Unifying Synergies between Self-supervised Learning and Dynamic Computation

Tarun Krishna¹,³

Ayush K. Rai¹,²

Alexandru Drimbarean⁴

Eric Arazo¹,³,⁵

Paul Albert¹,³,⁵

Alan F. Smeaton¹,²

Kevin McGuinness¹,³

Noel E. O’Connor¹,³

tarun.krishna2@mail.dcu.ie

ayush.rai3@mail.dcu.ie

Alexandru.Drimbarean@xperi.com

eric.arazo@insight-centre.org

paul.albert@insight-centre.org

alan.smeaton@dcu.ie

kevin.mcguinness@dcu.ie

noel.oconnor@dcu.ie

¹ Insight Centre for Data Analytics, Dublin City University, Dublin, Ireland
² School of Computing, Dublin City University, Dublin, Ireland
³ School of Electronic Engineering, Dublin City University, Dublin, Ireland
⁴ Xperi Corporation, Galway, Ireland
⁵ Equal Contribution

Abstract

Computationally expensive training strategies make self-supervised learning (SSL) impractical for resource-constrained industrial settings. Techniques like knowledge distillation (KD), dynamic computation (DC), and pruning are often used to obtain a lightweight model, which usually involves multiple epochs of fine-tuning (or distilling steps) of a large pre-trained model, making it more computationally challenging. In this work we present a novel perspective on the interplay between the SSL and DC paradigms. In particular, we show that it is feasible to simultaneously learn a dense and gated sub-network from scratch in an SSL setting without any additional fine-tuning or pruning steps. The co-evolution during pre-training of both dense and gated encoder offers a good accuracy-efficiency trade-off and therefore yields a generic and multi-purpose architecture for application-specific industrial settings. Extensive experiments on several image classification benchmarks including CIFAR-10/100, STL-10 and ImageNet-100, demonstrate that the proposed training strategy provides a dense and corresponding gated sub-network that achieves on-par performance compared with the vanilla self-supervised setting, but at a significant reduction in computation in terms of FLOPs, under a range of target budgets ($t_d$).
1 Introduction

Motivation. Self-supervised representation learning methods [4, 7, 11, 12, 14] are the standard approach for training large scale deep neural networks (DNNs). One of the main reasons for their popularity is their capability to leverage the inherent structure of data from a vast unlabeled corpus during pre-training, which makes them highly suitable for transfer learning [28]. However, this comes at the cost of substantially larger model size, computationally expensive training strategies (larger training times, large batch-sizes, etc.) [13, 28] and subsequently more expensive inference times. Though such strategies are effective for achieving state-of-the-art results in computer vision, they may not be practical in resource-constrained industrial settings that require lightweight models to be deployed on edge devices.

To lessen the computational burden, it is common to extract (or learn) a lightweight network from an off-the-shelf pre-trained model. This has been successfully achieved through techniques such as knowledge distillation (KD) [35], pruning [24], dynamic computation (DC) [58], etc. KD methods follow a standard two-step procedure of pre-training and distilling knowledge into a student network using self-supervised (SS) objective [1, 21, 51] or by together incorporating supervised and SS objectives [2], while pruning based approaches heavily rely on multiple steps of pre-train $\rightarrow$ prune $\leftrightarrow$ finetune to get a lightweight network irrespective of the objective, whereas methods based on dynamic/conditional computation [52, 58] again rely on a pre-trained model to obtain a lightweight network while keeping the network topology intact via a gating mechanism. These approaches are effective but using fine-tuning to obtain a sub-network from large pre-trained models (such as Large Language Models) can be computationally expensive and cumbersome. Also, since downstream tasks are diverse and vary widely, any change in the task requires repeating the entire procedure multiple times, making it inefficient and less transferable.

Research Questions. These limitations motivate us to ask the following question: “Can we unify the learning of a lightweight sub-network along with a dense network from scratch and in a completely self-supervised fashion?” A straightforward way to achieve this is via an online KD (with self-supervised objective) [4, 2] learning paradigm which involves training teacher ($f_{\theta}$) and student ($g_{\phi}$) networks simultaneously during a pre-training stage. Recognising that this adds to the computational burden during pre-training (extra $g_{\phi}$), we adopt a different route to attain the same goal but with a simpler, efficient pre-training objective and faster inference than online KD-based methods.

This enables us to reformulate the research question to: “Can we learn a single encoder (function) that could serve the dual purpose of being used as a dense and lightweight network with minimal additional overhead?”

Our objective is to simultaneously learn a dense and a lightweight model through a unified pre-training procedure to maintain high performance on the downstream task. We achieve this by exploiting the Siamese setting (a common setting for SSL [3]) combined with a gating mechanism for dynamic channel selection (DCS) [47, 58]. We opted for dynamic channel selection over KD/pruning for two main reasons: first, the gating mechanism preserves the networks topology adding enough flexibility to the approach; second, these gating modules are computationally inexpensive. For the self-supervised objective we choose VI-CReg [4] due to its symmetric nature and its ability to regularise each branch independently as dense and sparse branches will have different statistics. Figure 1 (left) demonstrates this dual setting of obtaining a dense and a lightweight network (derived from the dense one).

It should be noted that in this paper we do not follow the vocabulary of student-teacher
Figure 1: 1. Illustrates the unification of SSL and DC during the pre-training and testing (inference) phase. \( f_d \) and \( f_l \) denotes respective linear layer for dense and gated network respectively. **Note:** dimensional size of \( f_d \) and \( f_l \) is the same, while in the figure this may look otherwise but is done to depict the fact that empirically, the dimension of \( f_l \) is less than \( f_d \). 2. Shows the modification of ResNet-18 basic block to accommodate the gating network during inference. 3. Describes the gating module which comprises a gating network and sampling module.

networks. Instead we restrict the terminology to lightweight (gated)\(^1\) and a dense network, where encoder and gates are randomly initialised and trained jointly from scratch, with the aim that they co-evolve during pre-training. It is, however, important to mention that in this work we are not proposing any new KD objective or KD-induced learning algorithm nor any new DC objective or any pruning-based learning. Instead we provide a novel perspective on exploiting the synergies that exist between self-supervised representation learning and dynamic computing. This approach is easily extendable to other symmetric-twins like Barlow-Twins \([67]\), SimCLR \([12]\) or W-MSE \([67]\), while it may require some adjustments for non-symmetric methods like BYOL \([29]\), MoCo \([32]\), etc., which will be explored in future work. Our main contributions are:

- We present a novel perspective of unifying the learning of dense and lightweight networks by exploiting a symmetric joint embedding architecture of the SSL paradigm.

- We demonstrate that a single encoder can be exploited as a dense as well as a lightweight network; we show in Table 1 and Table 2 that a single base encoder can serve this dual purpose. This not only reduces computational overhead during training but also gives enough flexibility to use a single network and exploit it as per its requirement.

- We demonstrate exhaustively, through experiments that this unification preserves fea-

\(^1\)we use lightweight and gated networks interchangeably.
ture quality across different experimental settings and gives on-par performance when compared with strict baselines (Section 4).

2 Background and Related Work

Self-supervised representation learning: SSL methods can be broadly divided into contrastive and non-contrastive techniques in the current scenario. At the core of these approaches is the concept of learning joint embedding representations realised via a Siamese architecture through instance discrimination. Contrastive learning (CL) [12, 32, 33, 46, 52, 56, 61] based approaches intuitively try to bring similar instances closer while contrasting them with negative samples. Non-contrastive techniques include clustering methods [2, 7, 8], which alternate between cluster assignment and predicting clusters as pseudo labels. Approaches such as BYOL [29], OBoW [27], MoCo [32], and SimSiam [15] use a teacher-student approach to learn joint representations. While approaches like Barlow Twins [67], W-MSE [20], and VICReg [4] follow a more principled approach of information maximisation. We use VICReg for SSL because it can regulate each branch independently, which is more effective when each branch has different statistics.

Dynamic Computation: DC is a resource-efficient mechanism that reduces model complexity by skipping unimportant parts of the network while preserving the network’s topology. Several authors including [23, 40, 48, 59, 60] have proposed adding decision branches to different layers of convolutional neural networks (CNN) for learning early exiting strategies leading to faster inference. BlockDrop [62] and SpotTune [30] learn a policy network to adaptively route the inference path through fine-tuned or pre-trained layers. ConvNet-AIG [34, 58] introduced a network that adaptively selects specific layers of importance to execute depending on the input image by specifying a target rate of each layer. GaterNet [16] introduced a network to generate input-dependent binary gates to select filters in the backbone network. DGNet [47] proposed a dual gating mechanism to induce sparsity along spatial and channel dimensions. Furthermore, dynamic channel pruning methods have also been devised such as feature boosting and suppression (FBS) [14] to dynamically amplify and suppress output channels computed by CNN layers. Other works learn sparsity through a three-stage pipeline: pretrain-prune-finetune as in [57] or use pre-trained models. See [36] for a more detailed explanation of sparsity, pruning and dynamic computing.

Self-supervised dynamic computation and beyond: Most of the works on dynamic computation have been confined to supervised learning. Recently, [44] used SimSiam [14] as a self-supervised objective combined with a dynamic channel gating (DGNet) [47] mechanism and showed that comparable performance can be achieved under channel budget constraints. Likewise [50] used a channel gating-based dynamic pruning (CGNet) [39] augmented with CL to achieve inference speed-ups without substantial loss of performance. In a similar line of work, [11] used iterative magnitude pruning (IMP) to obtain a winning ticket [22] for a pre-trained task (self-supervised objective) and evaluated its performance on various downstream tasks. [13] extended the work done in [11] in a MoCo (pre-trained) setting augmented with ADMM [68] for systematic pruning. A self-supervised loss objective can serve as a tool for KD [35] and model compression (MC). [31] used a contrastive objective (along with a supervised loss for task specific distillation) to train a student network from a pre-trained network. Similar to [31] but in a completely self-supervised setting [1, 22] minimises the KL-divergence between the distribution of similarities for the teacher (pre-trained) and student networks, while SimReg [51] minimises the regression loss. The authors in [63] used...
a two-step strategy to train a teacher (with labels and then using an SSL head with a fixed backbone) followed by training a student using a KD loss. However, we follow a more simplistic approach through the unification of SSL (VICReg) and DCS, where DCS maintains the network topology, making fine-tuning easier on different downstream tasks, unlike other methods that make network structure irreversible.

3 Preliminaries and Setup

1.VICReg as SSL objective: VICReg [4] learns a joint embedding space governed by a loss objective, which consists of invariance ($s$) (mean squared error (MSE)), variance ($v$) and co-variance ($c$), depicted Equation 1. Let us consider some image dataset $X = \{x_i\}_{i=1}^{D}$ and a set of transformations $T$ (refer to supplementary material for details). An anchor image $x_i \in \mathbb{R}^{H \times W \times 3}$ is augmented through transformations $t_1, t_2 \sim T$ to get $x_1^i = t_1(x_i)$ and $x_2^i = t_2(x_i)$ respectively. Augmented views are encoded through $f_\theta$ (ResNet-18 [31] (R18) in this study) to get feature representations. Furthermore, these representations are mapped to an embedding space via expander ($g_\phi$) where the final VICReg loss is applied between the embedding vectors $z_1^i = g_\phi(f_\theta(x_1^i))$ and $z_2^i = g_\phi(f_\theta(x_2^i))$. Formally the loss is defined on a batch of embedding vectors $Z^1 = [z_1^1, \ldots, z_1^|B|$ and $Z^2 = [z_2^1, \ldots, z_2^|B|$ as:

$$L_{VICReg}(Z^1, Z^2) = \frac{\mu}{2} [v(Z_1^1) + v(Z_2^2)] + \frac{\nu}{2} [c(Z_1^1) + c(Z_2^2)] + \eta_s(Z_1^1, Z_2^2),$$

where $\mu = 25$, $\nu = 25$ and $\eta = 1.0$. For detailed description of Equation 1 refer to [4].

2. Gating for channel selection. The gating module comprises of a gating network [5, 13] and a sampling module [13] (Figure 1). The gating network can be thought of as a lightweight network that decides the relevance of channels referred to as importance vector. To enable a lightweight design of the gating network ($\mathcal{W}$) we follow the squeeze and excitation block design [8], similar to [13, 14]. This usually requires obtaining a context vector $z \in \mathbb{R}^{C_L-1}$ via global average pooling to accumulate spatial information. This context vector $z$ is processed through $\mathcal{W}$ to get relevance scores for each channel:

$$\mathcal{W}(z, \omega) = w_2 \ast \text{BatchNorm}(w_1 \ast z) \text{ReLU}, \quad \{w_1, w_2\} \in \omega$$

where $\ast$ denotes convolution, $w_1 \in \mathbb{R}^{C_L-1 \times C_L \times 1 \times 1}$, $w_2 \in \mathbb{R}^{C_L \times C_L \times 1 \times 1}$ and $r$ is defined as reduction rate (set to 4).

Finally, to make a selection over a subset of a channels, we need to map the output of $\mathcal{W}$ to a binary vector (or mask $m \in \mathbb{R}^{C_L}$). This discrete selection works perfectly during inference but breaks the computational graph during training. To make training possible, the sampling module utilises the Gumbel-softmax reparameterisation trick [22] to make this discrete selection without breaking the computational graph. Figure 1 (right 2) shows the modification of ResNet18 basic block (during inference).

In this work we follow the setting of DGNet [13] for channel selection where sparsity is induced by setting a global target budget ($t_d$) to optimise a loss objective:

$$L_G = \lambda \left( \frac{\sum_{l=1}^{L} F_l^R}{\sum_{l=1}^{L} F_l^O} - t_d \right)^2$$
where $F_l^R$ is the average FLOPs over the batch along with FLOPs contribution from the gating network $\mathcal{W}$ (which is fixed for each layer), while $F_l^O$ is the original FLOPs without a gating module, $\lambda = 5$ across all datasets and training regimes. Only blocks with gating modules take part in FLOP computation as they contribute to the sparsification of the network. We refer our approach as VICReg-Dual-Gating (VDG).

3.1 Experimental Setup and Implementation Details

1. Pre-training: We closely follow the implementation of VICReg [4] (as our self-supervised objective) suited with our computational constraints using solo-learn library [19], while for dynamic gating we follow DGNet [47] for inducing channel sparsity via gating mechanism with ResNet18. The unified framework is depicted in Figure 1: the two branches have a separate batch normalisation layer following [66]. The encoder and gating networks are randomly initialised and trained with SGD for 500 epochs with a batch size of 512 on 2 Nvidia 2080Ti GPUs, with a warmup start of 10 epochs following a cosine decay with a base learning rate of 0.3 using the LARS optimizer [65]. Since we are using a very lightweight model as our gating network, there is no significant computational overhead during training, to be precise there is slight increase in computation which amounts to 2.11% of extra model parameters (0.013% of FLOPs computation). We pre-train for a target budget, $t_d = \{10\%,30\%,50\%\}$ for each of the datasets except for ImageNet-100 where $t_d$ is restricted to $\{30\%,50\%\}$ due to computational constraints. We report the inference speedup in terms of a hardware-independent theoretical metric of FLOPs and not wall-clock time as we do not avail any hardware accelerators to utilise sparsity during training. Any scaling parameter in the loss term is derived from the respective paper. The code is available at https://github.com/KrishnaTarun/Unification.

2. Evaluation: Pre-training and evaluation is carried on the train and validation data of CIFAR-10/100 [45], ImageNet100 and STL-10 [18]. For pre-training with STL-10 we considered only the un-labelled set. We follow the standard practice of evaluating the trained encoder by freezing its weights and training a linear classifier on top of it. We trained a single linear layer for 100 epochs with a batch size of 512 on a single NVIDIA 2080Ti with a learning rate of 0.3 following step decay of 0.1 at 60th and 80th epochs. We report top-1 accuracy averaged over 5 runs.

3. Baselines: To exhaustively compare the performance of the dense and gated models we consider VICReg [4] as a SSL dense baseline while VICReg augmented with sparsity loss $L_G$ (following Krishna et al. [44]) serves as a gated baseline, here goal is to train a lightweight gated encoder from scratch with a self-supervised objective.

4. Comparison with self-supervised KD: In order to make a fair assessment of this unification, we compare the gated network’s performance with KD based methods specifically SEED [22] and SimReg [51] where the former was proposed to distill representational knowledge into a smaller network while the latter showed that a simple regression objective can serve as an effective tool for knowledge transfer. For SimReg: During pre-training (SSL) (500 epoch) we used VICReg [4] with a ResNet-18 (R18: Teacher) as base encoder trained on CIFAR-100 and ImageNet-100 ($\mathcal{D}_{Pretrain}$), while distillation is performed using the SimReg objective on $\mathcal{D}_{Target}$ ($= \mathcal{D}_{Pretrain}$) by further training it for another 130 epochs. For SEED: We use a MoCo-v2 pre-trained ResNet50 (R50) encoder trained on ImageNet-

\[\text{https://www.kaggle.com/datasets/ambityga/imagenet100}\]
\[\text{https://github.com/facebookresearch/moco}\]
Table 1: Linear Evaluation: ↑/↓ in orange font is comparison with Baseline-1, while blue font is comparison with Baseline-2. FLOPs R. denotes FLOP reduction. We report Top-1 accuracy averaged over 5 runs. Best viewed in color.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>VICReg</th>
<th>VICReg-Gating Baseline-2</th>
<th>VICReg-Dual-Gating this work</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dense</td>
<td>Gated</td>
<td>FLOPs R.</td>
</tr>
<tr>
<td></td>
<td>t_d(%)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>91.11 ± 0.05</td>
<td>7.03E8</td>
<td>88.99 ± 0.04 (1.212)</td>
</tr>
<tr>
<td></td>
<td>10%</td>
<td>87.75 ± 0.03</td>
<td>88.94 ± 0.06 (2.17)</td>
</tr>
<tr>
<td></td>
<td>30%</td>
<td>89.49 ± 0.04</td>
<td>90.27 ± 0.05 (1.70)</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>90.70 ± 0.12</td>
<td>90.40 ± 0.10 (0.30)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STL-10</td>
<td>86.15 ± 0.10</td>
<td>3.33E8</td>
<td>84.29 ± 0.21 (1.30)</td>
</tr>
<tr>
<td></td>
<td>10%</td>
<td>82.48 ± 0.15</td>
<td>83.29 ± 0.20 (2.39)</td>
</tr>
<tr>
<td></td>
<td>30%</td>
<td>84.16 ± 0.11</td>
<td>84.85 ± 0.14 (1.70)</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>85.40 ± 0.20</td>
<td>85.72 ± 0.02 (0.43)</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>65.86 ± 0.10</td>
<td>7.03E8</td>
<td>65.21 ± 0.06 (1.00)</td>
</tr>
<tr>
<td></td>
<td>10%</td>
<td>63.12 ± 0.09</td>
<td>64.31 ± 0.08 (1.10)</td>
</tr>
<tr>
<td></td>
<td>30%</td>
<td>65.41 ± 0.09</td>
<td>66.54 ± 0.02 (0.23)</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>65.75 ± 0.12</td>
<td>66.40 ± 0.14 (0.56)</td>
</tr>
<tr>
<td>ImageNet-100</td>
<td>77.74 ± 0.12</td>
<td>1.81E9</td>
<td>75.12 ± 0.07 (1.18)</td>
</tr>
<tr>
<td></td>
<td>30%</td>
<td>74.04 ± 0.08</td>
<td>75.04 ± 0.10 (1.16)</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>75.83 ± 0.07</td>
<td>76.24 ± 0.12 (1.31)</td>
</tr>
</tbody>
</table>

1K ($D_{\text{Pretrain}}$), while distillation using the SEED objective is performed for 200 epochs on $D_{\text{Target}}$ (same as SimReg). Student networks are derived by sampling from R18’s subspace with the number of filters for each basic block derived from our gating network i.e., channels are selected following policy learned by our gating module (see Figure (2,3) and Section 2 in supplementary) for fair comparison. The representation from the last average pooling layer is $l_2$ normalised and is evaluated using kNN as the evaluation criterion with $k = 1$ to report top-1 accuracy.

### 4 Results

1. **Quantitative assessment.** Table 1 compares the performance of VDG with the other two baselines for dense and gated. The lightweight gated network achieves improved performance across all datasets and target budgets ($t_d$) as compared to Baseline-2, with a negligible drop at $t_d = 50\%$ for CIFAR-10 only. The improved gated performance can be attributed to the fact that both dense and lightweight gated models co-learn during pre-training via weight sharing. However, the performance gain is compensated by a slightly smaller reduction in FLOPs as compared to Baseline-2. Improved performance of the gated network further closes the gap with a completely self-supervised dense model (Baseline-1). In comparison to Baseline-1, we observe minor drop in performance of VDG e.g., at $t_d = 10\%$ across CIFAR-10 (1.27\%), STL-10 (1.28\%), CIFAR-100 (1.35\%), ImageNet-100 (1.27\%) but with a significant reduction in the number of FLOPs. This illustrates that even under severe budget constraints our model achieves comparable performance to Baseline1. This drop further decreases on increasing $t_d$ to 50%.

Another important aspect of our learning method is the performance of the dense ($f_{\theta}$) model. Ideally our aim is to achieve fewer fluctuations with varying $t_d$ with a performance equivalent to dense model as in Baseline-1. However, we find that the performance of the dense network (this work) is slightly below the performance of the dense (Baseline-1) for CIFAR-10/STL-10/ImageNet-100 while for CIFAR-100 the performance is better than the self-supervised dense module. This is interesting because what we achieve from this single pre-training of 500 epochs, is a single base encoder (dense encoder) and gates (via gating modules) and their combination gives a gated lightweight network.
2. Transfer Learning: Table 2 compares the transfer performance of VDG (dense and gated) with VICReg. This experiment gives further insights into the quality of the learned representation in this joint setting. In general, there is a drop in performance for VICReg-Dual for both dense and gated, although the difference is not significant. However, for CIFAR-100 → CIFAR-10 dense and gated outperforms only dense in VICReg at a very low target budget. Even in the case when the model is pre-trained on ImageNet-100, performance is comparable. This is encouraging as this new perspective still maintains good generalisation and transferability.

3. SSL-KD vs SSL-Gating: Table 3 compares the performance of SSL-KD methods with our SSL-Gating framework. To avoid confusion, we would like to reiterate we don’t follow student-teacher paradigm, “Student” in Table 3 for “Ours (Gated)” is basically a R18 (base-encoder) with gates (see Figure 1) and $D_{\text{Pretrain}} = D_{\text{Target}}$. Results in Table 3 are very promising as we outperform both the KD methods by a substantial margin across all budgets. This result suggests that combining gating could serve as a general recipe to obtain a lightweight network along with a dense network during pre-training.

4. Qualitative assessment: Figure 2 shows uniform manifold approximation and projection (UMAP) [49] embeddings of the learned representations ($f_\theta(x) \in \mathbb{R}^{512}$) trained using the dual-setting on the STL-10 dataset and compares it with VICReg [4]. The learned structure is similar to dense (VICReg) at a very low budget. Furthermore, the classes appear to be visually distinct, similar to the VICReg setting, and this is observed for both the dense and gated networks of VICReg-Dual.
**5 Additional Insights**

**Table 4: BT vs VICReg in dual setting.**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$t_d$ (%)</th>
<th>BT-Dual</th>
<th>VICReg-Dual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dense</td>
<td>Gated</td>
<td>FLOPs R.</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>10%</td>
<td>53.64</td>
<td>53.01</td>
</tr>
<tr>
<td></td>
<td>30%</td>
<td>55.03</td>
<td>54.89</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>55.63</td>
<td>55.39</td>
</tr>
<tr>
<td>STL-10</td>
<td>10%</td>
<td>73.45</td>
<td>73.49</td>
</tr>
<tr>
<td></td>
<td>30%</td>
<td>75.51</td>
<td>76.02</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>76.99</td>
<td>77.10</td>
</tr>
</tbody>
</table>

For all experimental settings and studies discussed hereafter; models (and settings) were pre-trained for 500 epochs on CIFAR-100 and STL-10 datasets. Instead of using linear evaluation we use kNN as the evaluation criterion with $k = 1$ to report top-1 accuracy. In all tables, **bold** and underline are the best performing results for the dense and gated module, respectively. Representations are not $l_2$ normalised.

1. **Comparison with the symmetric Barlow Twins (BT) architecture.** VICReg is build upon the findings of BT [67] and it is straightforward to apply the dual setting to BT because it minimises the cross-correlation (regularisation term) to identity $I$ although the loss function is entirely mutual unlike in VICReg. In Table 4 we compare the performance of BT augmented with our setting. We observe that 1-NN performance of BT is low as compared to VICReg-Dual. The drop in performance could be attributed to the fact that VICReg applies independent regularisation which are later matched through the invariance loss. This further validates the hypothesis of choosing VICReg as our objective.

2. **Training with a different base encoder.** In this setting we train a model with two base encoders, one w/ gate (gated) and other w/o gate (dense) i.e, $(2 \times \text{ResNet-18})$ (results in Table 5). An interesting observation is that VICReg-Dual with a single base encoder outperforms a more powerful setting with two base encoders (Table 5) although the FLOPs reduction (FLOPs R.) is higher ($\uparrow$) in the setting of two different encoders. This is due to the fact that the sparsity loss ($\mathcal{L}_G$) operates solely on the un-shared branch so there is no trade-off involved, as in case of single

**Table 5: Alternative base encoders.**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$t_d$ (%)</th>
<th>VICReg-Dual (w/ Invariance)</th>
<th>VICReg-Dual (w/o Invariance)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dense</td>
<td>Gated</td>
<td>FLOPs R.</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>10%</td>
<td>54.92</td>
<td>54.66</td>
</tr>
<tr>
<td></td>
<td>30%</td>
<td>56.34</td>
<td>55.33</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>56.83</td>
<td>57.25</td>
</tr>
<tr>
<td>STL-10</td>
<td>10%</td>
<td>76.61</td>
<td>76.45</td>
</tr>
<tr>
<td></td>
<td>30%</td>
<td>78.55</td>
<td>78.20</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>79.69</td>
<td>79.95</td>
</tr>
</tbody>
</table>
base encoder which simultaneously tries to enforce sparsity and visual invariance.

3. Role of mean squared error in co-evolving. It’s a well known fact in SSL that these methods suffer from dimensional collapse [38, 42]. Training without any regularisation term or any trick [14, 29] would lead to dimensional collapse. Also, the authors in [43] showed that MSE serves as a better option for exact logit matching as compared to Kullback-Leibler (KL) divergence. So, in order to understand the role of MSE we trained a model w/o the invariance loss. As Table 6 shows, we found that there is large performance drop if we remove the invariance term. This implies that the invariance term plays a crucial role and seem to be an important factor, not only for co-evolving, but for self-supervision.

6 Discussion and Conclusion

In this work we presented a novel perspective on unifying synergies between SSL and DC. We exploit DC to induce sparsity into symmetric branches of self-supervised models enabling both branches to co-evolve with each other during training. In addition, this approach also allows simultaneous training of a dense and gated (sparse) sub-network from scratch with a target budget $t_d$ under a self-supervised training objective with minimal computational overhead via weight sharing, thereby offering a good accuracy-efficiency trade-off for a given downstream application. As a result, our single base encoder offers enough flexibility to serve a dual purpose to reduce excessive computational overhead, which we validated through exhaustive experimentation (Tables (1, 2, 3)). However, there are limitations with this work. First, the dense model performance degrades and its performance further fluctuates with varying $t_d$. We experimented with RotNet [26] as a extra proxy loss for the dense branch but it did not yield good performance. Second, we have not imposed any constraint in the training objective that enforces a uniform distribution of channel activations, i.e. preservation of channel diversity during inference (which could also be a solution to the first limitation). Third, in future we will extend this setting to contrastive and non-symmetric architectures.

The work in this paper is an initial attempt to draw parallels and make an inter-connection between both of these fields. However, more research is needed to build a better intuition and insight into these models and also help us understand their other attributes, such as generalisation and ability to transfer to other downstream tasks.

7 Acknowledgments

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number SFI/12/RC/2289_P2, co-funded by the European Regional Development Fund and Xperi FotoNation. This work is dedicated to the memory of Dr. Kevin McGuinness, whose contributions will always be remembered. This work stands as a testament to his enduring influence and our deep appreciation for his invaluable contributions.

References


[42] Li Jing, Pascal Vincent, Yann Lecun, and Yuandong Tian. UNDERSTANDING DIMENSIONAL COLLAPSE IN CON-TRASTIVE SELF-SUPERVISED LEARNING.


