojie Jin, Anran Wang, and Jiashi Feng. All tokens matter: Token labeling for training better vision transformers. *arXiv preprint arXiv:2104.10858*, 2021. 7

[23] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Tech Report*, 2009. 8

[24] Zeming Li, Songtao Liu, and Jian Sun. Momentum^2 teacher: Momentum teacher with momentum statis-

tics for self-supervised learning. *arXiv preprint arXiv:2101.07525*, 2021. 2

[25] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 8

[26] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013. 8

[27] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *CVPR*, 2020. 1, 4

[28] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 1, 4, 6, 9, 11

[29] Yun Sangdoo, Han Dongyoon, Oh Seong, Joon, Chun Sanghyuk, Choe Junsuk, and Yoo Youngjoon. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019. 4, 9

[30] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020. 2

[31] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014. 2

[32] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, 2017. 1

[33] Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs. *arXiv preprint arXiv:2102.06810*, 2021. 6

[34] Yuandong Tian, Lantao Yu, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning with dual deep networks. *arXiv preprint arXiv:2010.00578*, 2020. 6

[35] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *ICML*, 2020. 3, 9

[36] Xiao Wang, Daisuke Kihara, Jiebo Luo, and Guo-Jun Qi. Enaet: Self-trained ensemble autoencoding transformations for semi-supervised learning. *arXiv preprint arXiv:1911.09265*, 2019. 2

[37] Xiao Wang and Guo-Jun Qi. Contrastive learning with stronger augmentations. *arXiv preprint arXiv:2104.07713*, 2021. 2, 8, 11

[38] Yisen Wang, Weiyang Liu, Xingjun Ma, James Bailey, Hongyuan Zha, Le Song, and Shu-Tao Xia. Iterative learning with open-set noisy labels. In *CVPR*, 2018. 2

[39] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018. 5

[40] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018. 4

[41] Haohang Xu, Xiaopeng Zhang, Hao Li, Lingxi Xie, Hongkai Xiong, and Qi Tian. Seed the views: Hierarchical semantic alignment for contrastive representation learning. *arXiv preprint arXiv:2012.02733*, 2020. 2, 8

[42] Chun-Hsiao Yeh, Cheng-Yao Hong, Yen-Chi Hsu, Tyng-Luh Liu, Yubei Chen, and Yann LeCun. Decoupled contrastive learning. *arXiv preprint arXiv:2110.06848*, 2021. 6, 9

[43] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv:1708.03888*, 2017. 2, 7, 8

[44] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv preprint arXiv:2103.03230*, 2021. 1, 2, 7, 8, 11

[45] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 9

[46] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *NeurIPS*, 2014. 8

[47] Pan Zhou, Caiming Xiong, Xiao-Tong Yuan, and Steven Hoi. A theory-driven self-labeling refinement method for contrastive representation learning. *arXiv preprint arXiv:2106.14749*, 2021. 2, 8