Guillotine Regularization: Improving Deep Networks Generalization by Removing their Head

Anonymous authors
Paper under double-blind review

Abstract

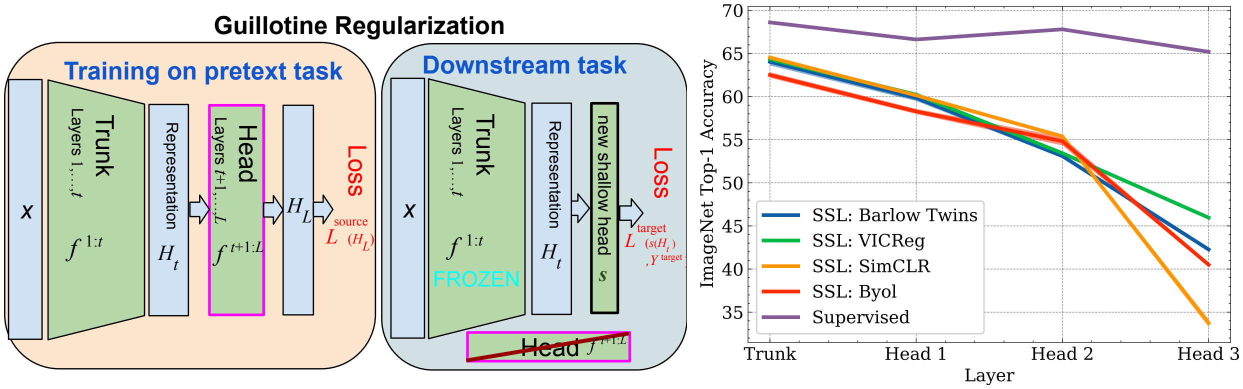
One unexpected technique that emerged in recent years consists in training a Deep Network (DN) with a Self-Supervised Learning (SSL) method, and using this network on downstream tasks but *with its last few layers entirely removed*. This usually skimmed-over *trick* is actually critical for SSL methods to display competitive performances. For example, on ImageNet classification, more than 30 points of percentage can be gained that way. This is a little vexing, as one would hope that the network layer at which invariance is explicitly enforced by the SSL criterion during training (the last layer) should be the one to use for best generalization performance downstream. But it seems not to be, and this study sheds some light on why. This trick, which we name Guillotine Regularization (GR), is in fact a generically applicable form of regularization that has also been used to improve generalization performance in transfer learning scenarios. In this work, through theory and experiments, we formalize GR and identify the underlying reasons behind its success in SSL methods. Our study shows that the use of this trick is essential to SSL performance for two main reasons: (i) improper data-augmentations to define the positive pairs used during training, and/or (ii) suboptimal selection of the hyper-parameters of the SSL loss.

1 Introduction

Many recent self-supervised learning (SSL) methods consist in learning invariances to specific chosen relations between samples – implemented through data-augmentations – while using a regularization strategy to avoid collapse of the representations (Chen et al., 2020a;b; Grill et al., 2020; Lee et al., 2021b; Caron et al., 2020; Zbontar et al., 2021; Bardes et al., 2022; Tomasev et al., 2022; Caron et al., 2021; Chen et al., 2021; Li et al., 2022; Zhou et al., 2022a;b). Incidentally SSL learning frameworks also heavily rely on a simple trick to improve downstream task performances: *removing the last few layers of the trained deep network*. From a practical viewpoint, this technique emerged naturally (Chen et al., 2020a) through the search of ever increasing SSL performances. In fact, on ImageNet (Deng et al., 2009), such technique can improve classification performances by around 30 points of percentage (Figure 8b).

Although it improves performances in practice, not using the layer on which the SSL training was applied is unfortunate. It means throwing away the representation that was explicitly trained to be invariant to the chosen set of data augmentations, thus breaking the implied promise of using a more structured, controlled, invariant representation. By picking instead a representation that was produced an arbitrary number of layers above, SSL practitioners end up relying on a representation that likely contains much more information about the input (Bordes et al., 2021) than should be necessary to robustly solve downstream tasks.

Although the use of this technique emerged independently in SSL, using intermediate layers –instead of the deepest layer where the initial training criterion was applied– of a neural networks has long been known to be useful in transfer learning scenarios (Yosinski et al., 2014). Features in upstream layers often appear more general and transferable to various downstream tasks than the ones at the deepest layers that are too specialized towards the initial training objective. This strongly suggests a related explanation for its success in SSL: does removing the last layers of a trained SSL model improve performances *because of a misalignment between the SSL training task (source domain) and downstream task (target domain)?*



(a)

(b)

Figure 1: a) An illustration of Guillotine Regularization. During training, a small neural network named the *Head* (also coined as *projector* in the SSL literature (Chen et al., 2020a)) is added on top of another deep network refereed as the *Trunk*. This *Head* can be viewed as a buffer between the training loss and the Trunk that can absorb any bias related to a ill optimisation. When using such network on downstream tasks, we throw away the Head. b) We measure with linear probes the accuracy at different layers on a resnet50 (as Trunk) (see Figure 8 for vision transformers) on which we added a small 3 layers MLP (as Head) for various supervised and self-supervised methods. For each method, we show the mean and standard deviation across 3 runs (The std between different runs is low). With traditional supervised learning, at a first glance there doesn't seem to be much gain in using the representation at the higher layer of the Head in comparison with the last layer of the Trunk. However, when looking at self-supervised methods, the gap in performances between the linear probe trained at different levels can be as high as 30 points of percentage.

In this paper, we examine that question thoroughly. We first reframe and formalize the *trick of removing layers post-training* as a generically applicable regularization strategy that we call **Guillotine Regularization** (Figure 8a). We then study its impact on generalization in various supervised and self-supervised scenarios. Our findings demonstrate that when the training and downstream tasks are identical, Guillotine Regularization has nearly no benefit i.e. performances with or without the last few layers are close. Through controlled experiments, we validate that Guillotine Regularization 's benefit grows with the misalignment between the training objective and downstream task. Beyond the standard distribution shifts in input/output, we also surprisingly notice that Guillotine Regularization proves useful when the choice of hyper-parameters are sub-optimal i.e. the benefits of Guillotine Regularization occur in much more general cases than solely from data distribution shifts.

To summarize, this paper's main contributions are the following:

- To formalize and motivate theoretically the common trick of using a projector in Self-Supervised-Learning as a general regularization method under the name Guillotine Regularization.
- To provide empirical insights that clarify in which situations this method should be expected to be beneficial.
- To show that the usefulness of Guillotine Regularization in Self-Supervised learning comes not only from the inherent misalignment between the SSL training task – with its data augmentations – and the downstream task, but also because it allows to be more robust to suboptimal choices of hyper-parameters in the objective.

2 Related work

Self-supervised learning Many recent works on self-supervised learning (Chen et al., 2020a;b; Grill et al., 2020; Lee et al., 2021b; Caron et al., 2020; Zbontar et al., 2021; Bardes et al., 2022; Tomasev et al., 2022;

Caron et al., 2021; Chen et al., 2021; Li et al., 2022; Zhou et al., 2022a;b) rely on the addition of few non linear layers (MLP) – termed *projection head* – on top of a well established neural network – termed *backbone* – during training. This addition is done regardless of the neural network used as backbone, it could be a ResNet50 (He et al., 2016) or a Vision Transformer (Dosovitskiy et al., 2021). Some works already tried to understand why a projection head is needed for self-supervised learning. Appalaraju et al. (2020) argue that the nonlinear projection head acts as filter that can separate the information used for the downstream task from the information useful for the contrastive loss. In order to support this claim, they used deep image prior (Ulyanov et al., 2018) to perform features inversion to visualize the features at the backbone level and also at the projector level. They observe that features at the backbone level seem more suitable visually for a downstream classification task than the ones at the projector level. Another related work (Bordes et al., 2021) similarly tries to map back the representations to the input space, this time by using a conditional diffusion generative model. The authors present visual evidence confirming that much of the information about a given input is lost at the projector level while most of it is still present at the backbone level. Another line of work tries to train self-supervised models without the use of a projector. Jing et al. (2022) shows that by removing the projector and cutting the representation vector in two parts, such that a SSL criteria is applied on the first part of the vector while no criterion is applied on the second part, improves considerably the performances compared to applying the SSL criteria directly on the entire representation vector. This however works mostly thanks to the residual connection of the resnet. In contrast with these approaches, our work focus on identifying which components of traditional SSL training pipelines can explain why the performances when using the final layers of the network are so much worse than the ones at the backbone level. This identification will be key for designing future SSL setups in which the generalisation performance doesn't drop drastically when using the embedding that the SSL criterion actually learns.

Transfer learning The idea of using the intermediate layers of a neural network is very well known in the transfer learning community. Work like Deep Adaptation Network (Long et al., 2015) freeze the first layers of a neural network, fine-tune the last layers while adding a head which is specific for each target domain. The justification behind this strategy is that deep networks learn general features (Caruana, 1994; Bengio, 2012; Bengio et al., 2011), especially the ones at the first layers, that may be reused across different domain (Yosinski et al., 2014). Oquab et al. (2014) demonstrate that when limited amount of training data are available for the target tasks, using the frozen features extracted from the intermediate layers of a deep network trained on classification can help solve object and action classification tasks on other datasets. SSL methods can be viewed as devising unsupervised training tasks (source tasks) that will yield *representations* that transfer well to typically supervised downstream tasks (target task). Modern SSL methods define a training objective that relies on data-augmentation based views of different inputs, without access to their associated targets/classes, it is direct to see that considering a classification task from a SSL trained models falls under the realm of transfer learning. That being said, the question that remains unanswered is to understand which specific aspects of the SSL techniques affect the transferability, and the usefulness of the guillotine trick. Is it to what degree the distribution resulting from data augmentations commonly used to train SSL models differs from the true input distribution of the downstream tasks? Is it a possible overfiting on the SSL training task? Or is it other inherent differences between the SSL training task and the downstream task?

Out of distribution (OOD) generalization Kirichenko et al. (2022) demonstrates that retraining only the last layer with a specific reweighting helps to "forget" the spurious correlations that were learned during the training. Such work emphasizes that most of the spurious correlation due to the training objective is contained in the last layers of the network. Thus, retraining them is essential to remove such bias and generalize better on downstream tasks. Similarly Rosenfeld et al. (2022) show that retraining only the last layers is most of the time as good as retraining the entire network over a subset of downstream tasks. Lastly, Evci et al. (2022) demonstrates the usefulness of using intermediate layers for OOD. Our study also confirms that Guillotine Regularization show important properties with respect to OOD generalization.

3 Guillotine Regularization: A regularization scheme to improve generalization of deep networks

In this section we formalize Guillotine Regularization along with a theoretical motivation which helps to highlight scenarios for which this technique is well-suited.

3.1 (Re)Introducing Guillotine Regularization From First Principles

We distinguish between a **source** *training task* with its associated training set, and a **target** *downstream task* with its associated dataset[1]. It is the performance on the downstream task that is ultimately of interest. In the simplest of cases both tasks could be the same, with their datasets sampled i.i.d. from the same distribution. But more generally they may differ, as in SSL or transfer learning scenarios. In SSL we typically have an *unsupervised* training task, that uses a training set with no labels, while the downstream task can be a supervised classification task. Also note that while the bulk of training the model's parameters happens with the training task, transferring to a different downstream task will require some additional, typically lighter, training, at least of a final layer specific for that task. In our study we will focus on the use of a representation computed by the network trained on the training task and then frozen, which gets fed to a simple linear layer that will be tuned for the downstream task. This "linear evaluation" procedure is typical in SSL and aims to evaluate the quality/usefulness of an unsupervised-trained *representation.* Our focus is to ensure good generalization to the downstream task. Note that training and downstream tasks may be misaligned in several different ways.

Informally, Guillotine Regularization consists in the following: for the *downstream task*, rather than using the last layer (layer L) representation from the network trained on the *training task*, instead use the representation from a few layers above (layer t, with $t < L$). We thus *remove* a small multilayer "head" (layers $t+1$ to L) of the initially trained network, hence the name of the technique. We call the remaining part (layers 1 to t) the *trunk*[2]. The method is illustrated in Figure 8a.

Formally, we consider a deep network that takes an input X and computes a sequence of intermediate representations $H_1, \dots, H_L$ through layer functions $f^{(1)}, \dots f^{(L)}$ such that $H_\ell = f^{(\ell)}(H_{\ell-1})$, starting from $H_0 = X$. The entire computation from input X to last layer representation H_L is thus a composition of layer functions[3]:

$$H_L = f_{\theta,\phi}(X) = (\underbrace{f^{(L)} \circ \cdots \circ f^{(t+1)}}_{\text{head } f_\phi^{t+1:L}} \circ \underbrace{f^{(t)} \circ \cdots \circ f^{(1)}}_{\text{trunk } f_\theta^{1:t}})(X)$$

The parameters θ and ϕ of trunk $f_\theta^{1:t}$ and head $f_\phi^{t+1:L}$ are then trained on the entire training set of examples $\mathbf{X}^{\text{source}}$ of the training task (optionally with associated targets $\mathbf{Y}^{\text{source}}$ that we may have in transfer scenarios, but will typically be absent in SSL), to minimize the training task objective L^{source}:

$$\hat{\theta}, \hat{\phi} = \arg\min_{\theta,\phi} L^{\text{source}}(f_\phi^{t+1:L}(f_\theta^{1:t}(\mathbf{X}^{\text{source}})), \mathbf{Y}^{\text{source}})$$

Then the multilayer head $f_\phi^{t+1:L}$ is discarded, we add to the trunk a (usually shallow) new head s_w and we train its parameters w, using the training set of examples for the downstream task $(\mathbf{X}^{\text{target}}, \mathbf{Y}^{\text{target}})$, to minimize the downstream task objective L^{target}:

$$\hat{w} = \arg\min_{w} L^{\text{target}}(s_w(\underbrace{f_{\hat{\theta}}^{1:t}(\mathbf{X}^{\text{target}})}_{\text{representation } \mathbf{H}^{\text{target}}}), \mathbf{Y}^{\text{target}})$$

The final network, whose performance we can evaluate on separate downstream task validation or test sets, is defined as: $f^{\text{target}} = s_{\hat{w}} \circ f_{\hat{\theta}}^{1:t}$.

---

[1] Terminology pretext-training / downstream comes from SSL, while source / target is used in transfer learning

[2] head / trunk are also known as projection head / backbone in the SSL literature

[3] Precisely, a "layer function" $f^{(\ell)}$ can correspond to a standard neural network layer (fully-connected, convolutional) with no residual or shortcut connections between them, or to entire blocks (as in densenet, or transformers) which may have internal shortcut connections, but none between them.

Information Theoretic Motivation. Consider that we have as input a random variable X and that we produce feature maps in a Markov Chain fashion $X \rightarrow H_1 \rightarrow H_2 \rightarrow \cdots \rightarrow H_L$, i.e. $H_{\ell+1}$ is conditionally independent of all the previous feature maps given H_ℓ.

Using the data processing inequality (Beaudry & Renner, 2011) we directly have the following inequality in terms of mutual information:

$$I(X, H_\ell) \geq I(X, H_{\ell+1}), \forall \ell \in \{1, \ldots, L-1\}$$

i.e. no processing of H_ℓ by layer ℓ can increase the amount of information that $H_{\ell+1}$ encodes about X (this led to the famous "garbage in, garbage out" adage). Furthermore, the only way for this inequality to be tight, is for the $H_\ell \rightarrow H_{\ell+1}$'s layer/block processing to be one-to-one (homeomorphism) which is almost surely never the case in most current Deep Network architectures, due amongst other things to the ubiquitous use of ReLU activations[4] Hence, without loss of generality, the above inequality will be strict for most current models.

When cross-validating, e.g. a model's architecture, optimizer, data-augmentation pipelines, one directly aims at maximizing the generalization performance of the model estimated from the validation set. One direct consequence of this process is that the best performing model will have a representation H_L that disregards as much information as possible about X, only maintaining the sufficient amount to predict Y^{source} (Xu & Raginsky, 2017; Steinke & Zakynthinou, 2020). In fact, this argument is at the origin of the Information Bottleneck principle that optimizes $\min_{\theta,\phi} I(X, H_L) - \beta I(H_L, Y^{\text{source}})$ (Tishby & Zaslavsky, 2015). Hence, whenever one leverages a well calibrated model to solve a different task, it is clear that the information contained within H_L will be insufficient unless Y^{target} is predictable from Y^{source}. This means that too much compression, although beneficial for the source task will generally be detrimental to the target task. This can occur whenever the source and target distributions or tasks differ.

3.2 Experimental insights showing the usefulness of GR

Misalignment between the training (source) and downstream (target) task while using the same input data distribution. The theoretical effectiveness of GR for transfer is not surprising since this technique has been used for years in the transfer learning research literature to improve generalization across different tasks. As a simple illustration, we present Figure 2 which show how much performances on a given task can vary depending on which layer has been chosen as features extractor. In this figure, we used an artificially created object dataset in which we are able to play with different factors of variations. The dataset consists of renderings of 3D models from 3D warehouse (Trimble Inc). Each scene is built from a 3D object, a floor and a spot placed on top of the object to add lighting. This allows us to control every factor of variation and produce complex transformations in the scene. We vary the rotation of the object defined as a quaternion, the hue of the floor, and the spot hue as well as it position on a sphere using spherical coordinates. We provide more details on the dataset and rendering samples in the appendix. We observe in Figure 2 that when training a supervised model on the object rotation prediction task and evaluating the linear probe on the same task across different layers, the best results are obtained on the last layer. However, when using the same frozen neural network to predict other attributes like the Spot θ, the best performances are obtained few layers before the last one. Similarly, when training with a self-supervised objective (SimCLR), we can see that the different factors of variation are most easily retrievable before the projector. This means that representations before the projector will be more versatile as they will contain information that was removed by the pretraining task. For example if our downstream task is to predict the rotation the representation at the block4 will be optimal. Such results highlight the need to use Guillotine Regularization when there is a shift in the prediction task.

Misalignment between the training and downstream tasks (different labels) and input distributions. Such shift occurs naturally in transfer learning. When using a pretrained model to predict new classes that weren't present in the original dataset, there is a bias in the data distribution as well as in the fine-tuning objective (with respect to the training settings). We did a first experiment in Figure 3a in which

---

[4] A notable exception are generative Normalizing Flow models (Rezende & Mohamed, 2015) which are explicitly constructed to provide one-to-one invertible mappings.

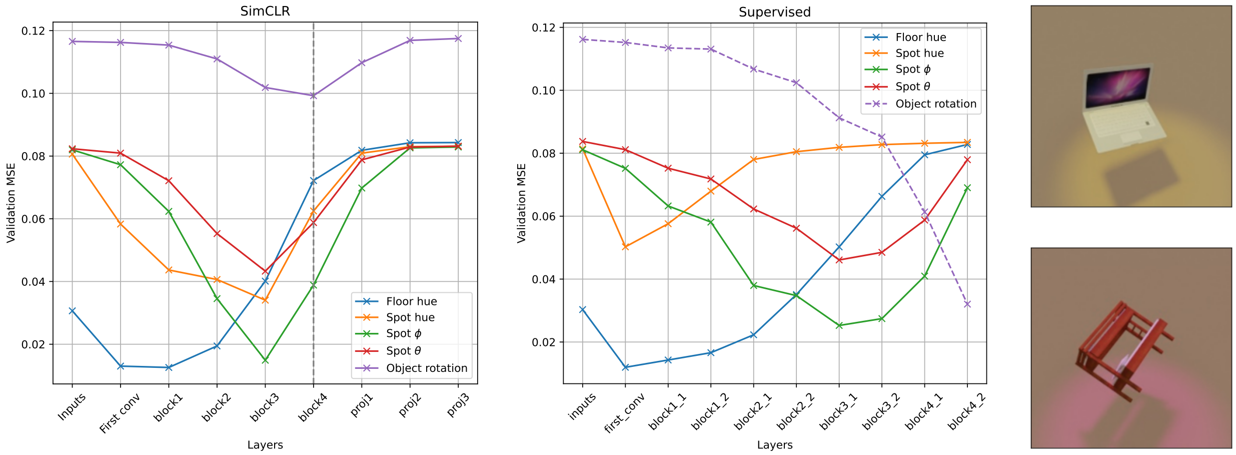
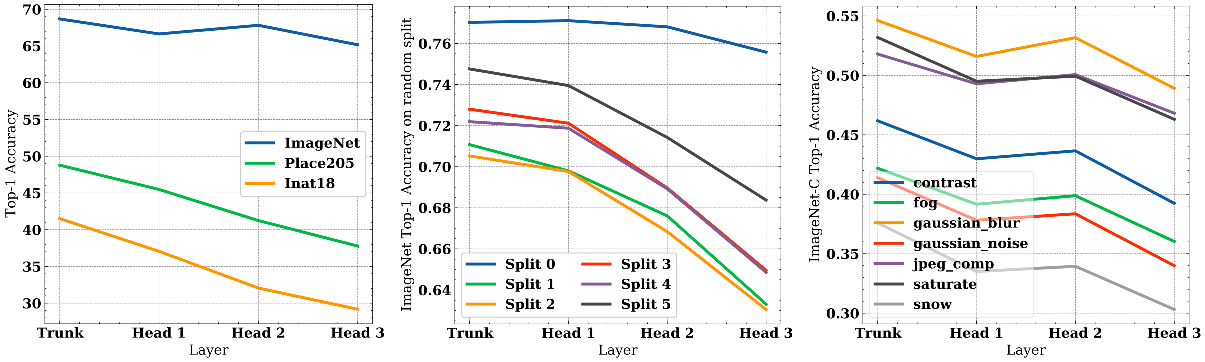


Figure 2: Training a linear regression to predict latent variables from pooled intermediate representations of a network trained with a self-supervised objective (using SimCLR) or a supervised objective (trained to predict 3D rotations of an object). The data used consists of renderings of 3d objects from 3D Warehouse (Trimble Inc) where we control the floor, lighting and object pose with latent variables, see samples on the right. In the supervised setting, when looking at the Validation Mean Squared Error for object rotations prediction, the lowest error is obtained with the linear probe at the last layer of the neural networks. In contrast, the lowest error for other attributes like the Spot θ prediction are obtained with the linear probes localized 3,4 or 5 layers before the output of the networks. In the self-supervised setting, we also see that the predictor is responsible for a lot of the invariance to augmentation, and that the information is most easily retrievable before it. These results highlight the need to use Guillotine Regularization i.e removing the last layers of the neural network to generalize better on other tasks.

we train a supervised Resnet50 over ImageNet. Then, we froze the weights of the model and train a linear probe over ImageNet (Deng et al., 2009), Place205 (Zhou et al., 2014) and Inaturalist18 (Horn et al., 2018) at different layers. We observe that the performances on ImageNet are fairly close in each layers whereas for Place205 and Inaturalist18, there is an important gap in performances between the last layer in the Head and the Trunk. In addition of this experiment, we train another Resnet50 but this time only on a random set of 250 classes of ImageNet with the objective to evaluate how much GR can help with respect to overfitting. We trained linear probes at each layers of the neural network over different 250 classes subset of the ImageNet classes. In Figure 3b, the subset $S0$ correspond to the original split used during training whereas the subset Split 1-5 are different 250 classes subsets. We observe an important drop in performances on the linear probe trained at the projector level for every 250 classes random split that is different from the 250 classes used during training. This last result shed light on how much Guillotine Regularization is useful to absorb bias during training. Another benefit from GR is that even on the original split of the data, we can observe a better accuracy at the trunk level than at the last Head level. This can easily be explained by the capacity of the neural network to overfit on this small training set. By using higher layer, GR might be able to partially remove the over-fitting bias.

Misalignment between the training input data distribution and testing input data distribution while using the same training and downstream task. Another type of bias can arise when using a wrongful data distribution after training of the model. This scenario is often refer as Out Of Distribution (OOD) since the distribution of the data used by the model become different from the one used during training. In our experiment, we used a trained Resnet50 model (on which we added a small 3 layers MLP as head) over a classification task on ImageNet (Deng et al., 2009). We trained a linear probe on each layers of this network (while keeping the weights frozen) on the same classification task as the ones that was used during training of the full network. Then, we use ImageNet-C (Hendrycks & Dietterich, 2019) which is a modified version of the validation set of ImageNet on which different data transformations were applied. This setup allow us to test the performances of each linear probe at different level inside the neural network. Our experiment in Figure 3c demonstrates that for many transformations that are applied on the data the



(a) Datasets (b) Random split (c) OOD

Figure 3: **Empirical benefits of GR** For each experiments, we trained a headed supervised Resnet50 over ImageNet (For a) and c), we trained this network over the full training set whereas for b) we use a random subset of 250 classes). Then, we froze the headed Resnet50 parameters and trained linears probes over representation at different layers. a) Validation accuracy given by linear probes trained on ImageNet (Deng et al., 2009), Place205 (Zhou et al., 2014) and Inaturalist18 (Horn et al., 2018). This experiment highlight that GR plays an important role when doing transfer on other datasets. b) Validation accuracy given by linear probes on different random subset of 250 ImageNet's classes for each layers. Split 0 corresponds to the same subset of classes that was used for training whereas Split 1-5 corresponds to different random split. This result highlight how much the task's overfitting bias is largely absorb at the head level. c) Validation accuracy given by linear probes trained on ImageNet-C (Hendrycks & Dietterich, 2019). We plot the accuracy on ImageNet-C for each transformations at different layers. Even in an OOD setting, GR is useful to improve robustness.

performances at the backbone (trunk) level are still better than the ones obtained at the head level. Such results highlight the need to use Guillotine Regularization when there is a shift in the data distribution.

4 Demystifying the Role of the Projection Head in Self-Supervised Learning

Self-Supervised Learning is often considered a distinct learning paradigm in between supervised and unsupervised learning. In reality, the distinction is not as sharp, and much of SSL can be understood as solving a pretext-tasks akin to a supervised task, merely with pseudo-labels obtained in another way than by human annotation. In this section we first highlight that SSL methods such as SimCLR can be seen as solving a specific supervised learning task. If most SSL methods can be formulated as supervised problems, it implies that the properties observed in the previous section should also apply to SSL. In the second part, we discuss some techniques that reduce the need for GR in SSL. Finally, we highlight how GR is not only useful in transfer but is an essential tool in SSL to gain robustness with respect to the choice of hyper-parameters.

4.1 On the relationship between Supervised and Self-Supervised learning

We closely follow the notation from Chen et al. (2020a) who define the SimCLR loss function

$$\ell_{i,j} = -\log \frac{\exp(\boldsymbol{Z}_{i,j}/\tau)}{\sum_{k=1,k\neq i}^{2N} \exp(\boldsymbol{Z}_{i,k}/\tau)}, \tag{1}$$

for all positive pairs (i, j), with τ the temperature parameter, and with $\boldsymbol{Z}$ the similarity matrix such that $\boldsymbol{Z}_{i,j} = \text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j) = \boldsymbol{z}_i^\top \boldsymbol{z}_j / \|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|$ the cosine similarity. Now, notice that if we use only one of the mini-batch to compute the negative examples, we get

$$\ell_{i,j} = -\log \frac{\exp(Z_i[j])}{\sum_{k=1}^{N} \exp(Z_i[k])} \ , \tag{2}$$

which is the traditional cross entropy loss used in a classification setting when using N classes. To summarize, instead of using a fixed set of labels, SimCLR defines N pseudo-labels which correspond to the size of the mini batch. This setting can be viewed as a supervised classification setup in which the distribution of the labels change at each iteration.

Now let's imagine a setting in which instead of using mini batch, we use the entire dataset for each update (Full batch setting). In this instance, M will be equal to the number of images in the dataset. Without the randomness caused by the mini batch sampling, the order on which the similarity matrix will be computed will always be the same. Meaning that every augmented transformation of the image at index 0 will always match the index 0 in the embedding space. When such order is fixed, the SimCLR criteria is equivalent to a negative crossentropy loss over the output of a classifier which predict the index of a given image with respect to a random transformation. Under such view, contrastive methods approaches like SimCLR can be seen as replacing the traditional supervised label classification task by an image index classification task. In such instance, we recognize a traditional transfer learning setting: where too few labels are available for a given downstream task, the model is trained on a (related) index prediction task with the hope that it will be able to generalize well on this downstream task.

However SimCLR is not bound to image index prediction. We could e.g. instead predict, from a given image, the index of the video from which this image has been extracted. There are many different possible pseudo-labels we could use to then train a model in a supervised way. These labels can be easy to obtain (like the index of an image) or costly to get (like classes or specific segmentation mask). However an ideal SSL method should be able to adapt for any given set of pseudo labels. In fact, Lee et al. (2021a); HaoChen et al. (2021); Balestriero & LeCun (2022) demonstrates how many SSL methods, s.a. SimCLR, VICReg and similar, can learn essentially (up to rotation) the same representations as those obtained with supervised learning, if we employ the true labels to define the positive pairs and use the same data-augmentation pipeline.

4.2 Reducing the Need for a Projector in Self-Supervised Learning by increasing the alignment with the downstream task

Since SSL methods rely on a pretext training task that differs from the downstream task, Guillotine Regularization seems to be perfectly adapted to absorb the bias towards the training task. As showed in the previous section, there can be a strong relationship between traditional supervised learning and self-supervised learning with the main differences being how "labels" are defined and the use of hyper-parameters associated to a specific SSL loss. To confirm the hypotheses that GR has better performances at the trunk level than at the head level because of a bias absorption, we need to verify that reducing the bias between the pretext and downstream task, results in reducing the performance gap between the Trunk and Head representations. Ideally, we would like to get close to the supervised scenario in Figure 1 that has no major performance gap between the different layers of the projector. To do so, we devise two experimental setups in which we replace the traditional data augmentation pipeline used in SSL, which consists of using handcrafted augmentation on each image to create a set of pairwise positive samples.

In the first setup, while using the exact same SSL criterion (SimCLR), we use as positive examples pairs of images that belong to the same class, and as negative examples images that don't belong to the same class. Note that the SSL training criteria will push towards a collapse in the representation space of all the images belonging to the same class, while pushing further apart the different class clusters. By doing so the training SSL objective becomes perfectly aligned with the downstream classification task, despite using a SSL training criteria instead of a traditional Cross Entropy Loss.

In the second setup, we use as positive pairs the closest neighbors found by a pretrained SSL model trained with the traditional data augmentation pipeline. The reasoning is that if instead of considering each image of the dataset as its own specific class, we use clusters of many images to define the positive pairs, we might be able to close the gap with respect to a supervised baseline without the need of labels.

In Figure 4, we show the differences in accuracy between the backbone and the projector with respect to these two new data augmentation scenarios. The baseline, using the traditional SimCLR positive pairs based on data augmentations is in blue, the nearest neighbors setup in orange and the class based setup in green. We

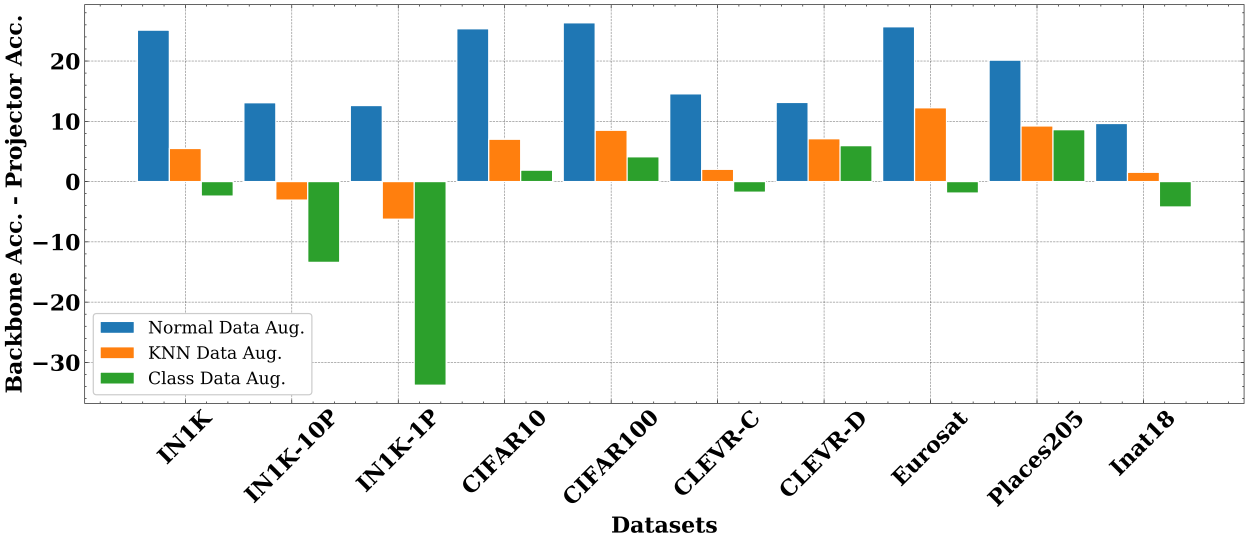
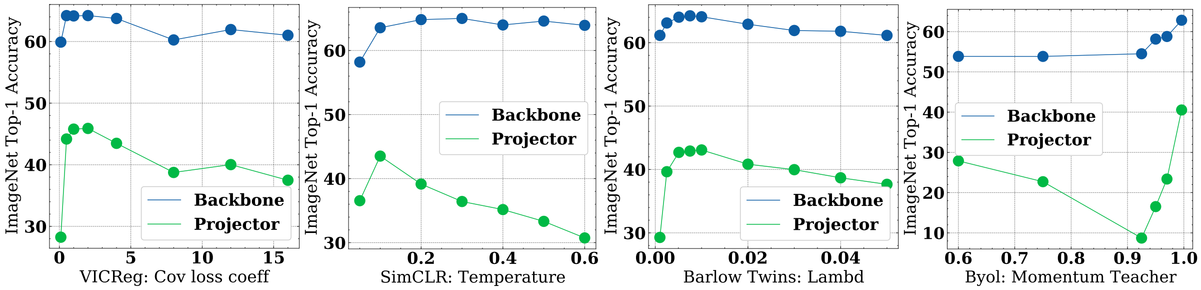


Figure 4: Difference in accuracy with linear probing between the projector and backbone representation with different alignments with respect to the classification downstream task. In this experiment we used SimCLR and we change how the positive pair are defined to better aligned with a classification downstream task. In blue, our baseline, we trained SimCLR with the traditional SSL data augmentations which defines the positive view as two augmentations of a same image. In orange, we use the embedding of a pretrained model to define the positive pair as two nearest neighbors under a pretrained model (while using the same data augmentation as the baseline). In green, we use a supervised class label selection to define the positive examples. In this scenario, SimCLR should learn to produces similar embedding to all images belonging to a given class. All three models are trained on ImageNet (IN1K), then we evaluate them with a linear probe across a wide range of downstream tasks at the backbone and projector level and show the difference in accuracy between both. **When the difference is positive, the accuracy at the backbone level is higher than the one at the projector level, highlighting the benefits of Guillotine Regularization. In contrast when the difference is negative, the accuracy at the projector level is higher than the one at the backbone level. In this instance, Guillotine Regularization is not needed.** When positives pairs are defined as belonging to a given class, there is no misalignment with the imagenet classification downstream task. Thus on ImageNet-1K, ImageNet1k-10P (10% of the training set to train the linear probe) and ImageNet1k-1P (1% of the training set to train the linear probe), we observe that the performances at the projector level are much higher than the ones at the backbone level. Interestingly, the nearest neighbors heuristic also reduce considerably the impact of Guillotine Regularization across several downstream tasks.

observe for SimCLR that using the nearest neighbors based heuristic is helping in reducing the gap between the pretext and downstream task while having a purely supervised heuristic to define the positive pair is removing the need for Guillotine Regularization across several downstream tasks. Hence confirming the hypothesis that the effectiveness of Guillotine Regularization depends of the alignment between the pretext and downstream task in self-supervised learning.

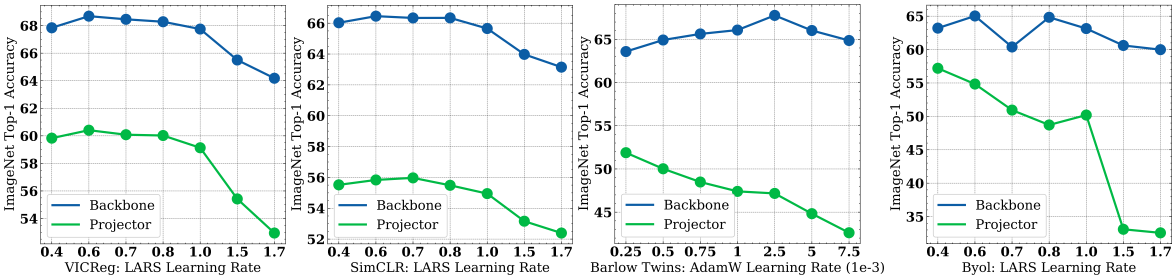
4.3 An important bonus given by Guillotine Regularization: Robustness to Incorrect Hyper-Parameters

In the previous section, we explained how, under certain conditions, SimCLR could be equivalent to a supervised model trained on an index prediction task. Thus, it is not surprising that GR help transfer learning generalization. However, in the specific context of SSL, GR is not only essential for transfer but also confers robustness in regards to a suboptimally defined objective. In Figure 5 we study the effect of GR with respect to an hyper-parameter grid search for various SSL methods (VICReg, SimCLR, Barlow Twins and Byol). When looking at the performances on the backbone level, one can observe an almost stable classification task performance for whatever value was chosen as hyper-parameter. However, when



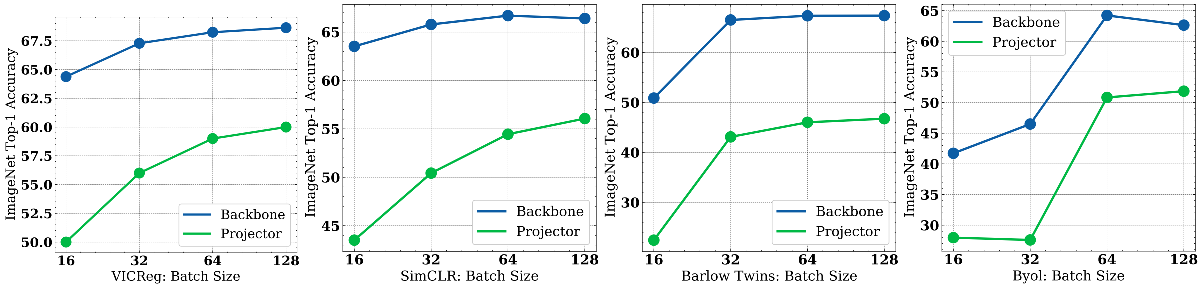
(a) VICReg (b) SimCLR (c) Barlow Twins (d) Byol

Figure 5.1: Robustness with respect to the loss hyper-parameters.



(e) VICReg (f) SimCLR (g) Barlow Twins (h) Byol

Figure 5.2: Robustness with respect to the learning rate.



(i) VICReg (j) SimCLR (k) Barlow Twins (l) Byol

Figure 5.3: Robustness with respect to the batch size.

Figure 5: Robustness with respect to the learning rate. We train SimCLR, VicReg, Barlow Twins and Byol with different hyper-parameters and evaluate with a linear prob, the performances at the backbone but also at the projector level on ImageNet classification task. For each model, we observe that the accuracy given by the linear probe at the backbone level is fairly stable across the grid search of hyper-parameters while the linear probe at the projector level can reach very low accuracy.

looking at the results at the Projector level, the performance accuracy might drop significantly depending on the hyper-parameters values. This Figure show how much GR can absorb the bias of an ill-defined task or objective. What is more impressive is that for a specific accuracy at the backbone level, the accuracy at the projector level can be arbitrary high or low. This indicates that even if the projector doesn't learn any feature that could be useful for a classification task, it doesn't matter as long as we use GR.

4.4 Experimental details

We use Pytorch (Paszke et al., 2019) and FFCV (Leclerc et al., 2022) as data loader. All the experiments were performed with a Resnet50 (He et al., 2016) (except if mentioned otherwise) as backbone. For each model, we use a batch of size 2048 and AdamW (Loshchilov & Hutter, 2019) as optimizer with an adaptive learning rate schedule. We run the training for 100 epochs. For each model, we add as head a small MLP of 3 layers of size 2048 (same dimension as the backbone) with ReLU (Glorot et al., 2011) as activation and batch normalization (Ioffe & Szegedy, 2015). When training different SSL methods, we always used the same set of data augmentations (with cropping, color-jitter, random grayscale, gaussian blur and solarization).

5 Conclusion

We re-framed the much used self-supervised learning trick of removing the top layers of a neural network before using it for a downstream task as a *regularization strategy* coined Guillotine Regularization . We demonstrated how this regularization is needed in SSL due to an inherent misalignment (at least given our current SSL setups) between the SSL training task and the downstream task. We also highlighted the fact that Guillotine Regularization induces increased robustness to suboptimal hyper-parameter selection. Despite, its usefulness, having to rely on a *trick* like Guillotine Regularization to increase performances reveals an important shortcoming of current self-supervised learning methods: the inability to design experimental setups and training criteria that learn structured and truly invariant representations with respect to an appropriate set of factors of variation. As future work, in order to escape from Guillotine Regularization, we should focus on finding new training schemes and criteria that are more *aligned* with the downstream tasks of interest.

References

Srikar Appalaraju, Yi Zhu, Yusheng Xie, and István Fehérvári. Towards good practices in self-supervised representation learning. In *NeurIPS Self-Supervision Workshop*, 2020.

Randall Balestriero and Yann LeCun. Contrastive and non-contrastive self-supervised learning recover global and local spectral embedding methods. *arXiv preprint arXiv:2205.11508*, 2022.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *ICLR*, 2022.

Normand J Beaudry and Renato Renner. An intuitive proof of the data processing inequality. *arXiv preprint arXiv:1107.0740*, 2011.

Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In Isabelle Guyon, Gideon Dror, Vincent Lemaire, Graham Taylor, and Daniel Silver (eds.), *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, pp. 17–36, Bellevue, Washington, USA, 02 Jul 2012. PMLR. URL `https://proceedings.mlr.press/v27/bengio12a.html`.

Yoshua Bengio, Frédéric Bastien, Arnaud Bergeron, Nicolas Boulanger–Lewandowski, Thomas Breuel, Youssouf Chherawala, Moustapha Cisse, Myriam Côté, Dumitru Erhan, Jeremy Eustache, Xavier Glorot, Xavier Muller, Sylvain Pannetier Lebeuf, Razvan Pascanu, Salah Rifai, François Savard, and Guillaume Sicard. Deep learners benefit more from out-of-distribution examples. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 164–172, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL `https://proceedings.mlr.press/v15/bengio11b.html`.

Florian Bordes, Randall Balestriero, and Pascal Vincent. High fidelity visualization of what your self-supervised representation knows about. *CoRR*, abs/2112.09164, 2021. URL `https://arxiv.org/abs/2112.09164`.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Herve Jegou, and Julien Mairal Piotr Bojanowski Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

Rich Caruana. Learning many related tasks at the same time with backpropagation. In G. Tesauro, D. Touretzky, and T. Leen (eds.), *Advances in Neural Information Processing Systems*, volume 7. MIT Press, 1994. URL `https://proceedings.neurips.cc/paper/1994/file/0f840be9b8db4d3fbd5ba2ce59211f55-Paper.pdf`.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020a.

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

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *ICCV*, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Utku Evci, Vincent Dumoulin, Hugo Larochelle, and Michael C Mozer. Head2Toe: Utilizing intermediate representations for better transfer learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 6009–6033. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/evci22a.html`.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL `https://proceedings.mlr.press/v15/glorot11a.html`.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS*, 2020.

Jeff Z HaoChen, Colin Wei, Adrien Gaidon, and Tengyu Ma. Provable guarantees for self-supervised deep learning with spectral contrastive loss. *Advances in Neural Information Processing Systems*, 34:5000–5011, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alexander Shepard, Hartwig Adam, Pietro Perona, and Serge J. Belongie. The inaturalist species classification and detection dataset. In *CVPR*, pp. 8769–8778. IEEE Computer Society, 2018. URL `http://dblp.uni-trier.de/db/conf/cvpr/cvpr2018.html#HornASCSSAPB18`.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/ioffe15.html`.

Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. In *ICLR*, 2022.

Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations, 2022. URL `https://arxiv.org/abs/2204.02937`.

Guillaume Leclerc, Andrew Ilyas, Logan Engstrom, Sung Min Park, Hadi Salman, and Aleksander Madry. ffcv. `https://github.com/libffcv/ffcv/`, 2022. commit xxxxxxx.

Jason D Lee, Qi Lei, Nikunj Saunshi, and Jiacheng Zhuo. Predicting what you already know helps: Provable self-supervised learning. *Advances in Neural Information Processing Systems*, 34:309–323, 2021a.

Kuang-Huei Lee, Anurag Arnab, Sergio Guadarrama, John Canny, and Ian Fischer. Compressive visual representations. In *NeurIPS*, 2021b.

Chunyuan Li, Jianwei Yang, Pengchuan Zhang, Mei Gao, Bin Xiao, Xiyang Dai, Lu Yuan, and Jianfeng Gao. Efficient self-supervised vision transformers for representation learning. In *ICLR*, 2022.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 97–105. JMLR.org, 2015.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL `https://openreview.net/forum?id=Bkg6RiCqY7`.

Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1717–1724, 2014.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *International Conference on Computer Vision*, 2021.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1530–1538, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/rezende15.html`.

Elan Rosenfeld, Pradeep Ravikumar, and Andrej Risteski. Domain-adjusted regression or: Erm may already learn features sufficient for out-of-distribution generalization, 2022. URL `https://arxiv.org/abs/2202.06856`.

Thomas Steinke and Lydia Zakynthinou. Reasoning about generalization via conditional mutual information. In *Conference on Learning Theory*, pp. 3437–3452. PMLR, 2020.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pp. 1–5. IEEE, 2015.

Nenad Tomasev, Ioana Bica, Brian McWilliams, Lars Buesing, Razvan Pascanu, Charles Blundell, and Jovana Mitrovic. Pushing the limits of self-supervised resnets: Can we outperform supervised learning without labels on imagenet? *arXiv preprint arXiv:2201.05119*, 2022.

Trimble Inc. 3d warehouse. `https://3dwarehouse.sketchup.com/`. Accessed: 2022-03-07.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *CVPR*, 2018.

Aolin Xu and Maxim Raginsky. Information-theoretic analysis of generalization capability of learning algorithms. *Advances in Neural Information Processing Systems*, 30, 2017.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL `https://proceedings.neurips.cc/paper/2014/file/375c71349b295fbe2dcdca9206f20a06-Paper.pdf`.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv preprint arxiv:2103.03230*, 2021.

Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL `https://proceedings.neurips.cc/paper/2014/file/3fe94a002317b5f9259f82690aeea4cd-Paper.pdf`.

Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. In *ICLR*, 2022a.

Pan Zhou, Yichen Zhou, Chenyang Si, Weihao Yu, Teck Khim Ng, and Shuicheng Yan. Mugs: A multi-granular self-supervised learning framework. 2022b.

A Datasets

In this work, we use ImageNet (Deng et al., 2009) (Term of license on https://www.image-net.org/download.php) as well as Co3D (https://github.com/facebookresearch/co3d BSD License) for our experiments (Reizenstein et al., 2021). We also used a synthetic 3D dataset that will be described in the next subsection.

A.1 3D models dataset

We will now discuss the dataset used for figure 2. As previously mentioned, this dataset consists of 3D models from 3D Warehouse (Trimble Inc), freely available under a General Model License, and rendered with Blender's Python API. We alter the scene by uniformly varying the latent variables described in table 1

Table 1: Latent variables used to generate views of 3D objects. All variables are sampled from a uniform distribution.

Latent variable	Min. value	Max. value
Object yaw	$-\pi/2$	$\pi/2$
Object pitch	$-\pi/2$	$\pi/2$
Object roll	$-\pi/2$	$\pi/2$
Floor hue	0	1
Spot θ	0	$\pi/4$
Spot ϕ	0	2π
Spot hue	0	1

The variety in the scenes that can be generated is illustrated in figure 6. We can see that each latent variables can significantly impact the scene, giving a significant variety in the rendered images.

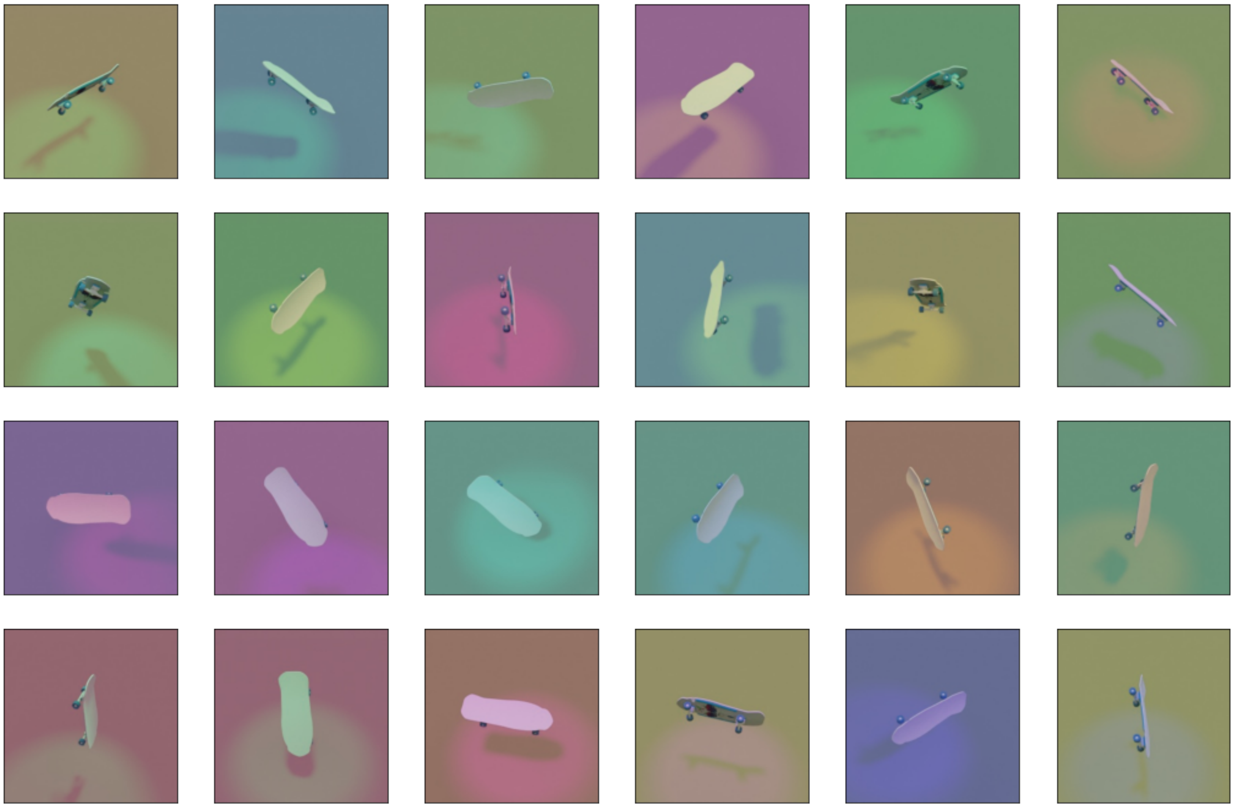


Figure 6: Rendered views of a skateboard generated by randomly sampling latent variables. The influence of each parameter is easily visible, which is expected to make their prediction easier.

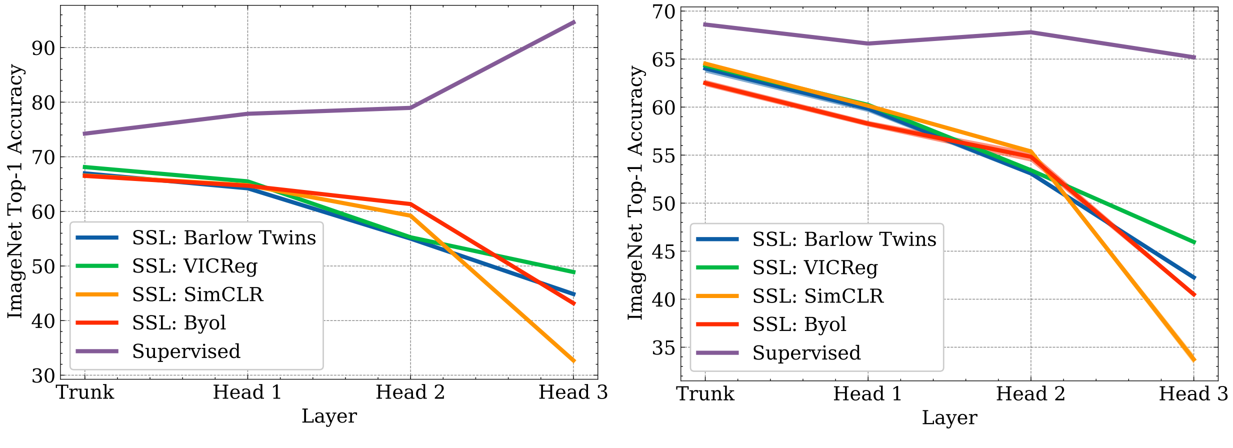
B Reproducibility

Our work does not introduce a novel algorithm nor a significant modification over already existing algorithm. Thus, to reproduce our results, one can simply use the public github repository of the following models: SimCLR, Barlow Twins, VicReg or the PyTorch Imagenet example (for supervised learning) with the following twist: adding a linear probe at each layer of the projector (and backbone) when evaluating the model. However, since many of these models can have different hyper-parameters, or data-augmentations, especially for the SSL models, we recommend to use a single code base with a given optimizer, a given set of data augmentations so that comparisons between models are fair and focus on the effect of Guillotine Regularization. In this paper, except if mentioned otherwise, we use as Head, a MLP with 3 layers of dimensions 2048 each (which match the number of dimensions at the trunk of a Resnet50) along with batch normalizaton and ReLU activations.

C Additional experimental results

In this section, we present additional experimental results. The first one in Figure 7 is an extended version of Figure 1 with additional results on the training set. Figure 8 is a similar setup to the one in Figure 7 where we compared the performances at different layers for SSL methods and a supervised one except that we use a VIT-B instead of a Resnet50. We observe an important gap on the classification performances reached with a linear probe on different layers with the VIT-B when using SSL methods.

In Figure 9 we show the accuracy computed with linear probes trained using projector's representations. This Figure is similar to Figure 4 except that we present the absolute accuracy value instead of the difference in accuracy with respect to the backbone.



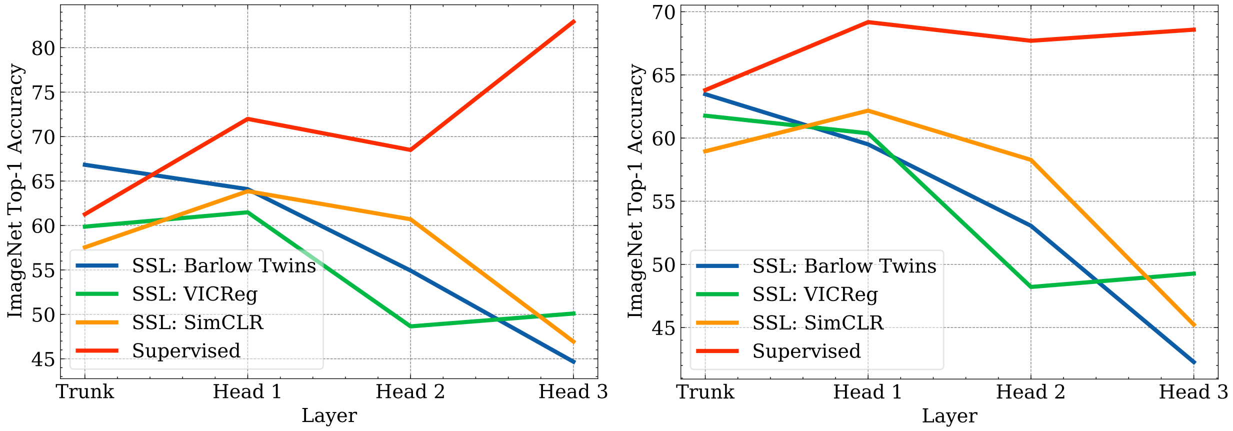
(a) Accuracy on the training set

(b) Accuracy on the validation set

Figure 7: a) An illustration of Guillotine Regularization. During training, a small neural network named the *Head* (also coined as *projector* in the SSL literature (Chen et al., 2020a)) is added on top of another deep network refereed as the *Trunk*. This *Head* can be viewed as a buffer between the training loss and the Trunk that can absorb any bias related to a ill optimisation. When using such network on downstream tasks, we throw away the Head. b) We measure with linear probes the accuracy at different layers on a resnet50 (as Trunk) on which we added a small 3 layers MLP (as Head) for various supervised and self-supervised methods on the training and validation set. For each method, we show the mean and standard deviation across 3 runs (The std between different runs is low). With traditional supervised learning, at a first glance there is a clear overfitting happening which might explain why the performances at the trunk are a slightly bit better than the ones at the projector level. When looking at self-supervised methods, the gap in performances between the linear probe trained at different levels can be as high as 30 points of percentage.

D Limitations

In this work we focused mostly on analyzing the use of Guillotine Regularization in the context of Self-Supervised Learning. However, this kind of regularization might be useful for a variety of other types of training methods which we don't investigate in this paper. We also mostly focus on generalization for classification tasks, but other tasks could also been worth exploring. Finally, it is unclear whether Guillotine Regularization will still be beneficial for very large models (more than 1B parameters) trained on very large dataset.



(a) Accuracy on the training set

(b) Accuracy on the validation set

Figure 8: Same experiment as in Figure 7 but this time, we measure with linear probes the accuracy at different layers on a VIT-B (as Trunk) on which we added a small 3 layers MLP (as Head) for various supervised and self-supervised methods. Since the outputs of the VIT-B has a lower number of dimensions than a Resnet, we added at the trunk of the VIT-B a linear layer with ReLU activation to project into a 2048 dimensional vector. In the supervised learning setting, the best performances are obtained when using the last layers of the model. But, when looking at self-supervised methods, the gap in performances between the linear probe trained at different levels can be as high as 20 points of percentage. Interesting, it seems for the VIT-B that we got the best performances at Head 1 for SimCLR whereas for the ResNet, the best performances were obtained at the Trunk. It is likely that for different architectures, the optimal number of layers on which to apply Guillotine Regularization will vary.

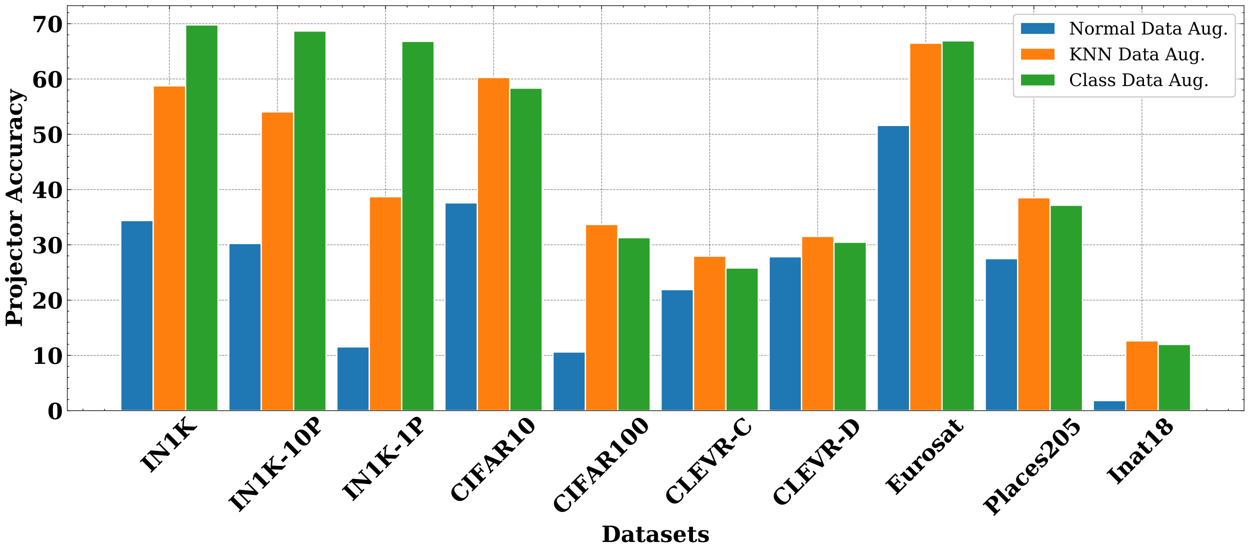


Figure 9: **Projector accuracy with linear probing with different alignment with respect to the classification downstream task.** In this experiment we used SimCLR and we change how the positive pair are defined to better aligned with a classification downstream task. In blue, our baseline, we trained SimCLR with the traditional SSL data augmentations which defines the positive view as two augmentations of a same image. In orange, we use the embedding of a pretrained model to define the positive pair as two nearest neighbor under this pretrained model (while using the same data augmentation as the baseline). In green, we use a supervised class label selection to define the positive example. In this scenario, SimCLR should learn to produces similar embedding to all images belonging to a given class. All three models are trained on ImageNet (IN1K), then we evaluate them with a linear probe across a wide range of downstream tasks at the projector level. When positives pairs are defined as belonging to a given class, there is no misalignment with the imagenet classification downstream task. Thus on ImageNet-1K, ImageNet1k-10P (10% of the training set to train the linear probe) and ImageNet1k-1P (1% of the training set to train the linear probe), we observe that the performances at the projector level are much higher than the ones using the traditional SSL augmentations. Interestingly, the nearest neighbors heuristic also reduce considerably the impact of Guillotine Regularization across several downstream tasks.