Fine-tuning can cripple your foundation model; preserving features may be the solution

Anonymous authors
Paper under double-blind review

Abstract

Pre-trained foundation models, due to their enormous capacity and exposure to vast amounts of data during pre-training, are known to have learned plenty of real-world concepts. An important step in making these pre-trained models extremely effective on downstream tasks is to fine-tune them on related datasets. While various fine-tuning methods have been devised and have been shown to be highly effective, we observe that a fine-tuned model's ability to recognize concepts on tasks *different* from the downstream one is reduced significantly compared to its pre-trained counterpart. This is an undesirable effect of fine-tuning as a substantial amount of resources was used to learn these pre-trained concepts in the first place. We call this phenomenon "concept forgetting" and via experiments show that most end-to-end fine-tuning approaches suffer heavily from this side effect. To this end, we propose a simple fix to this problem by designing a new fine-tuning method called LDIFS (short for ℓ_2 distance in feature space) that, while learning new concepts related to the downstream task, allows a model to preserve its pre-trained knowledge as well. Through extensive experiments on 10 fine-tuning tasks we show that LDIFS significantly reduces concept forgetting. Additionally, we show that LDIFS is highly effective in performing continual fine-tuning on a sequence of tasks as well, in comparison with both fine-tuning as well as continual learning baselines.

1 Introduction

Foundation models like CLIP Radford et al. (2021), ALIGN Jia et al. (2021) and CoCa Yu et al. (2022) are trained using self-supervised methods on hundreds of millions or even billions of samples scraped from the internet. This massive, compute intensive pre-training makes such models a knowledge store on a vast number of real-world concepts, enabling them to easily transfer to a wide variety of downstream tasks and applications. Indeed, this ability to recognize real-world concepts and thereby transfer to downstream tasks is the primary advantage of such models and is the very reason behind their name Bommasani et al. (2021).

While a foundation model can achieve impressive performance on a downstream task often without even requiring a single training sample from the task itself Radford et al. (2021), in order to maximise performance, it conventionally requires some form of fine-tuning on the task at hand. There are multiple types of fine-tuning methods like linear probing Radford et al. (2021), prompt-tuning Zhou et al. (2022b;a), adapters Gao et al. (2021); Zhang et al. (2021), weight-space interpolation Wortsman et al. (2022b); Ilharco et al. (2022a) and full end-to-end fine-tuning Radford et al. (2021); Kumar et al. (2022); Goyal et al. (2022); Xuhong et al. (2018). Among these types, end-to-end fine-tuning is well-known to produce the best downstream performance.

It is worth noting that the pre-training dataset of a foundation model, owing to its massive scale, contains information about several thousands of real-world concepts. Hence, it is highly likely that the downstream dataset for fine-tuning the model will only contain a significantly smaller number of concepts compared to its pre-training set. A natural question that arises then is: *How does end-to-end fine-tuning of a foundation model affect the vast knowledge it acquired through its pre-training?* This is precisely what we aim to answer.

Through a thorough study of popular end-to-end fine-tuning methods, we observe that for most of them, the fine-tuned model has significantly lost its ability to recognize real-world concepts outside the downstream task, a phenomenon which we call *concept forgetting*. This is highly undesirable as there are important

use-cases where we would want the foundation model's vast knowledge on concepts to be preserved even after fine-tuning.

One important use-case, for instance, is the requirement for *continual fine-tuning* to update a pre-trained model with previously unknown knowledge. Several organizations have spent millions of dollars pre-training large foundation models from scratch Knight (2023). Even after such powerful pre-training, these models can exhibit gaps in their knowledge. Specific examples include ChatGPT OpenAI (2020) having a knowledge-cutoff date of September 2021 or CLIP Radford et al. (2021) failing to perform in niche downstream areas like satellite or medical imagery. Furthermore, fields like medicine are constantly updating with new information like new diseases, new diagnoses and treatments etc. Therefore, having a fixed knowledge-base in medical foundation models like MedCLIP Wang et al. (2022) or MedPaLM Singhal et al. (2022) is not an option if we are to use them in practical downstream use cases. With their expensive pre-training process, re-training a foundation model from scratch by combining new data with prior training data, is too expensive to be feasible. This requires the pre-trained foundation model to be fine-tuned on new data while preserving its prior knowledge, thereby necessitating investigation into fine-tuning methods which prevent concept forgetting.

Although conventional end-to-end fine-tuning methods generally suffer from concept forgetting, we find that there can be a relatively simple fix to this problem. In particular, if a fine-tuning method ensures that the fine-tuned model is close in some sense to the original foundation model, it can significantly reduce concept forgetting. One way to define the vicinity of the original model is in terms of distance in the parameter space. This is seen in the case of the L2SP regularizer Xuhong et al. (2018). However, we find that it is much more effective to define vicinity in terms of distance in the model's feature space which captures its input-output behaviour. This leads us to propose a new regularizer, **LDIFS (ℓ_2 distance in Feature Space)**, which minimizes the distance between the features of the original model and the model being fine-tuned during fine-tuning. Furthermore, we observe that simply preserving the last-layer features is not effective in reducing concept forgetting. Motivated from observations in Zhang et al. (2018), we therefore preserve features extracted from different internal representations. We find that this relatively simple method of preserving the original model's features while fine-tuning on a downstream task can significantly alleviate the problem of concept forgetting in the fine-tuned model, without affecting its performance on the downstream task. Empirically we show this through an extensive evaluation of LDIFS on 10 different downstream tasks.

Finally, since LDIFS preserves pre-trained knowledge during fine-tuning, as a natural extension, we study a continual setup with a sequence of fine-tuning tasks. In particular, our continual setup uses 3 sequences, each of 3 tasks. Again, in every evaluation setting, we find LDIFS to outperform both fine-tuning as well as classic continual learning methods in minimizing concept forgetting, without compromising performance on the fine-tuned tasks themselves. Thus, to summarize, our contributions in this work are as follows:

1. **Investigate concept forgetting.** To the best of our knowledge, we are the first to perform a thorough analysis and evaluation of concept forgetting for fine-tuning multi-modal foundation models. We propose a simple way to quantify concept forgetting and benchmark 6 existing end-to-end fine-tuning methods on 10 different downstream tasks to find that concept forgetting, as a phenomenon, exists in all of them.

2. **Analyze different end-to-end fine-tuning methods.** We find a consistent ordering in concept forgetting for different fine-tuning methods. Particularly, we find the L2SP Xuhong et al. (2018) regularizer to outperform other fine-tuning baselines. We analyze why this is the case.

3. **Propose a new regularizer for end-to-end fine-tuning.** Analyzing L2SP helps us propose a simple new regularizer (LDIFS) which minimizes feature space distance between pre-trained and fine-tuned models during fine-tuning. Minimizing feature space distance fine-tunes the model on the downstream task while preserving much of the input-output behaviour of the pre-trained model. This helps LDIFS outperform other existing fine-tuning methods in minimizing concept forgetting during fine-tuning.

4. **A nudge towards continual fine-tuning.** Finally, as a natural extension of the evaluation setting we evaluate on a continual setup of 3 different sequences of 3 fine-tuning tasks each and find LDIFS to be superior to both fine-tuning methods as well as 5 classic continual learning baselines in preserving and accumulating knowledge in this setup.

2 A brief note on fine-tuning

Here we provide a description of CLIP Radford et al. (2021) as our foundation model of choice and briefly discuss existing state-of-the-art methods used for fine-tuning CLIP.

CLIP, a brief overview Broadly speaking, a CLIP model has two components: **i)** the vision or image encoder[1] $f_{\theta_v} : \mathbb{R}^{C,H,W} \to \mathbb{R}^D$, and **ii)** the text encoder $f_{\theta_t} : \mathbb{R}^L \to \mathbb{R}^D$. CLIP is pre-trained on 400 million pairs of images and corresponding text descriptions scraped from the internet. For pre-training, it uses a contrastive loss to maximize cosine similarity between the correct (image, text) pairs and minimize the same for the incorrect ones. Due to its large-scale self-supervised pre-training, CLIP exhibits impressive performance on several downstream tasks, often without requiring a single training sample from the task itself. However, in order to maximise performance on a specific downstream task, the pre-trained CLIP model is conventionally fine-tuned further on the task itself. Below we provide a brief description of such popular fine-tuning methods relevant to our study.

Zero-shot (ZS) In image classification, given an input image $\mathbf{x}$ and a set of K class names $\{\mathbf{c}_i\}_{i=1}^K$ as natural language text, the D-dimensional encoding[2] for each class name $\psi(\mathbf{c}_i) = f_{\theta_t}(\mathbf{c}_i)$ and the image $\phi(\mathbf{x}) = f_{\theta_v}(\mathbf{x})$ are first obtained. The text encodings $\psi(\mathbf{c}_i)$ are then used as parameters of a K-class linear classifier, and the classification inference on $\mathbf{x}$ is performed as $\arg\max_i \psi(\mathbf{c}_i)^{\mathrm{T}}\phi(\mathbf{x})$. This is known as the zero-shot (ZS) prediction and CLIP's best model has been shown to have competitive ZS accuracies with a fully supervised ResNet-101 on ImageNet Radford et al. (2021).

Linear Probe (LP) In this case, an additional linear layer $\mathbf{w} \in \mathbb{R}^{D\times K}$ is appended on top of the image encoder f_{θ_v} and the weights of this linear layer are trained by solving a standard logistic regression problem (e.g., scikit-learn's `LogisticRegression` module Pedregosa et al. (2011)). The linear layer $\mathbf{w}$ is normally initialized using the text representations $\{\psi(\mathbf{c}_i)\}_{i=1}^K$, known as ZS initialization. It is trivial to note that in the absence of any training, LP boils down to ZS.

End-to-end fine-tuning While a pre-trained CLIP encoder can obtain impressive ZS and LP accuracies on several tasks, in order to maximize performance on a specific downstream task, the general rule of thumb is to initialize a model from the weights of the pre-trained encoder and then fine-tune the model end-to-end on the downstream task. Here we list some of the most popular end-to-end fine-tuning methods which we study in this work. We provide a more detailed discussion on the different types of fine-tuning methods in §6.

1. **ZS-init-CE** Radford et al. (2021): This is the classic end-to-end fine-tuning method where, similar to the LP, a ZS initialized linear head $\mathbf{w} : \mathbb{R}^D \to \mathbb{R}^K$ is appended to the image encoder f_{θ_v}. However, differently from the LP, parameters of the entire model $\theta = \{\theta_v, \mathbf{w}\}$ (including the image encoder parameters) are fine-tuned using a cross-entropy loss $\mathcal{L}_{\text{CE}}$.

2. **LP-init-CE (LP-FT)** Kumar et al. (2022): This is similar to ZS-init-CE but instead of initializing the appended linear head via ZS, it is initialized by performing linear probing on the downstream task first. Once the linear head is initialized, the entire model is end-to-end fine-tuned using $\mathcal{L}_{\text{CE}}$.

3. **ZS-init-L2SP** Xuhong et al. (2018): In addition to the cross-entropy loss $\mathcal{L}_{\text{CE}}$, this method uses an additional regularizer to minimize the ℓ_2 distance between the pre-trained and fine-tuned *image encoder weights*, thereby trying to keep the fine-tuned model weights close to the pre-trained ones. Let the pre-trained image encoder weights be $\theta_{v(0)}$ and the encoder weights at time step t during fine-tuning be $\theta_{v(t)}$. Then, the fine-tuning loss in this case becomes

$$\mathcal{L}_{\text{L2SP}} = \mathcal{L}_{\text{CE}} + \lambda_{\text{L2SP}}||\theta_{v(t)} - \theta_{v(0)}||_2^2. \tag{1}$$

   Note that the ℓ_2 distance is only computed between the weights of the pre-trained and fine-tuned image encoders.

[1] Note, the vision encoder first extracts D_v dimensional image features and then projects them to a D dimensional space via a linear embedder $\mathbf{w}_v : \mathbb{R}^{D_v} \to \mathbb{R}^D$. Similarly for the text encoder.

[2] The natural language class names are often augmented with a prompt like "an image of a {class name}."

4. **LP-init-L2SP**: This is similar to ZS-init-L2SP but the linear head $\mathbf{w}$ is initialized by performing linear probing on the downstream dataset first. The loss for end-to-end fine-tuning then is the same as in Equation (1) [3].
5. **FLYP** Goyal et al. (2022): The Fine-tune Like You Pre-train or FLYP baseline fine-tunes both the image and the text encoders of CLIP and uses contrastive loss $\mathcal{L}_{\text{cont}}$ instead of cross-entropy $\mathcal{L}_{\text{CE}}$ for fine-tuning on the downstream task. The parameters being fine-tuned here are $\theta = \{\theta_v, \theta_t\}$, i.e., both image and text encoders of CLIP.
6. **FLYP-CE** Goyal et al. (2022): This is an ablation on FLYP where, instead of using contrastive loss, the fine-tuning is done using cross-entropy loss $\mathcal{L}_{\text{CE}}$, taking the cosine similarities between image and text embeddings as logits. Note that similar to FLYP, in this case as well, both image and text encoders are fine-tuned end-to-end.

3 The crippling effect of end-to-end fine-tuning

The contrastive pre-training dataset of CLIP contains 400 million (image, text) pairs scraped from the internet Radford et al. (2021). Consequently, any downstream task that CLIP is fine-tuned on is highly likely to contain only a small fraction of concepts compared to what it has already been exposed to during pre-training. To investigate the impact of fine-tuning, here we perform a thorough study benchmarking 6 fine-tuning methods on 10 downstream classification tasks. We find that for most of these methods, while the fine-tuned model attains excellent improved performance on the downstream task itself, its general ability to recognize concepts outside the task is significantly reduced over the course of fine-tuning. We call this phenomenon ***concept forgetting*** and find this to be an undesirable effect of most fine-tuning methods. To explore this in detail, in this section, we first discuss how we quantify concept forgetting, then we propose our benchmarking setup for fine-tuning methods and finally present our observations.

3.1 Quantifying Concept Forgetting

During ZS and LP evaluation of a model (refer §2) the pre-trained image encoder weights θ_v remain frozen and unchanged irrespective of the downstream task at hand. Therefore, ZS and LP performance on a specific downstream task can be a good indicator of the pre-trained model's existing knowledge about the task. This is the hypothesis we base our analysis on.

While ZS accuracy is based on how well the text encoder representations can form a linear classifier in the image encoder's feature space, LP performance is indicative of whether the image encoder representations are linearly separable in the first place. Note that this distinction is important. Fine-tuning the weights of just the image encoder θ_v can lead to a situation where its representations are no longer well-aligned with the text encoder. Even so, for a given task, if the image encoder representations are linearly separable, as captured by its LP accuracy, it shows that the model is still able to recognize concepts involved in the downstream task, thereby indicating the preservation of knowledge on the task. Thus, after fine-tuning, ZS accuracy may not reflect a model's ability to recognize concepts in a task and LP accuracy is a better candidate to do so. We illustrate this point in Figure 1.

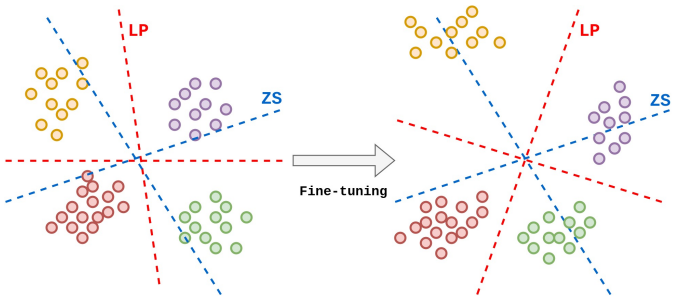


Figure 1: **Pictorial representation of classifiers from zero-shot (ZS) and linear probe (LP)**. While ZS can separate representations pre fine-tuning, the representations learnt after fine-tuning may not necessarily be linearly separable by the same ZS classifier anymore. However, if an LP is able to classify representations after fine-tuning, the model can still recognize the concepts involved.

[3]To the best of our knowledge, LP-init-L2SP, has not been evaluated or benchmarked prior to this work.

Therefore, in order to quantify concept forgetting on a particular task defined on a dataset $\mathcal{D}$, we measure the difference in LP accuracy between the pre-trained and the fine-tuned image encoders on $\mathcal{D}$. To formalize this, let $f_{\theta_{v(0)}}$ be the pre-trained image encoder and f_{θ_v} the one obtained via fine-tuning on a dataset $\mathcal{D}_{\text{ft}}$. Furthermore, let $\mathcal{A}_{\text{LP}}(f_{\theta_v}, \mathcal{D})$ represent the LP accuracy of image encoder f_{θ_v} on dataset $\mathcal{D}$. Then, we define the change in LP accuracy on $\mathcal{D}$ between pre-trained and fine-tuned models $\Delta_{\text{LP}}(\mathcal{D}, f_{\theta_v}, f_{\theta_{v(0)}})$ (or in short, Δ_{LP}) as:

$$\Delta_{\text{LP}}(\mathcal{D}, f_{\theta_v}, f_{\theta_{v(0)}}) = \mathcal{A}_{\text{LP}}(f_{\theta_v}, \mathcal{D}) - \mathcal{A}_{\text{LP}}(f_{\theta_{v(0)}}, \mathcal{D}) \tag{2}$$

Note, f_{θ_v} is fine-tuned on $\mathcal{D}_{\text{ft}}$, not on $\mathcal{D}$. The dataset $\mathcal{D}$ here represents a target dataset on which we would like to monitor/quantify concept forgetting. Clearly, if we use $\mathcal{D}_{\text{ft}}$ instead, we expect $\Delta_{\text{LP}}(\mathcal{D}_{\text{ft}}, f_{\theta_v}, f_{\theta_{v(0)}})$ to increase over the course of fine-tuning. However, the main objective here is to quantify the effect of fine-tuning on a task defined on a dataset $\mathcal{D}$ that was not part of the fine-tuning procedure itself ($\mathcal{D} \neq \mathcal{D}_{\text{ft}}$). A negative value of Δ_{LP} indicates **concept forgetting**, a zero indicates **knowledge accumulation** and a positive value indicates **knowledge gain** or positive forward transfer Lopez-Paz & Ranzato (2017) on the task under inspection. We would like to highlight that the dataset $\mathcal{D}$ here is a user-defined dataset on which monitoring the effect of fine-tuning would be desirable.

Catastrophic forgetting vs concept forgetting Catastrophic forgetting McCloskey & Cohen (1989); Kemker et al. (2018); Kirkpatrick et al. (2017) is a well-known phenomenon which signifies how, when a model is trained on a new task, its performance on the previous task drops catastrophically. For example, if a model trained on ImageNet is then trained on say CIFAR100, it significantly loses its performance on ImageNet. Though *concept forgetting* mentioned in this work is very similar to catastrophic forgetting and we do not claim much conceptual novelty here, there is a subtle difference that we believe requires a distinction between the two. The major difference lies in the fact that in the pre-foundation model era, the pre-training datasets were much smaller and the pre-training tasks were fully supervised, for example, classification on the ImageNet dataset. Therefore, it was possible to roughly quantify the degree of damage the model had on the pre-trained task (using pre-trained dataset) once it was fine-tuned on a new downstream task. However, in the case of foundation models, it is not trivial to quantify what the model knows as it was pre-trained on several millions or even billions of examples using self-supervised training and the pre-training dataset is often inaccessible. Therefore, it is not possible to quantify exactly what the fine-tuned model forgot and the catastrophe therein. Hence, it is necessary to devise poking and probing mechanisms, similar to the ones we present in this work using LP, to quantify the effect of fine-tuning on a smaller but relevant domain of concepts (represented by the dataset $\mathcal{D}$ above).

3.2 Benchmarking concept forgetting

To quantify concept forgetting, here we use CLIP Radford et al. (2021) ViT-B/32 pre-trained on the OpenAI dataset and released in the OpenCLIP repository Ilharco et al. (2021) and measure its LP performance over fine-tuning on 10 different image classification downstream tasks with a high variability in their semantic concepts. These datasets, along with their respective train/test splits are:

1. *Stanford Cars* Krause et al. (2013) containing $16,185$ images of 196 classes of cars with a train/test split of 8144 and 8041 images respectively,
2. *CIFAR-10/100 (C10/100)* Krizhevsky et al. (2009) containing $60,000$ images of vehicles, flora and fauna, divided into 10/100 classes with the train/test split having $50,000$ and $10,000$ images respectively,
3. *DTD* Cimpoi et al. (2014) containing 3760 images of 47 classes of textures found in the wild with 1880 images each in the train and test sets,
4. *EuroSAT* Helber et al. (2019) containing $25,000$ samples with 10 categories of satellite images of landscapes and $19,600/5400$ training/test images respectively,
5. *GTSRB* Stallkamp et al. (2012) containing $39,270$ images of 43 classes of German traffic signs with $26,640$ training images and $12,630$ test images,
6. *MNIST* LeCun et al. (1998) containing $60,000$ training images and $10,000$ test images of 10 handwritten digits from 0 to 9 in grayscale,

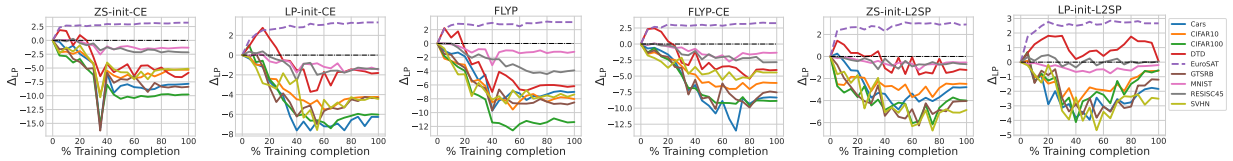
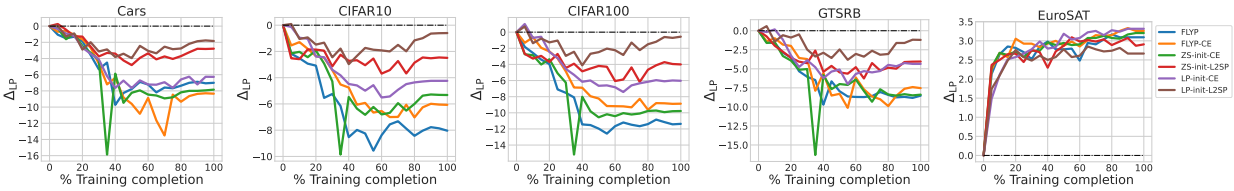


Figure 2: **Test set Δ_{LP} for models fine-tuned on EuroSAT using different fine-tuning methods.** While EuroSAT Δ_{LP} rises, Δ_{LP} on all other datasets is almost always negative throughout the fine-tuning with a sole exception of DTD when fine-tuned using LP-init-L2SP. See §3.2 for details.



(a) (b) (c) (d) (e)

Figure 3: **Test set Δ_{LP} evaluated on different datasets for models fine-tuned on EuroSAT via various fine-tuning methods.** The L2SP baselines (particularly LP-init-L2SP) have the lowest negative Δ_{LP} on datasets other than EuroSAT.

7. *RESISC45* (R45) Cheng et al. (2017) containing 25, 200 samples with 45 classes of various remote sensing image scenes with the train/test split having 18, 900 and 6300 images respectively,
8. *SVHN* Netzer et al. (2011) containing a total of 99, 289 colour images of street view house numbers, each image being categorized into one of 10 digits with 73, 257 training samples and 26, 032 test samples.
9. *ImageNet* Deng et al. (2009) containing a total of 1.28 million training images and 50, 000 validation images of 1000 classes.

Finally, for this study, we use the 6 end-to-end fine-tuning methods discussed in §2. The training details can be found in Appendix A.

Fine-tuning causes concept forgetting. In Figure 2, we present the Δ_{LP} for models fine-tuned on EuroSAT using the 6 fine-tuning methods discussed in §2. Across all fine-tuning methods, we observe that *while as expected, the performance on EuroSAT test increases and Δ_{LP} is positive (also see Figure 3e), performance on 8 other downstream datasets decreases and Δ_{LP} for all of them is negative.* This indicates that *all 6 fine-tuning methods suffer from concept forgetting.* The only exception to this is when we evaluate on DTD and the model is fine-tuned using LP-init-L2SP. We observe Δ_{LP} to be mostly positive before it goes to zero at the end of fine-tuning. This is an interesting case as it indicates that LP-init-L2SP on EuroSAT might actually be helping increase knowledge about DTD before the fine-tuning becomes too specific to EuroSAT. This may be an example of positive forward transfer Lopez-Paz & Ranzato (2017); Chaudhry et al. (2018) and exploring why such knowledge accumulation happens is an interesting avenue for further research.

Next, we compare between these fine-tuning methods by showing Δ_{LP} for different downstream datasets in Figure 3, where it is evident that *LP-init-L2SP consistently outperforms other baselines in lowering concept forgetting and preserving the fine-tuned model's original performance across multiple downstream tasks.* This is observable from its low negative Δ_{LP} compared to other baselines. While it suffers an initial dip in performance in the early stages of fine-tuning, in the later stages, LP-init-L2SP regains the accuracy, often ending up with a near zero Δ_{LP}. Its impressive performance on concept forgetting however, does seem to come at the cost of a relatively lower Δ_{LP} on the fine-tuning task, i.e., EuroSAT, itself. In this regard, note that most fine-tuning methods are specifically designed to maximise performance on the downstream fine-tuning task. Thus, the performance in Figure 3e is conventionally the primary evaluation criterion for a fine-tuning method. However, as shown here, concept forgetting is an undesirable additional effect of fine-tuning and requires its own evaluation. Our observations for other datasets are similar (see Appendix C). In what follows, we first investigate LP-init-L2SP further to gain insights on why it preserves concepts better

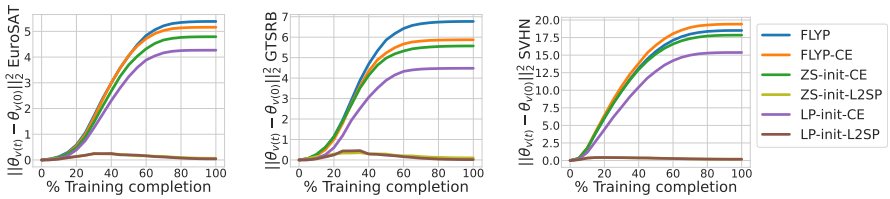


Figure 4: ℓ_2 **distance in the parameter space** $||\theta_{v(t)} - \theta_{v(0)}||_2^2$ of the image encoder over the course of fine-tuning. Except L2SP, all other baselines diverge away from their pre-trained counterparts.

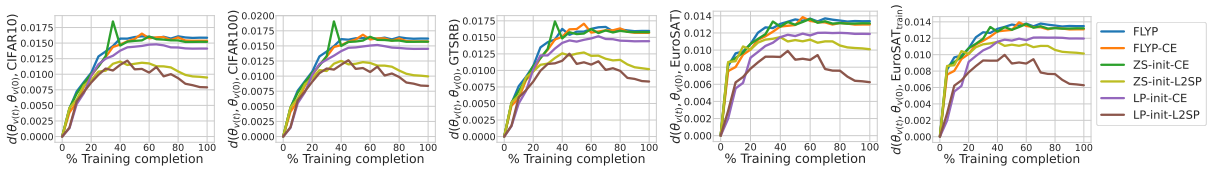


Figure 5: ℓ_2 **distance in feature space** $d(\theta_{v(t)}, \theta_{v(0)}, \mathcal{D})$ (see eq. (3)) between image encoders computed over the course of fine-tuning on EuroSAT. The ℓ_2 distance is computed for EuroSAT train and test sets as well as CIFAR10, CIFAR100 and GTSRB datasets.

than other baselines, and then use those insights to propose a new fine-tuning method that significantly outperforms all other baselines.

4 Can preserving features help?

The L2SP regularizer (eq. (1)) enforces the model $f_{\theta_{v(t)}}$ at time-step t of fine-tuning to be in the vicinity of the pre-trained model $f_{\theta_{v(0)}}$ by minimizing the ℓ_2 distance between the two in the **parameter space**. As evident from §3, this simple regularizer has a promising impact on the model's ability to avoid concept forgetting. To understand how correlated the parameter space ℓ_2 distance $||\theta_{v(t)} - \theta_{v(0)}||_2^2$ is to concept forgetting, in Figure 4, we show how ℓ_2 distance changes as we fine-tune different datasets (EuroSAT, GTSRB and SVHN) using all the discussed fine-tuning methods. *From Figure 4, relatively speaking, all the fine-tuning methods except the two L2SP baselines (ZS-init-L2SP and LP-init-L2SP) cause the parameters of the fine-tuned model to move away from its pre-trained counterpart.*

Though regularizing the fine-tuned model to be in the vicinity of the pre-trained one shows a promising effect in preserving concepts, vicinity in the parameter space need not necessarily capture the input-output behaviour of the pre-trained model. In fact, it is trivial to construct two different sets of model parameters, far in ℓ_2 distance, outputting similar values in a specific domain. Additionally, regularizing in the parameter space might not keep the fine-tuned model in the *desired* vicinity that preserves the pre-trained knowledge and performs well on the downstream task at the same time. Therefore, we conjecture that the vicinal distance in feature space as opposed to parameter space, might be a better indicator of the similarity of encoded concepts in the model. Indeed, in the end, the internal representation space is where models encode patterns. Thus, in Figure 5, we inspect ℓ_2 distance in the feature space over fine-tuning using the following distance function:

$$d(\theta_{v(t)}, \theta_{v(0)}, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} ||\Phi_{\theta_{v(t)}}(\mathbf{x}_i) - \Phi_{\theta_{v(0)}}(\mathbf{x}_i)||_2^2, \tag{3}$$

where, $\Phi_{\theta_{v(t)}}(\mathbf{x}_i)$ represents the features of the model with parameters $\theta_{v(t)}$ at time t for a given sample $\mathbf{x}_i$, and N is the number of samples in the dataset. Note that the feature vector $\Phi_{\theta_{v(t)}}(\mathbf{x}_i)$ is obtained by concatenating various internal representations (not just the last layer features) of the network architecture, similar to the perceptual features presented in Zhang et al. (2018). The exact details as to how we compute the concatenated feature vector for a ViT-B/32 model is mentioned in Appendix B, and similar plots for other datasets is shown in Appendix C.

A strong correlation between concept forgetting and feature-space distance. Similar to our observations in Figure 4 (parameter space), we note that except L2SP, all other fine-tuning methods suffering significantly from concept forgetting cause the fine-tuned model to move away from the pre-trained model

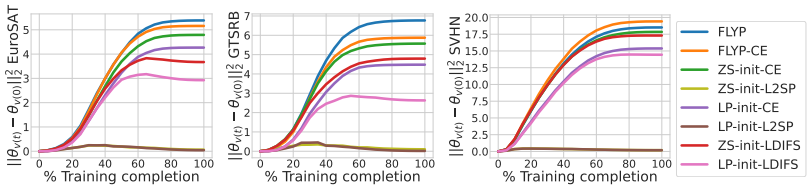


Figure 6: ℓ_2 **distance in weight space** $||\theta_{v(t)} - \theta_{v(0)}||_2^2$ between the image encoder fine-tuned to the current time-step $f_{\theta_{v(t)}}$ and the pre-trained image encoder $f_{\theta_{v(0)}}$ over the course of fine-tuning.

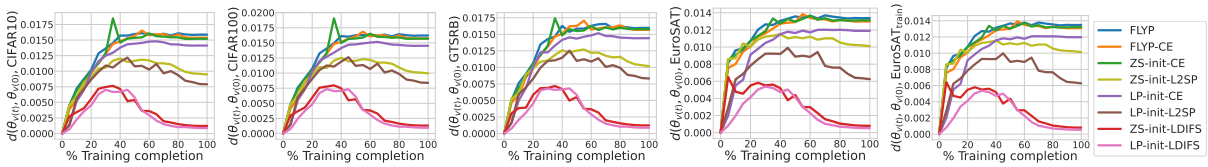


Figure 7: ℓ_2 **distance in feature space** $d(\theta_{v(t)}, \theta_{v(0)}, \mathcal{D})$, between image encoders, $f_{\theta_{v(t)}}$ and $f_{\theta_{v(0)}}$ computed over different fine-tuning methods for models fine-tuned on EuroSAT.

in terms of feature-space distance as well (Equation (3)). In case of ZS-init-L2SP and LP-init-L2SP, while initially diverging, the fine-tuned models recover their pre-trained behaviour to a certain extent in the later stages of fine-tuning. It is important to note that this observation is consistent for the fine-tuning training and test sets as well as for all other datasets.

Based on the results shown in Figure 3 and Figure 5 we conclude that the *fine-tuning methods which cause the model to significantly diverge away from the pre-trained model either in parameter space or in feature space suffer more from concept forgetting.* This observation, combined with our conjecture above leads to a natural extension of fine-tuning where we use $d(\theta_{v(t)}, \theta_{v(0)}, \mathcal{D})$ (distance in the feature space) as the regularizer. We call this regularizer **LDIFS: ℓ_2 distance in Feature Space**. The complete fine-tuning objective then becomes:

$$\mathcal{L}_{\text{LDIFS}} = \mathcal{L}_{\text{CE}} + \lambda_{\text{LDIFS}} \cdot d(\theta_{v(t)}, \theta_{v(0)}, \mathcal{D}_{\text{train}}) \tag{4}$$

where $\mathcal{D}_{\text{train}}$ is the training or fine-tuning set and λ_{LDIFS} is the regularization coefficient. Note that similar to L2SP, we can initialize the linear head with both zero-shot weights or linear probe weights leading to two variants: ZS-init-LDIFS and LP-init-LDIFS.

Analysing LDIFS on parameter space and feature space distance: In Figure 6 and Figure 7, we plot the ℓ_2 distance in parameter/weight and feature space respectively (same as Figure 4 and Figure 5), but with the LDIFS added in. In the weight space, LDIFS, while lower than other fine-tuning methods, has a relatively higher ℓ_2 distance from the pre-trained model compared to L2SP. On the contrary, in the feature space, LDIFS consistently gets the lowest distance from the pre-trained model. This is expected and was indeed the purpose behind LDIFS's design which is to minimize the difference in input-output behaviour, captured through feature space distance between the fine-tuned and pre-trained models and not necessarily constrain the parameter space of the fine-tuned model to lie close to the pre-trained model. Finally, note from Figure 7, that although we are applying the LDIFS regularizer only on the EuroSAT samples during fine-tuning, the preservation of features extends even to other datasets like CIFAR10, CIFAR100 and GTSRB, as seen from the low feature space distance even on these datasets. This indicates, that we don't need third party datasets during fine-tuning to preserve the pre-trained model's behaviour and regularizing on samples just from the fine-tuning dataset can be sufficient.

LP-init-LDIFS significantly reduces concept forgetting. First, we evaluate LDIFS on the same setting as in §3 and present the Δ_{LP} over fine-tuning in Figure 8 (additional results presented in Appendix C). Second, in Table 1, we report the $\mathcal{A}_{\text{LP}}$% on the fine-tuning test set and the Δ_{LP} averaged over other tasks. $\mathcal{A}_{\text{LP}}$% measures performance on the fine-tuning task itself and Δ_{LP} provides an estimate of the fine-tuned model's level of concept forgetting on the remaining tasks. For downstream tasks which are not ImageNet, we report Δ_{LP} averaged over the remaining 8 tasks, except ImageNet. For ImageNet, we leave out CIFAR-10/100 when calculating performance over other tasks since all of CIFAR-10/100's classes are within the set of

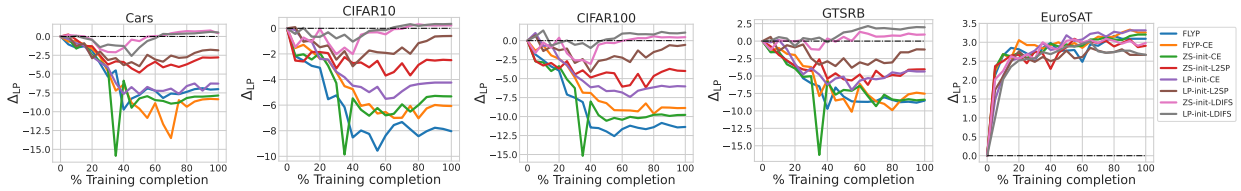


Figure 8: **Test set Δ_{LP} evaluated on different datasets for models fine-tuned on EuroSAT.** LDIFS (Our) baselines (both ZS-init and LP-init) provides the best results in terms of avoiding concept forgetting without affecting downstream performance on EuroSAT.

Dataset	Fine-tuning baselines															
	FLYP		FLYP-CE		ZS-init-CE		LP-init-CE		ZS-init-L2SP		LP-init-L2SP		ZS-init-LDIFS		LP-init-LDIFS	
	$\mathcal{A}_{\text{LP}}(\uparrow)$	$\Delta_{\text{LP}}(\uparrow)$	$\mathcal{A}_{\text{LP}}(\uparrow)$	$\Delta_{\text{LP}}(\uparrow)$	$\mathcal{A}_{\text{LP}}(\uparrow)$	$\Delta_{\text{LP}}(\uparrow)$	$\mathcal{A}_{\text{LP}}(\uparrow)$	$\Delta_{\text{LP}}(\uparrow)$	$\mathcal{A}_{\text{LP}}(\uparrow)$	$\Delta_{\text{LP}}(\uparrow)$	$\mathcal{A}_{\text{LP}}(\uparrow)$	$\Delta_{\text{LP}}(\uparrow)$	$\mathcal{A}_{\text{LP}}(\uparrow)$	$\Delta_{\text{LP}}(\uparrow)$	$\mathcal{A}_{\text{LP}}(\uparrow)$	$\Delta_{\text{LP}}(\uparrow)$
Cars	86.06	−0.01	84.77	−1.67	83.48	−1.56	84.95	−0.63	82.64	−1.07	83.87	**0.47**	85.52	−0.36	85.26	−0.18
CIFAR10	97.71	−3.35	97.58	−1.17	97.73	−1.6	97.71	−0.81	97.7	1.04	97.66	1.16	97.36	1.03	97.24	**1.18**
CIFAR100	88.98	−1.16	88.77	−0.5	88.6	−0.96	88.41	−0.11	87.84	0.65	86.94	**1.03**	87.94	0.82	88.99	0.86
DTD	76.12	−4.92	73.46	−3.44	77.18	−3.01	72.18	−1.76	72.98	−3.71	74.63	0.01	78.14	0.19	75.27	**0.53**
EuroSAT	98.65	−6.7	98.8	−5.44	98.76	−5.72	98.87	−3.75	98.46	−2.58	98.2	−0.85	98.54	0.92	98.22	**1.32**
GTSRB	99.26	−8.5	99.0	−3.76	98.52	−5.9	98.53	−0.94	97.4	−3.05	95.0	1.18	98.45	1.07	97.81	**1.27**
MNIST	99.62	−8.64	99.63	−7.53	99.67	−8.76	99.68	−6.02	99.43	−2.93	99.18	1.49	99.6	1.8	99.52	**2.64**
RESISC45	95.84	−5.42	95.79	−3.32	95.76	−3.79	95.56	−2.27	94.05	−0.91	94.13	0.66	95.22	0.41	95.13	**0.9**
SVHN	97.44	−10.74	97.4	−10.40	97.3	−11.12	97.5	−8.73	96.5	−2.78	96.54	−2.11	96.97	−0.68	96.95	**−0.29**
ImageNet	82.26	−1.6	82.18	−1.39	82.02	−1.26	82.12	−0.87	80.9	−0.24	80.78	−0.1	82.14	0.13	82.21	**0.35**

Table 1: **Test set accuracy $\mathcal{A}_{\text{LP}}$% and average Δ_{LP} computed over other datasets for models fine-tuned on 10 image classification tasks.** $\mathcal{A}_{\text{LP}}$ shows performance on the fine-tuning task itself and Δ_{LP} shows the level of concept forgetting on other tasks (higher Δ_{LP} shows lower concept forgetting and vice versa). The best Δ_{LP} numbers are shown in bold.

ImageNet classes. We use the remaining 7 tasks to quantify forgetting. Full set of results for all baselines can be found in Table 8 and an alternative visualization of these results are in Figure 14 of the Appendix. Our observations are as follows.

1. ***Competitive downstream accuracy***: From Table 1, we note that there is no clear winner in terms of accuracy on the downstream fine-tuning task itself. Additionally, from Figure 8 and Table 1, it is also evident that LDIFS is competitive with other fine-tuning baselines in terms of accuracy on the downstream fine-tuning task itself.
2. ***Minimal concept forgetting***: *LP-init-LDIFS, significantly minimizes concept forgetting over the course of fine-tuning.* This is evident from Figure 8 which shows its noticeably higher Δ_{LP} on other downstream tasks over the course of fine-tuning. It is also apparent from its consistently high Δ_{LP} scores in Table 1. Particulatly, LP-init-LDIFS gets the highest average Δ_{LP} on other tasks for 8 out of 10 fine-tuning cases. This indicates a significantly lower level of concept forgetting.
3. ***Positive Forward Transfer***: From Table 1, we also observe that in 8 out of 10 fine-tuning tasks, LP-init-LDIFS results in a positive average Δ_{LP} on other tasks, thereby generally achieving a positive forward transfer. This is not the case for other baselines where Δ_{LP} is generally negative indicating concept forgetting. The two exceptions for LP-init-LDIFS are when the finetuning tasks $\mathcal{D}_{\text{ft}}$ are Stanford Cars and SVHN, in which case the average Δ_{LP} on other tasks is negative. This observation also highlights how the ordering of fine-tuning tasks can also impact forward transfer. For instance, from Table 8, when fine-tuning on EuroSAT, LP-init-LDIFS achieves a Δ_{LP} of $+3.03$ on SVHN, whereas in the reverse scenario, fine-tuning on SVHN leads to a negative Δ_{LP} of -0.78 on EuroSAT.

We have similar conclusions on 3 other CLIP architectures: ResNet-50, VIT-B/16 and ViT-L/14, a FLAVA ViT-B/16 Singh et al. (2022), a DINO-v2 ViT-B/14 and ViT-L/14 Oquab et al. (2023) and Masked Autoencoder He et al. (2022) ViT-B/16 and ViT-L/16 models. See Appendix C.5 for details and results.

5 A nudge towards continual fine-tuning

Our results above indicate that fine-tuning on LP-init-LDIFS can make the foundation model learn new downstream information without forgetting pre-trained concepts. A natural question that then arises is: *Can we fine-tune on a sequence of downstream tasks without forgetting concepts?* For an ideal fine-tuning method, the final model should attain state-of-the-art performance on all fine-tuned tasks while still maintaining its

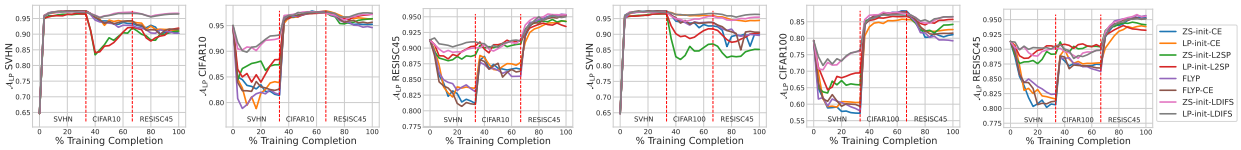


Figure 9: $\mathcal{A}_{\text{LP}}$ for sequence: SVHN $\rightarrow$ CIFAR10 $\rightarrow$ RESISC45 (left 3 plots) and ii) SVHN $\rightarrow$ CIFAR100 $\rightarrow$ RESISC45 (right 3 plots). Vertical red line indicates a switch in the fine-tuning tasks.

pre-trained knowledge. We empirically investigate this question by training on 3 sequences of 3 tasks each: **a)** SVHN $\rightarrow$ C10 $\rightarrow$ R45, **b)** SVHN $\rightarrow$ C100 $\rightarrow$ R45 and **c)** SVHN $\rightarrow$ Cars $\rightarrow$ R45. Note that this setup is similar to continual learning Chaudhry et al. (2018); Rebuffi et al. (2017); Lopez-Paz & Ranzato (2017) but for pre-trained foundation models. Due to their impressive performance on a wide range of downstream tasks, continual fine-tuning of foundation models is relatively unexplored in the literature. Nonetheless, as stated in §1, this is an important problem to investigate from the perspective of updating a foundation model with new previously unknown knowledge without forgetting previously known ones.

Quantifying concept forgetting in continual setups. Let the sequence of fine-tuning datasets be $\mathcal{D}_1 \rightarrow \mathcal{D}_2, \rightarrow ..., \mathcal{D}_k$ and the corresponding sequence of models be $f_{\theta_0}, ..., f_{\theta_k}$ with f_{θ_0} being the pre-trained model. In order to then quantify concept forgetting on a task $\mathcal{D}$ over this sequence of models, we extend the Δ_{LP} from Equation (2) as follows:

$$\Delta_{\text{LP}}(\mathcal{D}, f_{\theta_k}, \{f_{\theta_0}, f_{\theta_1}, ..., f_{\theta_{k-1}}\}) = \mathcal{A}_{\text{LP}}(f_{\theta_k}, \mathcal{D}) - \max_{i \in \{0,...,k-1\}} \mathcal{A}_{\text{LP}}(f_{\theta_i}, \mathcal{D}). \tag{5}$$

Hence, similar to Chaudhry et al. (2018), we find the difference in LP performance between the final fine-tuned model f_{θ_k} and the model having the maximum LP performance in the sequence $\{f_{\theta_0}, f_{\theta_1}, ..., f_{\theta_{k-1}}\}$. This accounts for the possibility of positive forward transfer on $\mathcal{D}$ over the sequence of fine-tuning tasks.

LP-init-LDIFS significantly reduces concept forgetting in continual setups. In Table 2, we present the Δ_{LP} and $\mathcal{A}_{\text{LP}}$ for the continual setup for all fine-tuning baselines. For each task sequence, we report the performance on datasets in the sequence in the first three rows and in the fourth row, we report the average performance on 6 other datasets. From Table 2, we observe:

1. For all 3 sequences, LP-init-LDIFS has minimal concept forgetting on tasks which appear earlier in the fine-tuning sequence, as well as other datasets which are not used for fine-tuning.
2. At the same time, LP-init-LDIFS is very competitive on accuracy with the best fine-tuning methods and consistently outperforms L2SP on accuracy on the last fine-tuning task, i.e., RESISC45.

These observations are further complemented in Figure 9 where we can see LP-init-LDIFS to have minimal forgetting on prior tasks, SVHN and C10/100 without compromising on performance on the last task.

LP-init-LDIFS outperforms classic continual learning methods. Due to the similarity of the proposed setup in this section with classic continual learning, we empirically compare performance of LP-init-LDIFS with 5 well-known continual learning baselines: LwF Li & Hoiem (2017), LFL Jung et al. (2016), iCARL Rebuffi et al. (2017), Distillation + Retrospection (D+R) Hou et al. (2018) and ZSCL Zheng et al. (2023) on our sequence setup. Results are in Table 3. Again, the results indicate that LP-init-LDIFS performs better than all other continual learning baselines, both in preventing forgetting as well as obtaining the best performance on the last task. This is evident from its consistently high Δ_{LP} and $\mathcal{A}_{\text{LP}}$ on all 3 sequences.

6 Related Work & Discussion

In this section, we first provide a brief discussion on the different types of fine-tuning approaches apart from end-to-end fine-tuning which have been applied on foundation models. We also discuss the relation and differences between our proposed LDIFS regularizer and previous works on continual learning which particularly use feature-distillation methods.

Fine-tuning in foundation models: There are many flavours of fine-tuning which improve on CLIP's zero-shot performance. A popular approach is prompt tuning where, methods like CoOp Zhou et al. (2022b), CoCoOp Zhou et al. (2022a), TPT Shu et al. (2022), Chain of Thought prompt tuning Ge et al. (2023),

Dataset Fine-tune	Eval	Fine-tuning baselines FLYP $\Delta_{LP}(\uparrow)$	FLYP $\mathcal{A}_{LP}(\uparrow)$	FLYP-CE $\Delta_{LP}(\uparrow)$	FLYP-CE $\mathcal{A}_{LP}(\uparrow)$	ZS-init-CE $\Delta_{LP}(\uparrow)$	ZS-init-CE $\mathcal{A}_{LP}(\uparrow)$	LP-init-CE $\Delta_{LP}(\uparrow)$	LP-init-CE $\mathcal{A}_{LP}(\uparrow)$	ZS-init-L2SP $\Delta_{LP}(\uparrow)$	ZS-init-L2SP $\mathcal{A}_{LP}(\uparrow)$	LP-init-L2SP $\Delta_{LP}(\uparrow)$	LP-init-L2SP $\mathcal{A}_{LP}(\uparrow)$	ZS-init-LDIFS $\Delta_{LP}(\uparrow)$	ZS-init-LDIFS $\mathcal{A}_{LP}(\uparrow)$	LP-init-LDIFS $\Delta_{LP}(\uparrow)$	LP-init-LDIFS $\mathcal{A}_{LP}(\uparrow)$
SVHN → C10 → R45	SVHN	−7.06	90.3	−5.77	91.61	−7.13	90.29	−6.46	90.97	−5.41	91.01	−4.53	91.93	−0.43	96.66	**−0.41**	**96.68**
	CIFAR10	−3.16	94.61	−1.92	95.65	−2.31	95.25	−1.57	96.31	−1.22	96.33	−0.25	97.26	−0.26	97.18	**−0.21**	**97.41**
	RESISC45	**4.06**	**95.33**	3.89	95.16	4.0	95.3	2.98	94.29	2.97	94.24	2.16	93.44	3.94	95.33	3.7	95.0
	Others	−6.59	78.91	−4.9	81.2	−5.08	80.91	−4.24	82.13	−1.82	85.27	−0.01	86.89	−0.3	86.52	**0.1**	**87.08**
SVHN → C100 → R45	SVHN	−7.75	89.64	−6.71	90.65	−7.28	90.05	−2.73	94.42	−11.36	85.18	−6.12	90.42	−1.5	95.47	**−0.65**	**96.32**
	CIFAR100	−8.85	79.22	−6.38	81.55	−7.18	81.08	−3.04	82.63	−3.46	84.2	−0.88	85.72	−0.99	86.45	**−0.3**	**86.54**
	RESISC45	**4.38**	**95.68**	3.9	95.21	4.13	95.4	2.51	93.81	2.79	94.13	1.9	93.21	4.29	95.59	3.83	95.11
	Others	−5.23	83.09	−4.68	83.97	−4.65	83.76	−4.02	85.14	−1.61	87.52	−0.37	89.04	−0.56	88.36	**−0.23**	**89.12**
SVHN → Cars → R45	SVHN	−1.26	96.07	−1.51	95.76	−1.45	95.93	−0.76	96.58	−1.14	95.42	−0.44	95.98	−0.49	96.56	**−0.17**	**96.9**
	Cars	−3.61	81.21	−4.3	76.87	−4.18	76.96	−8.36	71.6	−0.88	81.18	−0.4	81.82	−0.61	82.89	**0.47**	**84.23**
	RESISC45	**4.13**	**95.43**	3.81	95.08	3.89	95.17	3.0	94.35	2.83	94.13	2.13	93.43	3.94	95.22	3.73	95.27
	Others	−6.68	81.91	−5.48	83.2	−4.93	83.38	−4.51	84.39	−2.91	85.74	−1.67	87.15	−0.1	88.75	**0.23**	**89.39**

Table 2: Δ_{LP} and $\mathcal{A}_{LP}$% for models fine-tuned on (SVHN, C10, R45), (SVHN, C100, R45) & (SVHN, Cars, R45) sequences. The first 3 rows show performance on fine-tuned tasks and the third row shows averaged performance on 6 other datasets.

Dataset Fine-tune	Eval	Continual Learning baselines LwF $\Delta_{LP}(\uparrow)$	LwF $\mathcal{A}_{LP}(\uparrow)$	LFL $\Delta_{LP}(\uparrow)$	LFL $\mathcal{A}_{LP}(\uparrow)$	iCaRL $\Delta_{LP}(\uparrow)$	iCaRL $\mathcal{A}_{LP}(\uparrow)$	D+R $\Delta_{LP}(\uparrow)$	D+R $\mathcal{A}_{LP}(\uparrow)$	ZSCL $\Delta_{LP}(\uparrow)$	ZSCL $\mathcal{A}_{LP}(\uparrow)$	LP-init-LDIFS (Ours) $\Delta_{LP}(\uparrow)$	LP-init-LDIFS (Ours) $\mathcal{A}_{LP}(\uparrow)$
SVHN → C10 → R45	SVHN	−3.81	90.48	−3.21	91.9	−3.67	91.62	−2.78	93.3	−3.23	92.7	**−0.41**	**96.68**
	CIFAR10	−2.9	93.9	−2.32	94.88	−2.1	95.17	−1.9	95.41	−1.6	95.82	**−0.21**	**97.41**
	RESISC45	3.1	94.22	2.98	93.9	2.83	93.72	3.68	94.94	3.62	94.89	**3.7**	**95.0**
	Others	−4.2	80.73	−3.76	81.31	−4.11	80.78	−3.2	81.86	−2.8	83.1	**0.1**	**87.08**
SVHN → C100 → R45	SVHN	−4.34	89.48	−4.08	90.29	−4.31	90.97	−3.23	92.3	−3.92	91.81	**−0.65**	**96.32**
	CIFAR100	−3.25	83.24	−3.01	83.95	−3.13	84.06	−2.6	84.82	−2.13	85.07	**−0.3**	**86.54**
	RESISC45	3.21	93.8	3.62	94.91	3.54	94.87	3.71	95.08	3.65	94.96	**3.83**	**95.11**
	Others	−4.11	81.73	−3.8	82.04	−4.02	81.62	−3.43	82.17	−3.11	82.86	**−0.23**	**89.12**
SVHN → Cars → R45	SVHN	−3.64	91.43	−2.92	92.74	−3.13	91.75	−2.84	92.86	−2.72	92.98	**−0.17**	**96.9**
	Cars	−2.79	81.69	−2.64	81.82	−2.8	81.7	−2.12	82.11	−1.84	82.68	**0.47**	**84.23**
	RESISC45	3.34	93.92	3.55	94.96	3.58	94.97	3.72	95.19	3.63	95.04	**3.73**	**95.27**
	Others	−4.07	81.63	−3.6	82.24	−3.89	81.88	−3.12	82.73	−2.8	83.1	**0.23**	**89.39**

Table 3: Δ_{LP} and $\mathcal{A}_{LP}$% comparing LDIFS with 5 classic continual learning methods on the 3-task setup.

instead of hand-crafting prompts, learn the prompts specific to the fine-tuning task. Another category of methods uses adapters: CLIP-Adapter Gao et al. (2021), Tip-Adapter Zhang et al. (2021), Prompt Adapter Sun et al. (2023), SVL-Adapter Pantazis et al. (2022), which works on the principle of combining pre-trained features with a small non-linear adapter network where the adapter is fine-tuned on the downstream task at hand. Finally, along with end-to-end fine-tuning, there are weight space interpolation methods like Wise-FT Wortsman et al. (2022b), PAINT Ilharco et al. (2022b), task arithmetic Ilharco et al. (2022a) and model soups Wortsman et al. (2022a) which look at interpolating between pre-trained and fine-tuned models in the weight space in order to achieve the best of both worlds in terms of downstream task performance and robustness to distribution shift.

In Table 7 of the appendix, we compare linear probing, CoOp, CLIP-Adapter, Tip-Adapter and the classic end-to-end fine-tuning method, ZS-init-CE on downstream task performance on 9 image classification tasks. Not only do we find end-to-end fine-tuning to consistently outperform adapters and prompt tuning, but when there is a big performance gap between the linear probe and end-to-end fine-tuning, adapters and prompt tuning do not bridge that gap well. This observation reaffirms the importance of end-to-end fine-tuning and thereby necessitates the investigation of better end-to-end fine-tuning methods for tasks where the pre-trained model's performance is sub-par. In Table 6 in the appendix, we also show how concept forgetting in end-to-end fine-tuning methods used in our work can be further improved by combining them with Wise-FT.

Knowledge Distillation & Continual Learning: The LDIFS regularizer can be studied from the lens of knowledge distillation, an approach which has been applied to different ends like calibration He et al. (2023), pre-training Lee et al. (2022), transfer learning Zhou et al. (2022c), robust fine-tuning Mao et al. (2022) and continual learning Li & Hoiem (2017); Jung et al. (2016); Rebuffi et al. (2017); Hou et al. (2018). The main difference between these works and ours is that unlike classic distillation methods which mainly use the last layer features Jung et al. (2016) or the logits Li & Hoiem (2017); Rebuffi et al. (2017) directly for distillation, we concatenate features from shallower layers in the network when computing the LDIFS regularizer. Furthermore, while works like Zagoruyko & Komodakis (2016); Passban et al. (2021); Passalis & Tefas (2018); Heo et al. (2019) develop indirect methods of distilling from intermediate feature representations including additional attention layers Zagoruyko & Komodakis (2016); Passban et al. (2021), probability distribution matching Passalis & Tefas (2018), and hidden neuron activation boundaries Heo et al. (2019), LDIFS uses L2 distance between intemediate feature representations directly as the regularizer. This makes it simpler to implement without any additional layers to be trained on intermediate feature representations.

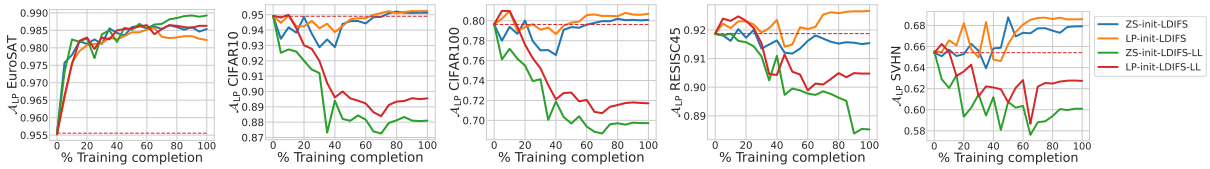


Figure 10: $\mathcal{A}_{\text{LP}}$ for models trained with LDIFS on EuroSAT using the full concatenated feature vector vs just the last layer (LL) feature vectors. Full feature vectors are crucial for LDIFS's performance.

Finally, the proposed modifications to distillation in LDIFS turn out to be crucial for performance. To investigate the importance of distilling from earlier features, in Figure 10, we compare performance of LDIFS when distilling from just last layer features (we call this ablation LL) during fine-tuning on EuroSAT. These plots provide evidence that using just the last layer features is not nearly as performant as using including the earlier features in the feature vector. We provide further insights on this in Appendix C.9. Finding which features contain pre-trained information for a downstream task is an interesting area of further research.

LDIFS vs LPIPS Zhang et al. (2018): One can find similarities between our proposed LDIFS regulariser and the LPIPS metric Zhang et al. (2018) popularly used for measuring perceptual similarity between images. However, while LPIPS uses feature space distance on a *pre-trained, frozen* model to find perceptual similarity between pairs or sets of images, LDIFS instead feature space distance between a pair of pre-trained and fine-tuned model for the same image, to preserve the input-output behaviour of the pre-trained model.

7 Conclusion & Remarks

We explore how end-to-end fine-tuning approaches often applied on foundation models can, in turn, significantly damage the model's ability to recognize concepts outside the fine-tuning task at hand, a phenomenon which we call concept forgetting. Such an effect is undesirable particularly for foundation models which have been specifically trained on millions of samples to encode information on a vast number of real-world concepts. However, we find that among fine-tuning methods, L2SP, which keeps the model near the pre-trained model in its parameter space suffers less from concept forgetting. From insights gained by analyzing L2SP, we also find that feature space distance provides a better definition for the vicinity of the pre-trained model as the pre-trained concepts are indeed encoded in the feature space. Our proposed new regularizer, LDIFS, encourages the features of fine-tuned model to be in the vicinity of the pre-trained ones. Through extensive evaluation on 10 different downstream classification tasks, as well as a continual fine-tuning setup of 3 different sequences of 3 tasks each, we showed that LDIFS significantly alleviates concept forgetting without impeding downstream task performance.

Limitations & Future work Though we investigated concept forgetting and proposed a promising fix, we believe that various other experiments and analyses along these lines would be valuable for the community. First, LDIFS has an associated hyperparameter λ_{LDIFS} which balances the objective of preserving pre-trained model behaviour with the objective of fine-tuning on the task at hand. Following convention, we cross-validate this hyperparameter on a held-out validation set (see Appendix A) from the task we fine-tune on. While this performs well, it is interesting to see if there can be better ways of setting this hyperparameter and in which cases the cross-validated hyperparameter may not yield the best model either at concept forgetting or downstream performance. Second, to compute LDIFS, we use a fixed set of features evenly spaced out in the network. In this regard, an analysis of which features are useful for encoding which concepts is still an open question worth investigating. Third, LDIFS tries to preserve the behaviour of the pre-trained model while fine-tuning. If there are biases and undesirable behaviours in the pre-trained model, these can percolate into the fine-tuned model. It is worth studying how such undesirable behaviours can be prevented from being distilled into the fine-tuned model while preserving the pre-trained knowledge. Fourth, understanding concept forgetting in other foundation model families such as Large Language Models, would be interesting. Finally, fundamentally defining what a "concept" is and the granularity at which we should study it for foundation models is important and requires research efforts to understand phenomena like "concept forgetting" beyond just performing task-based performance evaluation on a downstream dataset.

References

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 532–547, 2018.

Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. Clip-adapter: Better vision-language models with feature adapters. *arXiv preprint arXiv:2110.04544*, 2021.

Jiaxin Ge, Hongyin Luo, Siyuan Qian, Yulu Gan, Jie Fu, and Shanghang Zhan. Chain of thought prompt tuning in vision language models. *arXiv preprint arXiv:2304.07919*, 2023.

Sachin Goyal, Ananya Kumar, Sankalp Garg, Zico Kolter, and Aditi Raghunathan. Finetune like you pretrain: Improved finetuning of zero-shot vision models. *arXiv preprint arXiv:2212.00638*, 2022.

Guande He, Jianfei Chen, and Jun Zhu. Preserving pre-trained features helps calibrate fine-tuned language models. *arXiv preprint arXiv:2305.19249*, 2023.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019.

Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3779–3787, 2019.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Lifelong learning via progressive distillation and retrospection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 437–452, 2018.

Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, jul 2021. URL `https://doi.org/10.5281/zenodo.5143773`.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022a.

Gabriel Ilharco, Mitchell Wortsman, Samir Yitzhak Gadre, Shuran Song, Hannaneh Hajishirzi, Simon Kornblith, Ali Farhadi, and Ludwig Schmidt. Patching open-vocabulary models by interpolating weights. *arXiv preprint arXiv:2208.05592*, 2022b.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pp. 4904–4916. PMLR, 2021.

Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetting learning in deep neural networks. *arXiv preprint arXiv:1607.00122*, 2016.

Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Will Knight. Openai's ceo says the age of giant ai models is already over. *Wired, April*, 17:2023, 2023.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054*, 2022.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Seanie Lee, Minki Kang, Juho Lee, Sung Ju Hwang, and Kenji Kawaguchi. Self-distillation for further pre-training of transformers. *arXiv preprint arXiv:2210.02871*, 2022.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Xiaofeng Mao, Yuefeng Chen, Xiaojun Jia, Rong Zhang, Hui Xue, and Zhao Li. Context-aware robust fine-tuning. *arXiv preprint arXiv:2211.16175*, 2022.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

OpenAI. Chatgpt: An ai language model by openai. `https://openai.com`, 2020.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Omiros Pantazis, Gabriel Brostow, Kate Jones, and Oisin Mac Aodha. Svl-adapter: Self-supervised adapter for vision-language pretrained models. *arXiv preprint arXiv:2210.03794*, 2022.

Nikolaos Passalis and Anastasios Tefas. Learning deep representations with probabilistic knowledge transfer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 268–284, 2018.

Peyman Passban, Yimeng Wu, Mehdi Rezagholizadeh, and Qun Liu. Alp-kd: Attention-based layer projection for knowledge distillation. In *Proceedings of the AAAI Conference on artificial intelligence*, volume 35, pp. 13657–13665, 2021.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Manli Shu, Weili Nie, De-An Huang, Zhiding Yu, Tom Goldstein, Anima Anandkumar, and Chaowei Xiao. Test-time prompt tuning for zero-shot generalization in vision-language models. *arXiv preprint arXiv:2209.07511*, 2022.

Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela. Flava: A foundational language and vision alignment model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15638–15650, 2022.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. Large language models encode clinical knowledge. *arXiv preprint arXiv:2212.13138*, 2022.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012.

Jingchen Sun, Jiayu Qin, Zihao Lin, and Changyou Chen. Prompt tuning based adapter for vision-language model adaption. *arXiv preprint arXiv:2303.15234*, 2023.

Zifeng Wang, Zhenbang Wu, Dinesh Agarwal, and Jimeng Sun. Medclip: Contrastive learning from unpaired medical images and text. *arXiv preprint arXiv:2210.10163*, 2022.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23965–23998. PMLR, 17–23 Jul 2022a. URL `https://proceedings.mlr.press/v162/wortsman22a.html`.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7959–7971, 2022b.

LI Xuhong, Yves Grandvalet, and Franck Davoine. Explicit inductive bias for transfer learning with convolutional networks. In *International Conference on Machine Learning*, pp. 2825–2834. PMLR, 2018.

Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*, 2022.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.

Renrui Zhang, Rongyao Fang, Wei Zhang, Peng Gao, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. Tip-adapter: Training-free clip-adapter for better vision-language modeling. *arXiv preprint arXiv:2111.03930*, 2021.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

Zangwei Zheng, Mingyuan Ma, Kai Wang, Ziheng Qin, Xiangyu Yue, and Yang You. Preventing zero-shot transfer degradation in continual learning of vision-language models. *arXiv preprint arXiv:2303.06628*, 2023.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16816–16825, 2022a.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022b.

Yang Zhou, Zhanhao He, Keyu Lu, Guanhong Wang, and Gaoang Wang. Preserve pre-trained knowledge: Transfer learning with self-distillation for action recognition. *arXiv preprint arXiv:2205.00506*, 2022c.