EVOLVING OPTIMAL AUGMENTATION POLICIES FOR SELF-SUPERVISED LEARNING ALGORITHMS

by

Noah Barrett

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
June 2023

This thesis is dedicated to my mother, Christine Connolly.

Table of Contents

List of Tables

List of Figures

Abstract

In recent years, self-supervised learning has shown remarkable promise for expanding the capabilities of deep-learning-based computer vision models. In many self-supervised learning approaches, specifically those that employ a Siamese Network, data augmentation is a core component of the algorithm. However, typically a standard set of augmentations are employed without further investigation into improving the augmentation strategy used. This thesis aims to address this issue by taking a step forward to better understand the impact of data augmentation on cutting-edge computer vision based self-supervised learning algorithms. Inspired by supervised augmentation optimization approaches, this thesis explores the possibility of further optimizing four SOTA self-supervised learning algorithms, BYOL, SwAV, NNCLR, and SimSiam, by improving augmentation operators used in the pretext task. Using a Genetic Algorithm, it was possible to learn augmentation policies which yielded higher performance than the original augmentation policies for all four self-supervised learning algorithms, on two datasets, SVHN and CIFAR-10. This thesis shows that improving the augmentation policies used in computer vision based self-supervised learning algorithms is a fruitful direction for further improving on the cutting-edge performance yielded from this family of algorithms.

Acknowledgements

First, I would like to acknowledge the support, supervision, and guidance provided by Dr. Stan Matwin; I am forever grateful for the privilege of having his supervision. I also would like to acknowledge the guidance provided by Dr. Zahra Sadeghi for this work; the numerous discussions, collaborations, and critical feedback have greatly benefitted my development as a student and researcher. I also would like to acknowledge Dr. Michael McAllister for guiding me during a turbulent period of my studies and personal life. Lastly, I would like to acknowledge all of my lab mates for their support and friendship throughout my degree.

Chapter 1

Introduction

Vision is one of the vessels in which biological life understands the world around it; developing artificial intelligence techniques to leverage this powerful information stream via images and videos *i.e.* Computer Vision, has proven extremely challenging but astoundingly fruitful. Deep Learning has succeeded incredibly in complex and prevalent computer vision problems, such as image classification, object detection, visual tracking, semantic segmentation and image restoration [10]. However, much of this success hinges on the availability of massive, annotated, datasets such as ImageNet [18], which contains over 14 million labelled images. In many real-world problems, obtaining labelled datasets of this scale is extremely difficult or unattainable. An approach known as Self-Supervised Learning (SSL) has shown remarkable promise in expanding the capabilities of deep-learning-based computer vision models by alleviating the need for large labelled datasets.

The Siamese Network architecture [46] is a deep learning architecture which is core to many of the successful SSL algorithms for computer vision. For SSL algorithms which employ a siamese network, data augmentation, a method which uses one or more techniques to enhance the size and quality of training datasets [72], is central to the learning mechanism. Arguably, the most influential use of the siamese network for SSL, SimCLR, was proposed by Chen *et al.* [12]. In this work, they introduce the concept of employing data augmentation for forming the prediction task; specifically, SimCLR learns by maximizing the agreement between two different augmentations of the same image [12]. A core finding in the work presented by Chen *et al.* is that the composition of augmentations is crucial to the success of SimCLR. They found a particular combination of augmentations which proved to be most effective. Given the impressive results and theory behind SimCLR, this combination of augmentations has been used in many cutting-edge SSL techniques [79]. However, recently Cosentino *et al.* [15] has shown that the choice of augmentation operators is an important component which should be considered on a case to case basis. Therefore using only the carefully selected augmentations initially found by chen *et al.* could produce sub-optimal results. Inspired by the success of SSL for CV and the possibility of further improvement via the selection of augmentation operators, this work investigates if it is possible to systematically improve the choice of augmentation operators used for four state-of-the-art algorithms: SwAV [9], BYOL [1], NNCLR [22], and SimSiam [14].

The systematic selection of augmentation operators for deep learning algorithms is a research area known as augmentation optimization. AutoAugment [16] was the first augmentation optimization method, which applied reinforcement learning to optimize the augmentations used for a supervised classification task. Many other researchers built on the foundations provided in AutoAugment, such as improving the speed of the approach [55], [32], employing a population-based method [37], and leveraging a random search algorithm [17]. More recently, augmentation optimization has been employed for graph-neural-network-based SSL [82] [43], [81]. For augmentation optimization with image-based SSL methods, approaches such as manually choosing various combinations of augmentations [79], or employing non-population-based augmentation optimization algorithms [69] have been investigated. Although the current works investigating the selection of augmentation operators in image-based SSL show promising results, there is a lack of thorough evaluation of the many possible augmentation combinations.

This work proposes a novel framework for applying a Genetic Algorithm (GA) to the problem of augmentation optimization in the self-supervised computer vision domain. Therefore, the primary goal of this research is to understand if it is possible to use a GA to optimize the augmentation operators in image-based SSL methods, specifically those employing a siamese network. A core motivation behind employing a GA is that it is a population-based method, and although it does produce higher computational overhead, a large number of evaluations of augmentation combinations are carried out. A secondary goal of this research is to deeply investigate the many different augmentation combinations produced by the GA to understand if relationships exist between specific augmentations and the outcome of the respective SSL algorithm. This research investigates whether it is possible to improve the SSL algorithms SwAV [9], BYOL [1], NNCLR [22], and SimSiam [14] by using a GA to optimize the choice of augmentation operators, and additionally if it is possible to find any relationships between the specific augmentations and algorithms being used.

Previous work in augmentation optimization for SSL for image-based approaches is very limited, this work contributes the first evolutionary approach for augmentation optimization in the SSL computer vision domain. The investigation of augmentation optimization carried out by Reed *et al.* [69] and Yang *et al.* [81] apply a non-population-based approach, which is less interpretable than the proposed work. This work also contributes two metrics for measuring the impact that the augmentations have on the training process and employing them to reveal how specific augmentation operators affect the SSL training process. Given the outcomes of the GA and metrics to interpret the results, this work produces an analysis on the impact of augmentation optimization on the SSL algorithms of interest.

This thesis touches on many different sub-fields of machine learning, including self-supervised learning, evolutionary computation, and hyper-parameter optimization. Additionally, several works in augmentation optimization inspire and guide the work in this thesis. These topics are covered in Chapter 2. Combining all the mentioned sub-fields of machine learning, the proposed GA requires a detailed explanation of its components, including the genetic operators and other evolutionary components, the self-supervised learning algorithms and baselines, and the augmentations used in said algorithms in baselines. These components are discussed in detail in chapter 3. Many discoveries related to the impact of augmentation policies on the four self-supervised learning algorithms were made in the experiments carried out with the proposed genetic algorithm. Overall, it is clear that the SSL algorithms can be further optimized by changing only the augmentation policies. This finding is shown in chapter 4, where the results of the experimentation as well as in-depth analysis can be found. Lastly, many challenges exist in this work, which opens up several avenues of future works, this is discussed in Chapter 5.

Chapter 2

Background and Related Work

2.1 Deep Learning and Self-Supervised Learning

Deep Learning (DL) is a massively popular area of machine learning and is a central component of the proposed work. DL algorithms are based on the theory of Artificial Neural Networks (ANN) [80]. Through the use of multiple processing layers and back-propagation, DL discovers intricate structure within large datasets [53]. In many real-world applications, many factors of variation influence data; DL can solve this issue by learning representations expressed in terms of simpler representations [7]. Convolutional Neural Networks (CNNs) are a commonplace DL architecture inspired by the biological visual perception mechanism present in living creatures [28]. CNNs have been at the core of many significant breakthroughs in machine learning; some prevalent models include AlexNet [49], VGGNet [73], and ResNet [35]. In most SSL algorithms, CNNs are the backbone for learning representations [2]. This work employs a relatively small and simple CNN for the investigated SSL algorithms.

CNNs are commonly used in image-based SSL algorithms. Specifically, they are used when building a siamese neural network. A siamese neural network consists of twin neural networks which accept two different inputs and are joined by an energy function at the top of the neural networks. The parameters of the two networks are tied to each other; this ensures that two very similar images cannot be mapped to very different locations in the feature space [46]. Siamese networks maximize the similarity between different augmentations of the same image. They have become a common architecture in various unsupervised visual representation learning algorithms, including many SSL algorithms [14]. The four algorithms which are investigated in this work, SwAV [9], BYOL [1], NNCLR [22], and SimSiam [14] all employ a siamese network or siamese network variant.

The history of DL began in 1943 when it was shown that neurons can be connected together to build a Turing machine [3]. In 2012, Krizhevsky *et al.* [50] trained a deep convolutional neural network, AlexNet, for the ImageNet ILSVRC-2012 contest, which performed astoundingly well compared to the other proposed ML methods. Following this early monumental finding in DL, great success has been found in applying DL to supervised, unsupervised, semi-supervised and reinforcement learning problems. The initial DL breakthroughs were partly due to the massive amount of data curated and labelled to train large deep neural networks. For example, the above-mentioned famous computer vision data set, ImageNet, [18] contains over 14 million images. However, in most real-world settings, a large amount of unlabelled data is very costly to annotate. SSL tackles this issue by first learning implicit information from the data without the need for human annotation which allows for massive unlabeled datasets to be leveraged.

By learning structure within the data without human annotation, SSL can be thought of as an unsupervised learning method [59]. Unsupervised learning refers to all algorithms which employ unlabelled data to derive implicit information within the dataset, SSL achieves this with representation learning. There has been a growing interest in unsupervised learning for learning feature representations, SSL accounts for a large body of this work, and there also exists a large variety of generative methods which aim to solve this problem [2]. Generative methods aim to learn a function that encodes an input to a reduced vector space and a decoder that reconstructs the reduced vector back into the input space [56]. One type of generative method is Auto-regressive models such as Pixel-CNN [78], and WaveNet [64]; these approaches model images pixel-by-pixel and have the advantage of modelling context dependencies within images [56]. Another famous generative method is the variational auto-encoder [45], an approach which maps the input x to a reduced representation space and subsequently reconstructs the input x', the goal of the model is to minimize the reconstruction error between x and x'. The above-mentioned generative methods are widely known approaches which are similar to SSL in that they aim to leverage implicit information within the dataset to learn a representation in an unsupervised manner.

Semi-supervised learning is a learning method which can be thought of as in between supervised and unsupervised learning. Semi-supervised learning has components similar to self-supervised learning. Semi-supervised methods typically utilize large unlabeled data points combined with a small number of labelled data points [11]. Like SSL, semi-supervised learning can leverage unlabelled data. However, how this is accomplished is fundamentally different from SSL. Our method employs SSL to learn information about a dataset without labels, then train on the downstream task, a supervised classification task. Similarly, in semi-supervised learning, the labelled and unlabelled data are drawn from the same distribution [85].

The training process of first pretraining a model using SSL then training on a downstream task is a form of transfer learning. Transfer Learning (TL) aims to improve the performance of target learners on target domains by transferring knowledge from different, but related source domains [85]. Theoretical and empirical evidence exists that TL provides performance improvements and increases generalizability to target tasks [85]. However, a clear drawback to this approach is that it requires large labelled datasets for pretraining the models on the source domain, and these labels may not be accurate in the target task [2]. SSL can be considered a form of transfer learning, as labels are automatically generated and trained within the pretext task, and the resultant representation is subsequently used for training in the downstream task [59].

The first SSL methods relied on auxiliary tasks to learn representations, these methods work by using labels created automatically based on attributes present in the dataset [2]. More recently, contrastive learning has been found to be exceptionally useful in SSL; this method works by comparing similar inputs, known as positive pairs, with dissimilar pairs, known as negative pairs [52]. SSL has two main phases, first solving the pretext task and, subsequently, the downstream task. In the pretext task, a neural network is trained on a task that borrows implicit or generated labels from the data. The representation learned in the pretext task is then trained on the downstream task; in computer vision, some common downstream tasks are image classification, object detection and segmentation [41].

For SSL using auxiliary tasks, the model learns by obtaining supervision signals in the data through a specified task, *i.e.* an auxiliary task [2]. Auxiliary tasks can be designed for any data type, such as audio, text, image, and video [41]. This work focuses on image-based self-supervised learning; several interesting auxiliary tasks exist for this domain. Noroozi *(*et al.) [63] proposed a popular technique in which they train a CNN to solve jigsaw puzzles that are generated from unlabelled image data points; this auxiliary task was found to produce learned features which capture semantically relevant information. Zhang *et al.* [84] propose an auxiliary task where a CNN learns to colourize greyscale images; this method utilizes unlabelled image data and generates labels by converting RGB images to greyscale and employs the RGB images as ground truth for colourizing the image data. Gidaris *et al.* [25] proposes a rotation auxiliary task where a CNN is trained to predict how much an image has been rotated; in this case, the images are rotated, and the labels are derived from the ground truth non-rotated images. Many auxiliary tasks exist for self-supervised learning, and the above three only serve as examples of what has been proposed.

On the other hand, contrastive learning aims to embed augmented versions of the same sample (positive samples) close to each other while trying to push away embeddings from different samples (negative samples) [2]. Instead of deriving labels from an auxiliary task in an SSL setting, contrastive learning methods learn a discriminative model according to some notion of similarity using the unlabelled dataset [52]. Memory banks are common in some of the earlier contrastive SSL methods. It has been found that large negative sample sizes are required, leading to a significant increase in mini-batch sizes. To address this, memory banks are proposed to store the feature representations of large negative samples without increasing the mini-batch size [2].

Memory banks were a prevelant feature in earlier SSL approaches. For example the popular SSL algorithm PIRL [61] employs a memory bank and adopts the jigsaw puzzle pretext task. The goal of PIRL is to encourage the image representations to be invariant to the configuration of the generated jigsaw puzzle. Additionally, He *et al.* [34] propose Momentum Contrast (MoCo); similar to PIRL, MoCo utilizes a memory bank, but also employs a momentum encoder. MoCo formulates contrastive learning as a dictionary look-up, hinged on the idea that a dictionary covering many negative samples can be used to learn good features. Opting for a more simple algorithm than MoCo and PIRL, the framework presented by Chen *et al.* [12] known as SimCLR presented a method for contrastive learning which did not require any specialized architectures or memory banks. Chen *et al.* found that signicantly increasing the number of epochs and batch size greatly improved the performance of the algorithm. After the release of SimCLR, Moco-V2 [13] showed that with simple modifications to MoCo, using an MLP projection head and more data augmentation, considerable improvements can be made, outperforming both SimCLR and MoCo.

This work focuses on four SSL algorithms that belong to the family of contrastive algorithms or are related to contrastive learning, SwAV [9], BYOL [1], NNCLR [22], and SimSiam [14]. The choice of these four algorithms is two-fold, first they are superior to previous approaches such as SimCLR, MoCo or PIRL and secondly they all share many key components of interest. In most cases, they opt for simplicity and reduce the need for complex additions such as memory banks or momentum encoders. All four algorithms share fundamental similarities in their architecture and underlying methodology. Specifically, all algorithms employ a Siamese architecture at their core. A common issue present in many algorithms employing Siamese networks is solution collapse. Collapse occurs when the model maps every image to the same point, maximizing similarity when comparing but not learning critical information about the problem. This issue is acknowledged however tends to not be an issue in practice. All four algorithms employ similar augmentation strategies which are influenced by Chen *et al.* and the augmentations used for SimCLR. The choice of augmentations used for these approaches is not heavily investigated and therefore the proposed work aims to look deeper into the choice of augmentations for BYOL, SimSiam, NNCLR, and SwAV to understand better how they affect the performance of the different algorithms.

The key idea of BYOL [1] is that from a given representation, called the target representation; it is possible to train an enhanced representation, the online representation. BYOL works iteratively, improving the learned representation by using a slow-moving average of the online network as the target network [1]. BYOL does not require negative samples and claims to be insensitive to batch sizes and the types of augmentations used. In the proposed methodology of BYOL, collapse is not directly prevented. However, it was found that the solutions do not tend to converge to a collapsed state. BYOL's ability to not require large batch sizes and negative samples, as well as having a novel level of robustness to augmentations, made it an outstanding algorithm at its release. It was shown that BYOL suffers a much smaller performance drop than SimCLR when only using random crops. BYOL does not explicitly prevent collapse; however, this issue is alleviated using a stop-gradient, a core component also present in SimSiam [14].

SimSiam is an SSL method that opts for simplicity and highlights the significance of the siamese network architecture for SSL methods. Chen *et al.* [14] show that the shared weight configuration is vital to many core SSL algorithms, and a simple stop gradient approach can be used to prevent collapse. The outcomes of the experiments for SimSiam provided surprising evidence that meaningful representations can be learned without using negative samples, large batches, and momentum encoders. The first two were already unnecessary in the earlier work shown in BYOL. However, the momentum encoder component (slow-moving average) was crucial to avoiding collapsed solutions. The most remarkable discovery produced by the authors of SimSiam could be that a simple stop gradient is all that is needed needed to train a siamese architecture successfully.

Caron *et al.* [9] proposed a novel mechanism for self-supervised learning which differs greatly from BYOL and SimSiam. BYOL and SimSiam learn by predicting the "closeness" of two views of a sample; this is a core concept to siamese networks and their impactful progress in representation learning at large. whereas SwAV learns by computing a code from an augmented version of an image and then predicting this code from other augmented versions of the same image [9]. By employing a siamese architecture, SwAV, at its core, still aims to learn a robust latent representation of the problem. Using this alternate approach yields a breakthrough unfound by BYOL and SimSiam, the ability to significantly reduce the number of pairwise feature comparisons, reducing the computational requirements of the algorithm.

Compared to the other discussed SSL methods, Dwibedi *et al.* [22] introduced a novel concept concerning the selection of positive pairs. Rather than deriving positive samples for an image via augmentation, they show that using the image of the interest's nearest neighbours in the given data space can be used as positive samples. Similar to SimSiam, NNCLR employs a straightforward architecture. The novelty is in selecting positive samples from the given dataset rather than derivation from augmentation. NNCLR samples positive samples by computing the nearest neighbours of a given sample in the learned latent space of the dataset. This approach provides more semantic class-wise variations rather than pre-defined transformations, which tend to provide more geometric information [22].

All four algorithms share many similarities yet have differences which make the comparisons of the effect of augmentation optimization interesting. SimSiam can be thought of as the simplest version of these learning approaches using a siamese architecture. BYOL adds a slight bit of complexity to the learning method by employing a slow-moving average approach for the target component of the Siamese network. Like SimSiam, NNCLR also opts for simplicity in its architecture. However, it refrains from employing only augmentation to generate positive samples and instead determine the nearest neighbours of a given image to select positive samples. SwAV differs from all three methods because it utilizes a swapped prediction mechanism for learning. All four algorithms have recently shown remarkable breakthroughs in the self-supervised computer vision world, so they have been chosen as focal points of the proposed work. With the exception of BYOL and its novel multi-cropping augmentation, the authors of the four algorithms do not deeply explore the choice of augmentation operators and default to those which tend to be used by other SSL algorithms. Given the limited discussion on the effect of data augmentation, the question of its impact on the different SSL algorithms remains.

2.2 Genetic Algorithms

Genetic Algorithms (GA) are population-based optimization algorithms inspired by natural selection. One of the basic building blocks of a GA is the representation of a solution, this is known as a chromosome and is encoded as a genotype. Different encodings exist for building a GA, such as binary, permutation, value, or tree encoding

[44]. The genotype must be decoded into its phenotype and evaluated using a defined function, known as the fitness function. Given a fitness function and chromosome representation, different biological operators can be applied to learn an optimal configuration. Binary encodings are simple and easy to implement but are limited to problems encoded as a binary string, permutation is limited to task ordering problems, and tree encodings are challenging to design and are used primarily for evolving programs [44]. For problems involving neural networks, a value-encoding is commonly used. It has the double-edged sword of requiring to define custom crossover and mutation methods, which allows for a higher degree of flexibility but requires more thought for implementation [44]. In value-encodings, the gene or chromosome is represented by a string of values; these values can be integers, real numbers, characters or objects defined by the algorithm designer [51], [68]. For this work, a value encoding was employed; due to the flexibility of the value encoding for representing the augmentation policies, the details on the value-encoding can be found in the methodology section.

GAs must first be initialized with a random population; the main objective of population initialization is to produce a diverse population which covers the search space as uniformly as possible [60]. A GA consists of three primary biological operators, Selection, Crossover and Mutation [67]. Selection is the process of selecting individuals based on their relative fitness. Crossover takes two or more parent solutions and randomly combines them to produce one or more children. Mutation randomly alters a solution's contents, allowing for new solutions to be discovered.

When performing selection, individuals are probabilistically selected based on their fitness; this allows the more fit individuals to have more children than the less fit individuals [67]. There are many different well-researched methods of selection for GAs. The most common selection methods are rank, tournament, Boltzmann, stochastic universal sampling and roulette wheel [44]. Tournament selection works by selecting k individuals and ranking them according to their relative fitness; this is repeated n times for the entire population [36]. Stochastic Universal Sampling (SUS) is a Roulette wheel selection variant that aims to reduce the issue of premature convergence [42]. Boltzmann Selection scales individuals' fitness within a population according to the Boltzmann distribution [6]. Roulette wheel selection works by selecting individuals with probabilities proportionate to their fitness; it has been found that premature convergence can be an issue with this approach [5]. Roulette wheel selection is shown in equation 2.1. Roulette selection is simple to implement and depends upon variance in the fitness function [44]. For these reasons, roulette wheel selection is used in this

$$p(i) = \frac{f(i)}{\sum_{n}^{j-1} f(j)} \tag{2.1}$$

2.1: Equation for computing the probability of selection for an individual i, where n is the population size, $f()$ is the fitness and $p()$ is the probability of selection [5]

work. A common addition to selection algorithms is to employ elitism. Elitism is a strategy to reduce genetic drift by ensuring the most elite individuals are carried through to the next generation indefinitely [21]. This work employs elitism to ensure that the best-found solutions are carried over to the next generation.

In addition to selection, crossover is another core genetic operator in a GA. Crossover operators combine the genetic information of two or more parent chromosomes to produce offspring [44]. One-point, two-point, k-point, and uniform crossover operators are commonly used in GAs [75]. Additionally, partially matched (mapped) crossover (PMX) [26] is a crossover method which performs better than most crossover operators [44]. PMX works by having one parent donate part of its genetic material r[38]. Regarding this work, an issue with the above-mentioned approaches, except PMX, is that duplicate genes can occur in a chromosome following crossover. For augmentation policies, it is essential to avoid duplication of augmentation operators. This is because, in practice, augmentation policies are applied sequentially and do not repeat operators. Due to the superior performance and avoidance of duplicate genes, PMX was employed for the propsed GA in this work.

The proposed work formulates the chromosomes as a value-encoding. However, it could be possible to formulate the problem as a permutation of augmentations. In this configuration, the augmentations used would be fixed, and only the ordering and intensity of the augmentations would be altered in the evolutionary process. Many different methods for crossover in permutation-based genetic algorithms exist. Some permutation-based genetic algorithms are Order crossover 1, Order crossover 2, Cycle crossover, Partially-mapped crossover, One-point crossover, Two-point crossover, Uniform crossover, Position based crossover and Best-order crossover [21]. Order crossover 2 is a very intuitive crossover method; it produces a child individual from two parents by randomly choosing n random points from one parent and copying them in the offspring in the order imposed by the other parent [21]. Using a permutation-based encoding would limit our ability to have relatively small-sized chromosomes with diverse augmentation policies; this is the main reasoning behind employing a value-encoding.

The last of the three genetic operators is mutation. The mutation operator is applied to an individual produced by the crossover operation; mutation is applied with a probability of P_{mut} [4]. The mutation rate is typically set to a low value as, with a high mutation rate, GAs convert to a primitive random search algorithm [60]. Some standard mutation operators include Uniform, Non-uniform and Gaussian mutation [60]. For value-encodings, mutation operators typically need to be defined by the algorithm designer [44].

Mutation and crossover typically are applied with a predefined probability $pmut$, pcx. To build on this, the Adaptive Genetic Algorithm (AGA) was proposed by Srinivas *et al.* [76]. This approach adjusts the probability of mutation and crossover according to the individual's fitness. AGA ensures that High-fitness solutions are protected, while non-performant solutions are heavily disrupted *i.e.* crossed over and mutated [76]. For this reason, this work employs adaptive mutation and crossover rates rather than the classical fixed mutation and crossover rates.

2.3 Hyper-Parameter Optimization

It has been shown that machines have superhuman performance on very well-defined problems. However, due to the problems' narrow nature, there is still a massive gap between human and artificial intelligence. A key aspect of human intelligence is that we can quickly generalize knowledge from a previously learned task or concept and apply it to a new task or concept. Meta-Learning is the science of systematically observing how different Machine Learning approaches perform on a wide range of tasks and then learning from this experience to learn new tasks much faster [39]. Meta-Learning is interested in this by targeting the problem of *learning to learn*. The problem of augmentation optimization can be thought of as *learning to learn*, in the sense that we are learning which augmentations allow are most optimal for the learning process.

Augmentation Optimization can be more formally defined as a hyper-parameter optimization (HPO) problem. HPO is a subset of Meta-learning that aims to learn which configurations work better on a particular data set. It aims to learn from experience to design algorithms with optimal hyperparameters. In this formulation, the augmentation strategy can be thought of as a hyperparameter which we are optimizing. Defining an HPO problem consists of four main components: an estimator with an objective function, a search space *i.e.* hyper-parameter space, an search optimization method to find the hyper-parameter combinations and an evaluation function to measure how different hyper-parameter configurations affect the model [80]. Many approaches exist to solve this problem, such as the exhaustive grid search approach, random search method, Evolutionary approaches, reinforcement learning, and hyper gradient approaches [8]. In the case of this work, we investigate a small neural network with the four SSL algorithms, the search space is the set of augmentations and their respective parameters, we employ a GA to optimize the problem and utilize the performance of the SSL algorithms to measure how the different hyper-parameters are affecting the algorithms.

Evolutionary algorithms and population-based methods are popular choices for HPO because they can optimize many hyperparameters simultaneously in a manner that is better than random search [8]. Some popular evolutionary algorithms for HPO include evolutionary strategies, particle swarm optimization and GAs [8]. The technique CMA-ES [30] is a population-based technique that evaluates a randomly selected set of configurations, selects the best one, and then iteratively samples new configurations around the current best until it converges. More recently, CMA-ES has been returned to the light as a candidate for hyper-parameter optimization in deep neural networks [29]. Lorenzo *et al.* [57] employ a PSO algorithm to produce DNNs with a minimal topology to perform well on CIFAR-10.

For the purpose of HPO, GAs are easily implemented and do not require a good initialization because the genetic operators lower the possibility of missing global optima [80]. Reif *et al.* employs a GA to optimize the hyper-parameters of a support vector machine. GAs can also be used to optimize the hyperparameters of RL algorithms [71]. More recently, Yuan *et al.* [83] employs a tree-structured mutation strategy for a GA to optimize the hyperparameters of a Graph Neural Network. Another recent GA-based HPO for DL models is proposed by Erden *et al.* [23], who employs a GA to optimize a DL for a time-series prediction task. The promising results in the literature motivate the choice of a GA for HPO in this work.

DL can benefit significantly from HPO as the training pipeline has many different hyper-parameters which can be optimized [80]. Koutsoukas *et al.* [47] investigate the optimization of many different hyper-parameters in an ANN, such as the activation function, dropout regularization, number of hidden layers and number of neurons for modelling bioactivity data. Domhan *et al.* [20] propose a method that is agnostic to the optimizer used. It speeds up the hyperparameter search for CNNs, including learning rate, momentum, weight decay, number of pooling layers, learning weight decay and hyperparameters specific to the convolutional layers. Soon *et al.* [74] employ a Particle Swarm method to optimize the hyperparameters of a CNN. Overall, many attempts exist at optimizing different hyper-parameters for deep learning models, and the use of a GA has been shown to be effective for this task, therefore the proposed work employs a GA for the task of augmentation optimzation for computer vision based self-supervised learning.

2.4 Learning augmentation strategies

Data augmentation has significantly increased performance in many deep learning domains, especially computer vision. A problem with data augmentation is that the specific augmentations and hyperparameters are manually selected. This manual selection may lead to nonoptimal solutions. AutoAugment [16] was the first approach to solve this issue by automatically searching for augmentation policies. AutoAugment takes a reinforcement learning approach to find an optimal augmentation policy for a given data set. Given a set of 16 different common augmentation strategies, five strategies are selected to define a policy. Each augmentation additionally has an intensity level associated with it. In this framework, the action space is formed by selecting the different augmentation strategies and their respective intensities. The reward signal is defined by the training outcome of the neural network given the chosen policy. This approach was found to outperform manually selected augmentations. It was identified by Ho et al. [37] that due to the need to train a neural network at every step, AutoAugment was computationally infeasible to the ordinary user. To target the computational infeasibility issue present in AutoAugment, Population-Based Augmentation (PBA) was proposed. This approach significantly speeds up the computation time while maintaining comparable results to AutoAugment; this is done by learning a policy schedule rather than a fixed augmentation policy. By formulating the augmentation policy search problem as a particular case of hyperparameter schedule learning, the need to estimate the final performance of training a child model is avoided, and therefore significant speedups are possible. Like PBA, Fast AutoAugment [55], and Faster AutoAugment [32] are also proposed to accelerate the data augmentation policy search. Also building on AutoAugment, RandAugment [17] employs a random search technique which utilizes uniform sampling of different randomly generated data augmentation policies.

More recently, this idea of avoiding the requirement of estimating the final training performance has been tackled using meta-learning approaches. Meta Approach to Data Augmentation Optimization (MADAO) [33] optimizes the model and the augmentation policy by formulating the problem as a bi-level optimization problem. MADAO provides a solution that is both computationally efficient and superior in model performance. Similar to MADAO, You *et al.* [82] also optimize the augmentation policy for graph SSL using a bi-level optimization strategy. From a self-supervised perspective, a similar approach to MADAO, AutoSSL [43] is carried out for selecting pre-text tasks for self-supervised learning of graph neural networks. AutoSSL explores using meta-gradient descent and evolutionary strategies to learn SSL pre-text tasks efficiently. In addition to AutoSSL, Yang *et al.* [81] also explored using adversarial learning for optimizing augmentation policies for graph neural networks.

Although augmentation optimization is popular for graph-based SSL, for image-based SSL methods it is a relatively unexplored area. Typically with image-based SSL, the augmentation strategies are carefully selected and are heavily influenced by the seminal work carried out by chen *at al.* [12]. Wang *et al.* [79] explore alternative augmentations such as those explored in used in augmentation optimization methods such as AutoAugment. This work explores the augmentation policies used in most existing SSL methods, which they deem as *weak* augmentations, and *strong* augmentations, which can be found in augmentation optimization methods such as AutoAugment. Wang *et al.* find that it is possible to leverage the distributional divergence produced by *strong* augmentations to improve the learned representation produced using contrastive learning. Additionally, Reed *at al.* [69] proposes SelfAugment, a method which applies Fast AutoAugment and RandAugment to learn augmentation policies for image-based contrastive SSL. Reed *at al.* employs the self-supervised rotation auxiliary task to evaluate the learned representation in this work and shows that using the rotation task is highly correlated to supervised evaluation methods such as supervised fine-tuning. Due to the use of reinforcement learning, SelfAugment is thought of as a black-box approach to augmentation optimization. Through the use of a GA, the proposed approach aims to tackle the augmentation optimization problem in a more interpretable manner. This is achieved through the expansive evaluation of various augmentation configurations which is carried out in the evolution process of the GA. Therefore SelfAugment is more computationally efficient than the proposed GA, however this comes at the cost of a lower degree of interpretability. Overall, these works provide promise in the ability to optimize the augmentation policy used in image-based SSL algorithms, however, the use of a GA to do so has yet to be explored.

Chapter 3

Methodology

3.1 Genetic Data Augmentation Strategies

This work applies a Genetic algorithm to optimize the augmentation policies for four cutting-edge SSL algorithms. To represent the augmentation policies as chromosomes, we employ a value encoding. For the fitness function, we use a proxy function that aims to gauge the impact of changing the augmentation policies for a given SSL algorithm. Below, our formulation of the fitness function is elaborated. In addition to the chromosome definition and fitness function, specialized mutation and crossover methods were required to ensure that the chromosomes represented realistic augmentation strategies. However, it was possible to employ standard selection techniques for the GA. We experiment with two different modes of our GA, a single-objective and a multi-objective mode. With single-objective, we choose one SSL algorithm exclusively for the GA. With multi-objective, all four SSL algorithms are evolved together.

Taking inspiration from AutoAugment [16] and the proceeding work in learning augmentation policies, the proposed value encoding assumes a set of k augmentations, $A = \{a_1, a_2, \ldots, a_k\}$. For each augmentation, a_i an intensity value i can be defined. Figure 3.1 visualizes how intensity can be thought of through the example of the augmentation operator rotate. Given K augmentations, with an assigned intensity value i, in the single-objective mode, we represent a chromosome as a list of two-tuples representing each augmentation, of length l where $l \leq K$, *e.g.* $C_i = \{(a_1, i_1), (a_2, i_2), \ldots, (a_l, i_l)\}$. For the multi-objective mode, we represent the chromosome as a two-tuple, containing the name of one of the four SSL algorithms and then the augmentation policy, as defined for the single-objective mode, *e.g.* $C_i = (SwAV, \{((a_1, i_1), (a_2, i_2), \ldots, (a_l, i_l)\})$. When performing crossover and mutation, We do not consider the order of augmentations as important. For the multi-objective mode, we independently crossover and mutate the gene representing the SSL algorithm and the augmentation strategy. To ensure a realistic augmentation strategy without duplicate augmentations, each gene in a particular chromosome must have a unique augmentation operator.

Augmentation	**Intensity Range**	**Type**
HorizontalFlip	0.0, 1.0	float
VerticalFlip	0.0, 1.0	float
ShearX	0.0, 0.3	float
ShearY	0.0, 0.3	float
TranslateX	0, 14	int
TranslateY	0, 14	int
Rotate	-30, 30	int
Color	0.1, 1.9	float
Solarize	0.0, 1.0	float
Contrast	0.1, 1.9	float
Sharpness	0.1, 1.9	float
Brightness	0.1, 1.9	float

Table 3.1: All augmentation operators used and the associated intensity ranges. This set of operators is the same as AutoAugment [16].

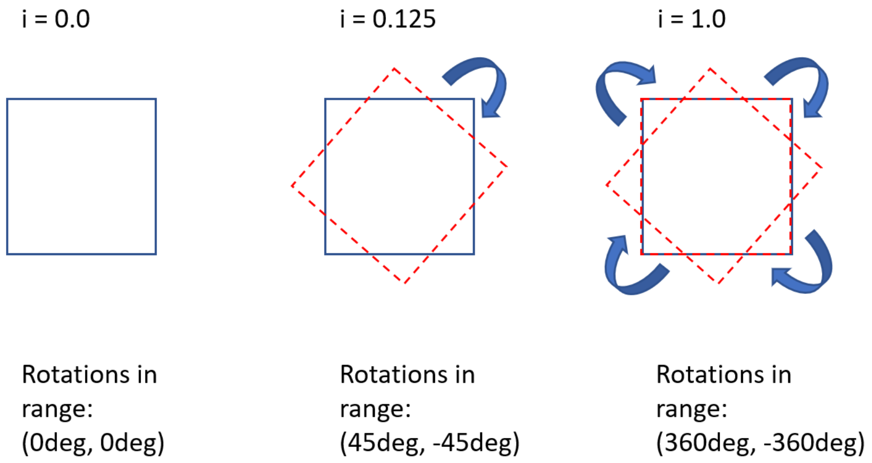

i = 0.0
i = 0.125
i = 1.0
Rotations in range: (0deg, 0deg)
Rotations in range: (45deg, -45deg)
Rotations in range: (360deg, -360deg)

Figure 3.1: Example of intensity values for the operator rotate.

the continuous ranges, whereas integer values are sampled from the respective ranges.

Once the population is initialized, the fitness of each individual must be computed. The proposed fitness function aims to evaluate an augmentation policy for a specific SSL algorithm by using the augmentation policy for training the model using the SSL algorithm and then using the learned representation in a downstream supervised task. The test accuracy in the downstream supervised task serves as the fitness. For this training process, most of the hyperparameters used in the pretext task are the same

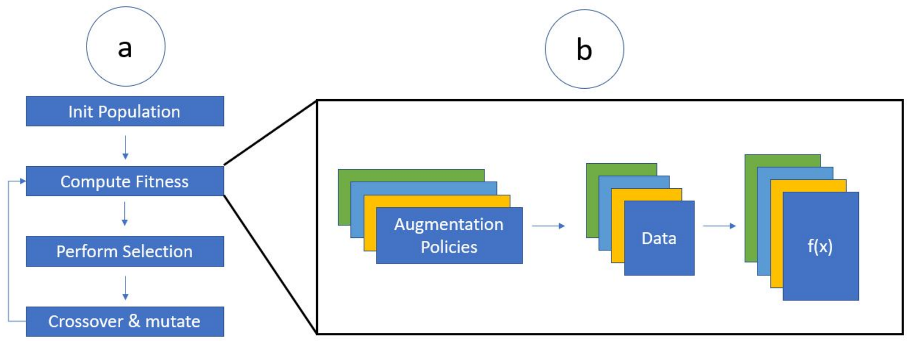


Figure 3.2: High-level overview of the proposed approach. In (a), the GA is shown, with the fitness function elaborated on in (b). The fitness function is considered the evaluation of the effect of a particular augmentation policy on the SSL training process.

as those used in the original papers. The chromosome *i.e.* augmentation policy being evaluated replaces the original augmentation policy and so this hyperparameter differs from the original SSL algorithms. Additionally, we experiment with batch sizes of 32 and 256 and train for 10 epochs. For the downstream task, the hyperparameters are fixed in all experiments, using a batch size of 32, 10 epochs, and an Adam optimizer with a learning rate of 0.001, a weight decay value of 0.0005, and no augmentation is used. In order to ensure that the results are related only to the changing of augmentation policies and not randomness within the DL pipeline, we fix model initialization and random shuffling of data for each chromosome evaluation during the run of the GA.

Following the evaluation of the population, the genetic operators selection, crossover and mutation must be applied. A large swath of possible selection methods exist for the proposed GA, such as Roulette, tournament, and Rank [67]. Our approach opts for a simple and common yet effective selection method, Roulette. In this approach, every chromosome has an equal chance of being selected proportional to its fitness.

When choosing a crossover operation for the proposed GA, caution must be taken to prevent the children's chromosomes from having duplicated augmentations; partially Mapped Crossover (PMX) [70] is employed to handle this issue. This method allows for safely crossed-over chromosomes that do not contain duplicate genes. When performing a crossover between two or more chromosomes, the mechanism for preventing duplicate genes is based purely on the augmentation operator and not the intensity. When two chromosomes are crossed over, the shared genes are completely copied, maintaining the same intensity as the parent's gene. When performing crossover for multi-objective, PMX is applied to the augmentation policy portion of the chromosome, but the SSL gene is probabilistically swapped with the other chromosome.

For the mutation operator in the proposed GA, a custom mutation function was created; we call this function $MutGaussianChoice$. With $MutGaussianChoice$, only the intensities of the augmentation operators are mutated. It does so by probabilistically mutating the augmentations intensity value within the acceptable range for that specific augmentation. The intensity is increased or decreased incrementally by the range of the intensity values divided by the increment value N, this increment is calculated using eq. 3.1 where i_{range} is the augmentation operators respective intensity range from table 3.1. For the multi-objective mode, $MutGaussianChoice$ is applied to the augmentation policy portion of the chromosome, but for the SSL gene, the gene is randomly mutated to a different SSL algorithm.

$$increment = \frac{max(i_{range}) - min(i_{range})}{10} \tag{3.1}$$

Inspired by Srinivas *et al.* [76] our method employs an adaptive rate for both crossover and mutation operations. The adaptive crossover and mutation rates are shown in equations 3.2 and 3.3. In these equations f_{max} and $\bar{f}$ denote the maximum fitness value and the average fitness value of the population, respectively, and f' denotes the larger fitness of the two chromosomes to be crossed or mutated.

$$\begin{aligned} p_{cx} = (f_{max} - f')/(f_{max} - \bar{f}), f' \geq \bar{f} \\ p_{cx} = 1, f' < \bar{f} \end{aligned} \tag{3.2}$$

$$\begin{aligned} p_{mut} = (f_{max} - f)/(f_{max} - \bar{f}), f' \geq \bar{f} \\ p_{mut} = 0.5, f < \bar{f} \end{aligned} \tag{3.3}$$

3.2 Self-Supervised Baselines, Evolutionary Self-Supervised Learning and Further Optimization of Evolved Augmentations

This work aims to understand the impact which augmentation has on the pretext task of four SOTA SSL algorithms. Using a GA to evolve augmentation policies, hundreds of models are trained during one run of the algorithm. It is critically important to provide baselines which are effective at showing how well the found results compare with the original approach. To provide baselines for the proposed GA, we use the respective SSL algorithms with their original augmentation operators. To compare the baseline with the proposed GA the downstream test accuracy is employed. A key issue with comparing the outcome of the proposed GA with the SSL baselines is that DL training procedures are inherently random. Therefore, extra precautions must be taken to ensure that the results are not due to randomness in the training pipeline. This was accomplished by controlling the random initialization and shuffling of data with a fixed seed, ensuring that the only pipeline component that changes is the augmentations itself.

For training both the baseline and evolved augmentation experiments, a small and simple convolutional neural network was used as the backbone for both pre-training and downstream classification. As pictured in 3.3, this network consisted of 3 convolutional blocks, each containing two convolutions with a kernel size of 3x3, followed by ReLu activations and a max pool and batch normalization operation. This backbone was used according to the specific algorithm for each SSL algorithm. For each of the four algorithms, the convolutional blocks shown in Figure 3.3 were used as the backbone, and the linear layers were replaced with the specific head for the SSL algorithm. As visualized in Figure 3.4, SimSiam, BYOL, and NNCLR all employ a prediction and projection head on top of the backbone shown in figure 3.3. SimSiam employs a prediction and projection head consisting of two 2-layer linear layers [14]. SimSiam is a simple approach which uses only the siamese architecture, this is not the case for BYOL, NNCLR and SwAV. BYOL employs two 2-layer linear heads for prediction and projection, and momentum encoders for both [1]. NNCLR employs a 2-layer prediction and 3-layer projection head and incorporates a memory bank [22]. Lastly, SwAV employs a 2-layer linear head, which then is passed into an additional linear layer to produce prototypes for the swapped prediction problem [9].

In addition to sharing the same neural network backbone, the original SSL training pipeline, including optimizer, loss function, and respective hyper-parameters with the exception of the batch size was employed. With batch size, it was of interest to understand its effect on the pretext task hence batch sizes of 32 and 256 were experimented with. The choice of a batch size of 32 and 256 were made to compensate for the small number of epochs used in the training process. This choice was also made with the understanding that the SSL algorithms of interest do not necessitate large epochs and large batch sizes [14], unlike earlier contrastive methods which greatly benefitted from training with a large batch size and number of epochs [12]. For the hyperparamaterization of the downstream task, a three-layer linear model with ReLu activations was added on top of the self-supervised pre-trained backbone as shown in figure 3.3, and trained for 10 epochs using an Adam optimizer with a learning rate of 0.001 and weight decay of 0.0005, cross-entropy loss, and a batch size of 32.

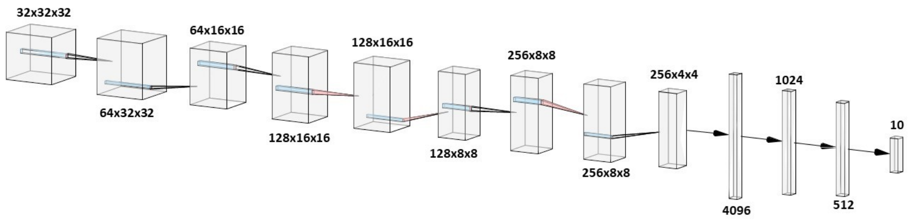


Figure 3.3: Architecture of Network used for the downstream task. The backbone (all convolutional layers leading up to the linear layers) is first pre-trained using one of the four SSL algorithms then the linear head is added to the network for fine-tuning on the downstream task.

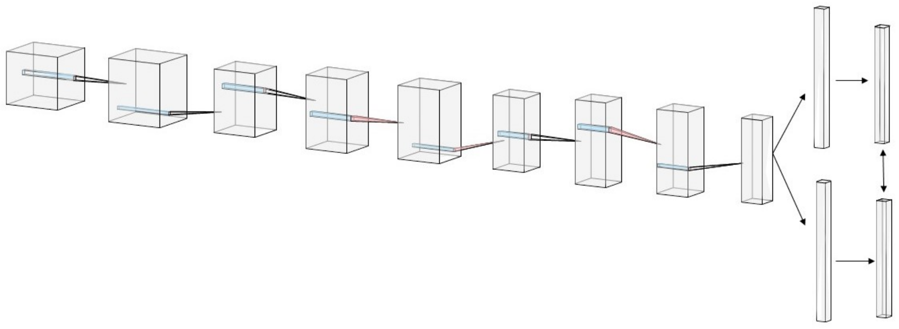


Figure 3.4: Siamese Architecture. Shared parameter neural networks are fed into a projection and prediction head.

Finally, this work is interested in exploring many different configurations of augmentations in the pretext task of common SSL algorithms. In order to do so, two relatively small yet very common image classification datasets were used, Street View House Numbers (SVHN) and Canadian Institute For Advanced Research (CIFAR-10) datasets. The CIFAR-10 dataset [48] is a popular coloured image dataset consisting of 10 classes airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. This dataset consists of 60000 32x32 images and has predetermined a train test split with 50000 train images and 10000 test images. The test set consists of 1000 randomly sampled instances from each class, and then the remaining 50000 are used for the training set. In this work, we use this default train/test split. The classes in this dataset are mutually exclusive, and the creators took extra caution to ensure the similar classes, truck and automobile, contain no overlap. It is well known that ImageNet [18] is a standard benchmark for Deep Learning-based image classification. Although CIFAR-10 is considerably smaller and more straightforward, it embodies a similar idea of colourful everyday objects. CIFAR-10 is more complicated and therefore more challenging than the SVHN dataset. The SVHN Dataset [62] is an image dataset of real-world images of house numbers taken from Google street view images. This dataset is presented as a more challenging version of MNIST [19] consisting of 600,000 32x32 labelled digits cropped from street view images. The classification problem posed in SVHN is significantly more challenging than MNIST as it is much more varied due to being real-life house numbers. SVHN is a relatively simple, commonly known CV benchmark that is a suitable level of difficulty for the problem we are investigating.

3.2.1 Technical Components

The implementation of the proposed work relies on several different libraries. For the deep learning components, such as model architectures, dataset loading, and supervised training, PyTorch is used [65]. Building on PyTorch, the python library, Lightly [77], implements cutting-edge SSL algorithms. This library was used for implementing the four SSL algorithms in this research. To implement the GA in this research, the python library Deap [24] was used. Experiments were carried out using the Big Data GPU1 server which consists of a Intel i7-8700K processor which has 6 cores with 2 threads per core, a NVIDIA TITAN Xp GPU and a NVIDIA TITAN RTX GPU.

Chapter 4

Results

When employing a GA to evolve augmentation strategies for the pretext task for BYOL, SwAV, SimSiam, and NNCLR, thousands of models were trained using different augmentation strategies. The experimentation of the proposed GA consisted of three sets of experiments. First, experimentation with the single objective GA was carried out using three random seeds for each dataset and batch size used in the pretext task. Next, initial experiments for the multi-objective approach were carried out. In our constrained setting, the multi-objective approach was found to be not very effective and therefore was not thoroughly investigated. However, due to the promising results of the single-objective approach further experimentation was carried out using more random seeds. The experiment results were compared against self-supervised baselines and were found to be statistically significant for the single objective approach. A key visualization in this section is a 2x2 line plot shown in figures 4.1, 4.2 and 4.4. In these plots, the x-axis shows the generation, and the y-axis shows the average of the best fitness over all the seeds for each of the four algorithms. The best fitness at each generation is extracted for each random seed experiment, and then the average overall seeds are computed.

Several hyper-parameters were fixed for all above described experiments, these experiments include the augmentation policy size, population size and number of generations. The seminal strategy provided by Chen et al. [12], SimCLR, employed a simple augmentation policy consisting of a random cropping, color jitter and Gaussian blur. Influenced by this, we set the chromosome length k to three for all experiments. In addition to the augmentation policy size, it is important to select an effective population size N which balances computational feasibility and population diversity. Given that a neural network must be trained in both the pretext and downstream task to evaluate an individual, the population size was fixed to a size of 15. The size of 15 allows for a fair amount of diversity in the different augmentation policies, and allows for feasible compute times given the available computational resources.

This research employs a population size of 15, the choice of 15 was to limit the number of evalautions required

4.1 Initial Experiments, Single Objective Approach

In order to understand the effect of changing the augmentation operators in the pretext task of the four algorithms, experimentation with the GA was carried out using multiple random seeds using the single objective approach. For the four algorithms, BYOL, SimSiam, SwAV, and NNCLR, two datasets, SVHN and CIFAR-10, and pretext batch sizes of 32 and 256, 3 seeds were used to control all randomness in the deep learning components of the algorithm. This randomness control approach ensures that the differences in training outcomes are not due to randomnesses in the training pipeline, such as model initialization or data shuffling.

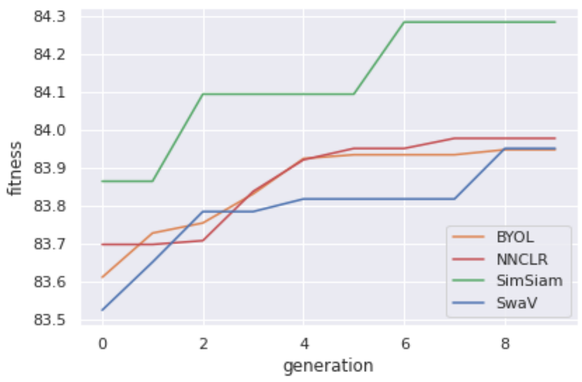
For each SSL algorithm, batch size, and data set, the proposed algorithm was employed with a population size of 15 and running for 10 generations. Running for 10 generations, with a population of 15, for two datasets and two different batch sizes resulted in a total of **7200** different augmentation in competition with one another over the 10 generations[1].

In these experiments, it was found that we can consistently improve the SSL algorithm's performance by only changing the augmentations used for the pretext task. Looking at figure 4.1, for both CIFAR-10 and SVHN, using a batch size of 32 and 256, an overall monotonic trend of optimization was observed, showing that with multiple random initializations, it is possible to improve the SSL algorithms by only changing the augmentation operators in the pretext task. This plot shows that, on average, with different random seeds, all four SSL algorithms can be improved by changing the augmentation operators used. In table 4.2, we observe that NNCLR shows the most negligible net improvement in accuracy over all the experiments, and BYOL shows the most significant improvement. However, overall, the improvements are minimal. As shown in figure 4.1; it is possible to produce small gains in performance by changing the augmentation operators.

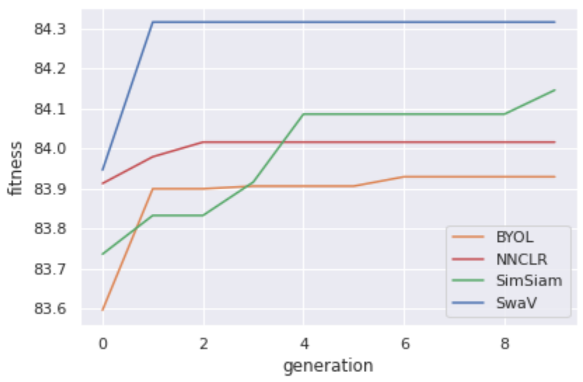
4.2 Initial Experiments, Multi-Objective Approach

In this work, the multi-objective GA was less extensively explored than the single-objective GA. This was because the added complexity of optimizing both the algorithm chosen and the augmentation policy with the multi-objective GA approach poses considerable challenges, especially given the constrained setting used in this work. As

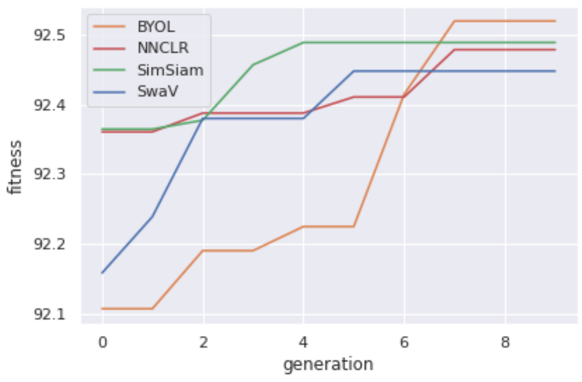
[1]Due to the nature of the GA, with individuals being carried over to the next generation, not all 7200 configurations are unique



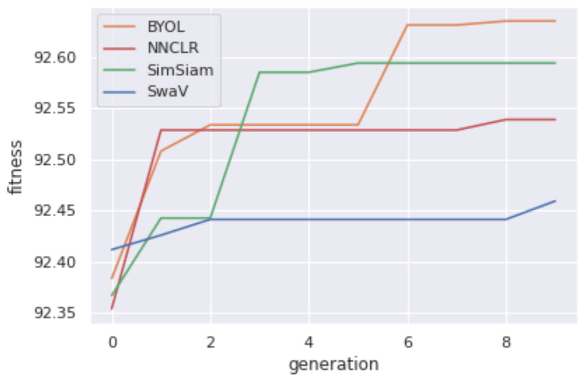
(a) CIFAR-10, BS 32



(b) CIFAR-10, BS 256



(c) SVHN, BS 32



(d) SVHN, BS 256

Figure 4.1: The average of the best-found fitness at each generation for three seeds with the single-objective approach.

shown in table 4.3, a set of random seed experiments was run for the multi-objective approach for the CIFAR-10 dataset, and only preliminary experiments were run for SVHN. The hyperparameters used for the initial single-optimization experiments were also used for the multi-objective experiments. Initial experiments showed that the multi-objective approach could not consistently optimize the unique SSL algorithm's performance. Because the population consists of all four SSL algorithms, the elitism mechanism does not preserve the best performance of all four algorithms, resulting in non-monotonic optimization for the algorithms. Due to low population size, the average best-found fitness tends to drop and increase, to the detriment of the optimization process for the multi-objective GA; this can be visualized in figure 4.2. Given the clear advantage of utilizing the single-objective approach, further investigation was conducted for only the single-objective approach.

Experiment	Algorithm	Net Improvement of Test Accuracy
bs=256, SVHN	BYOL	0.250973
bs=32, SVHN	BYOL	0.412313
bs=32, CIFAR-10	BYOL	0.336667
bs=256, CIFAR-10	BYOL	0.333333
bs=256, SVHN	NNCLR	0.184388
bs=32, SVHN	NNCLR	0.117804
bs=32, CIFAR-10	NNCLR	0.280000
bs=256, CIFAR-10	NNCLR	0.103333
bs=256, SVHN	SimSiam	0.226644
bs=32, SVHN	SimSiam	0.124206
bs=32, CIFAR-10	SimSiam	0.420000
bs=256, CIFAR-10	SimSiam	0.410000
bs=256, SVHN	SwaV	0.047378
bs=32, SVHN	SwaV	0.289387
bs=32, CIFAR-10	SwaV	0.426667
bs=256, CIFAR-10	SwaV	0.370000

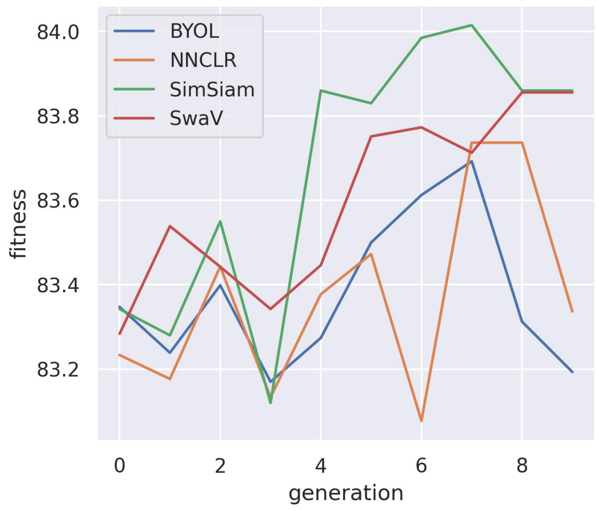
Table 4.1: The overall improvement during the optimization process for the single object GA

Algorithm	Net Improvement of Test Accuracy
BYOL	**0.333322**
NNCLR	0.171381
SimSiam	0.295213
SwaV	0.283358

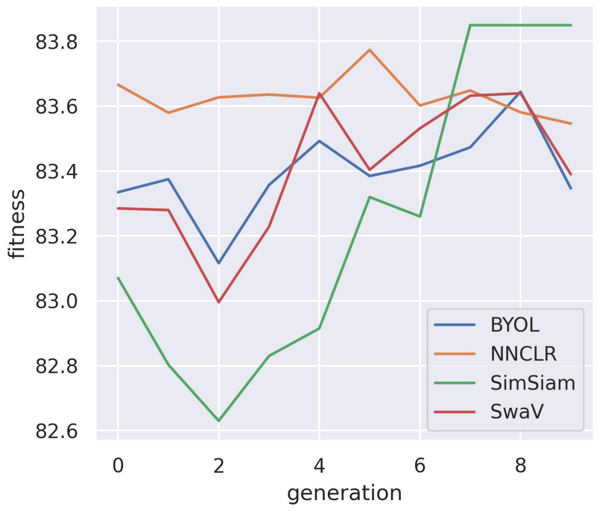
Table 4.2: Total net change in final test accuracy for the initial single-objective approach, derived from 4.1

Experiment	Algorithm	Number of Seeds
bs=32, CIFAR-10	BYOL	8
bs=256, CIFAR-10	BYOL	8
bs=256, SVHN	SwaV	1
bs=32, SVHN	BYOL	1

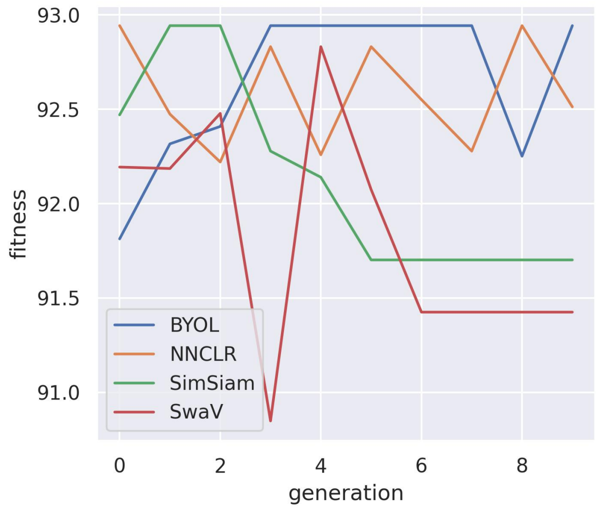
Table 4.3: Number of seeds used for initial Multi-Objective GA experiments



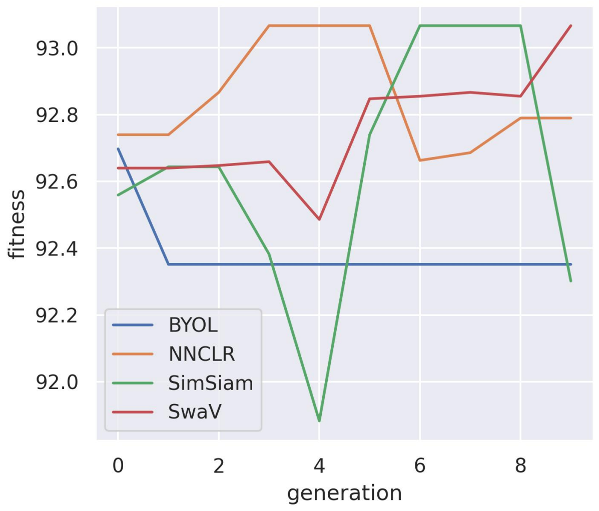
(a) CIFAR-10, BS 32



(b) CIFAR-10, BS 256



(c) SVHN, BS 32



(d) SVHN, BS 256

Figure 4.2: The average of the best-found fitness at each generation for multi-objective, for experiments listed in 4.3

4.3 Expanding on the Initial Experiments for the Single-Objective Approach

In order to further mitigate the bias due to randomness in the deep learning pipeline, more seeds were used for the single objective approach. Table 4.5 lists the total experiments. The training configuration remains the same for these experiments as the two above settings; we rerun the GA with additional random seeds for each

SSL algorithm. The number of seeds used for each experiment varies due to the choice to prioritize the investigation of specific experiments more than others. For example, those with a batch size of 256 have significantly more seeds than those with a batch size of 32 due to time constraints and interest in the slightly larger batch sizes. Overall, 52 new experiments were run, resulting in a total of **15000** different augmentation configurations in competition with one another[2]. Overall, similar trends are observed, adding more support to our initial finding that it is possible to improve the performance of the SSL algorithms by changing the augmentation policy. The best-found augmentation policies from these experiments can be visualized in figure 4.3 and the corresponding augmentation operators and intensity values are shown in table 4.4.

Dataset	Algorithm	aug1	op1	aug2	op2	aug3	op3
SVHN	SwaV	Color	1.48	TranslateX	3.00	ShearX	0.13
SVHN	BYOL	Sharpness	0.95	Contrast	1.28	Solarize	0.32
SVHN	SimSiam	Contrast	1.15	TranslateX	5.00	ShearY	0.12
SVHN	NNCLR	Color	0.90	ShearY	0.05	Solarize	0.32
CIFAR-10	NNCLR	Sharpness	0.88	Contrast	1.18	ShearX	0.10
CIFAR-10	SwaV	TranslateX	8.00	Brightness	0.69	Color	0.50
CIFAR-10	BYOL	Contrast	0.97	Sharpness	0.34	Rotate	-1.00
CIFAR-10	SimSiam	TranslateX	8.00	VerticalFlip	0.64	Contrast	1.24

Table 4.4: Best found augmentations as shown in Figure 4.3

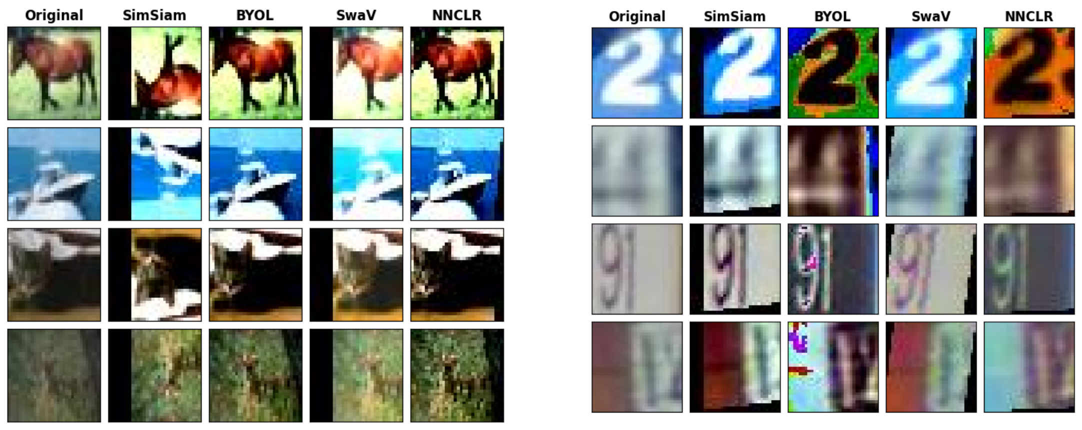


Figure 4.3: Visualization of best found operators

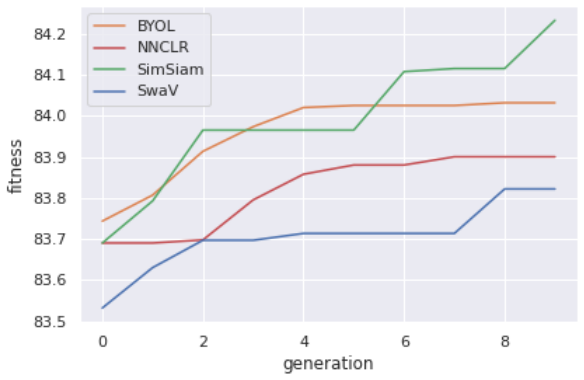
[2]Not all 15000 are unique augmentation operators.

Experiment	Algorithm	Number of Seeds
bsize=256, data=SVHN	SwaV	9
bsize=256, data=SVHN	BYOL	10
bsize=256, data=SVHN	SimSiam	7
bsize=256, data=SVHN	NNCLR	9
bsize=32, data=SVHN	SwaV	3
bsize=32, data=SVHN	BYOL	4
bsize=32, data=SVHN	SimSiam	3
bsize=32, data=SVHN	NNCLR	3
bsize=32, data=CIFAR-10	NNCLR	4
bsize=32, data=CIFAR-10	SwaV	6
bsize=32, data=CIFAR-10	BYOL	6
bsize=32, data=CIFAR-10	SimSiam	4
bsize=256, data=CIFAR-10	SwaV	8
bsize=256, data=CIFAR-10	BYOL	8
bsize=256, data=CIFAR-10	SimSiam	8
bsize=256, data=CIFAR-10	NNCLR	8

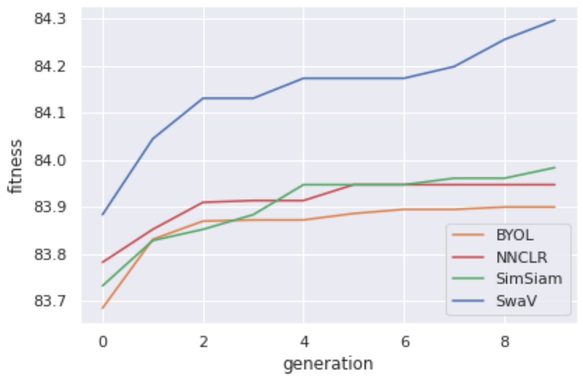
Table 4.5: Number of seeds run for each experiment,

It was found that with a higher number of random seeds, an increase in all four algorithms using a batch size of 32 and 256 for both CIFAR-10 and SVHN still occurred. We observe a monotonic optimization trend for all experiments when looking at the average best-found result. Similar to the initial experiments with fewer random seeds, figure 4.4 shows that the solutions continue to improve throughout the evolutionary process. In Figure, 4.5, the distribution of the best-found outcomes for the different random seed experiments can be observed. These four plots show that the results for all four algorithms are relatively similar. Only minor differences between the four algorithms are observed. As shown in table 4.11, it is found that NNCLR remains the lowest in terms of net improvement due to changing the augmentation policy. SimSiam is found to be the best, and BYOL appears to have taken a hit to performance gains with the introduction of more random seeds. This finding provides insight into the genetic algorithm's ability to improve the SSL algorithm's performance.

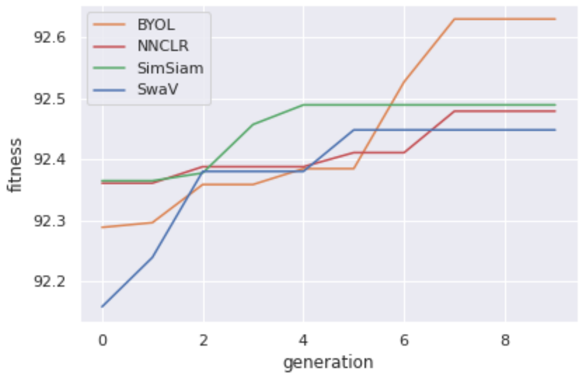
In addition to monotonic improvement, it was possible to outperform the default augmentations used in the four SSL algorithms. It was found in all cases that our approach outperforms the use of the augmentations found in the original papers. As shown in tables 4.6, 4.7, 4.8, 4.9, the single-objective GA is consistently able to outperform the SSL baseline in all experiments. In order to ensure that overall, the GA is consistently achieving better results than the baselines, statistical t-tests were



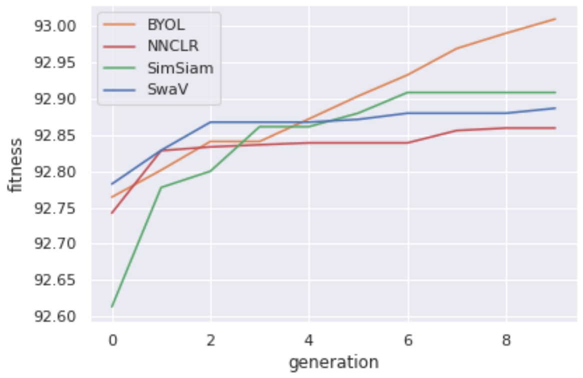
(a) CIFAR-10, BS 32



(b) CIFAR-10, BS 256



(c) SVHN, BS 32



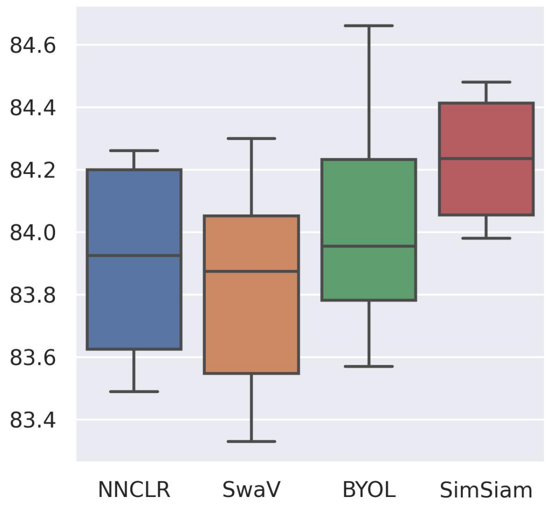
(d) SVHN, BS 256

Figure 4.4: The average of the best-found fitness for single-objective at each generation for all seeds listed in table 4.5

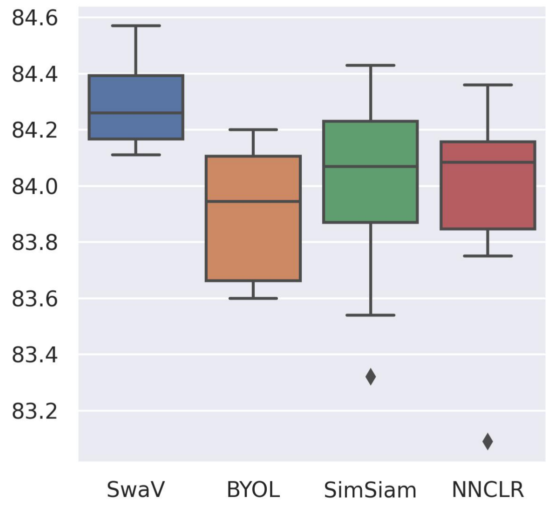
algo	BYOL	NNCLR	SimSiam	SwaV
SSL (default)	82.21963	82.443056	82.304	80.836852
SSL (multi-objective)	83.53125	83.588333	83.6625	83.75375
SSL (single-objective)	84.031667	83.9	84.2325	83.821667

Table 4.6: Average best accuracy for CIFAR-10 with a batch size of 32 in the pretext task. Results gathered from all experiments are listed in table 4.5.

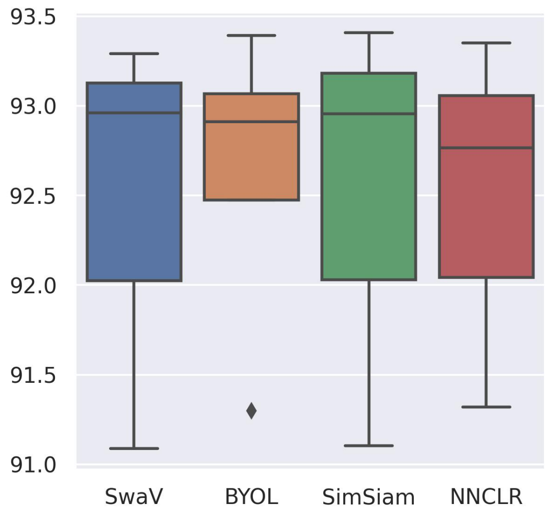
run with acceptance values of 0.95 and 0.99 for comparison with the self-supervised baselines 4.12. It can be observed that the out-performance of the self-supervised baselines is statistically significant in all cases but SVHN with a batch size of 32, showing that it is possible to outperform the original approaches by only changing the augmentation policy.



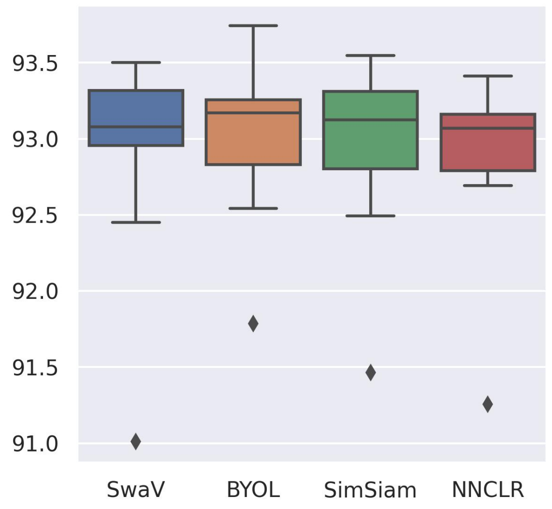
(a) CIFAR-10, BS 32



(b) CIFAR-10, BS 256



(c) SVHN, BS 32



(d) SVHN, BS 256

Figure 4.5: The distribution of best results in the final generation for all seeds as listed in Table 4.5

4.4 Further Optimizing best-found augmentation policies

It was found that through evolution, relative improvements were possible by changing only the augmentation policy. Given that the GA utilized a low number of epochs, it was of interest to understand the behaviour of training the models for larger

algo	BYOL	NNCLR	SimSiam	SwaV
SSL (default)	82.079417	82.183528	82.206167	82.847222
SSL (multi-objective)	83.735	83.82875	83.328571	83.525
SSL (single-objective)	83.9	83.9475	83.98375	84.2975

Table 4.7: Average best accuracy for CIFAR-10 with a batch size of 256 in the pretext task. Results gathered from all experiments are listed in table 4.5.

algo	BYOL	NNCLR	SimSiam	SwaV
SSL (default)	90.966475	91.144594	91.059859	90.603726
SSL (multi-objective)	N/A	N/A	N/A	N/A
SSL (single-objective)	92.629264	92.478488	92.488732	92.447757

Table 4.8: Average best found accuracy for SVHN with a batch size of 32 in the pretext task. Results gathered from all experiments are listed in table 4.5.

algo	BYOL	NNCLR	SimSiam	SwaV
SSL (default)	90.891087	91.007068	90.926667	90.779203
SSL (multi-objective)	N/A	N/A	N/A	N/A
SSL (single-objective)	93.010141	92.859643	92.908728	92.88696

Table 4.9: Average best found accuracy for SVHN with a batch size of 256 in the pretext task. Results gathered from all experiments are listed in table 4.5.

numbers of epochs. To better understand this, we trained models using the best-found augmentation policies for 50, 100, and 1000 epochs in the pretext task and 50 epochs in the downstream task. Due to the results being statistically insignificant for SVHN with a batch size of 32, this experiment was only carried out for CIFAR-10.

It was found that training for more epochs in the pretext task generally produces better results; however, it is possible to degrade this result when training for over 100 epochs in the pretext task. As visualized in Figure 4.6, we see an exciting pattern of improvement when using 50 or 100 epochs in the pretext task. However, this improvement degrades when using the significantly larger 1000 epochs. This finding suggests that training for too long and overfitting in the pretext task can lead to worse results than fewer epochs in the pretext task. Additionally, when considering the relative improvement from the initial results in the GA experiment, we see a similar trend shown in figure 4.7, where the test accuracy is improved for 50 and 100 epochs. However, for 1000 epochs, the results are consistently degraded.

Experiment	Algorithm	Net improvement of test accuracy
bs=256, SVHN	BYOL	0.245851
bs=32, SVHN	BYOL	0.340927
bs=32, CIFAR-10	BYOL	0.288333
bs=256, CIFAR-10	BYOL	0.215000
bs=256, SVHN	NNCLR	0.117377
bs=32, SVHN	NNCLR	0.117804
bs=32, CIFAR-10	NNCLR	0.210000
bs=256, CIFAR-10	NNCLR	0.165000
bs=256, SVHN	SimSiam	0.295790
bs=32, SVHN	SimSiam	0.124206
bs=32, CIFAR-10	SimSiam	0.542500
bs=256, CIFAR-10	SimSiam	0.251250
bs=256, SVHN	SwaV	0.104572
bs=32, SVHN	SwaV	0.289387
bs=32, CIFAR-10	SwaV	0.290000
bs=256, CIFAR-10	SwaV	0.413750

Table 4.10: The overall improvement during the optimization process for the single object GA

algorithm	Net improvement of test accuracy
BYOL	0.272528
NNCLR	0.152545
SimSiam	**0.303436**
SwaV	0.274427

Table 4.11: Net change for each algorithm, derived from table 4.10

4.5 Augmentation Sensitivity and Importance

In order to understand how the different augmentations affect the outcome of the self-supervised training process, we developed two metrics, Augmentation Sensitivity and Importance. Both look at the training results produced from the GA experiments to try to draw connections between final test accuracy and the different augmentations used. Augmentation importance considers the top 50 augmentation strategies based on the downstream test accuracy to determine how many different augmentation operators occur in the best outcomes. Whereas, augmentation sensitivity considers all augmentation strategies in the study and, for each augmentation operator, measures the change in test accuracy that occurs when removing it from the augmentation

batch size	dataset	Algorithm	p-val	95% C.I.	99% C.I.
32	CIFAR-10	BYOL	1.512641e-06	True	True
32	CIFAR-10	NNCLR	4.997037e-04	True	True
32	CIFAR-10	SimSiam	4.478013e-05	True	True
32	CIFAR-10	SwaV	7.108672e-08	True	True
32	SVHN	BYOL	1.121538e-02	True	False
32	SVHN	NNCLR	9.202957e-02	False	False
32	SVHN	SimSiam	1.135462e-01	False	False
32	SVHN	SwaV	6.275173e-02	False	False
256	CIFAR-10	BYOL	2.668383e-09	True	True
256	CIFAR-10	NNCLR	2.033618e-07	True	True
256	CIFAR-10	SimSiam	2.775093e-08	True	True
256	CIFAR-10	SwaV	1.132903e-09	True	True
256	SVHN	BYOL	3.205772e-10	True	True
256	SVHN	NNCLR	2.076243e-07	True	True
256	SVHN	SimSiam	9.571111e-06	True	True
256	SVHN	SwaV	4.653095e-07	True	True

Table 4.12: T-Test results for all single-objective experiments. single-objective vs. SSL baseline results

strategy.

4.5.1 Augmentation Sensitivity

In order to find the most influential augmentation operators, we analyze the chromosome sets which produced the highest classification accuracy. We investigate the contribution of each operator with the best accuracy in the downstream task. For this purpose, we run an ablation study to measure the difference in performance in the absence of each operator. Augmentation sensitivity, as shown in algorithm 1, is proposed to understand the effect of changing augmentations used in the pretext task. This method aims to understand how sensitive the SSL algorithm is to a given augmentation operator. In order to measure this, we consider the resultant accuracy of augmentation policies that include a given operator and compare it to all other augmentation policies which contain all the same operators but the one of interest. Then we obtain the difference between the average performance of this set and the performance of the original chromosome: OS(a_i) = performance(a_i) - average(performance(a_k)) for all k.

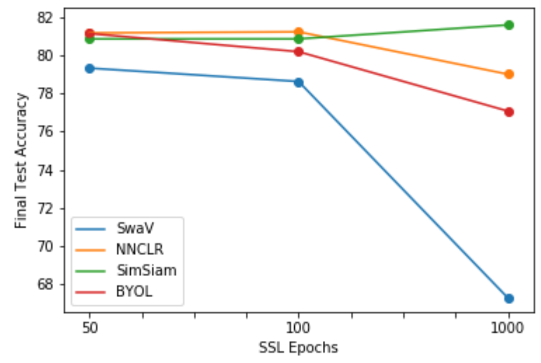
In figure 4.8, it is observed that the SSL algorithms are sensitive to specific

Algorithm 1 Computing Sensitivity

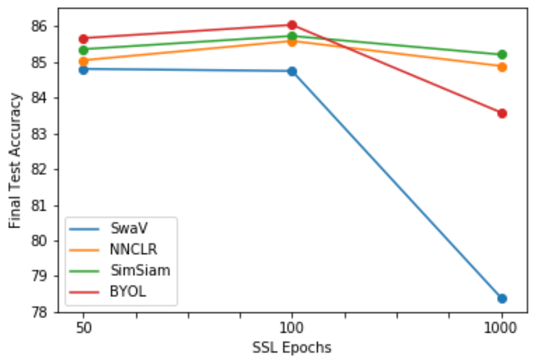
```
$aug \leftarrow$ Augmentation of interest
$C \leftarrow$Set of all Chromosomes
$Caug \leftarrow$Subset of chromosomes in $C$ that contain augmentation $aug$
$Sensitivity \leftarrow 0$
**for** $c_i$ in Caug **do**
    $AvgSimAcc \leftarrow 0$
    $TotalSimilarChromos \leftarrow 0$
    **for** $c_j$ in C **do**
        $NumEqualAugs \leftarrow 0$
        **for** $aug_i$ in $c_i$.augmentations **do**
            $NumEqualAugs \leftarrow NumEqualAugs + 1$
        **end for**
        **if** $aug$ not in $c_j$.augmentations and $NumEqualAugs \equiv c_i$.length $-1$ **then**
            $AvgSimAcc \leftarrow AvgSimAcc + c_j$.accuracy
            $TotalSimilarChromos \leftarrow TotalSimilarChromos + 1$
        **end if**
        $AvgSimAcc \leftarrow AvgSimAcc \div TotalSimilarChromos$
        $Sensitivity \leftarrow Sensitivity + |c_i$.accuracy $- AvgSimAcc|$
    **end for**
**end for**
$Sensitivity \leftarrow Sensitivity \div Caug$.length
```

$$S(aug) = \frac{\sum_{i=1}^{NCaug} |Acc(Caug_i) - AvgSimAcc(Caug_i, aug)|}{NCaug} \tag{4.1}$$

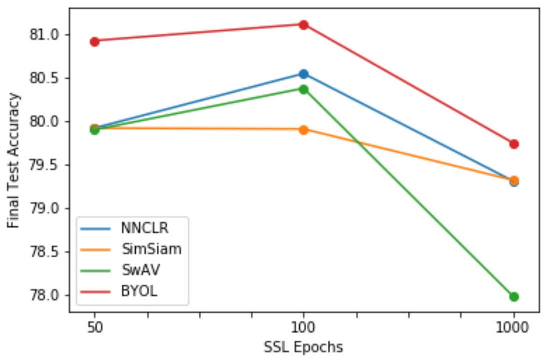
4.1: Equation for Augmentation Sensitivity, where aug is the augmentation of interest in the sensitivity computation, $Caug$ is the set of chromosomes containing augmentation aug, $NCaug$ is the number of chromosomes in $Caug$, $Acc(C_i)$ is the downstream accuracy of the chromosome C_i, and $AvgSimAcc(C_i, aug)$ is the average downstream accuracy of all the chromosomes that contain all of the same augmentations as C_i except the augmentation aug.



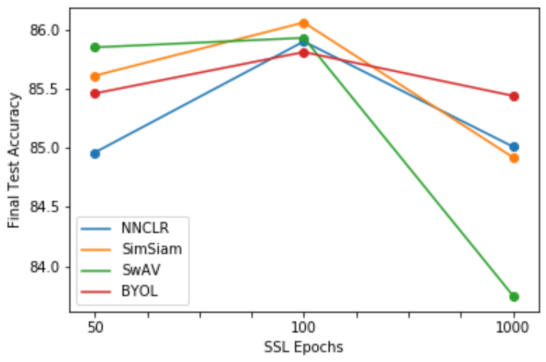
(a) BS 32, 10 epochs



(b) BS 32, 50 epochs



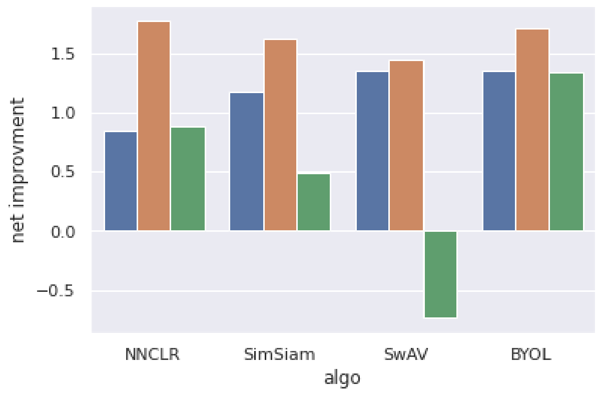
(c) BS 256, 10 epochs



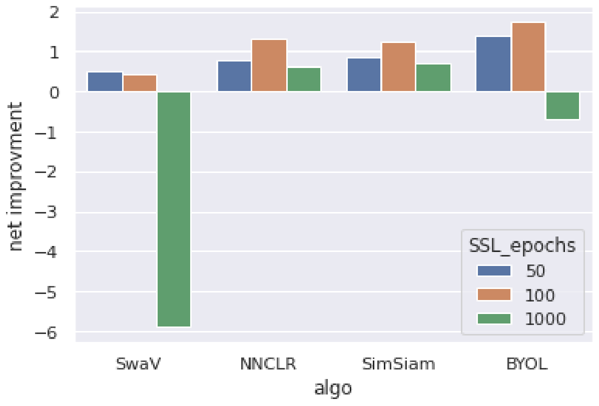
(d) BS 256, 50 epochs

Figure 4.6: Result of training models for more epochs. The X-axis on each plot represents the number of SSL pretext task epochs the models were trained for, and the Y-axis is the final test score. The top row of plots represents the best-found augmentation being trained using a pretext task batch size of 32 for 10 and 50 epochs in the downstream task. The bottom row of plots represents the best-found augmentation being trained using a pretext task batch size of 256 for 10 and 50 epochs in the downstream task.

augmentation operators; however, each algorithm is sensitive to different augmentation operators. For example, SimSiam appears to be consistently sensitive to the *translateX* augmentation. SwAV has a relatively high sensitivity to *contrast* in all but the experiment using a batch size of 32 and the SVHN dataset. BYOL consistently has a varied range of sensitivities to the different augmentation operators, yet in each experiment setting, the algorithm is most sensitive to different augmentation operators. Lastly, with NNCLR we see that the experiments using CIFAR-10 are most sensitive to *shearX*, and for SVHN, the augmentations are not as consistent as CIFAR-10, yet *colour* is relatively high for both batch sizes.



(a) BS 32, CIFAR-10



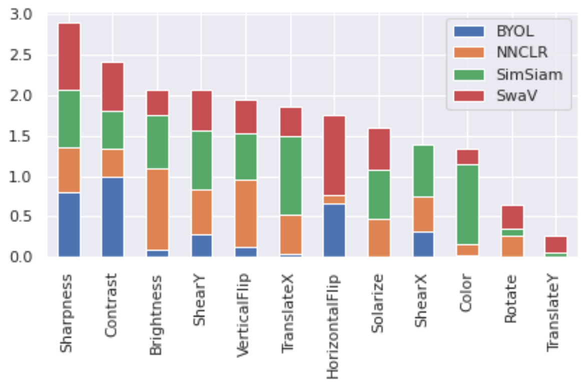
(b) BS 256, CIFAR-10

Figure 4.7: Relative improvement in test accuracy resultant of training models for more epochs in both the pretext and downstream task. The X-axis represents the different SSL algorithms, the hue represents the number of epochs used for the pretext task (50, 100, or 1000), and the y-axis represents the relative improvement in the initial accuracy of the augmentation policy.

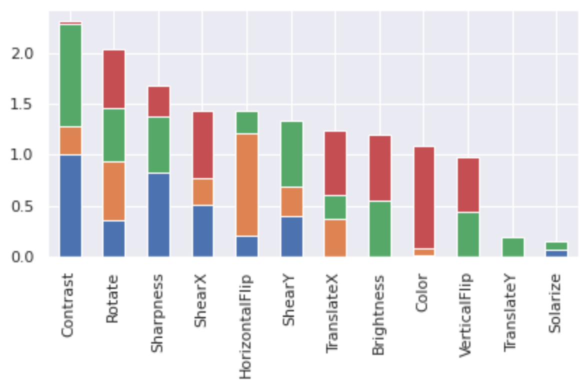
When considering the sums of sensitivities for the two batch sizes and data sets them up as shown in Figure 4.9 (a), it is clear to see specific augmentations to which the SSL algorithms are sensitive. It is found that overall, the algorithms are most sensitive to *contrast* and *sharpness* and least sensitive to *translateY*. Both flip operations have the second and third lowest overall sensitivity values, meaning that none of the algorithms appear overly sensitive to the flip operation. *ShearY*, *translateX*, *color*, *brightness*, *shearX*, *rotate*, and *solarize* all have a medium sensitivity value. These findings could suggest that all four SSL algorithms are more sensitive to the augmentations that are non-geometric transforms, *contrast*, and *sharpness* and are less sensitive to the geometric transforms, *horizontal* and *vertical flip*.

4.5.2 Augmentation Importance

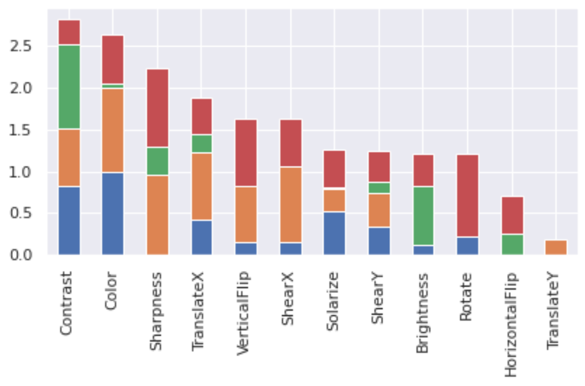
To compute augmentation importance, we calculate the number of times each augmentation operator is present in the top 50 chromosomes from all the generations (based on test accuracies). The computation of Augmentation Importance is shown in algorithm 2. Augmentation Importance can be defined as the number of times a specific augmentation appears in the top 50 chromosomes. This metric allows us to understand how different augmentations impact the top chromosomes. For each algorithm and training setting, it can be visualized that specific augmentations are



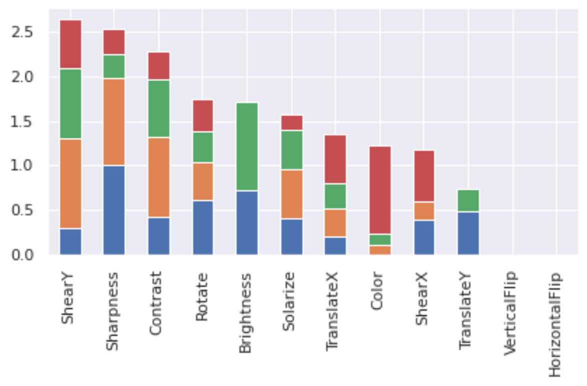
(a) CIFAR-10, BS32



(b) CIFAR-10, BS256



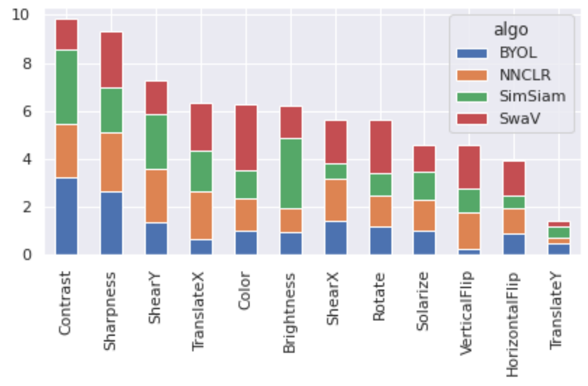
(c) svhn, BS32



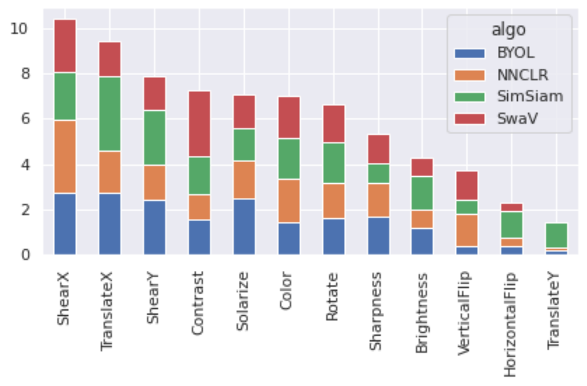
(d) svhn, BS256

Figure 4.8: Augmentation sensitivity

,



(a) Sensitivity



(b) Importance

Figure 4.9: Global View of Augmentation sensitivity and importance

dominantly important in the top 50 augmentation policies. As shown in figure 4.10, the augmentation importance for all experiments displays a skewed distribution. This finding shows that in all cases, one or several augmentations dominate, while others are uncommon or not even present in the top 50 augmentation policies.

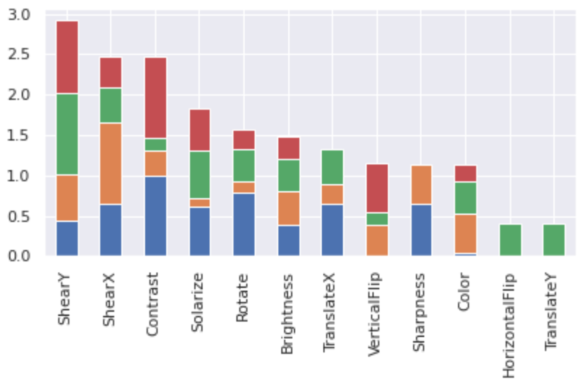
Adduitionally, it was found that certain augmentation operators are prevalent within all experimental settings, whereas others were not. In Figure 4.9 (b), it can be observed that *shearX* is prevalent in all experiments, and *horizontal flip* and *translateY* are found to be not prevalent. *Solarize*, *color* and *rotate* have relatively similar importance distributions for all experiments. Both shear operations appear to have high importance. *Contrast*, *solarize*, *color*, *rotate*, *sharpness*, and *brightness* do not show high similarity when comparing at the algorithm level. Both flip operations have globally low importance. It appears that *translateX* is a vital augmentation, but *translateY* is not. Overall, it is possible to observe that there is agreement among the four algorithms for the different augmentation's importance values.

Algorithm 2 Computing Augmentation Importance

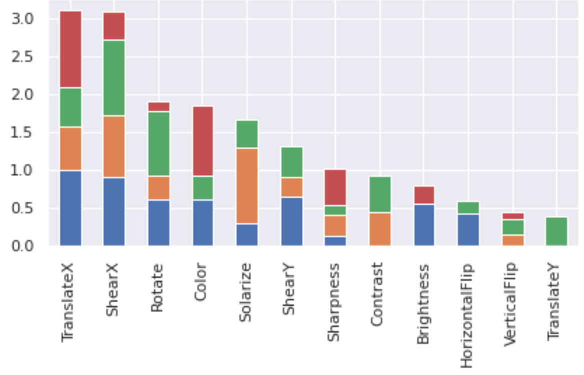
```
$aug \leftarrow$Augmentation of interest
$C \leftarrow$Set of all Chromosomes
$N \leftarrow$ Number of chromosomes to consider
$Importance \leftarrow 0$
$C \leftarrow$ sorted($C$)
for $c_i$ in C[: $N$] do
    if $aug$ in $c_i$.augmentations then
        $Importance \leftarrow Importance + 1$
    end if
end for
```

4.6 Loss Analysis

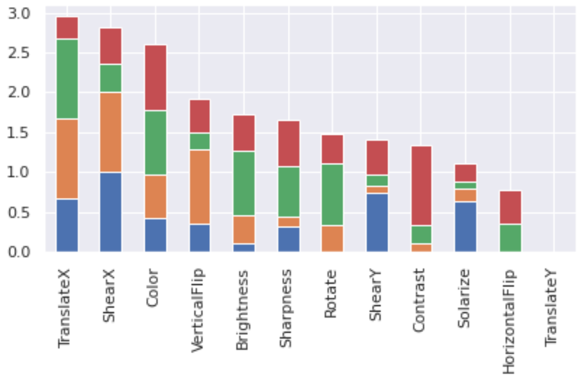
In order to understand the impact of the different SSL algorithms and the resultant representations used as the initialization point for the supervised downstream task, the loss of the downstream task was investigated. At the most basic level, the loss curves were analyzed to understand the learning process of the different algorithms. Overall, it was found that there are differences between the two batch sizes, 32 and 256, for the learning process in the downstream task. In figure 4.11, we observe that the smaller batch size results in all four SSL algorithms converge to similar solutions in the downstream task. With the larger batch size, there is more variation in the downstream optimization task optimization process between the four algorithms.



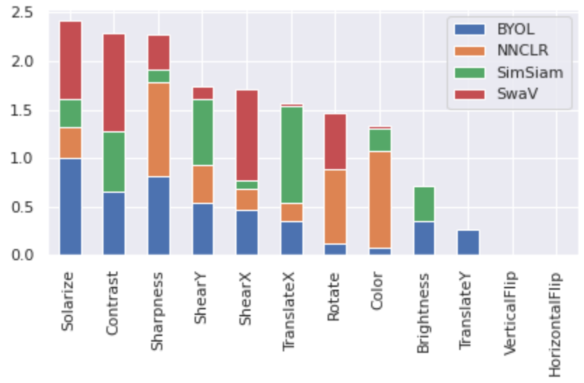
(a) CIFAR-10, BS32



(b) CIFAR-10, BS256



(c) svhn, BS32

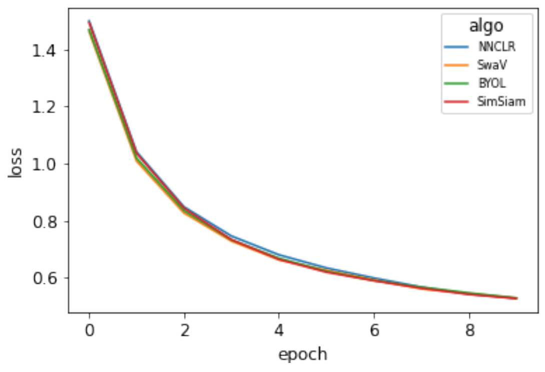


(d) svhn, BS256

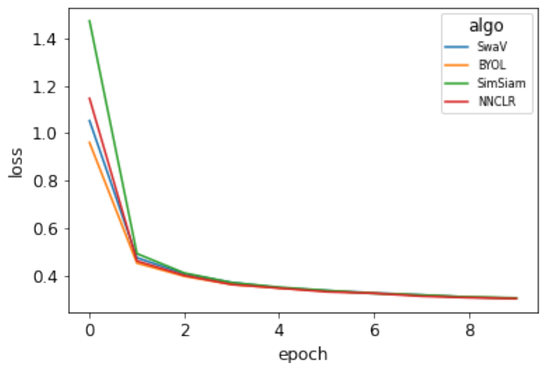
Figure 4.10: Augmentation Importance

In addition to interpreting the downstream loss curves, we employed a strategy for analyzing the loss of the resultant downstream models, known as loss landscape analysis. Loss landscape analysis aims to provide a deeper understanding of the optimized model parameters and their context in their respective parameter space by slightly perturbing the model parameters around the optimized parameterization. This method provides a *landscape* of the loss function for the given parameter space, allowing for insights into how convex or chaotic the loss space around the solution is. Producing a loss landscape for a given model architecture, problem *i.e.* dataset and loss function is not a trivial task; many different approaches exist. This work opts to employ a filter normalization approach presented by Li *et al.* [54] to visualize the loss landscape.

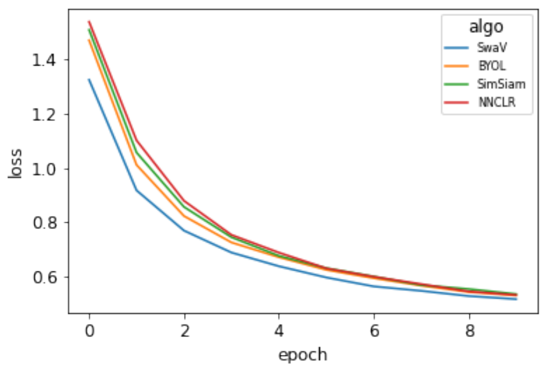
The filter normalization loss landscape visualization accurately captures the local sharpness and flatness of minimizers. Understanding the local geometry of the loss



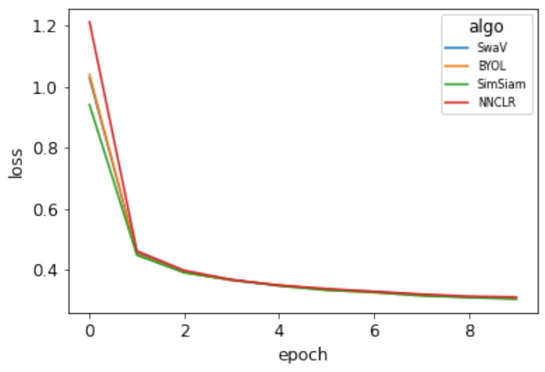
(a) CIFAR-10, BS=32



(b) SVHN, BS=32



(c) CIFAR-10, BS=256



(d) SVHN, BS=256

Figure 4.11: Downstream loss curves for the best-found solutions in the GA optimization process.

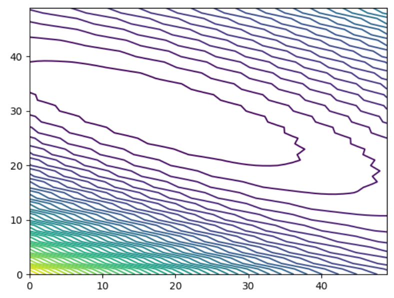
landscape is essential for understanding the behaviour of the loss function in the problems at hand. As discussed in [54], a vital component of the filter-wise normalization technique is a random vector technique used by [27], [40]. In this technique, two random vectors δ and η are sampled from a random Gaussian distribution, and a center point $\theta*$, where $\theta*$ is the optimized parameter configuration. Then a 2d contour is generated by plotting the function4.2. The critical component of this approach, which differs from earlier random vector approaches, is that the sampled directions are filter-wise normalized. The random vectors are scaled to the filters within the CNN to ensure that the area covered by the random vectors is not too small or too large for the set of parameters.

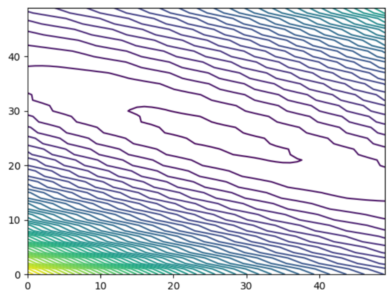
As with the experiments for further optimization, the loss landscape analysis was only carried out using the models resultant from training with CIFAR-10. For all four SSL algorithms, NNCLR, BYOL, SimSiam, and SwAV, and the two batch sizes 32

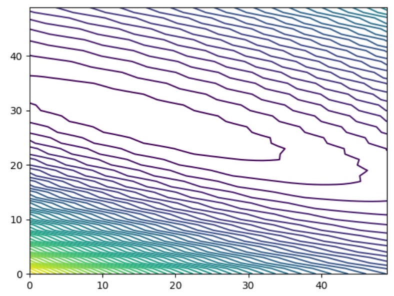
$$f(\alpha, \beta) = L(\theta * +\alpha\delta + \beta\eta) \tag{4.2}$$

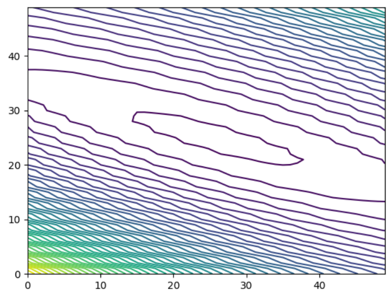
4.2: Random direction function[54], where α, and β are the respective lengths of random vectors δ and η. Used by [27], [40] and adapted by [54] with a filter normalization component.

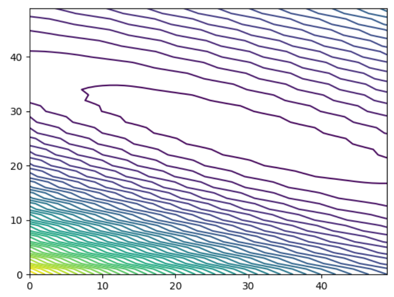
and 256, we use the filter normalization technique with a α and β both set to 10 and a sample size of 50 resulting in a $50x50$ loss landscape. It was found that changing the batch size in the pretext task while fixing the batch size in the downstream task results in visually discernible differences in sharpness in all four SSL algorithms. Li *et al.* [54] found that with the filter normalization method, it was possible to visualize how batch size impacts minima sharpness; large batches were found to produce visually sharper minima. Remarkably, in all four SSL algorithms, we notice a difference in sharpness when training with a batch size of 32 and 256 in the pretext task and keeping a fixed batch size of 32 in the downstream task. As visualized in Figure 4.12, we observe that BYOL, NNCLR, and SimSiam all have converged to sharp minima in the downstream task when using a batch size of 256 in the pretext task and a visibly flatter minimum when using a batch size of 32 in the pretext task, the opposite result is observed in SwAV.

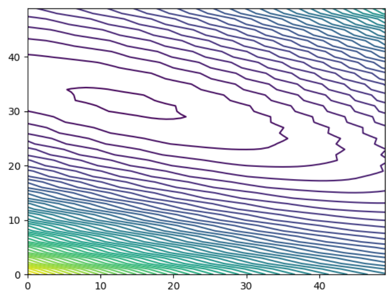

(a) BYOL, BS=32

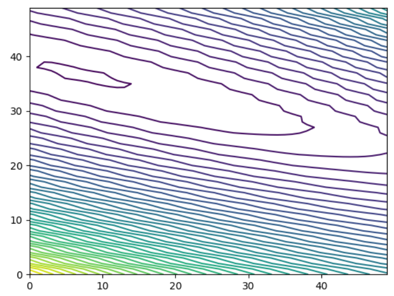

(b) BYOL, BS=256


(c) NNCLR, BS=32


(d) NNCLR, BS=256


(e) SimSiam, BS=32


(f) SimSiam, BS=256


(g) SwAV, BS=32

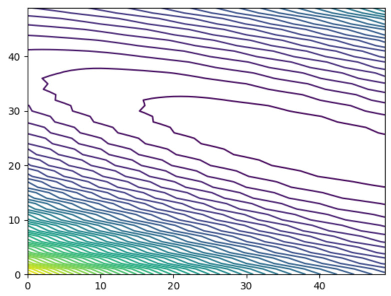

(h) SwAV, BS=256

Figure 4.12: Comparing the loss landscape of the downstream models, using SSL methods with a batch size of 32 (left column) and batch size of 256 (right column). The smaller batch size in the SSL pretraining leads to a less sharp minimization in the downstream task.

Chapter 5

Challenges and Future Work

Several challenges were faced while carrying out the experimentation for this thesis. Most notably, using a GA for the optimization of a deep learning training pipeline is inherently problematic due to the costly nature of training deep neural networks. To reduce the computational requirements of the algorithm, a relatively small neural network, a low number of epochs, a low population size, and a small number of generations were used. At the cost of performance, reducing these components lessens computational requirements in both time and space. The cost of performance due to low population size was especially detrimental for the multi-objective GA, which failed due to its inability to maintain the best-found augmentation policies. This finding could be attributed to the small population size; the loss of a single chromosome containing a specific SSL gene greatly impacted the SSL's average best test accuracy. With a larger population size, it may be possible to maintain better augmentation policies across all four algorithms. Additionally, with both single-objective and multi-objective, the design choice to only mutate the intensities was to limit the impact mutation can have on the population. It maybe possible that with a larger population size and higher number of elite individuals, also mutating the augmentation operator could lead to more diversity within the population. Given the constraints of this thesis, the increase in population size for both the single-objective and multi-objective approach is left for future work.

For the single-objective GA, it was possible to monotonically improve the test accuracy for all four algorithms. However, the experiments were run for 10 generations regardless of the trend of optimization; this means that the best possible accuracy was not found. The best possible accuracy for the given hyper-parameter configuration could be found if the experiments were run until the test accuracy no longer improved, this is left as an avenue for future work. Additionally, given the reduced parametrization of the deep learning pipeline the final test accuracies found are far from state-of-the-art. It may be possible in future work to improve on the state-of-the-art by applying the single-objective GA to a deep learning pipeline that employs a more significant number of epochs and a larger deep neural architecture. The increase in architecture size and training time increases the amount of compute resources required but provides the potential for being on par with or surpassing state-of-the-art. In addition to the hyperparameters for the deep neural network, an area of interest for future work would be to increase the population size and number of generations the single-objective GA is run for. By running larger-scale experiments with larger neural networks, longer epochs, for more generations, and larger population sizes, state-of-the-art results may be possible.

In addition to increasing the population size and number of generations, parallelization of the GA could be another avenue of improvement. A GA is well suited to parallelization because it is a population-based approach, and all solution candidates can be evaluated simultaneously [31]. The problem present in this thesis is slightly more complicated as the fitness function relies on GPU computation, so the parallelization of solution fitness evaluation requires multiple CPUs and GPUs. For example, if we want four workers in a parallelized version of the proposed algorithm, four CPUs and four GPUs would be required. The proposed algorithm's parallelization could yield significant computation time improvements if the overhead is sufficiently low.

Another challenge faced with this work is the ill-defined nature of measuring the *goodness* of a representation learned in a self-supervised manner. To measure the fitness of the learned representations, the proposed method uses a downstream classification task with the same dataset used in the pretext task of the SSL algorithm. This measurement allows us to understand the transferability of the learned representation, from the implicitly labelled pretext task to the explicitly labelled downstream classification task. However many different downstream tasks exist for measuring the *goodness* of the learned representation. One task is known as linear probing, in this method, the learned representation is frozen and a linear layer is trained on top to perform the classification task. A variant of the linear probing method would be to take a semi-supervised approach where a small set of labelled data is used to fine-tune the learned representation from the SSL task. Both of the described methods are introduced in the seminal work produced by Chen *et al.* [12]. It is possible that a method such as linear probing could provide a better fitness measurement, and is left as an area for future investigation.

An additional shortcoming is using only three augmentation operators. Cosentino *et al.* [15] showed empirically and theoretically that the larger the augmentation policy, the better the result will be. Additionally, many works in augmentation optimization including AutoAugment [43] employ an augmentation policy with five augmentation operators. The choice of an augmentation policy with three operators was to reflect the design choices made by the seminal work in the SimCLR approach. However, it is possible that with increasing the number of augmentation operators, the resultant policies would be more effective and facilitate higher performance in the SSL training process. The exploration of using larger augmentation policies with the proposed GA is an area of future work, which could lead to better augmentation policies.

In addition to improving the Multi and Single Objective GAs through more-extensive experiments, it may be possible to derive deeper insights from the data produced from the experiments in this thesis. As mentioned in chapter 4, 15000 chromosomes representing augmentation policies were recorded during the many single-objective Genetic Algorithm experiments. These results can be considered a meta-dataset, including the SSL algorithm name, dataset name, the three augmentation operators, respective intensities, and the resultant downstream test accuracy. This thesis designed two metrics for investigating this dataset, augmentation sensitivity and importance; however, it would be possible to dive deeper into these results using other methods. One possible method could be to fit a model to this data and then use explainability methods such as SHAP [66], or LIME [58] to explain the model and gain deeper insight into the relationship between the augmentations, datasets, and SSL algorithms.

Overall, many different challenges exist in this thesis which open up avenues for future work. A clear direction for future work is using a deep learning pipeline with larger batch sizes, neural networks, and epochs, as well as running the GA for more generations and with a higher population size. However, the increase in these hyperparameters leads to larger computational requirements in both time and space, the parallelization of the GA could significantly aid in handling this issue. In addition to larger-scale algorithms, the outcome of the proposed GA could be improved by using a different downstream task such as linear probing. Another area of potential improvement with the proposed GA is the number of augmentations used. In addition to improvements to the algorithm, it is also possible to gain insights from the results derived from the GA experiments using more complex explainability methods.

Chapter 6

Conclusion

Self-supervised learning for computer vision has proven to be an groundbreaking approach for training neural networks with large unlabelled datasets. Many of these algorithms leverage data augmentaiton as a core component of the algrothm, therefore it is of great importance that the selected augmentation operators are optimal for the given task. Motivated by this point, the proposed work aimed to improve and gain a deeper understanding of the augmentation operators used for four prevalent SSL algorithms: SwAV, BYOL, NNCLR, and SimSiam. It was found that the proposed GA was able to successfully optimize the augmentation policy used in the pretext task of the SSL algorithms, consistently outperforming the augmentations which were used in the original papers for the four algorithms. Due to the population-based nature of the proposed GA, we were not only able to optimize the SSL algorithms augmentation policy but also were able to reveal the extent to which the algorithms are impacted by the choice of augmentation operators used for the pretext task. It was found that specific augmentation operators can have varying effects depending on the dataset and SSL algorithm being used.

To design the GA, we define a chromosome representation for the augmentation policy to represent the hyper-parameter optimization problem, including three augmentations operators per chromosome. Crossover, selection, and mutation were performed on these augmentation policies to efficiently explore a diverse set of augmentation policies. In order to avoid results due to randomness within the system, such as model initialization, random splitting, and shuffling of the data, the experiments were carried out for multiple random seeds, and the observations made afterward looked at the entirety of the experiments.

Four sub-experiments were defined to understand the effect of changing augmentation operators in the SSL pretext tasks, one for each batch size, 32 and 256, and two datasets, SVHN and CIFAR-10. For all settings, it was found that it was possible to outperform the default augmentations used in the original SSL algorithms. Although these findings are well below SOTA, it was found that relative improvement is possible by just changing the augmentation operators. When running for many different random seeds, it is found that this is consistently the case. These findings show that it may be possible to produce test results on par with the SOTA baseline results with a more efficient optimization process. Additionally, when training for longer epochs in both the pretext and downstream tasks, it is possible to improve the downstream test accuracy.

To further analyze the results of the experiments, two metrics were proposed: augmentation sensitivity and importance. Augmentation sensitivity allows for the visualization that the different SSL algorithms and dataset configurations are sensitive to the presence of different augmentation operators in the pretext task. Overall, when looking at the sensitivities for each SSL algorithm over all experiments, there are trends of the algorithms being dominantly sensitive to specific augmentation operators. Similar results were found with Augmentation importance. Specific augmentations are significant for the resultant downstream test accuracy. Additionally, at the global level, augmentation operators consistently yield higher downstream test accuracy for the different SSL algorithms.

Self-supervised learning for computer vision provides hope for alleviating the need for massive labelled datasets and data Augmentation serves as a core component for many of the existing methods. This thesis highlighted how the choice of augmentation policies for the pretext task affects the performance of SSL algorithms. In a constrained setting, improving the classification performance for all four SSL algorithms on two common benchmark datasets was possible. To our knowledge, this thesis is the first attempt at employing a GA to optimize the augmentation policies in contrastive SSL algorithms. Although the results presented in this thesis do not compete with SOTA results, it is clear that in a very constrained setting, it is possible to achieve statistically significant improvement by only changing the augmentations in the pretext task. Albeit small, this thesis takes the first step toward employing a GA-based, method for improving the augmentation policies used in SOTA SSL algorithms.

Bibliography

[1] Bootstrap your own latent: A new approach to self-supervised Learning, September 2020. arXiv:2006.07733 [cs, stat].

[2] Saleh Albelwi. Survey on Self-Supervised Learning: Auxiliary Pretext Tasks and Contrastive Learning Methods in Imaging. *Entropy*, 24(4):551, April 2022. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

[3] Md Zahangir Alom, Tarek M. Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Esesn, Abdul A S. Awwal, and Vijayan K. Asari. The history began from alexnet: A comprehensive survey on deep learning approaches, 2018.

[4] Anca Andreica and Camelia Chira. Best-order crossover for permutation-based evolutionary algorithms. *Applied Intelligence*, 42(4):751–776, June 2015.

[5] Thomas Bäck, David B Fogel, and Zbigniew Michalewicz. Handbook of evolutionary computation. *Release*, 97(1):B1, 1997.

[6] Thomas Bäck, David B Fogel, and Zbigniew Michalewicz. *Evolutionary computation 1: Basic algorithms and operators*. CRC press, 2018.

[7] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning*, volume 1. MIT press Cambridge, MA, USA, 2017.

[8] Pavel Brazdil, Jan N. van Rijn, Carlos Soares, and Joaquin Vanschoren. Metalearning for Hyperparameter Optimization. In Pavel Brazdil, Jan N. van Rijn, Carlos Soares, and Joaquin Vanschoren, editors, *Metalearning: Applications to Automated Machine Learning and Data Mining*, Cognitive Technologies, pages 103–122. Springer International Publishing, Cham, 2022.

[9] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments, January 2021. arXiv:2006.09882 [cs].

[10] Junyi Chai, Hao Zeng, Anming Li, and Eric W.T. Ngai. Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Machine Learning with Applications*, 6:100134, 2021.

[11] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. Semi-supervised learning. adaptive computation and machine learning. *MIT Press, Cambridge, MA, USA. Cited in page (s)*, 21(1):2, 2010.

[12] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. *arXiv:2002.05709 [cs, stat]*, June 2020. arXiv: 2002.05709.

[13] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning, 2020.

[14] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021.

[15] Romain Cosentino, Anirvan Sengupta, Salman Avestimehr, Mahdi Soltanolkotabi, Antonio Ortega, Ted Willke, and Mariano Tepper. Toward a geometrical understanding of self-supervised contrastive learning. *arXiv preprint arXiv:2205.06926*, 2022.

[16] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. AutoAugment: Learning Augmentation Policies from Data. *arXiv:1805.09501 [cs, stat]*, April 2019. arXiv: 1805.09501.

[17] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020.

[18] 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*, pages 248–255, 2009.

[19] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[20] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[21] Haiming Du, Zaichao Wang, Wei Zhan, and Jinyi Guo. Elitism and Distance Strategy for Selection of Evolutionary Algorithms. *IEEE Access*, 6:44531–44541, 2018. Conference Name: IEEE Access.

[22] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a Little Help from My Friends: Nearest-Neighbor Contrastive Learning of Visual Representations, October 2021. arXiv:2104.14548 [cs].

[23] C Erden. Genetic algorithm-based hyperparameter optimization of deep learning models for pm2. 5 time-series prediction. *International Journal of Environmental Science and Technology*, pages 1–24, 2023.

[24] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.

[25] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *CoRR*, abs/1803.07728, 2018.

[26] David E Goldberg and Robert Lingle. Alleles, loci, and the traveling salesman problem. In *Proceedings of the first international conference on genetic algorithms and their applications*, pages 154–159. Psychology Press, 2014.

[27] Ian J. Goodfellow, Oriol Vinyals, and Andrew M. Saxe. Qualitatively characterizing neural network optimization problems, 2014.

[28] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, and Tsuhan Chen. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[29] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18, 2003.

[30] Nikolaus Hansen and Andreas Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2):159–195, June 2001. Conference Name: Evolutionary Computation.

[31] Tomohiro Harada and Enrique Alba. Parallel genetic algorithms: A useful survey. *ACM Comput. Surv.*, 53(4), aug 2020.

[32] Ryuichiro Hataya, Jan Zdenek, Kazuki Yoshizoe, and Hideki Nakayama. Faster autoaugment: Learning augmentation strategies using backpropagation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16*, pages 1–16. Springer, 2020.

[33] Ryuichiro Hataya, Jan Zdenek, Kazuki Yoshizoe, and Hideki Nakayama. Meta approach to data augmentation optimization. *CoRR*, abs/2006.07965, 2020.

[34] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. *arXiv:1911.05722 [cs]*, March 2020. arXiv: 1911.05722.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[36] Kassel Hingee and Marcus Hutter. Equivalence of probabilistic tournament and polynomial ranking selection. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 564–571. IEEE, 2008.

[37] Daniel Ho, Eric Liang, Ion Stoica, Pieter Abbeel, and Xi Chen. Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules. *arXiv:1905.05393 [cs, stat]*, May 2019. arXiv: 1905.05393.

[38] Abid Hussain, Yousaf Shad Muhammad, M Nauman Sajid, Ijaz Hussain, Alaa Mohamd Shoukry, Showkat Gani, et al. Genetic algorithm for traveling salesman problem with modified cycle crossover operator. *Computational intelligence and neuroscience*, 2017, 2017.

[39] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors. *Automated Machine Learning: Methods, Systems, Challenges.* The Springer Series on Challenges in Machine Learning. Springer International Publishing, Cham, 2019.

[40] Daniel Jiwoong Im, Michael Tao, and Kristin Branson. An empirical analysis of deep network loss surfaces. *CoRR*, abs/1612.04010, 2016.

[41] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning, 2020.

[42] Khalid Jebari. Selection methods for genetic algorithms. *International Journal of Emerging Sciences*, 3:333–344, 12 2013.

[43] Wei Jin, Xiaorui Liu, Xiangyu Zhao, Yao Ma, Neil Shah, and Jiliang Tang. Automated self-supervised learning for graphs, 2021.

[44] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80(5):8091–8126, February 2021.

[45] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, May 2014. arXiv: 1312.6114.

[46] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[47] Alexios Koutsoukas, Keith J Monaghan, Xiaoli Li, and Jun Huan. Deep-learning: investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data. *Journal of cheminformatics*, 9(1):1–13, 2017.

[48] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[49] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, may 2017.

[51] Anit Kumar. Encoding schemes in genetic algorithm. *International Journal of Advanced Research in IT and Engineering*, 2(3):1–7, 2013.

[52] Phuc H. Le-Khac, Graham Healy, and Alan F. Smeaton. Contrastive Representation Learning: A Framework and Review. *IEEE Access*, 8:193907–193934, 2020. Conference Name: IEEE Access.

[53] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[54] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the Loss Landscape of Neural Nets. Technical Report arXiv:1712.09913, arXiv, November 2018. arXiv:1712.09913 [cs, stat] type: article.

[55] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. *Advances in Neural Information Processing Systems*, 32, 2019.

[56] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised Learning: Generative or Contrastive. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2021. Conference Name: IEEE Transactions on Knowledge and Data Engineering.

[57] Pablo Ribalta Lorenzo, Jakub Nalepa, Michal Kawulok, Luciano Sanchez Ramos, and José Ranilla Pastor. Particle swarm optimization for hyper-parameter selection in deep neural networks. In *Proceedings of the genetic and evolutionary computation conference*, pages 481–488, 2017.

[58] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017.

[59] Huanru Henry Mao. A survey on self-supervised pre-training for sequential transfer learning in neural networks, 2020.

[60] Seyedali Mirjalili and Seyedali Mirjalili. Genetic algorithm. *Evolutionary Algorithms and Neural Networks: Theory and Applications*, pages 43–55, 2019.

[61] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[62] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[63] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. *CoRR*, abs/1603.09246, 2016.

[64] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. *arXiv:1609.03499 [cs]*, September 2016. arXiv: 1609.03499.

[65] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[66] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[67] Riccardo Poli, William B. Langdon, Nicholas F. McPhee, and John R. Koza. *A field guide to genetic programming.* Lulu Press], [Morrisville, NC, 2008.

[68] Gregory JE Rawlins. *Foundations of genetic algorithms.* Morgan Kaufmann Publishers Inc., 1992.

[69] 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 (CVPR)*, pages 2674–2683, June 2021.

[70] Colin Reeves. Genetic Algorithms. In Fred Glover and Gary A. Kochenberger, editors, *Handbook of Metaheuristics*, pages 55–82. Springer US, Boston, MA, 2003.

[71] Adarsh Sehgal, Hung La, Sushil Louis, and Hai Nguyen. Deep Reinforcement Learning Using Genetic Algorithm for Parameter Optimization. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pages 596–601, February 2019.

[72] Connor Shorten and Taghi M. Khoshgoftaar. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1):60, July 2019.

[73] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[74] Foo Chong Soon, Hui Ying Khaw, Joon Huang Chuah, and Jeevan Kanesan. Hyper-parameters optimisation of deep cnn architecture for vehicle logo recognition. *IET Intelligent Transport Systems*, 12(8):939–946, 2018.

[75] Gan Kim Soon, Tan Tse Guan, Chin Kim On, Rayner Alfred, and Patricia Anthony. A comparison on the performance of crossover techniques in video game. In *2013 IEEE International Conference on Control System, Computing and Engineering*, pages 493–498, 2013.

[76] M. Srinivas and L.M. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(4):656–667, 1994.

[77] Igor Susmelj, Matthias Heller, Philipp Wirth, Jeremy Prescott, and Malte Ebner et al. Lightly. *GitHub. Note: https://github.com/lightly-ai/lightly*, 2020.

[78] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional Image Generation with PixelCNN Decoders. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[79] Xiao Wang and Guo-Jun Qi. Contrastive learning with stronger augmentations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–12, 2022.

[80] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

[81] Xihong Yang, Yue Liu, Sihang Zhou, Siwei Wang, Xinwang Liu, and En Zhu. Contrastive deep graph clustering with learnable augmentation. *arXiv preprint arXiv:2212.03559*, 2022.

[82] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. Graph contrastive learning automated. In *International Conference on Machine Learning*, pages 12121–12132. PMLR, 2021.

[83] Yingfang Yuan, Wenjun Wang, and Wei Pang. A genetic algorithm with tree-structured mutation for hyperparameter optimisation of graph neural networks. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 482–489. IEEE, 2021.

[84] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. *CoRR*, abs/1603.08511, 2016.

[85] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2021.