

A Appendix

The Appendix is organized as follows:

- Section A.1 shows the details of the target and style datasets used in our downstream evaluations.
- Section A.2 provides insight on how to select an external style dataset by analyzing their style representations.
- Section A.3 includes additional information on the pretraining settings used in our experiments.
- Section A.4 includes detailed information on the downstream task settings used in our experiments.
- Section A.5 reports full downstream classification accuracy (mean and standard deviation) on our SASSL + MoCo v2 experiments.
- Section A.6 includes an additional ablation study to better understand the effect of Neural Style Transfer in the downstream performance.
- Section A.7 covers the computational requirements of our proposed method.

A.1 Target and Style Datasets

We provide the details of the image datasets used in our experiments. Table 8 covers both target and style datasets, including their size, splits and number of classes.

Table 8: **Target and Style Datasets.** Additional details on number of classes, data split and samples of the image datasets used in our experiments.

Dataset	Task	Classes	Train Split	Val. Split	Test Split
ImageNet (Deng et al., 2009)	Pretraining, Target, Style	1,000	1,281,167	–	50,000
iNaturalist ‘21 (iNaturalist 2021)	Target, Style	10,000	2,686,843	–	500,000
Diabetic Retinopathy Detection (Kaggle & EyePacs, 2015)	Target, Style	5	35,126	10,906	42,670
Describable Textures Dataset (Cimpoi et al., 2014)	Target, Style	47	1,880	1,880	1,880
Painter by Numbers (Kan, 2016)	Style	1,584	79,433	–	23,817
ImageNet 1% (Chen et al., 2020a)	Target	1,000	12,811	–	50,000
Food101 (Bossard et al., 2014)	Target	101	75,750	–	25,250
CIFAR10 (Krizhevsky, 2009)	Target	10	50,000	–	10,000
CIFAR100 (Krizhevsky, 2009)	Target	100	50,000	–	10,000
SUN397 (Xiao et al., 2010)	Target	397	76,128	10,875	21,750
Cars (Krause et al., 2013)	Target	196	8,144	–	8,041
Caltech-101 (Fei-Fei et al., 2004)	Target	102	3,060	–	6,084
Flowers (Nilsback & Zisserman, 2008)	Target	102	1,020	1,020	6,149

A.2 Style Dataset Selection

Our transfer learning results in Section 5 demonstrate that SASSL achieves improved or on-par downstream performance across multiple datasets by incorporating Neural Style Transfer (NST) as data augmentation. This raises an important question: how can we select an external style dataset to ensure downstream accuracy improvement? Here, we delve into the similarity between datasets in terms of their styles and establish its connection to the performance improvement gained by using them as external style references.

We focus on the linear-probing scenario as it freezes the representation model, forcing the classification head to rely on the representations learned during pretraining (rather than updating them as is the case with fine-tuning) to distinguish between the target categories.

As an example, in our linear probing experiments presented in Table 2, when Diabetic Retinopathy is used as the target dataset, the downstream accuracy achieved via SASSL + MoCo v2 is comparable to that of the default MoCo v2 algorithm, meaning there is no improvement in performance. We hypothesize that

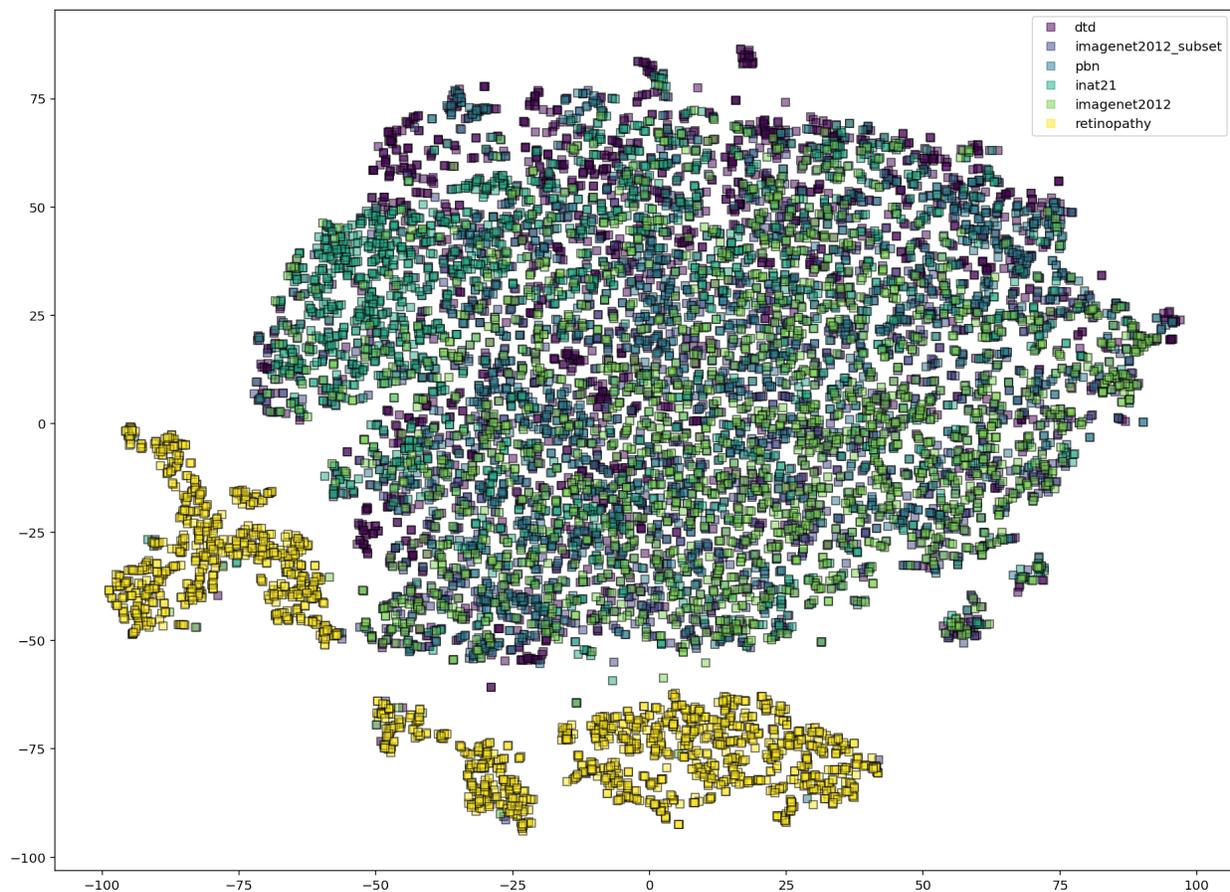


Figure 4: **t-SNE visualization of style representations.** Two-dimensional embeddings of the style representations of different datasets, extracted by the *Fast Style Transfer* method. Style embeddings of the Diabetic Retinopathy dataset (marked in yellow) form clusters that do not overlap with the rest of datasets, while embeddings from the remaining datasets are close to each other.

this is because the style representations of Diabetic Retinopathy are significantly different from those of the pretraining (ImageNet) and style datasets. Therefore, learning representations that are invariant to such a distinct set of styles does not contribute to distinguishing between target classes.

To support our hypothesis, we visualize the relationship between style representations using low-dimensional embeddings generated via t-SNE (Van der Maaten & Hinton, 2008) to capture the similarity between styles of different datasets. Style representations of each dataset, corresponding to vectors of length 100, are obtained using an InceptionV3 feature extractor, as done by the *Fast Style Transfer* algorithm. Next, we randomly select 1,800 style representations from each dataset and compute their two-dimensional embeddings using a perplexity of 30, early exaggeration of 12, and initializing the dimensionality reduction using PCA. We compute embeddings using 2,048 iterations.

Figure 4 depicts the low-dimensional embeddings obtained via t-SNE from six diverse datasets used in our transfer learning experiments. The low-dimensional representation shows that the style embeddings from Diabetic Retinopathy are significantly distinct from those of the rest of datasets, including ImageNet. This aligns with our hypothesis, suggesting that SASSL improves transfer learning when the pretraining and style references are similar to those of the target dataset. From the perspective of t-SNE embeddings, this implies that pretraining and style datasets must have a good overlap with the target dataset for SASSL to improve downstream performance.

Table 9: **SASSL + MoCo v2 downstream classification accuracy on ImageNet**. Linear probing accuracy of ResNet-50 pretrained via SASSL + MoCo v2. Mean and standard deviation reported over five random trials.

Method	Top-1 Acc. (%)	Top-5 Acc. (%)
MoCo v2 (Default)	72.55 \pm 0.67	91.19 \pm 0.34
SASSL + MoCo v2 (Ours)	74.64 \pm 0.43	91.68 \pm 0.36

A.3 Additional Experimental Details

All of our experiments were implemented in Tensorflow (Abadi et al., 2016). All our models were pretrained on 64 TPUs using a batch size of 4,096. Both the MoCo v2 models pretrained using the default augmentation and our proposed SASSL approach do not use a dictionary queue.

We used a ResNet-50 (He et al., 2016) as our representation backbone. For our projection head, we used a Multilayer perceptron with 4,096 hidden features, and an output dimensionality of 256. Our left tower used a prediction network with the same architecture as the projector, similar to the setup used in BYOL (Grill et al., 2020). Our right tower was a momentum encoder, having the same encoder and projector as the left tower, but whose parameters were an exponential moving average of the corresponding parameters in the left tower and were not trained via gradient descent. Similar to previous works, we used a momentum which started at 0.996, and which followed a cosine decay schedule ending at 1.0. For the pretraining loss, we used the InfoNCE loss with a temperature of 0.1, similar to what was done in both MoCo v2 and SimCLR.

For our pretraining augmentations, we followed the setup used in BYOL (Grill et al., 2020). The operations used and their hyperparameters (in order of application) are as follows:

1. Random cropping and rescaling to 224×224 with the area chosen randomly between 0.08 and 1.0 of the original image and with a logarithmically distributed axis ratio between $3/4$ and $4/3$. This was applied with a probability of 1.0 since it was necessary to get a fixed image shape.
2. Random horizontal flipping with a probability of 0.5 that it will be applied.
3. Random color jitter. Color jitter consists of 4 independent transformations, each of which is applied in a random order with randomly chosen values. This transform is described in greater detail in Chen et al. (2020a) and Grill et al. (2020). We used the same configuration as used in those papers.
4. Random grayscaling with a probability of 0.2 that it will be applied.
5. Random blurring with a kernel width distributed randomly between 0.1 and 2.0 pixels. Similar to Grill et al. (2020), we used a probability of 1.0 for the left view, and a probability of 0.1 for the right view.
6. Random solarization, which was only applied to the right view with a threshold of 0.5 and a probability of 0.2 that it will be applied.

When using SASSL, we applied NST after random cropping and before random horizontal flipping. Our default hyperparameters for SASSL were blending and interpolation factors drawn randomly from a uniform distribution between 0.1 and 0.3, and a probability of 0.8 that NST will be applied.

For optimization, we used the LARS optimizer (You et al., 2017) with a cosine decayed learning rate warmed up to 4.8 over the course of the first 10 epochs. Similar to previous works, we used a trust coefficient of 0.001, exempted biases and batchnorm parameters from layer adaptation and weight decay, and used a weight decay of 1.5×10^{-6} .

A.4 Downstream Training and Testing Settings

For performance evaluation on downstream tasks, all our models were trained on 64 TPUs, but using a batch size of 1,024. In this section, we provide additional details of the downstream training configuration used in

Table 10: **Additional experiments on transfer learning.** Downstream top-1 classification accuracy of ResNet-50 pretrained via MoCo v2 + SASSL on ImageNet. Accuracy evaluated on six out of twelve target datasets. SASSL generates specialized representations that improve transfer learning performance, both in linear probing and fine-tuning. Mean and standard deviation reported over five random trials.

		Target Dataset						
		ImageNet	ImageNet (1%)	iNat21	Retinopathy	DTD	Food101	
Style Dataset		<i>Linear Probing</i>						
		None (Default)	72.55 ± 0.67	53.23 ± 0.45	41.33 ± 0.2	75.88 ± 0.12	72.68 ± 0.7	73.82 ± 0.1
		ImageNet (Ours)	74.07 ± 0.46	56.87 ± 0.43	45.01 ± 0.04	75.75 ± 0.1	73.69 ± 1.22	74.43 ± 0.38
		iNat21 (Ours)	74.28 ± 0.38	56.76 ± 0.23	44.70 ± 0.37	75.75 ± 0.17	72.75 ± 1.01	74.3 ± 0.48
		Retinopathy (Ours)	74.02 ± 0.61	56.99 ± 0.26	44.9 ± 0.16	75.78 ± 0.08	73.73 ± 0.57	74.53 ± 0.29
		DTD (Ours)	74.32 ± 0.37	56.77 ± 0.36	45.08 ± 0.31	75.76 ± 0.11	74.41 ± 1.39	74.88 ± 0.32
		PBN (Ours)	74.64 ± 0.43	56.9 ± 0.18	45.02 ± 0.14	75.79 ± 0.07	72.77 ± 0.77	74.37 ± 0.19
		<i>Fine-tuning</i>						
		None (Default)	74.89 ± 0.67	51.61 ± 0.13	77.92 ± 0.14	78.89 ± 0.2	71.54 ± 0.43	87.25 ± 0.11
		ImageNet (Ours)	75.52 ± 0.23	51.74 ± 0.14	79.21 ± 0.07	79.64 ± 0.16	72.31 ± 1.85	87.48 ± 0.21
		iNat21 (Ours)	75.58 ± 0.47	51.86 ± 0.3	79.19 ± 0.12	79.6 ± 0.23	71.35 ± 1.58	87.4 ± 0.38
		Retinopathy (Ours)	75.52 ± 0.64	51.76 ± 0.26	79.23 ± 0.05	79.63 ± 0.13	72.07 ± 1.61	87.39 ± 0.19
		DTD (Ours)	75.24 ± 0.65	51.73 ± 0.19	79.24 ± 0.08	79.7 ± 0.15	70.59 ± 1.42	87.66 ± 0.23
		PBN (Ours)	75.05 ± 0.69	51.85 ± 0.16	79.2 ± 0.1	79.63 ± 0.13	71.35 ± 0.96	87.56 ± 0.38

our experiments. These cover data augmentation, optimizer and scheduler settings for both linear probing and fine-tuning scenarios.

Linear Probing Settings. We base our linear probing settings on those used by well-established SSL methods (Grill et al., 2020; Chen et al., 2020a; Kornblith et al., 2019) with some changes on the optimizer settings. We also adapt the augmentation pipeline based on the target dataset.

In all our linear probing experiments, the optimization method corresponds to SGD with Nesterov momentum using a momentum parameter of 0.9. We use an initial learning rate of 0.2 and no weight decay. We use a cosine scheduler with no warmup epochs and a decay factor of 10^{-6} . Similarly to previous work, for datasets including a validation split, we trained the linear probe on the training and validation splits together, and evaluated on the testing set.

For small target datasets (ImageNet 1%, Retinopathy, and DTD), models were trained for 5,000 iterations using a batch size of 1,024, which is consistent with the 20,000 iterations using a batch size of 256 reported by previous methods. No data augmentation is applied during training. Instead, during both training and testing, images are resized to 224 pixels along the shorter dimension followed by a 224×224 center crop and then standardized using the ImageNet statistics.

For iNat21, comprised by 2.6 million training images, we train the linear probe for 90 epochs. We empirically found that longer training significantly improved the downstream classification performance both for our proposed SASSL pipeline as well as the default augmentation pipeline.

Similarly, for ImageNet, comprised by 1.2 million training images, we also train the linear probe for 90 epochs. Additionally, we included random cropping, horizontal flipping and color augmentations (grayscale, solarization and blurring) during training.

Fine-tuning Settings. Our fine-tuning configuration follows the one used for linear-probing. In all cases, we use SGD with Nesterov momentum using a momentum parameter of 0.9. Training uses an initial learning rate of 0.2 and no weight decay. We use a cosine scheduler with no warmup epochs and a decay factor of 10^{-6} . Similarly to previous work, for datasets including a validation split, we fine-tune the model on the training and validation splits together, and evaluate on the testing set.

The number of training iterations and data augmentation depend on the target dataset, and are identical to those used for linear probing. Note that we do not run a hyperparameter sweep for selecting either the weight decay or initial learning rate, *i.e.*, these remain fixed for all experiments.

Table 11: **Additional experiments on transfer learning.** Downstream top-1 classification accuracy of ResNet-50 pretrained via MoCo v2 + SASSL on ImageNet. Accuracy evaluated on six out of twelve target datasets. SASSL generates specialized representations that improve transfer learning performance, both in linear probing and fine-tuning. Mean and standard deviation reported over five random trials.

		Target Dataset					
		CIFAR10	CIFAR100	SUN397	Cars	Caltech-101	Flowers
		<i>Linear Probing</i>					
Style Dataset	None (Default)	89.94 ± 0.24	71.93 ± 0.48	69.96 ± 0.33	53.15 ± 0.48	88.19 ± 0.75	93.39 ± 0.58
	ImageNet (Ours)	90.93 ± 0.28	73.26 ± 0.32	69.67 ± 0.26	64.87 ± 1.03	89.3 ± 0.23	95.27 ± 0.4
	iNat21 (Ours)	91.04 ± 0.16	73.29 ± 0.28	70.07 ± 0.37	63.96 ± 1.19	89.89 ± 1.07	94.7 ± 0.87
	Retinopathy (Ours)	90.8 ± 0.22	73.3 ± 0.38	69.63 ± 0.55	64.06 ± 0.79	89.17 ± 0.28	94.94 ± 0.85
	DTD (Ours)	91.04 ± 0.2	73.41 ± 0.23	69.71 ± 0.44	64.58 ± 0.71	89.3 ± 0.26	95.24 ± 0.22
	PBN (Ours)	90.85 ± 0.17	73.38 ± 0.22	69.69 ± 0.26	64.12 ± 0.95	89.59 ± 0.68	95.45 ± 0.34
	<i>Fine-tuning</i>						
	None (Default)	96.91 ± 0.12	83.4 ± 0.35	74.25 ± 0.13	83.63 ± 7.39	89.27 ± 0.15	95.75 ± 0.24
	ImageNet (Ours)	97 ± 0.09	83.21 ± 0.18	73.89 ± 0.39	90.33 ± 0.36	88.26 ± 0.35	96.6 ± 0.14
	iNat21 (Ours)	97.05 ± 0.14	83.29 ± 0.27	74.05 ± 0.3	90.04 ± 0.39	88.55 ± 0.48	95.76 ± 1.75
Retinopathy (Ours)	96.97 ± 0.09	83.68 ± 0.25	74.26 ± 0.12	89.96 ± 0.52	88.44 ± 0.44	96.34 ± 0.28	
DTD (Ours)	96.77 ± 0.41	83.28 ± 0.79	74.17 ± 0.42	89.59 ± 0.59	89.54 ± 1.94	95.59 ± 1.61	
PBN (Ours)	96.97 ± 0.11	83.36 ± 0.21	74.18 ± 0.46	89.75 ± 0.74	88.97 ± 0.78	95.77 ± 1.78	

A.5 Additional MoCo v2 Results

We complement the MoCo v2 results reported in Tables 1 and 2 by computing both their mean and standard deviation over five random trials.

Downstream performance on ImageNet. Table 9 reports the downstream performance of our SASSL + MoCo v2 representation model, pretrained and linearly probed on ImageNet. Top-1 and top-5 accuracy is computed over five random trials and reported in terms of their mean and standard deviation.

Results show SASSL pretraining and subsequent linear probing on ImageNet yield a notable boost in top-1 classification accuracy, exceeding the default model’s performance by over two standard deviations. This statistically significant improvement underscores the efficacy of incorporating SASSL augmentation into the self-supervised learning process.

Transfer learning performance. Tables 10 and 11 show the transfer learning performance of our SASSL + MoCo v2 model. Linear probing and fine-tuning top-1 accuracy is computed over five random trials and reported in terms of its mean and standard deviation.

Both linear probing and fine-tuning benefit from SASSL pretraining. Notably, linear probing achieves significant gains of up to 10%, surpassing default performance by over one standard deviation in most cases, although some target datasets show high variations. While fine-tuning exhibits a smaller improvement gap compared to default models, SASSL representation models consistently outperform baselines by up to 6% in average top-1 accuracy, showcasing their robustness across diverse datasets.

Additionally, the performance differences between different choices of style dataset are generally comparable to the measurement uncertainty, which is typically on the order of 0.3 percentage points. This suggests that the choice of style dataset does not appear to have as significant of an impact as may be expected. It seems like the main benefit is drawn from using a different style rather than due to anything about the style itself. We also show in Section A.6 that using novel styles drawn from natural images does provide a statistically significant improvement compared to synthetic styles.

It is important to highlight that this trend of shrinking improvement gaps between fine-tuning and linear probing occurs with other SSL methods as well. This is because fine-tuning adjusts the entire model to the target dataset, making pretrained weights act mainly as a refined model initialization.

A.6 Additional Ablation Studies

SASSL leverages NST to augment pre-training datasets within SSL methods. The proposed technique maintains semantic content by applying transformations solely to the image’s texture. Given a pre-training sample, SASSL treats it as the content image and employs established NST techniques to match the distribution of its low-level features to a chosen style reference. This process creates a new image where the scene’s objects, represented by high-level features (Johnson et al., 2016; Zhang et al., 2018), are preserved while the image’s texture, represented by the distribution of low-level features (Portilla & Simoncelli, 2000; Zhu et al., 2000; Heeger & Bergen, 1995), aligns with the provided style image.

The following section presents a comprehensive exploration of how the properties of the style reference affect SASSL’s generalization to downstream applications.

Effect of the Style Representation in Downstream Performance. We conduct additional experiments to better understand the effect of the style representation \hat{z} in the downstream task performance of models pretrained via SASSL. Specifically, we replace the style latent code, originally taken from a style image, by (i) i.i.d. Gaussian noise $\hat{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and (ii) the style representation of the content image $\hat{z} = z_c$. Note that latter case is equivalent to using the stylization model \mathcal{T} as an autoencoder, since no external style is imposed over the feature maps of the content image.

In both cases, we pretrain and linearly probe a ResNet-50 backbone on ImageNet using MoCo v2 equipped with SASSL. All training and downstream task settings follow our default configurations, as covered in the Appendix Section A.3 and A.4. We also use the recommended blending and interpolation factors $\alpha, \beta \sim \mathcal{U}(0.1, 0.3)$.

Table 12 shows the linear probing performance obtained by the two scenarios of interest. As reference, we also include the performance of MoCo v2 with the default data augmentation, as well as MoCo v2 via SASSL using an external style dataset (Painter by Numbers). Results show that using noise as style representation boosts top-1 accuracy by 0.24% with respect to the default data augmentation, while using the content as style reference improves performance by 0.8%. This implies that using noise as style representation hinders performance with respect to just encoding and decoding the input image via the stylization network \mathcal{T} . On the other hand, using an external style dataset boosts up performance by 2.4%, which is a significantly larger improvement over the two scenarios of interest.

Results suggest that the style reference has a strong effect on the downstream performance of the pretrained models. Either by replacing the latent representation of a style image by noise or removing the style alignment process and keeping the compression induced by the Stylization network \mathcal{T} (by forcing $\hat{z} = z_c$), the improvement provided via Style Transfer data augmentation is significantly smaller than that obtained with our full technique using external style images.

The combined insights from our ablation study and those in Section 5.5 demonstrate that simply incorporating NST into standard augmentations cannot fully account for the observed accuracy gains. Our findings suggest the existence of additional mechanisms that contribute to the delicate balance between the standard augmentation pipeline and NST. These include feature blending, pixel interpolation, feature maps to align, and style references.

A.7 Computational Requirements

We conducted additional experiments to compare the runtime of our proposed method against the default augmentation pipeline. We measured the *throughput* (augmented images per second) of SASSL relative to MoCo v2’s data augmentation. The throughput was calculated by averaging 100 independent runs on 128×128 -pixel images with a batch size of 2,048. We also report the relative change, which indicates the percentage decrease in throughput compared to the default data augmentation. All experiments were carried out on a single TPU.

Table 13 summarizes the throughput comparison. SASSL reduces throughput by approximately 20% due to the computational overhead of stylizing large batches, which involves running a forward pass of the NST model. However, empirical evidence shows that our approach achieves up to a 2% top-1 classification

Table 12: **Effect of the style representation in downstream performance.** Different style representation settings are analyzed when pretraining a ResNet-50 backbone via MoCo v2. Performance reported on a single random trial.

Augmentation	Configuration	Style Dataset	Top-1 Acc. (%)	Top-5 Acc. (%)
MoCo v2 (Default)	–	–	72.97	90.86
SASSL + MoCo v2 (Ours)	Probability: $p = 0.8$, Blending: $\alpha \in [0.1, 0.3]$ Interpolation $\beta \in [0.1, 0.3]$	Gaussian Noise $\hat{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	73.21	91.17
		$\hat{z} = z_c$	73.77	91.64
		PBN <i>(external)</i>	75.38	92.21

Table 13: **SASSL runtime.** Comparison of the throughput (processed images/second) of SASSL + MoCo v2’s data augmentation pipeline vs. the default MoCo v2’s pipeline.

Method	Throughput (images/second)	Relative Change (%)
MoCo v2 (Default)	37.45	–
SASSL + MoCo v2 (Ours)	29.48	21.28

accuracy improvement on multiple SSL techniques. Based on these findings, we consider that SASSL achieves a favorable trade-off between performance and execution time.