# Dynamic Channel Selection in Self-Supervised Learning

Tarun Krishna[*], Ayush K. Rai[*], Yasser A. D. Djilali, Alan F. Smeaton,
Kevin McGuinness and Noel E. O'Connor

*Insight Centre for Data Analytics, Dublin City University, Dublin, Ireland*

**Abstract**

Whilst computer vision models built using self-supervised approaches are now commonplace, some important questions remain. Do self-supervised models learn highly redundant channel features? What if a self-supervised network could dynamically select the important channels and get rid of the unnecessary ones? Currently, convnets pre-trained with self-supervision have obtained comparable performance on downstream tasks in comparison to their supervised counterparts in computer vision. However, there are drawbacks to self-supervised models including their large numbers of parameters, computationally expensive training strategies and a clear need for faster inference on downstream tasks. In this work, our goal is to address the latter by studying how a standard channel selection method developed for supervised learning can be applied to networks trained with self-supervision. We validate our findings on a range of target budgets $t_d$ for channel computation on image classification task across different datasets, specifically CIFAR-10, CIFAR-100, and ImageNet-100, obtaining comparable performance to that of the original network when selecting all channels but at a significant reduction in computation reported in terms of FLOPs.

**Keywords:** Dynamic Neural Networks, Self-Supervised Learning (SSL), Computer Vision

## 1 Introduction

Self-supervised pre-training of convolutional neural networks, as mentioned in [Chen et al., 2020], has almost matched the performance of supervised pre-training on the ImageNet [Deng et al., 2009] image classification task, but at a cost of a huge number of parameters and inefficient training and inference methods. In the supervised learning setting, as described in [Veit and Belongie, 2018], it is accepted that networks with dynamic data dependent (conditional) channel computation architectures during inference can lead to enhanced representation power, adaptivity, interpretability and can greatly reduce computation cost and memory resources without compromising on the accuracy by a significant margin. This motivates us to investigate the behaviour of neural networks with a channel selection mechanism trained under self-supervision. We hypothesise that self-supervised models are an ideal candidate for such dynamic network structures as they capture highly redundant channel features during pre-training. In addition, there is also a great need to explore more efficient inference methods on downstream tasks for SSL.

In order to establish the trade off between computation and performance, there are two well established research directions when it comes to introducing channel sparsity using dynamic structure in neural networks: channel pruning and channel conditional computation. Dynamic channel pruning, as reported in [Gao et al., 2018], estimates channel saliency measures and allows a network to learn and prioritise certain channels and ignore the irrelevant ones given a fixed target density. Models based on pruning usually learn sparsity through a three stage pipeline i.e, pretrain-prune-finetune while in other works like [Tiwari et al., 2021] the pruning stage itself consists of two steps, namely soft pruning and hard pruning. Conditional channel computation as proposed in [Herrmann et al., 2020] learns to compute only a subset of channels in every layer for the

[*]Equal contribution.

given input and hence benefits inference time efficiency and provides an insight into dataset specific network behaviour. Both channel pruning and conditional channel computation are categorical decisions that cannot be optimised by gradient descent methods; however, using the Gumbel-Softmax trick from [Jang et al., 2016] provides a way to overcome this challenge. Adafuse [Meng et al., 2020] proposed an adaptive temporal fusion network that learns a decision policy to dynamically fuse channels from current and history feature maps (i.e. dynamically deciding which channels to keep, reuse or skip per layer and per instance) for action recognition. Notwithstanding these works, the use of dynamic networks for channel selection has to date been mostly limited to supervised learning settings only.

To the best of our knowledge, there is no study on the impact of conditional channel selection on SSL. The work described in [Caron et al., 2020b] studies the effect of standard pruning techniques developed for supervised learning on a network trained with self-supervision. In particular they use an iterative magnitude based pruning technique described in [Han et al., 2015], which compresses the network by alternatively minimising a training objective and pruning the network parameters with smallest magnitude. The weights of the resulting sub-network are reset based on a weight initialisation scheme: the lottery winning ticket [Frankle and Carbin, 2018, Frankle et al., 2020]. We adopt a similar strategy but focus on exploring the application of standard conditional channel selection methods, as proposed in [Li et al., 2021], to self-supervised models during the pre-training stage and do not include any re-training. Our contributions can be summarised as follows:

1. *Do self-supervised models learn redundant channel features*? Through our exhaustive evaluation we demonstrate that the SSL model (SimSiam) does indeed learn redundant channel features.

2. We show in Table 1 that exploiting this redundancy leads to a drop in computational complexity (FLOPs), reducing inference time without excessively increasing training time, as we learn from scratch and on-the-fly, unlike competing approaches [Caron et al., 2020b] that involve re-training.

3. We demonstrate that this channel selection mechanism preserves the feature quality when evaluated on the task of image classification and gives comparable performance when compared with a vanilla (no channel selection) SSL approach.

## 2 Related Works

### 2.1 Self-Supervised Representation Learning

SSL has recently matched the performance of supervised learning on several computer vision benchmarks [Chen et al., 2020, Djilali et al., 2021, Krishna et al., 2021, Bachman et al., 2019, Grill et al., 2020].

**Contrastive Learning.** Contrastive learning refers to learning by comparison [Oord et al., 2018, Chen et al., 2020] where the final objective is based on some variation of Noise Contrastive Estimation (NCE). The main intuition is to bring similar instances closer in the embedding space while contrasting them with other negative samples to avoid feature collapse. These methods are usually trained in a Siamese setting with shared weights using a large batch size or memory bank [Chen et al., 2020, Oord et al., 2018, Wu et al., 2018, Misra and Maaten, 2020]. Here we present summaries of a series of important aspects that contribute to this.

**Clustering Methods.** One category of self-supervised methods for representation learning is based on clustering [Caron et al., 2018, Asano et al., 2019, Caron et al., 2020a], which alternates between clustering the representations and learning to predict the cluster assignment. These clustering method are also based on contrastive approaches but at cluster level, which also makes the training computationally expensive.

**Distillation Methods.** Recent approaches like BYOL [Grill et al., 2020] and SimSiam [Chen and He, 2021], need no negative samples, yet they learn useful representations and perform on-par with other SSL methods. They learn in a student-teacher setting and consequently avoid feature collapse. However, why and how they avoid collapse is still unclear and an open research area.

**Information Maximization.** A more principled way to avoid feature collapse is to capture information bottlenecks as in Barlow Twins [Zbontar et al., 2021] and VICReg [Bardes et al., 2021].

It is unclear how many channel redundant features are learned by these self-supervised approaches. In this work we aim to study this redundancy by exploiting a dynamic channel selection mechanism from the literature.

### 2.2 Dynamic Channel Computation

**Channel Pruning.** Channel pruning estimates channel saliency measures and eliminates all input and output connections from unimportant channels. The approach reported in [Wen et al., 2016] added group Lasso on channel weights to the model's training loss function resulting in a reduction of the magnitude of channel weights during training. The authors in [He et al., 2018] proposed pruning channels using thresholds by setting unimportant channels to zero. Network Slimming [Liu et al., 2017] used Lasso regularisation with global thresholds. However, deep models pruned with structured sparsity methods lose their capabilities and connections permanently. As a result, dynamic channel pruning methods were devised that learn sparsity through a three-stage pipeline pretrain-prune-finetune or use pretrained models. The authors of [Gao et al., 2018] propose feature boosting and suppression (FBS) to dynamically amplify and suppress output channels computed by convolutional layers. [Tiwari et al., 2021] presents a deterministic pruning strategy using the continuous heaviside function and *crispness loss* to identify a highly sparse subnetwork from an existing dense network.

**Conditional Channel Computation.** Regarding conditional computation at the channel level, the work proposed in [Lin et al., 2017] generates decisions to skip computation for a subset of output channels. The channel gating network [Hua et al., 2019] finds regions among the features that contribute less to the classification result and skips computation on a subset of the input channels for these ineffective regions. ConvAIG [Veit and Belongie, 2018] introduced a network with a hard attention mechanism that adaptively selects specific layers of importance for each input image to assemble an inference graph by specifying a target rate for each layer. The authors of [Herrmann et al., 2020] also study conditional computation at the channel level and extend ConvAIG by learning target rates for each gate by specifying the target rate for the whole network. DGNet [Li et al., 2021] proposed a dual gating mechanism by introducing sparsity along two separate dimensions, spatial and channel, in order to reduce model complexity at run time. For a more detailed background on sparsity, pruning and conditional computation, we recommend the review work presented in [Hoefler et al., 2021].

While [Caron et al., 2020b] studied the behaviour of self-supervised models under standard pruning techniques, we investigate the effect of standard channel selection methods described in DGNet on self-supervised models. We also analyse whether networks trained under self-supervision with channel selection can preserve performance on downstream tasks.

## 3 Method

### 3.1 Self-supervised Module

In this work we consider SimSiam [Chen and He, 2021] as our self-supervised objective. We use ResNet18 as a base encoder[*], which takes two augmented views $\mathbf{x}_1$ and $\mathbf{x}_2$ from an anchor view $\mathbf{x}$ by applying stochastic augmentation from a set of augmentations $\mathcal{P}$. $\mathcal{P}$ comprises random resized crop, color jitter, random gray scale, Gaussian blur and random horizontal flip. These augmented views are processed through $f_{\boldsymbol{\theta}}$ to get a compact representation of $f_{\boldsymbol{\theta}}(\mathbf{x}_1), f_{\boldsymbol{\theta}}(\mathbf{x}_2) \in \mathbb{R}^{512}$. One view is further processed by a prediction MLP head (bottleneck architecture) $g_{\boldsymbol{\phi}}$ giving rise to an asymmetric architecture i.e. $\mathbf{p}_1 \triangleq g_{\boldsymbol{\phi}}(f_{\boldsymbol{\theta}}(\mathbf{x}_1)))$ and $\mathbf{z}_2 \triangleq f_{\boldsymbol{\theta}}(\mathbf{x}_2)$. As a standard practise, a base encoder is augmented with a projection head MLP i.e., $f_{\boldsymbol{\theta}} = h \circ m$, where $m$ and $h$ represents ResNet18 and projection layers respectively. The SimSiam learning objective simplifies to a symmetric cosine similarity:

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

where $\mathcal{D}(\mathbf{a}, \mathbf{b}) = -\mathbf{a}^T\mathbf{b}$, with $\mathbf{a}$ and $\mathbf{b}$ being $L_2$ normalised vectors[*]. `SG` stands for `Stop-Grad()`.

[*]across all experiments

[*]i.e. $\mathcal{D}(\mathbf{a}, \mathbf{b})$ is negative cosine similarity.

## 3.2 Channel Selection via Gating

**Preliminaries.** Channel selection or conditional computation (data dependent gates) is often realised through a gating mechanism. A typical output for an input $\mathbf{x}$ from a convolutional (conv) layer $l$ is given by $f_l(\mathbf{x}_{l-1}) \in \mathbb{R}^{C\times H\times W}$ where $f_l(\mathbf{x}_{l-1})$ consists of a convolution operation with kernel size $k$ followed with a batch normalization layer (BN) and relu $((\cdot)_+)$ non-linearity with $\mathbf{x}_{l-1}$ being the output from the previous layer. The output from a gated convolutional network can be realised as: $\hat{f}_l(\mathbf{x}_{l-1}) = \pi_l(\mathbf{x}_{l-1}) \cdot \text{BN}(\texttt{conv}_l(\mathbf{x}_{l-1}))_+$, where $\pi_l(\mathbf{x}_{l-1})^* \in \{0,1\}^C$ is a gate dependent on input $\mathbf{x}_{l-1}$, which decides whether to keep ("on") or discard ("off") a particular channel. This can be seen as a form of *hard attention* (mask). This masking imposes a discrete structure over the network, making a computational graph for training and inference different. During training this structure is realised through stochastic gradient descent (SGD), while during inference it works as *hard attention*. One of the main reasons for channel selection is to induce sparsity i.e. operate on a lower computational budget (less FLOPs) during inference. In this work we closely follow DGNet using ResNet18 as our base encoder.

**Channel Selection (Gating).** In order for gates[*] to be effective, they need to estimate the importance of input features. This *importance* is often referred to as relevance/saliencies (vectors) of the input feature map (along the channels) in the literature. This relevance is crucial in order for the network to avoid trivial solutions. A simpler way is to use SE block [Hu et al., 2018], as was used in DGNet, to create a relevance vector. This usually requires getting a context vector $\mathbf{z} \in \mathbb{R}^C$ via global average pooling to accumulate spatial information. Finally, this context vector $\mathbf{z}$ is passed through a lightweight network to get channel attention $g_l(\mathbf{x}_{l-1})$, which can be summarized as:

$$g(\mathbf{x}_{l-1}) = \mathbf{W}_1\Big(\text{BN}\big(\mathbf{W}_0\mathbf{z}\big)\Big)_+, \quad \mathbf{W}_1 \in \mathbb{R}^{C_l\times\frac{C_{l-1}}{r}}, \mathbf{W}_0 \in \mathbb{R}^{\frac{C_l}{r}\times C_l}, \tag{2}$$

where $r$ is a reduction ratio. For more details please refer to [Hu et al., 2018]. Finally, to achieve binary mask $\pi_l(\mathbf{x}_{l-1})$ we can use the channel attention $g_l(\mathbf{x}_{l-1})$ and set $\pi_l^i(\mathbf{x}_{l-1}) = 1$ if $g_l^i(\mathbf{x}_{l-1}) \geq 0$ and $\pi_l^i(\mathbf{x}_{l-1}) = 0$ otherwise. This discrete selection works during inference but it breaks the computational graph during training. To make the training possible, the Gumbel-SoftMax Trick [Jang et al., 2016] is adopted. The Gumbel-Trick has been widely used as reparameterisation technique for the task of dynamic channel selection [Li et al., 2021, Herrmann et al., 2020, Veit and Belongie, 2018, Meng et al., 2020]. A gating block is introduced after the first convolution in `Basic Block` of Resnet18 following DGNet. Intuitively, the channel selection network could be interpreted as learning a policy whether to keep (compute) or discard (skip) a particular channel.

## 3.3 Optimisation

To remove unimportant channels and induce sparsity in the gating mask $\pi_l(\mathbf{x}_{l-1})$ we need to add an objective based on some budget $t_d$. To this end we use regularisation, a term used in DGNet as sparsity objective, which is a combination of sparsity and a bound regularisation term:

$$\mathcal{L}_G = \lambda \underbrace{\left(\frac{\sum_{l=1}^{L} F_l^R}{\sum_{l=1}^{L} F_l^O} - t_d\right)^2}_{\text{Sparsity}} + \gamma\mathcal{L}_{Bound},$$

where $F_l^R$ is the average FLOPs over the batch along with FLOPs computation of the gating block, while $F_l^O$ is the original FLOPs without a gating module. Only the blocks with gating modules take part in FLOP computation as they are responsible for any sort of sparsity introduced in the network. The purpose of $L_{Bound}$ is to control early optimisation as detailed in DGNet.

**Final Objective.** Overall training objective is defined as: $\mathcal{L} = \mathcal{L}_{\texttt{SSL}} + \mathcal{L}_G$, with $\lambda = 5$ and $\gamma = 1$ across all the datasets and training regimes.

[*]a vector of dimension equivalent to number of channels with ones and zeroes

[*]channel selection

## 4 Experimental Setup

Table 1: Performance comparison of SimSiam with dynamic channel selection during inference. Evaluated with $k$-nearest neighbours ($k = 1$) on the validation set of CIFAR-10, CIFAR-100 and ImageNet-100 across various target budgets $t_d$. * denotes baseline.

| **Dataset** | **Budget** ($t_d$) | **Acc%** | **FLOPs** |
|---|---|---|---|
| | * | 85.46% | 7.03E8 |
| CIFAR-10 | 10% | 76.72% | 6.64E7 (90.55%↓) |
| | 20% | 78.78% | 1.25E8 (82.11%↓) |
| | 30% | 80.82% | 2.01E8 (71.41%↓) |
| | 40% | 81.35% | 2.66E8 (62.18%↓) |
| | 50% | 81.93% | 3.29E8 (53.15%↓) |
| | 60% | 82.96% | 3.94E8 (43.89%↓) |
| | 70% | **83.08%** | 4.58E8 (34.76%↓) |
| | * | 52.96% | 7.03E8 |
| CIFAR-100 | 10% | 46.84% | 6.88E7 (90.21%↓) |
| | 20% | 49.50% | 1.48E8 (78.88%↓) |
| | 30% | 50.70% | 2.03E8 (71.01%↓) |
| | 40% | 52.05% | 2.65E8 (62.32%↓) |
| | 50% | 52.54% | 3.25E8 (53.67%↓) |
| | 60% | 53.18% | 3.95E8 (43.73%↓) |
| | 70% | **53.50%** | 4.66E8 (33.69%↓) |
| | * | 64.34% | 1.81E9 |
| ImageNet-100 | 30% | 56.08% | 5.30E8(70.43%↓) |
| | 40% | 57.86% | 7.13E8 (60.66%↓) |
| | 50% | **60.38%** | 8.78E8 (51.55%↓) |

**Implementation Details.** We closely follow the approach in DGNet for channel selection. For training, we use SimSiam as a self-supervised model with ResNet18 as a base encoder whose objective is modified as explained in section 3.3. We train the model with varying target densities $t_d$. The implementation of SimSiam is based on the solo-learn library [da Costa et al., 2022]. The base encoder is *randomly initialised*[*] and is trained with SGD for 500 epochs (for a given target budget $t_d$) with a batch size of 256 on 2 Nvidia 2080Ti GPUs, with a warm-start of 10 epochs following a cosine decay with base learning rate of 0.01. Since we are using a very lightweight model as our gating network, there is no significant computational overhead during training. We report the inference speedup in terms of the hardware-independent theoretical metric of FLOPs and not wall-clock time as we are not using any hardware accelerators to utilise sparsity during training. Code is made available here.

**Evaluation.** Training and evaluation is carried on train and validation data of CIFAR-10, CIFAR-100 and ImagNet-100 respectively. For Cifar-10/100 we train for $t_d = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$ while for ImageNet-100 we restrict $t_d$ to only $\{0.3, 0.4, 0.5\}$ due to computational constraints. We use $k$-nearest neighbours as our evaluation metric evaluated with $k = 1$. For *baseline* we train Simsiam with standard objective without any channel selection for each of the datasets under consideration (* in Table 1).

**Results.** Our main findings based on the evaluation criteria validate our initial hypothesis that self-supervised models can learn highly redundant channel features.

Table 1 shows that in the case of CIFAR-10, by keeping only 70% of the channels across the whole network, SimSiam achieves 83.08% accuracy on the KNN task, which is a minor drop from the baseline performance of 85.46% but at an ample reduction of 34.76% in FLOPs. Furthermore, we also find that an enormous 90.55% of FLOPs can be reduced by using only 10% of the channels across the whole network causing a drop of only 8.74% in KNN accuracy. For CIFAR-100, we found that by restricting the channel usage to only 60% over the whole network, SimSiam surpasses the baseline KNN accuracy of 52.95% by 0.22% reaching 53.18%. Additionally, FLOP computation can be reduced by 90.21% by keeping only 10% of the channels, leading to a drop of only 6.12% in KNN accuracy. On ImageNet-100, 50% of the channel usage in the entire network results in 60.38% KNN accuracy, which is 3.96% less than the baseline. However, this decrease in accuracy is compensated by $\sim$51.55% percent drop in FLOPs. Aside from this, we get a substantial 70.43% drop in FLOPs by fixing channel utilization to only 30% in the whole model. Therefore, channel selection can be thought of as a way to take advantage of the trade-off between performance and computation depending the downstream task and individual use case. These results also show that SSL models trained with channel selection preserve the performance in downstream tasks.

Figure 1 shows the channel activation distribution for CIFAR-10, CIFAR-100 and ImageNet-100 datasets, revealing a deeper insight into the dataset specific behaviour of the channel selection network by visualising how many channels in each ResNet18 blocks are always off (skipped), always on (computed), or input dependent.

[*] default initialisation in Pytorch

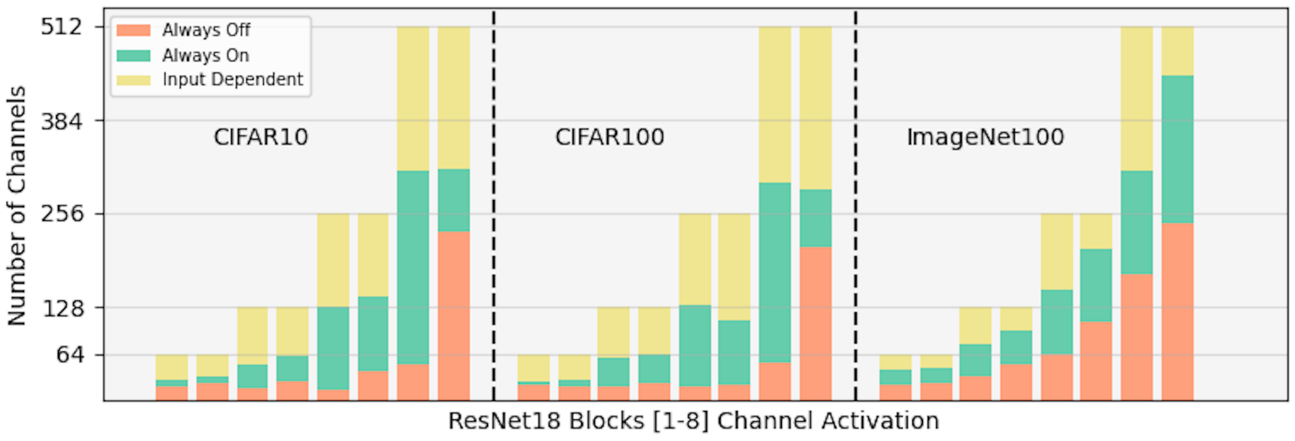

Figure 1: Channel distribution over validation set for $t_d = 0.5$ on CIFAR-10, CIFAR-100, ImageNet-100

We notice that significant number of channels are switched off and switched on all the time in all the three datasets and while others are input dependent. The channel distribution for CIFAR-10 and CIFAR-100 are very similar, which might be due to the fact that image statistics in both of these datasets are similar.

## 5 Conclusion

In this paper, we studied the behaviour of self-supervised learning when integrated with channel selection networks given a global target budget for computational cost. Our empirical results provided interesting insights about self-supervised learning when trained with channel selection. First, self-supervised models learn highly redundant channel features that can be discarded to reduce computational overhead (Figure 1, Table 1). Second, we showed that channel selection modules can significantly reduce FLOP computation and make inference more efficient (Table 1). Third, our results also provide intuition that representations learnt by self-supervised networks with channel selection can also be transferred to downstream tasks.

There are, however, some limitations with our work. First, we still need to evaluate the tranferability of learned representations beyond classification to other downstream tasks such as object segmentation, detection and instance retrieval to name a few. Second, the SSL training objective involves maximizing the agreement between augmented views of the same object or scenes (instance discrimination) and this forces them to have similar representations in the embedding space. In this work, we do not account for this by enforcing some consistency aware constraints for channel selection in the training objective. These limitations will be addressed in future work.

## Acknowledgments

This work has emanated from research supported by Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289_P2, co-funded by the European Regional Development Fund and Xperi FotoNation.

## References

[Asano et al., 2019] Asano, Y. M., Rupprecht, C., and Vedaldi, A. (2019). Self-labelling via simultaneous clustering and representation learning. *arXiv preprint arXiv:1911.05371*. 2

[Bachman et al., 2019] Bachman, P., Hjelm, R. D., and Buchwalter, W. (2019). Learning representations by maximizing mutual information across views. *Advances in neural information processing systems*, 32. 2

[Bardes et al., 2021] Bardes, A., Ponce, J., and LeCun, Y. (2021). Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*. 2

[Caron et al., 2018] Caron, M., Bojanowski, P., Joulin, A., and Douze, M. (2018). Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149. 2

[Caron et al., 2020a] Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. (2020a). Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924. 2

[Caron et al., 2020b] Caron, M., Morcos, A., Bojanowski, P., Mairal, J., and Joulin, A. (2020b). Pruning convolutional neural networks with self-supervision. *arXiv preprint arXiv:2001.03554*. 2, 3

[Chen et al., 2020] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR. 1, 2

[Chen and He, 2021] Chen, X. and He, K. (2021). Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758. 2, 3

[da Costa et al., 2022] da Costa, V. G. T., Fini, E., Nabi, M., Sebe, N., and Ricci, E. (2022). solo-learn: A library of self-supervised methods for visual representation learning. *Journal of Machine Learning Research*, 23(56):1–6. 5

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee. 1

[Djilali et al., 2021] Djilali, Y. A. D., Krishna, T., McGuinness, K., and O'Connor, N. E. (2021). Rethinking 360deg image visual attention modelling with unsupervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15414–15424. 2

[Frankle and Carbin, 2018] Frankle, J. and Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*. 2

[Frankle et al., 2020] Frankle, J., Dziugaite, G. K., Roy, D., and Carbin, M. (2020). Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pages 3259–3269. PMLR. 2

[Gao et al., 2018] Gao, X., Zhao, Y., Dudziak, Ł., Mullins, R., and Xu, C.-z. (2018). Dynamic channel pruning: Feature boosting and suppression. *arXiv preprint arXiv:1810.05331*. 1, 3

[Grill et al., 2020] Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. (2020). Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284. 2

[Han et al., 2015] Han, S., Pool, J., Tran, J., and Dally, W. (2015). Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28. 2

[He et al., 2018] He, Y., Kang, G., Dong, X., Fu, Y., and Yang, Y. (2018). Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*. 3

[Herrmann et al., 2020] Herrmann, C., Bowen, R. S., and Zabih, R. (2020). Channel selection using gumbel softmax. In *European Conference on Computer Vision*, pages 241–257. Springer. 1, 3, 4

[Hoefler et al., 2021] Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., and Peste, A. (2021). Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124. 3

[Hu et al., 2018] Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141. 4

[Hua et al., 2019] Hua, W., Zhou, Y., De Sa, C. M., Zhang, Z., and Suh, G. E. (2019). Channel gating neural networks. *Advances in Neural Information Processing Systems*, 32. 3

[Jang et al., 2016] Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*. 2, 4

[Krishna et al., 2021] Krishna, T., McGuinness, K., and O'Connor, N. (2021). Evaluating contrastive models for instance-based image retrieval. In *Proceedings of the 2021 International Conference on Multimedia Retrieval*, pages 471–475. 2

[Li et al., 2021] Li, F., Li, G., He, X., and Cheng, J. (2021). Dynamic dual gating neural networks. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5310–5319. 2, 3, 4

[Lin et al., 2017] Lin, J., Rao, Y., Lu, J., and Zhou, J. (2017). Runtime neural pruning. *Advances in neural information processing systems*, 30. 3

[Liu et al., 2017] Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. (2017). Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pages 2736–2744. 3

[Meng et al., 2020] Meng, Y., Panda, R., Lin, C.-C., Sattigeri, P., Karlinsky, L., Saenko, K., Oliva, A., and Feris, R. (2020). Adafuse: Adaptive temporal fusion network for efficient action recognition. In *International Conference on Learning Representations*. 2, 4

[Misra and Maaten, 2020] Misra, I. and Maaten, L. v. d. (2020). Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717. 2

[Oord et al., 2018] Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*. 2

[Tiwari et al., 2021] Tiwari, R., Bamba, U., Chavan, A., and Gupta, D. K. (2021). Chipnet: Budget-aware pruning with heaviside continuous approximations. *arXiv preprint arXiv:2102.07156*. 1, 3

[Veit and Belongie, 2018] Veit, A. and Belongie, S. (2018). Convolutional networks with adaptive inference graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–18. 1, 3, 4

[Wen et al., 2016] Wen, W., Wu, C., Wang, Y., Chen, Y., and Li, H. (2016). Learning structured sparsity in deep neural networks. *Advances in neural information processing systems*, 29. 3

[Wu et al., 2018] Wu, Z., Xiong, Y., Yu, S., and Lin, D. (2018). Unsupervised feature learning via non-parametric instance-level discrimination. *arXiv preprint arXiv:1805.01978*. 2

[Zbontar et al., 2021] Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320. PMLR. 2