Guillotine Regularization: Improving Deep Networks Generalization by Removing their Head

Florian Bordes
Meta AI
Mila, Universite de Montreal

Randall Balestriero
Meta AI

Quentin Garrido
Meta AI
Université Gustave Eiffel,
CNRS, LIGM

Adrien Bardes
Meta AI
Inria

Pascal Vincent
Meta AI
Mila, Universite de Montreal
Canada CIFAR AI Chair

Abstract

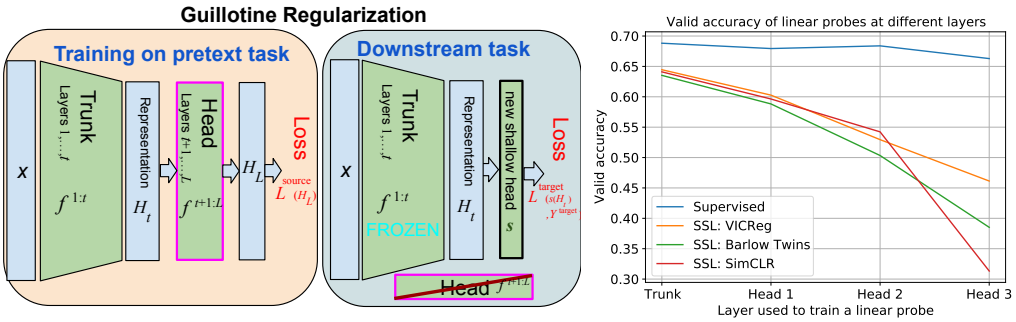
One unexpected technique that emerged in recent years consists in training a Deep Network (DN) with a Self-Supervised Learning (SSL) method, and using this network on downstream tasks but *with its last few layers entirely removed*. This usually skimmed-over *trick* is actually critical for SSL methods 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., 2021; 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 1b).

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

Preprint. Under review.



(a) (b)

Figure 1: a) An illustration of Guillotine Regularization. During training, we add a small neural network named the *Head* 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. 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.

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 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)?*

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 1a). 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 projection head 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., 2021; 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 that confirm 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 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 emphasize 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. Our study also confirms that Guillotine Regularization show important properties with respect to Out Of Distribution 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.

(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 1a.

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.

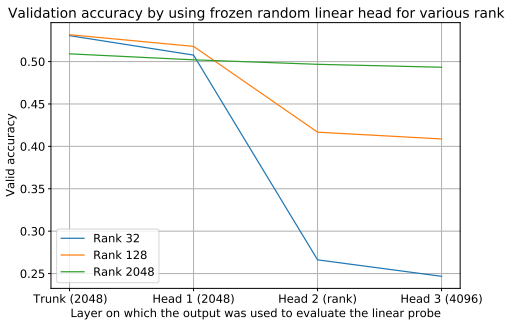


Figure 2: We trained with SimCLR a ResNet50 in which we replaced the traditional 3 layers MLP head by fully linear layers in which the middle layer is a bottleneck of variable sizes. By having a fully linear head, we can modify the size of the linear bottleneck to manipulate the rank of the matrix which characterize how much of information is drop. When having a low rank, the accuracy at the head level drop significantly. With full rank the information is preserved through the network. This result highlight the importance of having either non linearity or bottleneck in the head of the network.

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

Using the data processing inequality (Beaudry and 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 $H_\ell \to H_{\ell+1}$'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.

To highlight this point out, we propose a simple experiment. We take SimCLR, which was sown to benefit from Guillotine Regularization (see Figure 1a right), and progressively turn head $f_\phi^{t+1:L}$ to become one-to-one. This is done by replacing the traditional MLP used as projector head by multiple *linear* layers with a width bottleneck that controls the rank of the head. Whenever the rank is greater or equal to the output dimension of the trunk, the mapping is one-to-one i.e. the same information is contained before and after applying Guillotine Regularization. We report these results in Figure 2, and observe that Guillotine Regularization is only beneficial when the rank is low i.e. when most of the information that was contained at the trunk level was removed.

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 and Raginsky, 2017; Steinke and 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 and 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.

Practical Example with misalignment between source and target distribution for Classification Tasks. Let us consider a toy scenario in which Guillotine Regularization will prove useful. We consider a setting in which source and target tasks are the same classification task, except that we induce a misalignment between the source and target distributions by adding a data-augmentation only during source pretraining. Nothing guarantees that the employed data-augmentation correctly follow the target function level set i.e. the symmetries of the unobserved, ideal mapping f^*. In fact,

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

recall that when employing data-augmentation, we impose the following constraint on the model f

$$f_{\theta,\phi}(\mathcal{T}_\alpha(\boldsymbol{x})) \approx f_{\theta,\phi}(\boldsymbol{x}), \forall \alpha \in \mathbb{A}, \boldsymbol{x} \sim P_X$$

where we applied a parametric sampler $\mathcal{T}$ over a sample $\boldsymbol{x}$ and a data-augmentation parameter space $\mathbb{A}$. For computer vision tasks, $\mathcal{T}_\alpha$ corresponds to a specific data-transformation (DA) e.g. random noise, blurring, or rotation. After pretraining, using a powerful enough model, the level set of $f_{\theta,\phi}$ thus become

$$\mathcal{L}_{\boldsymbol{x}} = \{\boldsymbol{x}' \in \mathbb{X} : f_{\theta,\phi}(\boldsymbol{x}) \approx f_{\theta,\phi}(\boldsymbol{x}')\} = \{\boldsymbol{x}' \in \mathbb{X} \text{ s.t. } \exists \alpha \in \mathbb{A} \text{ for which} \mathcal{T}_\alpha(\boldsymbol{x}) \approx \boldsymbol{x}'\}, \quad (1)$$

where we denote by $\mathbb{X}$ the space of samples with nonzero probability with P_X. The true level set of the true target function f^* we try to model is

$$\mathcal{L}^*_{\boldsymbol{x}} = \{\boldsymbol{x}' \in \mathbb{X} : f^*(\boldsymbol{x}) \approx f^*(\boldsymbol{x}')\}, \quad (2)$$

and is in practice unknown to us. And nothing guarantees that $\mathcal{L}^*_{\boldsymbol{x}}$ and $\mathcal{L}_{\boldsymbol{x}}$ are aligned under some metric. In fact, when the data augmentation is not adapted or too strong, we could have in the worst scenario:

$$\mathcal{L}_{\boldsymbol{x}} \cap \mathcal{L}^*_{\boldsymbol{x}} = \{\} \text{ (or } \{\boldsymbol{x}\} \text{ if } \mathcal{T}_\alpha \text{ can be identity)}, \quad (3)$$

i.e., the level sets imposed by our data-augmentation and the ones of the true function f^* are entirely misaligned. In that scenario, it is quite clear that no classifier (linear or not) put on top of a frozen $f_{\theta,\phi}$ can recover the true mapping f^* regardless of the number of training samples. Hence, in that scenario, only Guillotine Regularization can provide a glimpse of hope to recover some of the information about $\boldsymbol{x}$ to allow the classifier to solve the task at hand.

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

Since a supervised trained network doesn't exhibit any statistically significant gap in performance between the Trunk and the layers at the Head level (Figure 1b), we use it as a baseline to understand in section 4.1 which type of misalignment will be the most impactful on the generalization performances. We use these findings to devise experimental self-supervised learning setup in section to 4.2 in which the gap in performances between the pretext and downstream task is reduced.

4.1 Controlled experiments to gain insights into Guillotine Regularization

To probe the different type of misalignment, we start by investigating a misalignment between the source and target domain while using the same input data distribution. Then, we study how a misalignment between the input data distribution during training and the input data distribution used for testing impact generalization while using the same source and target domain. Finally, we investigate the impact of misalignement between the target and source domain along a misalignement in the data distribution.

Misalignment between the training (source) and downstream (target) task while using the same input data distribution. In this experiment, we use 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)[5]. 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 3 that when training 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. Such results highlight the need to use Guillotine Regularization when there is a shift in the prediction task.

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

[5] 3D objects freely available under a General Model license

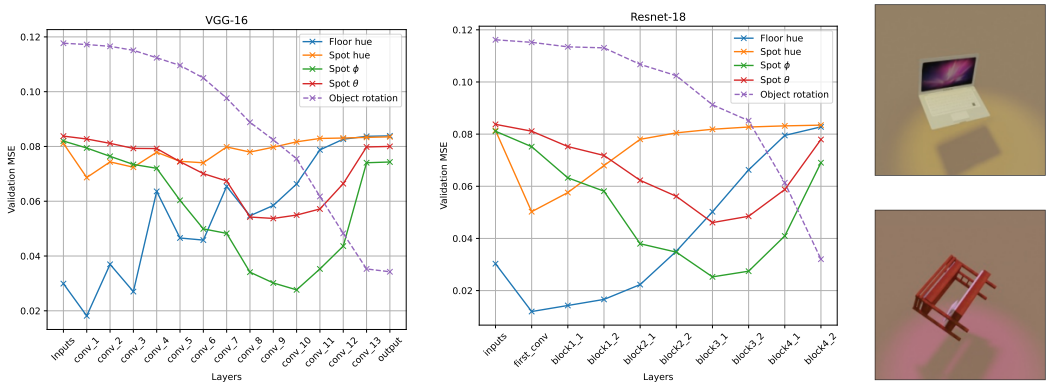
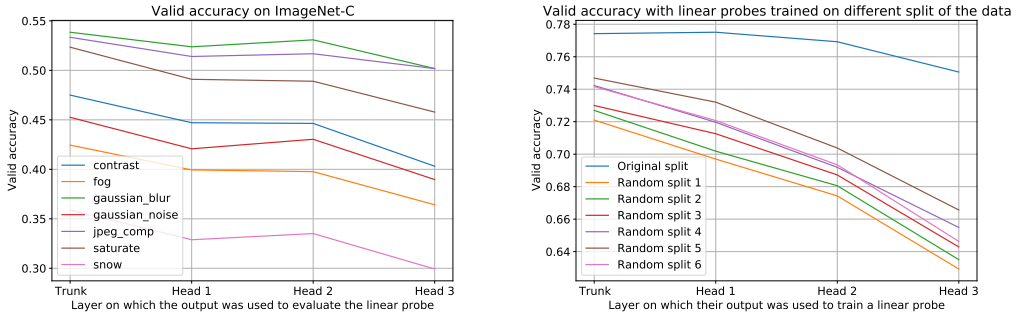


Figure 3: Training a linear regression to predict latent variables from pooled intermediate representations of a network trained to predict 3d rotations of an object (purple line). The data used consists of renderings of 3d objects from 3D Warehouse (Trimble Inc) where we control the floor, lighting and object pose with latent variables, see samples on the right. We compare both a VGG-16 and ResNet-18 to look at the impact of skip connections. We observe on both plot 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. This result highlight the need to use Guillotine Regularization, i.e removing the last layers of the neural network to generalize better on other tasks.

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 and Dietterich, 2019) which is a modified version of the validation set of ImageNet on which different data transformation 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 4a demonstrates that for many transformation that are applied on the data the 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.

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). In our experiment, we trained a ResNet50 with a 3 layer-MLP as head on a random set of 250 classes of ImageNet. Then we trained linear probes at each layers of the neural network over 10 different 250 classes subset of the ImageNet classes. In Figure 4b, 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. When looking at the linear probe trained with the frozen trunk representation (before the head), the accuracy stay high for every random split. This last result shed light on how much Guillotine Regularization is useful to absorb bias during training. In addition of the ImageNet 250 classes experiments, we present similar results with a ImageNet 1000 classes model evaluated over random 1000 classes subset of ImageNet 22K in the appendix.

The above series of experiment is supporting evidence that an inherent misalignment in the training task and downstream objectives make the use of a technique akin to Guillotine Regularization necessary to improve generalization on the downstream task. In the next section, we will demonstrate that these results also translate to SSL.



(a) Misalignment in the data

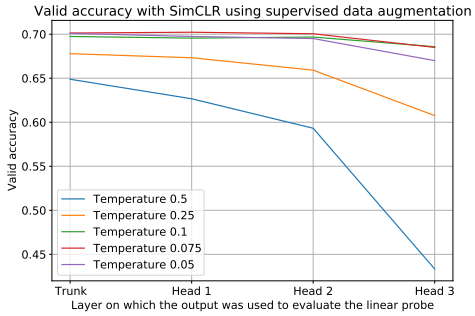
(b) Misalignment in the data and in the objectives

Figure 4: a) ImageNet validation set accuracy over different transformations that induce a misalignment between the training and validation data distribution. We trained a Resnet50 (as Trunk), on which we added a MLP (as Head), on ImageNet in a supervised way using a crossentropy loss (We don't use the validation set for crossvalidation of the model in this instance and use it only for testing purpose). Then, we use ImageNet-C (Hendrycks and Dietterich, 2019) to evaluate the performances of linear probes on frozen representation at different layers in this network. We plot the accuracy on ImageNet-C for each transformations at different layers. For some transformations like gaussian blur, the validation accuracy stay roughly the same from the Trunk to the Head's last layer. However, for some transformations like "fog", the accuracy drop of almost 7% at the head level in comparison with the Trunk. b) We trained a ResNet50 (as Trunk), in which we added MLP (as Head), over a random subset of 250 ImageNet classes. Then, with linear probes, we evaluate the classification performances on different random subset of 250 ImageNet's classes for each layers of this network. When looking at the subset of classes used to to train the model (blue curve), the test performances exhibit a small 3% gap between the Trunk and Head's last layer. However, when using other random subset of classes that is different from the ones used to train the model, we observe a gap of performances that can reach 10% accuracy. This result highlight how much the task's overfitting bias is largely absorb at the head level. In these instances, training a linear probe over the representation is much more suitable than training one at the projector level.

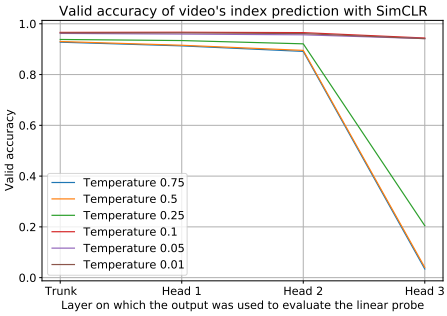
4.2 SSL Benefits from Guillotine Regularization due to the Use of Inappropriate Data-Augmentations During Training

The empirical observations in the previous section indicates that misalignment between the training and downstream task seems to be a key factor in enabling Guillotine Regularization to improve generalization performances of deep networks. 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. To confirm this intuition, we first devise an experimental setup in which we replaced the traditional data augmentation pipeline used in SSL (which consist of using handcrafted augmentation over a specific image to create a set of pairwise positive samples), by using supervised pairs of positives samples that are aligned with the downstream tasks. Additionally, we demonstrate (section 4.2) that Guillotine Regularization also provides a great benefit to SSL methods: test set performance robustness to suboptimal hyper-parameters (section 4.2).

We start our experiments with a downstream ImageNet classification task in mind. Thus, we devise an experimental setup in which we use pair of images that belong to the same class as positive examples and as negative examples, the images who don't belong to the same class. Note that the SSL training criteria will push towards the collapse in the representation space of all the images belonging to the same class while pushing further apart, the different classes cluster. 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 Figure 5a, we measure the performances of linear probes trained at different layers of the SSL model trained with supervised pair of examples. We also performed a grid search over different hyper-parameters since the optimal ones might be different for our setting. As expected, for some hyper-parameters, there is no benefice in using Guillotine Regularization. In this scenario, the performances in validation accuracy stay



(a) SimCLR with ImageNet supervised objective.



(b) SimCLR with Co3D supervised objective.

Figure 5: a) We trained a Resnet50 (as Trunk) on which we added a MLP (as Head), on ImageNet with a SimCLR training criteria (which use a temperature as hyper-parameter) but instead of using the traditionally handcrafted set of data augmentations to define the positive relations between the samples, we use only the ImageNet classes to define these positives relations. This way, the network's training training task is perfectly aligned with the downstream classification task which is an ideal setup to probe how this alignment impact the performances gain of Guillotine Regularization. We measure the performances on the validation set using linear probes that were trained on different layers in the network. We observe two things 1) With the correct temperature, the performances stay almost the same through each layer of the head. This support the hypotheses that Guillotine Regularization is essential in SSL because of the misalignment between the training and downstream task objectives. When we remove this misalignment, doing Guillotine Regularization is not as essential when having the correct temperature. 2) When we don't have the correct hyper-parameters (the temperature for SimCLR), there is still a gain in using Guillotine Regularization since the representations at the backbone level are more robust to the choice of the hyper-parameters than the ones we got at the head level. b) Similar setup as the one in a) except that we use the Co3D dataset (Reizenstein et al., 2021) to perform video's index classification with SimCLR. We use as positives pair images extracted from a given video, then we use as downstream the prediction of the video's index in which the images belongs to. We get the same conclusions as in a).

constant across the head of the network, This imply that there was no bias towards the training objective that hinder the generalisation performance over the validation set.

To further confirm that Guillotine Regularization is needed mostly when there is a misalignment on the training objective and downstream tasks, we did a second experiment by using the Co3D dataset (Reizenstein et al., 2021) which consist of frames extracted from a videos of specific objects. As in the previous experiment, we use a Resnet50 with a SimCLR critera on different frames extracted from a given video as positives pairs. In this instance, the deep network learned to cluster together images belonging to the same video while pushing away images from different videos. During validation we used as downstream task, the prediction of the video's index from a subset of video's frame that were not used during training. In Figure 5b, we observe a similar behavior as the one in 5a, with the correct hyper-parameters, there is almost not gap in performances between the head and the backbone.

Guillotine Regularization Helps Being Robust to Poor Hyper-Parameter Selection Another interesting take from the experiment in Figure 5 is the robustness of the backbone with respect to the choice of the hyper-parameters. For both experiments, the performances at the backbone level are close for whatever choice of temperature for SimCLR. However at the projector level we could get a 20% acc drop with an unlucky choice of hyper-parameters. This highlight an important feature of Guillotine Regularization, when having limited resources to perform a grid search of hyper-parameters, it might be better to add a head in top of your network in order to absorb any bias that could result in an unoptimal choice of hyper-parameters.

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.

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, editors, *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, pages 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, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 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, editors, *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.

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.

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.

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`.

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

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, page 97–105. JMLR.org, 2015.

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*, pages 1717–1724, 2014.

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, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 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*, pages 3437–3452. PMLR, 2020.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pages 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, editors, *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.

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.

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.

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 3. 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.

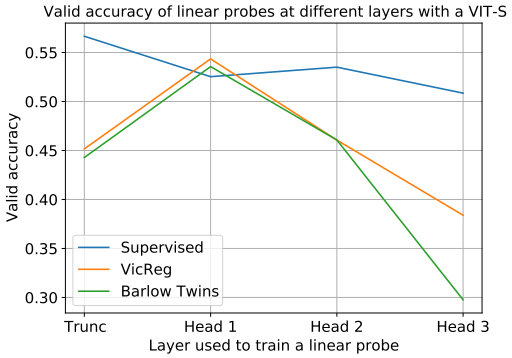


Figure 7: We measure with linear probes the accuracy at different layers on a VIT-S (as Trunk) on which we added a small 3 layers MLP (as Head) for various supervised and self-supervised methods. Since the outputs of the VIT-S has a lower number of dimensions than a Resnet, we added at the trunk of the VIT-S a linear layer with ReLU activation to project into a 2048 dimensional vector. Even with traditional supervised learning, we observe a small gain in using the representation at the Trunk in comparison with the last layer of the Head. 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-S that we got the best performances at Head 1 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.

B Reproducibility

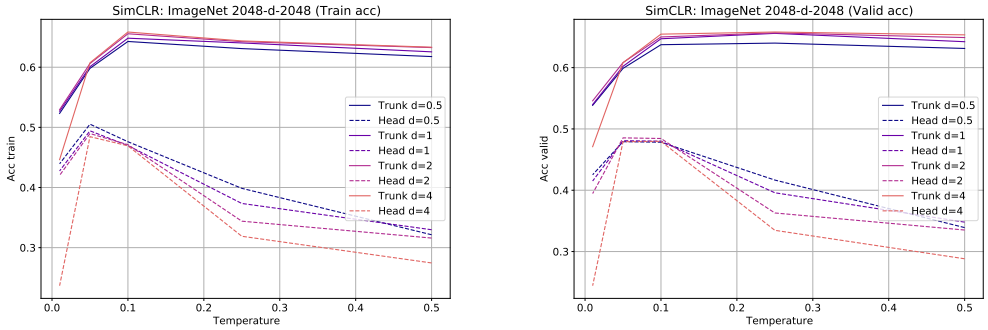
Our work does not introduce a novel algorithm nor a significant modification over already existing algorithm. Thus, to reproduce our results, one can simply use the public github repository of the following models: SimCLR, Barlow Twins, VicReg or the PyTorch Imagenet example (for supervised learning) 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 9 is a similar setup to the one in Figure 1b where we compared the performances at different layers for SSL methods and a supervised one except that we use a VIT-S instead of a Resnet50. We observe an important gap on the classification performances reached with a linear probe on different layers with the VIT-S when using SSL methods.

An additional experiment in Figure 8 shows the training and validation accuracy for SimCLR that we trained with different capacity of the Head and different values of the temperature hyperparameter. This figure shows how much the gap in performance increases as we vary the temperature. While the performances at the Head level are decreasing, the performances at the Trunk remained almost constant. This highlights how much Guillotine Regularization is helping to make the network more robust to the choice of hyper-parameters.

Lastly we followed the same setup as for Figure 4b, but using a supervised network trained on ImageNet 1K with random class split from ImageNet 22K.



(a) Training accuracy.

(b) Validation accuracy.

Figure 8: We trained a ResNet50 with different capacity for the head (d) as well as with different temperatures with a SimCLR criteria. In this Figure, we report only the performances at the Trunk and at the last layer of the Head. In this case, we used the traditional data augmentations for Self-Supervised Learning (Cropping, ColorJitter, Gaussian Blur). We observe for every hyper-parameter, a wide gap in performances on training and validation. However, for temperatures above 0.05, the training and validation accuracy at the Trunk doesn't change much whereas the classification performances at the head level are strongly decreasing.

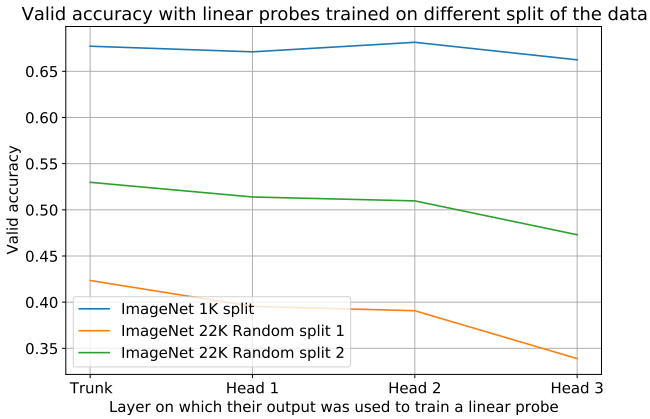


Figure 9: We measure with linear probes the accuracy at different layers, using a Resnet50 (as Trunk) on which we added a small 3 layer MLP (as Head), for a supervised training on ImageNet 1K. The linear probes are trained on a random subset of ImageNet 22K while the model is frozen. As showed in Figure 4b, when using as training class split for the linear probe the same training split as the one used to train the model, we don't observe much differences in term of accuracy between layers. However, when we use different random subset from ImageNet 22K, we observe a gap in performances between each layer of the Head.

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.