A APPENDIX

The appendix is organized as follows

- Sec. A.1 shows the details of the target and style datasets used in our downstream evaluations.
- Sec. A.2 provides insight on how to select an external style dataset by analyzing their style representations.
- Sec. A.3 includes additional information on the pre-training settings used in our experiments.
- Sec. A.4 includes detailed information on the downstream task settings used in our experiments.
- Sec. A.5 shows additional experiments on downstream performance using SimCLR and BYOL.
- Sec. A.6 reports few-shot classification performance of MoCo v2 equipped with SASSL.
- Sec. A.7 includes ablation studies to better understand the effect of each component of NST in the downstream performance.
- Sec. A.8 covers the computational requirements of our proposed method.
- Sec. A.9 covers recent work on semantic-aware data augmentation and neural style transfer, and discusses their differences with our proposed method.

A.1 TARGET AND STYLE DATASETS

We provide the details of the image datasets used in our experiments. Table 4 covers both target and style datasets, including their size and splits.

Table 4: **Target and Style Datasets.** Additional details on data split and number of samples of the image datasets used in our experiments.

Dataset	Task	Train Split	Val. Split	Test Split
ImageNet	Pre-training, Target, Style	$1,281,167$	–	$50,000$
ImageNet 1%	Target	$12,811$	–	$50,000$
i_Naturalist 2021	Target, Style	$2,686,843$	–	$500,000$
Diabetic Retinopathy Detection	Target, Style	$35,126$	$10,906$	$42,670$
Describable Textures Dataset	Target, Style	$1,880$	$1,880$	$1,880$
Painter by Numbers	Style	$79,433$	–	$23,817$

A.2 STYLE DATASET SELECTION

Our transfer learning results in Section 5 demonstrate that SASSL achieves improved or comparable downstream performance across multiple datasets by incorporating 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 pre-training (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 this is because the style representations of Diabetic Retinopathy are significantly different from those of the pre-training (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

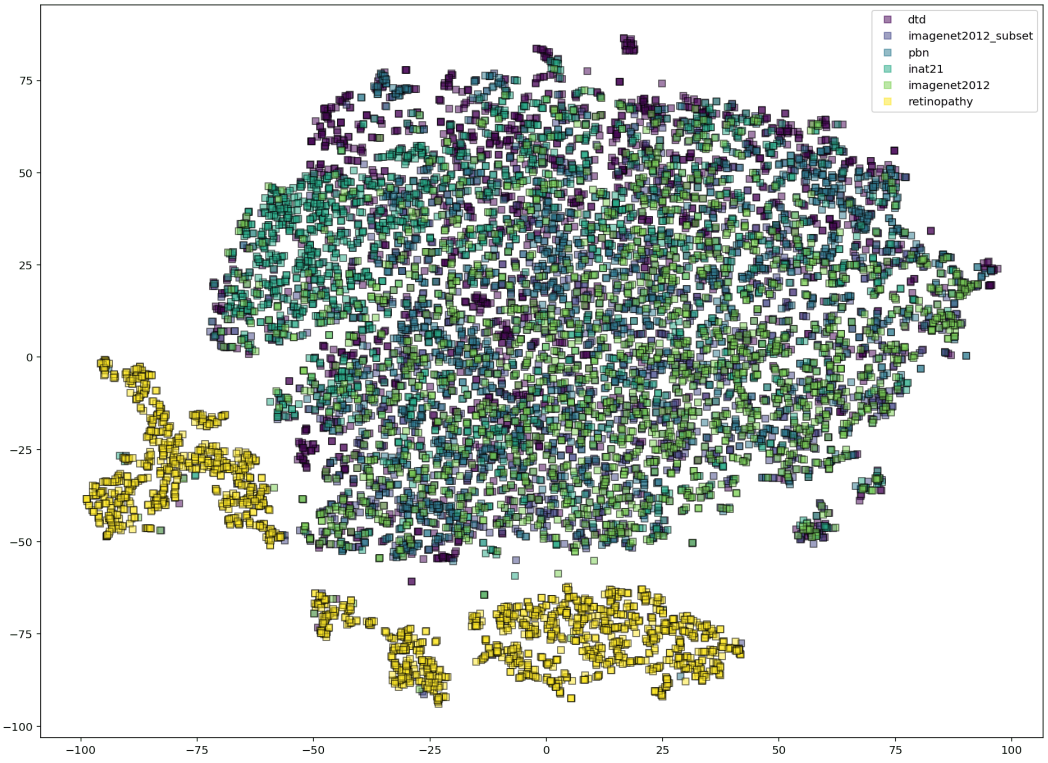

Figure 4: **t-SNE visualization of style representations.** Two-dimensional embeddings of the style representations of different datasets, extracted by the *Fast Style Transfer* method. Style embeddings of the Diabetic Retinopathy dataset (marked in yellow) form clusters that do not overlap with the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

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 *FastSt* algorithm. Next, we randomly select 1,800 style representations from each class 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 all datasets used in our transfer learning experiments. The low-dimensional representation shows that the style representations 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 pre-training and style references are similar to those of the target dataset. From the perspective of t-SNE embeddings, this implies that pre-training and style datasets must have a good overlap with the target dataset for SASSL to improve downstream performance.

A.3 Additional Experimental Details

All of our experiments were implemented in Tensorflow (Abadi et al., 2016). All our models were pre-trained on 64 TPUs using a batch size of $4,096$. Both the MoCo v2 models pre-trained 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 style transfer 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 style transfer 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 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.

Table 5: **SimCLR and BYOL Downstream classification performance on ImageNet**. Classification accuracy of SimCLR and BYOL equipped with our proposed style transfer augmentation approach (SASSL). Top-1 and top-5 accuracy reported on a single random seed.

Method	Top-1 Acc. (%)	Top-5 Acc. (%)
SimCLR (default)	68.62	88.7
SASSL + SimCLR **(Ours)**	**69.58**	**89.01**
BYOL (default)	74.09	91.83
SASSL + BYOL **(Ours)**	**75.13**	**92.12**

Table 6: **Few-shot classification performance**. One-shot and ten-shot classification performance of representations extracted via SASSL + MoCo v2 pretrained on ImageNet.

Method	Top-1 Acc. (%) One-shot	Top-1 Acc. (%) Ten-shot
MoCo v2 (default)	19.56	45.05
SASSL + MoCo v2 **(Ours)**	**20.55**	**46.73**

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.

A.5 Downstream Task Performance on other SSL Methods

We extend the downstream performance experiments reported in Sec. 5.1 by evaluating SASSL's impact on two other well-established SSL methods: SimCLR and BYOL. We pre-trained a ResNet-50 backbone using each method, both with and without SASSL, and then evaluated the resulting models on the ImageNet dataset using linear probing. In both cases, we employed the default pre-training and linear probing configurations provided by each method. SASSL was implemented with its recommended hyperparameters (blending and interpolation factor between 0.1 and 0.3, probability 0.8, and *Painter by Numbers* as external style dataset). The pre-training and fine-tuning settings for SimCLR and BYOL adhered to their default configurations.

Table 5 shows the top-1 and top-5 downstream classification accuracy. Results shows that SASSL consistently improves the downstream classification accuracy of both SimCLR and BYOL, suggesting its effectiveness across different SSL techniques.

A.6 Few-shot Classification

To further demonstrate the enhanced representation learning capabilities of SASSL, we conducted additional experiments on another downstream task: *few-shot classification*. We evaluated SASSL + MoCo v2 representations against the default MoCo v2 representations in the context of one-shot and ten-shot settings on the challenging ImageNet dataset. Note that we use the official TensorFlow Dataset's partitions, namely `imagenet2012_fewshot/1shot` and `imagenet2012_fewshot/10shot`.

Table 6 presents the classification accuracy in both scenarios. Results reveal that SASSL boosts few-shot classification top-1 accuracy by over 1% in both cases. This aligns with our classification experiments, suggesting that SASSL promotes more general image representations.

A.7 Additional Ablation Studies

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

In both cases, we pre-train 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 default blending and interpolation factors $\alpha, \beta \sim \mathcal{U}(0.1, 0.3)$.

Table 7 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 the input image. 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 pre-trained 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 $\boldsymbol{z}_s = \boldsymbol{z}_c$), the improvement provided by our proposed data augmentation is significantly smaller than that obtained with our full technique using external style images.

Effect of the Number of Stylized Layers in Downstream Accuracy. we conduct an ablation study to investigate the impact of the number of layers stylized during pre-training. We examine three scenarios: (i) stylizing the first two residual blocks (four layers corresponding to Residual blocks 1 and 2), (ii) stylizing the first four residual blocks (eight layers corresponding to Residual blocks 1 to 4), and (iii) stylizing all residual layers (ten layers corresponding to Residual layers 1 to 5).

For each scenario, we pre-trained and linearly probed a ResNet-50 on ImageNet using SASSL + MoCo v2 with its optimal configuration (blending and interpolation between 0.1 and 0.3, using Painter by Numbers as the external style dataset). To provide a clear comparison, we compared the classification accuracy of these three models to the model trained without feature alignment (using the stylization network as an autoencoder), and our complete SASSL + MoCo v2 model (stylizing all layers). Note that the latter stylizes all residual and upsampling blocks (a total of thirteen layers).

The results indicate a gradual improvement in accuracy as the number of stylized layers increases. Stylizing only the first four layers yielded no significant improvements, resulting in approximately the same accuracy as the model pre-trained without feature alignment. Stylizing the first eight layers enhanced top-1 classification accuracy by 0.34% over the model with no feature alignment. Finally, stylizing the first ten layers improved accuracy by 0.52%. These findings suggest that deeper layers have a more pronounced impact on accuracy improvement. Additionally, the model pre-trained with full stylization achieved a 1.61% improvement over the model with no feature alignment, indicating that the stylization occurring in the upsampling layers plays a dominant role in the gains provided by SASSL.

This study suggests that significant improvements can be achieved by solely stylizing deeper layers, potentially reducing the computational demands of SASSL while still enabling the extraction of robust image representations.

A.8 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

Table 7: **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 seed.

Augmentation	Configuration	Style Dataset	Top-1 Acc. (%)	Top-5 Acc. (%)
Baseline	–	–	72.97	90.86
SASSL **(Ours)**	Probability: $p = 0.8$, Blending: $\alpha \in [0.1, 0.3]$ Interpolation $\beta \in [0.1, 0.3]$	Gaussian Noise $\boldsymbol{z}_s \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	73.21	91.17
		$\boldsymbol{z}_s = \boldsymbol{z}_c$	73.77	91.64
		PBN (*external*)	**75.38**	**92.21**

Table 8: **Effect of the Number of Stylized layers in Downstream Performance.** Downstream classification accuracy of a ResNet-50 backbone pretraining via SASSL + MoCo v2, where NST is applied to a different number of layers. Performance reported on a single random seed.

Augmentation	Configuration	Style Dataset	Top-1 Acc. (%)	Top-5 Acc. (%)
Baseline	–	–	72.97	90.86
SASSL **(Ours)**	Probability: $p = 0.8$, Blending: $\alpha \in [0.1, 0.3]$ Interpolation: $\beta \in [0.1, 0.3]$	$\boldsymbol{z}_s = \boldsymbol{z}_c$	73.77	91.64
		PBN (*external*) Stylize first 4 layers	73.75	91.58
		PBN (*external*) Stylize first 8 layers	74.09	91.76
		PBN (*external*) Stylize first 10 layers	74.27	91.74
		PBN (*external*)	**75.38**	**92.21**

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 9 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 SASSL achieves up to a 2% performance improvement. We believe this represents a favorable trade-off between performance and runtime.

A.9 Comparison to Recent Data Augmentation Work

Neural Style Transfer Augmentation for Self-Supervised Learning. Recent work has explored the use of NST in self-supervised classification models to impose invariances towards improved classifiaction accuracy. Geirhos et al. (2018) studies how CNNs are biased towards texture in the context of supervised learning. Their main hypothesis is that CNN classifiers trained via supervised learning are biased towards texture. This is distinct from our approach, since we focus on preserving semantic information in self-supervised learning by exclusively distorting the style component of pre-training samples. More precisely, we decouple content and style in order to exclusively distort the style to obtain stronger self-supervised image representations. While both works rely on NST,

Table 9: **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 v2 + MoCo v2 **(Ours)**	29.48	21.28

their method uses it to create a stylized dataset (Stylized-ImageNet), whereas our approach is to incorporate style transfer in the augmentation pipeline to create stylized images on-the-fly while controlling their effect via blending and interpolation hyperparameters.

Similarly, Jackson et al. (2019) propose employing NST as a data augmentation technique for supervised learning, where the style reference is drawn from a normal distribution. In contrast, our experiments demonstrate that this approach hinders performance in the context of SSL. SASSL utilizes precomputed style representations from external datasets or samples from the content datasets themselves (in-batch stylization), resulting in improved or on-par downstream performance across style datasets. Furthermore, their approach solely considers the blending of style representations, overlooking the distortions introduced by the stylization network itself. Our work reveals that the compression generated by the stylization network, even without using a style representation, leads to a loss of information about the structure (edges and small image details) of pre-training images, deteriorating downstream performance. This is the rationale behind SASSL's reliance on both pixel interpolation and feature blending.

Zheng et al. (2019) utilize NST to create additional samples for supervised learning, using only eight style images. It's important to note that their approach does not incorporate NST as data augmentation; instead, the stylized images are fixed and added to the original training dataset. Additionally, their experiments are limited to a single classification dataset (Caltech). This contrasts sharply with our method, which proposes different stylization strategies (external and in-batch) and ways to control the stylization effect (via ablation and interpolation) to augment data on-the-fly in order to enhance SSL performance. While both methods share the use of style NST, their tasks and approaches are distinct.

Semantic-Aware Data Augmentation in Self-Supervised Learning. Purushwalkam & Gupta (2020) suggest an augmentation strategy based on natural (temporal) transformations that occur in videos as an alternative to learning from occluded (cropped) images from object-centric datasets. In contrast, our approach utilizes neural style transfer to decouple images into content and style, allowing us to apply transformations exclusively to the style of pre-training samples while preserving the content information. Their approach relies on datasets generated on video frames to incorporate temporal invariance, while SASSL can be used on standard image datasets, enabling its integration to default augmentation pipelines.

On the other hand, Lee et al. (2021) propose to include an auxiliary loss for SSL methods to capture the difference between augmented views, leading to better downstream performance on datasets where the task relies on information that may have been lost due to invariances introduced by aggressive augmentation. While their method modifies the pre-training loss and requires the use of data augmentation parameters to learn representations, SASSL does not require any additional loss components or auxiliary information from each applied augmentation to function. Our method complements the default data augmentation pipeline with a content-preserving transformation to obtain more general representations.

Finally, Bai et al. (2022) propose an alternative SSL augmentation pipeline to prevent the loss of semantic information in the learned representations by encouraging learning from weakly augmented views and gradually transitioning to strongly augmented views. The method avoids semantic shifts caused by aggressive image transformations by leveraging the tendency of SSL models to memorize clean views during early pre-training stages and by proposing a multi-stage augmentation pipeline to generate weak and strongly augmented views. Differently, our approach does not require additional augmented views or dynamic control over the loss weights during training. SASSL utilizes a content-preserving transformation that only modifies stylistic characteristics (color and texture), leaving the natural image structure and objects unchanged. Our approach incorporates a transformation into the default augmentation pipeline without altering the loss function or instance discrimination strategy.