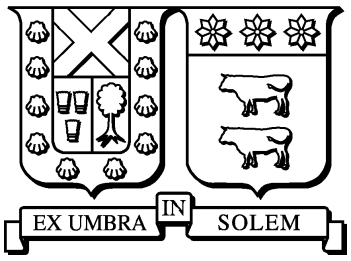
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE INFORMÁTICA

SANTIAGO – CHILE



"LEARNING ADVERSARIAL TRANSFORMATIONS VIA GRADIENT INVERSION IN A SELF SUPERVISED FRAMEWORK"

TOMÁS BARROS-EVERETT

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA CIVIL INFORMÁTICA

PROFESOR GUÍA: RICARDO ÑANCULEF

MARZO 2026

CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): [ ] Memoria o trabajo de título [X] Tesis de Postgrado

Título del trabajo: LEARNING ADVERSARIAL TRANSFORMATIONS VIA GRADIENT INVERSION IN A SELF SUPERVISED FRAMEWORK

Nombre del candidato(a): TOMÁS WILLIAM BARROS EVERETT

Carrera / Grado: MAGÍSTER EN CIENCIAS DE LA INGENIERÍA CIVIL INFORMÁTICA

Campus: CASA CENTRAL **Departamento: INFORMÁTICA**

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Ricardo Ñanculef, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

[X] El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

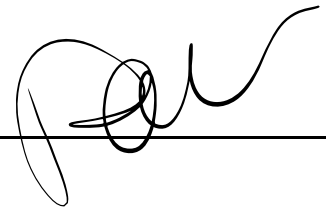
☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por **(marcar una opción)**:

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

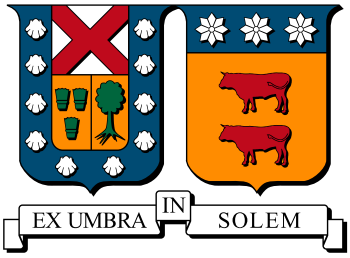
Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 29-07-2026 **Firma:** 

Estudiante o Candidato(a):

Fecha: 29-07-2026 **Firma:** 

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO – CHILE



"LEARNING ADVERSARIAL TRANSFORMATIONS VIA GRADIENT INVERSION IN A SELF SUPERVISED FRAMEWORK"

TOMÁS BARROS-EVERETT

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
MAGÍSTER EN CIENCIAS DE LA INGENIERÍA CIVIL INFORMÁTICA

PROFESOR GUÍA: RICARDO ÑANCULEF
PROFESOR CORREFERENTE: ROBERTO ASÍN

MARZO 2026

Índice de Contenidos

List of Tables

List of Figures

Motivation

Supervised Learning (SL) is heavily reliant on high-quality, manually labeled datasets. In domains requiring expert knowledge, such as medical imaging, or for large-scale applications, manual labeling is often prohibitively expensive and time-consuming. Self-Supervised Learning (SSL) addresses this limitation by developing useful data representations solely by exploiting the inherent structure of available unlabeled data. However, the quality of these representations in current state-of-the-art methods depends significantly on a fixed set of heuristic image augmentations. These manual transformations may limit the model's performance and generalization capacity, which we believe might provide an improvement opportunity in the research of this area.

Context

Supervised Learning (SL) is the traditional approach where models are trained on datasets containing pairs of inputs and manual labels to learn the transformation between them. In contrast, Self-Supervised Learning (SSL) is a paradigm where the model learns representations by identifying latent structures within unlabeled data through a ”pretext task”. In computer vision, these tasks typically involve learning an encoder function (f_θ) that maps an input image to a latent representation vector. To evaluate the quality of these learned representations, the field uses downstream tasks, which are standard problems like image classification that use a small set of labeled data. The effectiveness of the pretraining is usually quantified using the accuracy of a classification model trained on the encoder's outputs.

Term	Definition
Supervised Learning (SL)	A training paradigm in which a model learns the transformation between a set of inputs and their corresponding, manually-labelled outputs.
Self-Supervised Learning (SSL)	A method that learns data representations by exploiting the inherent structure of unlabeled data, avoiding the cost of manual labeling.
Encoder (f_θ)	A neural network that maps an input to a latent representation vector.
Latent Representation	A compressed, mathematical vector (embedding) that captures the semantic features of an image, allowing for semantic operations in embedding-space.
Pretext Task	An auxilliary task used to create a supervisory signal from the structure of the data itself. In the case of non-contrastive architectures, it is to reduce the distance between embeddings of differnt views of the same image.
Downstream Task	A standard application, such as image classification, used to test the quality and transferability of representations learned during pretraining.
Augmentation	A transformation applied to an image (e.g., cropping, flipping, or rotating) to create different ”views” of the same data point.
Backbone	The primary architectural structure of the encoder, often a standard model like ResNet-18.

Table 1: Definitions of fundamental Self-Supervised Learning concepts.

Objectives and Hypothesis

The general objective of this work is to implement an architecture capable of learning adversarial transformations within the context of self-supervised learning. The novelty lies in the combination of elements not previously used in this field, such as Spatial Transformer Networks (STNs) to learn differentiable transformations, and Gradient Reversal Layers (GRLs) to learn these transformations adversarially.

The specific objectives associated with this thesis project are to:

- Develop the proposed architecture.
- Benchmark the performance of the architecture against the state of the art.
- Evaluate the impact of our modification's parameters on model performance.
- Perform a qualitative analysis of the learned transformations.

The main hypothesis guiding this work is the following:

Learning adversarial transformations during model training can lead to improved generalization capacity of a self-supervised learning model's latent representations, providing an improvement opportunity versus the fixed set of canonical transformations used in the current state of the art.

Resumen

En años recientes, el Aprendizaje Auto-Supervisado (SSL) se ha destacado como un valioso paradigma de aprendizaje automático, permitiendo que los modelos aprendan representaciones visuales robustas sin depender de un costoso etiquetado manual. Para datos visuales, las arquitecturas de SSL no contrastivo aprenden representaciones maximizando la similitud entre las representaciones de vistas aumentadas de la misma imagen. Los métodos actuales del estado del arte dependen de aumentos de imagen heurísticos, cuya calidad impacta significativamente en el rendimiento final del modelo y en su capacidad de generalización. Proponemos una adaptación que integra el aprendizaje adversarial en el flujo de trabajo de SSL para aprender transformaciones espaciales adversariales. Nuestra arquitectura incorpora una Spatial Transformer Network (STN) para realizar aumentos de imagen diferenciables, junto con una Gradient Reversal Layer (GRL). La GRL permite que la STN se entrene de manera adversarial, maximizando la distancia entre embeddings, forzando así al encoder a desarrollar características robustas y mejorando la generalización. Evaluamos nuestro enfoque incorporando el módulo adversarial en FastSiam, una arquitectura de SSL del estado del arte. Nuestra arquitectura propuesta supera a los modelos actuales del estado del arte en CIFAR10, convergiendo a las mejores evaluaciones previas en aproximadamente un 40% menos de épocas, y estableciendo un nuevo record de 92.4% de precisión kNN a partir del 90.2 alcanzado previamente por FastSiam.

Abstract

In recent years, Self-Supervised Learning (SSL) has stood out as a valuable machine learning paradigm, allowing models to learn robust visual representations without relying on costly manual labeling. For visual data, non-contrastive SSL architectures learn representations by maximizing the similarity between representations of augmented views of the same image. Current state-of-the-art methods rely on heuristic image augmentations, the quality of which significantly impacts the final model performance and generalization capacity. We propose an adaptation that integrates adversarial learning into the SSL pipeline to learn adversarial spatial transformations. Our architecture incorporates a Spatial Transformer Network (STN) to perform differentiable image augmentations, paired with a Gradient Reversal Layer (GRL). The GRL enables the STN to be trained adversarially, maximizing the embedding-distance loss, thereby forcing the encoder to develop robust features, improving generalization. We evaluate our approach by incorporating the adversarial module into FastSiam, a state-of-the-art SSL architecture. Our proposed architecture outperforms the current state-of-the-art models on CIFAR10, converging to previous best evaluations in roughly 40% less epochs, and setting a new benchmark of 92.4% kNN accuracy from FastSiam's previously attained value of 90.2.

Chapter 1

Problem Definition

Supervised Learning (SL) is heavily reliant on high-quality, manually labeled datasets. For domains like medical imaging, where annotation requires expert knowledge, or for large-scale applications, manual labeling is often prohibitively expensive and time-consuming. Self-Supervised Learning (SSL) addresses this limitation by developing useful data representations solely by exploiting the inherent structure of available unlabeled data. The goal of SSL is to learn a transferable latent representation z from a large dataset of unlabeled data $\mathcal{D} = \{x_i\}_{i=1}^{N}$. This is done by defining a pretext task that creates a supervisory signal from the data's structure. The aim is to learn an encoder function, $f_\theta : \mathcal{X} \to \mathcal{Z}$, parameterized by θ, that maps an input image $x \in \mathcal{X}$ to a representation vector $z \in \mathcal{Z}$. The utility of the learned representation $\mathbf{z}$ is measured by its transferability to downstream tasks, meaning its performance on standard computer vision problems using labeled data. After the self-supervised pretraining phase, the encoder network $\mathbf{f}_\theta$ is typically frozen. A simple, lightweight classifier, such as a linear evaluator (a single fully-connected layer), is placed on top of the frozen encoder output. This linear layer is then trained exclusively using a small set of labeled data specific to the downstream task. The quality of the representations is quantified by reporting standard metrics on these tasks, most commonly the Top-1 and Top-5 classification accuracies. A higher accuracy indicates that the features learned during the self-supervised pretext task capture the semantic information present in the images.

Chapter 2

Related Work

2.1. Non-contrastive Self-Supervised Learning Architectures

Non-contrastive SSL represents the current state-of-the-art in self-supervised computer vision. Unlike early contrastive architectures [4, 10], which require identifying and pushing apart embeddings of negative pairs (different images) while pulling together positive pairs (augmented views of the same image), non-contrastive methods [9, 5, 11, 2] achieve robust representation learning without relying on negative pairs. Non-contrastive architectures employ a Siamese-like structure consisting of two identical neural networks (encoders), often termed the online network (f_θ) and the target network ($f_{\theta'}$) (see Fig 2.1). The architecture's pretext task is to produce the same output embedding for two differing views of the same input image.

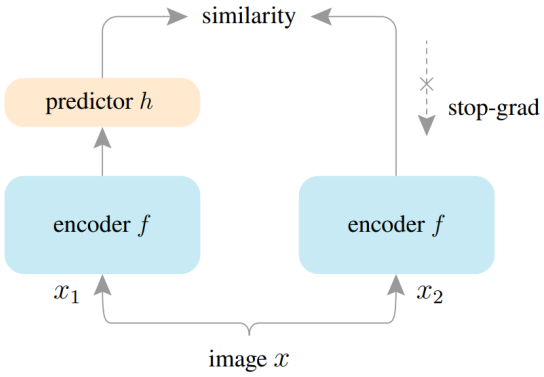


Figure 2.1: SimSiam [5]

The online network's weights, θ, are updated via gradient descent to minimize the loss, while the target network's weights, θ', are updated solely using a momentum-based moving average of the online network's weights. This asymmetry, combined with a prediction head present only on the online branch, are the key mechanisms used to prevent trivial solutions known as representational collapse [9]. The two views, v and v', of the input image are produced using a set of canonical stochastic image augmentations, such as rotation, zoom and cropping. By minimizing the distance between the embeddings of these two views, the encoder is pushed towards learning latent features that capture the semantic content of training images.

Current state-of-the-art architectures

Non-contrastive SSL methods differ primarily their training objectives, despite sharing the common architecture of having a prediction (or student) branch and a target (or teacher) branch.

BYOL (Bootstrap Your Own Latent) [9]: The standard non-contrastive architecture. It adapts SimCLR's [4] contrastive architecture but omits the concept of negative pairs. The target network's weights are not updated by gradients but are instead a momentum-based moving average of the online network's weights. This asymmetry, combined with a prediction head and a stop-gradient operation, prevents the model from collapsing into trivial

solutions.

SimSiam [5]: A further simplification of BYOL, SimSiam uses identical encoders with shared weights. It demonstrates that the momentum encoder is not strictly necessary for stability. Instead, it relies entirely on the asymmetry created by the prediction head on one branch and a stop-gradient on the target branch to learn meaningful features.

SwAV (Swapping Assignments between Views) [1]: SwAV introduces a clustering-based pretext task. Instead of comparing embeddings directly, it computes a ”code” (cluster assignment) for one view and predicts that code from the other view. By enforcing a consistency constraint between assignments across different views, the model learns features that capture the underlying semantic categories of the data.

Barlow Twins [20]: Unlike the previous methods that focus on architectural asymmetry, Barlow Twins turns the training objective into an *Information Bottleneck* problem, maximizing the embedding's information correlated with the views and minimizing the information correlated with the augmentations.

DINO (Self-distillation with no labels) [2]: DINO reinterprets the student-teacher framework through the lens of self-distillation, though the architecture remains identical to other non-contrastive methods. The main contribution of DINO is to use a vision transformer as an encoder instead of the standard ResNet, which produces properties such as semantic image segmentation. Due to the vision transformer's training requirements, it mainly works for longer training times than other methods.

FastSiam

As this work focuses on the application and modification of the FastSiam architecture, a detailed explanation of the architecture is presented. FastSiam [11] is a non-contrastive Self-Supervised Learning method that extends the simpler SimSiam architecture to achieve faster convergence and resource efficiency. In FastSiam, the prediction branch processes an input by generating an augmented view by randomly sampling a set of transformation T, and applying them to an image $v = T(x)$, then passing it through an encoder (f) and a prediction

head (h) to produce an embedding p:

$$p = h(f(v(x)))$$

The target branch processes one or more augmented views through a copy of the prediction branch's encoder (f). A stop-gradient operation is applied to the target branch output, z, meaning its weights are not updated by the loss gradient. The weights of the prediction branch are updated via backpropagation from a negative cosine similarity loss between both branches outputted vectors:

$$L(z, p) = -\frac{p}{||p||_2} \cdot \frac{z}{||z||_2}$$

FastSiam's key innovation is in how it reduces the effect of target instability in training. Instead of using just one augmented crop (view) for the target as in SimSiam, FastSiam uses K views ($K = 3$ being standard practice), and takes the mean of the outputted target vectors as the final target with which to calculate the loss. Thus, the loss function is defined as follows:

$$L(p, z_1, z_2, ...z_K) = -\frac{p}{||p||_2} \cdot \frac{\bar{z}}{||\bar{z}||_2}$$

with

$$\bar{z} = \frac{1}{K} \sum_{i=1}^{K} z_i$$

This ensemble of views stabilizes the training target by providing a more reliable estimate of the underlying feature distribution's mean value, reducing the standard error of the mean. By stabilizing the training target, FastSiam achieves faster convergence than all other state-of-the-art methods, and performs well even with small batch sizes on a single GPU.

2.2. The Role of Augmentations in SSL

[16] propose a standardized framework to unify the current state-of-the-art SSL architectures into a single template. The authors compare several SSL methods and find that many algorithmic additions, such as prediction heads and novel loss functions, have a minor impact on downstream task performance, while enhanced augmentation techniques provide more significant improvements.

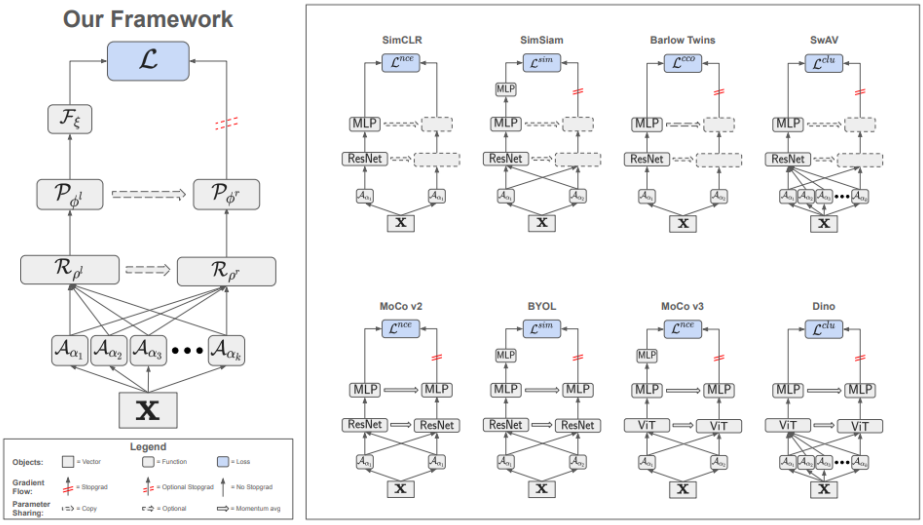


Figure 2.2: A unified framework for SSL architectures [16]

Image augmentations provide the learning signal in SSL architectures by eliminating spurious features in the training data while maintaining semantic content. The augmentations must therefore be non-trivial while also remaining strong enough to provide a learning signal. Weak or trivial transformations lead to representational collapse, where the network predicts the same vector for any input, while augmentations that are too disruptive will impede proper representation learning. The augmentations used in both SimSiam and FastSiam are as follows:

- Random resized crop
- Random horizontal flip
- Color jitter
- Random gray scale
- Gaussian blur

2.3. Spatial Transformer Network (STN)

The main obstacle in learning spatial augmentations from data is requirement of differentiability. The Spatial Transformer Network (STN) resolves this by providing a module capable of performing geometric transformations within the network, allowing parameters to be learned via backpropagation. An STN has three components:

2.3.1. Localization Network

The Localization Network $\theta = f_{loc}(U)$ takes an input feature map U and regresses a set of parameters θ that define the spatial transformation to be applied. For affine transformations, θ is a six-dimensional vector used to construct the 2×3 transformation matrix A_θ.

$$A_\theta = \begin{pmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{pmatrix}$$

2.3.2. Grid Generator

The Grid Generator uses the transformation matrix A_θ to generate a sampling grid G, which is a set of points where U should be sampled to produce the transformed feature map V. This mapping is formally defined by a matrix product:

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = A_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} \tag{2.1}$$

Where (x_i^t, y_i^t) are normalized coordinates such that $x_i^t, y_i^t \in [-1, 1]$. Notation: The s in (x_i^s, y_i^s) corresponds to a source pixel, with the t superscript denoting a target pixel.

2.3.3. Sampler

To produce the output pixel value V_i^c at location (x_i^t, y_i^t) for channel c, the sampler applies bilinear interpolation at the calculated source coordinates:

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c \max(0, 1 - |x_i^s - m|) \max(0, 1 - |y_i^s - n|)$$

This sum effectively identifies the four nearest neighbor pixels to (x_i^s, y_i^s) and performs a weighted average based on distance.

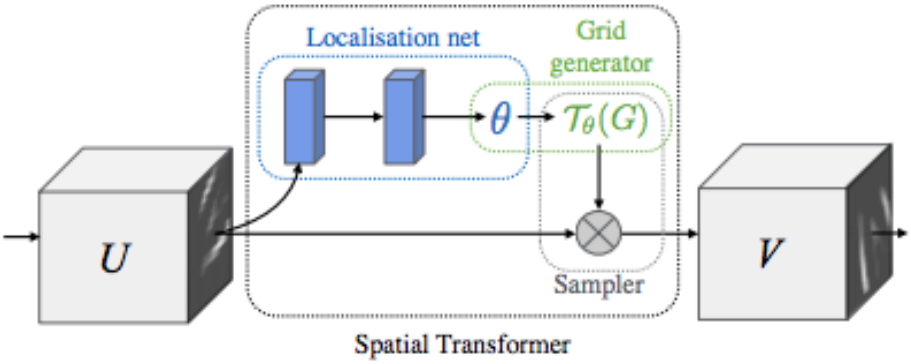


Figure 2.3: Spatial Transformer Network [12]

Transformation Learning

To train the localization network, the gradient of the loss is propagated through the sampler. Following the chain rule, the gradient with respect to the input coordinates x_i^s is:

$$\frac{\partial V_i^c}{\partial x_i^s} = \sum_n^H \sum_m^W U_{nm}^c \max(0, 1 - |y_i^s - n|) \times \begin{cases} 0 & \text{if } |m - x_i^s| \geq 1 \\ 1 & \text{if } m \geq x_i^s \\ -1 & \text{if } m < x_i^s \end{cases}$$

(With an analogous equation for $\frac{\partial V_i^c}{\partial y_i^s}$). Finally, to update the parameters θ of the localization network, we use:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \sum_i \left(\frac{\partial \mathcal{L}}{\partial V_i} \left(\frac{\partial V_i}{\partial x_i^s} \frac{\partial x_i^s}{\partial \theta} + \frac{\partial V_i}{\partial y_i^s} \frac{\partial y_i^s}{\partial \theta} \right) \right)$$

Where i denotes the i-th pixel. Since x_i^s is a linear function of θ (from Section 2.3.2), the term $\frac{\partial x_i^s}{\partial \theta}$ simplifies to the corresponding coordinate from the target grid $(x_i^t, y_i^t, 1)$.

2.4. Gradient Reversal Layer (GRL)

The Gradient Reversal Layer (GRL) is a specialized layer in a neural network designed to flip the sign of the gradient during backpropagation. It is typically used to enable two components of a network to be trained with opposing objectives, such as in [8]. During the forward pass, the GRL acts as an identity function:

$$\mathcal{F}(z) = z$$

This ensures that the layer does not distort the data signal or interfere with the primary objective being calculated by the loss function downstream. During the backward pass, the GRL inverts the gradient with a parameter λ to control the magnitude of the reversed gradient:

$$GRL\left(\frac{\partial \mathcal{L}}{\partial z}\right) = -\lambda \cdot \frac{\partial \mathcal{L}}{\partial z}$$

2.5. Relationship to Existing Methods

Current research on automatic augmentation focuses on searching through the parameter space of a discrete set of canonical transformations [6] [14]. The set of augmentations is the same as the set of stochastically applied transformations used in all SSL architectures, however, SSL architectures eliminate the need of searching for optimal parameters by sampling through the complete search space; all possible rotations, for example.

Our method can be seen as a generalization of Contrastive Learning (CL) [13] [19], in which adversarial samples are obtained by randomly sampling a set of parameters for fixed transformations, and selecting the subset with the highest loss. [7] extends this with a pair of novel, but deterministic views. With respect to spatial augmentations, our architecture acts similarly (maximizing the loss of the encoder w.r.t the augmentation parameters) when using

the set of affine transformations common to SSL architectures. However, with our approach, we can learn from any set of differentiable transformations, as will be shown in section 3.2.

The combination of a Spatial Transformer Network (STN) with a Gradient Reversal Layer (GRL) stems from research in the field of *domain adaptation*, in which models are trained to become invariant to domain shifts in a dataset's distribution. [8] introduce the GRL to train a model to learn domain-independent latent features, while [18] expand on the concept by also introducing an affine-transformation STN for data augmentation. However, our approach presents the first time this combination has been applied to the problem of self-supervised representation learning, while also being the first to extend the set of SSL augmentations beyond affine transformations, into learnable non-linear transformations.

Chapter 3

Proposed Method: Adversarial Transformation Learning

This section presents our proposed architecture. We provide a geometric motivation borrowing from Vicinal Risk Minimization [3] to justify the architectural design, followed by a description of the module components and training objective.

3.1. The Manifold Hypothesis

Let $\mathcal{X} \subseteq \mathbb{R}^d$ represent the high-dimensional input space of all possible images. Following the manifold hypothesis, *natural* images are assumed to be supported on a low-dimensional manifold $\mathcal{M} \subset \mathcal{X}$, where $\dim(\mathcal{M}) \ll d$. We define the data-generating distribution as a probability measure $\mathbb{P}_r$ supported on $\mathcal{M}$. In a finite dataset $\mathcal{D}$, we only access an empirical distribution $\hat{\mathbb{P}}_n$, represented as a sum of Dirac delta functions at the training samples $\{x_i\}_{i=1}^n$ [3]:

$$\hat{\mathbb{P}}_n = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i}$$

Since $\hat{\mathbb{P}}_n$ consists of isolated samples on $\mathcal{M}$, it provides a sparse approximation of the underlying data manifold. This sparsity limits the ability of learned representations to generalize

to unseen but semantically valid variations of the data.

3.2. Data Augmentations as Empirical Support Extensions

We define a data augmentation as a set-valued mapping $V : \mathcal{X} \to \mathcal{X}$ that associates each sample x_i with a local neighborhood $V(x_i)$. We assume that a quality augmentation satisfies the *coherence property*: $V(x_i) \subseteq \mathcal{M}$, ensuring that augmented samples remain on the natural image manifold. This is important, as severe augmentations may push the outputted image outside of it. Augmentations that satisfy this property effectively expand the empirical support of the dataset by replacing each isolated sample with a local region on $\mathcal{M}$. This increases the sampling density of the empirical distribution and reduces gaps between observed regions of the manifold. From this perspective, augmentations improve the geometric coverage of $\mathcal{M}$ rather than modifying the underlying data-generating distribution. While the standard affine transformations used in SSL architectures explore a restricted subspace of the manifold, non-rigid transformations such as Thin Plate Splines (TPS) allow for localized deformations. We hypothesize that such transformations enable exploration of additional subspaces on $\mathcal{M}$ that are underrepresented by the set of canonical augmentations used in SSL architectures.

3.3. Coherent adversarial transformations increase empirical support

Our architecture combines a Spatial Transformer Network (STN) with a Gradient Reversal Layer (GRL) to learn adversarial transformation parameters:

$$\theta^* = \arg\max_{\theta \in \Theta} \mathcal{L}(f, T_\theta(x)).$$

Applying only a single adversarial transformation per sample yields a new empirical distribution which remains discrete and does not resolve empirical sparsity. In this case, probability masses are displaced on the manifold without increasing coverage. To address this, we

introduce a stochastic linear interpolation between the control points of the identity transformation and those of the learned transformation:

$$T_{\text{final}} = (1 - \alpha)I + \alpha T_{\theta^*}, \quad \alpha \sim \text{Uniform}(0, 1).$$

Sampling α expands each training sample into a continuous one-dimensional trajectory on $\mathcal{M}$, connecting the original sample to its strongest adversarial deformation. This construction exposes the encoder to a set of semantically valid variations (assuming T_θ is coherent), increasing local coverage and reducing gaps in the empirical support. Furthermore, as the TPS transformations are learned adversarially, they provide support to the least represented sub-spaces of $\mathbb{P}_r$, extending the empirical support of the dataset.

3.3.1. Validity of linearly interpolating Thin Plate Splines transformations

For fixed source control points and kernel, a TPS transformation is linear in the target control point coordinates. If we define the identity transformation by a set of source control points P and the adversarial transformation by target points P^*, any linear interpolation of the control points:

$$P_\alpha = (1 - \alpha)P + \alpha P^*$$

results in a valid TPS transformation that represents a smooth, intermediate state of the warp.

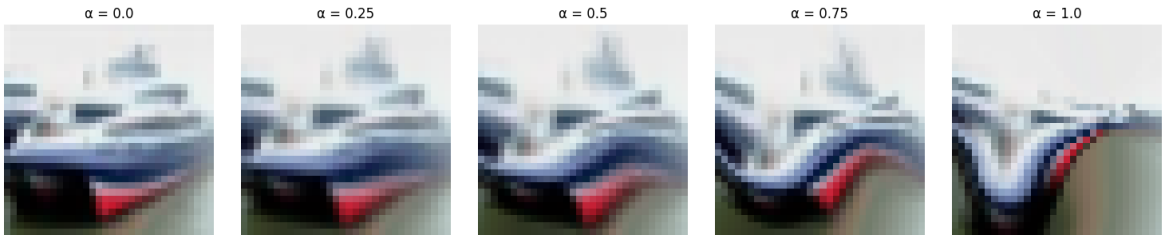


Figure 3.1: Linear interpolations of a TPS transformation for a CIFAR10 image

3.3.2. Unbiased linear interpolation

Using the previous formulation for the linear interpolation of control points introduces bias in the data perceived by the encoder. Given that α is sampled uniformly, the expected parameters become:

$$E[\theta_\alpha] = \frac{\theta_{id} + \theta}{2}$$

If the localization network f_{loc} produces a biased output θ, the "mean" image seen by the encoder during training is effectively shifted away from the original distribution. This is not the case for most stochastic transformations, such as rotations and translations, which are zero-centered. Since the encoder minimizes the loss over the expected distribution of α, it will fit to this "average distortion". When the learned transformation is consistently biased and not centered around the identity, we introduce a domain shift at test time, which actually decreases the model's accuracy on clean test data. To ensure that the extended empirical support remains centered around the true data manifold, we reformulate our interpolation as:

$$T_{\text{final}} = (1 - \alpha)I + \alpha T_{\theta^*}, \quad \alpha \sim \text{Uniform}(-1, 1).$$

Which has the benefits of being zero-centered: $E[\theta_\alpha] = \frac{\theta_{\alpha=-1} + \theta_{\alpha=1}}{2} = \theta_{id}$ and also doubles the span of the sampled trajectory on $\mathcal{M}$, further extending support.

Figure 3.2: Left to right: A learned transformation, the identity, and the learned transformation's inverse, corresponding to α values of 1, 0 and -1 respectively. Generally, compressed areas become expanded, and vice-versa.

3.4. Proposed Architecture

This section provides a breakdown of our architecture's forward pass from loss function to inputs. While we adopt most elements from FastSiam [11], their descriptions are included for completeness. For a visual summary of the architecture's structure, see Fig. 3.3.

Optimization Objective

The model is trained by minimizing a Negative Cosine Similarity (NCS) loss. For an image n used to generate V stochastic views $\{x_{n,1}, x_{n,2}, \ldots, x_{n,V}\}$, the loss $\mathcal{L}$ for a specific view i of image n is defined as:

$$\mathcal{L}_{n,i} = \mathcal{D}(p_n, \bar{z}_n)$$

Where $\mathcal{D}$ is the negative cosine similarity: $\mathcal{D}(p, z) = -\frac{p \cdot z}{\|p\|_2 \cdot \|z\|_2}$.

Branch Definitions and Asymmetry

The model consists of a backbone encoder f, a projection head g, and a prediction head h. The architecture is comprised of two branches:

1. **The Prediction Path** ($p_{n,i}$): Processes the adversarially transformed view through all three components:

   $$p_n = h(g(f(\tilde{x}_n)))$$

   where $\tilde{x}_n$ is the output of the STN.

2. **The Target Path** ($z_{n,j}$): Processes stochastically transformed views through the encoder and projection head only:

   $$z_n = g(f(x_{n,j}))$$

   The reference target for view i is the mean of the embeddings of the other $V - 1$ views:

   $$\bar{z}_n = \frac{1}{V-1} \sum_{j \neq i} z_{n,j}$$

The transformed input $\tilde{x}_n$ is the result of applying a TPS transformation to the view x_n using an STN:

$$\tilde{x}_n = \text{STN}_{\theta_\alpha}(x_n)$$

Where the parameters θ_α are determined by the localization network f_{loc} and a stochastic linear interpolation with the identity parameters θ_{id}:

$$\theta_\alpha = (1 - \alpha)\theta_{id} + \alpha\theta$$
$$\theta = f_{loc}(x_n)$$
$$\alpha \sim \text{Uniform}(0, 1)$$

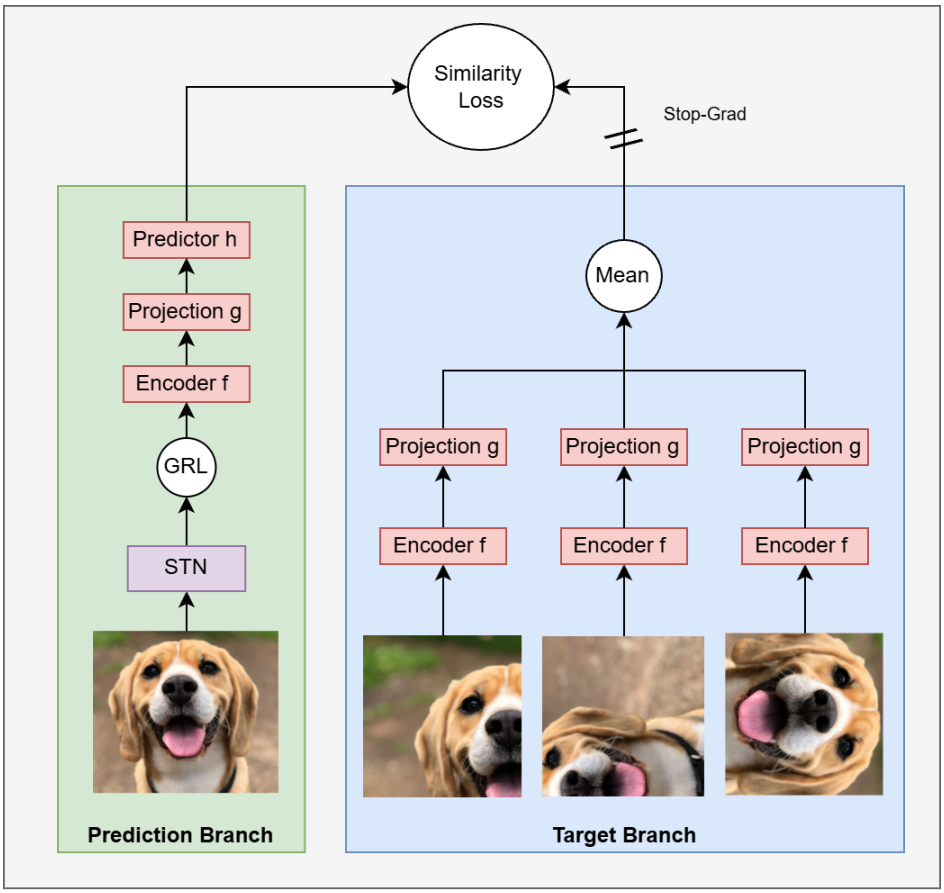


Figure 3.3: Summary of our architecture, which follows the structure of FastSiam but adding an adversarial learning component through the composition of an Spatial Transformer Network and a Gradient Reversal Layer

Chapter 4

Experimental Setup

This section describes the experimental setup used to evaluate the proposed architecture, including the datasets, evaluation metrics and technical implementation details.

4.1. Datasets and Metrics

We conduct our primary experiments on the CIFAR-10 dataset, which consists of 60,000 32×32 color images distributed across 10 balanced classes. We report test-set accuracy of a kNN model trained on the fixed CIFAR10 train set of 50,000 images and evaluate on the standard test set of 10,000 images with $k = 200$. The dataset preprocessing, evaluation metrics and base FastSiam implementation are taken directly from the official Lightly SSL benchmarks [17].

We additionaly report results on Imagenette, a subset of 10 classes from ImageNet, that we downsample to (48,48) resolution. As Lightly does not provide a benchmark for this version of the dataset, we compare our architecture only with the default FastSiam implementation. We report test-set accuracy of a kNN model trained on the fixed Imagenette train set of 9469 images and evaluate on the standard test set of 3925 images with $k = 200$. We also train a linear classifier for 50 epochs and report the top-1 and top-5 classification accuracies, following standard linear probing methodology.

4.2. Implementation Details

We implement the model using PyTorch Lightning. The encoder f utilizes a ResNet-18 backbone modified for 32×32 (or 48×48) inputs by removing the initial max-pooling layer and reducing the stride of the first convolution, following standard CIFAR10 ResNet adaptations [15].

Projection Head (g): A 2-layer MLP [512, 512]. Both layers use Batch Normalization. ReLU activation is applied only after the first layer.

Prediction Head (h): A bottleneck 2-layer MLP [512, 128, 512]. The 128-dimensional hidden layer includes Batch Normalization and ReLU. The output layer has no activation function.

STN Architecture

The localization network f_{loc} is a convolutional neural network with the following structure:

- **Layer 1**: Convolutional layer with 8 filters (7×7 kernel), followed by 2×2 Max-Pooling and ReLU activation.
- **Layer 2**: Convolutional layer with 10 filters (5×5 kernel), followed by 2×2 Max-Pooling and ReLU activation.
- **Layer 3**: Fully connected layer mapping the flattened 160-dimensional feature vector to 32 hidden units with ReLU activation.
- **Output Layer**: A final linear layer regressing N parameters, representing the (x, y) coordinates for the learnable control points.

All convolutional and hidden linear weights are initialized to zero. The output layer bias is initialized to the set of control points that produce the identity transformation.

TPS Configuration

Control Points

We utilize a grid of $N = 16$ control points. Importantly, to maintain image boundary integrity, the control points located on the edges of the image remain fixed and unlearnable, while the inner control points are regressed by the localization network. As such, the output layer of the STN only needs to regress 8 of the 16 control points (see Fig. 4.1).

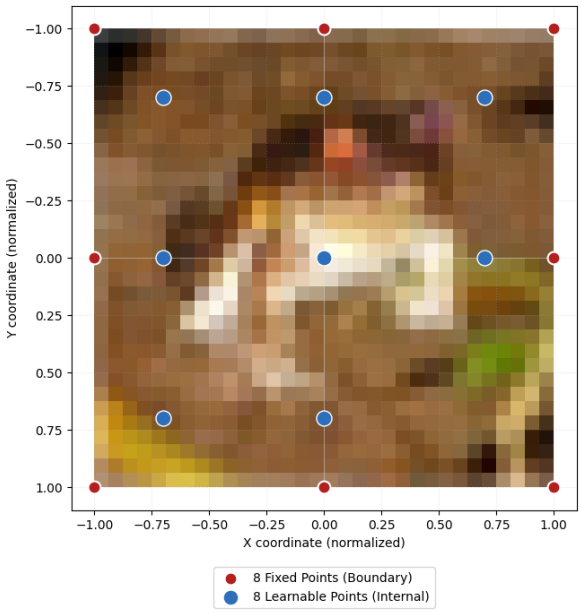


Figure 4.1: Initial configuration of TPS control points. Boundary points are fixed, whereas internal points are learnable.

Displacement Clamping

To prevent the STN from learning severely destructive warps that would break the coherence property, we introduce a parameter d_{max} to clamp the regressed displacements to a maximum, relative percentage-wise to the image dimensions (see Figure 4.2).

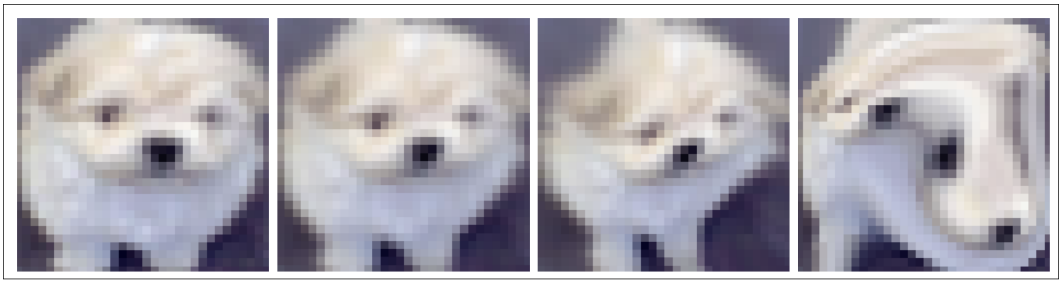


Figure 4.2: The severity of TPS transformations depends on the maximum allowed relative displacement, d_{max}. Images shown from left to right, for values [0, 0.05, 0.1, 0.25]

Optimization

Following FastSiam's training configuration, we use Stochastic Gradient Descent (SGD) with the following hyperparameters:

- Base learning rate: 0.06
- Momentum: 0.9
- Weight decay: 5×10^{-4}
- Learning rate schedule: Cosine annealing over the total number of epochs.

Chapter 5

Results

We compare our proposed model against the standard FastSiam architecture to isolate the effects adversarial transformations have on the quality of learned representations. We run our experiments on an NVIDIA T4 GPU with 16 GB GDDR6 and 4 vCPUs, with training for 200 epochs taking roughly 8 hours.

5.1. Performance Comparison

All results shown use the following hyperparameters unless specified:

- $\lambda = 1$
- $batch\_size = 128$
- $epochs = 200$,
- $d_{max} = 0.15$.

At the end of every epoch we embed all training images and use the features for a kNN classifier with k=200 on the test set. Evaluations are performed by embedding the training

image set and using the features for a kNN classifier with k = 200 on the test set. Figure 5.1 shows the following sets of results:

- **Default**: The default FastSiam implementation, with no STN or GRL components added.
- **Beneficial Transformations**: Using $\lambda = -1$ to essentially omit the GRL from the architecture, allowing the STN to learn beneficial transformations that minimize the loss.
- **Disabled random interpolation**: Here, a single adversarial transformation per image is learned and applied deterministically.
- **Biased random interpolation**: Here, we enable random interpolation with $\alpha \in [0, 1]$.
- **Our Proposed Architecture**: Our architecture as previously described, using unbiased random interpolation with $\alpha \in [-1, 1]$.

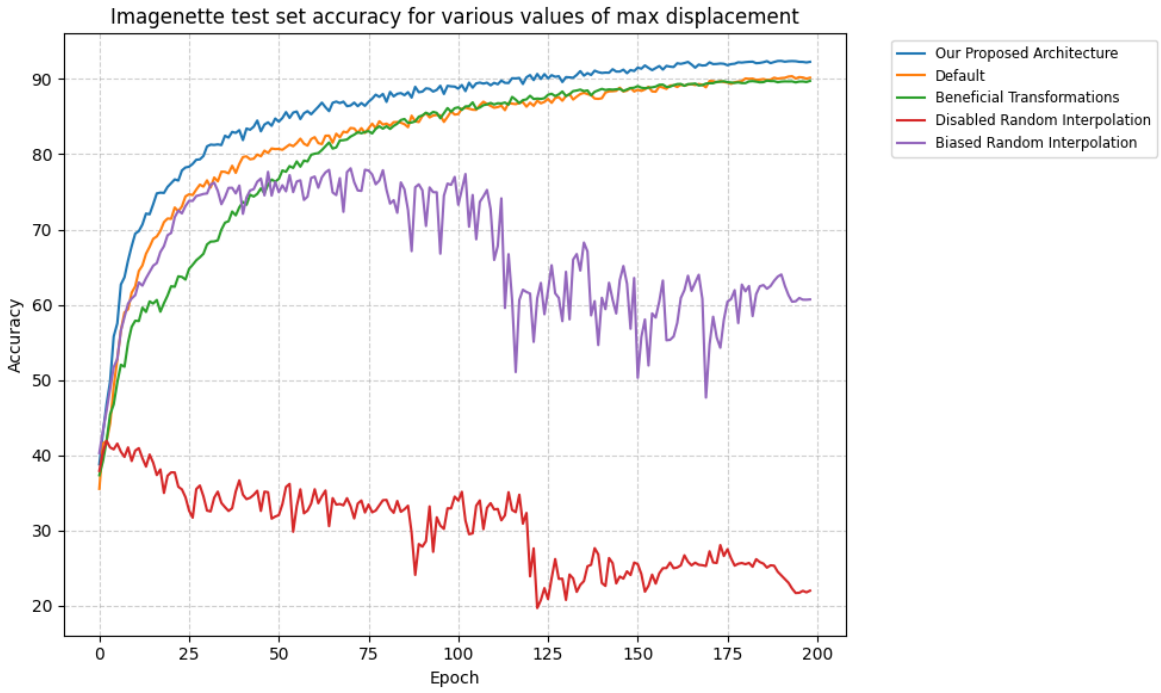


Figure 5.1: Test set training curves for the CIFAR10 dataset. Accuracy is calculated by training a kNN classifier with $k = 200$ on the encoder f's train-set embeddings, and applying it to the test set.

Analysis of Results

Disabling random interpolation leads the model's generalization capacity to worsen as training goes on. This can be understood geometrically, as the δ_{x_i} of each data point is displaced outside of the data-generating distribution's manifold, thus reducing empirical support (see Fig 5.2).

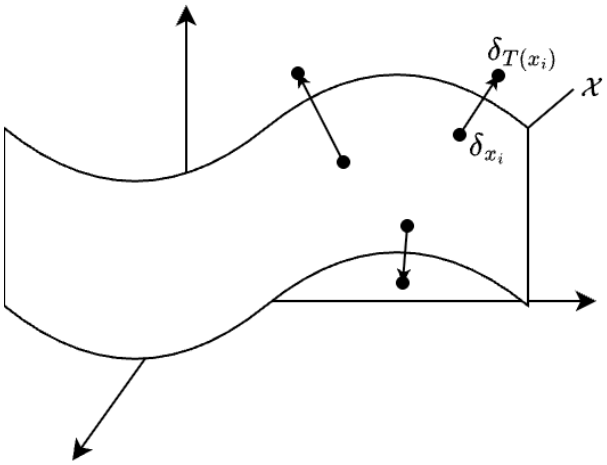


Figure 5.2: Single, deterministic adversarial transformations push the empirical data outside of the data-generating distribution, reducing empirical support.

Once we enable biased random interpolations, we see a stark improvement. However, despite the model's initially increasing accuracy, convergence leads the model to worse evaluations in the test set. This makes sense, as the encoder initially learns robust features through a near-identity STN, but later converges to a mean transformation which is unaligned with the clean images present in the test set, as discussed in section 3.3.2.

Beneficial transformations train similarly to the default FastSiam model, though presenting slower initial growth. This is due to the STN slightly facilitating the task of predicting similar vectors between branches. As empirical support does not significantly change due to near-identity learned transformations, the resulting generalization capacity of the encoder remains roughly equal to that of the default FastSiam architecture. Finally, our proposed architecture, using unbiased learned adversarial transformations, sets a new benchmark for SSL architectures on CIFAR10, with a 200-epoch maximum kNN accuracy of 92.40 against the current SOTA of 90.06 for FastSiam. Our architecture also reaches the maximum accuracy of FastSiam in roughly 40% less epochs (epoch 115 vs epoch 190). As the only architectural additions are a shallow convolutional regressor in the STN and a small linear system resolution from TPS, per-epoch training times remain roughly equal.

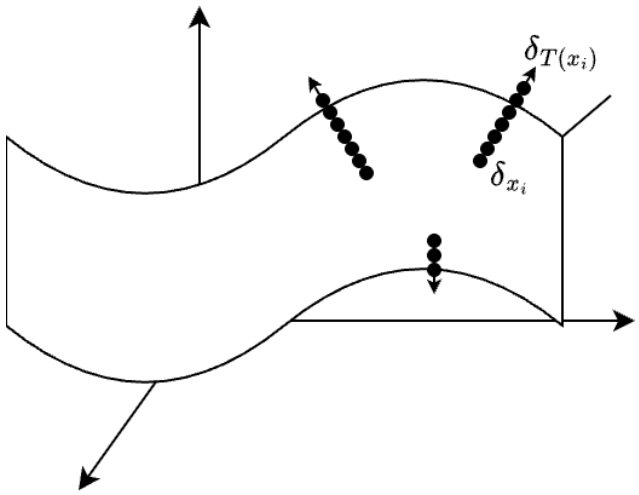


Figure 5.3: Stochastic, interpolated adversarial transformations extend empirical support towards the limits of the natural image manifold, increasing coverage. The ratio of coherent to non-coherent transformations is proportional to d_{max}.

We report our primary results and a comparison with the state of the art on CIFAR-10 using the LightlySSL benchmark 5.1. We complement these results with a one to one comparison of our architecture with FastSiam on a secondary dataset, Imagenette, which is a subset of ImageNet that we sampled at $(48, 48)$ pixels to validate the robustness of our solution 5.2, with the associated training curves 5.4.

Model	Backbone	Batch Size	Epochs	kNN Top 1
BarlowTwins	Res18	128	200	0.842
BYOL	Res18	128	200	0.869
DCL	Res18	128	200	0.844
DCLW	Res18	128	200	0.833
DINO	Res18	128	200	0.84
FastSiam	Res18	128	200	0.906
Moco	Res18	128	200	0.838
NNCLR	Res18	128	200	0.834
SimCLR	Res18	128	200	0.847
SimSiam	Res18	128	200	0.819
SwaV	Res18	128	200	0.812
SMoG	Res18	128	200	0.743
Ours	Res18	128	200	**0.924**

Table 5.1: Benchmark results provided by LightlySSL for CIFAR10. We use the same configurations and dataloaders to guarantee comparability.

Model	Backbone	Batch Size	Epochs	kNN Top 1	Lin. Top 1	Lin. Top 5
FastSiam	Res18	128	200	0.847	0.8443	0.9827
Ours	Res18	128	200	**0.871**	**0.8823**	**0.9860**

Table 5.2: Direct comparison between our architecture and FastSiam on Imagenette sampled at (48,48) resolution. Linear probing results were obtained after 50 epochs of training on the frozen backbone.

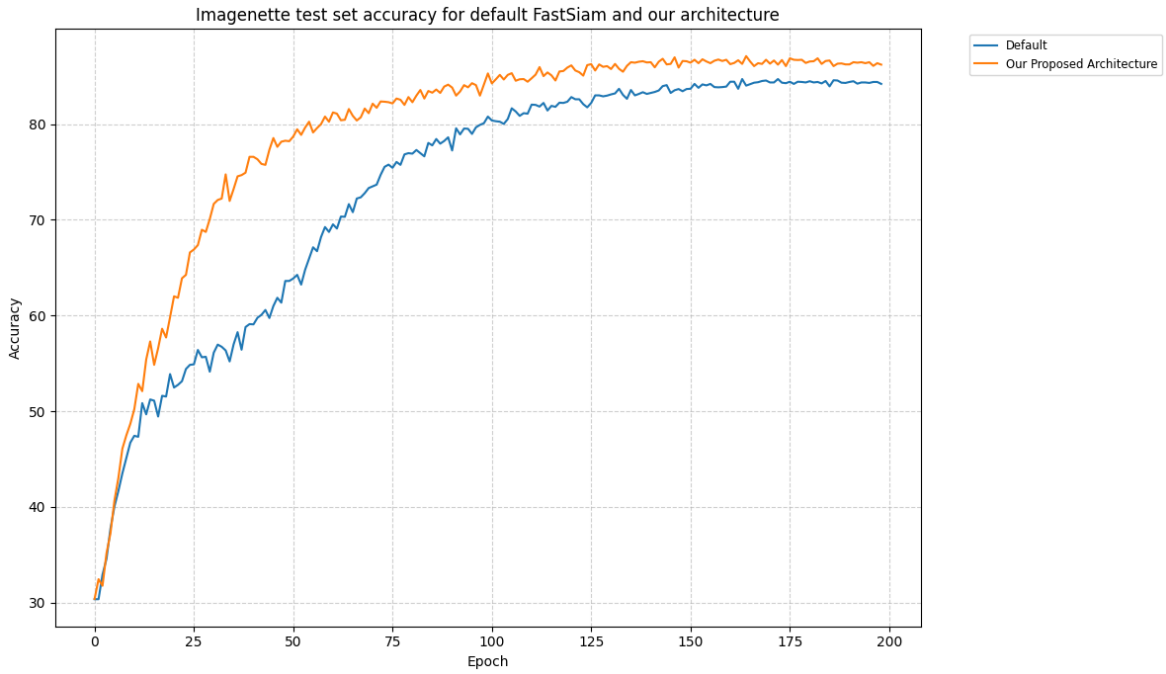


Figure 5.4: Test set training curves for the Imagenette dataset. Our architecture reaches the default FastSiam implementation's 200th epoch accuracy in roughly half the amount of epochs; consistent with performance on CIFAR10.

5.2. Qualitative Analysis of Learned Transformations

When using positive values for the GRL's λ parameter, learned transformations distort the input images as far as the parameter d_{max} allows (see Figures 5.5 and 5.6).

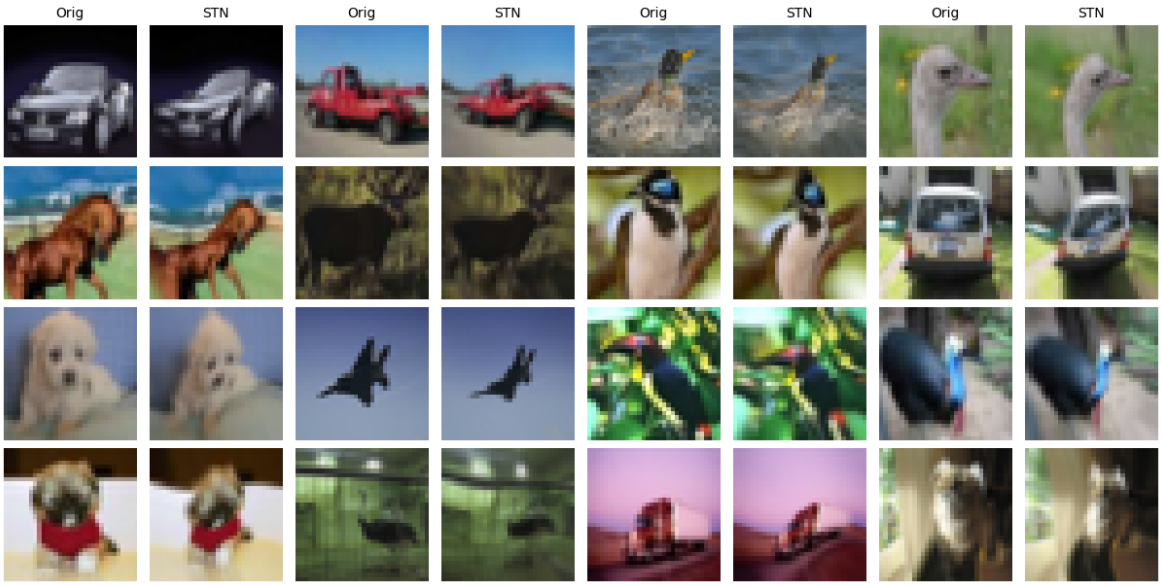


Figure 5.5: A set of transformations learned after 200 epochs, with $\lambda = 1$ and $d_{max} = 0.1$. The transformations maintain semantic information while hindering recognition.

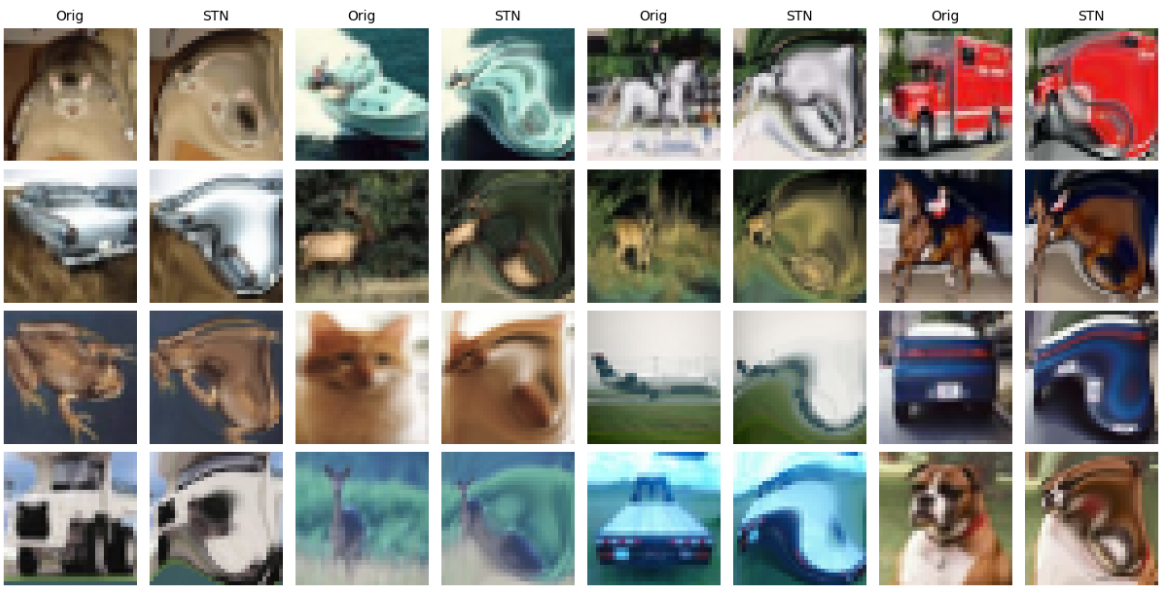


Figure 5.6: A set of transformations learned after 200 epochs, with $\lambda = 1$ and $d_{max} = 0.25$. The transformations become incoherent, destroying most semantic information.

We also try learning beneficial transformations using $\lambda = -1$. In this case, the STN does not learn any transformations, maintaining the initial identity configuration (see Fig. 5.7). This is expected, as the STN learns to output whichever image is closest to the average of all stochastic transformations applied in the target branch. As the stochastic transformations are zero-centered, this average image is the identity.

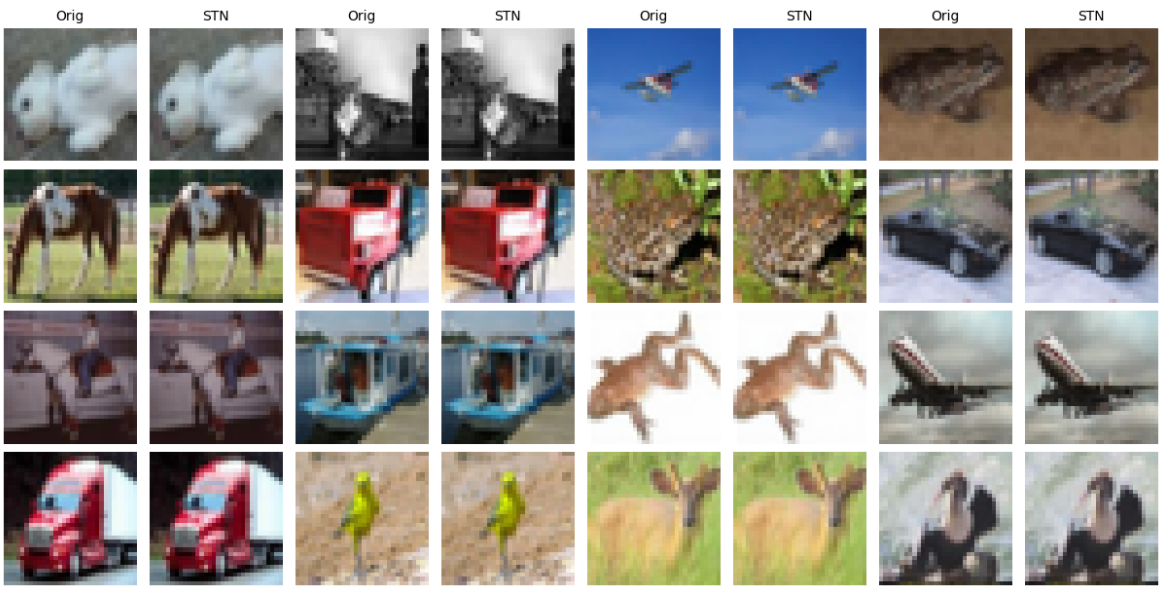


Figure 5.7: A set of transformations learned after 200 epochs, with $\lambda = -1$, and $d_{max} = 0.1$. The final learned transformation approximates the identity.

5.3. Sensitivity Analysis of the α parameter

While sampling α from $U(-1, 1)$ produces an unbiased set of input images, it doesn't guarantee that the non-linear encoder's outputs are also unbiased around $\alpha = 0$. To measure the drift in latent representations, we measure the response of the encoder's embeddings as a function of α.

For a set of N images, we establish the clean reference in latent space by passing each image through the encoder f and projection head g without applying spatial transformations ($\alpha = 0$), producing a set of embeddings z_{id}. We then sample a grid of M values of $\alpha \in [-1, 1]$ to obtain a set z_α.

For each corresponding (z_{id}, z_α) pair, we measure the Cosine Similarity and Euclidian distance, as well as the encoder's loss. We calculate the mean and variance over the batch of N images to measure the sensitivity of the model to the parameter α.

Figures 5.8, 5.9 and 5.10 shows the aforementioned metrics of our sensitivity analysis with $N = 100$ images, and $M = 1000$ alpha values.

We observe a very symmetrical response around the identity transformation ($\alpha = 0$), validating that our unbiased linear interpolation leads to unbiased learned representations. The mean cosine similarity remains consistently high (above 0.85) across the entire range of α, indicating that the backbone encoder has learned representations that are robust to relatively strong transformations.

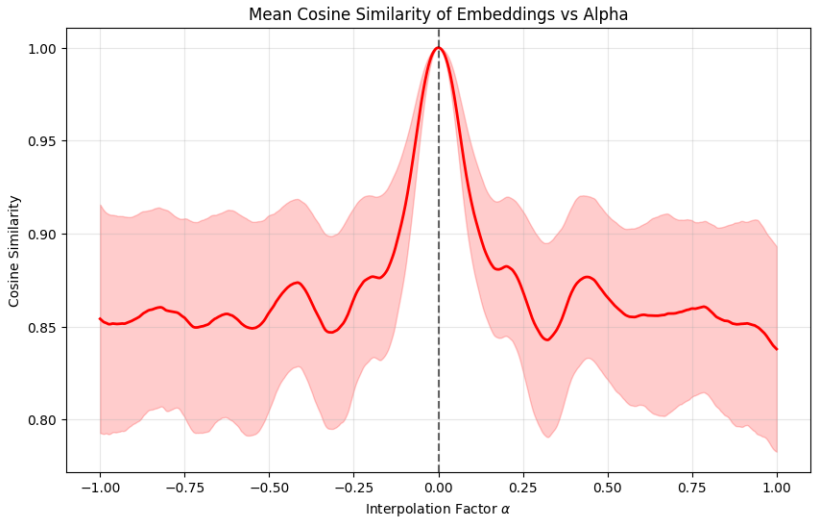


Figure 5.8: Cosine similarity between α-transformed embeddings and their corresponding base images. We note how similarity is roughly symmetrical around $\alpha = 0$: Adversarial, $\alpha > 0$ embeddings are roughly as distant from the base image as their inverse, $alpha < 0$, counterparts.

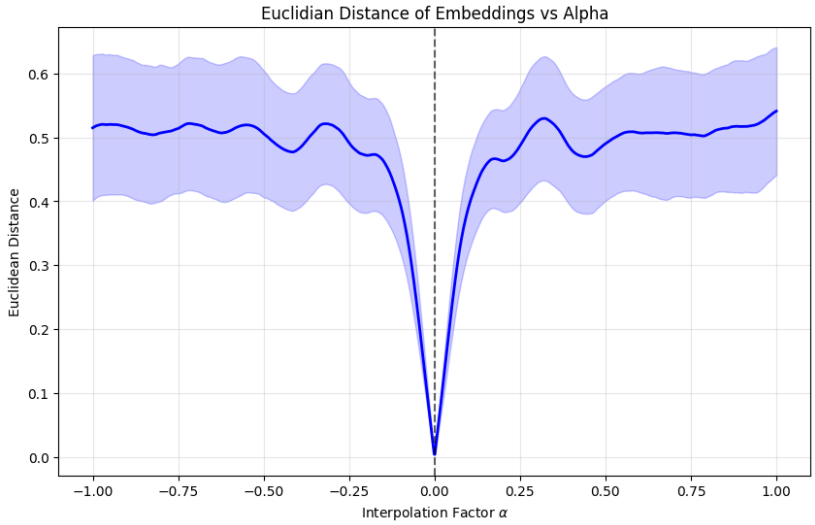


Figure 5.9: Euclidian distance between α-transformed embeddings and their corresponding base images. Distance between embeddings is proportional to the magnitude of α, with their being no particular bias towards negative or positive values.

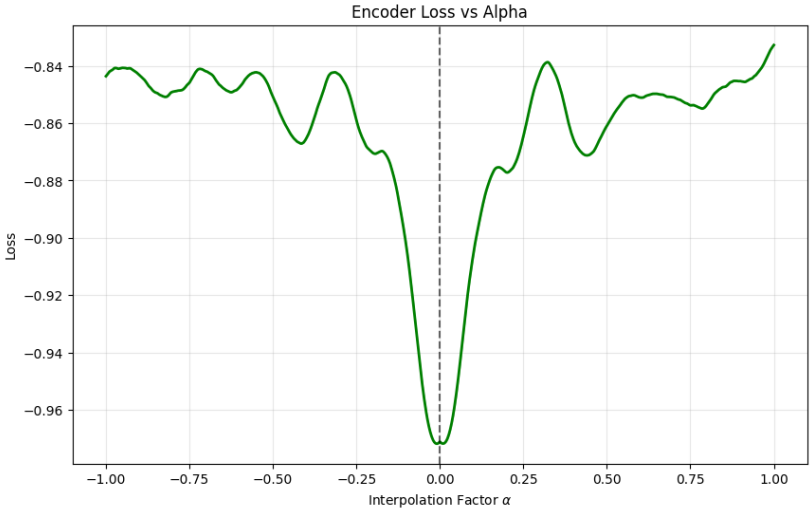


Figure 5.10: Encoder Loss between a batch of α-transformed embeddings and their corresponding base images. Loss is also symmetric around $\alpha = 0$ and roughly proportional to the magnitude of α, showing that stronger adversarial and inverse transformations effectively increase the encoder's loss.

5.4. Effects of d_{max} on model performance

We run our architecture on Imagenette for $d_{max} \in \{0.05, 0.15, 0.3, 0.5, 0.75, 1\}$ to see the effect this parameter has on model performance (see Table 5.3 and Figure 5.11).

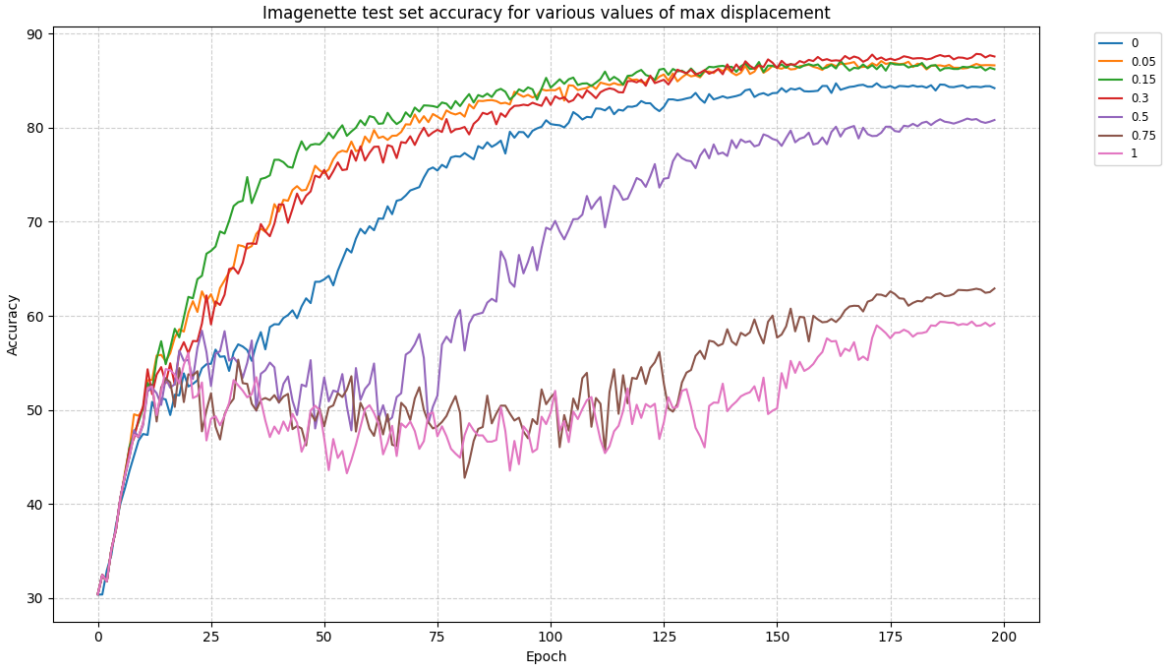


Figure 5.11: Training curves on Imagenette with varying values of d_{max}

d_{max}	**kNN Accuracy**
0	0.847
0.05	0.870
0.15	0.871
0.3	0.878
0.5	0.809
0.75	0.629
1	0.594

Table 5.3: Imagenette best-accuracy per d_{max} threshold. 0 behaves identically to the default FastSiam implementation, as no displacement is allowed. Small values produce a large improvement, with larger values eventually being counterproductive as image coherence is lost.

While larger values of d_{max} perform relatively well, this behavior is not transferable to CIFAR10, as values higher than 0.5 lead to a stark collapse in performance. The optimal value for d_{max} will likely depend on each dataset's feature granularity and image resolution.

Chapter 6

Conclusions

In this work, we presented a novel adaptation that learns adversarial transformations to improve the generalization capacity of SSL architectures. As the module is self-contained, it can be added to any Siamese, self-distillation or teacher-student SSL architecture to improve generalization capacity and convergence speed, with no increase in training times. Our results validate the hypothesis that adversarial transformations can be leveraged to improve a model's latent representation quality. We have also empirically demonstrated the relevance of using zero-centered, unbiased transformations for data augmentations, a fact that should be considered for future heuristic or automatic transformation design.

Chapter 7

Future Work

We propose a few directions in which our proposed architecture may be driven to further improve performance.

Gaussian Sampling over unbounded transformations

We require the hyperparameter d_{max} to bind the range of transformations into a coherent set. We may remove this parameter by allowing arbitrarily strong transformations and using $\alpha \in N(0, 1)$. This would also be an unbiased transformation set, but with the additional property of having transformation-severity be inversely proportional to it's probability, which should be congruent with the data-generating distribution.

Stabilizing the prediction branch's output

FastSiam saw tremendous improvement over SimSiam by averaging the predicted vectors of each set of stochastic transformations. The same procedure could be applied to our architecture's prediction branch to stabilize it's output to the mean of the learned transformation's linear interpolation. This may provide a more stable learning signal to the encoder, accelerating convergence.

Learning input-specific transformations

After training, the STN's outputs are roughly constant with respect to it's inputs, presenting near zero variance in the regressed output matrix. The STN seems to converge to a local optimum of learning a constant transformation that maximizes the average loss over each image batch. This may be improvable by working with more TPS control points and a deeper STN architecture.

Bibliography

[1] Mathilde Caron, Ishan Misra, Julien Mairal, Piotr Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[2] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *International Conference on Computer Vision (ICCV)*, 2021.

[3] Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik. Vicinal risk minimization. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 200.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E Hinton. A simple framework for contrastive learning of visual representations. *International Conference on Machine Learning (ICML)*, pages 21–31, 2020.

[5] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[6] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space, 2019.

[7] Lijie Fan, Sijia Liu, Pin-Yu Chen, Gaoyuan Zhang, and Chuang Gan. When does contrastive learning preserve adversarial robustness from pretraining to finetuning? *CoRR*, abs/2111.01124, 2021.

[8] Yaroslav Ganin, Evgeniya Ustinova, Henri Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[9] Jean-Bastien Grill, Florent Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo A Pires, Zaidian D Guo, Mohammad G Azar, Bill Piot, Koray Kavukcuoglu, Rémi Munos, and Mikkel Valko. Bootstrap your own latent: A new approach to self-supervised learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[10] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[11] Jintao Hou, Zihang Zhang, Yongxin Zhao, Yanqing Huang, Yujun Wu, Yuxin Wu, Lin Yang, and Xiaolin Wei. Fastsiam: Fast, strong, and flexible architecture search for siamese representation learning. *arXiv preprint arXiv:2205.14815*, 2022.

[12] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[13] Minseon Kim, Jihoon Tack, and Sung Ju Hwang. Adversarial self-supervised contrastive learning. *CoRR*, abs/2006.07589, 2020.

[14] Lujun Li and Anggeng Li. A2-aug: Adaptive automated data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 2267–2274, June 2023.

[15] Lightly AI. Benchmarks — LightlySSL 1.5.22 documentation. `https://docs.lightly.ai/self-supervised-learning/getting_started/benchmarks.html`, 2024. Accessed: 2024-05-20.

[16] Warren Morningstar, Alex Bijamov, Chris Duvarney, Luke Friedman, Neha Kalibhat, Luyang Liu, Philip Mansfield, Renan Rojas-Gomez, Karan Singhal, Bradley Green, and Sushant Prakash. Augmentations vs algorithms: What works in self-supervised learning, 2024.

[17] Philipp Wirth and Igor Susmelj. Lightlyssl: A python library for self-supervised learning. `https://github.com/lightly-ai/lightly`, 2020.

[18] Liang Xiao, Jiaolong Xu, Dawei Zhao, Erke Shang, Qi Zhu, and Bin Dai. Adversarial and random transformations for robust domain adaptation and generalization, 2022.

[19] Qiying Yu, Jieming Lou, Xianyuan Zhan, Qizhang Li, Wangmeng Zuo, Yang Liu, and Jingjing Liu. Adversarial contrastive learning via asymmetric infonce, 2022.

[20] Jure Zbontar, Li Muhamed, Igor Gilitschenski, Shean Nunkesser, Sreyas Roy, Andrew Zisserman, and Andrei A Rusu. Barlow twins: Self-supervised learning via redundancy reduction. *International Conference on Machine Learning (ICML)*, 2021.