SASSL: Enhancing Self-Supervised Learning via Neural Style Transfer

Anonymous authors
Paper under double-blind review

Abstract

Existing data augmentation in self-supervised learning, while diverse, fails to preserve the inherent structure of natural images. This results in distorted augmented samples with compromised semantic information, ultimately impacting downstream performance. To overcome this limitation, we propose *SASSL: Style Augmentations for Self Supervised Learning*, a novel data augmentation technique based on Neural Style Transfer. SASSL decouples semantic and stylistic attributes in images and applies transformations exclusively to their style while preserving content, generating diverse samples that better retain semantic information. SASSL boosts top-1 image classification accuracy on ImageNet by up to 2 percentage points compared to established self-supervised methods like MoCo, SimCLR, and BYOL, while achieving superior transfer learning performance across various datasets. Because SASSL can be performed asynchronously as part of the data augmentation pipeline, these performance impacts can be obtained with no change in pretraining throughput.

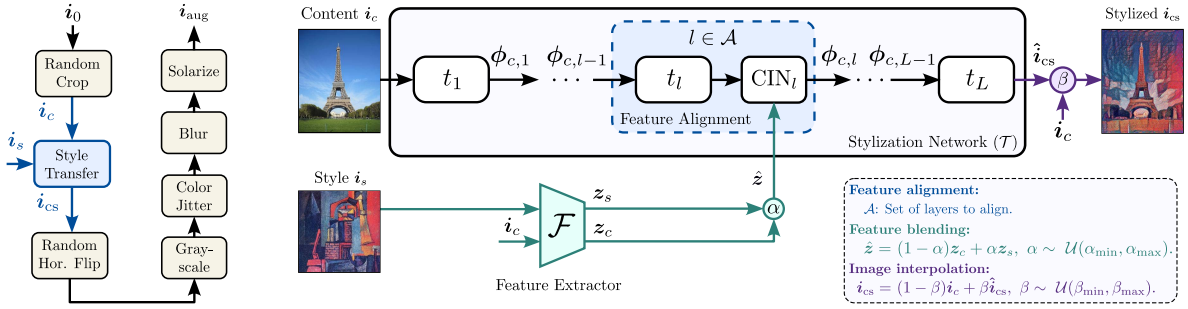
1 Introduction

Data labelling is a challenging and expensive process, which often serves as a barrier to build machine learning models to solve real-world problems. Self-supervised learning (SSL) is an emerging machine learning paradigm that helps to alleviate the challenges of data labelling, by using large corpora of unlabeled data to pretrain models to learn robust and general representations. These representations can be efficiently transferred to downstream tasks, resulting in performant models which can be constructed without access to large pools of labeled data. SSL methods have shown promising results in recent years, matching and in some cases exceeding the performance of bespoke supervised models with small amounts of labelled data.

Given the lack of labels, SSL relies on pretext tasks, *i.e.*, predefined tasks where pseudo-labels can be generated. These include contrastive learning (Chen et al., 2020a; He et al., 2020), clustering (Caron et al., 2021; 2020; Assran et al., 2022), and generative modeling (He et al., 2022; Devlin et al., 2018). Many pretext tasks involve training the model to distinguish between different views of the same input and inputs corresponding to different samples. For these tasks, the way input data is augmented is crucial to learn useful invariances and extract robust representations (Chen et al., 2020a). While state-of-the-art augmentations incorporate a wide range of color, spectral and spatial transformations, they often disregard the natural structure of an image. As a result, SSL pretraining methods may generate augmented samples with degraded semantic information, and may be less able to capture diverse visual attributes.

To tackle this challenge, we propose *Style Augmentations for Self Supervised Learning (SASSL)*, a novel SSL data augmentation technique based on Neural Style Transfer to generate semantically consistent augmented samples. In contrast to augmentation techniques operating on specific formats (e.g. pixel or spectral domain), SASSL disentangles an image into perceptual (*style*) and semantic (*content*) representations that are learned from data. Applying transformations only to the style of an image while preserving its content, we can generate images with diverse appearance that retain the original semantic properties.

Our contributions:



(a) **Data augmentation pipeline.** (b) **Style Transfer preprocessing**.

Figure 1: **Towards diverse SSL data augmentation via Neural Style Transfer.** We propose *SASSL*, a novel augmentation technique that leverages Style Transfer to create pretraining views that are semantically aware, focusing solely on modifying the image's appearance while preserving its content. SASSL combines the image's content with the texture of an external reference style, generating augmented views that better retain the image's semantic meaning. By incorporating Style Transfer into traditional SSL augmentation pipelines and controlling the stylization strength through gradual blending of style features and pixel values, SASSL promotes stronger representations compared to well-established SSL methods.

- We propose SASSL, a novel data augmentation technique based on Style Transfer that naturally preserves semantic properties while diversifying style (Section 4).
- We empirically show improved downstream performance on ImageNet Deng et al. (2009) by incorporating SASSL in methods such as MoCo, BYOL and SimCLR without hyperparameter tuning (Sections 5.1, 5.4).
- We show SASSL learns stronger representations by measuring their transfer learning capabilities on various datasets. Our method boosts linear probing performance by up to 10% and fine-tuning by up to 6% on out-of-distribution tasks (Section 5.2).

2 Related Work

2.1 Data Augmentation in SSL

Typical data augmentation methods applied to vision tasks include image cropping and resizing, flipping, rotation, color augmentation, noise addition, and solarization. Examples of methods using these are MoCo (He et al., 2020), SimCLR (Chen et al., 2020a), BYOL (Grill et al., 2020), and SimSiam (Chen & He, 2021), among others. Other work (Caron et al., 2020) shows how generating additional augmentations using this strategy can improve performance relative to two view approaches, though this strategy decreases throughput and requires additional compute. Other research explore how to select augmentation hyperparameters to learn more robust and general features (Wagner et al., 2022; Reed et al., 2021; Tian et al., 2020). In contrast, SASSL proposes a new preprocessing technique that can be incorporated in existing augmentation pipelines, boosting performance without additional hyperparameter tuning or changes in pretraining throughput.

Previous SSL work has explored semantic-aware augmentation approaches. Purushwalkam & Gupta (2020) leverage natural video transformations occuring in videos as an alternative to learning from object-centric datasets. Lee et al. (2021) introduces an auxiliary loss to capture the difference between augmented views, leading to better performance on tasks where semantic information is lost due to aggressive augmentation. Bai et al. (2022) propose an alternative augmentation pipeline to prevent loss of semantics by gradually increasing the strength of augmentations. While these methods modify the pretraining loss and require keeping track of augmentation hyperparameters, SASSL integrates seamlessly into existing pipelines without additional loss terms or auxiliary data. Our method complements default data augmentation pipelines with a content-preserving transformation to obtain stronger image representations.

2.2 Neural Style Transfer

Recent image generation and Style Transfer algorithms (Liu et al., 2021; Heitz et al., 2021; Jing et al., 2020; Yoo et al., 2019; Risser et al., 2017; Gatys et al., 2017; Chen et al., 2021a) use CNNs to measure the texture similarity between two images in terms of the distance between their neural activations. Feature maps of a pretrained classifier such as VGG-19 (Simonyan & Zisserman, 2014) are extracted and low-order moments of their distribution (Kessy et al., 2018; Huang & Belongie, 2017; Sheng et al., 2018) are used as texture descriptors. By matching such feature statistics, these techniques have shown promising results transferring texture between arbitrary images, improving over classic texture synthesis approaches (Portilla & Simoncelli, 2000; Zhu et al., 2000; Heeger & Bergen, 1995).

A large body of work focuses on *artistic* applications, reproducing an artwork style over a scene of interest. These methods adopt either (i) an *iterative optimization* approach (Risser et al., 2017; Gatys et al., 2016; 2017; Li et al., 2017a), where an initial guess is gradually transformed to depict a style of interest or (ii) an *autoencoding* approach (Liu et al., 2021; Li et al., 2017b; Chen et al., 2021a; Wang et al., 2020), where one or more CNN image generators are trained to impose a target texture in a single forward pass. While selecting an approach implies a trade-off between synthesis quality and computational cost, in both cases the generated stylization shows an unnatural appearance, *i.e.*, it often lacks the qualities of a real-world scene.

In the context of data augmentation for supervised learning, Geirhos et al. (2018) and Zheng et al. (2019) addressed texture bias and generated training samples via pre-stylized datasets or stylizing using a small collection of style images. Hong et al. (2021) explored Style Transfer to improve robustness against adversarial attacks. Jackson et al. (2019) incorporated Style Transfer as a transformation in the augmentation pipeline. While their approach of randomly mixing content and style representations yield promising results, it neglects potential distortions introduced by the Style Transfer network bottleneck. SASSL, in contrast, integrates Style Transfer in a self-supervised setting. Our approach generates diverse augmentations via either pre-computed style representations from external datasets or in-batch stylization with training samples as style references. Importantly, SASSL preserves semantic information through pixel interpolation and feature blending, mitigating the loss of details inherent in Style Transfer networks.

3 Preliminaries

3.1 Self-Supervised Learning

Traditional SSL methods learn compressed representations by maximizing the agreement between differently augmented views of the same data example in a latent space. They do this following the template originally proposed by SimCLR (Chen et al., 2020a). In this setup the input is split into multiple views using data augmentations, encoded into a representation, and then further projected into an embedding over which the loss is computed. There are many potential augmentations that can be used, including (but not limited to) random cropping, flipping, color jitter, blurring, and solarization. By maximizing the similarity of augmented training samples, the network learns to create robust representations that separate meaningful semantic content from simple distortions that could occur in the real world, and which should not affect the semantic content of an image.

Given a batch of N input images $\{\boldsymbol{i}_k\}_{k=1}^{N}$, $2N$ augmented samples are generated by applying distinct transformations to each image. These transformations correspond to the same data augmentation pipeline. Let R correspond to all possible augmentations. Then, positive pairs correspond to augmented views of the same input sample, and negative pairs correspond to views coming from different input images. Based on this, the $2N$ augmented samples $\{\tilde{\boldsymbol{i}}_l\}_{l=1}^{2N}$ can be organized so that indices $l = 2k-1$ and $l = 2k$ correspond to views of the k-th input sample

$$\tilde{\boldsymbol{i}}_{2k-1} = r(\boldsymbol{i}_k), \quad \tilde{\boldsymbol{i}}_{2k} = \hat{r}(\boldsymbol{i}_k), \quad \hat{r}, r \sim R \tag{1}$$

Once augmented views are obtained, a representation is computed using an image encoder (typically a CNN model). The representations are then fed to a projection head which further compresses them into a lower-dimensional manifold where different views of the same image are close together and those from different

images are far apart. Let h and g be the encoder (e.g. a ResNet-50 backbone) and projection head (e.g. an MLP layer), respectively. Then, embeddings are obtained for each augmented sample as $\boldsymbol{z}_l = g \circ h(\tilde{\boldsymbol{i}}_l)$.

SimCLR uses the normalized temperature-scaled cross entropy loss (NT-Xent) to learn how to identify positive pairs of augmented samples. First, the cosine similarity of every pair of embeddings is computed

$$s_{m,n} = \frac{\langle \boldsymbol{z}_m, \boldsymbol{z}_n \rangle}{\|\boldsymbol{z}_m\| \|\boldsymbol{z}_n\|} \tag{2}$$

The model is then trained using a contrastive loss by comparing the embeddings of positives, forcing them to be similar to each other. Since the loss is normalized, it naturally forces the representations of views from two different images (negatives) to be distant from each other.

$$\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} \big[\ell(2k-1, 2k) + \ell(2k, 2k-1)\big], \quad \ell(m,n) = -\log\left(\frac{\exp(s_{m,n}/\tau)}{\sum_{l=1}^{2N} \mathbb{1}_{m \neq n} \exp(s_{m,l}/\tau)}\right) \tag{3}$$

where $\tau \in \mathbb{R}_{++}$ is the temperature factor and $\mathbb{1}$ the indicator function. While SimCLR is a simple framework, it pushed the state-of-the-art significantly on a wide range of downstream tasks including image classification, object detection, and semantic segmentation.

Follow up works to SimCLR such as MoCo (Chen et al., 2020b; 2021b), BYOL (Grill et al., 2020) and SimSiam (Chen & He, 2021), among others (Caron et al., 2020; 2021; Assran et al., 2022; Zbontar et al., 2021; Bardes et al., 2021), have largely maintained this template, but have proposed modifications to this setup (e.g. new losses, architectures, or augmentation strategies) which attempt to further improve the downstream task performance.

3.2 Neural Style Transfer

Style Transfer techniques combine the semantics (*content*) of an image with the visual characteristics (*style*) of another image. These assume that the statistics of shallower layers of a trained CNN encode style, while deeper layers encode content. Seminal techniques are based on an optimization-based approach, passing a pair of content and style images to a CNN encoder and optimizing over a randomly initialized image to produce activations with similar statistics to the style image at shallower layers and similar activations to the content image at deeper ones (Gatys et al., 2015). This way, a *stylized* image is generated, comprising the semantic and texture attributes of interest.

While optimization-based methods generate a diverse stylization due to a random image initialization, autoencoding methods utilize an image decoder to efficiently stylize arbitrary image pairs on a single forward pass. In what follows, we introduce the autoencoding Style Transfer technique adopted by our proposed method due on its generalization and efficiency properties. For an in-depth survey of Style Transfer methods, refer to Jing et al. (2019).

Fast Style Transfer. Dumoulin et al. (2017) proposed an arbitrary Style Transfer method with remarkable generalization properties. Their algorithm, *Fast Style Transfer*, accurately represents unseen artistic styles by training a model to predict first and second moments of latent image representations at multiple scales. Such moments are used as arguments of a special form of instance normalization, denominated *conditional instance normalization* (CIN), to impose style over arbitrary input images.

Given a content image $\boldsymbol{i}_c \in \mathbb{R}^{C \times H_c \times W_c}$ and a style image $\boldsymbol{i}_s \in \mathbb{R}^{C \times H_s \times W_s}$, Fast Style Transfer produces a stylized image $\boldsymbol{i}_{\text{cs}}$ that corresponds to

$$\boldsymbol{i}_{\text{cs}} = \mathcal{T}(\boldsymbol{i}_c, \boldsymbol{z}_s) \in \mathbb{R}^{C \times H_c \times W_c} \tag{4}$$

where $\mathcal{T}$ is a stylization network and $\boldsymbol{z}_s = \mathcal{F}(\boldsymbol{i}_s) \in \mathbb{R}^D$ is an embedding extracted from the style image via a feature extractor $\mathcal{F}$, e.g., InceptionV3 (Szegedy et al., 2016).

We assume $\boldsymbol{z}_s$ to be a contracted embedding of the style image ($D \ll CH_sW_s$). The stylization network $\mathcal{T}$ is comprised by L blocks $\{t_l\}_{l=1}^{L}$. $\mathcal{T}$ extracts high-level features from the content image, aligns them to the style embedding $\boldsymbol{z}_s$ and maps the resulting features to the pixel domain.

The style of $\boldsymbol{i}_s$ encapsulated in $\boldsymbol{z}_s$ is transferred to the content image using CIN. This is applied to a particular set of layers to impose the target texture and color scheme by aligning feature maps at different scales. We define the set of layers where CIN is applied as $\mathcal{A}$. The normalization imposed via CIN consists of an extended form of instance normalization where the target mean and standard deviation are extracted from a style representation $\boldsymbol{z}$.

Given an input $\boldsymbol{i} \in \mathbb{R}^{C\times H\times W}$ and a style representation $\boldsymbol{z} \in \mathbb{R}^{D}$, CIN is defined as

$$\hat{\boldsymbol{i}} = \mathrm{CIN}(\boldsymbol{i}, \boldsymbol{z}) \in \mathbb{R}^{C\times H\times W} \tag{5}$$

$$\hat{\boldsymbol{i}}^{(k)} = \gamma^{(k)}(\boldsymbol{z})\left(\frac{\boldsymbol{i}^{(k)} - \mathbb{E}[\boldsymbol{i}^{(k)}]}{\sigma(\boldsymbol{i}^{(k)})}\right) + \lambda^{(k)}(\boldsymbol{z}) \tag{6}$$

where $\boldsymbol{i}^{(k)}$, $k \in \{1, \ldots, C\}$ corresponds to the k-th input channel, and the sample mean $\mathbb{E}[\boldsymbol{i}^{(k)}]$ and standard deviation $\sigma(\boldsymbol{i}^{(k)})$ are computed along its spatial support. Here, $\gamma^{(k)}, \lambda^{(k)} : \mathbb{R}^{D} \mapsto \mathbb{R}$ are trainable functions that predict scaling and offset values from the latent representation $\boldsymbol{z}$ for the k-th input channel. Following this, the layers in the stylization network are characterized by

$$\boldsymbol{\phi}_{c,l} = \begin{cases} \mathrm{CIN}_l\big(t_l(\boldsymbol{\phi}_{c,l-1}), \boldsymbol{z}_s\big), & l \in \mathcal{A} \\ t_l(\boldsymbol{\phi}_{c,l-1}), & l \notin \mathcal{A} \end{cases} \tag{7}$$

where the input of $\mathcal{T}$ corresponds to $\boldsymbol{\phi}_{c,0} = \boldsymbol{i}_c$. The subscript l in the CIN operation indicates that each layer has its own γ and λ functions to normalize features independently.

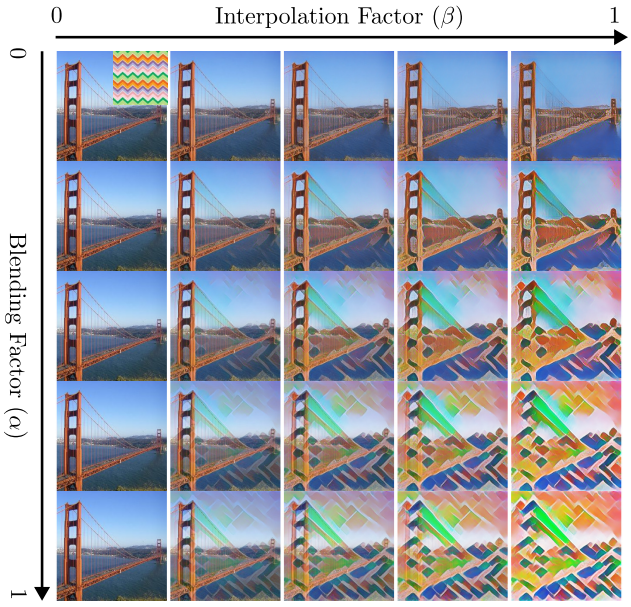


Figure 2: **Feature blending and image interpolation**. A fine-grained control over the final stylized image is obtained by introducing interpolation and blending factors α and β to operate in the feature space and pixel domain, respectively. This prevents augmented views from losing semantic information due to strong style transformations.

Our method uses the Fast Style Transfer algorithm. Given its generalization properties and low-dimensional style representations, it is a good match for our framework, where style representations from multiple domains must be efficiently extracted, manipulated and transferred.

4 Proposed Method: SASSL

We provide a detailed description of our SSL augmentation technique. First, we break down SASSL's key components and hyperparameters. Then, we tackle the problem of making the augmented images more diverse by utilizing style references from different domains in an efficient manner.

Style Transfer as data preprocessing. We incorporate Style Transfer to the default preprocessing pipeline of SSL methods. It is worth noting that SASSL is not specific to a particular SSL approach, and can be readily applied with different methods. Figure 1 shows an example of our augmentation pipeline, where Style Transfer is applied after random cropping. A raw input image $\boldsymbol{i}_0$ is cropped, producing a view that is taken as the content image $\boldsymbol{i}_c$. Given an arbitrary style image $\boldsymbol{i}_s$ (we discuss the choice of $\boldsymbol{i}_s$ below), the Style Transfer block generates a stylized image $\boldsymbol{i}_{\mathrm{cs}}$ by imposing the texture of $\boldsymbol{i}_s$ over $\boldsymbol{i}_c$. Finally, the stylized image $\boldsymbol{i}_{\mathrm{cs}}$ is passed to the remaining augmentation blocks to produce an augmented sample $\boldsymbol{i}_{\mathrm{aug}}$.

As discussed in recent work on SSL augmentation (Han et al., 2022; Chen et al., 2020a), adding a strong transformation to a self-supervised method tends to degrade performance. For this reason, it is crucial to control the amount of stylization imposed in the augmentation stage. We do so by introducing three hyperparameters: probability $p \in [0, 1]$, which dictates whether an image is stylized or not, a blending factor $\alpha \in [0, 1]$ to combine content $\boldsymbol{z}_c$ and style $\boldsymbol{z}_s$ representations, and an interpolation factor $\beta \in [0, 1]$ to combine content $\boldsymbol{i}_c$ and stylized $\hat{\boldsymbol{i}}_{\mathrm{cs}}$ images.

Algorithm 1: Style transfer augmentation block

Input: $\boldsymbol{i}_c, \boldsymbol{i}_s, \mathcal{F}, \mathcal{T}, \alpha_{\min}, \alpha_{\max}, \beta_{\min}, \beta_{\max}$
Output: $\boldsymbol{i}_{\text{cs}}$

$\boldsymbol{z}_c \leftarrow \mathcal{F}(\boldsymbol{i}_c)$; # *Style representation of content image*
$\boldsymbol{z}_s \leftarrow \mathcal{F}(\boldsymbol{i}_s)$; # *Style representation of style image*

$\alpha \sim \mathcal{U}(\alpha_{\min}, \alpha_{\max})$; # *Blending factor*
$\hat{\boldsymbol{z}} \leftarrow (1-\alpha)\boldsymbol{z}_c + \alpha \boldsymbol{z}_s$; # *Feature blending*

$\hat{\boldsymbol{i}}_{\text{cs}} \leftarrow \mathcal{T}(\boldsymbol{i}_c, \hat{\boldsymbol{z}})$;
$\beta \sim \mathcal{U}(\beta_{\min}, \beta_{\max})$; # *Interpolation factor*
$\boldsymbol{i}_{\text{cs}} \leftarrow (1-\beta)\boldsymbol{i}_c + \beta \hat{\boldsymbol{i}}_{\text{cs}}$; # *Stylized image*

Figure 3: **Style Transfer examples**. Views generated using style references from the same domain *(in-batch)* as well as other domains *(external)*. Stylization obtained using a blending factor $\alpha = 0.5$.

Given style representations extracted from content and style images $\boldsymbol{z}_c = \mathcal{F}(\boldsymbol{i}_c)$ and $\boldsymbol{z}_s = \mathcal{F}(\boldsymbol{i}_s)$, respectively, we obtain an intermediate stylized image $\hat{\boldsymbol{i}}_{\text{cs}}$ by applying a convex combination based on blending factor α.

$$\hat{\boldsymbol{i}}_{\text{cs}} = \mathcal{T}(\boldsymbol{i}_c, \hat{\boldsymbol{z}}), \quad \hat{\boldsymbol{z}} = (1-\alpha)\boldsymbol{z}_c + \alpha \boldsymbol{z}_s \tag{8}$$

Then, the final stylization output is obtained via a convex combination between the intermediate stylized image $\hat{\boldsymbol{i}}_{cs}$ and the content image $\boldsymbol{i}_c$ based on interpolation factor β.

$$\boldsymbol{i}_{\text{cs}} = (1-\beta)\boldsymbol{i}_c + \beta \hat{\boldsymbol{i}}_{\text{cs}} \tag{9}$$

Algorithm 1 describes our proposed Style Transfer data augmentation block. Figure 2 illustrates the effect of the feature blending and image interpolation operations, showcasing their importance to control the stylization effect without degrading the semantic attributes.

SASSL operates over minibatches, allowing efficient data pre-processing. Let $\boldsymbol{I}_c \in \mathbb{R}^{B \times C \times H_c \times W_c}$ and $\boldsymbol{I}_s \in \mathbb{R}^{B \times C \times H_s \times W_s}$ be content and style minibatches, respectively, comprised by B images $\boldsymbol{I}_c^{(b)}$ and $\boldsymbol{I}_s^{(b)}, b \in \{1, \dots, B\}$. Then, the stylized minibatch $\boldsymbol{I}_{\text{cs}} \in \mathbb{R}^{B \times C \times H_c \times W_c}$ is generated by applying Style Transfer between a sample from the content batch and a sample from the style batch, given an arbitrary selection criterion. We propose two alternatives for selecting style images to balance between augmentation diversity and efficiency.

Diversifying style references. In contrast to traditional data augmentation, Style Transfer can leverage a second dataset to extract style references. This opens the possibility of selecting style images from different domains, diversifying the transformations applied to the pretraining dataset. SASSL relies on two approaches for sampling style references: *external* and *in-batch* stylization.

External stylization consists on pre-computing representations of an arbitrary style dataset and sampling from them during pretraining. This allows controlling the styles to impose on the augmented views while reducing the computational overhead of Style Transfer. Under this configuration, the Style Transfer block receives a content minibatch along with a minibatch of pre-computed style representations extracted from an arbitrary style dataset, and generates a minibatch of stylized images.

On the other hand, *in-batch* stylization uses the styles depicted in the content dataset itself by using other images of the content minibatch as style references. This is of particular interest for large-scale pretraining datasets covering multiple image categories and thus textures (e.g. ImageNet). So, enabling the use of a single dataset for both pretraining and stylization is a valid alternative.

Following this, samples from the same minibatch can be used as style references by associating pairs of images in a circular fashion. More precisely, a style minibatch $\boldsymbol{I}_s \in \mathbb{R}^{B \times C \times H_c \times W_c}$ is generated by applying a circular shift on the content minibatch indices

$$\boldsymbol{I}_s^{(b)} = \boldsymbol{I}_c^{((b-b_0) \bmod B)} \tag{10}$$

where mod denotes the modulo operation and b_0 is an arbitrary offset. Figure 3 shows ImageNet samples stylized using pre-computed style representations from the Painter by Numbers dataset (Kan, 2016) via *external* stylization, as well as using other ImageNet samples taken from the content minibatch via *in-batch* stylization.

5 Experiments

5.1 Downstream Task Performance

We evaluate the downstream ImageNet classification accuracy of SSL models pretrained via SASSL on the MoCo framework. We compare a MoCo v2 model pretrained with our data augmentation vs. a MoCo v2 baseline with default augmentation (Chen et al., 2020b). Note that MoCo v2 and SimCLR use the same loss, architecture, and augmentations (they differ by MoCo's momentum encoding).

Table 1: **SASSL + MoCo v2 downstream classification performance on ImageNet**. Linear probing accuracy (%) of a ResNet-50 backbone pretrained using SASSL + MoCo v2. Mean accuracy reported over five random trials.

Method	Top-1 Acc.	Top-5 Acc.
MoCo v2 (Default)	72.55	91.19
SASSL + MoCo v2 **(Ours)**	**74.64**	**91.68**

Pretraining settings. Our pretraining setup is similar to the canonical SSL setup used to pretrain SimCLR and BYOL. We use the same loss, architecture, optimizer, and learning rate schedule as MoCo v2 for fair comparison. We pretrain a ResNet-50 encoder on ImageNet for $1,000$ epochs via SASSL. To measure downstream accuracy, we add a linear classification head on top of the pretrained backbone and train in a supervised fashion on ImageNet.

SASSL pretraining applies Style Transfer only to the left view (no changes in augmentation are applied to the right view). It is applied with a probability $p = 0.8$ using blending and interpolation factors drawn from a uniform distribution $\alpha, \beta \sim \mathcal{U}(0.1, 0.3)$. We found that this modest stylization best complimented the existing augmentations, avoiding overly-strong transformations that can hinder performance (Han et al., 2022; Chen et al., 2020a).

Results. Table 1 compares the downstream classification accuracy obtained by our SASSL augmentation approach on MoCo v2 using *external* stylization from the Painter by Numbers dataset. Results indicate our proposed augmentation improves downstream task performance by 2.09% top-1 accuracy. This highlights the value of Style Transfer augmentation in self-supervised training, where downstream task performance significantly boosts by incorporating transformations that decouple content and style. We also report results with *in-batch* stylization in Section 5.5.

5.2 Transfer Learning Performance

To understand the robustness and generalization of representations learned using our approach, we evaluate transfer learning performance across various tasks. By incorporating Style Transfer, we hypothesize that learned representations become invariant to changes in appearance (e.g. color and texture). This forces the feature extraction to rely exclusively on semantic attributes. As a result, the learned representations may become more robust to domain shifts, improving downstream task performance across datasets. We empirically show this by evaluating the transfer learning performance of representations trained using SASSL.

Downstream settings. We compare the transfer learning accuracy of ResNet-50 pretrained via MoCo v2 using SASSL against a MoCo v2 baseline with default augmentations. Models are pretrained on ImageNet and transferred to eleven target datasets: ImageNet-1% subset (Chen et al., 2020a), iNaturalist '21 (iNat21) (iNaturalist 2021), Diabetic Retinopathy Detection (Retinopathy) (Kaggle & EyePacs, 2015), Describable Textures Dataset (DTD) (Cimpoi et al., 2014), Food101 (Bossard et al., 2014), CIFAR10/100 (Krizhevsky, 2009), SUN397 (Xiao et al., 2010), Cars (Krause et al., 2013), Caltech-101 (Fei-Fei et al., 2004), and Flowers (Nilsback & Zisserman, 2008).

To have a clear idea of the effect of the style dataset in SASSL's pipeline, we pretrain five ResNet-50 backbones, each using a different style. We use ImageNet, iNat21, Retinopathy, DTD, and Painter by

Table 2: **Transfer learning performance.** Downstream top-1 classification accuracy (%) of SASSL + MoCo v2 pretrained on ImageNet. Our data augmentation method generates specialized representations that improve transfer learning performance, both in linear probing and fine-tuning. Mean accuracy reported over five random trials.

		Target Dataset											
		ImageNet	ImageNet-1%	iNat21	Retinopathy	DTD	Food101	CIFAR10	CIFAR100	SUN397	Cars	Caltech-101	Flowers
	Linear Probing												
Style Dataset	None (Default)	72.55	53.23	41.33	**75.88**	72.68	73.82	89.94	71.93	69.96	53.15	88.19	93.39
	ImageNet (**Ours**)	74.07	56.87	45.01	75.75	73.69	74.43	90.93	73.26	69.67	**64.87**	89.3	95.27
	iNat21 (**Ours**)	74.28	56.76	44.70	75.75	72.75	74.3	**91.04**	73.29	**70.07**	63.96	**89.89**	94.7
	Retinopathy (**Ours**)	74.02	**56.99**	44.9	75.78	73.73	74.53	90.8	73.3	69.63	64.06	89.17	94.94
	DTD (**Ours**)	74.32	56.77	**45.08**	75.76	**74.41**	**74.88**	**91.04**	**73.41**	69.71	64.58	89.3	95.24
	PBN (**Ours**)	**74.64**	56.9	45.02	75.79	72.77	74.37	90.85	73.38	69.69	64.12	89.59	**95.45**
	Fine-tuning												
	None (Default)	74.89	51.61	77.92	78.89	71.54	87.25	96.91	83.4	74.25	83.63	89.27	95.75
	ImageNet (**Ours**)	75.52	51.74	79.21	79.64	**72.31**	87.48	97.0	83.21	73.89	**90.33**	88.26	**96.6**
	iNat21 (**Ours**)	**75.58**	**51.86**	79.19	79.6	71.35	87.4	**97.05**	83.29	74.05	90.04	88.55	95.76
	Retinopathy (**Ours**)	75.52	51.76	79.23	79.63	72.07	87.39	96.97	**83.68**	**74.26**	89.96	88.44	96.34
	DTD (**Ours**)	75.24	51.73	**79.24**	**79.7**	70.59	**87.66**	96.77	83.28	74.17	89.59	**89.54**	95.59
	PBN (**Ours**)	75.05	51.85	79.2	79.63	71.35	87.56	96.97	83.36	74.18	89.75	88.97	95.77

Numbers (PBN) as style datasets. More precisely, we transfer five models, each pretrained on a different style, to each of eleven target datasets. We also include ImageNet as target to compare the effect of different styles on downstream performance. This leads to 60 transfer learning scenarios used to better understand the effect of various styles on different image domains.

Transfer learning is evaluated in terms of top-1 classification accuracy on linear probing and fine-tuning. All models were pretrained as described in Section 5.1. We report mean accuracy across five trials. Refer to Appendix A.4 for linear probing and fine-tuning training and testing settings.

Results. Table 2 shows the top-1 classification accuracy obtained via transfer learning. For linear probing, SASSL significantly improves the average performance on eleven out of twelve target datasets by up to 10% top-1 classification accuracy. For Retinopathy, SASSL obtains on-par linear probing accuracy to the default MoCo v2 model.

For fine-tuning, all models trained via SASSL outperform the baseline. Results show the average top-1 classification accuracy improves by up to 6%. This suggests SASSL generalizes across datasets, spanning from textures (DTD) to medical images (Retinopathy). Note that, for a fair comparison, we do not perform hyperparameter tuning. Interestingly, the relative performance obtained from different style datasets generally differs comparably to the measurement uncertainty, which is shown for these experiments in Section A.5 of the supplementary material. This suggests that the choice of style dataset is secondary in importance, while the main benefit comes from the use of SASSL itself.

5.3 Few-shot Learning Performance

To further demonstrate the representation learning capabilities of SASSL, we conduct experiments on *few-shot classification*. We compare our ResNet-50 backbone pretrained via SASSL + MoCo v2 against a MoCo v2 baseline in the context of one and ten-shot learning on ImageNet.

Table 3: **SASSL + MoCo v2 Few-shot learning performance**. One and ten-shot top-1 classification accuracy (%) of representations learned via SASSL + MoCo v2. Accuracy reported on a single trial.

Method	One-shot Acc.	Ten-shot Acc.
MoCo v2 (Default)	19.56	45.05
SASSL + MoCo v2 **(Ours)**	**20.55**	**46.73**

Table 3 shows the few-shot classification accuracy. Results reveal that SASSL boosts few-shot classification top-1 accuracy by over 1% in both one and ten-shot learning. This aligns with our previous experiments, suggesting that SASSL promotes more general image representations.

5.4 Additional Downstream Performance Evaluation

Performance on other SSL methods. To assess SASSL's broader impact, we evaluate its effectiveness on two other SSL methods, SimCLR and BYOL. We pretrain ResNet-50 backbones with each method, and then use linear probing on ImageNet to compare the quality of their learned representations. For each method, default pretraining and linear probing configurations are used. For SASSL, we employ its recommended hyperparameters ($\alpha, \beta \in [0.1, 0.3]$, $p = 0.8$) and PBN as style dataset.

Table 4 shows the accuracy attained by SimCLR and BYOL equipped with SASSL. Results show our proposed data augmentation technique boosts top-1 accuracy by approximately 1% in both cases, highlighting its potential across multiple SSL techniques.

Table 4: **SASSL downstream performance using alternative SSL methods**. Linear probing accuracy (%) on ImageNet using a ResNet-50 backbone pretrained via SimCLR and BYOL. Accuracy reported on a single trial.

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**
SwAV (Default)	70.45	89.6
SASSL + SwAV **(Ours)**	**71.3**	**90.39**

Table 5: **SASSL + MoCo downstream performance using alternative backbones**. Linear probing accuracy (%) on ImageNet using ResNet-50 (x4) and ViT-B/16 representation models. Accuracy reported on a single trial.

Backbone	Method	Top-1 Acc.	Top-5 Acc.
ResNet-50 x4 (375M)	MoCo v2 (Default)	77.2	93.32
	SASSL + MoCo v2 **(Ours)**	**78.21**	**93.98**
ViT-B/16 (86M)	MoCo v3 (Default)	75.01	92.43
	SASSL + MoCo v3 **(Ours)**	**75.51**	**92.56**

Performance on other representation models. We explore SASSL's performance on models with varying complexity and architecture. For complexity, we employ ResNet-50 (x4), a scaled-up version of the previously evaluated ResNet-50 (from 24 to 375 million parameters). This allows us to probe how SASSL scales with increased model size. In terms of architecture, we employ ViT-B/16, a Transformer-based backbone with 86 million parameters and a distinct design compared to previous CNN models.

We pretrain and linearly probe a ResNet-50 (x4) representation model on ImageNet via MoCo v2. Pretraining and downstream settings follow our default configuration, as documented in the Appendices A.3 and A.4. Similarly, we pretrain and linear probe a ViT-B/16 model on ImageNet via MoCo v3. In this case, SASSL employed a blending factor α uniformly sampled between 0.1 and 0.5.

Table 5 reports the downstream classification accuracy for ResNet-50 (x4) and ViT-B/16. ResNet-50 (x4) results show SASSL improves top-1 classification accuracy by 1.1%, mirroring its earlier improvement. Similarly, ViT-B/16 results show SASSL improves top-1 accuracy by 0.5%. These suggest that SASSL is not limited to CNN backbones, but can also be extended to ViTs. While this margin is currently smaller for ViTs, we emphasize that no hyperparameter tuning was employed in these experiments.

5.5 Ablation Studies

To shed light on how SASSL affects accuracy on ImageNet, we break down its components and assess individual contributions to downstream performance. We also study how aligning different layers in the stylization network $\mathcal{T}$ boosts accuracy. See Appendices A.6 and A.7 for additional ablations and SASSL's computational requirements.

SASSL components. For the ablation study, we cover four cases: (i) MoCo v2 with default augmentation, (ii) SASSL + MoCo v2 using in-batch representation blending and no pixel interpolation ($\beta = 1$), (iii) SASSL + MoCo v2 using in-batch representation blending and pixel interpolation, and (iv) SASSL + MoCo v2 using all its attributes (blending, interpolation and an external style dataset).

Table 6 shows our ablation study using MoCo v2 as SSL technique. Results highlight the importance of controlling the amount of stylization using both representation blending and image interpolation. With-

out image interpolation, using Style Transfer as data augmentation degrades the downstream classification performance by more than 1.5% top-1 accuracy.

On the other hand, by balancing the amount of stylization via blending and interpolation, SASSL boosts performance by more than 1.5%. This is a significant improvement for the challenging ImageNet scenario. Finally, by incorporating an external style dataset such as PBN, we further improve downstream task performance by almost 2.1% top-1 accuracy. This shows the importance of diverse style references and their effect on downstream tasks.

Table 6: **Ablation study.** Linear probing accuracy (%) for representations learned via SASSL under different configurations. Mean accuracy reported over five random trials.

Method	Configuration	Style	Top-1 Acc.	Top-5 Acc.
MoCo v2 (Default)	–	–	72.55	91.19
SASSL + MoCo v2 **(Ours)**	$p = 0.8$, $\alpha \in [0.1, 0.3]$ $\beta = 1$	ImageNet (*in-batch*)	70.87	89.33
	$p = 0.8$, $\alpha \in [0.1, 0.3]$ $\beta \in [0.1, 0.3]$	ImageNet (*in-batch*)	74.07	91.58
	$p = 0.8$, $\alpha \in [0.1, 0.3]$ $\beta \in [0.1, 0.3]$	PBN (*external*)	**74.64**	**91.68**

Table 7: **Effect of the number of stylized layers in downstream performance.** Linear probing classification accuracy (%) of a ResNet-50 model pretrained via SASSL + MoCo v2, where Style Transfer is applied using a subset of the available layers. Accuracy reported on a single trial.

Method	Stylized Layers	Top-1 Acc.	Top-5 Acc.
MoCo v2 (Default)	–	72.97	90.86
SASSL + MoCo v2 **(Ours)**	None ($\hat{\boldsymbol{z}} = \boldsymbol{z}_c$)	73.77	91.64
	First 4 layers	73.75	91.58
	First 8 layers	74.09	91.76
	First 10 layers	74.27	91.74
	All (13 layers)	**75.38**	**92.21**

Number of stylized layers. We explore how the number of layers used to apply style transfer via CIN affects downstream performance. We analyze three cases: (i) stylizing using the first two residual blocks of the Stylization Network $\mathcal{T}$ (four layers from blocks 1 and 2), (ii) the first four residual blocks (eight layers from blocks 1 to 4), and (iii) all five residual blocks (ten layers).

For each case, we pretrain and linearly probe a ResNet-50 on ImageNet using SASSL + MoCo v2 with its recommended settings ($\alpha, \beta \in [0.1, 0.3]$, $p = 0.8$) and PBN as style dataset. To fully remove the effect of a style embedding $\boldsymbol{z}_s$, our comparison includes a model pretrained using the content image itself as style reference ($\hat{\boldsymbol{z}} = \boldsymbol{z}_c$). We also compare our full SASSL + MoCo v2 model, stylizing all residual and upsampling blocks of $\mathcal{T}$ (thirteen layers).

Table 7 shows a progressive enhancement in accuracy with increasing stylization depth. Adding stylization to the first four layers showed negligible gains, mirroring the accuracy of the unaligned model. Stylizing the first eight and ten layers yielded modest improvements of 0.34% and 0.52%, respectively, implying a growing influence of deeper layers on accuracy. Notably, pretraining with full stylization, encompassing both residual and upsampling layers, attains a 1.61% accuracy boost, suggesting the importance of aligning deeper upsampling layers for downstream performance.

6 Conclusion

We propose SASSL, a novel data augmentation approach based on Neural Style Transfer that exclusively transforms the style of training samples, diversifying data augmentation during pretraining while preserving semantic attributes. We empirically show our approach outperforms well-established methods such as MoCo v2, SimCLR and BYOL by up to 2% top-1 classification accuracy on ImageNet. SASSL also improves the transfer capabilities of learned representations, enhancing linear probing and fine-tuning performance across domains by up to 10% and 6% top-1 accuracy, respectively. Our technique can be extended to other SSL methods and models with minimum hyperparameter changes, as experimentally shown.

Broader Impact Statement

This work proposes a novel data augmentation approach leveraging Neural Style Transfer to enhance Self-supervised Learning, particularly for domains with limited data or expensive annotations. Our method utilizes semantic-aware image preprocessing to extract robust representations that generalize across diverse domains. This advancement tackles the critical challenge of using unlabeled data for Deep Learning, which has many potential positive impacts in both technical and societal fronts. SASSL's style transfer component helps to reduce sensitivity to image texture, potentially improving model robustness to texture bias. However, since our method primarily modifies data augmentation, it may not fully address other potential biases arising from pre-training datasets or learning strategies.

References

Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283, 2016. URL https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf. 18

Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Florian Bordes, Pascal Vincent, Armand Joulin, Mike Rabbat, and Nicolas Ballas. Masked siamese networks for label-efficient learning. In *European Conference on Computer Vision*, pp. 456–473. Springer, 2022. 1, 4

Yingbin Bai, Erkun Yang, Zhaoqing Wang, Yuxuan Du, Bo Han, Cheng Deng, Dadong Wang, and Tongliang Liu. Rsa: Reducing semantic shift from aggressive augmentations for self-supervised learning. *Advances in Neural Information Processing Systems*, 35:21128–21141, 2022. 2

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021. 4

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014. 7, 16

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020. 1, 2, 4

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9650–9660, 2021. 1, 4

Haibo Chen, Zhizhong Wang, Huiming Zhang, Zhiwen Zuo, Ailin Li, Wei Xing, Dongming Lu, et al. Artistic style transfer with internal-external learning and contrastive learning. *Advances in Neural Information Processing Systems*, 34:26561–26573, 2021a. 3

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020a. 1, 2, 3, 5, 7, 16, 18, 19

X. Chen, S. Xie, and K. He. An empirical study of training self-supervised vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9620–9629, Los Alamitos, CA, USA, oct 2021b. IEEE Computer Society. doi: 10.1109/ICCV48922.2021.00950. URL https://doi.ieeecomputersociety.org/10.1109/ICCV48922.2021.00950. 4

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15750–15758, 2021. 2, 4

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020b. 4, 7

M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014. 7, 16

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009. 2, 16

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1

Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=BJO-BuT1g. 4

Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Pattern Recognition Workshop*, 2004. 7, 16

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015. 4

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2414–2423, 2016. 3

Leon A Gatys, Alexander S Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Controlling perceptual factors in neural style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3985–3993, 2017. 3

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018. 3

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020. 2, 4, 18, 19

Junlin Han, Pengfei Fang, Weihao Li, Jie Hong, Mohammad Ali Armin, Ian Reid, Lars Petersson, and Hongdong Li. You only cut once: Boosting data augmentation with a single cut. In *International Conference on Machine Learning*, pp. 8196–8212. PMLR, 2022. 5, 7

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 18

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020. 1, 2

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022. 1

David J Heeger and James R Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques*, pp. 229–238, 1995. 3, 21

Eric Heitz, Kenneth Vanhoey, Thomas Chambon, and Laurent Belcour. A sliced Wasserstein loss for neural texture synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9412–9420, 2021. 3

Minui Hong, Jinwoo Choi, and Gunhee Kim. Stylemix: Separating content and style for enhanced data augmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14862–14870, 2021. 3

Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1501–1510, 2017. 3

iNaturalist 2021. iNaturalist 2021 competition dataset. https://github.com/visipedia/inat_comp/tree/master/2021, 2021. 7, 16

Philip TG Jackson, Amir Atapour Abarghouei, Stephen Bonner, Toby P Breckon, and Boguslaw Obara. Style augmentation: data augmentation via style randomization. In *CVPR workshops*, volume 6, pp. 10–11, 2019. 3

Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural style transfer: A review. *IEEE transactions on visualization and computer graphics*, 26(11):3365–3385, 2019. 4

Yongcheng Jing, Xiao Liu, Yukang Ding, Xinchao Wang, Errui Ding, Mingli Song, and Shilei Wen. Dynamic instance normalization for arbitrary style transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 4369–4376, 2020. 3

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pp. 694–711. Springer, 2016. 21

Kaggle and EyePacs. Kaggle diabetic retinopathy detection, jul 2015. URL https://www.kaggle.com/c/diabetic-retinopathy-detection/data. 7, 16

Wendy Kan. Painter by numbers, 2016. URL https://kaggle.com/competitions/painter-by-numbers. 7, 16

Agnan Kessy, Alex Lewin, and Korbinian Strimmer. Optimal whitening and decorrelation. *The American Statistician*, 72(4):309–314, 2018. 3

Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2661–2671, 2019. 19

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 7, 16

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 7, 16

Hankook Lee, Kibok Lee, Kimin Lee, Honglak Lee, and Jinwoo Shin. Improving transferability of representations via augmentation-aware self-supervision. *Advances in Neural Information Processing Systems*, 34: 17710–17722, 2021. 2

Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. Demystifying neural style transfer. *arXiv preprint arXiv:1701.01036*, 2017a. 3

Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 385–395, Red Hook, NY, USA, 2017b. Curran Associates Inc. ISBN 9781510860964. 3

Songhua Liu, Tianwei Lin, Dongliang He, Fu Li, Meiling Wang, Xin Li, Zhengxing Sun, Qian Li, and Errui Ding. Adaattn: Revisit attention mechanism in arbitrary neural style transfer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6649–6658, 2021. 3

M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008. 7, 16

Javier Portilla and Eero P Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–70, 2000. 3, 21

Senthil Purushwalkam and Abhinav Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. *Advances in Neural Information Processing Systems*, 33:3407–3418, 2020. 2

Colorado J Reed, Sean Metzger, Aravind Srinivas, Trevor Darrell, and Kurt Keutzer. Selfaugment: Automatic augmentation policies for self-supervised learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2674–2683, 2021. 2

Eric Risser, Pierre Wilmot, and Connelly Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. 2017. URL http://arxiv.org/abs/1701.08893. 3

Lu Sheng, Ziyi Lin, Jing Shao, and Xiaogang Wang. Avatar-net: Multi-scale zero-shot style transfer by feature decoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8242–8250, 2018. 3

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016. 4

Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *Advances in neural information processing systems*, 33:6827–6839, 2020. 2

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 17

Diane Wagner, Fabio Ferreira, Danny Stoll, Robin Tibor Schirrmeister, Samuel Müller, and Frank Hutter. On the importance of hyperparameters and data augmentation for self-supervised learning. *arXiv preprint arXiv:2207.07875*, 2022. 2

Zhizhong Wang, Lei Zhao, Haibo Chen, Lihong Qiu, Qihang Mo, Sihuan Lin, Wei Xing, and Dongming Lu. Diversified arbitrary style transfer via deep feature perturbation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7789–7798, 2020. 3

J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3485–3492, June 2010. doi: 10.1109/CVPR.2010.5539970. 7, 16

Jaejun Yoo, Youngjung Uh, Sanghyuk Chun, Byeongkyu Kang, and Jung-Woo Ha. Photorealistic style transfer via wavelet transforms. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9036–9045, 2019. 3

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017. 18

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pp. 12310–12320. PMLR, 2021. 4

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018. 21

Xu Zheng, Tejo Chalasani, Koustav Ghosal, Sebastian Lutz, and Aljosa Smolic. Stada: Style transfer as data augmentation. *arXiv preprint arXiv:1909.01056*, 2019. 3

Song Chun Zhu, Xiu Wen Liu, and Ying Nian Wu. Exploring texture ensembles by efficient Markov chain monte carlo-toward a "trichromacy" theory of texture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):554–569, 2000. 3, 21

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

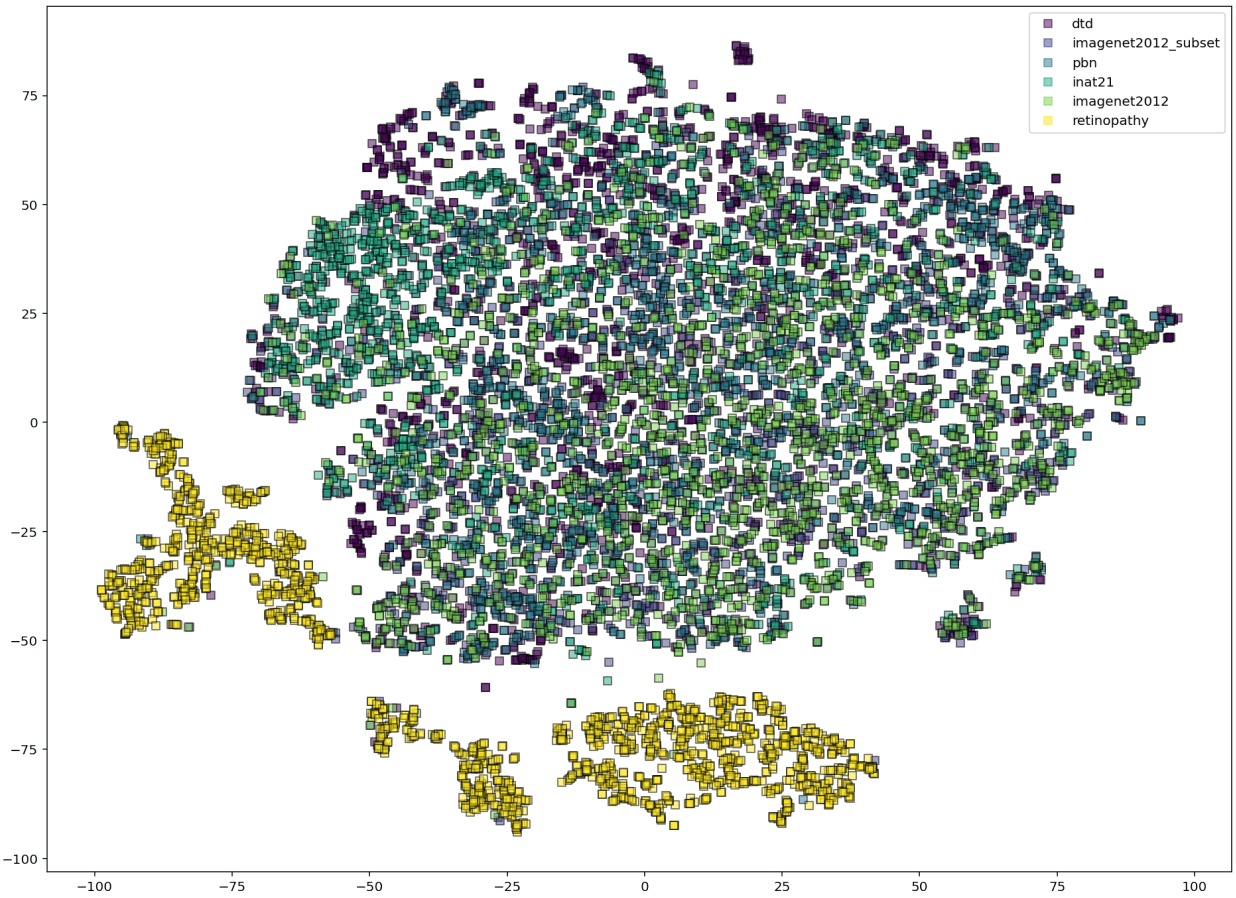


Figure 4: **t-SNE visualization of style representations.** Two-dimensional embeddings of the style representations of different datasets, extracted by the *Fast Style Transfer* method. Style embeddings of the Diabetic Retinopathy dataset (marked in yellow) form clusters that do not overlap with the 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 ± 0.67	91.19 ± 0.34
SASSL + MoCo v2 **(Ours)**	$\mathbf{74.64 \pm 0.43}$	$\mathbf{91.68 \pm 0.36}$

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
		Linear Probing					
Style Dataset	None (Default)	72.55 ± 0.67	53.23 ± 0.45	41.33 ± 0.2	$\mathbf{75.88 \pm 0.12}$	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	$\mathbf{56.99 \pm 0.26}$	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	$\mathbf{45.08 \pm 0.31}$	75.76 ± 0.11	$\mathbf{74.41 \pm 1.39}$	$\mathbf{74.88 \pm 0.32}$
	PBN (**Ours**)	$\mathbf{74.64 \pm 0.43}$	56.9 ± 0.18	45.02 ± 0.14	75.79 ± 0.07	72.77 ± 0.77	74.37 ± 0.19
		Fine-tuning					
	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	$\mathbf{72.31 \pm 1.85}$	87.48 ± 0.21
	iNat21 (**Ours**)	$\mathbf{75.58 \pm 0.47}$	$\mathbf{51.86 \pm 0.3}$	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	$\mathbf{79.24 \pm 0.08}$	$\mathbf{79.7 \pm 0.15}$	70.59 ± 1.42	$\mathbf{87.66 \pm 0.23}$
	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
		Linear Probing					
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	$\mathbf{64.87 \pm 1.03}$	89.3 ± 0.23	95.27 ± 0.4
	iNat21 (**Ours**)	$\mathbf{91.04 \pm 0.16}$	73.29 ± 0.28	$\mathbf{70.07 \pm 0.37}$	63.96 ± 1.19	$\mathbf{89.89 \pm 1.07}$	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**)	$\mathbf{91.04 \pm 0.2}$	$\mathbf{73.41 \pm 0.23}$	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	$\mathbf{95.45 \pm 0.34}$
		Fine-tuning					
	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	$\mathbf{90.33 \pm 0.36}$	88.26 ± 0.35	$\mathbf{96.6 \pm 0.14}$
	iNat21 (**Ours**)	$\mathbf{97.05 \pm 0.14}$	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	$\mathbf{83.68 \pm 0.25}$	$\mathbf{74.26 \pm 0.12}$	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	$\mathbf{89.54 \pm 1.94}$	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 $\hat{\boldsymbol{z}}$ 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 $\hat{\boldsymbol{z}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and (ii) the style representation of the content image $\hat{\boldsymbol{z}} = \boldsymbol{z}_c$. 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 $\hat{\boldsymbol{z}} = \boldsymbol{z}_c$), 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 $\hat{\boldsymbol{z}} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	73.21	91.17
		$\hat{\boldsymbol{z}} = \boldsymbol{z}_c$	73.77	91.64
		PBN (*external*)	**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.