HOW TO PREPARE YOUR TASK HEAD FOR FINETUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In deep learning, transferring information from a pretrained network to a downstream task by finetuning has many benefits. The choice of task head plays an important role in fine-tuning, as the pretrained and downstream tasks are usually different. Although there exist many different designs for finetuning, a full understanding of when and why these algorithms work has been elusive. We analyze how the choice of task head controls feature adaptation and hence influences the downstream performance. By decomposing the feature's learning dynamics, we find the key aspect is the training accuracy and loss at the beginning of finetuning, which determines the "energy" available for the feature's adaptation. We identify a significant trend in the effect of changes in this initial energy on the resulting features after finetuning. Specifically, as the energy increases, the Euclidean and cosine distances between the resulting and original features increase, while their dot product (and the resulting features' norm) first increases then decreases. Inspired by this, we give several practical principles that lead to better downstream performance. We analytically prove this trend in an overparamterized linear setting, and verify its applicability to different experimental settings.

1 INTRODUCTION

In the era of deep learning, pretraining a model on a large dataset and adapting it to downstream tasks is a popular workflow. With the help of large amount of data and huge computing resources, the pretrained model can usually provide powerful features which benefit the downstream tasks a lot. Such a framework is proven to be efficient and effective in many domains and tasks, e.g. natural language processing (Kenton and Toutanova, 2019), computer vision (Chen et al., 2020b), graph based learning (Liu et al., 2022), and so on. Although different variants of pretraining and finetuning (FT) methods are widely applied – including direct finetuning, finetuning after linear probing (Kumar et al., 2022), side-tuning (Zhang et al., 2020a), using different learning rates for different layers (Zhang et al., 2021), and more – a detailed understanding of how features are adapted during finetuning under different settings remains elusive.

Our work builds significantly off the analysis of Kumar et al. (2022), who study the interactions between the "task head" (the final layer of the network, usually randomly initialized) and the "backbone" (usually copied from the pretrained model). Kumar et al. claim that the standard finetuning method, randomly initializing a task head then updating all parameters of the whole network, can distort the pretrained features and hence can deteriorate the generalization ability if (as they assume) the previous backbone features were optimal for the downstream task. By analyzing an overparameterized linear model, they prove that linear probing (i.e., only updating the parameters of the task head) first, followed by finetuning the whole network, leads to better performance in their setting.

In this work, we consider less stringent assumptions than they made, and study more practical settings from a different perspective. First, we consider scenarios where the pretrained features are not optimal for the downstream tasks, and so feature adaptation is indeed beneficial. Towards this goal, we consider intermediate cases between the two algorithms studied by (Kumar et al., 2022), which were finetuning with a randomly initialized head (which might distort the features) and a fully-trained task head (which leaves the features unchanged): we consider stopping early during the linear probing process, which leads to features are mildly adapted. To better understand the feature's behavior, we decompose the learning dynamics of the feature vector during finetuning based on "energy" and "direction" of the learning. We discover a non-trivial trend in how this "energy" affects the way that features change from their initialization, which can help us design an appro-

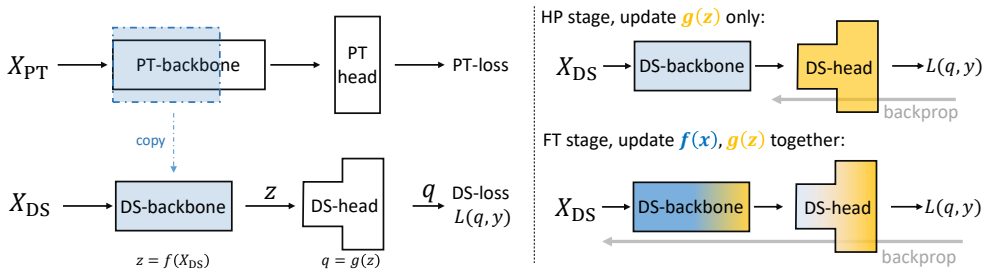


Figure 1: A general example of pretraining, HP and FT procedure. The setting of this paper differs from Kumar et al. (2022) in two important ways: 1) the task head can involve non-linear layers; 2) in FT stage, information exchanges between two parts of the network, like the color mixing phenomenon depicted in the figure.

priate finetuning procedure. Under this framework, we demonstrate that the "unchanged feature" assumption of Kumar et al. (2022) is hard to achieve.

Second, our task heads are not necessarily linear. For example, inspired by the illustrations of Olah et al. (2020), it is reasonable to only preserve the lower layers of the pretrained model, which usually extract some common low-level features. Then, the task head could combine a non-linear head with the last several layers (randomly re-initialized) of the pretrained network. Our analysis can also help explain feature behavior in this setting. We refer to this more general process as "head probing" (HP) rather than linear probing.

Finally, following our insights, we provide some examples of how to enhance downstream performance in practice. In short, we suggest to pay more attention to how much and in what ways we want the features to adapt. The practical suggestions are: 1.) stop head probing earlier, **before** convergence; 2.) use label smoothing during head probing; 3.) consider treating more than just the final layer as the "task head." By analyzing a simplified model and further experimentation, in this work:

- we formalize and explain feature adaptation by decomposing the learning dynamics;
- we find a non-trivial trend in feature adaptation and verify it in many cases;
- and we show how controlling feature adaptation can improve downstream performance.

2 Background

We first clarify the scope of our analysis. We don't consider the details of the pretraining procedure, instead just assuming that there are some well-trained checkpoints for a particular dataset or task. Also, Our formulation is not restricted to classification tasks; our use of the term "label" or "target" should be considered to apply to any form of supervisory signal. Throughout, in parallel to our notation and analysis of a general model, we also consider motivation from the simple overparameterized model of Kumar et al. (2022).

2.1 Problem Setting and Two Stage Training

When training a model for a downstream task, our goal is to find a predictor $f \circ g : \mathbb{R}^d \to \mathbb{R}^k$ that can map a high-dim input signal $\mathbf{x} \in \mathbb{R}^d$ to a task-related prediction $\mathbf{q} \in \mathbb{R}^k$. As depicted by the left-bottom panel of Figure 1, we split the predictor into two parts: the backbone $\mathbf{z} = f(\mathbf{x}; \mathbf{B})$, which maps the input signal to meaningful representations $\mathbf{z} \in \mathbb{R}^h$, and the task head $\mathbf{q} = g(\mathbf{z}; \mathbf{v})$, which gives the prediction vector (e.g. logits in tasks using cross-entropy loss). Usually, the backbone $f(\mathbf{x}; \mathbf{B})$ is parameterized by $\mathbf{B}$ and initialized by copying from a pretrained model. The task head $g(\mathbf{z}; \mathbf{v})$, parameterized by $\mathbf{v}$, is usually randomly initialized and can also have complex structure. For example, $f(\mathbf{x}; \mathbf{B})$ could be the first 3 blocks of a standard ResNet18 (He et al., 2016) while $g(\mathbf{z}; \mathbf{v})$ is a concatenation of the last ResNet block with a linear layer. The loss function $\mathcal{L}(\mathbf{q}, y)$ will be specified during analysis.

In our analysis, we assume that training is composed of two distinct stages, as illustrated in Figure 1. In head probing (HP), we fix f and only update the parameters $\mathbf{v}$ of g. In finetuning (FT), the parameters $\mathbf{v}$ of g and $\mathbf{B}$ of f are updated together. We will primarily analyze how the FT stage is influenced by the structure and initialization of the head parameters $\mathbf{v}$, as an avenue towards asking:

How can we best use a pretrained model to prepare the downstream head for better learning?

2.2 A Simple Linear Regression Problem

Besides the general model mentioned above, we also study a simple overparameterized linear regression (or equivalently, a binary classification) special case to get more insights. In this problem, after stacking N input signals to $X = [\mathbf{x}^{(1)}, ..., \mathbf{x}^{(N)}]^\top \in \mathbb{R}^{N\times d}$, the loss function can be written

$$\mathcal{L}_{\mathbf{B},\mathbf{v}} = \frac{1}{2}\|X\mathbf{B}^\top\mathbf{v} - Y\|_2^2, \tag{1}$$

where $\mathbf{B} \in \mathbb{R}^{h\times d}$, $\mathbf{v} \in \mathbb{R}^h$, and $Y \in \mathbb{R}^N$. That is, $k = 1$, $\mathbf{z} = \mathbf{B}\mathbf{x}$, $q = \mathbf{z}^\top\mathbf{v}$, and $\mathcal{L}(q, y) = \frac{1}{2}(q-y)^2$.

2.3 What to Expect During Adaptation

Compared with training a randomly initialized model, downstream adaptation needs more care. One reason is that copying the parameters from a pretrained model to $f(\mathbf{x}; \mathbf{B})$ inherits all the pretrained information, including the bias and noise in the pretraining task. Furthermore, as mentioned by Kumar et al. (2022) and Du et al. (2018), $f(\mathbf{x}; \mathbf{B})$ is tied to $g(\mathbf{z}; \mathbf{v})$ at each time step during FT. Hence before conducting downstream adaptation, we might consider the following question: to what extent do we *want* to change the feature extractor $f(\mathbf{x}; \mathbf{B})$? We consider three cases:

- Strong: the pretrained model overfits, or the downstream data (task) is very different from the pretrained one, so we need substantial feature adaptation.
- Mild: $f(\mathbf{x}; \mathbf{B})$ is reasonably good, but adapting more to the downstream domain is helpful.
- Tiny: we believe $f(\mathbf{x}; \mathbf{B})$ is almost perfect and only want a slight adaptation.

In the rest of the paper, we first analyze how $f(\mathbf{x}; \mathbf{B})$ and $g(\mathbf{z}; \mathbf{v})$ interact with each other, under both general case and the simplified case. Based on these observations, we propose several practical principles to achieve good downstream performance.

3 Interaction between Backbone and Head during Finetuning

Rather than linking the choice of $\mathbf{v}_0$ (the initialized task head at the beginning of FT) to the loss or risk functions, we want to provide a sketch of how $f(\mathbf{x}; \mathbf{B})$ and $g(\mathbf{z}; \mathbf{v})$ interact during FT, which depicts how $\mathbf{z}$ change accordingly. Although this indirect analysis cannot provide any performance guarantee, knowing *how* $\mathbf{z}$ *changes under different* $\mathbf{v}_0$ will lead us to better HP-FT design.

3.1 Initial Value: Two Extreme Cases

We analyze the behavior of $\mathbf{z}_t^{(0)} = f(\mathbf{x}^{(0)}; \mathbf{B}_t)$, the representation of $\mathbf{x}^{(0)}$ at time t, when the network parameters are updated with a batch $\mathbf{x}^{(1)}, ..., \mathbf{x}^{(N_B)}$ using gradient descent. For cross-entropy loss,[1]

$$\mathbf{z}_{t+1}^{(0)} - \mathbf{z}_t^{(0)} = \frac{\eta}{N_B}\sum_{n=1}^{N_B}\left(\underbrace{\kappa_t^{(0,n)}}_{\text{slow-change}} \cdot \underbrace{\left(\nabla_{\mathbf{z}}\mathbf{q}_t^{(n)}\right)^\top}_{\text{direction}} \cdot \underbrace{\left(\mathbf{e}_{y_n} - \mathbf{p}_t^{(n)}\right)}_{\text{energy}}\right) + \mathcal{O}(\eta^2), \tag{2}$$

where η is the learning rate. $\kappa_t^{(0,n)} = \left(\nabla_{\mathbf{B}}\mathbf{z}_t^{(0)}\right)\left(\nabla_{\mathbf{B}}\mathbf{z}_t^{(n)}\right)^\top$, an $h\times h$ matrix, is the empirical neural tangent kernel (NTK) of the backbone between $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(n)}$ at time t.[2] $\nabla_{\mathbf{z}}\mathbf{q}_t^{(n)}$, a $k \times h$ matrix,

[1] The case for MSE loss, as well as the derivation of this equation, can be found in Appendix A.

[2] In a linear model $\mathbf{z} = \mathbf{B}\mathbf{x}$, $\kappa_t^{(0,n)} = (\mathbf{x}^{(0)})^\top(\mathbf{x}^{(n)}) \cdot I_{h\times h}$ is invariant during FT (i.e., independent of t).

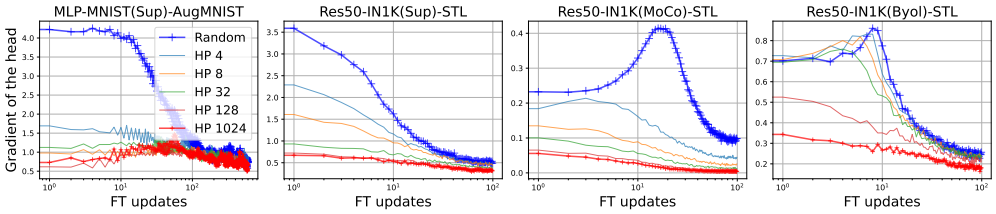


Figure 2: Approximated change of the 'direction' term in Equation (2) when different $\mathbf{v}_0$ is chosen. The titles represent the settings: e.g., 'Res50-IN1K(Sup)-STL' means pretrain a ResNet50 on ImageNet-1k using supervised classification, then transfer to a downstream task on STL10.

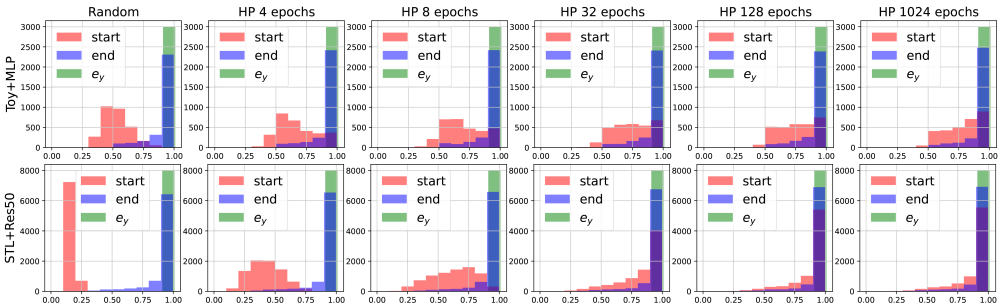


Figure 3: Histograms of $\mathbf{p}_0$, $\mathbf{p}_t$ and $\mathbf{e}_y$ on their dimensions of the correct labels under different $\mathbf{v}_0$.

is the gradient of task head's prediction to the representation vector at time t, which depends on the parameters of current task head $\mathbf{v}_t$.[3] Finally, $\mathbf{p}_t^{(n)} = \mathrm{Softmax}(\mathbf{q}_t^{(n)})$ is the predicted probability vector for input $\mathbf{x}^{(n)}$, and $\mathbf{e}_{y_n}$ is the one-hot indicator for the label.

In this decomposition, the first term is often understood to change only slowly during FT (i.e., the lazy-parameters setting of Chizat et al. (2019)). The second term determines the direction, and the last term provides "energy" for the update of $\mathbf{z}$. Note that we care relatively little about the directional change of the last term, as $\mathbf{p}_t$ is in a k-simplex and $\mathbf{e}_{y_n}$ is at a fixed corner: the direction of $\mathbf{p}_t - \mathbf{e}_{y_n}$ always points to a stable corner. Hence when fine-tuning the network with a randomly initialized head ($\mathbf{v}_0 = \mathbf{v}_{rnd}$), $\mathbf{z}$ would first move in random directions at the beginning of FT (as $\nabla_{\mathbf{z}}\mathbf{q}_t$ changes a lot in early updates), and finally converge to very different locations (as $\mathbf{q}_t$ is quite different from $\mathbf{e}_y$, hence the energy is large). On the contrary, if we use a fully converged HP-head ($\mathbf{v}_0 = \mathbf{v}_{hp}^{\infty}$), $\mathbf{z}$ would only drift slightly during FT (as the second term doesn't change too much and the last term cannot provide enough energy). Figure 11 illustrates the change of $\mathbf{z}_t$ on a toy problem.

The above discussions are supported by experiments in different settings. For the "direction" term in (2), we can observe its change for a linear head as $\|\nabla_{\mathbf{z}}\mathbf{q}_{t+1} - \nabla_{\mathbf{z}}\mathbf{q}_t\|_F^2 = \|\mathbf{v}_{t+1}^\top - \mathbf{v}_t^\top\|^2 = \eta^2\|\nabla_{\mathbf{v}}\mathcal{L}\|^2$. (Non-linear heads have similar behavior in NTK regimes.) As illustrated by Figure 2, when $\mathbf{v}_0 = \mathbf{v}_{rnd}$, $\nabla_{\mathbf{z}}\mathbf{q}_t$ changes a lot at the beginning of FT, which makes different $\mathbf{z}^{(n)}$ change in inconsistent directions. When $\mathbf{v}_0 = \mathbf{v}_{hp}^{1024}$, the direction term changes only a little through fine-tuning. Compare the supervised and unsupervised pretraining cases: although the curves behave differently when the task head is less probed, longer HP always makes the term more stable.

For the "energy" term, we can count the predicted probability of the correct class, i.e., $[\mathbf{p}^{(n)}]_{y_n}$, for each sample. In Figure 3, we show histograms of $\mathbf{p}_0$, $\mathbf{p}_t$ and the ideal $\mathbf{e}_y$ for different $\mathbf{v}_0$. The "energy" is the gap between $\mathbf{p}_t$ and $\mathbf{e}_y$, which is huge for $\mathbf{v}_{rnd}$ but tiny for $\mathbf{v}_{hp}^{1024}$. However, in many practical applications, it is hard to obtain zero energy (unlike the assumption of Kumar et al. (2022)): even with high accuracy there will always be some gap between $\mathbf{p}_t$ and $\mathbf{e}_y$ by definition, and moreover the training accuracy after HP is sometimes far from perfect (see Section 4).

[3] In a linear model $\mathbf{q} = \mathbf{z}^\top\mathbf{v}$, we have $\nabla_{\mathbf{z}}\mathbf{q}_t^{(n)} = \mathbf{v}^\top$, which is independent of n.

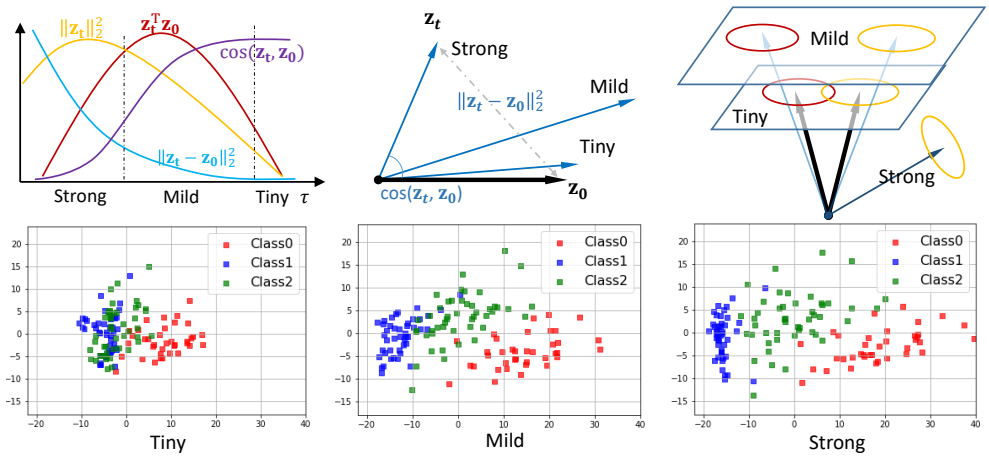


Figure 4: How $\mathbf{z}_t$ changes from $\mathbf{z}_0$. The ellipses in the last panel represent the scattered features. The scatter plots are the PCA projection of the resulting $\mathbf{z}_t$ after finetuning (in the toy MLP settings). It is clear that the mild adaptation can make the features more separable.

3.2 Initial Value: Intermediate Case

Using $\tau = 0$ epochs of probing may result in too much random change to the backbone features $\mathbf{z}$, while fully updated head, $\tau = \infty$, keeps the backbone features nearly static. Thus, in settings where we would like to mildly adapt the features, it is natural to consider a partially-trained head process, i.e. an intermediate value of τ. To quantitatively characterize the change of $\mathbf{z}$, we directly analyze four quantities related to the similarity between the features $\mathbf{z}_0$ before finetuning and $\mathbf{z}_t$ afterwards: $\|\mathbf{z}_t - \mathbf{z}_0\|_2^2$, $\|\mathbf{z}_t\|_2^2$, $\mathbf{z}_t^\top \mathbf{z}_0$, and $\cos(\mathbf{z}_t, \mathbf{z}_0)$. We begin with analysis for a simplified model, which is also illustrated in the first panel of Figure 4.

Proposition 1 (Informal)**.** *In an overparameterized two-layer linear model, when the adaptation energy at the begining of finetuning increases (e.g., when the head is trained using a smaller τ), $\|\mathbf{z}_t - \mathbf{z}_0\|_2^2$ monotonically decreases while $\|\mathbf{z}_t\|_2^2$ and $\mathbf{z}_t^\top \mathbf{z}_0$ exhibit a quadratic trend.*

The details of the analysis are given in Appendix E. With this information, we can infer the behavior of $\mathbf{z}_t$ and sketch that in the second panel. In the "tiny" case, when $\mathbf{v}_{hp}^\tau$ is fully converged ($\tau \to \infty$), the features are almost stable during finetuning; this only works well when the pretrained features are perfect for the downstream task. In the "mild" case, when τ is reasonably large, the resulting $\mathbf{z}_t$ will be stretched ($\|\mathbf{z}_t\|_2^2$ increases) in a similar direction (cosine changes little). See the example in the third panel of Figure 4. This kind of $\mathbf{z}$ can make the original overlapped features become separable after adapting to a similar manifold. Hence the mild-case will make the features adapt more without changing the direction too much, which is desirable in most transfer learning scenarios. For the "strong" case, where we only HP for a few updates or simply use a random head, $\mathbf{z}_t$ will change in an unpredictable way, especially for the early updates. Although the fine-tuned model may generalize well, $\mathbf{z}_t$ might be quite different from $\mathbf{z}_0$. Thus, if we believe the pretrained feature are much too specific to the pretraining dataset, the "strong" case is a reasonable choice.

3.3 Backbone Depth and Head Capacity

Besides how long we train the task head, the structure and capacity of the head also influence the training prediction after this stage, and hence influence the adaptation of $\mathbf{z}_t$. We briefly discuss the trend here and verify them in experimental sections.

For the backbone depth, there is a common belief in the deep learning community that the lower layers extract fundamental features (like edges, curves, texture) while the higher layers learn more semantic features (like dog heads or wheels) (Baldock et al., 2021; Olah et al., 2020). Hence if we believe the downstream task treats the low-level features beneficial while the high-level features

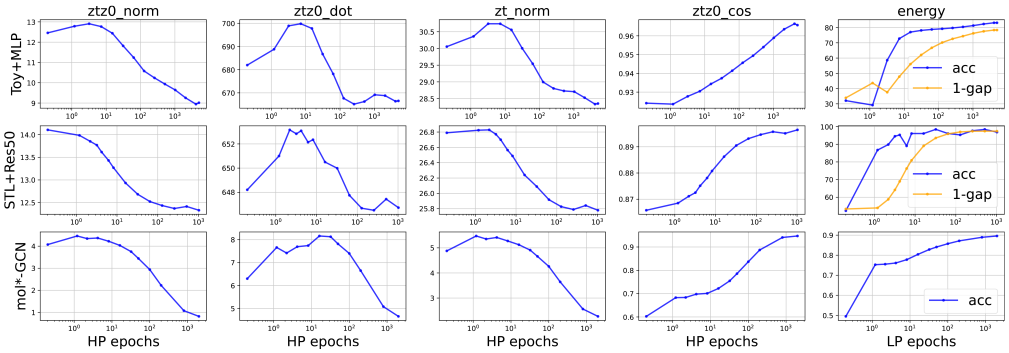


Figure 5: Experimental verification of the trend proposed in Figure 4. More results in Appendix C.

harmful, re-initializing the higher layers (like the fortuitous forgetting mentioned by Zhou et al. (2022)) and incorporating them as part of the task head can be a good solution.

Furthermore, the head's capacity also plays an important role in the HP-FT framework. When using a low-capacity head (e.g., a linear head, which is a $h \times k$ matrix), it might be hard to preserve the pretrained $\mathbf{z}_0$ even a very large τ is chosen, as the head cannot achieve a high enough training accuracy to decrease the "energy" in HP. On the other hand, if the capacity of the task head is much bigger than the backbone, the information copied from the pretrained network might be easily distorted. For example, consider using only the first block of a pretrained ResNet18 as the backbone and concatenating the other 3 blocks with a 10-layer wide MLP. If this huge task head is randomly initialized, the information contained in the backbone could be completely rewritten, as the random changing phase of $\mathbf{z}_t$ can be very long (remember that in (2), $\mathbf{z}_t$ will change in random directions before $\nabla_{\mathbf{v}}\mathbf{q}_t$ becomes less noisy). To avoid this, we can sufficiently train the head in HP stage. However, as the head has larger capacity, we should also be careful about overfitting the head, even if we have a good pretrained backbone.

4 Examples on Real Tasks

We will now turn to experimental verification of the trends we've identified. Our goals here, further clarified in Figure 8, fall into two parts. First, we show how different $\mathbf{v}_0$ determines the adaptation energy and hence influence the resulting $\mathbf{z}_t$. Second, we show how to manipulate the energy (e.g., stop the HP stage earlier, label smoothing only in HP, non-linear head, etc.) to enhance the downstream performance. Although we do not analyze the relationship between the change of $\mathbf{z}_t$ and out-of-distribution performance – this adds too many "moving parts" to the current setting – our analysis is also relevant to finding features which will work in new, unseen domains.

4.1 Relationship between head and feature adaptation

Recall the concepts of "energy" and "direction" from (2). The resulting $\mathbf{z}_t$ is the accumulated influence of these two terms throughout the finetuning stage. In a general HP-FT procedure, we usually first fix the backbone and update the head for τ epochs and then train the whole network for another several epochs. Hence it is reasonable to use the *training* accuracy at the end of head probing (or equivalently, at the beginning of finetuning) to estimate the energy. Intuitively, smaller accuracy means more energy (i.e., more randomness in directions at the early stage of finetuning) and hence makes $\mathbf{z}_t$ change more, as verified by the results in Figure 6. Note that in this figure, we also use the predicting probability gap, i.e., $1 - \mathbb{E}_{\mathbf{x}_n, y_n}[\mathbf{p}^{(n)}]_{y_n}$, to more accurately estimate the energy term (i.e., $\mathbf{e}_y - \mathbf{p}$). [4] This metric is helpful when comparing results under different settings, or when some tricks like label-smoothing-HP (ls-HP, introduced shortly) are applied. The results demonstrate a clear trend that higher HP accuracy (or lower gap) leads to smaller change of $\mathbf{z}_t$ in terms of Euclidian and cosine distances, which aligns well with our analysis in Section 3. Furthermore, our analysis points out a more interesting phenomenon that when moving from full HP to partial HP,

[4] Recall how we generate results in Figure 3, which applies the concept of predicting probability gap.

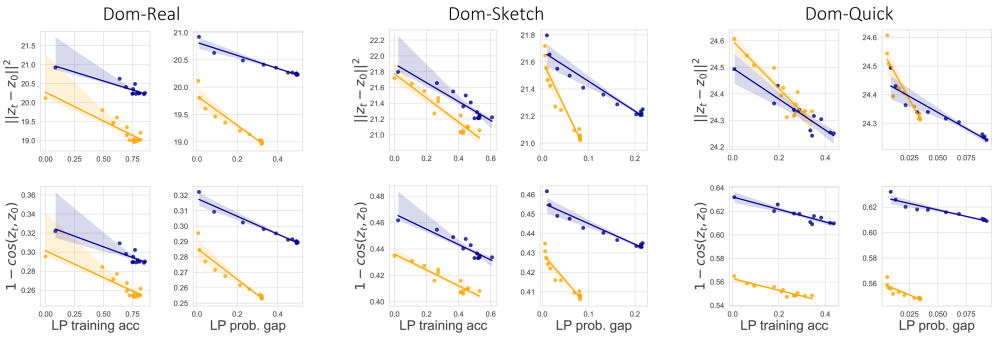


Figure 6: Correlation between HP training accuracy (or the average of predicting probabilities on correct labels) and different distance metrics. The experiments are conducted on three themes in DomainNet. With a pretrained ResNet50 on IN1K in a classification task, each point represents a HP-FT run with different τ, and the FT hyperparameters of all runs are the same. The orange line is the full DomainNet (with 345 classes) while the blue line is its subset (first 200 classes).

the feature vector would first be stretched with slight direction changing, which makes the $\mathbf{z}_t$ vector adapt like the example in Figure 4. This pattern is well-supported by the change of $\mathbf{z}_t^\top \mathbf{z}_0$ and $\|\mathbf{z}_t\|^2$ in Figure 5.

In summary, this part demonstrates the influence of different $\mathbf{v}_0$ on $\mathbf{z}_t$. The take-away messages are: 1.) more energy leads to more adaptation; 2.) too much energy makes the adaptation unstable.

4.2 Relationship between feature adaptation and Downstream Performance

The influence of $\mathbf{z}_t$'s adaptation on downstream performance is more intricate. Generally, the best $\mathbf{z}$'s adaptation depends on how different the pretraining and downstream tasks are: more adaptation is required if the current features (optimal for the pretraining task) are unsuitable for the downstream task. This discrepancy also influences the potential enhancement we can obtain by manipulating $\mathbf{v}_0$.

In the vision experiments in Figure 7, adapting a ResNet50 to STL10 behaves very differently depending on the pretraining task. In the first column, we see a large τ (small energy) is quite harmful for the downstream performance, because the features pretrained on CIFAR10 are far from optimal. In contrast, the features pretrained on ImageNet all provide good results. However, there are still some subtle differences. The model pretrained on a supervised classification task provides the best overall downstream performance, but large τ is still harmful. In other words, the features pretrained this way might be somewhat too specific to the pretraining dataset, and hence mild adaptation is beneficial. For the unsupervised pretraining cases, Byol (Grill et al., 2020) is less sensitive to the choice of τ, while MoCo (He et al., 2020) behaves more similar to the supervised case. A similar trend occurs in many different settings (see Appendix C), including graph neural networks on a molecular graph dataset (see the last column of this figure).

From these results, we see that mild feature adaptation is beneficial in most of the transfer learning cases, but choosing the best τ is hard: in particular, HP training accuracy or validation accuracy seem to have little connection to final downstream accuracy (see the second row of Figure 7). Remember our original motivation is choosing a $\mathbf{v}_0$ that can provide enough energy for $\mathbf{z}$'s adaption. Hence besides changing τ, we can also use a smoothed label (Müller et al., 2019, e.g.) in HP to control the energy (lsHP). This method is beneficial as it allows for a long HP procedure, stabilizing the "direction" term at the beginning of FT. In Table 1, we see the lsHP method usually gives a comparable performance to the best τ (especially when the HP performance is high), with a default smoothing factor ($\alpha = 0.9$). In Table 2, we see even if the final training accuracy after HP doesn't change, lsHP makes $\mathbf{z}$ adapts more and hence generalize better in the downstream task. To demonstrate the performance gain comes from the feature adaptation (not directly the smoothed label), we conduct an ablation study in Table 4 in Appendix D.

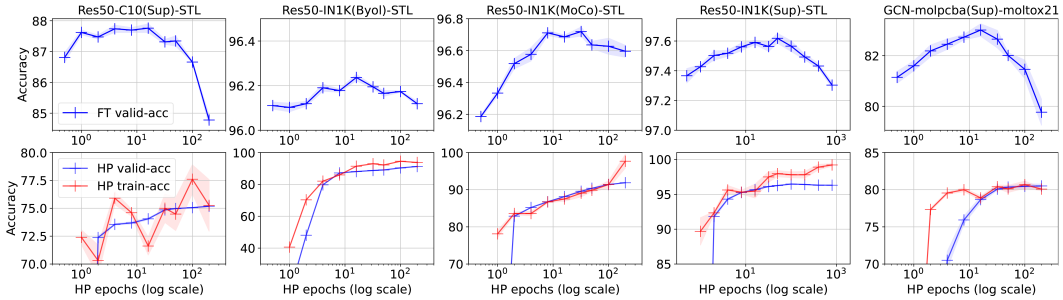


Figure 7: Sweeping τ from 0 to 200 (in 2^n fashion). The valid-accuracy of FT-only setting is the left most point in each panel. The first 4 columns are on vision task while the last one is on graph task.

downstream data:	Flowers			STL10			CIFAR100			Tox21
pretraining task-data:	MoCo-IN	Sup-IN	Sup-C10	MoCo-IN	Sup-IN	Sup-C10	MoCo-IN	Sup-IN	Sup-C10	Sup-pcba
HP200-only	59.711	85.826	7.031	91.889	96.291	75.181	56.397	62.709	11.735	79.121
HP0-FT	76.953	91.295	60.714	96.136	97.452	86.811	80.432	84.736	65.608	81.274
HP200-FT	84.882	91.653	43.973	96.363	97.374	84.778	83.744	84.347	63.992	79.666
HPτ^*-FT	**86.831**	**92.299**	**63.711**	**96.753**	**97.697**	**87.739**	**83.814**	**84.971**	**65.966**	**83.263**
lsHP200-FT	86.946	92.746	42.299	96.639	97.959	85.912	83.412	85.357	65.244	83.853

Table 1: Downstream validation accuracy across different settings. The HPτ^* always performs best. The lsHP can further improve when the HP-only method achieves a high accuracy (hence no enough energy for adaptation), but fails when the pretraining features are not suitable (in Sup-C10 case).

4.3 Achieving zero energy

In all the experiments above, we observe a positive correlation between the HP accuracy and the change of $\mathbf{z}_t$, matching our analysis to the results of Kumar et al. (2022) well. However, another claim in Kumar et al. (2022) needs more consideration. That study assumes a perfect pretrained feature extractor $f(\mathbf{x}; \mathbf{B}^*)$, and argues that using $\mathbf{v}_{hp}^{\infty}$ will make $\mathbf{B}_t$ unchanged during FT, claiming that $\mathbf{v}_{hp}^{\infty}$ can make the gradient of the backbone to be zero. In our terminology, this means the "energy" will be zero during FT, which is almost impossible in practice. With cross-entropy, finite-norm networks can never achieve exactly 1 predictions;[5] more importantly, it is frequently the case that the linear task head cannot achieve perfect training accuracy, as in the Dom-Sketch and Dom-Quick experiments. In these scenarios, we can choose a task head with larger capacity, e.g., an MLP head or combinining several layers (details in Appendix D), to ensure a higher training accuracy in HP. This will be beneficial if we believe $\mathbf{z}_t$ changes too much but increasing τ doesn't help, as illustrated in Table 2.

5 Related Work and Discussions

HP, FT and HP-FT. Head probing and fine tuning are two fundamental algorithms in transfer learning, which have also attracted much discussion and debate. Intuitively, by freezing the pretrained backbone, HP will re-combine the features without distorting them, and hence yields better performance than FT when features are perfect (Peters et al., 2019). However, when pretrain and downstream tasks are very different, adapting the features is important and FT outperforms HP (Chen et al., 2020b; Zhai et al., 2019; He et al., 2022). Combining the strengths of HP and FT, authors of Kumar et al. (2022) demonstrate that HP-FT (i.e., first HP, then FT) yields the best performance on both in-distribution and out-of-distribution cases. Although they provide some theoretical guarantees for the superiority of HP-FT, some of their assumptions are dubious (e.g. see Section 4.3). To shed more lights on the pros and cons of these two methods, this paper analyze in detail *how the features change* under different task-head settings. Specifically, by controlling the HP epochs τ or otherwise influencing the energy term, we can design different types of feature adaptations for different downstream tasks. The relationship between the choice of task head and resulting $\mathbf{z}$'s adaptation is explained using Equation (2), verified by various experiments, and proved in an over-

[5] Common techniques like dropout or weight decay in FT also bring large energy even we have a perfect HP accuracy (as the loss function changes, the energy terms becomes more complex).

		Sim-STL	Sup-STL		Sim-Real	Sim-Sketch	Sim-Quick
Linear-head Baseline	HP-train-acc	96.875	100	Linear-head Baseline	92.676	78.213	64.307
	$1 - cos(\mathbf{z}_t, \mathbf{z}_0)$	0.0044	0.017		0.1269	0.1422	0.1934
	$\|\mathbf{z}_t - \mathbf{z}_0\|^2$	1.792	6.716		5.247	4.752	4.361
	val-acc	93.914	97.374		78.075	61.545	59.703
LS on HP	HP-train-acc	96.875	100	2MLP-head	99.121	94.883	82.354
	$1 - cos(\mathbf{z}_t, \mathbf{z}_0)$	0.014	0.0549		0.1098	0.1104	0.1522
	$\|\mathbf{z}_t - \mathbf{z}_0\|^2$	1.841	9.278		4.942	4.206	3.868
	val-acc	94.304	97.959		78.453	63.773	63.609

Table 2: How ls-HP and larger head influence the feature adaptation. The models are pretrained using SimCLR (Sim) or supervised (Sup) classification on IN1K. Number in blue (red) represent an decrease (increase) compared with its counterpart in the baseline. See more results in Appendix D.

parameterized linear model. Although the paper only analyze these fundamental settings, we believe the analysis provided here can also be combined with (or explain the benefits of) other more complex finetuning mechanisms, like those of Guo et al. (2019); Zhang et al. (2020a); Aghajanyan et al. (2021); Howard and Ruder (2018). For example, a common practice in transfer learning is to use small learning rate for backbone while a large learning rate for task head. From our perspective, this method can weaken the influence of the "noisy direction" term and make the features adapts in a low energy condition.

Backbone and head depth. Besides the influence of the initial value of the head, this paper also discusses the influence of the relative capacity between backbone and head. Chen et al. (2020b) show that a simple MLP head sometimes also brings enhancement. When the backbone only copies the early layers of the pretrained network, as the downstream task might need different levels of features (Olah et al., 2020; Baldock et al., 2021; Zhang et al., 2020b), the later layers and the original head can be combined as the new complex task-head, like the example in Figure 1. Furthermore, in some encoder-decoder style models, like some the models for some language tasks (Peters et al., 2019; Zhu et al., 2019), the head (decoder) might have comparable capacity with the backbone (encoder). In another extreme case, if we use pretrained word2vec features (Mikolov et al., 2013) and plan to FT them in a downstream task, the task head is the whole network, which is much bigger than the embedding layer. We believe the discussion in this paper might also help inspire the design of HP-FT strategy for these practical scenarios.

6 Conclusion

This paper studies how the choice of task heads influence the pretrained features $\mathbf{z}$'s adaptation and hence influence the downstream performance. By decomposing the learning dynamics of $\mathbf{z}$, we find the keys are the energy and direction terms, which highly correlate with the accuracy at the beginning of the FT process (i.e., the accuracy after HP). Hence under a common HP-FT framework, we carefully analyze the relationship between the HP epochs (τ) and the features' adaptation (discrepancy between $\mathbf{z}_t$ and $\mathbf{z}_0$). Different from most existing works, which mainly focus on Euclidean or cosine distance, we further analyze $\mathbf{z}_t^\top \mathbf{z}_0$ and $\|\mathbf{z}_0\|_2^2$ to depict a more comprehensive relationship between τ and $\mathbf{z}_t$, like the examples in Figure 4. This non-trivial trend is strictly proved in an overparameterized linear model and experimentally verified under different practical settings.

Based on these observations, we speculate that a suitable adaptation of $\mathbf{z}_t$ is beneficial when the pretrained features are not perfect for the downstream tasks. Under different experimental settings, we illustrate how to achieve better downstream performance by controlling the adaptations of $\mathbf{z}_t$ (using early-stopping HP, label smoothing during HP, or manipulating the head's capacity).

Finally, there are still many open questions and phenomena not address by this study. For example, methods to quantitatively or even adaptively analyze the discrepancy between the pretrain and downstream tasks would be very useful in knowing what kind of head probing to perform in advance; Li et al. (2020), for instance, have made some progress in this direction. Other remaining questions include incorporating the influence of different regularizations and augmentations, and extending the findings to more general types of training (such as using different learning rates for the backbone and the task head). However, we believe the methods proposed in this paper can help provide a new perspective on understanding feature adaptation, which we hope will aid future work in this area.

REFERENCES

Aghajanyan, A., Shrivastava, A., Gupta, A., Goyal, N., Zettlemoyer, L., and Gupta, S. (2021). Better fine-tuning by reducing representational collapse. In *International Conference on Learning Representations*.

Baldock, R., Maennel, H., and Neyshabur, B. (2021). Deep learning through the lens of example difficulty. *Advances in Neural Information Processing Systems*.

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

Chen, X., Fan, H., Girshick, R., and He, K. (2020b). Improved baselines with momentum contrastive learning. *arXiv preprint*.

Chizat, L., Oyallon, E., and Bach, F. (2019). On lazy training in differentiable programming. *Advances in Neural Information Processing Systems*, 32.

Coates, A., Ng, A., and Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings.

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

Du, S. S., Hu, W., and Lee, J. D. (2018). Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. *Advances in Neural Information Processing Systems*.

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

Guo, Y., Shi, H., Kumar, A., Grauman, K., Rosing, T., and Feris, R. (2019). Spottune: transfer learning through adaptive fine-tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4805–4814.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022). Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., and Li, M. (2019). Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 558–567.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. *ACL*.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. (2020). Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*.

Kenton, J. D. M.-W. C. and Toutanova, L. K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. (2013). 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Kumar, A., Raghunathan, A., Jones, R., Ma, T., and Liang, P. (2022). Fine-tuning can distort pretrained features and underperform out-of-distribution. *International Conference on Learning Representations*.

LeCun, Y. (1998). The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. (2019). Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*.

Li, H., Chaudhari, P., Yang, H., Lam, M., Ravichandran, A., Bhotika, R., and Soatto, S. (2020). Rethinking the hyperparameters for fine-tuning. In *International Conference on Learning Representations*.

Liu, S., Wang, H., Liu, W., Lasenby, J., Guo, H., and Tang, J. (2022). Pre-training molecular graph representation with 3d geometry. In *International Conference on Learning Representations*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Müller, R., Kornblith, S., and Hinton, G. E. (2019). When does label smoothing help? *Advances in neural information processing systems*.

Nilsback, M.-E. and Zisserman, A. (2008). Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. (2020). Zoom in: An introduction to circuits. *Distill*.

Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., and Wang, B. (2019). Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1406–1415.

Peters, M. E., Ruder, S., and Smith, N. A. (2019). To tune or not to tune? adapting pretrained representations to diverse tasks. *ACL*.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. (2018). Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530.

Zhai, X., Puigcerver, J., Kolesnikov, A., Ruyssen, P., Riquelme, C., Lucic, M., Djolonga, J., Pinto, A. S., Neumann, M., Dosovitskiy, A., et al. (2019). A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint*.

Zhang, J. O., Sax, A., Zamir, A., Guibas, L., and Malik, J. (2020a). Side-tuning: a baseline for network adaptation via additive side networks. In *European Conference on Computer Vision*, pages 698–714. Springer.

Zhang, T., Wu, F., Katiyar, A., Weinberger, K. Q., and Artzi, Y. (2020b). Revisiting few-sample bert fine-tuning. In *International Conference on Learning Representations*.

Zhang, T., Wu, F., Katiyar, A., Weinberger, K. Q., and Artzi, Y. (2021). Revisiting few-sample {bert} fine-tuning. In *International Conference on Learning Representations*.

Zhou, H., Vani, A., Larochelle, H., and Courville, A. (2022). Fortuitous forgetting in connectionist networks. *International Conference on Learning Representations*.

Zhu, J., Xia, Y., Wu, L., He, D., Qin, T., Zhou, W., Li, H., and Liu, T. (2019). Incorporating bert into neural machine translation. In *International Conference on Learning Representations*.

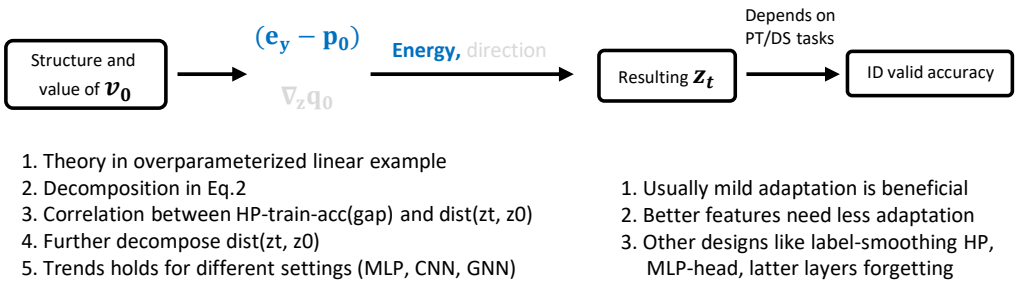


Figure 8: The flow of this paper.

A Decomposition of change of Z

Recall $\mathbf{z} = f_{\mathbf{B}}(\mathbf{x})$, $\mathbf{q} = g_{\mathbf{z}}(\mathbf{z})$ and $\mathbf{p} = \mathrm{Softmax}(\mathbf{q})$. Use $\mathbf{b} \in \mathbb{R}^{hd\times 1}$ to represent the vector form of $\mathbf{B} \in \mathbb{R}^{h\times d}$. The MSE loss on one sample is $\mathcal{L}_{mse}\left(\mathbf{q}^{(n)}, \mathbf{e}_{y_n}\right) = \frac{1}{2}\|\mathbf{q}^{(n)} - \mathbf{e}_{y_n}\|_2^2$. The cross-entropy loss on one sample is $\mathcal{L}_{ce}\left(\mathbf{q}^{(n)}, \mathbf{e}_{y_n}\right) = \mathcal{H}\left(\mathbf{e}_{y_n}, \mathbf{q}^{(n)}\right)$, where $\mathcal{H}(\mathbf{p}, \mathbf{q}) = -\sum_k p_k \log q_k$ is the cross entropy for categorical distribution. Using 1st Taylor expansion, we have:

$$\underbrace{\mathbf{z}_{t+1}^{(0)} - \mathbf{z}_t^{(0)}}_{h\times 1} = \underbrace{\nabla_{\mathbf{b}} \mathbf{z}_t^{(0)}}_{h\times hd} \cdot \underbrace{(\mathbf{b}_{t+1} - \mathbf{b}_t)}_{hd\times 1} + \mathcal{O}\left(\|\mathbf{b}_{t+1} - \mathbf{b}_t\|^2\right). \tag{3}$$

We then calculate $\mathbf{b}_{t+1} - \mathbf{b}_t$ assuming the parameters are updated in batch-SGD:

$$\underbrace{\mathbf{b}_{t+1} - \mathbf{b}_t}_{hd\times 1} = -\frac{\eta}{N_B} a \sum_{n=1}^{N_B} \left(\underbrace{\nabla_{\mathbf{b}} \mathcal{L}\left(\mathbf{q}_t^{(n)}, \mathbf{e}_{y_n}\right)}_{1\times hd} \right)^\top \tag{4}$$

$$= -\frac{\eta}{N_B} \sum_{n=1}^{N_B} \left(\underbrace{\nabla_{\mathbf{q}} \mathcal{L}\left(\mathbf{q}_t^{(n)}, \mathbf{e}_{y_n}\right)}_{1\times k} \cdot \underbrace{\nabla_{\mathbf{z}} \mathbf{q}_t^{(n)}}_{k\times h} \underbrace{\nabla_{\mathbf{b}} \mathbf{z}_t^{(n)}}_{h\times hd} \right)^\top \tag{5}$$

$$= -\frac{\eta}{N_B} \sum_{n=1}^{N_B} \underbrace{\left(\nabla_{\mathbf{b}} \mathbf{z}_t^{(n)}\right)^\top}_{hd\times h} \underbrace{\left(\nabla_{\mathbf{z}} \mathbf{q}_t^{(n)}\right)^\top}_{h\times k} \underbrace{\left(\nabla_{\mathbf{q}} \mathcal{L}\left(\mathbf{q}_t^{(n)}, \mathbf{e}_{y_n}\right)\right)^\top}_{k\times 1} \tag{6}$$

For different loss functions, we have different expressions for the last term:

$$\nabla_{\mathbf{q}} \mathcal{L}_{mse}\left(\mathbf{q}_t^{(n)}, \mathbf{e}_{y_n}\right) = \left(\mathbf{q}_t^{(n)} - \mathbf{e}_{y_n}\right); \quad \nabla_{\mathbf{q}} \mathcal{L}_{ce}\left(\mathbf{q}_t^{(n)}, \mathbf{e}_{y_n}\right) = \left(\mathbf{p}_t^{(n)} - \mathbf{e}_{y_n}\right), \tag{7}$$

where $\mathbf{q}$ is the logits and $\mathbf{p} = \mathrm{Softmax}(\mathbf{q})$ is the predicting probability. We see these two kinds of loss have similar expression on this term, so we only explain the cross-entropy version (as it is more common in practices) in the main content.

Using the above expressions, we can first bound the high-order term (cross-entropy version):

$$\mathcal{O}(\|\mathbf{b}_{t+1} - \mathbf{b}_t\|^2) = \mathcal{O}(\eta^2 \|\nabla_{\mathbf{z}} \mathbf{q}_t^{(n)}\|_{op}^2 \cdot \|\nabla_{\mathbf{b}} \mathbf{z}_t^{(n)}\|_{op}^2 \cdot \|\mathbf{q}_t^{(n)} - \mathbf{e}_{y_n}\|_{op}^2) = \mathcal{O}(\eta^2), \tag{8}$$

as long as the hyperparameters are appropriately chosen and the loss doesn't blow up (or gradient clipping is applied) in FT stage.

Finally, combining all the expressions, Equation (3) can be rewritten as:

$$\underbrace{\mathbf{z}_{t+1}^{(0)} - \mathbf{z}_t^{(0)}}_{h\times 1} = -\frac{\eta}{N_B}\sum_{n=1}^{N_B}\underbrace{\left(\nabla_{\mathbf{b}}\mathbf{z}_t^{(0)}\right)}_{h\times hd}\underbrace{\left(\nabla_{\mathbf{b}}\mathbf{z}_t^{(0)}\right)^\top}_{hd\times h}\underbrace{\left(\nabla_{\mathbf{z}}\mathbf{q}_t^{(n)}\right)^\top}_{h\times k}\underbrace{\left(\nabla_{\mathbf{q}}\mathcal{L}_{ce}\left(\mathbf{q}_t^{(n)}, \mathbf{e}_{y_n}\right)\right)^\top}_{k\times 1} + \mathcal{O}(\eta^2) \quad (9)$$

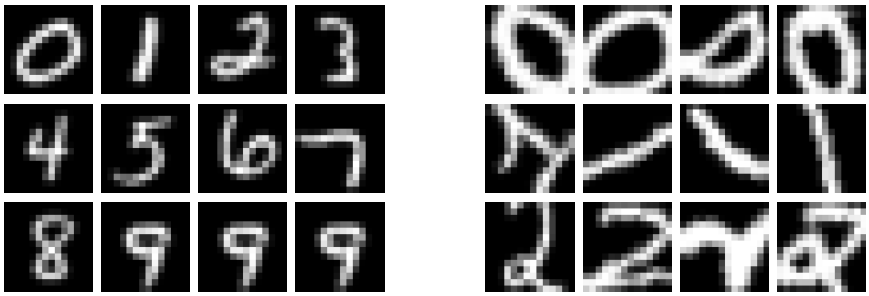
$$= \frac{\eta}{N_B}\sum_{n=1}^{N_B}\underbrace{\kappa_t^{(0,n)}}_{h\times h}\cdot\underbrace{\left(\nabla_{\mathbf{z}}\mathbf{q}_t^{(n)}\right)^\top}_{h\times k}\underbrace{\left(\mathbf{e}_{y_n} - \mathbf{p}_t^{(n)}\right)}_{k\times 1} + \mathcal{O}(\eta^2), \quad (10)$$

which is the same with Equation (2) in the main context.

B Experimental Settings

B.1 Toy case

In the toy setting, we first pretrain a 4+1 (4 layers of backbone and 1 layer of task head) layers MLP on the full MNIST dataset (LeCun, 1998). The hidden width is 128 and relu activation is applied for each layer in the backbone. In the downstream task, we only consider the first three classes and randomly select 1000 samples for each class (i.e., 3*1000 samples in total). We also apply random rotation and center crop augmentations to simulate the distribution shift in downstream task, like the right panel in Figure 9. The downstream linear task head is then a 128*3 matrix (weights) and a 128*1 vector (bias). To ensure a fine-grained observation of feature's adaptation, we use a small constant learning rate (10^{-4}) in all the experiments. The FT epoch is 50, and the HP epoch ranges from 0 to 50,000 (50,000 is simulating the $\mathbf{v}_{hp}^{\infty}$ case, usually train 0 to 1,000 epochs).



(a) Pretrain dataset. (b) Downstream (LP/FT) dataset.

Figure 9: Dataset for toy experiments. We use full MNIST dataset to pretrain the network. The downstream dataset is a subset of MNIST (only 0, 1 and 2) applying random rotation and zoom-in.

B.2 Practical Settings

Besides the simple MNIST and MLP, we also verify our analysis in many real settings. To demonstrate the generality of our findings, we conduct experiments on image classification and molecular graph classification tasks. The qualitative trend is quite consistent across different tasks and networks.

Specifically, we consider ResNet18/34/50 for image classification task. We pretrain the ResNet on CIFAR10 using common recipe (10^{-2} learning rate with a cosine scheduler (He et al., 2019), $5{*}10^{-4}$ weight decay, simple augmentations like random flipping and cropping, etc.). For the model pretrained on ImageNet, we directly download the checkpoints from opensource implementations. Due to the time and computing resource limitation, for vision tasks, we only consider MoCo (He et al., 2020), Byol (Grill et al., 2020), SimCLR (Chen et al., 2020a), and supervised classification tasks for pretraining. For the graph dataset case, we consider a fundamental 5-layers graph convolutional

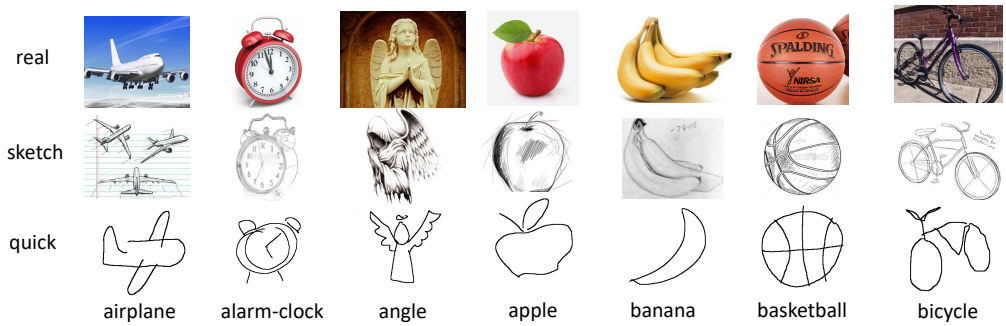


Figure 10: DomainNet dataset. Only three domains are used: 'real' is similar to IN1K, 'sketch' is less sensitive to color and background, 'quick' only contains some lines, which is quite different from IN1K.

	# train	# test	# class	Comments
MNIST	60,000	10,000	10	Toy downstream use a subset, as in Figure 9.
STL10	5,000	8,000	10	Coates et al. (2011)
CIFAR10	50,000	10,000	10	Krizhevsky et al. (2009)
CIFAR100	50,000	10,000	100	Krizhevsky et al. (2009)
Flowers102	6,149	2,040	102	Nilsback and Zisserman (2008)
StanfordCars	8,144	8,041	196	Krause et al. (2013)
Dom-real	<20,000	<4,000	200 (345)	A subset, original 345 classes and more samples.
Dom-sketch	<20,000	<4,000	200 (345)	Less sensitive to color and texture.
Dom-quick	<20,000	<4,000	200 (345)	Only contains simple curves. (Peng et al., 2019)
ImageNet-1K	1,281,167	–	1000	Deng et al. (2009)
ogbg-moltox21	6,272	1,742	12	Wu et al. (2018)
ogbg-molhiv	32,901	8,226	regression	Hu et al. (2020)
ogbg-molpcba	437,929	–	128	Hu et al. (2020)

Table 3: Datasets (vision and molecular graph) used in experiments.

network (GCN, (Kipf and Welling, 2017)), the hyper-parameters and other detailed settings can be found in our github repo. The datasets applied in this paper are listed in Table 3.

For most of the experiments, the HP learning rate is chosen from $\{10^{-2}, 3 * 10^{-3}, 10^{-3}\}$, which is generally larger than the FT learning rate (chosen from $\{10^{-3}, 3 * 10^{-4}, 10^{-4}\}$). When demonstrating the behavior of $\mathbf{z}$, like the experiments in Figure 5, we switch off the weight decay in the FT case, because such a regularization can change the form of Equation (2). The training epochs for head probing or finetuning can be different when datasets changes (as the number of training samples varies a lot). A general principle is that the maximum τ should make the HP training accuracy converge, while the finetuning epochs should make the training accuracy converge when $\tau = 0$. For example, if downstream tasks is CIFAR10, which has 50,000 training samples and only 10 classes, finetune 50 epochs is enough. But CIFAR100, which also has 50,000 training samples but with more classes, 100 finetuning epochs are required. For the Flowers102, which only has 6,149 training samples, we finetune the network for 1,000 epochs.

C More experiments on the change of z

In Equation (2), we decompose the learning dynamics of $\mathbf{z}_t$ into three parts: relatively stable kernel, direction, and energy. Based on this, we can expect a larger adaptation when the initial energy is large. Furthermore, to get a more precise description of $\mathbf{z}_t$'s adaptation, we analyze a simple overparameterized linear model in Appendix E. We show that as the initial prediction gap (i.e., energy) increases, the resulting $\mathbf{z}_t$ will first be stretched with a small direction change (compared with the original $\mathbf{z}_0$). If the initial energy keeps increasing, the resulting $\mathbf{z}_t$ might be more and more dissimilar to the original one. We use four distance related quantity to describe this trend, and verify it under many different settings. In short, the trends are: 1.) more energy leads to larger $\|\mathbf{z}_t - \mathbf{z}_0\|_2^2$

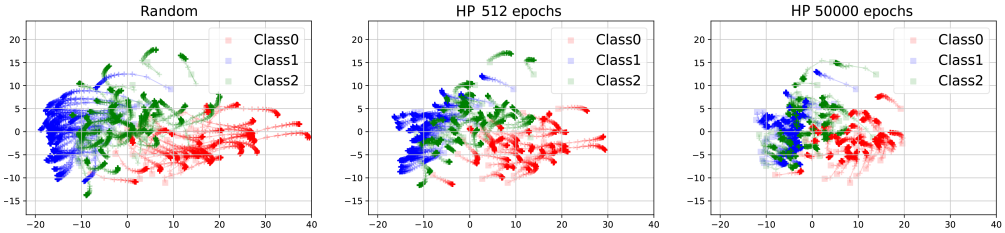


Figure 11: Change of $\mathbf{z}$ during FT under a toy setting. Using transparent square markers, we first draw the 2-D projections (the first two components in PCA) of 100 randomly selected $\mathbf{z}_0^{(n)}$. Then keeping the eigen-directions of this PCA, we project $\mathbf{z}_t^{(n)}$ of different t and encode t by transparency. The converged $\mathbf{z}$ is represented by a cross marker with full transparency. It is clear that FT using $\mathbf{v}_{rnd}$ makes $\mathbf{z}$ change a lot, while $\mathbf{v}_{hp}^{50000}$ doesn't change $\mathbf{z}$ too much.

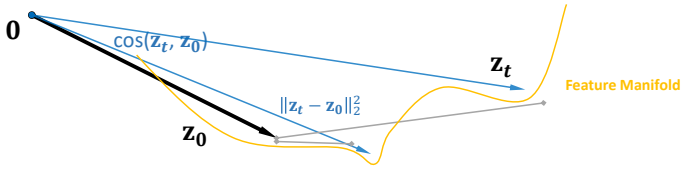


Figure 12: Explanation of why small change of $\mathbf{z}_t^\top \mathbf{z}_0$, $cos(\mathbf{z}_t, \mathbf{z}_0)$ can influence a lot.

and smaller $cos(\mathbf{z}_t, \mathbf{z}_0)$; 2.) $\mathbf{z}_t^\top \mathbf{z}_0$ and $\|\mathbf{z}_t\|_2^2$ has a quadratic shape when energy decrease. Figure 11 provides an example of how $\mathbf{z}_t$ changes when different τ is chosen.

Before diving deep into these examples, we provide some general commands. First, we do not expect the experimental results to align perfectly with the examples in Figure 4, because too many designs (like dropout, data augmentation, SGD noise, locking in strange local minima, learning rate schedule, etc.) can influence the shape of the curve, hence we believe observing specific trends in many cases is already a strong support for our analysis. Second, one might be curious about why in some cases, the ranges between $\mathbf{z}_t^\top \mathbf{z}_0$ and $\cos(\mathbf{z}_t, \mathbf{z}_0)$ are so small. Hence in Figure 12, we provide an illustration of why this happens: as the dot product measures two long vectors (remember they have the same origin), a small change of $\mathbf{z}_t^\top \mathbf{z}_0$ can make a big difference. A large range of $\mathbf{z}_t^\top \mathbf{z}_0$ and $\cos(\mathbf{z}_t, \mathbf{z}_0)$ is a sign that $\mathbf{z}_t$ already moves to another basin, which is the strong-case in Figure 4. Third, in Appendix E, the change of $\mathbf{z}$ is linked to $\mathbf{q}_0 - \mathbf{e}_y$ under an MSE loss, while in these experiments, we use τ as our x-axis and consider a cross-entropy loss. Combining with the NTK approximation applied in this ideal model, the experimental trends might not be exactly the same with the theoretical ones.

Then, in Figure 13, Figure 14 and Figure 15, we demonstrate the trend when the model is pretrained on different datasets, on different tasks, and when the downstream tasks are different, respectively. Besides the general trends, i.e., decreasing $\|\mathbf{z}_t - \mathbf{z}_0\|_2^2$, increasing $\cos(\mathbf{z}_t, \mathbf{z}_0)$, quadratic $\mathbf{z}_t^\top \mathbf{z}_0$ and $\|\mathbf{z}_t\|_2^2$, we can observe another interesting phenomenon when the model is pretrained using different tasks. See Figure 14, in Byol and SimCLR, sometimes we might get a very big $\mathbf{z}_t^\top \mathbf{z}_0$ (together with a very small $\cos(\mathbf{z}_t, \mathbf{z}_0)$), which makes it hard to observe any quadratic trend in some metrics. But in the supervised pretraining case (no matter what dataset we use in pretrain and downstream tasks), we never observe such a phenomenon – the $\|\mathbf{z}_t - \mathbf{z}_0\|_2^2$ always changes in a relatively small range. We speculate that in the first several updates during head probing, the features first adapt to the downstream task (i.e., classification), then gradually adapt to the downstream dataset distribution (e.g., STL, Flowers, etc.). So a randomly initialized head might influence more when the pretrain task is different from the downstream one. The detailed mechanism of adaptation of task and data distribution might be more complex than we expect, so we left this for the future work.

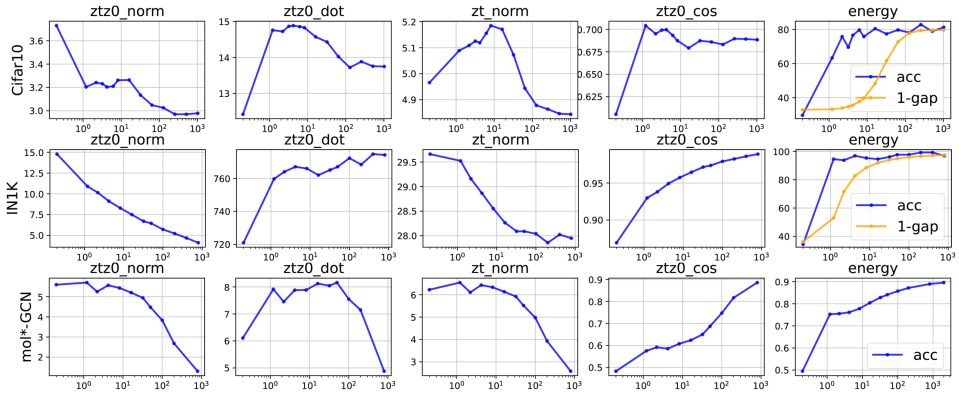


Figure 13: Adaptation of $\mathbf{z}_t$ when model is pretrained on different datasets.

D MORE EXPERIMENTS ON THE DOWNSTREAM PERFORMANCE

D.1 THE IMPORTANCE OF A SUITABLE τ

Besides the examples provided in Figure 7, which all demonstrate a quadratic shape when sweeping the value of τ, we provide some results on other different settings in Figure 16. These curves are generated when we searching for the hyper-parameters, like the learning rate, finetune epochs, etc. It is clear that in most cases, head probing the task head to convergence or finetuning the whole network with a randomly task head are not the best choice.

D.2 ABLATION STUDY OF LABEL SMOOTHING

		Sim-STL	Sup-STL		Sim-STL	Sup-STL
No LS-baseline	HP-train-acc	96.875	100	LS on FT	97.982	100
	1-$cos(\mathbf{z}_t, \mathbf{z}_0)$	0.0044	0.017		0.0109	0.0849
	$\|\mathbf{z}_t - \mathbf{z}_0\|_2^2$	1.792	6.716		3.041	13.881
	val-acc	93.914	97.581		93.208	97.039
LS on HP	HP-train-acc	96.875	100	LS on HP LS on FT	97.607	100
	1-$cos(\mathbf{z}_t, \mathbf{z}_0)$	0.014	0.0549		0.0049	0.0256
	$\|\mathbf{z}_t - \mathbf{z}_0\|_2^2$	1.841	9.278		1.731	7.391
	val-acc	94.304	97.969		94.015	97.694

Table 4: Ablation study of label smoothing in head probing. Extension of Table 2.

The main conclusion of this paper is that controlling initial energy on demand leads to better downstream performance. From our analysis, the key is to control the Brier score, i.e., $\|\mathbf{p}_t - \mathbf{e}_y\|_2^2$. Hence we propose two methods: earlier stopping or use label smoothing during head probing. In the earlier stopping method, smaller τ leads to a lower training accuracy at the beginning of finetuning, hence makes the Brier score larger. While in the label smoothing method, the $\mathbf{p}_t$ after head probing would converge to the smoothed one-hot distribution, which still has gaps to the true one-hot distribution. Hence even having the same training accuracy as the baseline, we see the ls-LP method in Table 4 make $\mathbf{z}_t$ adapt more and hence have better validation accuracy. However, one might think the performance enhancement comes from label smoothing, not more feature's adaptation. To answer this, we can compare the other two related cases, i.e., label smoothing only on finetuning and label smoothing on both (all use the same smoothing factor). In this table, we see using label smoothing on finetuning leads to more feature adaptation, but worse performance. That is because the features are adapted in an **opposite** direction during finetuning: from a peaky distribution ($\mathbf{p}_t$) to a flatter one (smoothed label in finetuning stage). If we apply label smoothing on both stages, the energy will still be small as the target distribution in the two stages are the same. Hence the performance and the feature's adaptations don't change a lot in this case.

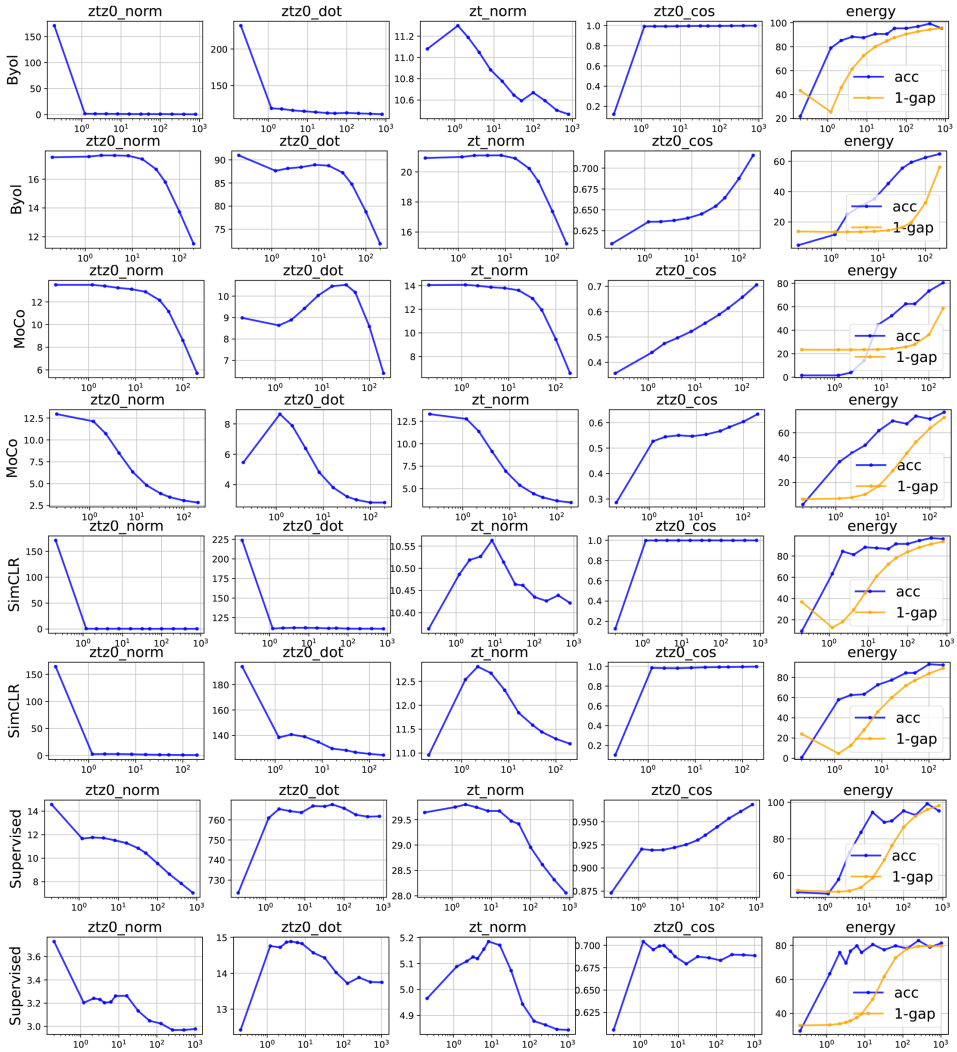


Figure 14: Adaptation of $\mathbf{z}_t$ when model is pretrained using different tasks.

D.3 Influence of the backbone and the task-head capacity

In Section 3.3, we mention a case where the task head is more complex. For example, the head can be a two-layers MLP rather than one linear layer, or we can only copy part of the pretrained model as our backbone. However, these methods and tricks are quite complex and might not necessarily enhance the performance, hence we put some results and discussions here. Remember that our analysis of how energy influences the features adaptation is still valid in these cases, as illustrated by the distance related metrics in the tables.

From Table 5, 6, 7, and 8, we can draw the following three conclusions. First, the two-layers MLP design will surely increase the training accuracy after head probing, and hence make the features adapt less (but this might not enhance the downstream performance). Second, we compare the downstream performance when some layers of the backbone are reinitialized as part of the task head. The title "+L4.3" means we take the last layer (i.e., the 3-rd layer) of the last block (i.e., the 4-th block) in a ResNet50 out, and treat it as part of the task head. The title "+L4.2" and "+L4.1" means we continue treating the 2-nd or the 1-st layer as the backbone. Under such a setting, the "+L4.1" case will inherit the least amount of information from the pretrained model, and at the same

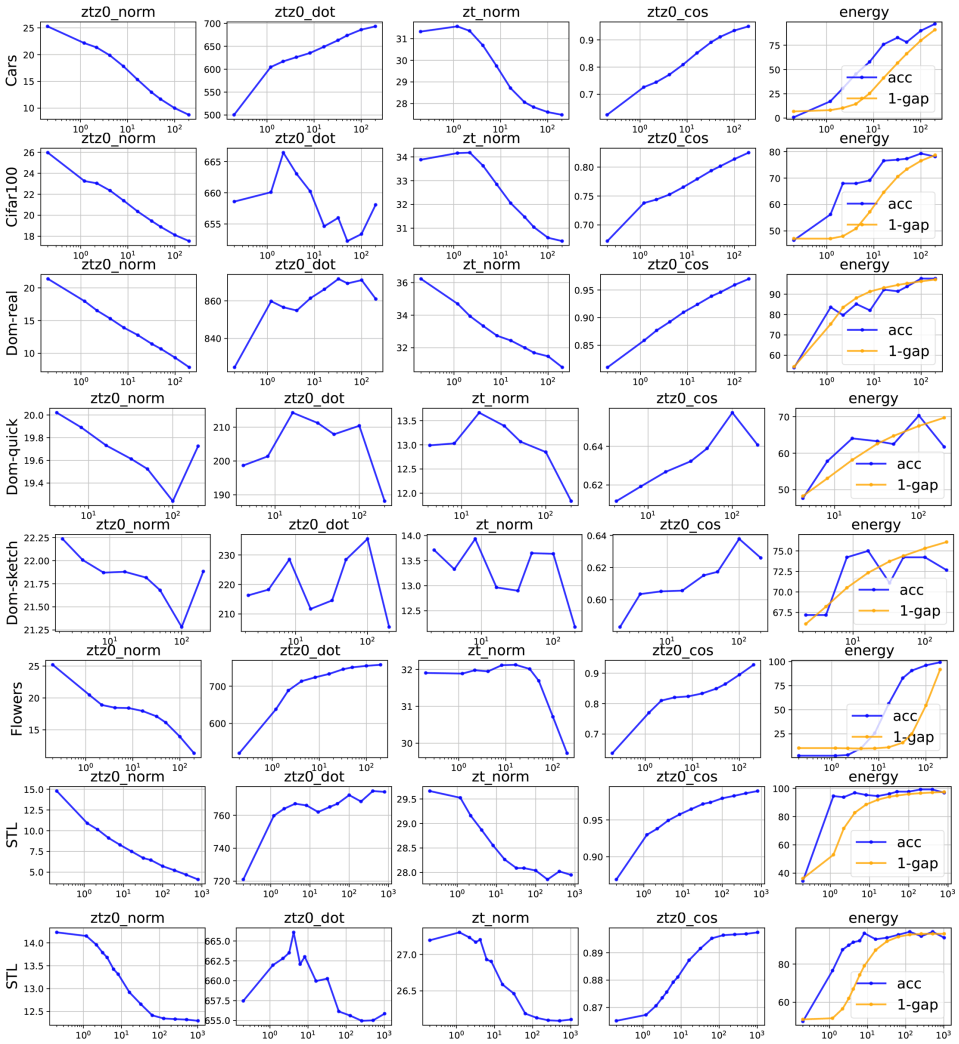


Figure 15: Adaptation of $\mathbf{z}_t$ when downstream tasks are different.

time, have the biggest task head. For the results in these tables, we train the head until convergence in head probing phase, then fine tune the whole network together. Hence we see the "HP-train-acc" value all increase to roughly 100%. However, the best setting differs when different downstream datasets are considered. When the pretrained features are good (i.e., trained using IN1K), we see the Domain-real dataset needs less adaptation and usually performs the best in the baseline case (i.e., copy all parameters from the pre-trained model). But Domain-quick dataset, which only contains some black-and-white lines, prefers the "+L4.1" setting. In other words, preserving the features captured by the earlier layers of the pretrained model is beneficial. When the pretrained features are bad, like the CIFAR-pretraining case in Table 7, all these datasets prefer throwing away the later layers.

In summary, although the correlation between features adaptation and downstream task performance under different settings is quite complicated, our analysis of energy can still explain some phenomena well.

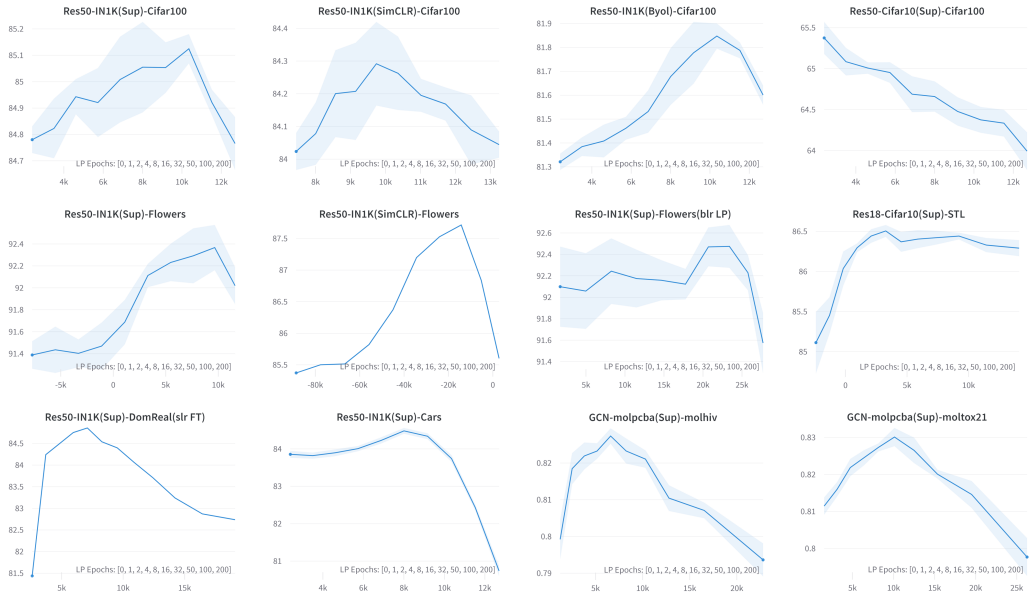


Figure 16: The relationships between τ and validation performance under different settings. In most of the cases, a suitable τ is important.

		Real	Sketch	Quick		Real	Sketch	Quick
Linear-head Baseline	HP-train-acc	98.994	63.369	48.555	2-MLP Head	100	93.447	63.75
	$1 - cos(\mathbf{z}_t, \mathbf{z}_0)$	0.0315	0.2275	0.3506		0.0149	0.2032	0.3375
	$\|\mathbf{z}_t - \mathbf{z}_0\|^2$	17.868	22.246	27.954		17.694	21.171	27.634
	val-acc	**54.839**	**37.789**	61.416		53.856	35.446	59.199
+L4.3	val-acc	53.629	35.966	**61.568**	HP-train-acc	99.492	99.072	93.975
+L4.2	val-acc	53.226	36.69	61.391	HP-train-acc	99.033	99.336	98.193
+L4.1	val-acc	53.856	35.446	59.199	HP-train-acc	99.863	93.447	63.75

Table 5: Results on a ResNet50 pertrained on Domain-Real. The pretrained validation accuracy is only 54.612 while the training accuracy is 100. Maybe because the DomainNet-Real dataset only contains less than 20k samples.

E ANALYZE LINEAR OVERPARAMETERIZATION PROBLEM

E.1 FORMALIZE THE CHANGE OF REPRESENTATIONS

Section 3 provides some intuitive explanations of how feature extractor $f(\mathbf{x}; \mathbf{B})$ changes given different $g(\mathbf{z}; \mathbf{v})$, which are well supported by the experimental results. To provide more insights, we analyze the change of features (i.e., $\mathbf{z}$) in a simplified overparameterization problem, i.e., the one provided in Section 2 and in Kumar et al. (2022). Under some mild assumptions and approximations, we provide an overview of how the norm and direction of $\mathbf{z}$ changes under different choice of $\mathbf{v}_0$. We show that the $\mathbf{v}_0$ satisfying $\mathbf{q}_0 = Y$ or $\mathbf{q}_0 = \frac{1}{2}Y$ are two critical points in feature adaptation.

We first rewrite Equation (1) in a non-matrix form:

$$\mathcal{L}_{\mathbf{B},\mathbf{v}} = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{2} \|\mathbf{v}^\top \mathbf{B}\mathbf{x}^{(n)} - y\|_2^2, \tag{11}$$

where $y \in \mathbb{R}$ as we are considering a regression problem (or a classification problem using MSE loss). We use the subscript to represent the time step, e.g., $\mathbf{q}_0$, $\mathbf{z}_0$, $\mathbf{v}_0$ and $\mathbf{B}_0$ are output, feature, head parameters and backbone parameters before finetuning. Similarly, $\mathbf{q}_t$, $\mathbf{z}_t$, $\mathbf{v}_t$ and $\mathbf{B}_t$ are the corresponding values after finetuned t steps. Note that $\mathbf{q}(X)$ and $\mathbf{z}(X)$ are functions of N input samples $X \in \mathbb{R}^{N \times d}$, but we omit it for simplicity. We use lowercase letters to represent the n-

		Real	Sketch	Quick		Real	Sketch	Quick
Linear-head Baseline	HP-train-acc	60.039	39.092	51.855	2-MLP Head	88.965	79.58	76.445
	$1 - cos(\mathbf{z}_t, \mathbf{z}_0)$	0.3492	0.4181	0.5034		0.3296	0.3672	0.4923
	$\|\mathbf{z}_t - \mathbf{z}_0\|^2$	9.963	9.72	9.287		9.104	9.209	9.874
	val-acc	58.846	45.023	61.593		58.77	43.171	60.307
+L4.3	val-acc	59.929	42.882	59.602	HP-train-acc	100	100	98.252
+L4.2	val-acc	60.837	44.734	61.542	HP-train-acc	100	100	100
+L4.1	val-acc	**61.164**	**45.775**	**62.626**	HP-train-acc	100	100	100

Table 6: Results on a ResNet34 pertrained on CIFAR-100 in a classification task.

		Real	Sketch	Quick		Real	Sketch	Quick
Linear-head Baseline	HP-train-acc	96.875	85.938	64.063	2-MLP Head	96.671	98.438	84.375
	$1 - cos(\mathbf{z}_t, \mathbf{z}_0)$	0.1384	0.2807	0.4432		0.1544	0.2661	0.4147
	$\|\mathbf{z}_t - \mathbf{z}_0\|^2$	16.569	18.005	21.869		16.843	17.672	21.389
	val-acc	**85.685**	**72.51**	**69.531**		85.031	69.651	66.683
+L4.3	val-acc	84.929	69.734	68.422	HP-train-acc	100	100	95.232
+L4.2	val-acc	83.342	68.547	68.7	HP-train-acc	100	100	98.087
+L4.1	val-acc	81.074	69.155	68.246	HP-train-acc	100	100	98.842

Table 7: Results on a ResNet50 pertrained on ImageNet-1K in a classification task.

		Real	Sketch	Quick		Real	Sketch	Quick
Linear-head Baseline	HP-train-acc	92.676	78.213	64.307	2-MLP Head	99.121	94.883	82.354
	$1 - cos(\mathbf{z}_t, \mathbf{z}_0)$	0.1269	0.1422	0.1934		0.1098	0.1104	0.1522
	$\|\mathbf{z}_t - \mathbf{z}_0\|^2$	5.247	4.752	4.361		4.942	4.206	3.868
	val-acc	78.075	61.545	59.703		78.453	63.773	63.609
+L4.3	val-acc	**78.528**	**65.683**	67.011	HP-train-acc	100	100	97.666
+L4.2	val-acc	78.427	65.336	67.087	HP-train-acc	100	100	100
+L4.1	val-acc	76.689	65.017	**68.07**	HP-train-acc	100	100	100

Table 8: Results on a ResNet50 pertrained on ImageNet-1K in a SimCLR task.

th element of the vector. Before discussing a specific element, we will clarify whether we are discussing the initialized case or the finetuned case. For example, specifying $\mathbf{q}_t$, we use $q_n = f(\mathbf{x}_n) = (f(X))_n$ to represent the prediction of $\mathbf{x}_n$ after finetuning.

Remember the goal of this paper is finding a suitable way to select task head, i.e., $\mathbf{v}_0$, given the pretrained feature extractor, i.e., $\mathbf{B}_0$. Depending on the discrepancy between the pretraining task and the downstream task, we might expect $\mathbf{z}_t$ change differently after finetuning. In other words, we care about the expected change of $\mathbf{z}_t$ compared to $\mathbf{z}_0$, i.e., $\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[d(\mathbf{z}_t, \mathbf{z}_0)\right]$, where $\mathcal{D}$ is the data distribution of the downstream task. Depending on what distance measurement (i.e., $d(\cdot,\cdot)$) we choose, there are three different metrics:

- Euclidean: $\bar{d}_{euc} \triangleq \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\|\mathbf{z}_t - \mathbf{z}_0\|_2^2\right]$,
- Dot product: $\bar{d}_{dot} \triangleq \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\mathbf{z}_t^\top \mathbf{z}_0\right]$,
- Cosine: $\bar{d}_{cos} \triangleq \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\frac{\mathbf{z}_t^\top \mathbf{z}_0}{\|\mathbf{z}_t\|_2 \cdot \|\mathbf{z}_0\|_2}\right]$.

We start from the Euclidean case:

$$
\begin{aligned}
\bar{d}_{euc} &= \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[(\mathbf{z}_t - \mathbf{z}_0)^\top(\mathbf{z}_t - \mathbf{z}_0)\right] \\
&= \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\mathbf{z}_t^\top\mathbf{z}_t - \mathbf{z}_t^\top\mathbf{z}_0 - \mathbf{z}_0^\top\mathbf{z}_t + \mathbf{z}_0^\top\mathbf{z}_0\right] \\
&= \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\mathbf{z}_t^\top\mathbf{z}_t - 2\mathbf{z}_0^\top\mathbf{z}_t + \mathbf{z}_0^\top\mathbf{z}_0\right] \\
&= \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\mathbf{x}^\top\mathbf{B}_t^\top\mathbf{B}_t\mathbf{x} - 2\mathbf{x}^\top\mathbf{B}_0^\top\mathbf{B}_t\mathbf{x} + \mathbf{x}^\top\mathbf{B}_0^\top\mathbf{B}_0\mathbf{x}\right] \\
&= \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[tr(\mathbf{x}^\top\mathbf{B}_t^\top\mathbf{B}_t\mathbf{x}) - 2tr(\mathbf{x}^\top\mathbf{B}_0^\top\mathbf{B}_t\mathbf{x}) + tr(\mathbf{x}^\top\mathbf{B}_0^\top\mathbf{B}_0\mathbf{x})\right] \\
&= \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[tr(\mathbf{B}_t^\top\mathbf{B}_t\mathbf{x}\mathbf{x}^\top) - 2tr(\mathbf{B}_0^\top\mathbf{B}_t\mathbf{x}\mathbf{x}^\top) + tr(\mathbf{B}_0^\top\mathbf{B}_0\mathbf{x}\mathbf{x}^\top)\right] \\
&= tr(\mathbf{B}_t^\top\mathbf{B}_t\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[\mathbf{x}\mathbf{x}^\top]) - 2tr(\mathbf{B}_0^\top\mathbf{B}_t\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[\mathbf{x}\mathbf{x}^\top]) + tr(\mathbf{B}_0^\top\mathbf{B}_0\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[\mathbf{x}\mathbf{x}^\top]) \\
&= M\cdot\left(tr(\mathbf{B}_t^\top\mathbf{B}_t) - 2tr(\mathbf{B}_0^\top\mathbf{B}_t) + tr(\mathbf{B}_0^\top\mathbf{B}_0)\right),
\end{aligned} \tag{12}
$$

where $M \triangleq \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[\mathbf{x}\mathbf{x}^\top]$.

Similarly, we can get the expressions of $\bar{d}_{dot}$ and $\|\mathbf{z}_t\|_2^2$, which are building blocks of $\bar{d}_{cos}$:

$$
\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\mathbf{z}_t^\top\mathbf{z}_0\right] = M\cdot tr(\mathbf{B}_0^\top\mathbf{B}_t) \tag{13}
$$

$$
\mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\|\mathbf{z}_t\|_2^2\right] = M\cdot tr(\mathbf{B}_t^\top\mathbf{B}_t) \tag{14}
$$

E.2 Critical Points

Our goal is to find a good initialized task head, i.e., $\mathbf{v}_0$, which can lead to better downstream performance. Instead of directly linking $\mathbf{v}_0$ to the expected risk, which is a common practice for generalization analysis [6], we consider this problem from another indirect way. Specifically, we assume the SGD algorithm with appropriate regularization in FT stage can find a good optimum for the learning task. What we care more about is whether the features learned from the pretraining stage adapts well to the downstream task. We believe that if the features are properly adapted to the new environment, the model has more potential to generalize better. Hence in this part, considering the aforementioned three distance metrics as the target functions, we formalize how $\mathbf{v}_0$ influence them.

From Equation (12), (13) and (14), we find that the behavior of $tr(\mathbf{B}_0^\top\mathbf{B}_t)$ and $tr(\mathbf{B}_t^\top\mathbf{B}_t)$ are the keys. In these two terms, $\mathbf{B}_0$ is given and cannot change, while $\mathbf{B}_t$ is determined by the choice of $\mathbf{v}_0$. One way to link these two quantities is using Lemma A.4 in Kumar et al. (2022) or theorem 2.2 in Du et al. (2018):

$$
\mathbf{v}_0\mathbf{v}_0^\top - \mathbf{B}_0\mathbf{B}_0^\top = \mathbf{v}_t\mathbf{v}_t^\top - \mathbf{B}_t\mathbf{B}_t^\top, \quad \forall t, \tag{15}
$$

however, the expression of $\mathbf{v}_t$ is still hard to obtain (but collecting and visualizing $\mathbf{v}_t$ is much more cheaper than $\mathbf{B}_t$). We left this direction for our future work. In this paper, we analyze the problem in the NTK regime (this is the main assumption of our analysis, which can cause discrepancies between the theory and experiments).

Closed-form of parameters under NTK approximation:

To get more insights, we approximate the behavior of this model in the NTK regime, in which the converged parameters can be analytically calculated. Specifically, be applying Equation (8) in Lee et al. (2019) and assuming $t \to \infty$, we can have:

$$
\theta_t = \theta_0 - (\nabla_\theta\mathbf{q}_0)^\top\mathcal{K}_0^{-1}(\mathbf{q}_0 - Y), \tag{16}
$$

where $\mathbf{q}_0 \in \mathbb{R}^{N\times 1}$ is the model's prediction on N training samples, i.e., X, and $\theta_t, \theta_0 \in \mathbb{R}^{(d+1)*h\times 1}$ are the stacked parameters. Without loss of generality, the first $d*h$ parameters in θ come from $\mathbf{B}$ and the last h parameters come from $\mathbf{v}$.

[6] But it is hard to get tight and informative bounds in deep learning, especially in such a practical scenario.

The $\mathcal{K}_0 = \nabla_\theta \mathbf{q}_0 \cdot (\nabla_\theta \mathbf{q}_0)^\top \in \mathbb{R}^{N\times N}$ here is the empirical NTK on X. Specifically, by stacking the paramters, we can calculate each elements in this kernel as:

$$\begin{aligned}\kappa(\mathbf{x}, \mathbf{x}') &= \mathbf{x}^\top \mathbf{B}^\top \mathbf{B}\mathbf{x}' + \sum_{i=1}^{h}\sum_{j=1}^{d}(v_i x_j)(v_i x_j') \\ &= \mathbf{x}^\top \mathbf{B}^\top \mathbf{B}\mathbf{x}' + \sum_{i=1}^{h} v_i^2 \sum_{j=1}^{d} x_j x_j' \\ &= \mathbf{x}^\top (\mathbf{B}^\top \mathbf{B} + \|\mathbf{v}\|_2^2 \cdot I_{d\times d})\mathbf{x}', \end{aligned} \tag{17}$$

where v_i and x_i is the i-th element in $\mathbf{v}$ and $\mathbf{x}$ respectively. Then, the matrix form of emperical NTK is $\mathcal{K}_0 = X(\mathbf{B}_0^\top \mathbf{B}_0 + \|\mathbf{v}_0\|_2^2 \cdot I_{d\times d})X^\top$.

With the help of Equation (16), we can get the closed-form expression of $\mathbf{b}_t \triangleq \text{vec}(\mathbf{B}_t)$ (i.e., the vectorization of matrix $\mathbf{B}_t$):

$$\mathbf{b}_t = \mathbf{b}_0 - (\nabla_{\mathbf{b}} \mathbf{q}_0)^\top \mathcal{K}_0^{-1}(\mathbf{q}_0 - Y), \tag{18}$$

In Equation (18), we know $\mathbf{q}_0 = X\mathbf{B}_0^\top \mathbf{v}_0$. The term $\nabla_{\mathbf{b}}\mathbf{q}_0$ also depends on $\mathbf{v}_0$. As controlling $\mathbf{q}_0$, i.e., the model's prediction on training samples, is easier in practice (we can directly observe the training loss or training accuracy), we will consider $\mathbf{q}_0$ as the optimizing variable in the rest of the paper. The following lemma can link $\nabla_{\mathbf{b}}\mathbf{q}_0$ to $\mathbf{q}_0$:

Lemma 1. *For $\mathbf{b}_0 \in \mathbb{R}^{h*d\times 1}$ and $\nabla_{\mathbf{b}}\mathbf{q}_0 \in \mathbb{R}^{N\times h*d}$, we have $\nabla_{\mathbf{b}}\mathbf{q}_0 \cdot \mathbf{b}_0 = \mathbf{q}_0$.*

Proof. We check that equation elementwise. For the n-th element in the RHS, we have:

$$\begin{aligned} q_n &= \mathbf{x}_n^\top \mathbf{B}_0^\top \mathbf{v}_0 \\ &= \sum_{i=1}^{d}\sum_{j=1}^{h} x_i v_j (\mathbf{B}_0^\top)_{i,j} \\ &= \sum_{i=1}^{d}\sum_{j=1}^{h} x_i v_j (\mathbf{B}_0)_{j,i}, \end{aligned} \tag{19}$$

where q_n is the n-th element of $\mathbf{q}_0$, x_i is the i-th element of $\mathbf{x}_n$, and v_j is the j-th element of $\mathbf{v}_0$.

For the LHS, the n-th value is $\mathbf{e}_n^\top \nabla_{\mathbf{b}}\mathbf{q}_0 \cdot \mathbf{b}_0$, where $\mathbf{e}_n^\top$ is a one-hot row vector selecting the n-th row of $\nabla_{\mathbf{b}}\mathbf{q}_0$. From the definition, we know $\mathbf{e}_n^\top \nabla_{\mathbf{b}}\mathbf{q}_0 = \text{vec}(\mathbf{v}_0\mathbf{x}_n^\top)^\top = [x_1\mathbf{v}_0^\top, x_2\mathbf{v}_0^\top, ..., x_d\mathbf{v}_0^\top]$, which is a long row vector. Then, the n-th element in the LHS should be:

$$\begin{aligned} \mathbf{e}_n^\top \nabla_{\mathbf{b}}\mathbf{q}_0 \cdot \mathbf{b}_0 &= \text{vec}(\mathbf{v}_0\mathbf{x}_n^\top)^\top \cdot \mathbf{b}_0 \\ &= \sum_{l}^{h*d} (\text{vec}(\mathbf{v}_0\mathbf{x}_n^\top))_l (\mathbf{b}_0)_l \\ &= \sum_{i=1}^{d}\sum_{j=1}^{h} x_i v_j (\mathbf{B}_0)_{j,i} \\ &= q_n, \end{aligned} \tag{20}$$

where the last equation holds the rule of stacking elements in a matrix. As each elements of the two sides are equal, the lemma holds. □

Critical points of $tr(\mathbf{B}_0^\top \mathbf{B}_t)$ and $tr(\mathbf{B}_t^\top \mathbf{B}_t)$:

Lemma 2. $\mathbf{q}_0^* = \frac{1}{2}Y$ *is a maximum of* $tr(\mathbf{B}_0^\top \mathbf{B}_t)$.

Proof. By definition, we have:

$$\begin{aligned} tr(\mathbf{B}_0^\top \mathbf{B}_t) &= \mathbf{b}_0^\top \mathbf{b}_t \\ &= \mathbf{b}_0^\top \left(\mathbf{b}_0 - (\nabla_{\mathbf{b}}\mathbf{q}_0)^\top \mathcal{K}_0^{-1}(\mathbf{q}_0 - Y)\right) \\ &= \mathbf{b}_0^\top \mathbf{b}_0 - (\nabla_{\mathbf{b}}\mathbf{q}_0 \cdot \mathbf{b}_0)^\top \mathcal{K}_0^{-1}(\mathbf{q}_0 - Y) \\ &= \text{const.} - \mathbf{q}_0^\top C_1(\mathbf{q}_0 - Y), \end{aligned} \tag{21}$$

where we call $C_1 = \mathcal{K}_0^{-1}$ for convenience. By taking first derivative to $\mathbf{q}_0$ and let is equal zero, we know $\mathbf{q}_0^* = \frac{1}{2}Y$ is a critical point. As the second derivative is $-2C_1$ and C_1 is positive definite, this critical point is a maximum. □

Lemma 3. $\mathbf{q}_0^* = \alpha Y, \alpha \in (0, 0.5)$ *is a critical point (usually maximum) of* $tr(\mathbf{B}_t^\top \mathbf{B}_t)$. *Here* $\alpha = \left(I_{N\times N} + (C_2 - 2C_1)^{-1}C_1\right)Y$, *where* $C_2 = \mathcal{K}_0^{-1}\tilde{\mathcal{K}}_0\mathcal{K}_0^{-1}$ *and* $\tilde{\mathcal{K}}_0 = X\mathbf{B}_0^\top \mathbf{B}_0 X^T = (\nabla_{\mathbf{v}}\mathbf{q}_0)(\nabla_{\mathbf{v}}\mathbf{q}_0)^\top$ *is the NTK when fixing the parameters of the backbone.*

Proof. Similar to Lemma 2, we can write $tr(\mathbf{B}_t^\top \mathbf{B}_t)$ as $\mathbf{b}_t^\top \mathbf{b}_t$ and substitute Equation (18):

$$\begin{aligned} tr(\mathbf{B}_t^\top \mathbf{B}_t) &= \mathbf{b}_t^\top \mathbf{b}_t \\ &= \left(\mathbf{b}_0 - (\nabla_{\mathbf{b}}\mathbf{q}_0)^\top \mathcal{K}_0^{-1}(\mathbf{q}_0 - Y)\right)^\top \left(\mathbf{b}_0 - (\nabla_{\mathbf{b}}\mathbf{q}_0)^\top \mathcal{K}_0^{-1}(\mathbf{q}_0 - Y)\right) \\ &= \mathbf{b}_0^\top \mathbf{b}_0 - 2(\nabla_{\mathbf{b}}\mathbf{q}_0 \cdot \mathbf{b}_0)^\top \mathcal{K}_0^{-1}(\mathbf{q}_0 - Y) + (\mathbf{q}_0 - Y)^\top \mathcal{K}_0^{-1}(\nabla_{\mathbf{b}}\mathbf{q}_0)(\nabla_{\mathbf{b}}\mathbf{q}_0)^\top \mathcal{K}_0^{-1}(\mathbf{q}_0 - Y) \\ &= \mathbf{b}_0^\top \mathbf{b}_0 - 2\mathbf{q}_0^\top \mathcal{K}_0^{-1}(\mathbf{q}_0 - Y) + (\mathbf{q}_0 - Y)^\top \mathcal{K}_0^{-1}\tilde{\mathcal{K}}_0\mathcal{K}_0^{-1}(\mathbf{q}_0 - Y) \\ &= \mathbf{b}_0^\top \mathbf{b}_0 - 2\mathbf{q}_0^\top C_1(\mathbf{q}_0 - Y) + (\mathbf{q}_0 - Y)^\top C_2(\mathbf{q}_0 - Y) \\ &= (\mathbf{b}_0^\top \mathbf{b}_0 + Y^\top C_2 Y) + \mathbf{q}_0^\top (2C_1 - 2C_2)Y + \mathbf{q}_0^\top (C_2 - 2C_1)\mathbf{q}_0 \\ &= \text{const.} + \mathbf{q}_0^\top (2C_1 - 2C_2)Y + \mathbf{q}_0^\top (C_2 - 2C_1)\mathbf{q}_0 \end{aligned} \tag{22}$$

By taking first derivative and letting it equal zero, assuming $(C_2 - 2C_1)$ is invertible, we know $\mathbf{q}_0^* = \left(I_{N\times N} + (C_2 - 2C_1)^{-1}C_1\right)Y$ is a critical point. The second derivative is $2C_2 - 4C_1$, which is usually positive definite in our settings (explain later). □

To get more insights, we can look deeper into the critical point mentioned in Lemma 3. Following the definition of C_1 and C_2, we can have:

$$\begin{aligned} C_2 - 2C_1 &= \mathcal{K}_0^{-1}\tilde{\mathcal{K}}_0\mathcal{K}_0^{-1} - 2\mathcal{K}_0^{-1} \\ &= (\mathcal{K}_0^{-1}\tilde{\mathcal{K}}_0 - 2I_{N\times N})\mathcal{K}_0^{-1}, \end{aligned} \tag{23}$$

where $\mathcal{K}_0^{-1}\tilde{\mathcal{K}}_0$ is the key of understanding this term.

The exact form of this expression is hard to obtain, but as all the NTK or covariance matrices mentioned here are sysmetric, we can compare the trace of them to get some insights. By definition, $tr(\tilde{\mathcal{K}}_0) = tr(X\mathbf{B}_0^\top \mathbf{B}_0 X^\top) = tr(\mathbf{B}_0^\top \mathbf{B}_0 X^\top X)$. If each dimensions of the data samples are independent, then $X^\top X \approx I_{d\times d}$, and $tr(\tilde{\mathcal{K}}_0) \approx tr(\mathbf{b}_0^\top \mathbf{b}_0) = \|\mathbf{b}_0\|_2^2$. Then similarly, $tr(\mathcal{K}_0) \approx \|\mathbf{b}_0\|_2^2 + \|\mathbf{v}_0\|_2^2$. Thus the behavior of term $\mathcal{K}_0^{-1}\tilde{\mathcal{K}}_0$ can be described by $\frac{\|\mathbf{b}_0\|_2^2}{\|\mathbf{b}_0\|_2^2 + \|\mathbf{v}_0\|_2^2} \cdot I_{N\times N}$:

- if $\|\mathbf{b}_0\|_2^2 \gg \|\mathbf{v}_0\|_2^2$, which might happen as the backbone contains more parameters than the head, $\mathcal{K}_0^{-1}\tilde{\mathcal{K}}_0 \approx I_{N\times N}$,
- if $\|\mathbf{b}_0\|_2^2 \ll \|\mathbf{v}_0\|_2^2$, which might happen if we split the network in the earlier layer, $\mathcal{K}_0^{-1}\tilde{\mathcal{K}}_0 \approx 0 \cdot I_{N\times N}$.

But in either case, the negative of the second derivative of $tr(\mathbf{B}_t^\top \mathbf{B}_t)$, i.e., Equation (23), is likely to be positive definite, which means the critical point in Lemma 3 is usually a maximum. Another interesting fact of Lemma 3 is $\mathbf{q}_0^*$ under the aforementioned two extreme conditions. After some calculation, we can verify that $\mathbf{q}_0^* \to 0 \cdot Y$ if $\|\mathbf{b}_0\|_2^2 \gg \|\mathbf{v}_0\|_2^2$ and $\mathbf{q}_0^* \to \frac{1}{2}Y$ if $\|\mathbf{b}_0\|_2^2 \ll \|\mathbf{v}_0\|_2^2$.

Critical points of $\bar{d}_{euc}$, $\bar{d}_{dot}$ and $\bar{d}_{cos}$:

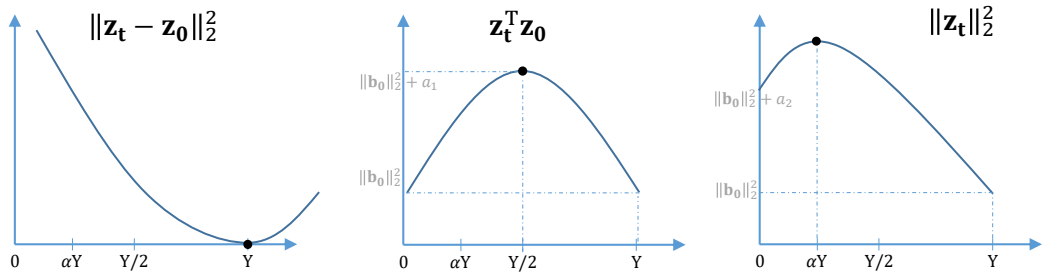


Figure 17: Illustrations of different metrics if they are 1-D quadratic functions. Here $a_1 = \alpha Y^\top C_1 Y$ and $a_2 = Y^\top C_2 Y$. The x-axis represents the choice of $\mathbf{q}_0$, where αY is an example of the critical point of $\|\mathbf{z}_t\|_2^2$. The black dots are the critical points.

Recall the definitions of different distance metrics we care about. First, $\bar{d}_{dot}$ is proportional to $\mathbf{z}_t^\top \mathbf{z}_0$, hence its shape would like the second panel in Figure 17. For $\bar{d}_{euc}$, we know it is proportional to $tr(\mathbf{B}_t^\top \mathbf{B}_t) - 2tr(\mathbf{B}_0^\top \mathbf{B}_t) + tr(\mathbf{B}_0^\top \mathbf{B}_0)$. Substituting results in Equation (21) and Equation (22), we have:

$$\begin{aligned}\bar{d}_{euc} &\propto tr(\mathbf{B}_t^\top \mathbf{B}_t) - 2tr(\mathbf{B}_0^\top \mathbf{B}_t) + \text{const.} \\ &= \mathbf{q}_0^\top (2C_1 - 2C_2)Y + \mathbf{q}_0^\top (C_2 - 2C_1)\mathbf{q}_0 + 2\mathbf{q}_0^\top C_1(\mathbf{q}_0 - Y) + \text{const.} \\ &= -2\mathbf{q}_0^\top C_2 Y + \mathbf{q}_0^\top C_2 \mathbf{q}_0 + \text{const.} \end{aligned} \tag{24}$$

Obviously, $\mathbf{q}_0^* = Y$ is the minimum of $\bar{d}_{euc}$, as depicted in the first panel in Figure 17. The shape of $\bar{d}_{cos}$ is hard to obtain. However in the last panel of Figure 17, we demonstrate $\|\mathbf{z}_t\|_2^2$, its denominator, to assist our further analysis.

E.3 Learning dynamics of the representation

$\mathbf{q}_0 =$	d_{euc}	d_{dot}	$\|\mathbf{z}_t\|_2$	d_{cos}
$Y \to \frac{1}{2}Y$	$\uparrow$	$\uparrow$	$\uparrow$	? ($\downarrow$)
$\frac{1}{2}Y \to \alpha Y$	$\uparrow$	$\downarrow$	$\uparrow$	$\downarrow\downarrow$
$\alpha Y \to 0$	$\uparrow$	$\downarrow$	$\downarrow$	? ($\downarrow$)

Table 9: Question marks in the last column means we cannot accurately predict its change. But it usually decreases in experiments.

In this subsection, we will put everything together to provide an overview of how $\mathbf{z}_t$ changes compared with $\mathbf{z}_0$. Remember when $\mathbf{q}_0 = Y$, the gradient of $\mathbf{B}_0$ is zero, hence $\mathbf{z}_t = \mathbf{z}_0$ (Kumar et al., 2022). When $\mathbf{q}_0$ moves linearly from Y to 0, we have the following three phases (see Table 9):

- $Y \to \frac{1}{2}Y$: $\mathbf{z}_t$ lengthen its norm with a slightly change in the direction, hence $\bar{d}_{dot}$ also increase;
- $\frac{1}{2}Y \to \alpha Y$: the norm of $\mathbf{z}_t$ keeps increasing, but its direction drastically changes in this phase, which makes $\bar{d}_{dot}$ decrease;
- $\alpha Y \to 0$: the norm of $\mathbf{z}_t$ begins to decrease and the angle between $\mathbf{z}_t$ and $\mathbf{z}_0$ keeps increasing, which makes $\mathbf{z}_t$ changes a lot.