NASIAM: EFFICIENT REPRESENTATION LEARNING USING NEURAL ARCHITECTURE SEARCH FOR SIAMESE NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Siamese networks are one of the most trending methods to achieve unsupervised visual representation learning. Meanwhile, Neural Architecture Search (NAS) is becoming increasingly important as a technique to discover efficient deep learning architectures. In this article, we present NASiam, a novel approach that uses for the first time differentiable NAS to improve the Multilayer Perceptron projector and predictor (encoder/predictor pair) architectures inside Siamese networks frameworks while preserving the simplicity of previous baselines. We show that these new architectures allow backbone convolutional models to learn strong representations efficiently. NASiam reaches competitive performance in both small-scale (CIFAR) and large-scale (ImageNet) image classification datasets. We discuss the composition of the NAS-discovered architectures and emit hypotheses on why they manage to prevent collapsing behavior.

1 INTRODUCTION

Deep Learning (DL) has experienced rapid growth in the past few years. Two DL subfields have received much attention: Unsupervised Representation Learning and Automated Deep Learning (AutoDL).

Unsupervised representation learning aims to make DL models learn strong representations from unlabeled data. This is especially useful when considering that data labeling is often a costly and laborious human-made process. One of the most common approaches to unsupervised visual representation learning is siamese networks (Bromley et al., 1993). Siamese networks consist of two weight-sharing branches (i.e., "twins") applied to two or more inputs. The output feature vectors of the two branches are compared to compute a loss (e.g., a "contrastive" loss). In the case of representation learning, the inputs are usually data augmentations of the same image, and the siamese networks seek to maximize the similarity between the output feature vectors of the two branches (Chen et al., 2020a; 2021a; 2020b).

On the other hand, AutoDL tries to remove the human factor from the DL pipeline. Architecture design is one part of this pipeline that has proven particularly relevant to automate. Most DL architectures are handcrafted and lack the certainty of an optimal solution (Chollet, 2017; Szegedy et al., 2017; He et al., 2016). Neural Architecture Search (NAS) aims to solve this issue by using a meta-learner to search for neural network architectures relevant to a given task (e.g., image classification, semantic segmentation, or object detection). NAS algorithms efficiently browse large search spaces that would prove challenging to navigate manually. The first NAS works used reinforcement learning (Zoph & Le, 2017; Zoph et al., 2018) or evolutionary methods (Chen et al., 2019; Real et al., 2019) but proved particularly inefficient, with thousands of GPU days needed to obtain a competitive architecture (e.g., 2000 GPU days for NASNet (Zoph & Le, 2017)). Nowadays, most approaches use differentiable NAS (Chu et al., 2020; Ye et al., 2022; Dai et al., 2021) (i.e., gradient-based search process) as it requires far less computational resources and often yields better results.

This article leverages differentiable NAS to discover encoder (projector) and predictor architectures (i.e., Multilayer Perceptrons) that enable backbone Convolutional Neural Networks (CNNs) to efficiently learn strong representations from unlabeled data. To the extent of our knowledge, this is the first time that NAS has been applied to enhance the architecture of Siamese networks. Thus,

we improved the performance of several Siamese network frameworks such as SimSiam (Chen & He, 2021), SimCLR (Chen et al., 2020a), or MoCo (Chen et al., 2020b), with an encoder-predictor pair discovered by a meta-learner inspired by DARTS (Liu et al., 2019), a popular differentiable NAS method. We dubbed our approach NASiam ("Neural Architecture Search for Siamese Networks"). We show that NASiam reaches competitive results on small-scale (CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009)) and large-scale (ImageNet (Russakovsky et al., 2015)) datasets. Thus, Section 3 highlights our main following contributions:

- A novel way to design encoder/predictor pairs for Siamese Networks using Differentiable Neural Architecture Search.
- A novel search space specifically designed for the Multi-Layer Perceptron (MLP) heads of encoder/predictor pairs.

The rest of the article is structured as follows: Section 2 features a short survey on related differentiable NAS and Siamese Networks works, Section 4 presents the results of different image classification experiments and showcases a discussion on the composition of the discovered encoder/predictor pair architectures, and Section 5 brings a conclusion to our work while giving some insights about future work.

2 Related Work

This section briefly recalls related work in Differentiable Neural Architecture Search (DNAS), Siamese Networks for Representation Learning, and Neural Architecture Search for Contrastive Learning.

2.1 Differentiable Neural Architecture Search

One of the most trending differentiable NAS family of methods is derived from Differentiable ARchiTecture Search (DARTS) (Liu et al., 2019). This method uses Stochastic Gradient Descent (SGD) to optimize a set of weights (denoted α) that represent operations inside a building block (i.e., an elementary component of the network) called a "cell". A cell C can be considered a direct acyclic graph whose nodes represent states. Its edges are a mix of the K different operations $O = \{o_1, ..., o_K\}$ that define the search space S. As part of a weight-sharing mechanism, DARTS only searches for two types of cells: *normal* cells (i.e., cells that make up most of the network) and *reduction* cells (i.e., cells that perform dimension reduction). The *reduction* cells are typically positioned at the 1/3 and 2/3 of the network. As the SGD on α occurs while the supernet containing all cells is being trained on a given dataset, DARTS is practically solving a bi-level optimization problem. Moreover, the α weights are discretized through a softmax (Bridle, 1990) operation as follows:

$$\overline{o}_{i,j}(x) = \sum_{k=1}^{K} \frac{exp(\alpha_{i,j}^{k})}{\sum_{k'=1}^{K} exp(\alpha_{i,j}^{k'})} o_k(x) \tag{1}$$

where $\overline{o}_{i,j}(x)$ is the mixed output of edge $e_{i,j}$ for input feature x and $\alpha_{i,j}^{k} \in \alpha_{i,j}$ is the weight associated with operation $o_k \in O$ for $e_{i,j}$.

Several works attempted to improve on DARTS. P-DARTS (Chen et al., 2021a) significantly reduced the search time by progressively deepening the architecture when searching, leading to a better search space approximation and regularization. PC-DARTS (Xu et al., 2019) attempted to reduce DARTS' memory cost by sampling only a portion of the supernet to avoid redundancy in the search space exploration. FairDARTS (Chu et al., 2020) tried to solve two critical problems that occurred in DARTS, the over-representation of *skip* connections and the uncertainty in the probability distribution of operations. To this end, the authors used the sigmoid function rather than softmax (see Eq. 1) and crafted a novel loss function that can push α values towards 0 or 1. DARTS- (Chu et al., 2021) introduced auxiliary skip connections that are less prone to become dominant, thus ensuring a fairer competition with the other operations. β-DARTS (Ye et al., 2022) introduced a new regularization method, called *Beta-Decay*, that prevents the architectural parameters from saturating. *Beta-Decay* led to increased robustness and better generalization ability. Finally, D-DARTS (Heuillet et al., 2021) proposed a mechanism to distribute the search process to the cell level. This

approach led to the individualization of each cell, thus increasing the diversity among the candidate architectures and expanding the search space.

2.2 SIAMESE NEURAL NETWORKS

Bromley et al. (1993) first proposed the Siamese Neural Networks (SNNs) architecture as "twin" (i.e., identical and sharing the same weights) models that process two or more inputs and compare their outputs. The central intuition behind this concept is that comparing the output feature vectors will highlight the discrepancies between the inputs. Hence, this approach is advantageous in signature (Bromley et al., 1993) or face (Taigman et al., 2014) recognition applications.

Another application of SNNs is Unsupervised Representation Learning, also designated as Self-Supervised Learning (SSL). In particular, it is possible to learn representations from unlabeled data by feeding variations of the same input to twin Convolutional Neural Network (CNN) (LeCun et al., 1995) models and computing the similarity between the output feature vectors. This similarity metric is used as a loss function, leading the SNNs to learn robust representations (i.e., resisting disturbance in the input data). This process is denoted as contrastive unsupervised learning. Momentum Contrast (MoCo) (He et al., 2020) pre-trains a CNN using unsupervised learning with a momentum encoder and fine-tunes its classifier head on standard supervised linear classification. The authors of MoCo show that the unsupervised pre-trained approach can surpass standard CNN on multiple ImageNet (Russakovsky et al., 2015) computer vision tasks. SimCLR (Chen et al., 2020a) added a Multi-Layer Perceptron (MLP) head as a predictor and highlighted the critical role of strong data augmentation and large batches (e.g., 8000) in contrastive learning. Following up on this, Chen et al. (2020b) proposed an improved version of MoCo (dubbed MoCo V2) that added a two-layer MLP head in the encoder and modified the data transforms according to those of SimCLR. Bootstrap Your Own Latent (BYOL) (Grill et al., 2020) proposed an SNN framework centered around an *online* network and a *target* network. The output of the *target* network is iteratively bootstrapped to serve as input to the *online* network. The authors showed that BYOL could learn more robust representations than previous approaches. Finally, SimSiam (Chen & He, 2021) introduced a simpler SNN architecture that removes the need for negative sample pairs, momentum encoders, and large batches. More specifically, SimSiam implements a *stop-grad* mechanism that stops gradient backpropagation in one of the two branches of the twin model. Despite being a more straightforward approach than previous baselines, SimSiam reaches a competitive score on ImageNet classification.

2.3 NEURAL ARCHITECTURE SEARCH FOR CONTRASTIVE SELF-SUPERVISED LEARNING

A handful of previous works have already explored using NAS for Contrastive Self-Supervised Learning (SSL). Kaplan & Giryes (2020) first introduced a method to leverage NAS to improve existing SSL frameworks such as SimCLR (Chen et al., 2020a). Their approach, dubbed SSNAS, is derived from DARTS (Liu et al., 2019) and reached competitive performance compared with supervised models. Nguyen & Chang (2021) proposed CSNAS, a novel way to search for SSL-focused CNN architectures using Sequential Model-Based Optimization. CSNAS leverages a cell-based search space similar to DARTS (Liu et al., 2019) and performs Constrastive SSL using PIRL (Misra & van der Maaten, 2020). The authors showed that CSNAS managed to overperform or match both handcrafted architectures and supervised NAS models on image classification tasks. Another work of note is SSWP-NAS (Li et al., 2022), a proxy-free weight-preserving NAS method for SSL. Similarly to CSNAS, SSWP-NAS is based on DARTS and navigates through a cell-based search space to discover new CNN architectures. SSWP-NAS overperformed previous SSL NAS methods and reached competitive results compared to supervised NAS approaches. In a drastically different approach, Contrastive Neural Architecture Search (CTNAS) (Chen et al., 2021b) refactors NAS with Contrastive Learning. A Neural Architecture Comparator is designed to drive the search process by comparing candidate architectures with a baseline architecture. Thus, in this approach, contrary to other works, Contrastive Learning is used to enhance NAS rather than the other way around.

In this article, we propose to go further than the previous works listed above by using differentiable NAS to directly enhance the Siamese (i.e., MLP) architecture rather than improve the backbone CNN (which is similar to what trending NAS frameworks such as DARTS (Liu et al., 2019) or FBNet (Dai et al., 2021) do). In Section 4, we show that our NASiam approach is able to discover

novel Siamese architectures reaching higher performance than standard Contrastive SSL frameworks such as SimCLR (Chen et al., 2020a) or MoCo (Chen et al., 2020b).

3 PROPOSED APPROACH

This section highlights the key ideas behind our proposed approach: searching for the Multilayer Perceptron components of the encoder/predictor pair, and crafting an original search space specific to Contrastive Learning with Siamese Neural Networks.

3.1 SEARCHING FOR AN ENCODER/PREDICTOR PAIR

First, we focused on SimSiam (Chen & He, 2021) as a simple baseline upon which to build our approach. SimSiam uses a Siamese Neural Network architecture consisting of an encoder f and a predictor h. The encoder f is composed of a baseline CNN (e.g., ResNet50 (He et al., 2016)) and of a projector head (i.e., a three-layer MLP) that is duplicated on twin branches that take variations of the same image as input. A two-layer MLP h is then added on top of one of the branches to act as a predictor head. The discrepancy between the output feature vectors of the two branches is computed using a contrastive loss as follows:

$$\mathcal{L} = \frac{1}{2}(\mathcal{D}(p_1, \texttt{stopgrad}(z_2)) + \mathcal{D}(p_2, \texttt{stopgrad}(z_1)) \tag{2}$$

where $z_1 = f(x_1)$, $z_2 = f(x_2)$, $p_1 = h(z_1)$, $p_2 = h(z_2)$ for input images x_1 and x_2, `stopgrad` is a mechanism that stops gradient backpropagation, and $\mathcal{D}$ is the negative cosine similarity defined as follows:

$$\mathcal{D}(p, z) = -\frac{p}{||p||_2}.\frac{z}{||z||_2} \tag{3}$$

where $||.||_2$ is the l_2 norm.

In our proposed approach, we kept most of the global structure of the underlying Siamese framework. However, we used used a DARTS-like (Liu et al., 2019) NAS method to search for an encoder projector head architecture up to n layers and a predictor architecture up to m layers. More specifically, we consider a set $O = \{o_1, ..., o_K\}$ of candidate operations. We search for two cells (see Section 2.1) C_e and C_p in which each layer is a mix of $|O| = K$ operations weighted by sets of parameters denoted respectively α_e and α_p. Similarly to Eq. 1, operation values in each layer are discretized as follows:

$$\overline{o}_i(x) = \sum_{k=1}^{K} \sigma_{SM}(\alpha_i^k) o_k(x) \tag{4}$$

where $\overline{o}_i$ is the mixed operation of layer i, α_i^k is the architectural weight assigned to $o_k \in O$ for layer i, and σ_{SM} denotes the *softmax* operation. The supernet encompassing f and h is trained on a portion of a dataset while C_e and C_p are simultaneously searched on another portion of the same dataset. Hence, we solve a bi-level optimization problem. Once the search phase is complete, for each layer i of each cell, we select the top operation according to the discretized weights α_e and α_p to form the encoder/predictor architecture genotype G. This operation parsing process is detailed in Algorithm 1.

Our approach, dubbed NASiam (Neural Architecture Search for Siamese Networks), is summarized in Fig. 1. In addition to SimSiam, we also experimented NASiam on other Siamese frameworks such as SimCLR(Chen et al., 2020a), MoCo V2 (Chen et al., 2020b), and BYOL (Grill et al., 2020). However, some frameworks (SimCLR and MoCo V2) do not rely on a predictor. Hence, in that case, we only performed NAS for the MLP head of the encoder (i.e., only searching for cell C_e).

In Section 4, we show that NASiam can consistently improve the performance of popular Siamese frameworks (SimSiam, SimCLR, MoCo, and BYOL) in both small-scale (CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009)) and large-scale (ImageNet (Russakovsky et al., 2015)) image classification datasets.

Algorithm 1 Algorithm describing the operation parsing process of NASiam

Require: List: C, list of layers containing weights
Require: Boolean: $edge$, whether to select 2 operations instead of a single one
Require: List: O, list of operations
  $C_f \leftarrow$ empty_list()
  $n \leftarrow |C|$
  for i in $[0, n]$ **do**
    $g \leftarrow$ empty_list()
    $ops \leftarrow$ sort($C[i]$)
    append($ops[-1], g$)
    if $edge$ **then**
      append($ops[-2], g$)
    end if
    append(g, C_f)
  end for
  return C_f

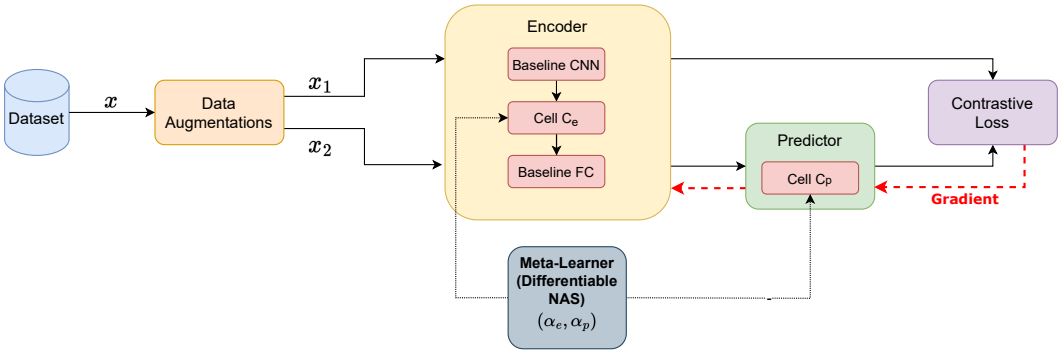

Figure 1: Layout of the NASiam architecture. Siamese networks encoder/predictor (projection MLPs) architectures are searched using differentiable NAS wrapped around a Siamese framework such as SimSiam (Chen et al., 2020a) which is the baseline used in the present figure.

3.2 Crafting a Constrastive Learning-Specific Search Space

To accompany our novel NASiam approach (see Section 3.1), we crafted an original search space S specifically designed for MLPs. S comprises the following 4 operation blocks: `linear` + `batch_norm` + `ReLU`, `max_pool_3x3` (1-dimensional) + `batch_norm`, `avg_pool_3x3` (1-dimensional) + `batch_norm`, `identity`.

While unconventional, including pooling layers in the search space is helpful, as we show in Section 4 that they can help prevent collapsing. Moreover, the authors of SimSiam (Chen & He, 2021) indicated that insufficient or too many Batch Normalization (BN) layers could cause the model to underperform severely or become unstable. They empirically demonstrate that the optimal setting for SimSiam is to place BNs after every layer except for the predictor's output layer. Hence, we follow this assertion by adding BNs after every `linear` and `pooling` operation except for the predictor's final layer. In addition, we also included the standard ReLU (Dahl et al., 2013) activation function after BN for `linear` operations. Finally, we also included the `identity` operation so that the search algorithm can modulate the number of layers in the architecture. This way, we can indicate a maximum number of layers n, and the search algorithm can craft an architecture of size $m < n$ by "skipping" layers.

4 Experiments

This section presents the results of our image classification experiments on small-scale (CIFAR-10, CIFAR-100), and large-scale (ImageNet) datasets.

4.1 EXPERIMENTAL SETTINGS

We used RTX 3090 and Tesla V100 Nvidia GPUs to conduct our experiments. We searched for predictor/encoder pairs for 100 epochs on CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009) using the `SGD` optimizer with $lr = 0.03$ and $wd = 5e - 4$. We set a maximum of 6 layers for the encoder. If the baseline Siamese framework relies on a predictor, we search for a 4-layer predictor architecture. The whole search process on these settings takes around 1 GPU day on a single GPU. We did not search directly on ImageNet (Russakovsky et al., 2015) as it is prohibitively expensive (i.e., it takes around 12 GPU days on a single GPU). For the pre-training and linear classification phases, we kept the same settings as Chen & He (2021). Our code is based on PyTorch 1.12.

4.2 ABLATION STUDY ON THE IMPORTANCE OF POOLING LAYERS

We conducted an ablation study on the importance of including pooling layers in our novel space search S (see Section 3.2). To this end, we simply removed `max_pool_3x3` and `avg_pool_3x3` from S to form S'. When comparing the results in Table 1, we can observe that, when searching on S' rather than S, the validation top-1 accuracy of NASiam drops significantly (by around 3 %). Moreover, when analyzing the genotypes searched on CIFAR-10 and CIFAR-100 using S, it appears that the predictor architectures always contain pooling layers (making up to 40 % of the total architecture). In addition, Fig. 2 shows that the model (see Eq. 2) achieved better similarity and faster convergence when searched on S rather than S'. Thus, these findings highlight the critical role pooling layers play in ensuring high performance and preventing collapse, especially concerning the encoder architecture.

Table 1: Results on CIFAR-10 linear classification of two NASiam models using search space S and S' respectively. Both models were pre-trained for 100 epochs. The baseline framework is SimSiam with a ResNet18 backbone.

Search Space	Search Epochs	Pre-Train Epochs	Validation Top-1 (%)	Validation Top-5
S	50	100	**66.6**	91.3
S'	50	100	63.7	86.1

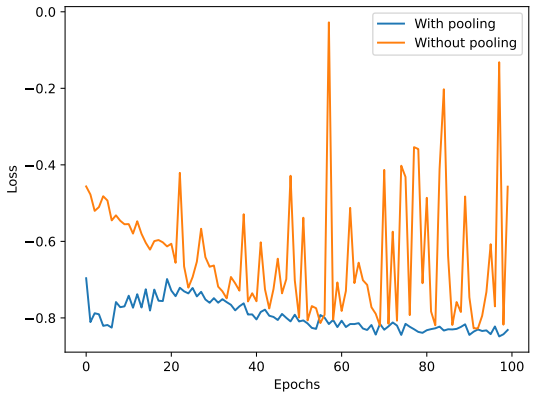

Figure 2: Plot of the negative cosine contrastive loss while pretraining two NASiam models on CIFAR-10. The baseline framework is SimSiam with a ResNet18 backbone. The two models are searched on search spaces S (blue line) and S' (red line) respectively. The model searched on S achieves better similarity thus making the relevance of pooling layers clear.

4.3 PRELIMINARY RESULTS ON CIFAR

To quickly assess the behavior of our novel approach NASiam, we first conducted preliminary experiments on small-scale CIFAR datasets (Krizhevsky et al., 2009). We searched NASiam architectures for 50 epochs on CIFAR-10 and CIFAR-100 using the CIFAR version of ResNet18 (He et al., 2016) as the encoder backbone. Then, we performed unsupervised pretraining for 800 epochs with a cosine annealing schedule before training a linear classifier using frozen features for 100 epochs. In these settings, NASiam overperforms SimSiam by 1.5 % and 0.3 % on CIFAR-10 and CIFAR-100 respectively (see Table 2 and Table 3). In addition, Fig. 3 shows us that NASiam can achieve better similarity than SimSiam without saturating the contrastive loss to -1 (i.e., a "collapsing" behavior). Furthermore, results were also positive when using alternative Siamese frameworks with NASiam overperforming both MoCo V2 (Chen et al., 2020b) and SimCLR (Chen et al., 2020a).

Table 2: Results of pre-training for 800 epochs on CIFAR-10 linear classification with SGD. The backbone is the CIFAR version of ResNet18. †: Result obtained by running the official implementation with the hyperparameters suggested by the authors for CIFAR-10.

	Model	Batch Size	Pre-Train Epochs	Train Epochs	Validation Top-1 (%)
	MoCo V2 (Chen et al., 2020b)†	256	800	100	89.8
	SimCLR (Chen et al., 2020a)†	256	800	100	91.1
	SimSiam (Chen & He, 2021)†	256	800	100	89.5
	NASiam (SimSiam)	256	800	100	**91.0**
Ours	NASiam (MoCo V2)	256	800	100	**90.4**
	NASiam (SimCLR)	256	800	100	**91.8**

Table 3: Results of training for 800 epochs on CIFAR-100 linear classification with SGD. The backbone is the CIFAR version of ResNet18. †: Result obtained by running the official implementation with the hyperparameters suggested by the authors for CIFAR-10.

	Model	Batch Size	Pre-Train Epochs	Train Epochs	Validation Top-1 (%)
	MoCo V2 (Chen et al., 2020b)†	256	800	100	62.9
	SimCLR (Chen et al., 2020a)†	256	800	100	63.6
	SimSiam (Chen & He, 2021)†	256	800	100	63.7
	NASiam (SimSiam)	256	800	100	**64.1**
Ours	NASiam (MoCo V2)	256	800	100	**64.3**
	NASiam (SimCLR)	256	800	100	**68.2**

4.4 RESULTS ON IMAGENET

We conducted image classification experiments on ImageNet (Russakovsky et al., 2015) as a standard practice to evaluate the performance of our novel approach on large-scale datasets. As stated in Section 4.1, it is prohibitively expensive to search directly on ImageNet. Hence, we searched for an encoder/predictor pair for 100 epochs on CIFAR-100 using ResNet50 as the encoder backbone. Then, we performed unsupervised pretraining on ImageNet for 100 epochs before training a linear classifier with frozen features for 100 epochs. The results are presented in detail in Table 4.

4.5 DISCUSSION ON THE COMPOSITION OF THE ARCHITECTURES

Some facts are noteworthy when comparing encoder/predictor architectures discovered by our novel approach (see Section 3.1) with those of SimSiam (Chen & He, 2021). First, in Fig. 4, we can see that the ResNet50 CIFAR-100 (Krizhevsky et al., 2009) NASiam architectures retained the simplicity of SimSiam by having selected only a single `Linear-BN-ReLU` block to form the encoder architecture (instead of two). The predictor architecture is also similar to the one in SimSiam but

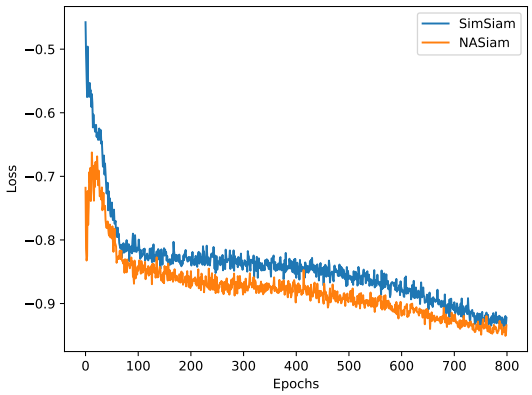

Figure 3: Plot of the negative cosine constrastive loss when pretraining SimSiam and NASiam for 800 epochs on CIFAR-10. NASiam converges faster without collapsing and achieves better similarity than SimSiam.

Table 4: Results of training for 100 epochs on ImageNet linear classification with SGD. The backbone is ResNet50. Models were pre-trained for 100 epochs on ImageNet. †: Score obtained by the authors of SimSiam by using an improved version of the model.

Model		Batch Size	Pre-Train Epochs	Train Epochs	Validation Top-1 (%)
MoCo V2 (Chen et al., 2020b)†		256	100	100	67.4
BYOL (Grill et al., 2020)†		4096	100	100	66.5
SimCLR (Chen et al., 2020a)†		4096	100	100	66.5
SimSiam (Chen & He, 2021)		256	100	100	67.1
	NASiam (SimSiam)	256	100	100	**67.4**
Ours	NASiam (SimCLR)	4096	100	100	**67.2**

features an additional `AvgPool3x3-BN` block. However, the architecture discovered for ResNet18 (CIFAR version, see Section 4.1) is much deeper with 6 `Linear-BN-ReLU` blocks for the encoder architecture and 2 `AvgPool3x3-BN` blocks intertwined between 2 `Linear-BN-ReLU` blocks for the predictor architecture. One hypothesis to explain this discrepancy in architectural sparsity is that ResNet18, being a shallower model than ResNet50, has a less powerful innate ability to learn robust representations. Thus, the encoder/predictor pair play a more decisive role in representation learning.

Interestingly, this assumption could also explain why the authors of SimSiam succeeded with such a simple approach: the intrinsic representation learning ability of ResNet50 is sufficient to learn robust representations without the help of additional tricks such as negative pairs or momentum encoders. To confirm this hypothesis, we tried to fit a ResNet50 model on CIFAR-10 with the deeper encoder/predictor pair discovered for ResNet18. Fig. 5 clearly shows that this architectural setting quickly led to a collapsing behavior (with the contrastive loss rapidly saturating to -1 as soon as epoch 3) with a higher variance than for the ResNet50-searched architecture.

5 Conclusion

In this article, we presented NASiam, a novel approach for Contrastive Learning with Siamese Networks that searches for efficient encoder/predictor pairs using differentiable Neural Architecture Search (see Section 3.1). This universal method can enhance many existing Siamese frameworks while preserving their underlying structure. Section 4 showed that NASiam discovers encoder/predictor architectures that efficiently learn robust representations and overperform previous baselines in small-scale and large-scale image classification datasets. Hence, we demonstrated the

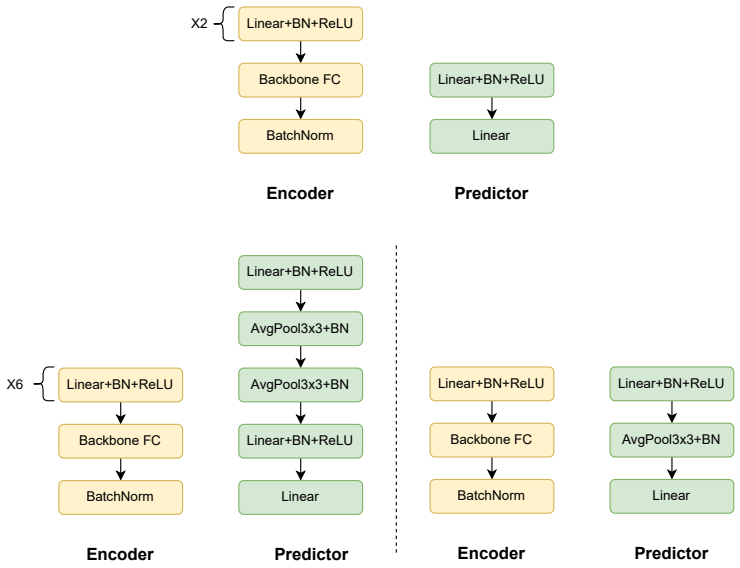

Figure 4: Composition of encoder/predictor pair architectures. (**Top**) SimSiam model. (**Bottom left**) NASiam model searched for 50 epochs on CIFAR-10 using SimSiam as baseline framework with ResNet18 as backbone. (**Bottom right**) NASiam model searched for 50 epochs on CIFAR-100 using SimSiam as baseline framework with ResNet50 as backbone. ResNet18-searched and ResNet50-searched architectures are clearly different with ResNet18 needing both a deeper encoder and a deeper predictor.

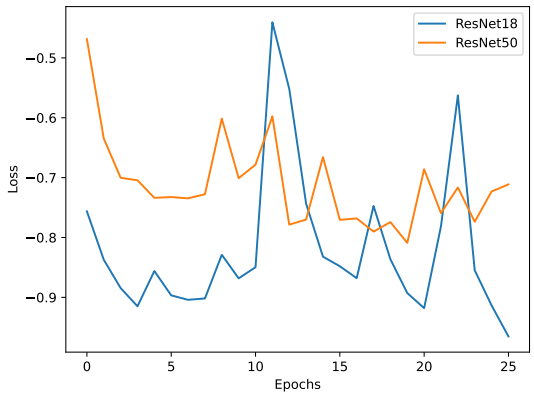

Figure 5: Plot of the negative cosine similarity loss while pretraining NASiam with a ResNet50 using encoder/pedictor pair architectures searched either with ResNet18 or ResNet50 as backbone. The ResNet18-searched architecture quickly collapses and has higher variance than the ResNet50-searched one, hence confirming the better intrinsic representation learning ability of ResNet50.

relevance of applying NAS to MLP-headed Siamese Networks and hope this work will pave the way to further improvements in Contrastive Learning.

REFERENCES

John S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In Françoise Fogelman Soulié and Jeanny Hérault (eds.), *Neurocomputing*, pp. 227–236, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. *Advances in neural information processing*

systems, 6, 1993.

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

Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive darts: Bridging the optimization gap for nas in the wild. *International Journal of Computer Vision*, 129(3):638–655, 2021a.

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

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

Yaofo Chen, Yong Guo, Qi Chen, Minli Li, Wei Zeng, Yaowei Wang, and Mingkui Tan. Contrastive neural architecture search with neural architecture comparators. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9502–9511, 2021b.

Yukang Chen, Gaofeng Meng, Qian Zhang, Shiming Xiang, Chang Huang, Lisen Mu, and Xinggang Wang. Renas: Reinforced evolutionary neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4787–4796, 2019.

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair darts: Eliminating unfair advantages in differentiable architecture search. In *European conference on computer vision*, pp. 465–480. Springer, 2020.

Xiangxiang Chu, Xiaoxing Wang, Bo Zhang, Shun Lu, Xiaolin Wei, and Junchi Yan. Darts-: Robustly stepping out of performance collapse without indicators. In *International Conference on Learning Representations*, 2021.

George E Dahl, Tara N Sainath, and Geoffrey E Hinton. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 8609–8613. IEEE, 2013.

Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Bichen Wu, Zijian He, Zhen Wei, Kan Chen, Yuandong Tian, Matthew Yu, Peter Vajda, et al. Fbnetv3: Joint architecture-recipe search using predictor pretraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16276–16285, 2021.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

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

Alexandre Heuillet, Hedi Tabia, Hichem Arioui, and Kamal Youcef-Toumi. D-darts: Distributed differentiable architecture search. *arXiv preprint arXiv:2108.09306*, 2021.

Sapir Kaplan and Raja Giryes. Self-supervised neural architecture search. *arXiv preprint arXiv:2007.01500*, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Zhuowei Li, Yibo Gao, Zhenzhou Zha, Zhiqiang Hu, Qing Xia, Shaoting Zhang, and Dimitris N Metaxas. Towards self-supervised and weight-preserving neural architecture search. *arXiv preprint arXiv:2206.04125*, 2022.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

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

Nam Nguyen and J Morris Chang. Csnas: Contrastive self-supervised learning neural architecture search via sequential model-based optimization. *IEEE Transactions on Artificial Intelligence*, 3 (4):609–624, 2021.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pp. 4780–4789, 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1701–1708, 2014.

Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*, 2019.

Peng Ye, Baopu Li, Yikang Li, Tao Chen, Jiayuan Fan, and Wanli Ouyang. β-darts: Beta-decay regularization for differentiable architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 8697–8710, 2018.

A Appendix

A.1 Object Detection and Instance Segmentation Results on COCO

Table 5 displays the results of transferring Siamese models pretrained on ImageNet (Russakovsky et al., 2015) to Microsoft COCO (Lin et al., 2014) object detection and instance segmentation tasks.

Table 5: Comparison of backbone models for MaskRCNN (He et al., 2017) on COCO (Lin et al., 2014) using a 1x schedule and ResNet50 (He et al., 2016) as the baseline CNN. All models are pretrained for 200 epochs on ImageNet, finetuned for 12 epochs on COCO 2017 train, and evaluated on COCO 2017 val.

Models	AP_{50} (%)	AP (%)	AP_{75} (%)	AP^{mask}_{50} (%)	AP^{mask} (%)	AP^{mask}_{75} (%)
ImageNet supervised	58.2	38.2	41.2	54.7	33.3	35.2
SimCLR	57.7	37.9	40.9	54.6	33.3	35.3
SimSiam	57.5	37.9	40.9	54.2	33.2	35.2
MoCo V2	58.8	39.2	42.5	55.5	34.3	36.6
BYOL	57.8	37.9	40.9	54.3	33.2	35.0
NaSiam (SimSiam)	**58.6**	**39.0**	**42.1**	**55.2**	**34.1**	**36.3**