Warsaw University of Technology

DISCIPLINE OF SCIENCE INFORMATION AND COMMUNICATION TECHNOLOGY

FIELD OF SCIENCE ENGINEERING AND TECHNOLOGY

Ph.D. Thesis

Wojciech Masarczyk, M.Sc.

Data representations in non-stationary optimization

Supervisor

prof. Tomasz Trzciński

WARSAW 2025

Data representations in non-stationary optimization

Deep learning has achieved remarkable success across various fields, including image generation, natural language processing, protein folding, and material design. Despite differences in architectures and optimization strategies, these breakthroughs share a fundamental assumption: a fixed training dataset. Currently, breaking this assumption leads to many issues, such as catastrophic forgetting or loss of plasticity. However, moving beyond the fixed-dataset paradigm would enable a continuous adaptation of these models to the ever-changing environment, an ability of any intelligent entity. To make it possible, we need to address several fundamental research questions. One of these questions explores the connection between the fundamental building blocks of deep neural networks, data representations, and the model's ability to learn in a non-stationary fashion. Specifically, this thesis comprises five publications that partially answer the following question.

What role do data representations play in non-stationary optimization?

The first part of our work aims to understand how carefully crafted modifications to data representations before training can either reduce or induce the effect of catastrophic forgetting. To achieve the former, we design a meta-learning scheme that creates a synthetic dataset to reduce catastrophic forgetting. The latter is achieved by fine-tuning data with highly discriminative features, which can dramatically deteriorate the model's performance. The last work of this group explores how the use of surrogate loss helps to build data representations that are more robust to catastrophic forgetting.

The second part of the thesis explores deep neural networks in greater detail and focuses on hidden data representations. The first work from this group introduces the Tunnel Effect Hypothesis, which states that layers of sufficiently deep networks divide into two distinct parts that contribute differently to the task. The initial layers produce linearly separable representations, while the subsequent layers, *the tunnel*, compress these representations. The tunnel is non-trivial in the model's generalization and continual learning, challenging existing perspectives on catastrophic forgetting. As a continuation, the subsequent publication analyzes representations' dynamics to understand the Tunnel Effect's origins. As an effect, the work discovers rank-deficit bias, a novel class of solutions findable by neural networks thanks to softmax function's surprising ability to increase the representations' rank. The rank-deficit bias leads to multiple consequences among them are learning more compressed data representations which generalize better in distribution and worse out-of-distribution.

Keywords: Continual Learning, Hidden Representations, Transfer Learning

Reprezentacje danych w optymalizacji niestacjonarnej

Uczenie głębokie osiągnęło znaczące sukcesy w różnych dziedzinach, takich jak generowanie obrazów, przetwarzanie języka naturalnego, przewidywanie struktur białkowych czy projektowanie materiałów. Pomimo różnic w architekturach i strategiach optymalizacji, te przełomy opierają się na wspólnym założeniu: stałym zbiorze danych treningowych. Obecnie odstąpienie od tego założenia prowadzi do wielu problemów, takich jak katastroficzne zapominanie czy utrata plastyczności. Jednak wyjście poza paradygmat stałego zbioru danych umożliwiłoby ciągłą adaptację modeli do zmieniającego się środowiska – cechę charakterystyczną każdego inteligentnego bytu. Aby to osiągnąć, należy odpowiedzieć na kilka fundamentalnych pytań badawczych. Jedno z nich dotyczy związku między podstawowymi elementami składowymi głębokich sieci neuronowych, reprezentacjami danych, a zdolnością modelu do uczenia się w warunkach niestacjonarnych. W szczególności, niniejsza praca doktorska składa się z pięciu publikacji, które częściowo odpowiadają na następujące pytanie:

Jaką rolę odgrywają reprezentacje danych w optymalizacji niestacjonarnej?

Pierwsza część naszej pracy analizuje, jak celowo wprowadzone modyfikacje reprezentacji danych przed treningiem mogą zmniejszyć lub wywołać efekt katastroficznego zapominania. Aby osiągnąć ten pierwszy cel, proponujemy schemat meta-uczenia, który tworzy syntetyczny zbiór danych zmniejszający katastroficzne zapominanie. Drugi cel osiągamy poprzez dotrenowanie na danych z dodatkiem wysoce dyskryminacyjnych cech, co może znacząco pogorszyć wydajność modelu. Ostatnia praca z tej grupy bada, jak zastosowanie zastępczej funkcji straty pomaga budować reprezentacje danych bardziej odporne na katastroficzne zapominanie.

Druga część pracy doktorskiej dogłębniej analizuje głębokie sieci neuronowe, skupiając się na ukrytych reprezentacjach danych. Pierwsza praca z tej grupy wprowadza Hipotezę Efektu Tunelu, która stwierdza, że warstwy wystarczająco głębokich sieci dzielą się na dwie odrębne części, które w różny sposób przyczyniają się do rozwiązania zadania. Początkowe warstwy generują reprezentacje liniowo separowalne, podczas gdy kolejne warstwy, *tunel*, kompresują te reprezentacje. Tunel odgrywa kluczową rolę w generalizacji modelu i uczeniu ciągłym, podważając dotychczasowe perspektywy dotyczące katastroficznego zapominania. Kontynuując ten wątek, kolejna publikacja analizuje dynamikę reprezentacji, aby zrozumieć źródła Efektu Tunelowego. W efekcie praca odkrywa efekt *deficytu rzędu* (ang. *rank-deficit bias*), nową klasę rozwiązań znajdowanych

przez sieci neuronowe dzięki zaskakującej zdolności funkcji softmax do zwiększania rzędu reprezentacji. Efekt *deficytu rzędu* prowadzi do wielu konsekwencji, w tym uczenia bardziej skompresowanych reprezentacji danych, które lepiej generalizują w obrębie dystrybucji, ale gorzej poza nią.

Słowa kluczowe: Uczenie ciągłe, ukryte reprezentacje, transfer wiedzy

Contents

1. Introduction

Deep learning has revolutionized many fields in recent years, driving unprecedented advancements and achieving remarkable milestones. Systems powered by deep learning now dominate coding competitions [1, 2, 3] and outperform human experts in complex games such as go [4, 5], chess [5, 6], and poker [7]. Beyond gaming, these methods have proven transformative in scientific research [8], enabling breakthroughs in material design [9] and protein folding [10]. What's particularly intriguing is their ability to excel in inherently less structured and predictable tasks, such as natural language understanding [11, 12], human-like conversational interactions [13, 14], and generating images [15, 16], videos [17], and music [18, 19].

Continuous innovations in neural network architectures [20, 21], training methodologies [22, 23], and optimization techniques [24, 25] have propelled the rapid progress in deep learning. These advancements have expanded what deep learning can achieve, enabling increasingly sophisticated and ambitious applications [10, 12, 1]. However, despite these impressive achievements, all deep learning systems share a fundamental dependency: the need for carefully prepared training datasets. These datasets are labor-intensive to create and come with strict requirements imposed by the nature of deep learning algorithms. One of the core assumptions underlying these systems is the static nature of the training data. The dataset must be fully defined before training begins, and the data samples are expected to be independently and identically distributed (i.i.d.) [26, 27]. The same assumption applies to test data, meaning that a model's performance is only reliable within the narrow scope defined by its training dataset. This limitation underscores a critical vulnerability of deep learning: its inability to adapt to new or evolving situations or to generalize beyond the specific distribution of the training data [28, 29].

This inherent constraint contrasts human learning, where individuals can effortlessly repurpose existing knowledge [30] or continuously refine it based on new information [31]. For deep neural networks, however, attempting to adapt to new data often leads to a phenomenon known as *catastrophic forgetting* [32]. In this scenario, the model improves its performance on the new data but loses its ability to perform well on previously learned tasks [33]. This trade-off highlights

a significant challenge in the field. Without addressing this limitation, deep learning models will remain powerful yet inherently constrained tools, limited to narrow domains defined by their training datasets. Overcoming this challenge is widely regarded as one of the key milestones in achieving artificial general intelligence (AGI) [34], where systems can learn, adapt, and generalize in ways that more closely resemble human cognition. Addressing this issue is crucial for advancing the field and unlocking the full potential of deep learning in real-world, dynamic environments.

To achieve these ambitious goals, continual learning [35] emerged as a research community focused on designing deep learning methods tailored to the needs of dynamic environments where models seamlessly adapt to ever-changing data without losing previously acquired knowledge. Despite the efforts and numerous ideas [36, 37] or workarounds [38, 39] proposed and explored by this community, the progress in overcoming catastrophic forgetting and enabling deep learning methods to learn continually has been moderate [40].

1.1. Research questions

This thesis is motivated by the urgent need to develop deep learning methods capable of operating in dynamic environments. Rather than proposing new methods, it takes a step back to investigate the root causes of catastrophic forgetting and the limitations of models' generalization ability. At the core of this exploration is the idea that data representations serve as the foundational building blocks of deep learning models. By examining these representations, we aim to deepen our understanding of why models are vulnerable to catastrophic forgetting and struggle to generalize beyond their training domains. The central thesis of this work is that *data representations play a critical role in determining a model's performance in non-stationary optimization scenarios*.

To explore this hypothesis, we address the following research questions:

Research questions (RQ)

1. How do different data representations influence catastrophic forgetting? Can we learn data representations that inherently make models more resistant to catastrophic forgetting?
2. Are all data representations equally susceptible to catastrophic forgetting?
3. How do training dynamics affect the learned representations and their ability to generalize outside the training distribution?

The remainder of this thesis is organized into chapters, each representing a

distinct publication. The initial works [41, 42, 43] focus on the first question (RQ1). More specifically, works [41, 42] demonstrate that it is indeed possible to learn data representations that significantly reduce a model's vulnerability to catastrophic forgetting. Specifically, work [41] employs meta-learning to generate synthetic data samples for use in continual learning scenarios, while work [42] introduces a surrogate reconstruction loss that encourages the learning of task-invariant representations, thereby mitigating catastrophic forgetting. Additionally, work [43] investigates how even minimal manipulations of data representations can trigger catastrophic forgetting, even in models that achieve perfect accuracy on the same dataset.

The second part of the thesis addresses the second (RQ2) and third questions (RQ3). Works [44, 45] introduce and explore the origins of the Tunnel Effect. The Tunnel Effect Hypothesis posits that neural networks implicitly divide into two distinct components, each contributing differently to the model's overall performance. Importantly, this division aligns with the model's ability to generalize beyond its training distribution and its susceptibility to catastrophic forgetting. These studies provide new insights into the relationship between data representations, generalization, and catastrophic forgetting, offering a pathway toward more robust and adaptable deep learning systems.

1.2. Thesis contributions

1.2.1. Reducing catastrophic forgetting with learning on synthetic data

In the first work [41], presented in Chapter **??**, we begin our exploration of how data representations influence catastrophic forgetting by addressing a key question: *Can data representations be modified in advance (a priori) so that a simple fine-tuning strategy does not suffer from catastrophic forgetting as severely as when trained on standard data representations*? To answer this question, we proposed a generative model that uses meta-learning to create a sequence of synthetic data samples. When trained on these synthetic samples, a model performs similarly on a held-out dataset while exhibiting significantly reduced susceptibility to catastrophic forgetting.

Interestingly, as illustrated in Figure 1.2.1, the synthetic samples generated by the model (top row) do not visually resemble the original data samples (bottom row). Despite this, training on these synthetic representations yields performance on the held-out dataset comparable to training on the original data. Furthermore, as shown in the right plot of Figure 1.2.1, training on a sequence of these synthetic representations results in a final model that experiences much less catastrophic forgetting than the models trained on standard representations.

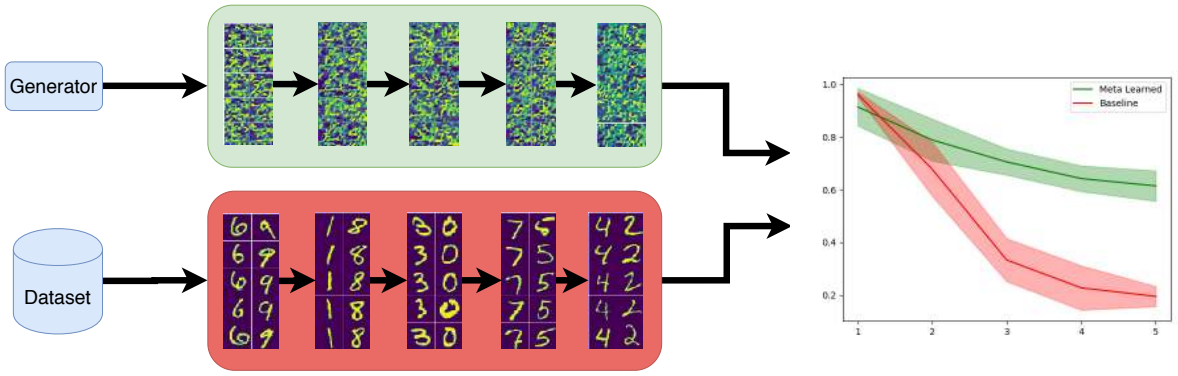


Figure 1.2.1. We propose a method to meta-learn a generative network capable of generating synthetic data representations that are learned in a sequence, resulting in reduced catastrophic forgetting compared to the sequence of tasks composed of original data samples.

This finding provides a clear affirmative answer to our initial question, demonstrating that it is indeed possible to meta-learn data representations that mitigate catastrophic forgetting. However, the practical application of this method is limited due to two significant challenges. First, the meta-learning process is computationally expensive, leading to high computational costs. Second, the

method requires access to the held-out test dataset during the meta-learning phase, which usually violates the constraints imposed by continual learning scenarios.

As the primary author of this work, the PhD candidate was responsible for identifying the motivation behind the experiment, designing and implementing the method, and conducting all experiments and ablation studies included in the paper. The second author served as a mentor, providing guidance throughout the research process and assisting with writing and structuring the manuscript.

1.2.2. On robustness of generative representations against catastrophic forgetting

The second work [42], presented in Chapter **??**, builds on the findings of the first publication [41] and further investigates the first research question while addressing the limitations of the previously proposed method. Having demonstrated that data representations can be manipulated to reduce catastrophic forgetting, this work aims to develop a more computationally feasible and scalable approach for creating representations that are inherently robust to catastrophic forgetting, particularly for larger datasets.

To achieve this, we hypothesize that representations learned by discriminative models exhibit minimal, if any, similarities across different tasks. In contrast, we posit that representations learned by generative models, especially for natural images, are likely to share significant similarities between tasks. This leads to the central hypothesis of this work: *Representations learned by generative models are less prone to catastrophic forgetting than those learned by discriminative models.*

To test this hypothesis, we design experiments comparing representations learned by discriminative models with those learned by generative models, specifically Autoencoders (AEs) and Variational Autoencoders (VAEs), in a continual learning scenario. In both cases, we evaluate the utility of the learned representations for classification tasks. The results, illustrated in Figure 1.2.2, confirm our hypothesis, showing that representations learned by AEs and VAEs are indeed more robust to catastrophic forgetting when trained on a sequence of five tasks. These findings are consistent across multiple datasets, including MNIST, FashionMNIST, and CIFAR-10.

Further analysis of representation similarities reveals that generative models produce significantly more consistent representations than those learned by discriminative models, which tend to change drastically between consecutive tasks. Additionally, the computational cost of using a surrogate loss is minimal,

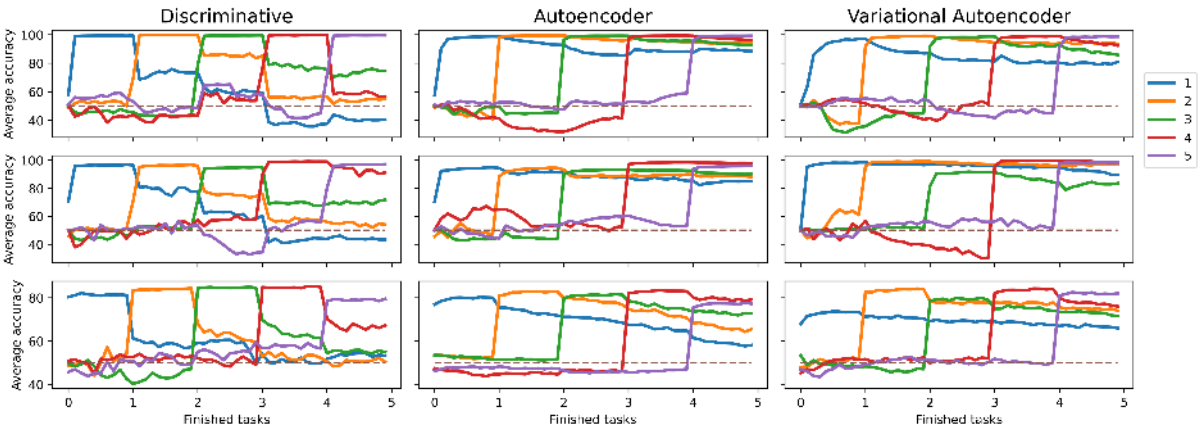


Figure 1.2.2. The results demonstrate that representations learned by generative models are significantly more robust to catastrophic forgetting. We hypothesize and confirm through additional experiments that this robustness stems from generative models learning more versatile and transferable representations across tasks.

as it primarily involves training an additional classifier head on top of the learned representations—a negligible cost compared to the overall training process. This approach also opens the possibility of combining discriminative and generative losses during training, where the generative loss could implicitly regularize the model's representations, making them more robust to catastrophic forgetting.

As the primary author of this work, the PhD candidate formulated the main hypothesis, designed and implemented the method, and conducted all experiments and ablation studies included in the paper. The second and third authors were mentors, guiding the research process and assisting with writing and structuring the manuscript.

1.2.3. On consequences of finetuning on data with highly discriminative features

The next work [43], presented in Chapter **??**, explores an orthogonal perspective from the previous studies. Rather than developing methods to enhance robustness to catastrophic forgetting, it challenges existing assumptions about its causes through a thought-provoking experiment. While prior research suggests a non-monotonic relationship between data distribution similarities and catastrophic forgetting, with the strongest forgetting occurring for similar but distinct distributions [46], this work offers a new perspective.

The study constructs a sequence of two tasks using the same dataset, differing only in the color value of a single pixel. In the first task, the pixel's color is random, while it correlates with the image's class in the second. The

network is first trained on the initial task until it achieves perfect training accuracy, then fine-tuned on the nearly identical second task for the same number of epochs.

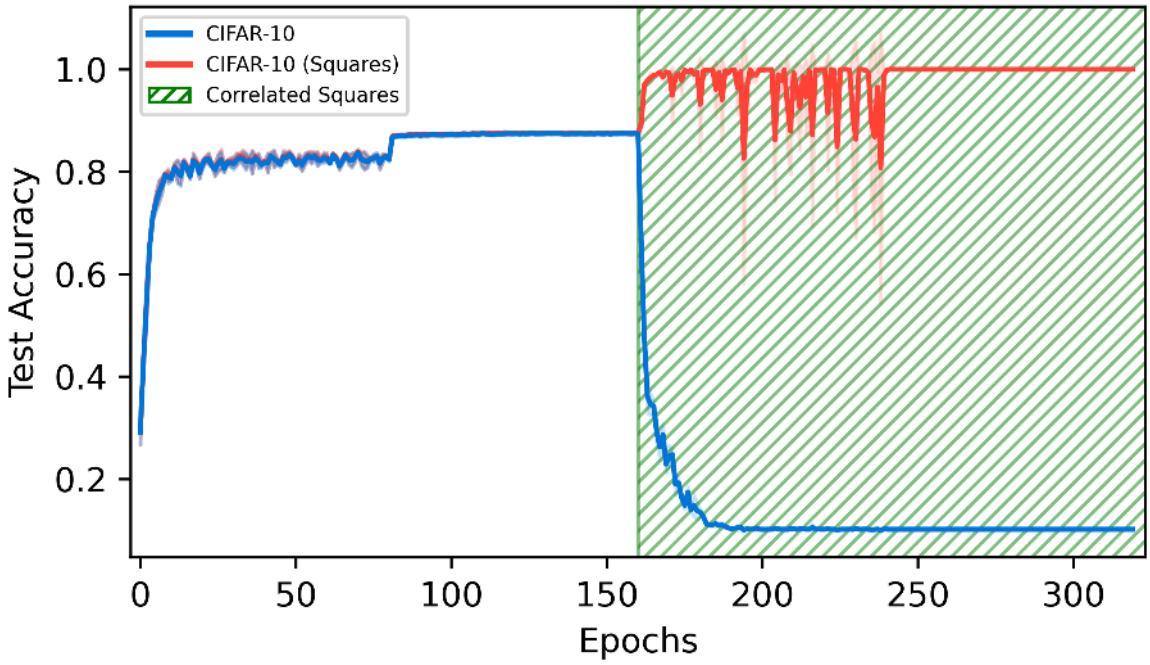


Figure 1.2.3. Introducing a simpler discriminative feature (green hatch area) solely focuses the model on that feature (red curve) and abruptly erases previous knowledge about the meaningful data representations(blue line).

As shown in Figure 1.2.3, the test accuracies on both datasets evolve unexpectedly during training. While the first phase shows aligned accuracies for both tasks, training on the second task leads to severe catastrophic forgetting on the first task, alongside a significant performance boost on the second. This reveals two key insights: (1) catastrophic forgetting is not tied to input data similarities (semantic or structural) but rather to differences in input-output correlations between tasks, and (2) even models with perfect training accuracy exhibit simplicity bias, favoring the most straightforward rules to achieve their objectives. This finding is particularly relevant in fine-tuning foundational models, where spurious correlations or small datasets can trigger similar effects.

The PhD candidate formulated the hypothesis, implemented experiments, and collected initial results. Part of the results, as well as ablation experiments, were designed by the third author. All authors were involved in writing the manuscript.

1.2.4. The Tunnel Effect: Building data representations in deep neural networks

Typical studies in continual learning have often treated neural networks as monolithic entities that learn or forget uniformly. While this perspective is not incorrect, it overlooks the nuanced influences of architecture, training dynamics, and other subtle factors that significantly impact model performance. To address this gap, the next two works provide a more detailed perspective. They demonstrate that hidden data representations—those transformed by intermediate layers within neural networks—evolve not uniformly during training and play a critical role in a model's ability to learn continually and generalize beyond its training distribution. These works collectively address the second and third research questions of this thesis.

Specifically, work [44], presented in Chapter **??**, introduces the *Tunnel Effect Hypothesis*, which posits that layers in sufficiently deep networks divide into two distinct parts, each contributing differently to the task. As illustrated in Figure 1.2.4, the initial layers produce linearly separable representations that enhance model performance, while the subsequent layers—referred to as *the tunnel*—compress these representations, as measured by their rank, while maintaining similar performance levels. This division between the *extractor* and the *tunnel* directly correlates with the network's generalization capabilities and its performance in continual learning. The tunnel layers compress representations losslessly for data from the training distribution, but this compression reduces the model's ability to generalize to out-of-distribution data. Additionally, continual learning experiments reveal that the tunnel acts as a task-agnostic compressor under certain assumptions, remaining stable during fine-tuning on new tasks. At the same time, catastrophic forgetting occurs exclusively in the extractor part of the network.

Although the experiments in work [44] highlight the tunnel effect's importance in continual learning and out-of-distribution generalization, the work lacks a formal understanding of why and when it emerges or how it can be induced.

The PhD candidate formulated the main hypothesis and implemented most of the codebase required for the experiments. The second author contributed to the experimental work and writing process, while other co-authors provided mentorship and assisted with writing and structuring the manuscript.

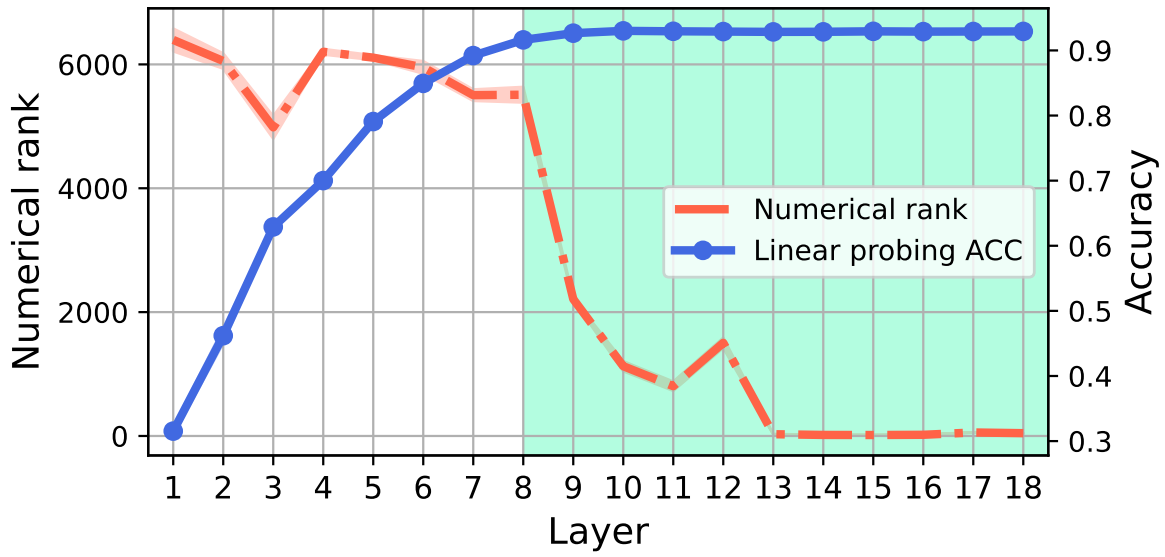


Figure 1.2.4. The tunnel effect in VGG19 trained on CIFAR-10. In the tunnel (shaded area), the performance of linear probes attached to each layer saturates (blue line), while the representations are compressed, as indicated by the steep decline in representation rank (red dashed line).

1.2.5. Unpacking Softmax: How Temperature Drives Rank Collapse, Compression and Generalization

Building on the significant role of the tunnel effect [44] in generalization and continual learning, the publication [45] takes a theoretical approach to investigate the dynamics of representations. This work aims to uncover the origins of the tunnel effect and examine how factors such as architectural design choices influence its emergence and intensity.

The analysis reveals that the softmax function, commonly used to map network outputs to probabilities, can increase the rank of a matrix. As presented in Figure 1.2.5, this characteristic leads to a phenomenon termed *rank deficit bias*, where linear neural networks converge to solutions with a rank substantially lower than the number of classes in the task. Despite this, the softmax allows these solutions to achieve perfect accuracy on the training distribution. However, further analysis aligns with findings from [44].

Additional experiments demonstrate that training dynamics in nonlinear networks are heavily influenced by the softmax input (i.e., the norm of the network's logits). While various architectural choices impact this value, the softmax temperature is a hyperparameter directly controlling it—a significant deficit in logits norm forces the network to increase it to minimize the loss. The analysis reveals that neural networks align singular vectors between consecutive

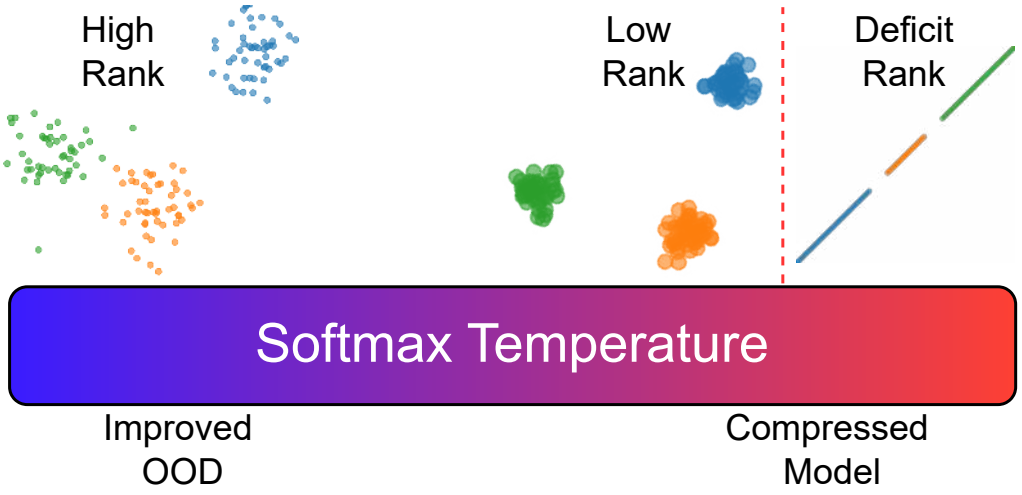


Figure 1.2.5. The softmax function and temperature play a fundamental role in shaping network representations, leading to solutions that generalize well with high rank for small temperatures or compressed solutions with *rank deficit* at the other extreme.

layers to amplify this norm, leading to collapsed representation ranks and diminished out-of-distribution performance.

The work demonstrates that the tunnel effect is not universally present and is influenced by architectural choices such as network depth, width, initialization type, or normalization. These factors affect the logits norm, determining whether the tunnel effect forms. In conjunction with the work [44], the results establish a link between popular architectural choices and the model's ability to generalize outside of training distribution and its robustness to catastrophic forgetting.

The PhD candidate designed and implemented all the experiments in this work and proved theoretical results. Second and third co-authors helped with results analysis and interpretation of the results. All co-authors provided guidance and assisted with writing and structuring the manuscript. The last author suggested to use softmax temperature to steer models evolution.

1.3. Publications not included in the thesis

The following is a list of publications that I have co-authored but are not included in the thesis.

1. **Masarczyk Wojciech**, Paweł Wawrzyński, Daniel Marczak, Kamil Deja and Tomasz Trzciński. "Logarithmic continual learning.", IEEE Access 10, (2022).
2. Ostaszewski Mateusz, Trenkwalder Lea, **Masarczyk Wojciech**, Scerri Eleonor, Vedran Dunjko. "Reinforcement learning for optimization of variational quantum circuit architectures", Advances in Neural Information Processing Systems 34 (NeurIPS 2021), (2021)
3. Cheng Tin Sum, Zhao Jim, **Masarczyk Wojciech,** Lucchi Aurelien, " From Partial to Full Neural Collapse: Effects of Implicit Biases in Deep Neural Networks ", Submitted to ICML, (2025)
4. Grabowski Bartosz, **Masarczyk Wojciech**, Głomb Przemysław, Mendys Agata, "Automatic pigment identification from hyperspectral data", Journal of Cultural Heritage 31, (2018)
5. **Masarczyk Wojciech**, Głomb Przemysław, Grabowski Bartosz, Ostaszewski Mateusz, "Effective Training of Deep Convolutional Neural Networks for Hyperspectral Image Classification through Artificial Labeling ", Remote Sensing 12, (2020)
6. Deja Kamil, Wawrzyński Paweł, Marczak Daniel, **Masarczyk Wojciech**, Trzciński Tomasz, " Binplay: A binary latent autoencoder for generative replay continual learning", 2021 International Joint Conference on Neural Networks (IJCNN), (2021)
7. Deja Kamil, Wawrzyński Paweł, **Masarczyk Wojciech**, Marczak Daniel, Trzciński Tomasz, "Multiband VAE: latent space alignment for knowledge consolidation in continual learning", Vienna: International Joint Conferences on Artificial Intelligence Organization, (2022)

2. Reducing catastrophic forgetting with learning on synthetic data

Abstract

Catastrophic forgetting is a problem caused by neural networks' inability to learn data in sequence. After learning two tasks in sequence, performance on the first one drops significantly. This is a serious disadvantage that prevents many deep learning applications to real-life problems where not all object classes are known beforehand; or change in data requires adjustments to the model. To reduce this problem we investigate the use of synthetic data, namely we answer a question: Is it possible to generate such data synthetically which learned in sequence does not result in catastrophic forgetting? We propose a method to generate such data in two-step optimisation process via meta-gradients. Our experimental results on Split-MNIST dataset show that training a model on such synthetic data in sequence does not result in catastrophic forgetting. We also show that our method of generating data is robust to different learning scenarios.

2.1. Introduction

Deep learning methods have succeeded in many different domains such as: scene understanding, image generation, natural language processing [47, 48, 49, 50]. While deep learning methods differ in architecture choice, objective function or optimization strategy, they all assume that the training data is independent and identically distributed (i.i.d). Methods built on this assumption are effective for fixed environments with stationary data distributions – where tasks to be solved do not change over time or classes present in the dataset are known from the beginning. However, in most real-life scenarios this assumption is violated and there is a need for methods that are able to handle such cases. Among many examples of such scenarios, a few can be highlighted: new object class is introduced, however the dataset used to train the baseline model is no

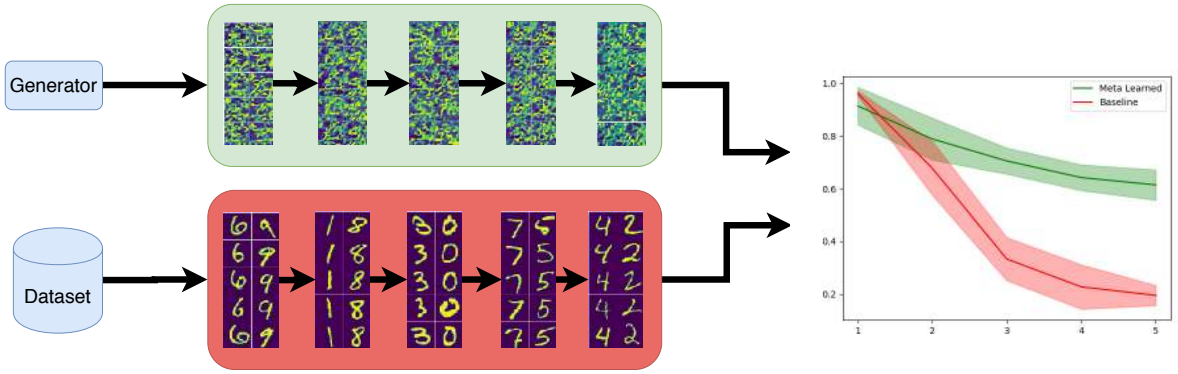


Figure 2.1.1. Synthetic data created from generator is divided into five tasks according to classes and learner (green) learns tasks sequentially. The same procedure is applied to learner with real data (red). The right plot shows that accuracy at the end of each task does not decrease on learned data in contrast to real data where it deteriorates sharply.

longer available; the data characteristics seem to change seasonally and model needs to change its predictions accordingly to these trends. Continual learning [51] is a paradigm where data is presented sequentially to the algorithm without the ability to manipulate this sequence. Additionally, there is no assumption about the structure of the sequence. A successful continual learning algorithm needs to be able to learn a growing number of tasks, be resistant to catastrophic forgetting [32] and be able to adapt do distribution shifts. The memory and computational requirements of such algorithm should scale reasonably with the incoming data.

Although the problem of continual learning is known for many years [51, 32], only recently has the field gained significant traction and many interesting ideas have been proposed. Most of continual learning contributions can be divided into three categories [52, 35]: optimization, architecture and rehersal. Methods based on optimization modifications usually add additional regularization terms to objective function to dampen catastrophic forgetting [36, 53]. Second category gathers methods that propose various architectural modifications e.g. Progressive Net [54] where increasing capacity is obtained by initialising new network for each task. The last category – rehersal based methods – consists of methods that assume life-long presence of a subset of historical data that can be re-used to retain knowledge about past tasks [55, 56].

This work proposes a new data-driven path that is orthogonal to existing approaches. Specifically, we would like to explore the possibility of creating input data artificially in a coordinated manner in such a way that it reduces the catastrophic forgetting phenomena. We achieve this by combining two separate neural networks connected by two-step optimisation. We use generative model

to create synthetic dataset and form a sequence of tasks to evaluate learner model in continual learning scenario. The sequence of synthetic tasks is used to train the learner network. Then, the learner network is evaluated on real data. The loss obtained on real data is used to tune the parameters of the generative network. In the following step, the learning network is replaced with a new one.

Differently from existing approaches, our method is independent of training method and task and it can be easily incorporated to above-mentioned strategies providing additional gains.

2.2. Related Work

One line of research for continual learning focuses on optimization process. It draws inspiration from the biological phenomena known as synaptic plasticity [57]. It assumes that weights (connections) that are important for particular task become less plastic in order to retain the desired performance on previous tasks. An example of such approach is Elastic Weight Consolidation (EWC) [36], where regularisation term based on Fisher Information matrix is used to slow down the change of important weights. However accumulation of these constrains prevents network from learning longer sequences of tasks. Another optimization based method is Learning without Forgetting (LwF) [53]. It tries to retain the knowledge of previous tasks by optimizing linear combination of current task loss and knowledge distillation loss. LwF is conceptually simple method that benefits from knowledge distillation phenomenon [58]. The downside of such approach is that applying LwF requires additional memory and computation resources for each optimization step.

Methods based on architectural modifications allow to dynamically expand/shrink networks, select sub-networks, freeze weights or create additional networks to preserve knowledge. Authors of [54] propose algorithm that for each new task creates a separate network (a column) that is trained to solve particular task. Additionally, connections between previous columns and the current column are learned to enable forward transfer of knowledge. This algorithm avoids catastrophic forgetting completely and enables effective transfer learning. However the computational cost of this approach is prohibitive for longer sequences of tasks. Other methods [59, 60] address the problem of computational cost by expanding single layers/neurons instead of whole networks, however these methods has less capacity to solve upcoming tasks. Different approaches that modify architectures are based on selecting sub-networks

used for solving current task in such a way that only a fraction of network's parameters relevant to current task is changed [38, 61, 62]. The challenge here is to balance the number of frozen and active weights in such way that network is still able to learn new tasks and preserve current knowledge.

Rehearsal methods are based on the concept of memory replay. It is assumed that subset of previously processed data is stored in memory bank and interleaved with upcoming data in such a way that neural network learns to solve current task in addition to preserving current knowledge [55, 63, 56]. A naive rehearsal method would be to save random data samples that were present during training. However such approach is inefficient, since samples are not equally informative, hence the challenge of rehearsal methods is to choose the most representative samples for a given dataset, such that minimum storage is occupied. In [63], authors apply method of dataset distillation based on meta-gradient optimization to reduce the size of memory bank. It is possible to represent whole class of examples just by storing one carefully-optimized example. Unfortunately, applying this meta-optimization method is computationally exhaustive. The biggest downside of using rehearsal based methods is the need to store the actual data which in some cases can violate data privacy rules or can be computationally prohibitive. To mitigate this issue solution based on Generative Networks was proposed [64, 39]. Namely, they use dual model architecture composed of learner network and generative network. Role of the generative network is to model data previously experienced by the learner network. Data sampled from the generator network is used as a rehersal data for learner network to reduce the effect of catastrophic forgetting.

Our method is also dual architecture model based on generative network, however the aim of generative network is radically different. In contrast to authors [64, 39] we do not aim to capture the statistics of real data, instead we try to generate entirely synthetic data such that when learner does learn on a sequence of such data it does not suffer from catastrophic forgetting.

2.3. Method

The main idea of our approach is to generate data samples such that network trained on them in sequence would not suffer from catastrophic forgetting. One of many ways to generate artificial data is to use meta-optimization strategy introduced in [65]. It is shown that by applying meta-learning it is possible to use gradient optimization both to hyperparameters and to input data. However, this approach is limited to small problems, since each data point must be opti-

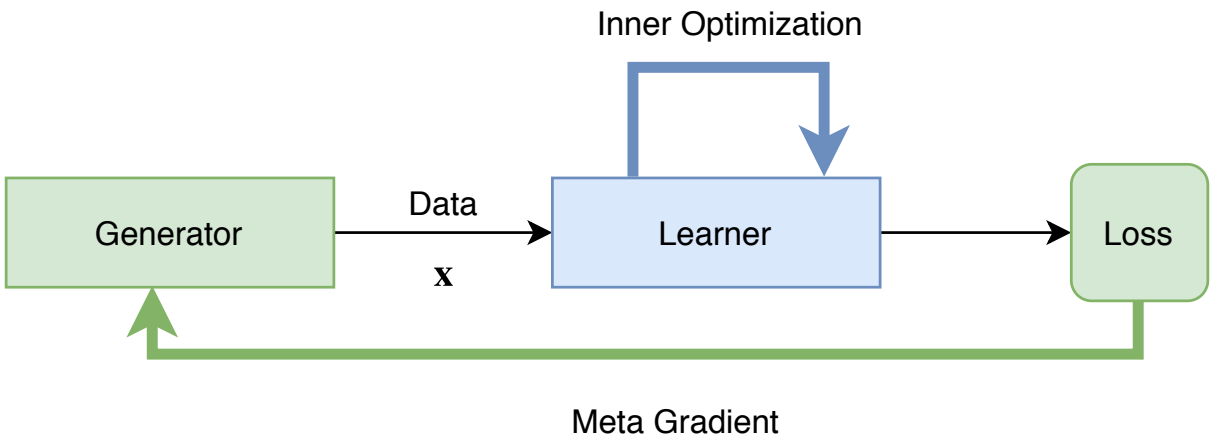


Figure 2.3.1. Synthetic data from generator is passed to learner where the inner optimization is performed and meta-loss is backpropagated to $\mathcal{G}$.

mised separately. To overcome this bottleneck, authors of Generative Teaching Networks (GTNs) [66] use generative network to create artificial data samples instead of directly optimizing the data input. We adopt similar approach in our method, namely, we use generative network – green rectangle "Generator" in Fig. 2.3.1 – to produce synthetic data from noise vectors sampled from a random distribution. Next, we split the data into separate tasks according to classes and form a continual learning task for the learner network – blue rectangle in Fig. 2.3.1. Learner network after completing whole sequence of tasks in evaluated on real training data. The loss from real data classification after learning all tasks in sequence is then backpropagated to generator network to tune the parameters as shown in Fig. 2.3.1.

Our approach is similar to one proposed in work [67]. Using two step meta-learning optimization they try to learn best representation of input data such that the model learned in with standard optimization does not suffer from catastrophic forgetting.

Differently from [66], we do not use curriculum based learning as our goal is to have a realistic continual learning scenario where the order of data sequence is not known beforehand. To ensure that the Generator network does not generate data suitable for particular sequence of tasks at each meta-optimization we shuffle order of tasks. Precisely, at each step we generate p samples for each class and then randomly create a sequence of binary classification tasks with particular data.

Precisely, let $\mathcal{G}$ be a generative neural network, $\mathcal{S}$ a standard convolutional network for classification, $\mathbf{t} = (t_1, t_2, \ldots t_n)$ a sequence of tasks, where each tasks

is binary classification task and classes in each task form mutually disjoint sets.

The inner training loop consists of sequence of tasks, where generated samples from previous tasks are not replayed once the task is finished. To achieve this, the sequence of tasks $\mathbf{t} = (t_1, t_2, \dots t_n)$ must be defined a priori and samples generated by network $\mathcal{G}$ are conditioned on the information of particular task. For each task t_i the network $\mathcal{G}$ generates two batches of samples $\mathbf{x} = \mathcal{G}(\mathbf{z}, \mathbf{y_{i_j}})$ for $j = 1, 2$, where $\mathbf{z}$ is a batch of noise vectors sampled from Normal distribution and $\mathbf{y_{i_j}}$ is a class indicator for task $\mathbf{t_i}$. Note that generator networks has access to class indicators since we aim to learn in continual learning scenario only the learner network.

Neural network $\mathcal{S}$ learns sequentially on following tasks using standard SGD optimizer with learning rate and momentum optimized through meta-gradients. At the end of the sequence $\mathbf{t}$ network $\mathcal{S}$ is evaluated on real dataset $(\mathbf{x_r}, \mathbf{y_r})$ obtaining meta-loss as shown in Fig. 2.3.1. This meta-loss is backpropagated through all training inner-loops of model $\mathcal{S}$ to optimize network $\mathcal{G}$. Parameters θ of network $\mathcal{G}$ are updated according to the equation:

$$\theta = \theta - \eta \nabla_\theta \mathcal{L}(\mathcal{S}(\mathbf{x_r}; \mathbf{w_m}), \mathbf{y_r}), \tag{2.1}$$

where $\mathbf{w_m}$ are parameters of the network $\mathcal{S}$ after m optimization steps, η is fixed learning rate, $\mathcal{L}$ is a cross entropy loss function, $\mathbf{x_r}, \mathbf{y_r}$ are real data samples and labels respectively.

2.4. Experiments

To test our hypothesis we use popular continual learning benchmark Split-MNIST [68, 69]. In first experiment, we use 5-fold split with two classes for each task to create a moderately difficult sequence of tasks. Network $\mathcal{G}$ generates 250 samples per each class. During inner optimisation learner network is optimized on batch size formed with 40 generated images (20 samples per class drawn randomly from the pool of 250 samples per class). We train the learner network on each task for 5 inner steps with batch size 40. Once the task is over, samples from this task are not shown to the network to the end of training. At test time, after learning on each task the network is evaluated on part of a test set composed of classes seen in previous taks. Both networks are simple convolutional neural networks with two convolutional layers with addition of

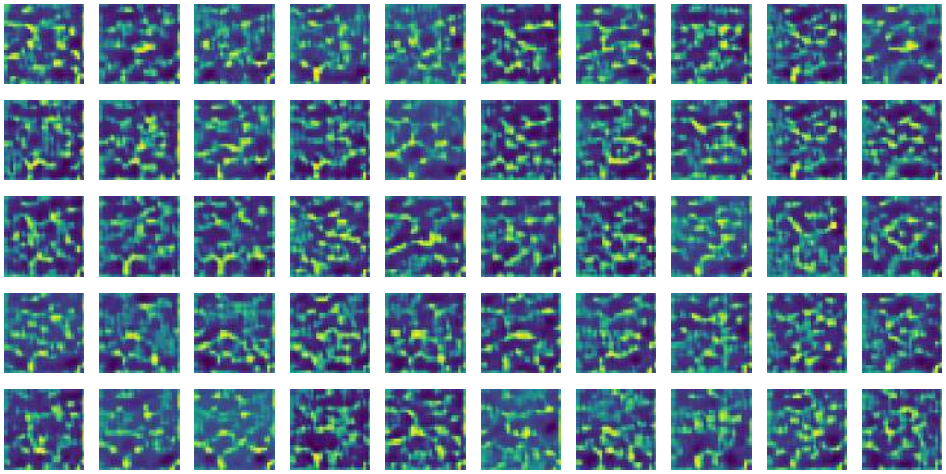


Figure 2.4.1. Samples generated by network $\mathcal{G}$ at the end of meta-optimisation. Starting from zero (leftmost), each sample to the right represents the following class.

one and two fully connected layers for classification and generative network respectively. Each layer is followed by a batch normalisation layer.

As a baseline to compare with, we use simple fully connected network proposed in [70] ('MLP' – red – in Fig. 2.4.2). To further investigate the impact of generated data we use the same network architectures and optimizer settings with learning rate and momentum optimized with by a meta learning process as described in Section 2.3 but for optimizing the learner network we use real data ('Real Data' – yellow – in Fig. 2.4.2). We also compare our results with GAN-based data samples. In this scenario we follow the setting of 'Real Data' scenario except for the source of data. We use Conditional-GAN [71] to model the original data distribution and then sample 250 samples per each class ('GAN based' – blue – in Fig. 2.4.2).

We implement experiments in PyTorch library, which is well suited for computing higher-order gradients [72].

Results – obtained results support our hypothesis, that it is possible to generate synthetic data such that, even if networks learns this data in sequence (one time per sample), the learning process does not result in castastrophic forgetting.

Figure 2.4.2 shows how learning on synthetic data in sequence results in less catastrophic forgetting compared to learning on a sequence of real data samples. Note that additional performance could be gained with careful hyperparameter tuning, however we did not want to compete for best performance and rather

show the potential of this approach. Higher accuracy of 'Real data' scenario over 'MLP' can be attributed to the effectiveness of optimised learning rate and momentum parameters, however the main advantage comes from using meta learned data samples. Results obtained with data generated with GAN are almost identical to ones obtained with real data. This result is expected as the data modeled by a GAN resembles original data closely.

An example batch of generated samples is shown in Figure 2.4.1. The samples are ordered according to classes (starting from 0). In contrast to [66] the data samples are abstract blobs, rather than interpretable images. We verify experimentally that the reason for the lack of structure in generated samples is the lack of curriculum learning in our scenario. We skip it intentionally to provide more realistic continual learning scenario for the learner network.

Fig. 2.4.3 shows the impact of change of learning scenario of network $\mathcal{S}$ *after* network $\mathcal{G}$ is trained. In this experiment data generated by a network $\mathcal{G}$ in first experiment is used. Here, we investigate how the final accuracy after learning five consecutive tasks changes with the number of inner optimization steps. Note that $\mathcal{G}$ was optimised to create samples that are robust to catastrophic forgetting with inner optimization loop of 5 steps. As we can see, in case of longer learning horizon, network learned on synthetic (green plot Fig. 2.4.3) data suffers significantly less than the same network learned on real data (yellow plot Fig. 2.4.3). Even though accuracy of the networks drops with

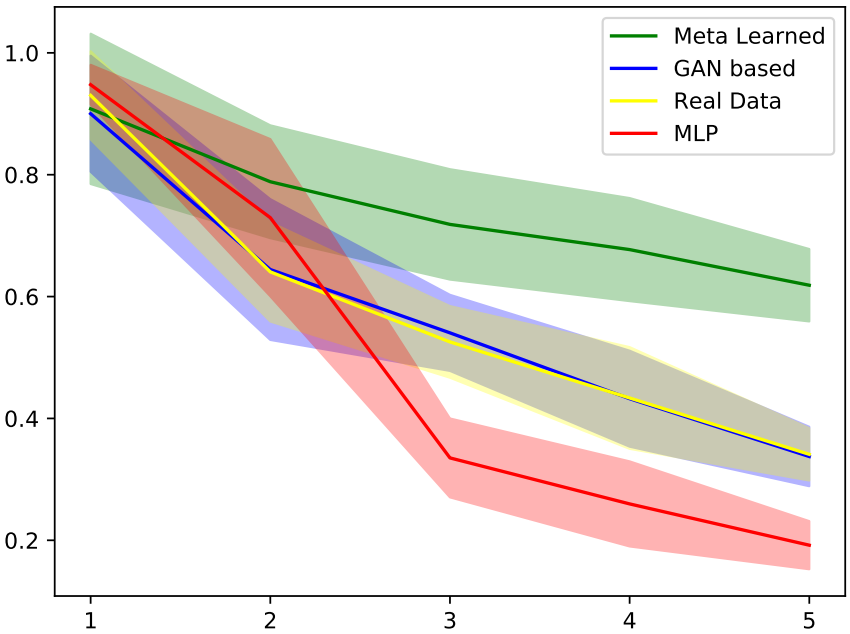


Figure 2.4.2. Overall accuracy measured on test data subset. After learning each task, test data subset is made of samples only from classes seen during recent and previous tasks.

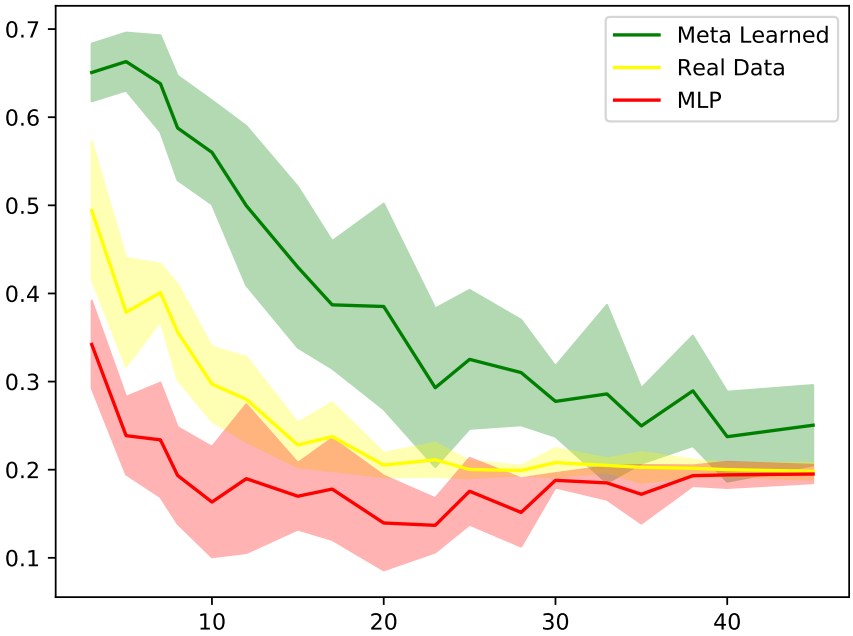


Figure 2.4.3. Overall accuracy measured on test set after learning network $\mathcal{S}$ with synthetic data for x inner steps on each task.

increasing number of inner steps, the drop is smoother in case of synthetic data.

2.5. Conclusions

The aim of this work was to answer a question, whether it is possible to create data that would dampen the effect of catastrophic forgetting. Experiments show that this hypothesis is true – it is possible to generate such samples, however usually they do not visually resemble real data. Surprisingly, even applying the method alone can result in high performing network. Additional interesting advantage of this synthetic data is the robustness to changes of inner optimisation parameters – increasing 15-fold size of a batch and length on training still results in compelling performance. We believe that our experiments open a new and exciting path in continual learning research. As a future work we plan to adjust current method to datasets of higher complexity and test its effectiveness in online learning scenario.

3. On robustness of generative representations against catastrophic forgetting

Catastrophic forgetting of previously learned knowledge while learning new tasks is a widely observed limitation of contemporary neural networks. Although many continual learning methods are proposed to mitigate this drawback, the main question remains unanswered: what is the root cause of catastrophic forgetting? In this work, we aim at answering this question by posing and validating a set of research hypotheses related to the specificity of representations built internally by neural models. More specifically, we design a set of empirical evaluations that compare the robustness of representations in discriminative and generative models against catastrophic forgetting. We observe that representations learned by discriminative models are more prone to catastrophic forgetting than their generative counterparts, which sheds new light on the advantages of developing generative models for continual learning. Finally, our work opens new research pathways and possibilities to adopt generative models in continual learning beyond mere replay mechanisms.

3.1. Introduction

Neural networks are widely used across many real-life applications, ranging from image recognition [73] to natural language processing [74]. Nevertheless, neural models used in those applications assume identical and independently distributed training data - the assumption rarely met in practice. As a result, contemporary neural network models are prone to catastrophic forgetting [33] - a well-known limitation of neural networks that results in the erosion of previously learned knowledge. Continual learning is a field of machine learning that aims at addressing this pitfall of neural models by constant adaption to new data. The majority of works in this field focus on developing methods for mitigating the effects of catastrophic forgetting [75]. These methods can

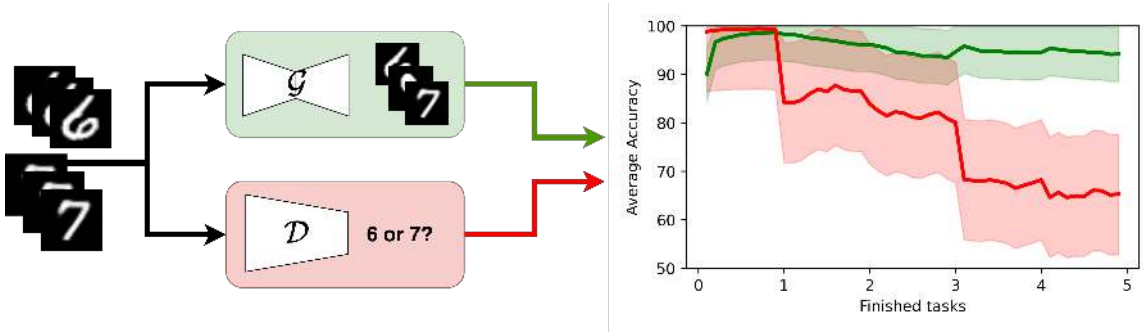


Figure 3.1.1. A schematic overview of our main experiment. Representations learned by the generative model are less susceptible to catastrophic forgetting than discriminative ones. Results of the main experiment and details can be found at Sec. 3.3.1.

be grouped into three categories – regularization based [37, 76], rehearsal methods [77, 78, 79, 80, 81, 82] and methods using dynamic architectures [83, 84, 85, 86, 87, 88, 89]. Nevertheless, recent findings [40, 90] show that it is possible to surpass well-established continual learning methods across popular benchmarks with heuristic-based baselines. The surprising effectiveness of these baselines indicates that the root cause of catastrophic forgetting is yet to be discovered, and we follow this intuition in our work.

We investigate the catastrophic forgetting with methodological rigor; we state and empirically validate research hypotheses that shed new light on this phenomenon. We build upon the works of [91, 92] where the roots of forgetting are analyzed and the findings of [90, 93] that analyze the effectiveness of intuitive solutions to this problem. Here, contrary to previous works, we postulate to look at the continual learning from the perspective of internal representations of neural networks and analyze its impact on the final performance of the continually learned model. Inspired by [90], we argue that the dynamics of catastrophic forgetting depends on the task at hand, yet contrary to this work, we analyze this observation from the perspective of internal neural representations rather than peculiarities of continual learning tasks.

To summarize, the main contribution of our work is the statement of the following research hypotheses, along with their empirical validation:

Hypothesis 1 – *Representations learned by autoencoders and variational autoencoders are less prone to catastrophic forgetting than representations of discriminative models.*

Hypothesis 2 – *Autoencoders and variational autoencoders learn more transferable features than discriminative models.*

Moreover, the results from our experiments provide an explanation for the effect observed in [90], where the authors argue that continual reconstruction tasks do not suffer from catastrophic forgetting. Our experiments show that

this observation can be explained from the perspective of generative representations [1]. This suggests that the lack of catastrophic forgetting is not exclusively linked to the continual reconstruction task.

Last but not least, our experiments show that it is possible to achieve the performance of over 90% average accuracy on popular continual learning benchmarks without any specific mechanism to overcome catastrophic forgetting. This raises the question of whether these datasets and currently used evaluation protocols should be used for benchmarking novel continual learning methods as they no longer pose any significant challenge, as the results in Table 3.3.1 show.

3.2. Related works

Following [75], continual learning methods can be grouped into three categories: regularization, dynamic architectures, and rehearsal.

Regularization These methods typically train a single model on the subsequent tasks and impose specifically defined regularization techniques which penalize the change of important parameters of the model [37, 76].

Dynamic architectures Here, the solution comprises many substructures, which are usually trained in isolation for specific tasks. On top of these structures, a separate mechanism decides which substructure to use during evaluation. In [88, 89, 94] new structural elements are added to the model for each new task which requires growing memory for the whole model. In [83, 84, 85, 86, 87] a large model is considered from which independent submodels are selected for subsequent tasks.

Rehearsal Methods in this group try to prevent forgetting by retraining the model with a combination of new and previous data examples. Methods presented in [82, 95] employ a memory buffer to store all or possibly most relevant previous data examples. To overcome the scalability issues of the standard buffer approach, authors [81] propose to replace the buffer with a generative

[1] Throughout this work, we refer to the hidden representations of autoencoders or variational autoencoders as generative representations. Similarly, we refer to the reconstruction task as the generative task. Although autoencoders are not strictly generative models, we use this term to highlight the difference in the objective of the particular model. In the case of the reconstruction task, the aim is to *generate* a sample, while in the case of classification, the aim is to *discriminate* samples.

model. This method introduced a general schema that was further extended in several new approaches with various generative models [77, 78, 79, 80].

While most of the works develop new methods that are more robust to catastrophic forgetting, only several works investigate the actual phenomenon [91, 96, 97, 92]. Specifically, in [92] authors analyze the relationship between semantic similarity of tasks and magnitude of catastrophic forgetting. In [91] authors show that the effect of catastrophic forgetting diminishes with the prolonged exposure to the domain, which suggests that catastrophic forgetting could be an artifact of immature systems. Another approach [96] investigates the problem with the tools of Explainable Artificial Intelligence (XAI), comparing the effects of catastrophic forgetting for different layers of CNN.

Our work is directly influenced by Thai et al. [90], where authors claim that networks trained in continual reconstruction tasks do not suffer from catastrophic forgetting. We show that different dynamics of forgetting in reconstruction and classification tasks are linked to the representations of the data constructed by the neural networks. Specifically, we argue that learning representations through generative modeling is naturally more aligned with continual learning.

3.3. Methodology

To examine and compare representations from different models, we need to design a fair method to learn these models and collect respective representations. To that end we consider three types of models: discriminative – $\mathcal{D}$, generative based on Autoencoder – $\mathcal{G}_{AE}$ and generative based on Variational Autonecoder [98] – $\mathcal{G}_{VAE}$. To make a fair comparison, these networks share the same architecture, except for the last layer, defined by the model's objective. In the case of the discriminative task, the last layer has output neurons to discriminate between classes. In the generative task, the final layer outputs vectors of equal size as input data. Depending on the model, we train the networks with different objective functions. $\mathcal{D}$ model is trained with CrossEntropy, $\mathcal{G}_{AE}$ minimizes the MSE and the $\mathcal{G}_{VAE}$ uses ELBO. As shown in Fig. ,3.3.1 we train all models on the same training sequence T_N but with different objectives. The index of particular task is denoted by k, where $k = 1,\ldots,N$. To obtain the representations of data for a particular model, we feed the data to the model and collect the representations from the penultimate layer of the model. We

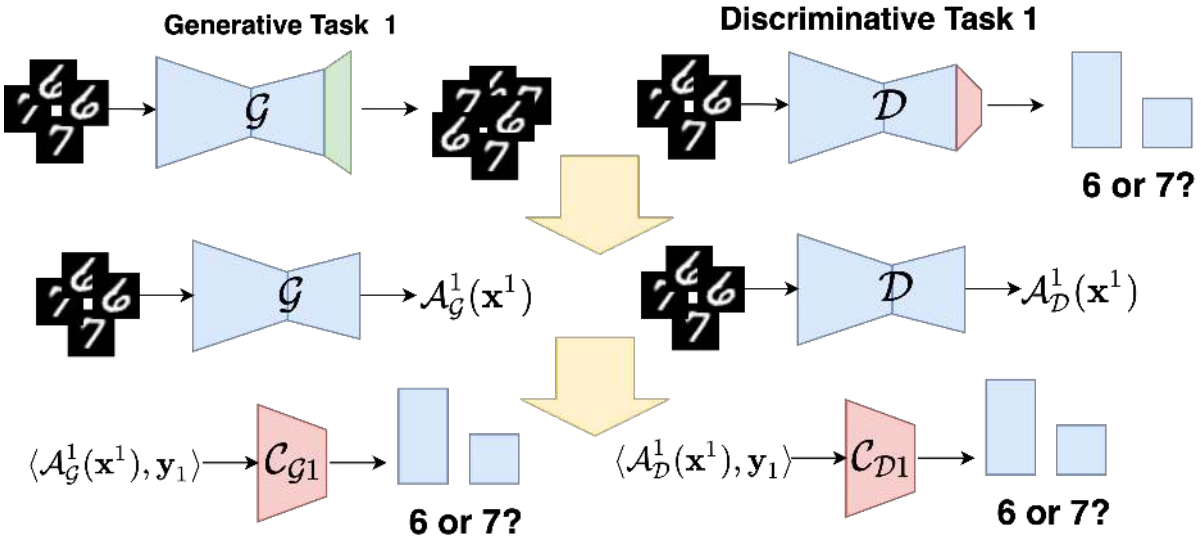


Figure 3.3.1. Schematic overview describing the process of learning and collecting discriminative and generative representations.

use $\mathcal{A}^t_{\mathcal{D}}(\mathbf{x}^k)$, to denote activations from model $\mathcal{D}$ after finishing task t for input data $\langle \mathbf{x}^k \rangle$ from task k.

Datasets We use three commonly used continual learning benchmarks to create a continuous sequence of tasks: splitMNIST, splitFashionMNIST, splitCIFAR. We follow the typical formulation of the continual learning task and split the datasets to 5 disjoint subsets with two classes for each task.

Architectures We use simple autoencoder architecture composed of three encoding fully connected layers and three decoding fully connected layers for all the tasks, followed by ReLU activations except for the last layer. For the generative task, the output layer is a single fully connected layer that maps the penultimate layer's activations to the vector of equal size with input data. For the classification task, we use a multi-head output layer with two neurons per task. For MNIST and FashionMNIST datasets, we use architecture with (512, 256, 8, 256, 512) neurons on each hidden layer. For the CIFAR dataset, we use the same architecture with a bottleneck of 128 neurons.

3.3.1. Hypothesis 1

Discriminative abilities of representations

First, to test our hypothesis, we look at the problem of catastrophic forgetting from the accuracy perspective, as it is the most widely adopted measure of the

catastrophic forgetting phenomenon. To measure the robustness of representations to the catastrophic forgetting, we train the models $\mathcal{D}$, $\mathcal{G}_{AE}$ and $\mathcal{G}_{VAE}$ on identical data sequences T_N and collect their representations from penultimate layer for all data splits. These representations are used to train a set of linear classifiers. Although we collect the representations for all data splits throughout the training, we train respective classifiers only when the corresponding task is active. This means that we train only the first classifier during the first task, and the rest remain untrained. After finishing the first task, we freeze the first classifier for the rest of the training. This way, degradation of its performance can be attributed exclusively to forgetting the representations for the first task, and we can directly measure the amount of forgetting in the model.

More precisely, after each epoch of training, for task data of the form $\langle \mathbf{x}^k, \mathbf{y}^k \rangle$, where k denotes the respective data split, we collect the representations for model $\mathcal{D}$ from its penultimate layer $\mathcal{A}^k_{\mathcal{D}}(\mathbf{x}^j)$ for $j = k$ and train softmax regression classifier $\mathcal{C}_{\mathcal{D}j}$ on dataset of the form $\langle \mathcal{A}^k_{\mathcal{D}}(\mathbf{x}^k), \mathbf{y}^k \rangle$, where k denotes the present task. Next, the trained softmax classifiers are evaluated on the validation dataset after extracting features with respective backbones. Note that in the above approach, classifiers $\mathcal{C}_{\mathcal{D}j}$ for $j > k$, remain untrained. Since the classifiers $\mathcal{C}_{\mathcal{D}j}$ are frozen after completing j-th task, the loss of performance may only be attributed to the drift of representations from the penultimate layer. Therefore, the smaller the loss of the classifier's performance, the more resilient the features are to the catastrophic forgetting. The same procedure is applied to the models $\mathcal{G}_{AE}$ and $\mathcal{G}_{VAE}$ as shown in Fig 3.3.1.

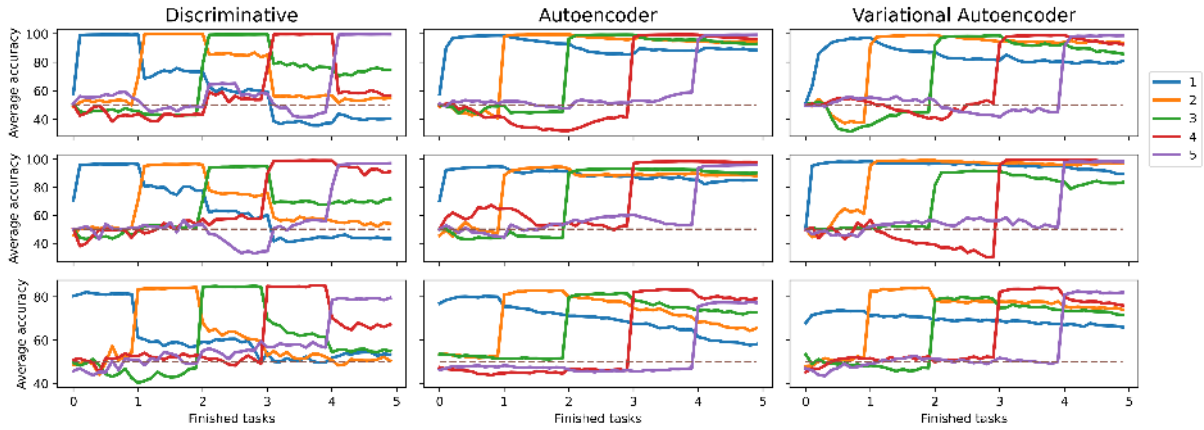


Figure 3.3.2. Results of the first experiment. Each curve represents the average accuracy on specific task with respect to the finished tasks in the continual training. The dashed line is for the reference presenting the performance of a random classifier. Columns represent results for representations of $\mathcal{D}$, $\mathcal{G}_{AE}$ and $\mathcal{G}_{VAE}$ respectively. Top row – MNIST, middle - FashionMNIST, bottom – CIFAR-10.

Fig. 3.3.2 depicts the results of this experiment. Columns represent results for $\mathcal{D}$, $\mathcal{G}_{AE}$, $\mathcal{G}_{VAE}$ respectively. Starting from the top, the rows present the results for MNIST, FashionMNIST, and CIFAR. Different colors present the accuracy for different tasks. As one can see in the left column of Fig. 3.3.2, for all three benchmarks, the classifier's performance trained with representations of discriminative model suddenly drops after introducing new tasks and degrades further to the region of random guessing (depicted as dashed line). In contrast, for the generative case for both models, the performance is stable throughout the whole sequence of training, suggesting that the representations of tested generative models are almost immune to the problem of catastrophic forgetting, especially in the case of simpler datasets as MNIST and FashionMNIST. In the case of CIFAR-10, the amount of forgetting is considerable for the case of $\mathcal{G}_{AE}$. However, the degradation of the performance is more stable than in the case of the discriminative model. This suggests that forgetting in generative models has different nature than forgetting in discriminative models. The right column shows that the representations of $\mathcal{G}_{VAE}$ model suffer the least in the case of CIFAR-10. Explaining this difference requires further analysis, and we foresee it as future work.

Surprisingly, in MNIST and FashionMNIST, achieving an average accuracy above 90% without any dedicated mechanism to overcome catastrophic forgetting is possible. This raises the question of whether these datasets and evaluation protocols should be used to benchmark novel methods in continual learning.

The above experiment proves the validity of the first hypothesis through the lenses of accuracy as it is a widely adopted measure to estimate catastrophic forgetting. However, such an approach may not be fully informative as generative representations can have poor discriminative abilities. To address this limitation, we propose to measure catastrophic forgetting through the index of representations similarity.

Centered Kernel Alignment (CKA)

To further examine the validity of our hypothesis that representations learned by generative models are less susceptible to catastrophic forgetting, we investigate the evolution of network representations in time. Since the above experiment is directly linked to the discriminative abilities of the representations, here we use a task-agnostic measure to directly estimate the drift of the representations in continual learning training. For that purpose, we use the

well-established method of Centered Kernel Alignment (CKA) [99], defined as:

$$\mathrm{CKA}(X,Y) = \frac{\left\|X^T Y\right\|_F^2}{\left\|X^T X\right\|_F \left\|Y^T Y\right\|_F}, \tag{3.1}$$

where $X \in \mathbb{R}^{n \times m_x}$ and $Y \in \mathbb{R}^{n \times m_y}$. CKA takes values from 0 to 1, where 1 means identical representations. Using CKA, it is possible to estimate the similarity of representations obtained for different datasets or different dimensionality. We measure the similarity between representations of the same network collected at different moments of the training of the neural network. Specifically, to analyze the evolution of representations from model $\mathcal{D}$, we collect the reference representations $\mathcal{A}^k_{\mathcal{D}}(\mathbf{x}^k)$ just after finishing task k. Then, to measure the relative drift of representations on task k, we compute the CKA index between the reference and current representations $\mathrm{CKA}(\mathcal{A}^k_{\mathcal{D}}(\mathbf{x}^k), \mathcal{A}^j_{\mathcal{D}}(\mathbf{x}^k))$, for $j \geq k$. We follow this procedure for each task in training sequence T_N. The analogous procedure is applied to the representations of models $\mathcal{G}_{AE}$ and $\mathcal{G}_{VAE}$.

Fig. 3.3.3 presents the results of the experiment. In the case of discriminative representations (top row), we can see an abrupt change of representations just after introducing the new task on all tested datasets. This is in line with the results from the previous experiment, which show that most of the performance is lost during the next task. In the following tasks, the representations stabilize, but these representations are no longer valuable since classifying with them is not better than random guessing (see Fig. 3.3.2). The results from the middle row of Fig. 3.3.3 show that representations from $\mathcal{G}_{AE}$ slightly evolve during the training on consecutive tasks and remain similar to the reference representations. These results directly support hypothesis 1. The representations of $\mathcal{G}_{VAE}$ changes significantly on the first two tasks of simpler datasets (MNIST and FashionMNIST). These results indicate that it takes more time for the VAE model to learn stable features. In CIFAR-10, the amount of forgetting on VAE representations is almost equal to the amount of forgetting on AE representations which in both cases is almost negligible. This may suggest that the more complex the data is, the more general features the generative models learn.

Additionally, the dynamics of changes are less chaotic for the generative model, as the similarity of representations monotonically degrades with each task. This is in stark contrast to the discriminative representations which evolve chaotically. For instance, in the FashionMNIST dataset (middle column, top row), representations of the first task drastically change during the second and

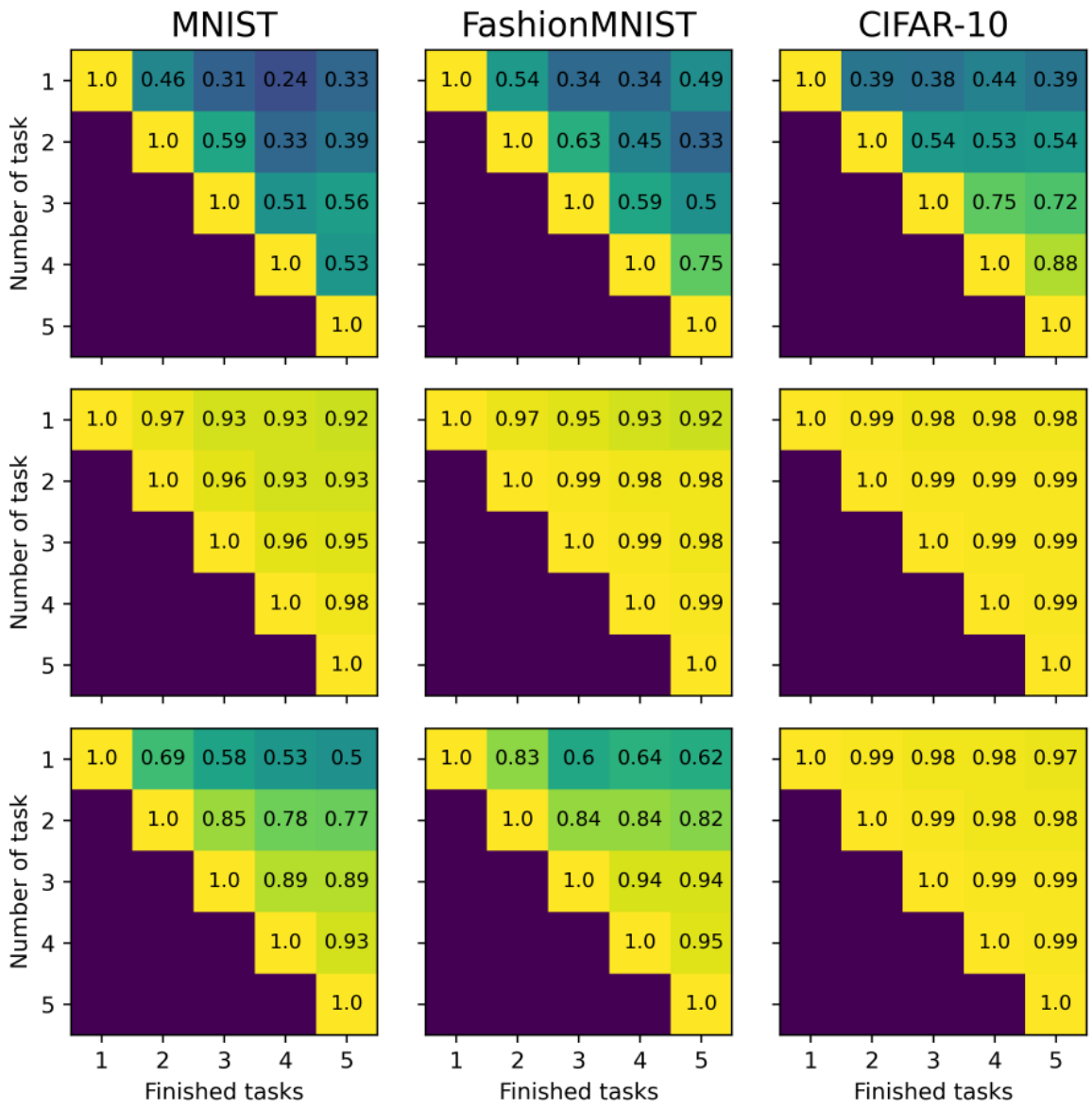


Figure 3.3.3. Illustration of the evolution of representations similarity measure with CKA index. Starting from top the rows represent results of different models: $\mathcal{D}$, $\mathcal{G}_{AE}$ and $\mathcal{G}_{VAE}$ respectively. Columns represent results on different datasets (from left to right) – MNIST, FashionMNIST, CIFAR-10.

third task, then stabilize on the fourth task and unexpectedly become more similar during the last task. This may suggest that although the forgetting occurs in the generative case, it has a gradual form and is more predictable than in the discriminative part.

Model	MNIST	FashionMNIST	CIFAR-10
$\mathcal{D}$	76.8 $_{\pm 12.0}$	74.3 $_{\pm 11.5}$	69.2 $_{\pm 6.4}$
$\mathcal{G}_{AE}$	**96.8 $_{\pm 5.4}$**	92.1 $_{\pm 7.2}$	**80.5 $_{\pm 2.0}$**
$\mathcal{G}_{VAE}$	88.9 $_{\pm 6.7}$	**94.2 $_{\pm 5.5}$**	78.9 $_{\pm 3.0}$

Table 3.3.1. Average accuracy (in %± std) over all tasks after finishing the first task.

Model	MNIST	FashionMNIST	CIFAR-10
$\mathcal{D}$	31.4 $_{\pm 3.1}$	36.0 $_{\pm 3.0}$	23.3 $_{\pm 2.3}$
$\mathcal{G}_{AE}$	**82.9 $_{\pm 2.4}$**	**74.6 $_{\pm 1.6}$**	**44.4 $_{\pm 1.0}$**
$\mathcal{G}_{VAE}$	58.6 $_{\pm 5.2}$	61.1 $_{\pm 0.9}$	37.4 $_{\pm 0.2}$

Table 3.3.2. Average accuracy (in %± std) for classification on the whole validation dataset (without splitting for tasks).

3.3.2. Hypothesis 2

To validate the second hypothesis, which states that *autoencoders and variational autoencoders learn more transferable features than discriminative models*, we change the experimental protocol and learn the models $\mathcal{D}$, $\mathcal{G}_{AE}$ and $\mathcal{G}_{VAE}$ only on the first task in the learning sequence T_N. We adopt this approach as we are interested in measuring the transferability of learned features in the context of future tasks.

After the training for the first task is finished, we follow the procedure visualized in Fig 3.3.1 and collect representations for all data splits. Next, we train all classifiers on respective data splits. Then, we evaluate these classifiers on the validation datasets. Because the backbone of the neural network is trained only for the first task, we measure this way the transferability of the features learned during the first task to other tasks (results of this experiment are presented in the Table 3.3.1). To gain further insights, we train a single classifier on all collected representations. This way, for each dataset, the task is a full 10-way classification. The results of this experiment tell us how general and useful are the features learned during the first task in the context of classifying the whole dataset. Results of this experiment are in the Table 3.3.2.

The superiority of the generative representations in terms of transferability is visible on all considered benchmarks. Although classifiers learned on $\mathcal{G}_{VAE}$ representations from the VAE model performs worse than classifiers trained

on $\mathcal{G}_{AE}$ representations, both of these approaches obtain significantly better results than classifiers trained of model $\mathcal{D}$ representations. In the case of the model $\mathcal{G}_{AE}$ for MNIST and FashionMNIST datasets, the classifiers have a nearly perfect average accuracy of 96.8% and 92.1%, respectively. The results of the classifiers trained on joint datasets (displayed in the brackets in Table 3.3.1) are even more surprising. It turns out that the representations learned by the $\mathcal{G}_{AE}$ model on one task generalize well beyond that one task, which is proved by average accuracy of 82.9%, 74.6%, and 44.36% in 10-way classification tasks on MNIST, FashionMNIST, and CIFAR-10 respectively. On the other hand, the poor performance of classifiers trained with the representations learned by the discriminative model confirms that representations learned this way do not generalize beyond the current task. This poor performance using representations of discriminative models is expected as the discriminative model aims to find the set of features that only separate the current data, resulting in useless features for upcoming tasks. We postulate that this poor transferability of features for upcoming tasks is the main reason for the catastrophic forgetting in discriminative models.

Guided by the above results, we create an additional experiment where we analyze how the training on the first task impacts the reconstruction quality on other tasks. To that end, we carry out experiments only with the generative models $\mathcal{G}_{AE}$ and $\mathcal{G}_{VAE}$. During training on the first task, we evaluate the model on the reconstruction task with data samples from other tasks. In contrast to previous experiments, we do not perform any additional finetuning before evaluating the model. To evaluate the model on different tasks, we feed the data from this task and compute the corresponding reconstruction loss.

Fig. 3.3.4 presents the results of this experiment. The top row presents the results for $\mathcal{G}_{AE}$ for MNIST, FashionMNIST, and CIFAR-10, respectively. The bottom row presents results for $\mathcal{G}_{VAE}$ with the same dataset ordering. In each plot, the blue curve represents the loss on the first task – the one that the model learns. All the plots have the same interesting tendency – the reconstruction loss on all tasks decreases proportionally with the loss of the first task. This shows that the unvisited tasks directly benefit from the training on the first task. In other words, the representations learned during the first task are useful for the following tasks.

In MNIST and FashionMNIST, there is a significant gap between the loss on the current task (blue curve) and other tasks. The gap is more significant in the case of $\mathcal{G}_{VAE}$, which is in line with the results from previous experiments suggesting that VAE based models need more complex datasets to develop gen-

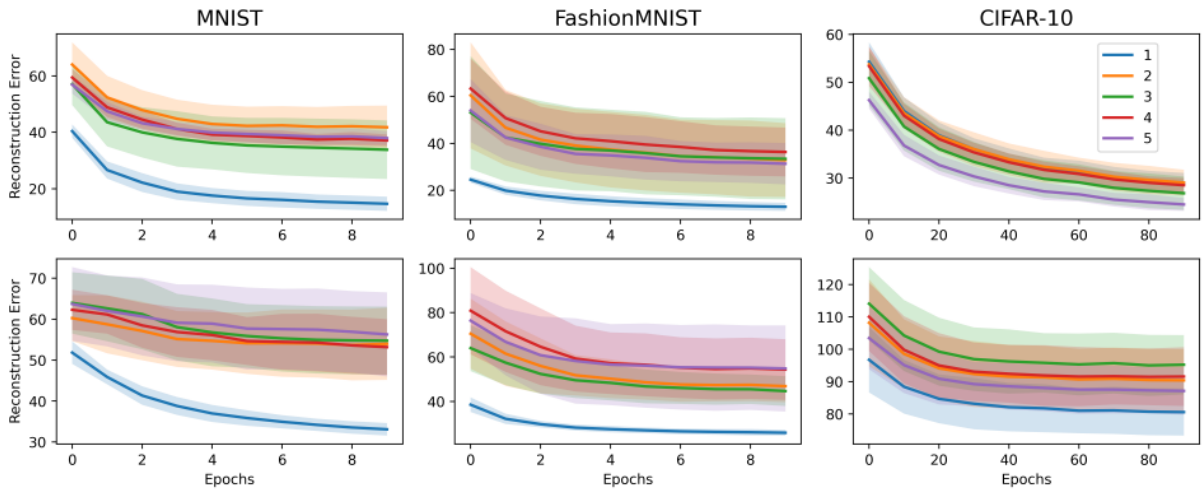


Figure 3.3.4. Reconstruction loss on different tasks for SplitMNIST (left), FashionMNIST (middle) and CIFAR-10 (right) with respect to training epochs on first task. Top row – model $\mathcal{G}_{AE}$, bottom row – $\mathcal{G}_{VAE}$. Results are averaged over 5 independent runs. The shaded area represents the standard deviation.

eral features. This is clearly visible in the rightmost column presenting results for CIFAR-10, where this gap does not exist. This may seem counter-intuitive as CIFAR-10 is the most challenging dataset in this group. However, the more complex the training data is, the more general the features learned by the generative model are.

With these experiments, we provide an intuitive explanation to the phenomenon observed by Thai et al. [90], where authors suggest that the continual reconstruction task does not suffer from catastrophic forgetting. We argue that this is thanks to the generality of features learned by the generative model during the first task since the features learned by the autoencoder during the first task are also useful for the upcoming tasks. As can be observed in Fig 3.3.4) the model directly benefits from the learned features on previous tasks, and thus there is no interference between them, which results in the lack of catastrophic forgetting.

3.4. Discussion

In this work, we stated two hypotheses concerning different representations and their impact on catastrophic forgetting. We carried out experiments to empirically validate our hypotheses. From the performed experiments, it follows that the dynamics of the catastrophic forgetting for discriminative and generative representations are different. In particular, results from Fig. 3.3.3 show that

the forgetting of generative representations is gradual and monotonic. These properties suggest that it might be possible to model the effect of catastrophic forgetting in generative models precisely. However, to that end, one has to measure the effects of catastrophic forgetting exactly. Currently used proxy tasks for that purpose, such as classification tasks, hinder the analysis. Since it becomes more evident that catastrophic forgetting is not a homogeneous phenomenon and its character depends on the task or learning paradigm, we foresee the need for task agnostic measures of catastrophic forgetting. Such work would greatly benefit the community and give us unprecedented insights into the nature of the phenomenon.

From the practical point of view, this paper offers a new understanding of the dynamics of autoencoders and variational autoencoders learning in the continual scenario and their ability to generalize obtained knowledge to future tasks. These results can be directly adapted to the currently used generative rehearsal methods making them more robust or even immune to catastrophic forgetting.

Last, this work is the first step towards a deeper understanding of the similarities and dissimilarities of generative and discriminative representations. While several papers discuss this relation, *e.g.,* [100] many questions remain unanswered, especially in the context of continual learning. How can one use the mechanisms of generative learning to obtain better representations for discrimination tasks in continual learning? Is the generic nature of generative representations the only explanation for the lack of catastrophic forgetting? Answering these and other questions, we left as future work.

4. On consequences of finetuning on data with highly discriminative features

4.1. Introduction

Deep learning has witnessed remarkable advancements in various domains, driven by the ability of neural networks to learn intricate patterns from data. One key aspect contributing to their success is the process of transfer learning, where pre-trained models are fine-tuned on specific tasks, leveraging knowledge acquired from previous training [101, 102]. This technique is especially important in the advent of training ever-growing models such as Large Language Models (LLMs) [11, 103, 104] or massive ViTs [105, 106]. However, while transfer learning is a powerful tool, it is not without its nuances.

This work presents a thought-provoking experiment exposing a network's tendency to greedily follow the simplest discriminative features present in the data. This phenomenon was observed in multiple works and is commonly called simplicity bias [107, 108, 109]. Here, we take a step further and investigate the implications of this behavior in the realms of transfer learning.

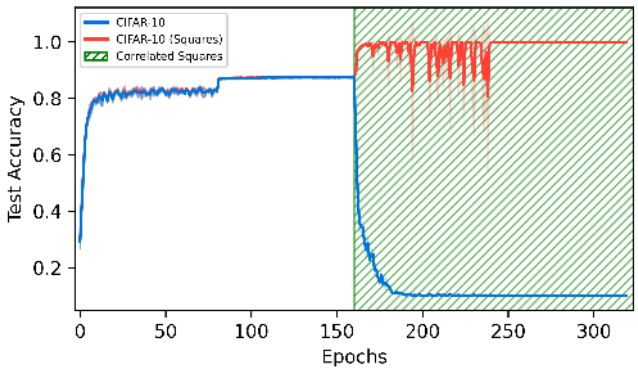
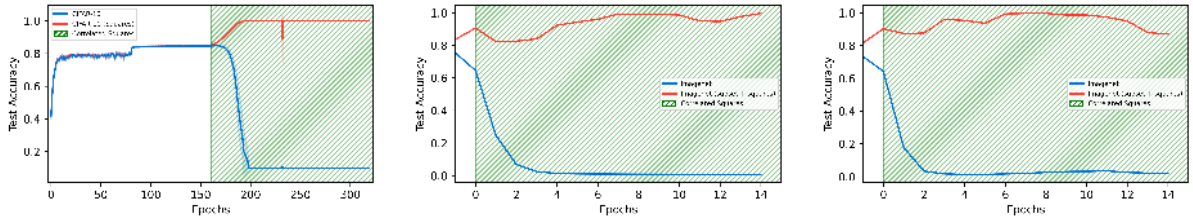


Figure 4.1.1. Feature erosion for VGG-19 on CIFAR-10. Introducing a simpler discriminative feature (green hatch area) solely focuses the model on that feature (red curve) and abruptly erases previous knowledge about the data (blue line).

To investigate this effect, we train the network on CIFAR-10 to perfect training accuracy (error-free). Next, we introduce a highly discriminative, class-correlated pattern to the corner of each dataset image and proceed with the training. Surprisingly, as shown in Fig. 6.1.1, despite the model's perfect accuracy, finetuning it on the oversimplified task causes an abrupt

performance loss (blue curve) and pushes the model to focus solely on the novel pattern. We call this phenomenon *feature erosion*. Our analysis shows that during the fine-tuning phase, the pretrained model greedily abandons salient, generalizing features in favor of the new discriminative ones.

In Section 4.2, we define details of the experiments and investigate the breadth of the phenomenon 4.1.2, presenting that all tested contemporary neural networks exhibit this behavior. Next, we investigate the detrimental effect of feature erosion on the model's representation formation, transfer learning, and plasticity. Section 4.3 discusses the phenomenon's implications, its novelty concerning related works, and its hazards to real-world applications.



(a) ResNet 18 - CIFAR-10 **(b)** ResNet 50 - ImageNet **(c)** VGG-19 - ImageNet

Figure 4.1.2. Feature erosion effect for ResNet18 trained on CIFAR-10 (left) ResNet50 trained of ImageNet (middle) and VGG-19 trained on ImageNet (right). ImageNet models used pretrained weights provided by PyTorch. ResNet18 reached 100% training accuracy before introducing the discriminative pattern to the dataset.

4.2. Experiments and results

In this section, we explore the phenomenon we introduced in the previous section, aiming to understand its severity and impact on network behavior. We initiate our investigation by examining the robustness of feature erosion across different network architectures and datasets (Fig. 4.1.2). Next, we assess the detrimental effect of the phenomenon on the network's representations (Fig. 4.2.1) and show that pursuing simpler, discriminative patterns collapse the network's rank (Fig. 5.2.1). We hypothesize that this effect is linked with the loss of plasticity observed in further training of that network (Fig. 4.2.3).

Experimental setup We will now delve into feature erosion using ResNet-18, ResNet-50, and VGG-19 models trained on either the CIFAR-10 or ImageNet dataset. For CIFAR-10, these networks underwent 160 training epochs, consistently achieving 100% training accuracy before introducing the oversimplified second task. The same hyperparameters were maintained for an additional 160

epochs during the second task, following recommendations for optimal model performance by Liu et al. [110].

Regarding the ImageNet models, we utilized PyTorch's pre-trained weights and randomly selected a subset of 10 classes from the ImageNet dataset for the oversimplified task. To create this dataset, we superimposed squares of the same size and placement on the training images, with each square's color representing the image's class. In most of our experiments, we applied these squares to all training images, the exception is presented in Fig. 4.3.1, where we investigated the impact of that ratio on the model's performance.

Feature Erosion for different models and datasets

The performance of these models is illustrated in Figure 4.1.2, showcasing test accuracy curves during both the pretraining phase (white background) and the subsequent fine-tuning phase on the oversimplified task (green-hatched background). The results clearly indicate that all neural architectures and datasets exhibit feature erosion, resulting in a noticeable decline in test accuracy on the pre-training dataset. Notably, as the sole distinction between the first and second tasks is the presence of these colored squares, the dramatic shift from near-perfect accuracy in the first task to random-chance accuracy in the second task implies that the model exclusively focuses on the colored squares.

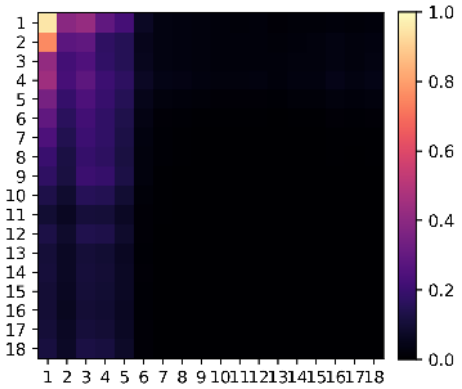


Figure 4.2.1. Feature erosion impacts CKA similarity between representations from different layers extracted from the model after completing the 1st and 2nd tasks. The experiment was conducted using the VGG-19 model and the CIFAR-10 dataset.

Unpacking Feature Erosion: Analyzing Representations

Having observed significant shifts in model performance on established benchmarks, our objective is to investigate the impact of fine-tuning on oversimplified datasets on the model's representations.

In our subsequent experiment, we perform a comparative analysis of representations at each layer after training on the first task and subsequent training on the second task. We employ Centered Kernel Alignment (CKA) [111] as a metric to quantify similarity. We extract representations from standard CIFAR-10 datasets (without color squares) for both models to isolate the impact of weight evolution on model representations.

As illustrated in Figure 4.2.1, the representations show substantial dissimilarity. Most notably, most diagonal elements are black, indicating minimal similarity between representations within the same layers after undertaking different tasks. The lighter colors in the top left corner suggest that changes during the fine-tuning phase commence early in the network, particularly in the bottom layers.

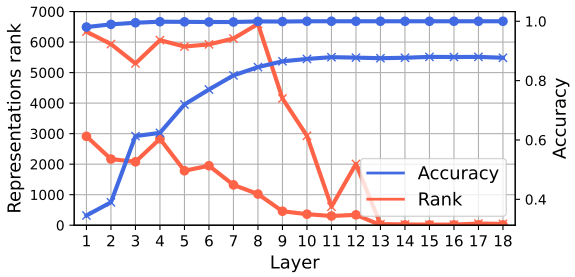


Figure 4.2.2. Feature Erosion collapses rank of the hidden representations and impacts linear probing accuracy of VGG-19 trained on CIFAR10. Crosses refer to model after pre-training, dots refer to the model after finetuning. Blue color refer to the test accuracy, orange color refers to the representations rank.

Given the observed dramatic changes in representations across nearly all layers, we delve deeper into understanding the underlying dynamics. In Fig. 5.2.1, we investigate feature erosion within each layer using linear probing and the numerical rank of representations. A comparison of the linear probing plots reveals that after fine-tuning (blue dots), the model fully adapts to the new data, with the second layer achieving accuracy levels comparable to those of the entire model. Furthermore, the numerical rank of representations experiences a substantial decline at each layer following fine-tuning (red dots), indicating that the model, starting from the initial layers, probably adheres to the simplicity bias [112] and projects the data into smaller subspaces.

Loss of plasticity

To better understand the impact of feature erosion on network performance, we examined the network's ability to relearn information from a previous task. We conducted experiments involving a sequence of three tasks: CIFAR-10 (Task 0), an oversimplified version of CIFAR-10, and the standard CIFAR-10 once again (Task 2).

Typically, relearning is expected to be faster and require fewer computational resources than training from scratch. However, our results, as shown in Figure 1, indicate a deviation from this expected behavior. In this setup, the network not only learns more slowly compared to training from scratch but also fails to achieve the same level of performance within the same computational budget as the network trained from scratch. This performance difference is commonly referred to as "loss of plasticity" and is often associated with the degradation of the penultimate layer's rank. While we speculate that this explanation may

apply in our case, a thorough investigation of this hypothesis is beyond the scope of our current research.

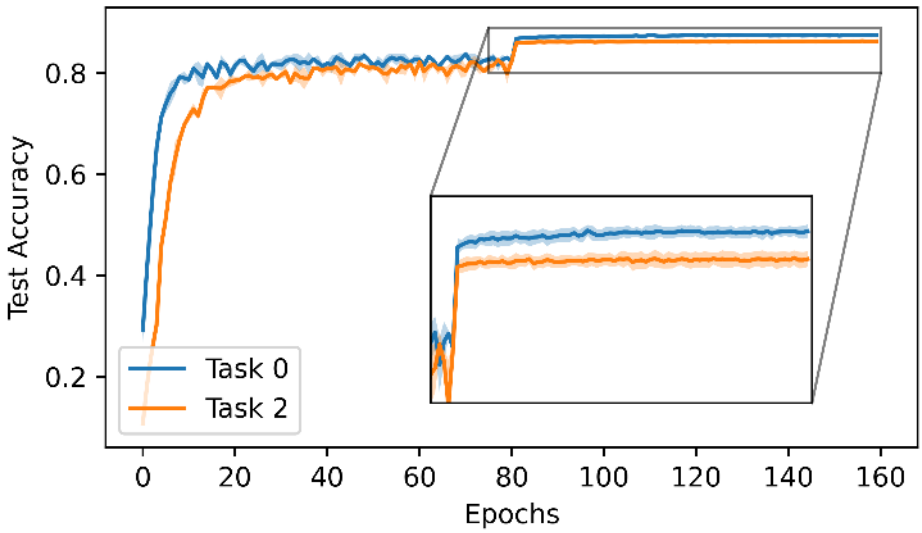


Figure 4.2.3. Loss of plasticity. The model is trained on the sequence of 3 tasks. The first (Task 0) and the last (Task 2) are standard CIFAR-10 datasets. The middle task is CIFAR-10 with correlated squares.

4.3. Discussion

In this study, we have uncovered a nuanced aspect of catastrophic forgetting, demonstrating that it can occur even when the full dataset is present in the new task data. Our experiments, which include GradCam analysis, reveal that despite achieving 100% training accuracy after introducing highly discriminative features to the model, it begins to focus exclusively on these new features, leaving behind previously developed ones. A deeper investigation using techniques such as Centered Kernel Alignment (CKA) exposes significant changes in representations, along with a deterioration in model performance indicated by a rank collapse of representations in nearly all layers.

Our research expands our knowledge of the complex relationship between task similarity [46, 113], forgetting, and transfer dynamic [114]. On the one hand, recent studies have revealed that intermediate task similarity tends to contribute most to catastrophic forgetting [46, 113]. The task similarity is the use of a "data-mixing framework," which combines images and labels from two distinct datasets of equal size. However, our experimental setup features identical datasets, differing only in a small image segment transitioning from

random to highly correlated with a specific class. While this does not contradict earlier findings, it certainly introduces a novel perspective on the phenomenon. On the other hand, our observation also has implications in light of recent research [114], which suggests that less forgetful representations result in improved performance on new tasks, indicating a robust relationship between retaining previous information and enhanced learning efficiency. In this context, our toy example falsifies the reverse implication, i.e., the model exhibits perfectly transferable features yet forgets them in favor of features with greater predictive power.

In our concluding experiment, extending our analysis of feature erosion, we explore the relation between forgetting and the ratio of samples containing oversimplified discriminative patterns. In Fig. 4.3.1, we observe a non-linear relationship between the number of oversimplified samples in the training dataset and the extent of forgetting. However, even modest ratios of oversimplified data are enough to induce forgetting in the model. This reinforces the importance of the phenomenon.

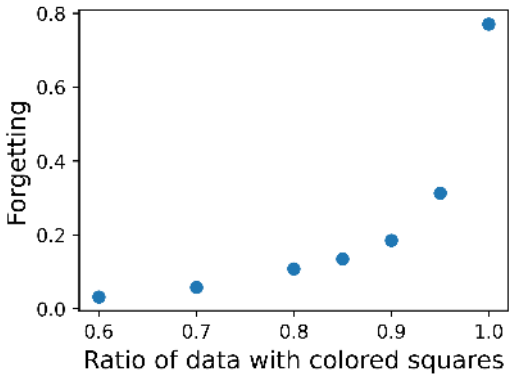


Figure 4.3.1. The stronger the discriminative pattern (higher ratio), the higher the forgetting of the model. Each dot represents a single model trained on CIFAR-10 and fine-tuned on an oversimplified task with different ratios of images with colored squares.

The situation we present in this study has practical significance, especially in scenarios involving incremental learning in various domains. Real-world applications that continually receive new data with limited human involvement might come across samples containing highly predictive patterns. This can lead to the loss of previously acquired knowledge. For instance, in medical imaging, where data artifacts may correlate with task objectives, the phenomenon of feature erosion could be a frequent concern.

Additionally, our findings connect with existing literature on concepts such as simplicity bias [115, 116] and gradient starvation [117]. Our results suggest that simplicity bias not only affects generalization but can also disrupt previously well-functioning representations. The resilience of simplicity bias to approaches like ensembles or adversarial training raises questions about the effectiveness of common continual learning methods.

Finally, there may be a positive aspect to this phenomenon. In the current

era of heightened focus on AI ethics, machine unlearning [118] and fairness in deep learning [119] are prominent topics. Our study prompts the question of whether intentionally introducing highly discriminatory patterns to unwanted samples can facilitate the intentional forgetting of such samples, a topic that warrants further exploration.

In summary, our work reveals a novel facet of catastrophic forgetting, challenging conventional wisdom about its occurrence and implications. These findings have relevance for both the field of machine learning and practical applications that involve continual learning with evolving data.

5. The Tunnel Effect: Building Data Representations in Deep Neural Networks

Abstract

Deep neural networks are widely known for their remarkable effectiveness across various tasks, with the consensus that deeper networks implicitly learn more complex data representations. This paper shows that sufficiently deep networks trained for supervised image classification split into two distinct parts that contribute to the resulting data representations differently. The initial layers create linearly-separable representations, while the subsequent layers, which we refer to as *the tunnel*, compress these representations and have a minimal impact on the overall performance. We explore the tunnel's behavior through comprehensive empirical studies, highlighting that it emerges early in the training process. Its depth depends on the relation between the network's capacity and task complexity. Furthermore, we show that the tunnel degrades out-of-distribution generalization and discuss its implications for continual learning.

5.1. Introduction

Neural networks have been the powerhouse of machine learning in the last decade. A significant effort has been put into understanding the mechanisms underlying their effectiveness. One example is the analysis of building representations in neural networks applied to image processing [120]. The consensus is that networks

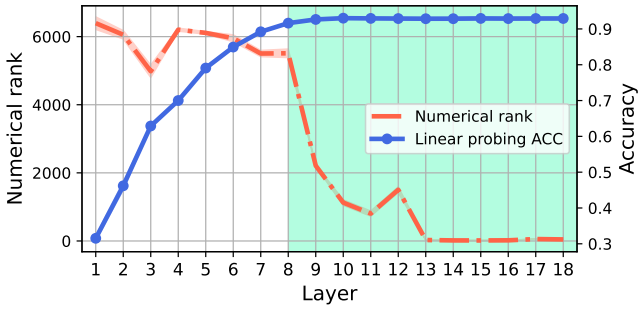


Figure 5.1.1. The tunnel effect for VGG19 trained on the CIFAR-10. In the tunnel (shaded area), the performance of linear probes attached to each layer saturates (blue line), and the representation rank is steeply reduced (red dashed line).

learn to use layers in the hierarchy by extracting more complex features than the layers before [121, 122], meaning that each layer contributes to the final network performance.

Extensive research has shown that increasing network depth exponentially enhances capacity, measured as the number of linear regions [123, 124, 125]. However, practical scenarios reveal that deep and overparameterized neural networks tend to simplify representations with increasing depth [126, 127]. This phenomenon arises because, despite their large capacity, these networks strive to reduce dimensionality and focus on discriminative patterns during supervised training [126, 127, 128, 129]. Motivated by these findings, we aim to investigate this phenomenon further and formulate the following research question:

How do representations depend on the depth of a layer?

Our investigation focuses on severely overparameterized neural networks through the prism of their representations as the core components for studying neural network behavior [130, 131].

We extend the commonly held intuition that deeper layers are responsible for capturing more complex and task-specific features [132, 133] by showing that neural networks after learning low and high-level features use the remaining layers for compressing obtained representations.

Specifically, we demonstrate that deep neural networks split into two parts exhibiting distinct behavior. The first part, which we call the extractor, builds representations, while the other, dubbed *the tunnel*, propagates the representations further to the model's output, compressing them significantly. As we show, this behavior has important implications for generalization, transfer learning, and continual learning. To investigate the tunnel effect, we conduct multiple experiments that support our findings and shed some light on the potential source of this behavior. Our findings can be summarized as follows:

- We conceptualize and extensively examine the tunnel effect, namely, deep networks naturally split into *the extractor* responsible for building representations and *the compressing tunnel*, which minimally contributes to the final performance. The extractor-tunnel split emerges early in training and persists later on.
- We show that the tunnel deteriorates the generalization ability on out-of-distribution data.
- We show that the tunnel exhibits task-agnostic behavior in a continual

learning scenario. Simultaneously it leads to higher catastrophic forgetting of the model.

5.2. The tunnel effect

The paper introduces and studies the dynamics of representation building in overparameterized deep neural networks called *the tunnel effect*. The following section validates the tunnel effect hypothesis in a number of settings. Through an in-depth examination in Section 5.3.1, we reveal that the tunnel effect is present from the initial stages and persists throughout the training process. Section 5.3.2 focuses on the out-of-distribution generalization and representations compression. Section 5.3.3 hints at important factors that impact the depth of the tunnel. Finally, in Section 5.4, we confront an auxiliary question: How does the tunnel's existence impact a model's adaptability to changing tasks and its vulnerability to catastrophic forgetting? To answer these questions we formulate our main claim as:

The tunnel effect hypothesis: *Sufficiently large* [1] *neural networks develop a configuration in which network layers split into two distinct groups. The first one which we call the extractor, builds linearly-separable representations. The second one, the tunnel, compresses these representations, hindering the model's out-of-distribution generalization.*

5.2.1. Experimental setup

To examine the phenomenon, we designed the setup to include the most common architectures and datasets, and use several different metrics to validate the observations.

Architectures We use three different families of architectures: MLP, VGGs, and ResNets. We vary the number of layers and width of networks to test the generalizability of results. See details in Appendix 5.8.1.

Tasks We use three image classification tasks to study the tunnel effect: CIFAR-10, CIFAR-100, and CINIC-10. The datasets vary in the number of classes: 10 for CIFAR-10 and CINIC-10 and 100 for CIFAR-100, and the number of samples: 50000 for CIFAR-10 and CIFAR-100 and 250000 for CINIC-10). See details in Appendix 5.8.2.

[1] We note that 'sufficiently large' covers most modern neural architectures, which tend to be heavily overparameterized.

We probe the effects using: *the average accuracy of linear probing, spectral analysis of representations, and the CKA similarity between representations*. Unless stated otherwise, we report the average of 3 runs.

Accuracy of linear probing: a linear classification layer is attached to a given layer ℓ of the neural network. We train this layer on the classification task and report the average accuracy. This metric measures to what extent ℓ's representations are linearly separable.

Numerical rank of representations: we compute singular values of the sample covariance matrix for a given layer ℓ of the neural network. Using the spectrum, we estimate the numerical rank of the given representations matrix as the number of singular values above a certain threshold σ. The threshold σ is set to $\sigma_1 * 1e-3$, where σ_1 is the highest singular value of the particular matrix. The numerical rank of the representations matrix can be interpreted as the measure of the degeneracy of the matrix.

CKA similarity: is a metric computing similarity between two representations matrices. Using this normalized index, we can identify the blocks of similar representations within the network. The definition and more details can be found in Appendix 5.12.

Inter and Intra class variance: inter-class variance refers to the measure of dispersion or dissimilarity between different classes or groups in a dataset, indicating how distinct they are from each other. Intra-class variance, on the other hand, measures the variability within a single class or group, reflecting the homogeneity or similarity of data points within that class. The exact formula for computing these values can be found in Appendix 5.13

5.2.2. The main result

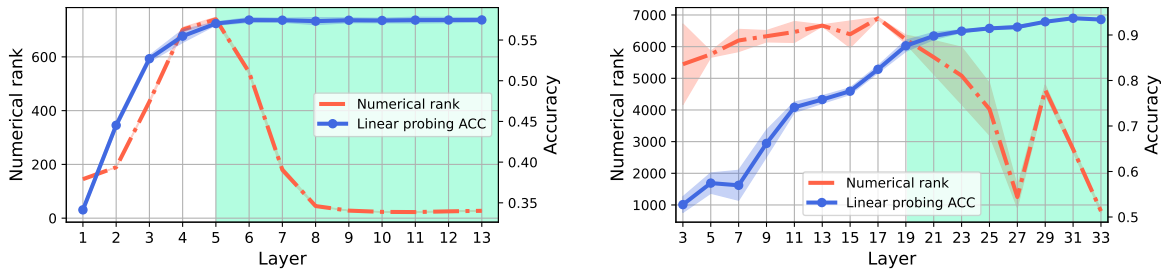
Table 5.2.1 presents our main result. Namely, we report the network layer at which the tunnel begins which we define as the point at which the network reaches 95% (or 98%) of its final accuracy. We found that all tested architectures exhibit the extractor-tunnel structure across all datasets used in the evaluation, but the relative length of the tunnel varies between architectures.

We now discuss the tunnel effect using MLP-12, VGG-19, and ResNet-34 on CIFAR-10 as an example. The remaining experiments (for other architectures, datasets combinations) are available in Appendix 5.9. As shown in Figure 6.1.1 and Figure 5.2.1, the early layers of the networks, around five for MLP and eight for VGG, are responsible for building linearly-separable representations. Linear probes attached to these layers achieve most of the network's final performance. These layers mark the transition between the extractor and the tunnel part

Architecture	# layers	Dataset	>0.95	>0.98
MLP	13	CIFAR-10	4 (31%)	5 (38%)
VGG	19	CIFAR-10	7 (36%)	7 (36%)
		CIFAR-100	8 (42%)	8 (42%)
		CINIC-10	7 (36%)	7 (36%)
ResNet	34	CIFAR-10	20 (58%)	29 (85%)
		CIFAR-100	29 (85%)	30 (88%)
		CINIC-10	17 (50%)	17 (50%)

Table 5.2.1. The tunnel of various lengths is present in all tested configurations. For each architecture and dataset, we report the layer for which the *average linear probing accuracy is above* 0.95 *and* 0.98 *of the final performance*. The values in the brackets describe the part of the network utilized for building representations with the extractor.

(shaded area). In the case of ResNets, the transition takes place in deeper stages of the network at the 19^{th} layer.



(a) MLP 12 **(b)** ResNet 34

Figure 5.2.1. The tunnel effect for networks trained on CIFAR-10. The blue line depicts the linear probing accuracy, and the shaded area depicts the tunnel. The red dashed line is the numerical rank of representations. The spike in the ResNet-34 representations rank coincides with the end of the penultimate residual stage.

While the linear probe performance nearly saturates in the tunnel part, the representations are further refined. Figure 5.2.1 shows that the numerical rank of the representations (red dashed line) is reduced to approximately the number of CIFAR-10 classes, which is similar to the neural collapse phenomenon observed in [129]. For ResNets, the numerical rank is more dynamic, exhibiting a spike at 29^{th} layer, which coincides with the end of the penultimate residual block. Additionally, the rank is higher than in the case of MLPs and VGGs.

Figure 5.2.2 reveals that for VGG-19 the inter-class representations variation decreases throughout the tunnel, meaning that representations clusters contract towards their centers. At the same time, the average distance between the centers of the clusters grows (inter-class variance). This view aligns with the

observation from Figure 5.2.1, where the rank of the representations drops to values close to the number of classes. Figure 5.2.2 (right) presents an intuitive explanation of the behavior with UMAP [134] plots of the representations before and after the tunnel.

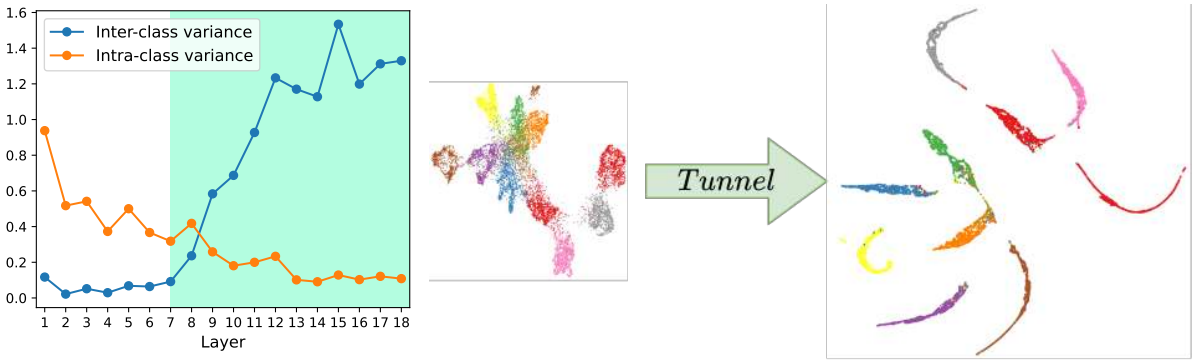


Figure 5.2.2. The tunnel compresses the representations discarding indiscriminative features. **Left:** The evolution and inter and intra-class variance of representations within the VGG-19 network. **Right:** UMAP plot of representations before (7^{th} layer) and after (18^{th} layer) the tunnel.

To complement this analysis, we studied the similarity of MLPs representations using the CKA index and the L1 norm of representations differences between the layers. Figure 5.2.3 shows that the representations change significantly in early layers and remain similar in the tunnel part when measured with the CKA index (left). The L1 norm of representations differences between the layers is computed on the right side of Figure 5.2.3.

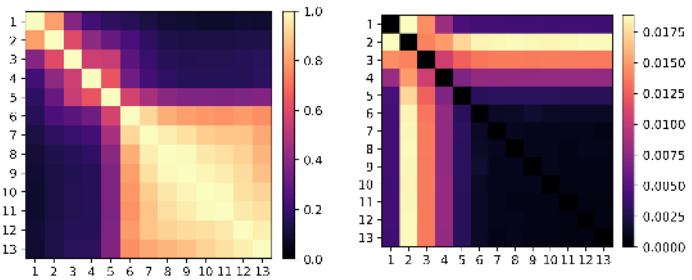


Figure 5.2.3. Representations within the tunnel are similar to each other for MLP with 12 hidden layers trained on CIFAR-10. Comparison of representations with CKA index (left) and average L1 norm of representations differences.

5.3. Tunnel effect analysis

This section provides empirical evidence contributing to our understanding of the tunnel effect. We hope that these observations will eventually lead to explanations of this phenomenon. In particular, we show that a) the tunnel develops early during training time, b) it compresses the representations and

hinders OOD generalization, and c) its size is correlated with network capacity and dataset complexity.

5.3.1. Tunnel development

Motivation In this section, we investigate tunnel development during training. Specifically, we try to understand whether the tunnel is a phenomenon exclusively related to the representations and which part of the training is crucial for tunnel formation.

Experiments We train a VGG-19 on CIFAR-10 and save intermediate checkpoints every 10 epochs of training. We use these checkpoints to compute the layer-wise weight change during training (Figure 5.3.1) and the evolution of numerical rank throughout the training (Figure 5.3.2).

Results Figure 5.3.1 shows that the split between the extractor and the tunnel is also visible in the parameters space. It could be perceived already at the early stages, and after that, its length stays roughly constant. Tunnel layers change significantly less than layers from the extractor. This result raises the question of whether the weight change affects the network's final output. Inspired by [135], we reset the weights of these layers to the state before optimization. However, the performance of

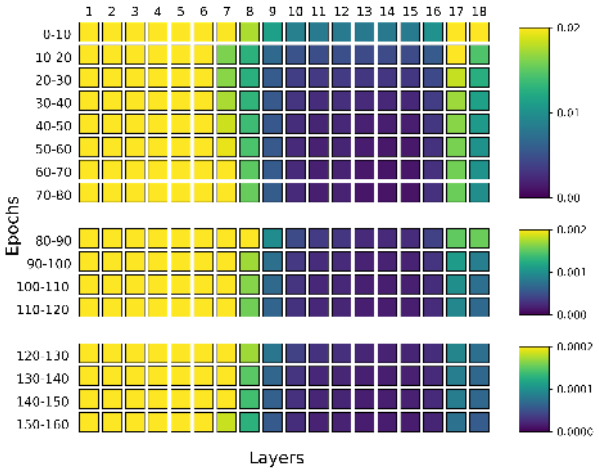


Figure 5.3.1. Early in training, tunnel layers stabilize. Color-coded: weight difference norm between consecutive checkpoints for each layer. Norm calculated as $\frac{1}{\sqrt{nm}} \left\| \theta_d^{\tau_1} - \theta_d^{\tau_2} \right\|_2$, where $\theta_d^{\tau} \in \mathbb{R}^{nm}$ is flattened weight matrix at layer d, checkpoint τ. Values capped at 0.02 for clarity. The learning rate decayed (by 10^{-1}) at epochs 80 and 120, and the scale adapted accordingly. Experiment: VGG-19 on CIFAR-10.

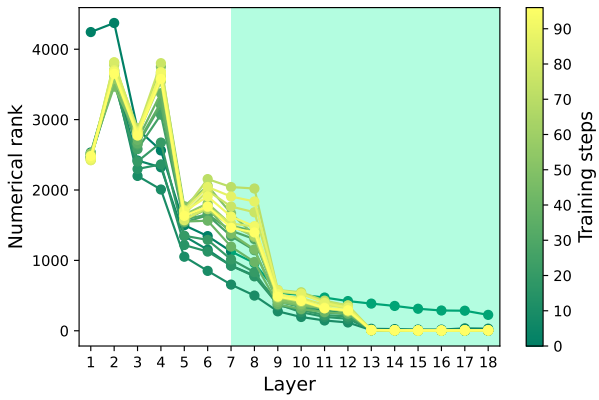


Figure 5.3.2. The representations rank for deeper layers collapse early in training. The curves present the evolution of representations' numerical rank over the first 75 training steps for all layers of the VGG-19 trained on CIFAR-10. We present a more detailed tunnel development analysis in Appendix 5.14.

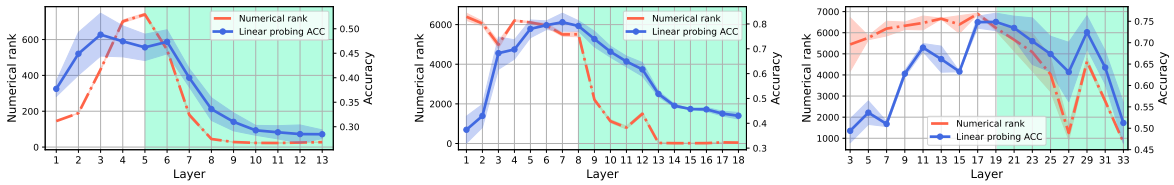
the model deteriorated significantly. This suggests that although the change within the tunnel's parameters is relatively small, it plays an important role in the model's performance. Figure 5.3.2 shows that this apparent paradox can be better understood by looking at the evolution of representations' numerical rank during the very first gradient updates of the model. Throughout these steps, the rank collapses to values near-the-number of classes. It stays in this regime until the end of the training, meaning that the representations of the model evolve within a low-dimensional subspace. It remains to be understood if (and why) low-rank representations and changing weights coincide with forming linearly-separable representations.

Takeaway Tunnel formation is observable in the representation and parameter space. It emerges early in training and persists throughout the whole optimization. The collapse in the numerical rank of deeper layers suggest that they preserve only the necessary information required for the task.

5.3.2. Compression and out-of-distribution generalization

Motivation Practitioners observe intermediate layers to perform better than the penultimate ones for transfer learning [136, 137, 138]. However, the reason behind their effectiveness remains unclear [139]. In this section, we investigate whether the tunnel and, specifically, the collapse of numerical rank within the tunnel impacts the performance on out-of-distribution (OOD) data.

Experiments We train neural networks (MLPs, VGG-19, ResNet-34) on a source task (CIFAR-10) and evaluate it with linear probes on the OOD task, in this case, a subset of 10 classes from CIFAR-100. We report the accuracy of linear probing and the numerical rank of the representations.



(a) MLP 12 **(b)** VGG-19 **(c)** ResNet 34

Figure 5.3.3. The tunnel degrades the out-of-distribution performance correlated with the representations' numerical rank. The accuracy of linear probes (blue) was trained on the out-of-distribution data subset of 10 classes from CIFAR-100. The backbone was trained on CIFAR-10. The shaded area depicts the tunnel, and the red dashed line depicts the numerical rank of representations.

Results Our results presented in Figure 5.3.3 reveal that *the tunnel is responsible for the degradation of out-of-distribution performance*. In most of

our experiments, the last layer before the tunnel is the optimal choice for training a linear classifier on external data. Interestingly, we find that the OOD performance is tightly coupled with the numerical rank of the representations, which significantly decreases throughout the tunnel.

To assess the generalization of our findings we extend the proposed experimentation setup to additional dataset. To that end, we train a model on different subsets of CIFAR-100 while evaluating it with linear probes on CIFAR-10. The results presented in Figure 5.3.4 are consistent with our initial findings. We include detailed analysis with reverse experiment (CIFAR-10 → CIFAR-100), additional architectures and datasets in the Appendix 5.10.

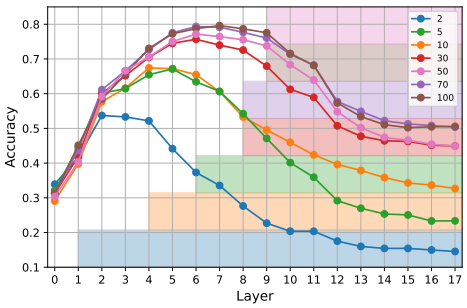


Figure 5.3.4. Fewer classes in the source task create a longer tunnel, resulting in worse OOD performance. The network is trained on subsets of CIFAR-100 with different classes, and linear probes are trained on CIFAR-10. Shaded areas depict respective tunnels .

In all tested scenarios, we observe a consistent relationship between the start of the tunnel and the drop in OOD performance. An increasing number of classes in the source task result in a shorter tunnel and a later drop in OOD performance. In the fixed source task experiment (Appendix 5.10), the drop in performance occurs around the 7^{th} layer of the network for all tested target tasks, which matches the start of the tunnel. This observation aligns with our earlier findings suggesting that the tunnel is a prevalent characteristic of the model rather than an artifact of a particular training or dataset setup.

Moreover, we connect the coupling of the numerical rank of the representations with OOD performance, to a potential tension between the objective of supervised learning and the generalization of OOD setup. Analogous tension was observed in [140] where adversarial robustness is at odds with model's accuracy. The results in Figure 5.3.3 align with the findings presented in Figure 5.2.2, demonstrating how the tunnel compresses clusters of class-wise representations. In work [141], the authors show that reducing the variation within each class leads to lower model transferability. Our experiments support this observation and identify the tunnel as the primary contributor to this effect.

Takeaway Compression of representations happening in the tunnel severely

degrades the OOD performance of the model which is tightly coupled with the drop of representations rank.

5.3.3. Network capacity and dataset complexity

Motivation In this section, we explore what factors contribute to the tunnel's emergence. Based on the results from the previous section we explore the impact of dataset complexity, network's depth, and width on tunnel emergence.

Experiments First, we examine the impact of networks' depth and width on the tunnel using MLPs (Figure 5.3.5), VGGs, and ResNets (Table 5.3.1) trained on CIFAR-10. Next, we train VGG-19 and ResNet34 on CIFAR-{10,100} and CINIC-10 dataset investigating the role of dataset complexity on the tunnel.

Results Figure 5.3.5 shows that the depth of the MLP network has no impact on the length of the extractor part. Therefore increasing the network's depth contributes only to the tunnel's length. Both extractor section and numerical rank remain relatively consistent regardless of the network's depth, starting the tunnel at the same layer. This finding suggests that overparameterized neural networks allocate a fixed capacity for a given task independent of the overall capacity of the model.

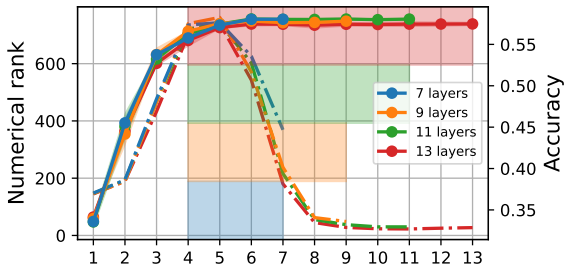


Figure 5.3.5. Networks allocate a fixed capacity for the task, leading to longer tunnels in deeper networks. The extractor is consistent across all scenarios, with the tunnel commencing at the 4th layer.

	¼	1	2
VGG-16	8 (50%)	7 (44%)	7 (44%)
VGG-19	8 (42%)	7 (37%)	7 (37%)
ResNet18	15 (83%)	13 (72%)	13 (72%)
ResNet34	24 (68%)	20 (59%)	24 (68%)

Table 5.3.1. Widening networks layers results in a longer tunnel and shorter extractor. Column headings describe the factor in which we scale each model's base number of channels. The models were trained on the CIFAR-10 to the full convergence. We use the 95% threshold of probing accuracy to estimate the tunnel beginning.

Results in Table 5.3.1 indicate that the tunnel length increases as the width of the network grows, implying that representations are formed using fewer layers. However, this trend does not hold for ResNet34, as the longest tunnel is observed with the base width of the network. In the case of VGGs, the number of layers in the network does not affect the number of layers required to form representations. This aligns with the results in Figure 5.3.5.

model	dataset	30%	50%	100%
VGG-19	CIFAR-10	6 (32%)	7 (37%)	7 (37%)
	CIFAR-100	8 (42%)	8 (42%)	9 (47%)
	CINIC-10	6 (32%)	7 (37%)	7 (37%)
ResNet34	CIFAR-10	19 (56%)	19 (56%)	21 (61%)
	CIFAR-100	30 (88%)	30 (88%)	31 (91%)
	CINIC-10	9 (27%)	9 (27%)	17 (50%)

Table 5.3.2. Networks trained on tasks with fewer classes utilize fewer resources for building representations and exhibit longer tunnels. Column headings describe the size of the class subset used in training. Within the (architecture, dataset) pair, the number of gradient steps during training in all cases was the same. We use the 95% threshold of probing accuracy to estimate the tunnel beginning.

The results presented above were obtained from a dataset with a consistent level of complexity. The data in Table 5.3.2 demonstrates that the number of classes in the dataset directly affects the length of the tunnel. Specifically, even though the CINIC-10 training dataset is three times larger than CIFAR-10, the tunnel length remains the same for both datasets. This suggests that the number of samples in the dataset does not impact the length of the tunnel. In contrast, when examining CIFAR-100 subsets, the tunnel length for both VGGs and ResNets increase. This indicates a clear relationship between the dataset's number of classes and the tunnel's length.

Takeaway Deeper or wider networks result in longer tunnels. Networks trained on datasets with fewer classes have longer tunnels.

5.4. The tunnel effect under data distribution shift

Based on the findings from the previous section and the tunnel's negative impact on transfer learning, we investigate the dynamics of the tunnel in continual learning scenarios, where large models are often used on smaller tasks typically containing only a few classes. We focus on understanding the impact of the tunnel effect on transfer learning and catastrophic forgetting [33]. Specifically, we examine how the tunnel and extractor are altered after training on a new task.

5.4.1. Exploring the effects of task incremental learning on extractor and tunnel

Motivation In this section, we aim to understand the tunnel and extractor dynamics in continual learning. Specifically, we examine whether the extractor and the tunnel are equally prone to catastrophic forgetting.

Experiments We train a VGG-19 on two tasks from CIFAR-10. Each task consists of 5 classes from the dataset. We subsequently train on the first and second tasks and save the corresponding extractors E_t and tunnels T_t, where $t \in \{1,2\}$ is the task number. We also save a separate classifying head for trained on each task, that we use during evaluation.

Results As presented in Table 5.4.1, in any combination changing T_1 to T_2 or vice versa have a marginal impact on the performance. This is quite remarkable, and suggests that the tunnel is not specific to the training task. It seems that it *compresses the representations in a task-agnostic way*. The extractor part, on the other hand, is *task-specific* and prone to forgetting as visible in the first four rows of Table 5.4.1. In the last two rows, we present two experiments that investigate how the existence of a tunnel affects the possibility of recovering from this catastrophic forgetting. In the first one, referred to as $(E_2 + T_1(FT))$, we use original data from Task 1 to retrain a classifying head attached on top of extractor E_2 and the tunnel T_1. As visible, it has minimal effect on the accuracy of the first task. In the second experiment, we attach a linear probe directly to the extractor representations $(E_2(FT))$. This difference hints at a detrimental effect of the tunnel on representations' usability in continual learning.

	First Task	Second Task
$E_1 + T_1$	92.04%	56.8%
$E_1 + T_2$	92.5%	58.04 %
$E_2 + T_2$	50.84 %	93.94 %
$E_2 + T_1$	50.66 %	93.72 %
$E_2 + T_1(FT)$	56.1%	–
$E_2(FT)$	74.4%	–

Table 5.4.1. The tunnel part is task-agnostic and can be freely mixed with different extractors retaining the original performance. We test the model's performance on the first or second task using a combination of extractor E_t and tunnel T_t from tasks $t \in \{1,2\}$. The last two rows (FT) show how much performance can be recovered by retraining the linear probe attached to the penultimate layer $E_1 + T_1$ or the last layer of the E_2.

In Appendix 5.11.1 we study this effect further by training a tunnels on two tasks with a different number of classes, where $n_1 > n_2$. In this scenario, we observe that tunnel trained with more classes (T_1) maintains the performance on both tasks, contrary to the tunnel (T_2) that performs poorly on Task 1.

This is in line with our previous observations in Section 5.2.2, that the tunnel compresses to the effective number of classes.

These results present a novel perspective in the ongoing debate regarding the layers responsible for causing forgetting. However, they do not align with the observations made in the previous study [46]. In Appendix 5.11, we delve into the origin of this discrepancy and provide a comprehensive analysis of the changes in representations with a setup introduced with this experiment and the CKA similarity.

Takeaway The tunnel's task-agnostic compression of representations provides immunity against catastrophic forgetting when the number of classes is equal. These findings offer fresh perspectives on studying catastrophic forgetting at specific layers, broadening the current understanding in the literature.

5.4.2. Reducing catastrophic forgetting by adjusting network depth

Motivation Experiments from this section verify whether it is possible to retain the performance of the original model by training a shorter version of the network. A shallower model should also exhibit less forgetting in sequential training.

Experiments We train VGG-19 networks with different numbers of convolutional layers. Each network is trained on two tasks from CIFAR-10. Each task consists of 5 classes from the dataset.

Results: The results shown in Figure 5.4.1 indicate that training shorter networks yields similar performance compared to the original model. However, performance differences become apparent when the network becomes shorter than the extractor part in the original model. This observation aligns with previous findings suggesting that the model requires a certain capacity to perform the task effectively. Additionally, the shorter models exhibit significantly less forgetting, which corroborates the conclusions drawn in previous works [142, 143] on the importance of network depth and architecture in relation to forgetting.

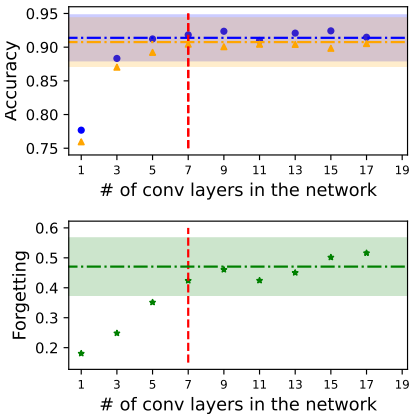


Figure 5.4.1. Training shorter networks from scratch gives a similar performance to the longer counterparts (top) and results in significantly lower forgetting (bottom). The horizontal lines denote the original model's performance. Top image: blue depicts accuracy on first task, orange depicts accuracy on the second task.

Takeaway It is possible to train shallower networks that retain the performance of the original networks and experience significantly less forgetting. However, the shorter networks need to have at least the same capacity as the extractor part of the original network.

5.5. Limitations and future work

This paper empirically investigates the tunnel effect, opening the door for future theoretical research on tunnel dynamics. Further exploration could involve mitigating the tunnel effect through techniques like adjusting learning rates for specific layers. One limitation of our work is its validation within a specific scenario (image classification), while further studies on unsupervised or self-supervised methods with other modalities would shed more light and verify the pertinence of the tunnel elsewhere.

In the experiments, we observed that ResNet-based networks exhibited shorter tunnels than plain MLPs or VGGs. This finding raises the question of whether the presence of skip connections plays a role in tunnel formation. In Appendix 5.15, we take the first step toward a deeper understanding of this relationship by examining the emergence of tunnels in ResNets without skip connections.

5.6. Related work

The analysis of representations in neural network training is an established field [102, 144, 145]. Previous studies have explored training dynamics and the impact of model width [146, 147, 148, 149, 150, 151], but there is still a gap in understanding training dynamics [152, 46, 102, 153]. Works have investigated different architectures' impact on continual learning [154, 143] and linear models' behavior [155, 156, 157, 158]. Our work builds upon studies examining specific layers' role in model performance [135, 130, 147, 152, 139, 159] and sheds light on the origins of observed behaviors [129, 160, 161, 162].

Previous works have explored the role of specific layers in model performance [135, 130, 147, 152, 139, 159]. While some studies have observed a block structure in neural network representations, their analysis was limited to ResNet architectures and did not consider continual learning scenarios. In our work, we investigate a similar phenomenon, expanding the range of experiments and gaining deeper insights into its origins. On the other hand, visualization of layer

representations indicates that higher layers capture intricate and meaningful features, often formed through combinations of lower-layer features [132]. This phenomenon potentially accounts for the extension of feature extractors for complex tasks. Work [163] builds a theoretical picture that stacked sequence models tend to converge to a fixed state with infinite depth and proposes a method to compute the finite equivalent of such networks. The framework of [163] encompasses previous empirical findings of [164, 165, 166]. Independently, research on pruning methods has highlighted a greater neuron count in pruned final layers than in initial layers [167], which aligns with the tunnel's existence. Furthermore, in [168, 169], authors showed that training neural networks may lead to compressing information contained in consecutive hidden layers.

Yet another work [135] offers a different perspective, where the authors distinguish between critical and robust layers, highlighting the importance of the former for model performance, while individual layers from the latter can be reset without impacting the final performance. Our analysis builds upon this finding and further categorizes these layers into the extractor and tunnel, providing insights into their origins and their effects on model performance and generalization ability.

Our findings are also related to the Neural Collapse (NC) phenomenon [129], which has gained recent attention [160, 161, 162]. Several recent works [170, 171, 172] have extended the observation of NC and explored its impact on different layers, with a notable emphasis on deeper layers. [171] establishes a link between collapsed features and transferability. In our experiments, we delve into tunnel creation, analyzing weight changes and model behavior in a continual learning scenario, revealing the task-agnostic nature of the tunnel layers.

5.7. Conclusions

This work presents new insights into the behavior of deep neural networks during training. We discover the tunnel effect, an intriguing phenomenon in modern deep networks where they split into two distinct parts - the extractor and the tunnel. The extractor part builds representations, and the tunnel part compresses these representations to a minimum rank without contributing to the model's performance. This behavior is prevalent across multiple architectures and is positively correlated with overparameterization, i.e., it can be induced by increasing the model's size or decreasing the complexity of the task.

We emphasise that our motivation for investigating this phenomenon aimed at

building a coherent picture encompassing both our experiments and evidence in the literature. Specifically, we aim to understand better how the neural networks handle the representation-building process in the context of depth.

Additionally, we discuss potential sources of the tunnel and highlight the unintuitive behavior of neural networks during the initial training phase. This novel finding has significant implications for improving the performance and robustness of deep neural networks. Moreover, we demonstrate that the tunnel hinders out-of-distribution generalization and can be detrimental in continual learning settings.

Overall, our work offers new insights into the mechanisms underlying deep neural networks. Building on consequences of the tunnel effect we derive a list of recommendations for practitioners interested in applying deep neural networks to downstream tasks. In particular, focusing on the tunnel entry features is promising when dealing with distribution shift due to its strong performance with OOD data. For continual learning, regularizing the extractor should be enough, as the tunnel part exhibits task-agnostic behavior. Skipping feature replays in deeper layers or opting for a compact model without a tunnel can combat forgetting and enhance knowledge retention. For efficient inference, excluding tunnel layers during prediction substantially cuts computation time while preserving model accuracy, offering a practical solution for resource-constrained situations.

Appendix

5.8. Experimental setup

5.8.1. Architectures and hyperparameters

In this section, we detail the model architectures examined in the experiments and list all hyperparameters used in the experiments.

VGG [173] In the main text use two types of VGG networks, namely VGG-19 and VGG-16. Both architectures consist of five stages, each consisting of a combination of convolutional layers with ReLU activation and max pooling layers. The VGG-19 has 19 layers, including 16 convolutional layers and three fully connected layers. The first two fully connected layers are followed by ReLU activation. On the other hand, VGG-16 has a total of 16 layers, including 13 convolutional layers and three fully connected layers. In additional experiments, we extend our analysis by VGG-11 and VGG-16. The base number of channels in consecutive stages for VGG architectures equals 64, 128, 256, 512, and 512.

ResNet [174] In experiments, we utilize two variants of the ResNet family of architectures, i.e., ResNet-18 and ResNet-34. ResNet-N is a five-staged network characterized by depth, with a total of N layers. The initial stage consists of a single convolutional layer – with kernel size 7×7 and 64 channels and ReLU activation, followed by max pooling 2×2, which reduces the spatial dimensions. The subsequent stages are composed of residual blocks. Each residual block typically contains two convolutional layers and introduces a shortcut connection that skips one or more layers. Each convolutional layer in the residual block is followed by batch normalization and ReLU activation. The remaining four stages in ResNet-18 and ResNet-34 architectures consist of 3x3 convolutions with the following number of channels: 64, 128, 256, and 512.

MLP [175] An MLP (Multi-Layer Perceptron) network is a feedforward neural network architecture type. It consists of multiple layers of artificial neurons – in our experiments, we consider MLPs with 6,8,10,12 layers with ReLU activations (except last layer, which has linear activation). In our experiments, the underlying architecture has 1024 neurons per layer.

In VGGs, MLPs, and ResNets without skips, we use the 98% threshold to estimate the tunnel for the plots. In the case of ResNets, we use the 95% threshold. In the case of ResNets, we report the results for the 'conv2' layers.

Due to computational constraints, we randomly choose a subset of 8000 features to compute the numerical rank.

Hyperparameters Hyperparameters used for neural network training are presented in the leftmost Table 5.8.1. Each column shows the values of the hyperparameters corresponding to a different architecture. The presented hyperparameters are recommended for the best performance of these models on the CIFAR-10 dataset [176]. However, in experiments focused on continual learning scenario (Section 5.4.2), we refrain from decaying the learning rate and shorten the network's training to 30 epochs to mimic the actual settings used in continual learning settings.

Hyperparameters used for training linear probes in our experiment are presented in the rightmost table. Linear probes were trained with Adam optimizer instead of SGD.

Parameter	VGG	ResNet	MLP
Learning rate (LR)	0.1	0.1	0.05
SGD momentum	0.9	0.9	0.0
Weight decay	10^{-4}	10^{-4}	0
Number of epochs	160	164	1000
Mini-batch size	128	128	128
LR-decay-milestones	80, 120	82, 123	-
LR-decay-gamma	0.1	0.1	0.0

Parameter	Value
Learning rate	0.001
Weight decay	0
Number of epochs	30
Mini-batch size	512

5.8.2. Datasets

In this article, we present the results of experiments conducted on following datasets: **CIFAR-10 [177]** CIFAR-10 is a widely used benchmark dataset in the field of computer vision. It consists of 60,000 color images in 10 different classes, with each class containing 6,000 images. The dataset is divided into 50,000 training images and 10,000 test images. The images in CIFAR-10 have a resolution of 32×32 pixels.

CIFAR-100 [177] CIFAR-100 is a dataset commonly used for image classification tasks in computer vision. It contains 60,000 color images, with 100 different classes, each containing 600 images. The dataset is split into

50,000 training images and 10,000 test images. The images in CIFAR-100 have a resolution of 32×32 pixels. Unlike CIFAR-10, CIFAR-100 offers a higher level of granularity, with more fine-grained categories such as flowers, insects, household items, and various types of animals and vehicles.

CINIC-10 [178] CINIC-10 is a dataset that stands as a 'bridge' between CIFAR-10 and ImageNet for image classification tasks. It combines 60,000 images of CIFAR-10, and 210,000 downsampled images of ImageNet. The images in CINIC-10 have a resolution of 32×32 pixels.

Food-101 [179] The Food-101 dataset is a collection of food images commonly used for image classification tasks. It contains 101 categories of food, with each category consisting of 1,000 images. The dataset covers a wide range of food items from various cuisines, including fruits, vegetables, desserts, and main dishes.

102-Flower [180] The 102-Flower dataset is a collection of images representing 102 different categories of flowers. Each image in the dataset has a fixed resolution of 256 pixels in both width and height. The dataset provides a diverse set of flower images.

The Oxford-IIIT Pet Dataset [181] The Oxford-IIIT Pet Dataset is a collection of images of cats and dogs belonging to 37 different breeds. The dataset includes a total of 7,349 images.

Places-365 [182] The Places-365 dataset is a large-scale dataset consisting of 365 different scene categories. It contains over 1.8 million images, each depicting a specific scene or environment. The images in the dataset have 256x256 pixels.

STL-10 [183] STL-10 dataset is a benchmark image dataset consisting of 10 different classes, including various animals, vehicles, and household objects. It contains a total of 5,000 training images and 8,000 test images, each with a resolution of 96 pixels by 96 pixels. The STL-10 dataset is derived from the larger ImageNet dataset but is specifically designed for low-resolution image classification tasks.

SVHN [184] The SVHN (Street View House Numbers) dataset is a large-scale dataset for digit recognition from real-world images. It consists of labeled images of house numbers captured from Google Street View. The dataset includes over 600,000 images for training and 26,032 images for testing. Each image is RGB and has a resolution of 32 pixels by 32 pixels.

We preprocess all datasets with standardization, additionally we rescale each image to $32px \times 32px$.

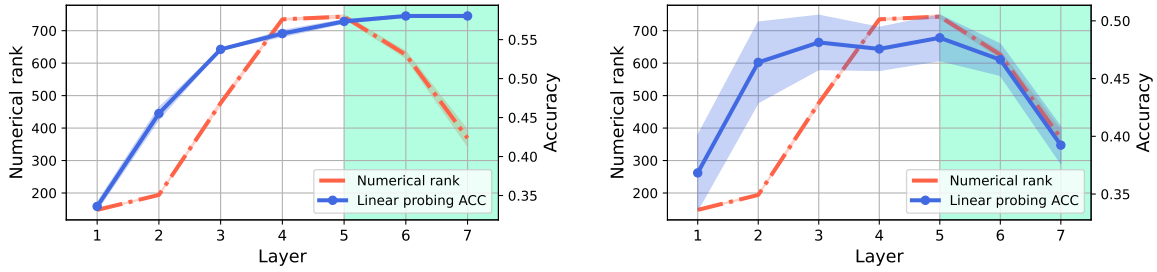
5.8.3. Compute

We conducted approximately 300 experiments to finalize our work, each taking about three wall-clock hours on a single NVIDIA A5000 GPU. We had access to a server with eight NVIDIA A5000 GPUs, enabling us to parallelize our experiments and reduce total computation time. We estimate to perform over 2000 experiments (including failed ones) during the development phase of the project.

5.9. Full results

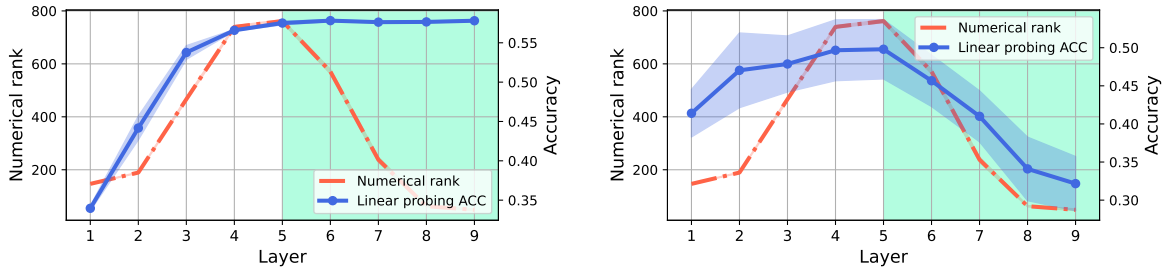
5.9.1. MLPs

In this section, we present the results of the tunnel effect for MLP architectures with different depths. All models are trained on CIFAR-10, and their OOD properties are evaluated on ten randomly selected classes of CIFAR-100.



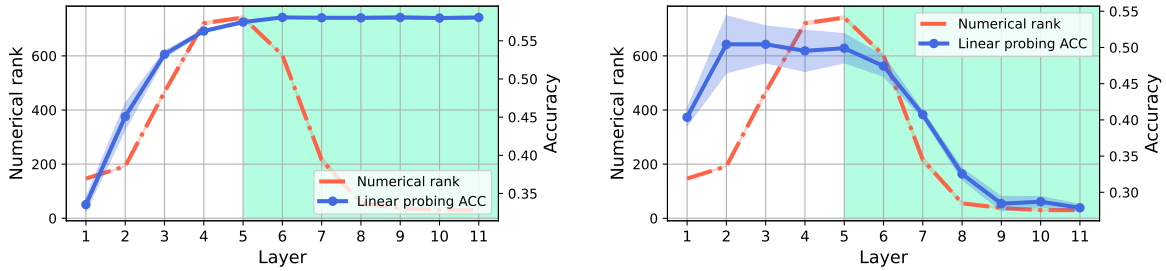
(a) In distribution **(b)** Out of distribution

Figure 5.9.1. In and out of distribution linear probing performance for MLP-6 trained on CIFAR-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed with random 10 class subsets of CIFAR-100.



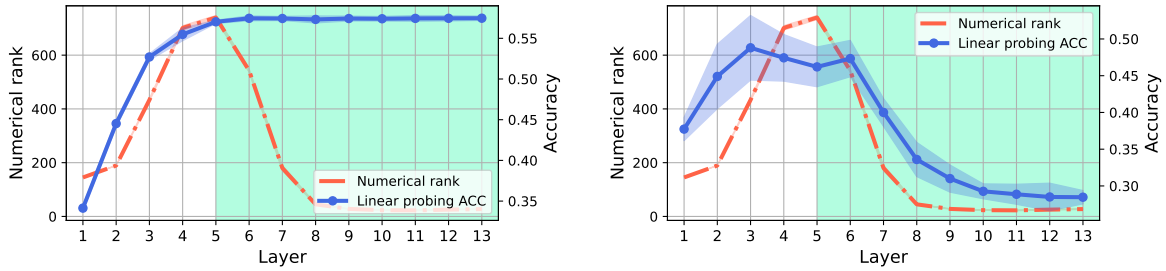
(a) In distribution **(b)** Out of distribution

Figure 5.9.2. In and out of distribution linear probing performance for MLP-8 trained on CIFAR-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed with random 10 class subsets of CIFAR-100.



(a) In distribution **(b)** Out of distribution

Figure 5.9.3. In and out of distribution linear probing performance for MLP-10 trained on CIFAR-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed with random 10 class subsets of CIFAR-100.



(a) In distribution **(b)** Out of distribution

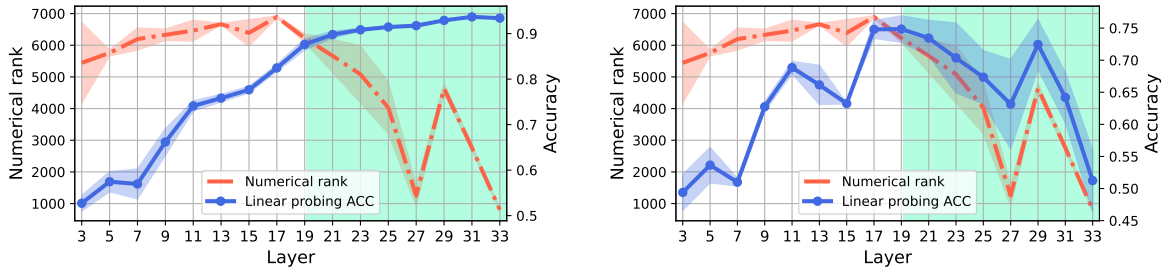
Figure 5.9.4. In and out of distribution linear probing performance for MLP-12 trained on CIFAR-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed with random 10 class subsets of CIFAR-100.

5.9.2. ResNet-34

In this section, we present the results of the tunnel effect for ResNet architectures with different depths. All models are trained on datasets CIFAR-10, CIFAR-100, and CINIC-10.

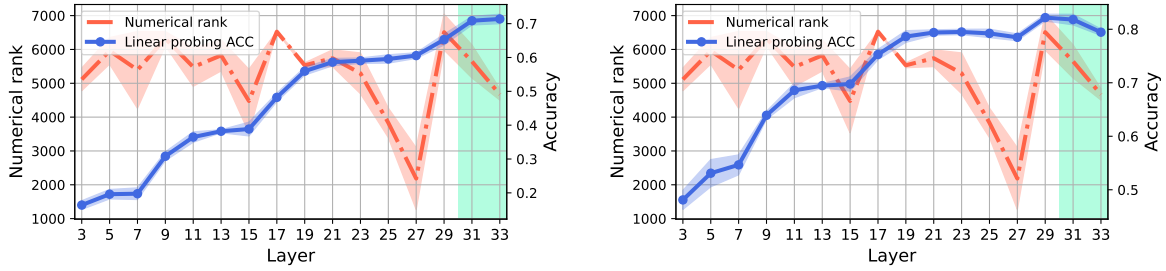
VGG-19

In this section, we present the results of the tunnel effect for VGG architectures with different depths. All models are trained on datasets CIFAR-10, CIFAR-100, and CINIC-10.



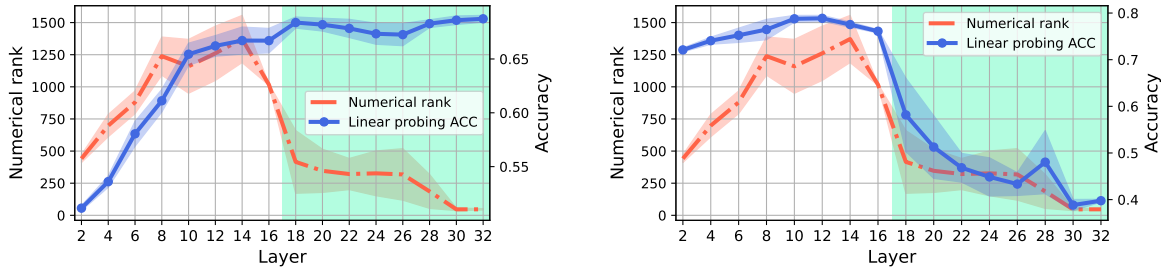
(a) In distribution **(b)** Out of distribution

Figure 5.9.5. In and out of distribution linear probing performance for ResNet-34 trained on CIFAR-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed with random 10 class subsets of CIFAR-100.



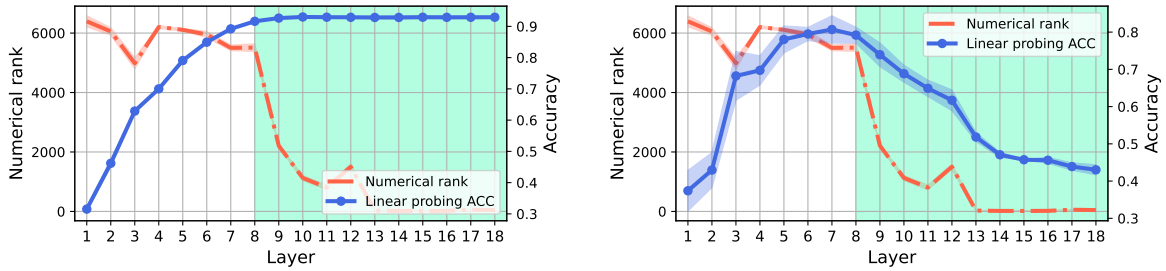
(a) In distribution **(b)** Out of distribution

Figure 5.9.6. In and out of distribution linear probing performance for ResNet-34 trained on CIFAR-100. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed on CIFAR-10.



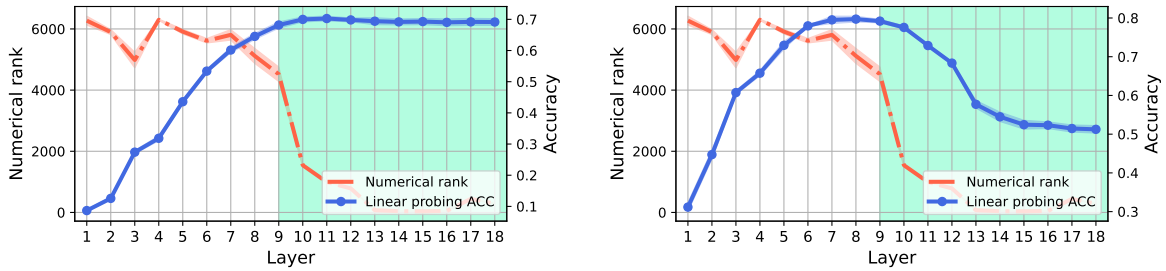
(a) In distribution **(b)** Out of distribution

Figure 5.9.7. In and out of distribution linear probing performance for ResNet-34 trained on CINIC-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed on subset of ten classes from CIFAR-100.



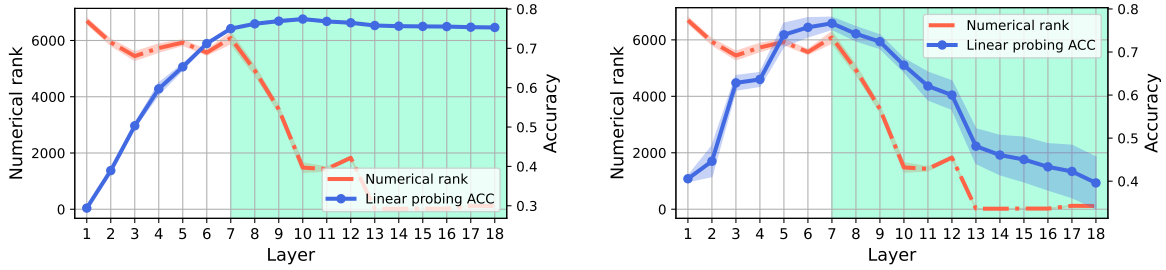
(a) In distribution **(b)** Out of distribution

Figure 5.9.8. In and out of distribution linear probing performance for VGG-19 trained on CIFAR-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed with random 10 class subsets of CIFAR-100.



(a) In distribution **(b)** Out of distribution

Figure 5.9.9. In and out of distribution linear probing performance for VGG-19 trained on CIFAR-100. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed on CIFAR-10.

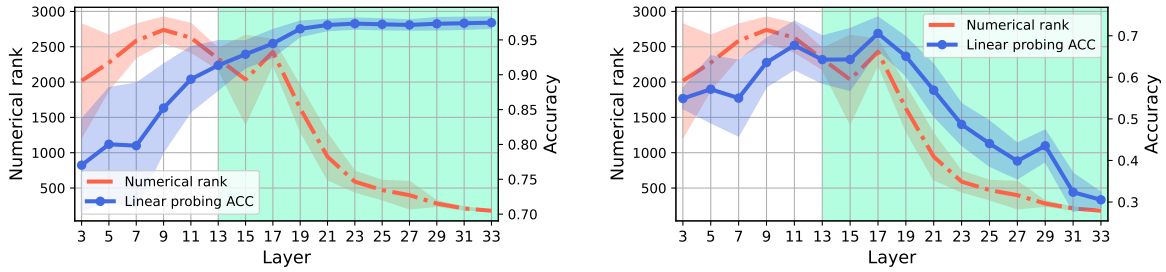


(a) In distribution **(b)** Out of distribution

Figure 5.9.10. In and out of distribution linear probing performance for VGG-19 trained on CINIC-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed on subset of ten classes from CIFAR-100.

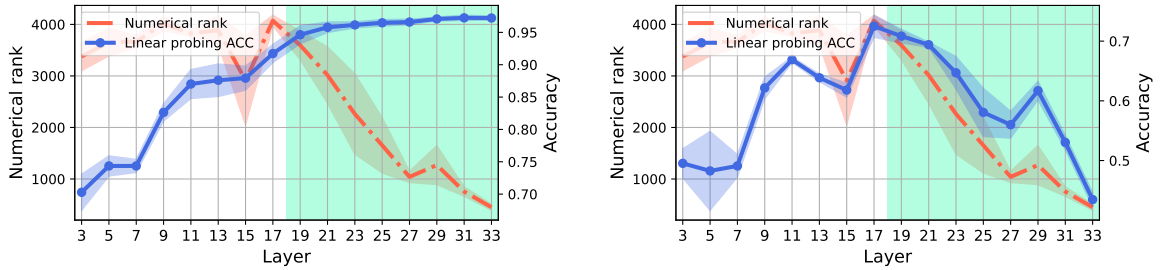
5.9.3. Dataset complexity experiments

ResNet-34



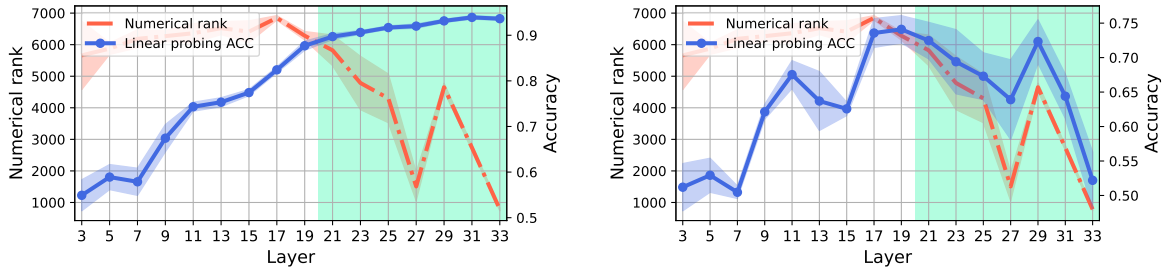
(a) In distribution **(b)** Out of distribution

Figure 5.9.11. In and out of distribution linear probing performance for ResNet-34 trained on a 3-class subset of CIFAR-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank, and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed with random 10 class subsets of CIFAR-100.



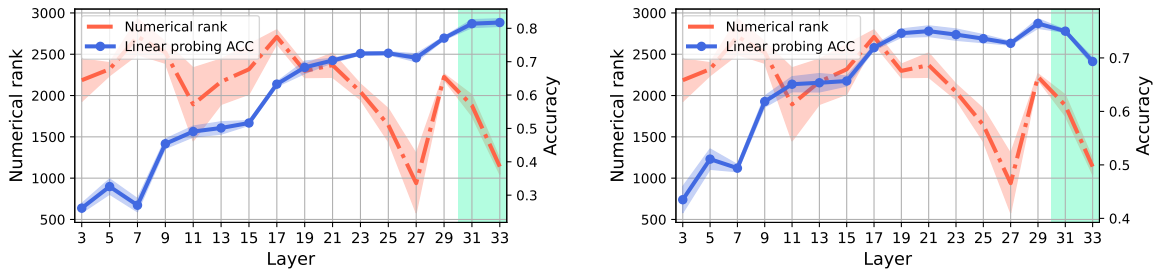
(a) In distribution **(b)** Out of distribution

Figure 5.9.12. In and out of distribution linear probing performance for ResNet-34 trained on a 5-class subset of CIFAR-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank, and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed with random 10 class subsets of CIFAR-100.



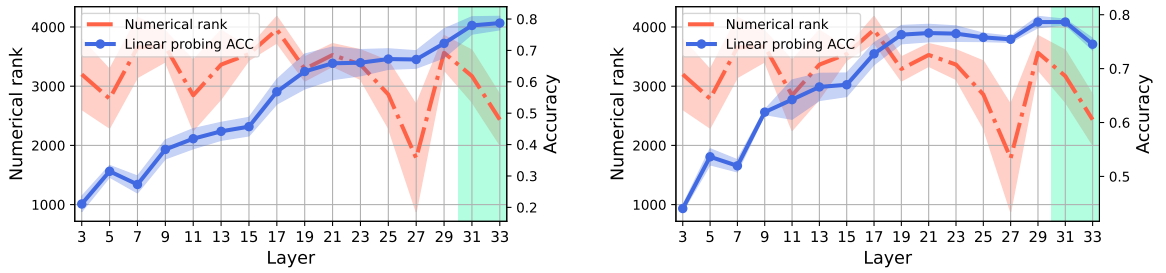
(a) In distribution **(b)** Out of distribution

Figure 5.9.13. In and out of distribution linear probing performance for ResNet-34 trained on CIFAR-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank, and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed with random 10 class subsets of CIFAR-100.



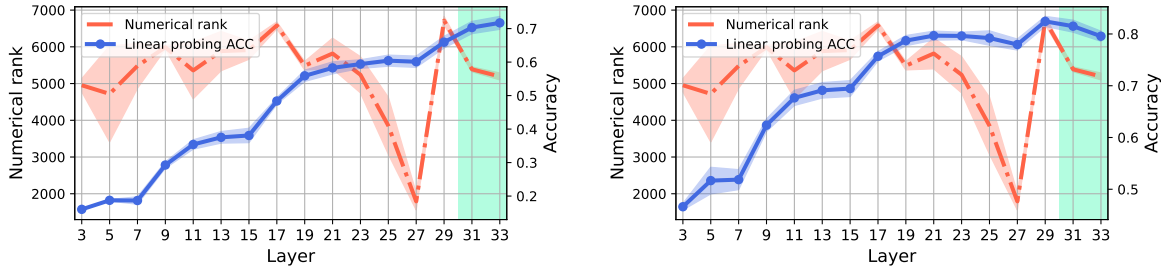
(a) In distribution **(b)** Out of distribution

Figure 5.9.14. In and out of distribution linear probing performance for ResNet-34 trained on a 30-class subset of CIFAR-100. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank, and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed on CIFAR-10.



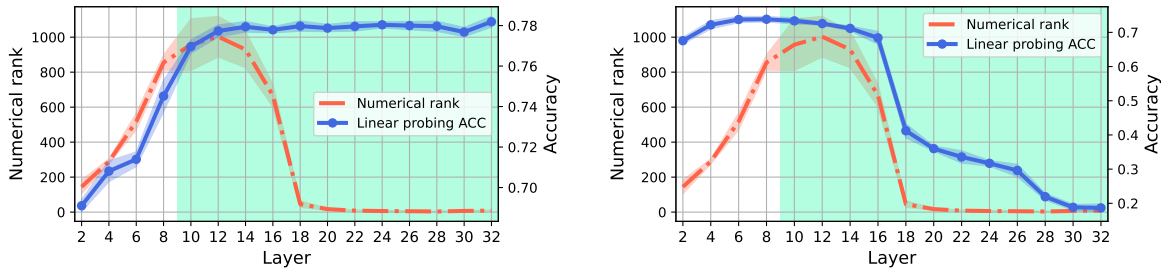
(a) In distribution **(b)** Out of distribution

Figure 5.9.15. In and out of distribution linear probing performance for ResNet-34 trained on a 50-class subset of CIFAR-100. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank, and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed on CIFAR-10.



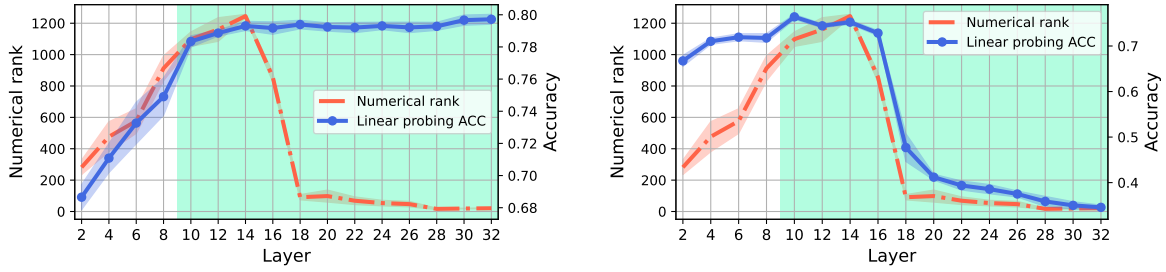
(a) In distribution **(b)** Out of distribution

Figure 5.9.16. In and out of distribution linear probing performance for ResNet-34 trained on CIFAR-100. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank, and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed on CIFAR-10.



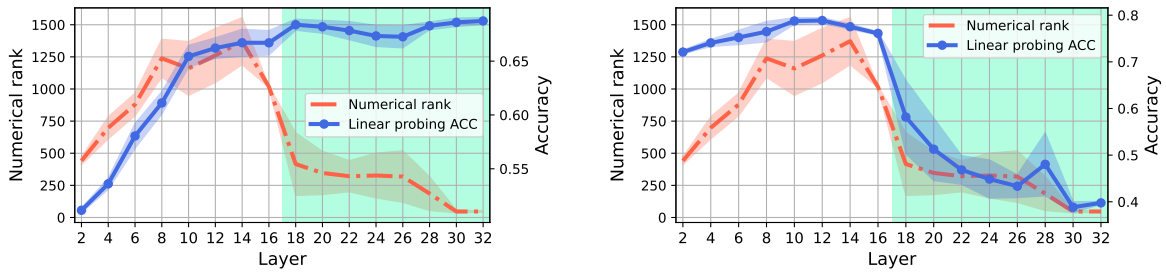
(a) In distribution **(b)** Out of distribution

Figure 5.9.17. In and out of distribution linear probing performance for ResNet-34 trained on a 3-class subset of CINIC-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank, and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed on CIFAR-10.



(a) In distribution **(b)** Out of distribution

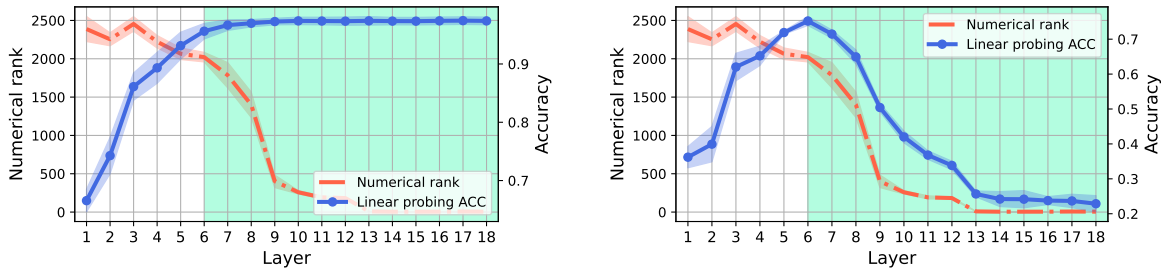
Figure 5.9.18. In and out of distribution linear probing performance for ResNet-34 trained on a 5-class subset of CINIC-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank, and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed on CIFAR-10.



(a) In distribution **(b)** Out of distribution

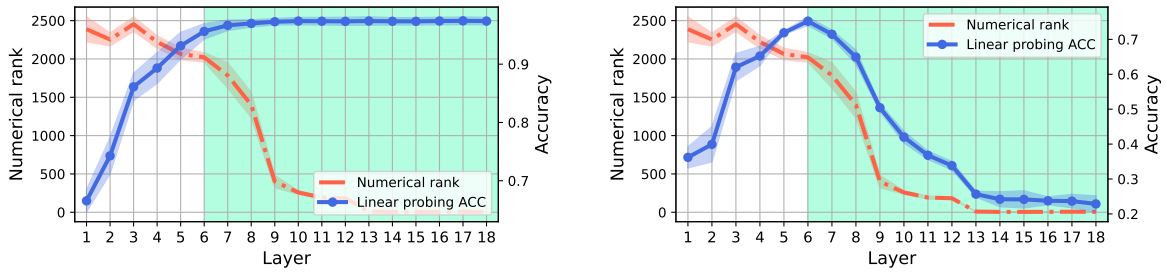
Figure 5.9.19. In and out of distribution linear probing performance for ResNet-34 trained on CINIC-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank, and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed on CIFAR-10.

VGG-19



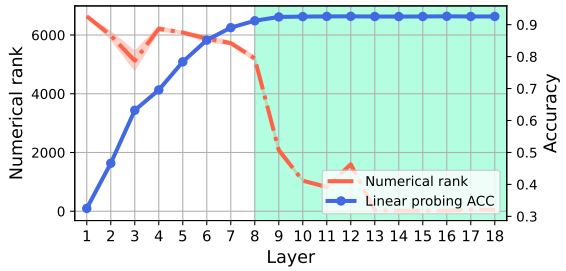
(a) In distribution **(b)** Out of distribution

Figure 5.9.20. In and out of distribution linear probing performance for VGG-19 trained on a 3-class subset of CIFAR-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank, and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed with random 10 class subsets of CIFAR-100.

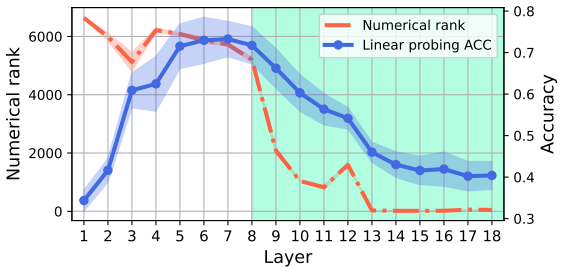


(a) In distribution **(b)** Out of distribution

Figure 5.9.21. In and out of distribution linear probing performance for VGG-19 trained on a 5-class subset of CIFAR-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank, and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed with random 10 class subsets of CIFAR-100.

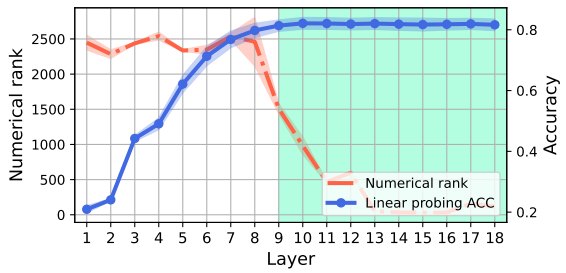


(a) In distribution

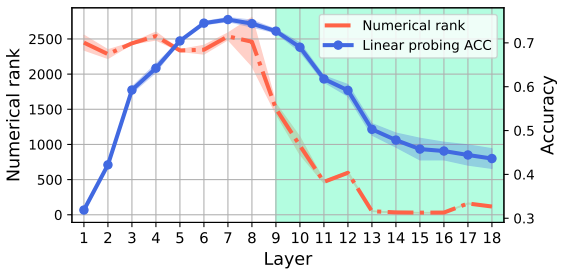


(b) Out of distribution

Figure 5.9.22. In and out of distribution linear probing performance for VGG-19 trained on CIFAR-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank, and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed with random 10 class subsets of CIFAR-100.

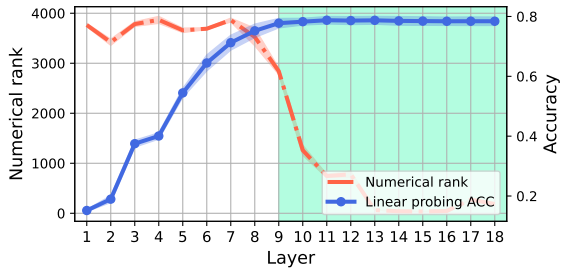


(a) In distribution

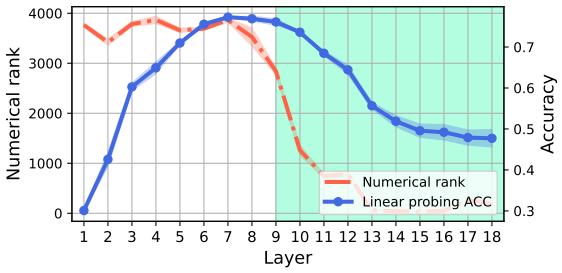


(b) Out of distribution

Figure 5.9.23. In and out of distribution linear probing performance for VGG-19 trained on a 30-class subset of CIFAR-100. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank, and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed on CIFAR-10.

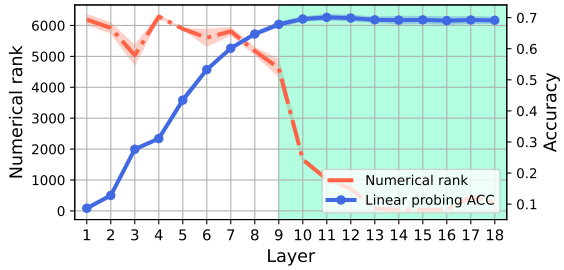


(a) In distribution

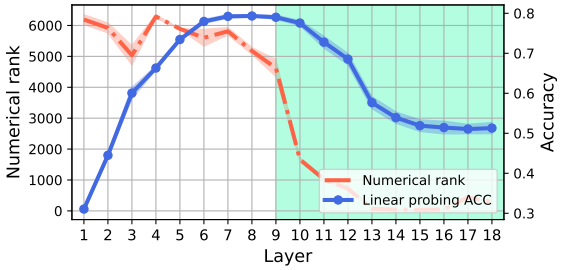


(b) Out of distribution

Figure 5.9.24. In and out of distribution linear probing performance for VGG-19 trained on a 50-class subset of CIFAR-100. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank, and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed on CIFAR-10.

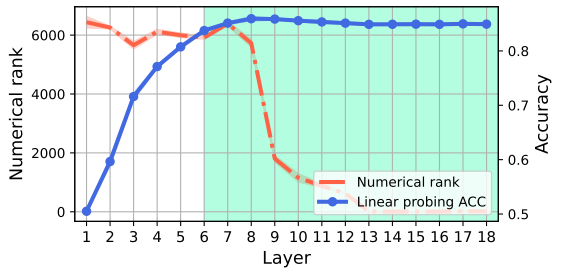


(a) In distribution

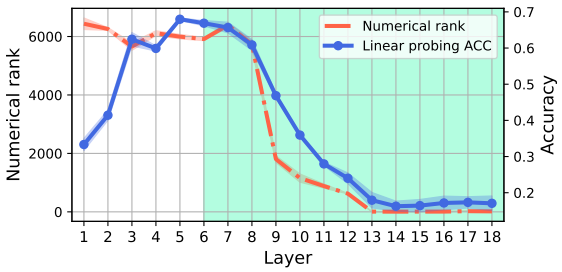


(b) Out of distribution

Figure 5.9.25. In and out of distribution linear probing performance for VGG-19 trained on CIFAR-100. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank, and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed on CIFAR-10.

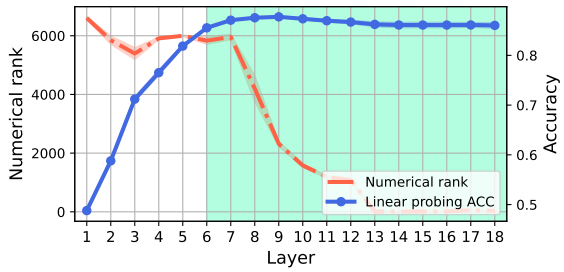


(a) In distribution

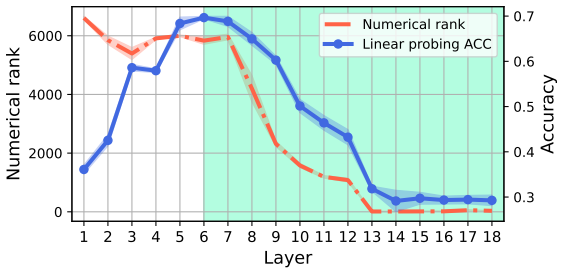


(b) Out of distribution

Figure 5.9.26. In and out of distribution linear probing performance for VGG-19 trained on a 3-class subset of CINIC-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank, and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed on CIFAR-10.

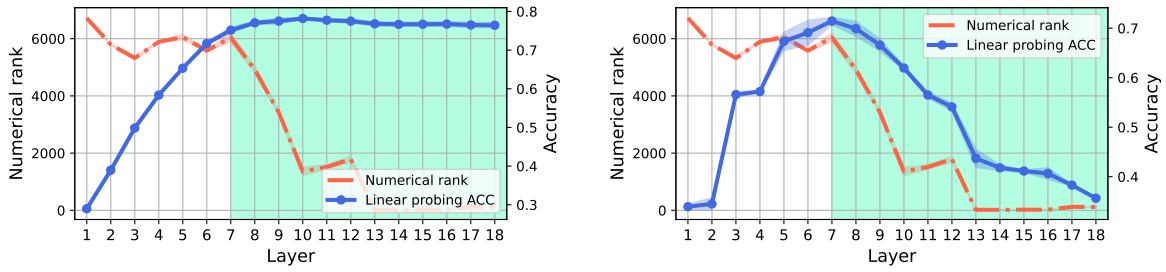


(a) In distribution



(b) Out of distribution

Figure 5.9.27. In and out of distribution linear probing performance for VGG-19 trained on a 5-class subset of CINIC-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank, and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed on CIFAR-10.

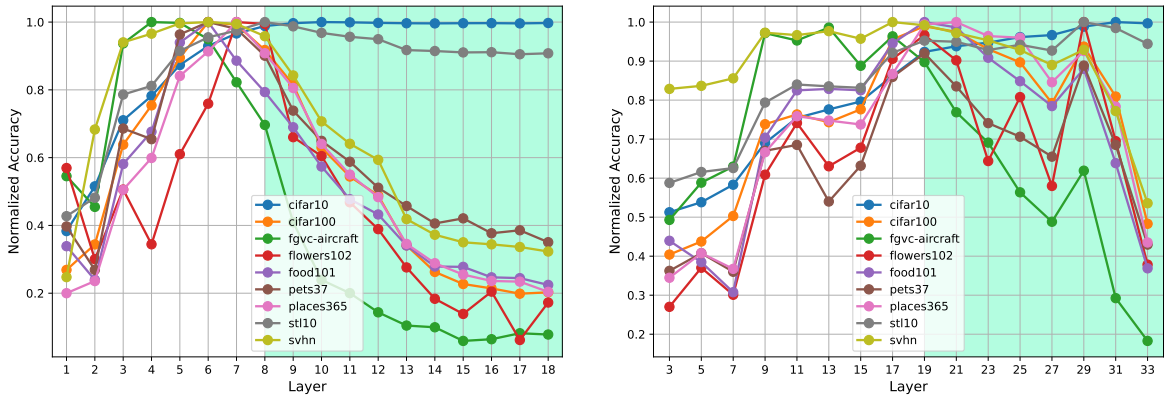


(a) In distribution **(b)** Out of distribution

Figure 5.9.28. In and out of distribution linear probing performance for VGG-19 trained on CINIC-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank, and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed on CIFAR-10.

5.10. Out of distribution generalization - extended results

In this experiment, we aim to determine if the tunnel consistently decreases the performance of models on out-of-distribution (OOD) datasets. To achieve this, we trained VGG-19 and ResNet34 models on CIFAR-10 and conducted linear probing on various OOD datasets. The results, depicted in Figure 5.10.1, are consistent across both the tested models and the datasets used. Notably, in all cases except for the training dataset (CIFAR-10), we observe a decline in performance starting from the beginning of the tunnel and continuing to degrade further. In the case of ResNet-34, there is a spike in performance at the 29^{th} layer, which aligns with the findings in the main paper. Interestingly, the dataset that exhibits the least deterioration is STL-10. This dataset consists of 10 classes, 9 of which overlap with classes found in CIFAR-10. However, the images in STL-10 are sampled from the ImageNet dataset. These results suggest that models can generalize well to OOD data that share semantic similarities with the in-distribution data. Note that the linear probing performance was normalized for better presentation.



(a) VGG-19 **(b)** ResNet-34

Figure 5.10.1. Out of distribution normalized linear probing performance for different datasets. The shaded area depicts the tunnel, different colors depict the linear probing performance on given dataset. Note that all the results are normalized for clarity of presentation.

The following experiment complements the analysis presented in the main paper, aiming to further explore the degradation of out-of-distribution performance caused by the tunnel effect. In this particular setup, the network is trained using CIFAR-10, and linear probes are trained and evaluated using subsets of CIFAR-100 with varying numbers of classes. The results, depicted in Figure 5.10.2, consistently demonstrate that regardless of the number of classes used to train the linear probes, the tunnel effect consistently leads to a decline in their performance. These findings confirm our observations from the main paper, indicating that the tunnel effect is a prevalent characteristic of the model rather than a peculiar artifact of the dataset or training setup.

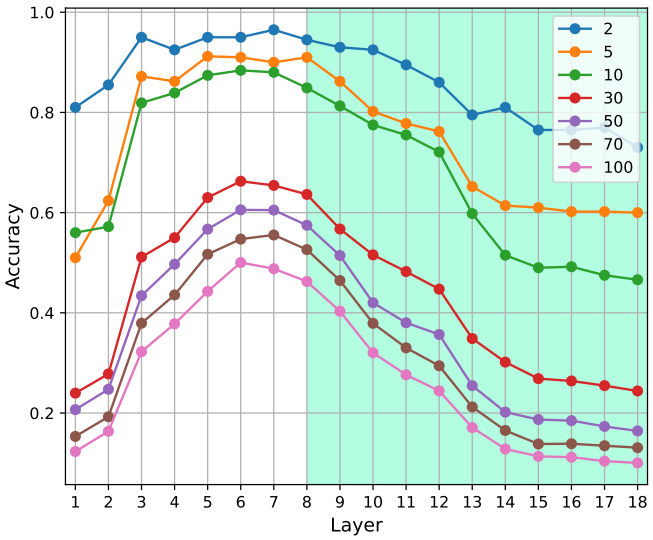


Figure 5.10.2. Source task with a fixed number of classes results in a tunnel consistently degrading the OOD performance for a different number of classes. VGG-19 is trained on CIFAR-10 and linear probes are trained on different subsets of CIFAR-100 with different numbers of classes. The tunnel is marked with a shaded color.

5.11. Exploring the effects of task incremental learning on extractor and tunnel – extended results

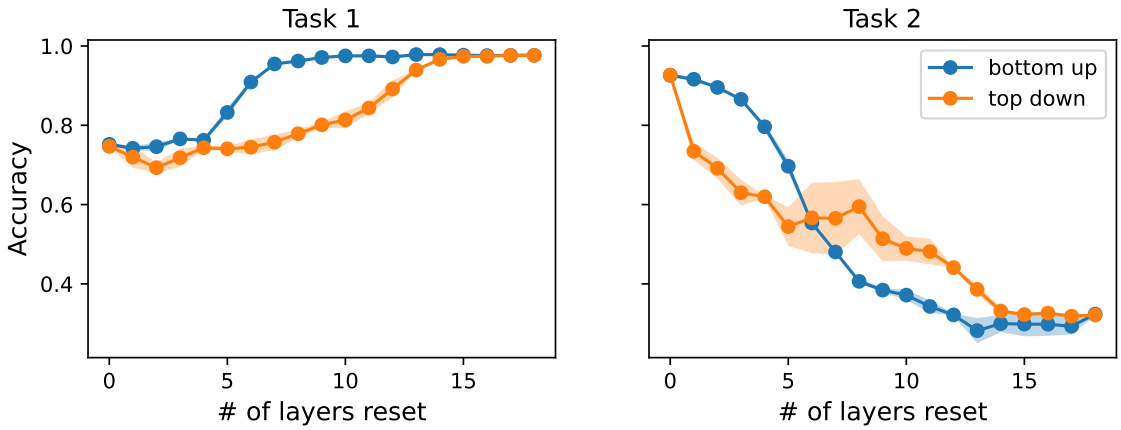


Figure 5.11.1. Substituting layer experiment. VGG-19 trained on the sequence of two tasks on split-CIFAR10. First task 3 class, second task 7 classes.

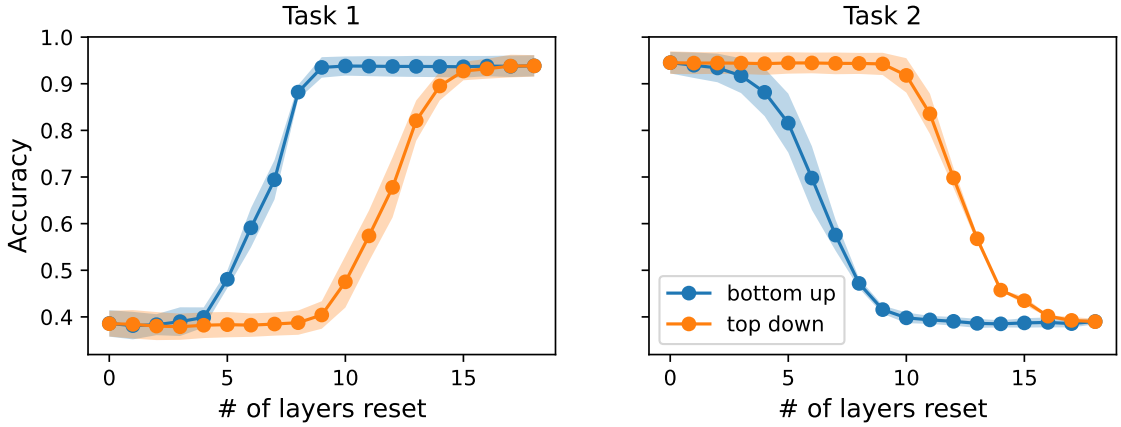


Figure 5.11.2. Substituting layer experiment. VGG-19 trained on the sequence of two tasks on split-CIFAR10. First task 5 class, second task 5 classes.

In this section we discuss in greater details experiment from section 5.4.1. First, we focus on examining reset layers experiment in case of sequence of tasks with different number of classes in section 5.11.1. Next, we discuss the the discrepancies between our results and results presented in work [46].

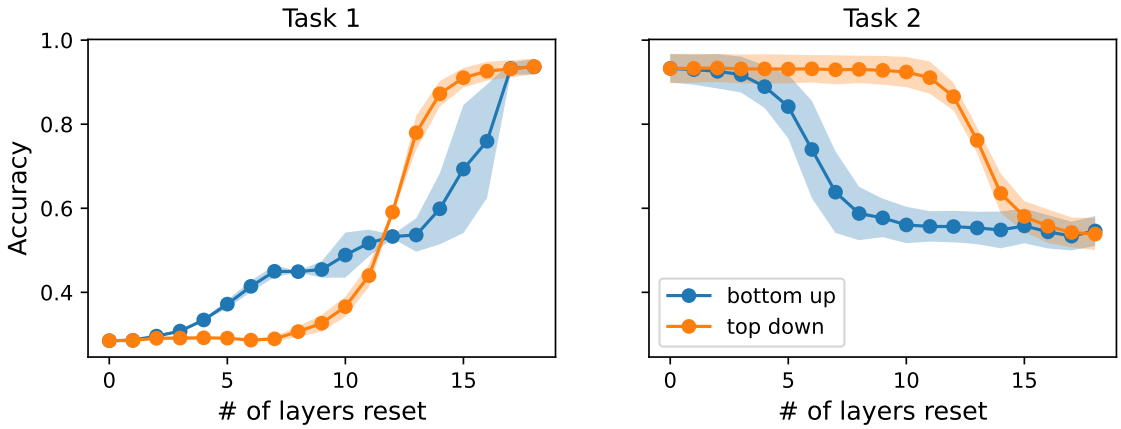


Figure 5.11.3. Substituting layer experiment. VGG-19 trained on the sequence of two tasks on split-CIFAR10. First task 7 class, second task 3 classes.

5.11.1. Different number of classes in source and target tasks.

In this experiment, our aim is to gain a better understanding of tunnel immunity to catastrophic forgetting. Specifically, we are interested in exploring scenarios where the number of classes differs in each task. To analyze this scenario, we conducted three experiments using the VGG-19 network. We trained the network on sequences of two tasks, each composed of CIFAR-10 classes with different splits: (3,7), (5,5), and (7,3).

During training, we saved the model after completing the first and second tasks, denoted as M_1 and M_2 respectively. When we refer to $M_1^{1:x} + M_2^{x+1:n}$, we mean that the network consists of the first x layers with parameters from after completing the first task, combined with the remaining $n-x$ layers from the network after completing the second task.

Here, instead of a table we present the results using plots, see Figure 5.11.1 for the reference. The y-axis values represent the accuracy of the model when substituting a certain number of layers, denoted as x. The blue plot represents the situation where we substitute layers starting from the bottom ($M_1^{1:x} + M_2^{x+1:n}$), while the orange plot represents the opposite scenario ($M_2^{1:x} + M_1^{x+1:n}$). Please note the change in subscripts.

In Figure 5.11.2, we observe that when the tasks have an equal number of classes, the tunnel is preserved perfectly. Specifically, substituting 10 layers from the top down does not affect the performance on the second task, and substituting more than 8 layers does not yield any improvement on the first task.

Conversely, in Figure 5.11.1, substituting more than 7 layers from the bottom up does not lead to any improvement in the second task. Additionally,

substituting any layers from the top down actually harms the performance on the second task. This suggests that while the network encountered more classes in the second task, it built upon the existing tunnel, maintaining its performance on the first task.

In the opposite scenario, where the second task involves fewer classes, a reverse situation is observed. Substituting any layers from the top down negatively impacts the performance on the first task, while substituting 10 layers from the top down does not affect the performance on the second task. This suggests that the network successfully reused a portion of the tunnel from the first task while discarding the unnecessary part.

5.11.2. On the primary source of catastrophic forgetting on split-CIFAR10 task.

There is an ongoing discussion surrounding the layers responsible for driving the phenomenon of forgetting. In a study [46], authors claim that "Higher layers are the primary source of catastrophic forgetting on split CIFAR-10 task." However, our findings present a different perspective compared to the conclusions drawn in that research. Specifically, the results presented in Section 5.4.1 and Section 5.11.1 indicate that there exist continual learning scenarios where the deeper layers do not contribute to catastrophic forgetting. Instead, we show that in certain scenarios the earlier layers are responsible for performance degradation, while the deeper layers remain unaffected due to their task-agnostic nature. This insight is of particular significance because many studies have built upon the assumption that mainly deeper layers are responsible for catastrophic forgetting, potentially leading to inadequate or inefficient continual learning mechanisms [185, 186].

It is important to note that the tunnel hypothesis effect holds for overparameterized networks. In contrast, the authors of [46] evaluated their claims using the VGG-13 network, with the width of the layers reduced by a factor of four. This discrepancy plays a crucial role in tunnel formation, as it reduces the model's capacity. Figures 5.11.4- 5.11.9 illustrate the disparity between these models in the reset experiment.

From this comparison, the main conclusion emerges that the question of "which layers are the primary source of catastrophic forgetting?" is nuanced and contingent upon multiple factors.

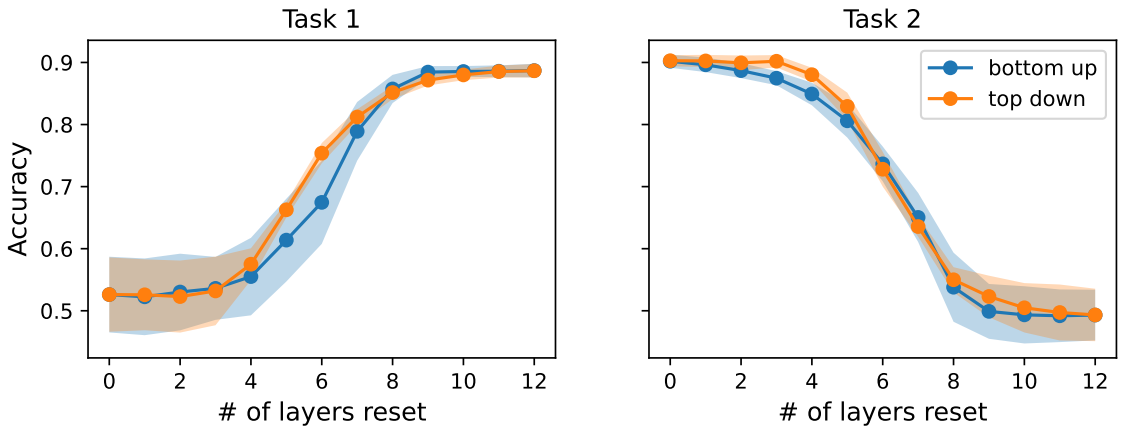


Figure 5.11.4. Substituting layer experiment. VGG-13, width factor = 0.25, trained on the sequence of two tasks on split-CIFAR10.

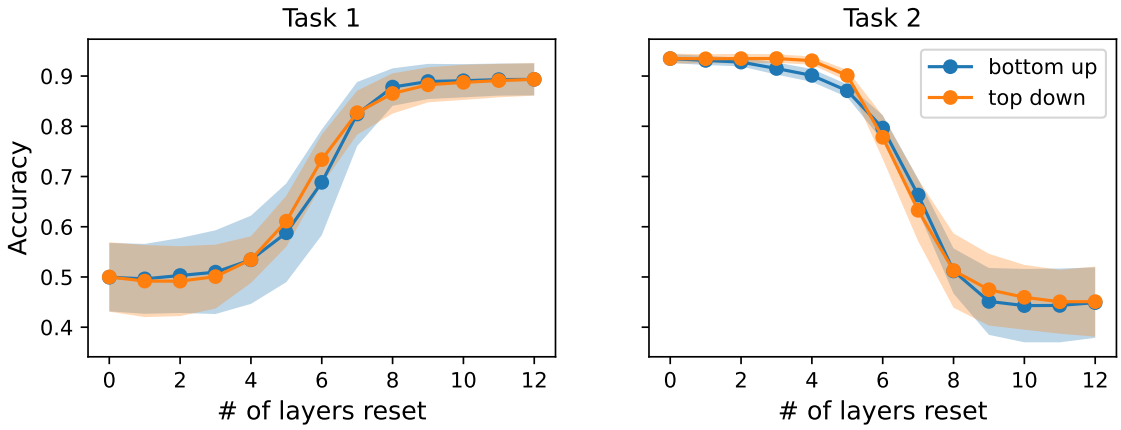


Figure 5.11.5. Substituting layer experiment. VGG-13, width factor = 1, trained on the sequence of two tasks on split-CIFAR10.

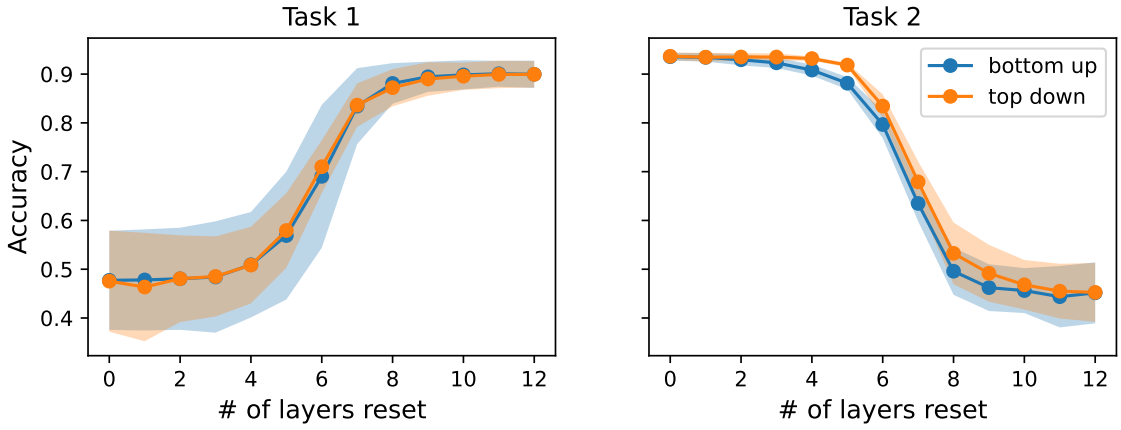


Figure 5.11.6. Substituting layer experiment. VGG-13, width factor = 2, trained on the sequence of two tasks on split-CIFAR10.

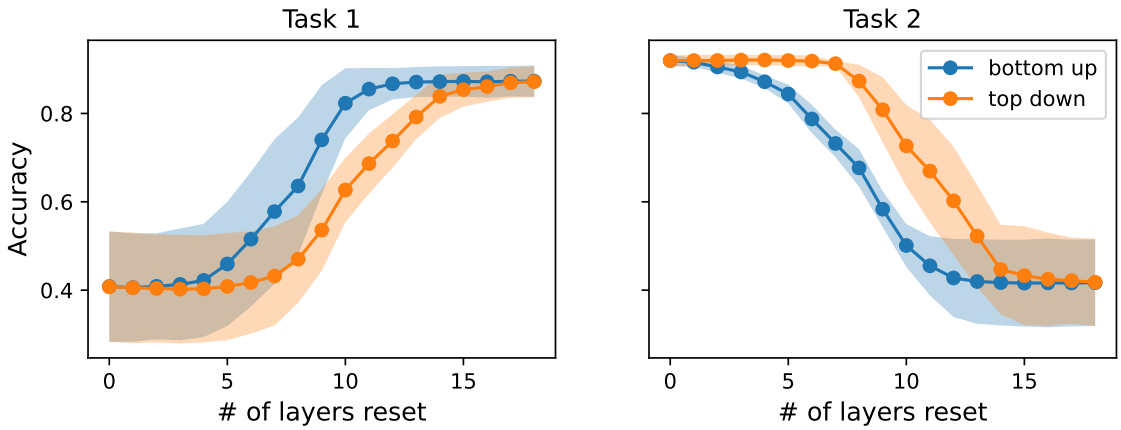


Figure 5.11.7. Substituting layer experiment. VGG-19, width factor = 0.25, trained on the sequence of two tasks on split-CIFAR10.

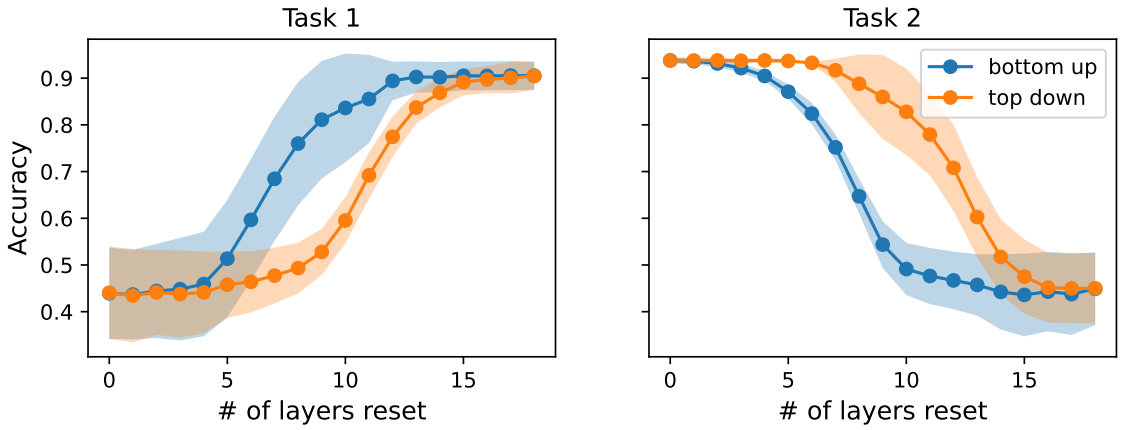


Figure 5.11.8. Substituting layer experiment. VGG-19, width factor = 1, trained on the sequence of two tasks on split-CIFAR10.

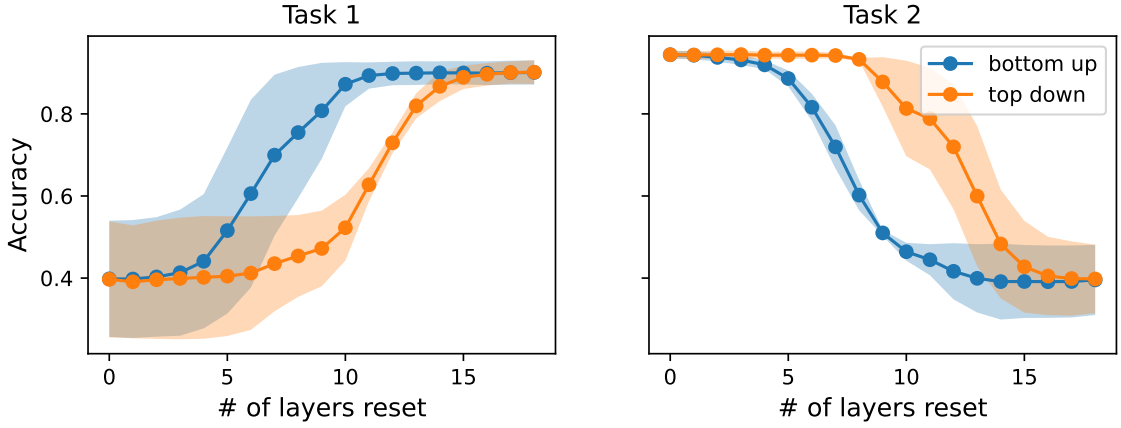


Figure 5.11.9. Substituting layer experiment. VGG-19, width factor = 2, trained on the sequence of two tasks on split-CIFAR10.

5.12. CKA similarity

The Centered Kernel Alignment (CKA) similarity is a measure commonly used in machine learning and neuroscience to quantify the similarity between two representations or feature spaces. It provides a way to assess the similarity of representations learned by different models or layers, even when the representations may have different dimensionalities. CKA is invariant to orthogonal transformations, such as isotropic scaling, permutations, reflections and rotations. This invariance property is particularly valuable when comparing representations that have undergone different preprocessing or normalization steps. By accounting for the underlying relationships between representations while being insensitive to orthogonal transformations, CKA enables a more meaningful and reliable assessment of similarity, aiding in tasks such as model comparison, representation learning, and understanding the neural code. For computational reasons, we use a modification of CKA index given by the following formula.

CKA similarity: Let $\mathbf{X}_i \in \mathbb{R}^{m \times p_1}$, $\mathbf{Y}_i \in \mathbb{R}^{m \times p_2}$ be the representations matrices from i^{th} minibatch of two layers of m samples and p_1 and p_2 number of features respectively. Similarly to [130], we estimate CKA index by averaging over k mini-batches:

$$sCKA = \frac{\frac{1}{k}\sum_{i=1}^{k} \mathrm{HSIC}\left(\mathbf{X}_i\mathbf{X}_i^\top, \mathbf{Y}_i\mathbf{Y}_i^\top\right)}{\sqrt{\frac{1}{k}\sum_{i=1}^{k} \mathrm{HSIC}\left(\mathbf{X}_i\mathbf{X}_i^\top, \mathbf{X}_i\mathbf{X}_i^\top\right)}\sqrt{\frac{1}{k}\sum_{i=1}^{k} \mathrm{HSIC}\left(\mathbf{Y}_i\mathbf{Y}_i^\top, \mathbf{Y}_i\mathbf{Y}_i^\top\right)}}, \tag{5.1}$$

where HSIC is an unbiased estimate or of the HSIC score [130]:

$$\mathrm{HSIC}(\mathbf{K}, \mathbf{L}) = \frac{1}{n(n-3)}\left(\mathrm{tr}(\tilde{\mathbf{K}}\tilde{\mathbf{L}}) + \frac{\mathbf{1}^\top\tilde{\mathbf{K}}\mathbf{1}\mathbf{1}^\top\tilde{\mathbf{L}}\mathbf{1}}{(n-1)(n-2)} - \frac{2}{n-2}\mathbf{1}^\top\tilde{\mathbf{K}}\tilde{\mathbf{L}}\mathbf{1}\right), \tag{5.2}$$

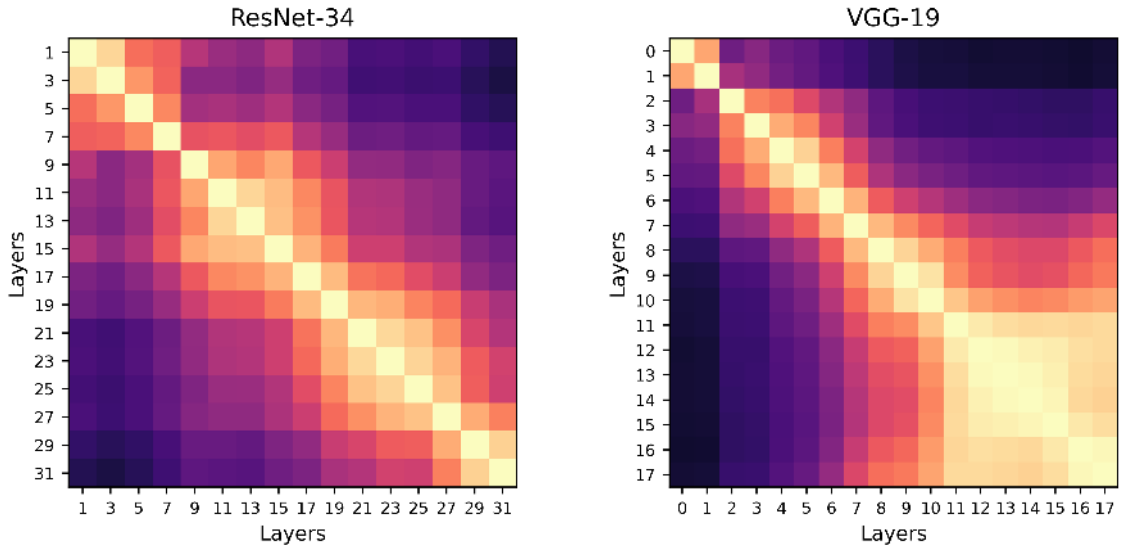
where $\tilde{\mathbf{L}} = \mathbf{L} - diag(\mathbf{L})$.

CKA is a normalized similarity index, hence value 1 means that representations matrices are identical.

In Figure 5.12.1, we present the plots of representations similarity for ResNet-18 and VGG-19 networks.

5.13. Inter and Intra class variance

Understanding the concepts of inter-class and intra-class variance is particularly important in the context of deep neural network representations anal-



(a) ResNet-34 **(b)** VGG-19

Figure 5.12.1. CKA-similarity across network's layers for ResNet-34 and VGG-19.

ysis for classification tasks. In this context, inter-class variance refers to the variability between different classes or categories of data. On the one hand, they capture the representation of the linear separability of a given task [187]. On the other hand, intra-class variance is an indicator of representations transferability [188, 189].

Let $\mathbf{X}_j \in \mathbb{R}^{t_j \times p}$ be the representations matrix for samples from j^{th} class. $\frac{1}{C}\sum_{j=1}^{C}\left(\frac{1}{t_j}\sum_{x_i \in \mathbf{X}_j}\left\|\mathbf{f}_i - \mu\left(\mathbf{X}_j\right)\right\|^2\right)$ is the intra-class variance, where $\mathbf{f}_i$ is a representation of sample x_i, $\mu\left(\mathbf{X}_j\right)$ is the mean representation of representations matrix $\mathbf{X}_j$, and C is the number of classes. Then $\frac{1}{C(C-1)}\sum_{j=1}^{C}\sum_{k=1,k\neq j}^{C}\|\mu\left(\mathbf{X}_j\right) - \mu(\mathbf{X}_k)\|^2$ is the inter-class variance.

5.14. Tunnel development

In this section, we provide a more detailed analysis of the evolution of numerical rank in VGG-19 dataset. In this experiment, we save the checkpoint of the network every epoch and calculate its numerical rank. The results are depicted in Figure 5.14.1.

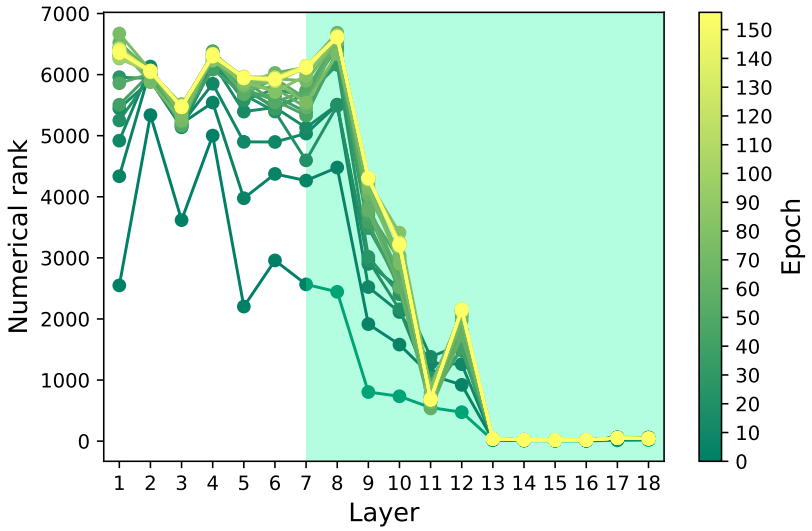


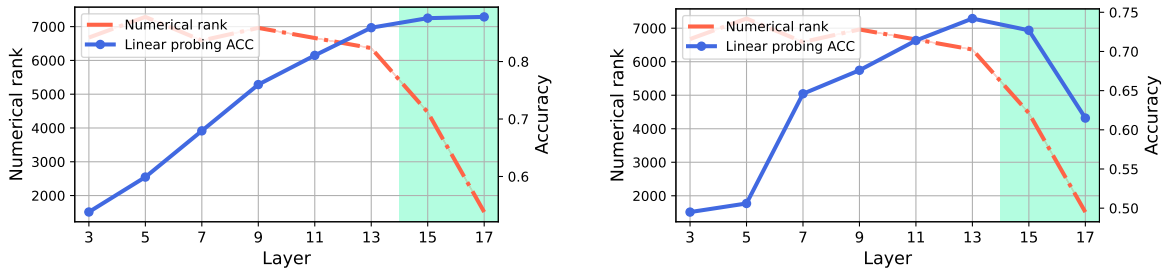
Figure 5.14.1. Evolution of numerical rank of VGG-19 representations throughout the training on CIFAR-10.

Initially, during the early epochs, the rank collapses primarily in the deeper layers. Throughout the training process, two distinct patterns can be observed. Firstly, the numerical rank of representations from the earlier layers tends to increase. Secondly, the numerical rank of representations from the deeper layers decreases. Interestingly, the place of this transition aligns with the beginning of the tunnel in the network. Once the numerical rank in deeper layers collapsed in the first gradient steps, as shown in Figure 5.3.2, it remained collapsed throughout the whole training.

5.15. ResNets without skip connections

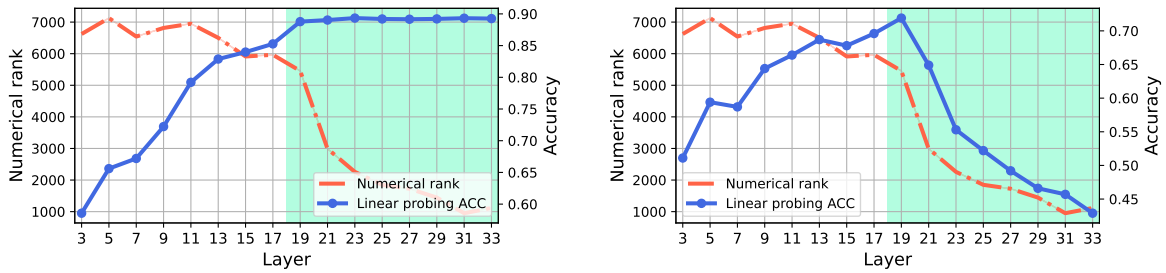
In this section, we delve into the impact of skip connections on the formation of the tunnel effect. To investigate this relationship, we trained ResNet models (ResNet-18 and ResNet-34) without skip connections on CIFAR-10 and conducted the same analysis used in the main paper. Specifically, we examined the linear probing performance for both in-distribution and out-of-distribution data and estimated the representations' numerical rank. The results, depicted in Figure 5.15.1 and Figure 5.15.2, highlight the significance of skip connections in the formation of the tunnel effect. Firstly, in the absence of skip connections (Plainnets), the tunnel effect is slightly more pronounced, with the model's performance saturating two layers earlier than standard ResNet

networks. Secondly, the rank of the representations exhibits a more predictable pattern without the spike at 29^{th} layer. This suggests that the spike in the numerical rank and in OOD performance is related to the skip connections. Interestingly, the numerical rank in both networks is higher than in the case of VGGs. The reason for this difference needs a further investigation. Lastly, the presence or absence of skip connections does not alter the degradation of out-of-distribution performance. However, in the absence of skip connections, the deterioration is more severe, aligning with the observation that it correlates with the numerical rank of the representations.



(a) In distribution **(b)** Out of distribution

Figure 5.15.1. In and out of distribution linear probing performance for ResNet-18 without skip connections trained on CIFAR-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed with random 10 class subsets of CIFAR-100.



(a) In distribution **(b)** Out of distribution

Figure 5.15.2. In and out of distribution linear probing performance for ResNet-34 without skip connections trained on CIFAR-10. The shaded area depicts the tunnel, the red dashed line depicts the numerical rank and the blue curve depicts linear probing accuracy (in and out of distribution) respectively. Out-of-distribution performance is computed with random 10 class subsets of CIFAR-100.

6. Unpacking softmax: How temperature drives rank collapse, generalization and compression

Abstract

The softmaxfunction is a fundamental building block of deep neural networks, commonly used to define output distributions in classification tasks or attention weights in transformer architectures. Despite its widespread use and proven effectiveness, its influence on the model's learning dynamics and, therefore, learned representations remains poorly understood. This paper studies the pivotal role of the softmaxfunction in shaping the model's representation. We introduce the concept of *rank deficit bias* — a phenomenon in which softmax-based deep networks converge to lower-rank solutions compared to shallow counterparts or architecture that do not rely on the softmaxactivation. We show that this effect is due to the impact of the activation on learning and demonstrate that the softmaxtemperature explicitly regulates this behavior. Furthermore, we demonstrate how to exploit the softmaxdynamics to learn compressed representations or to enhance their performance on out-of-distribution data. We validate our findings across multiple architectures and real-world datasets, highlighting the broad applicability of temperature tuning. We believe this work sheds light on the underlying mechanisms of softmaxand its impact on deep learning models.

6.1. Introduction

The softmaxfunction defined in Equation 6.1, with a hyperparamter $T > 0$ as the *temperature*, is a cornerstone of deep learning and is primarily used in tasks such as classification and text generation to transform raw model outputs into probability distributions with maximum entropy. By amplifying the largest values and diminishing smaller ones, softmaxenables models to make confident

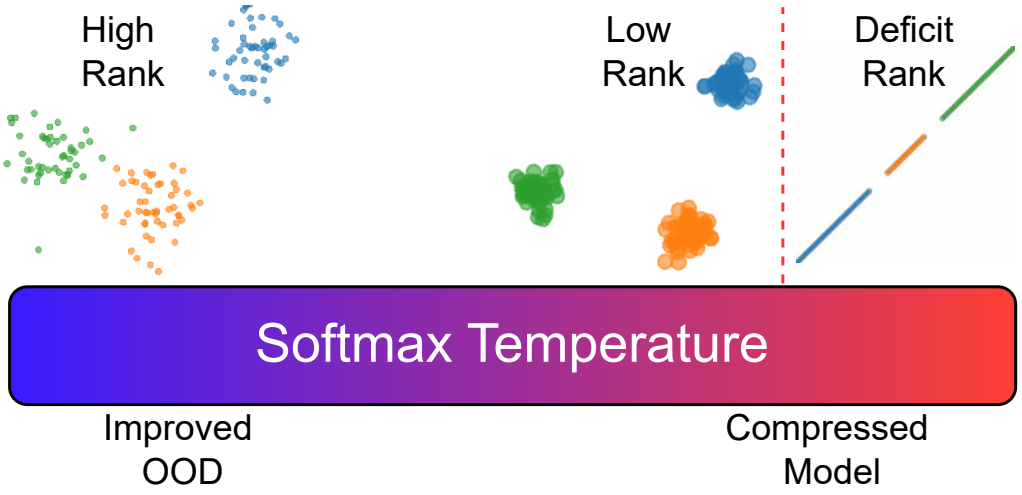


Figure 6.1.1. The softmaxfunction and temperature play a fundamental role in shaping network representations leading to well generalizing solutions with high rank for small temperatures or compressed solutions with *rank deficit* at the other extreme.

predictions, making it essential for decision-making among multiple options. However, its "winner-takes-all" nature [190, 191, 192] can sometimes lead to training challenges [193, 194, 195, 196].

$$\text{softmax}_T(\mathbf{e}) = \left[\frac{\exp(e_1/T)}{\sum_k \exp(e_k/T)} \quad \cdots \quad \frac{\exp(e_n/T)}{\sum_k \exp(e_k/T)} \right] \tag{6.1}$$

Originally, softmaxwas employed as the final layer in classification tasks. However, with the advent of the self-attention mechanism [74], it became a common feature in intermediate layers, particularly for text generation and later for image classification [197]. Despite numerous proposed alternatives for self-attention layers [198, 199], softmaxbased attention remains the most widely used choice in attention-based models.

This enduring popularity has spurred extensive research into the implicit trade-offs of using softmax, such as its impact on the sharpness of output distributions [200] and its role in normalizing activations and gradients [198]. Yet, the deeper effects of softmaxon model behavior—particularly its influence on learned representations and generalization—remain poorly understood. This gap motivates our central research question:

How does softmaxshape neural network representations?

Our answer to this question unfolds in three key insights presented in Figure 6.1.1:

First, we introduce the concept of *rank-deficit bias* – a novel phenomenon

– where neural networks trained with softmax(magenta curve) converge to solutions with rank significantly lower than the number of classes, a solution found by neural networks trained without softmax(blue curve) as illustrated in Figure 6.2.1. While both solutions achieve perfect performance on the training data, the rank-deficient solution generalizes poorly on out-of-distribution data, as demonstrated in Figure 6.2.2.

Second, we demonstrate that carefully adjusting the softmaxtemperature provides a robust mechanism to control the network's ability to compress data representation and it affects its out-of-distribution performance (Section 6.3). In section 6.3.2 we explain why training deep networks at high temperatures leads to compressed models.

Finally, building on this explanation, we identify architectural choices that steer the network toward either compactness or improved generalization (Section 6.3.4). These insights offer a novel perspective on some popular building blocks in deep learning.

These findings deepen our understanding of how softmaxshapes neural network representations and provide actionable insights for optimizing model design and performance.

6.2. Rank deficit bias

This section introduces the rank deficient bias through a toy example model enabling us to explain the phenomenon without confounding factors. Section 6.2.2, explores the consequences of the rank deficit bias of the softmaxfunctions and its consequences on the model's generalization. Lastly, in Section 6.2.3, we provide a sketch of the theoretical analysis of the bias explaining the softmaxsurprising ability to increase the rank of the matrix. The complete analysis can be found in Appendix 6.7.

Experimental setup To build intuition, we begin by experimenting with a simple task to learn an identity mapping between $\mathbf{X} = \mathbf{Y} = \mathbf{I} \in \mathbb{R}^{d \times d}$. Thus, each data sample is a vector from the canonical basis assigned to a different class. We set $d = 5$ and use a deep linear network with three layers $\mathbf{W} \in \mathbb{R}^{d \times d}$. The network is trained with full-batch GD and Mean Squared Error (MSE) or CrossEntropy (CE) loss. During training, we measure the rank of the representation at the output layer of the network. In the case of networks trained with softmax, we consider representations before applying the softmaxfunction. We use the PyTorch [201] implementation of rank, which is defined as the number of singular values greater than the threshold based on the top singular value.

$$\text{rank}(\mathbf{A}) := \Sigma_{i=1}^{n} 1_{\sigma_i(\mathbf{A})>\eta}, \tag{6.2}$$

where $\mathbf{A} = \mathbf{W}_3\mathbf{W}_2\mathbf{W}_1\mathbf{X}$, and $\sigma_i(\mathbf{A})$ is the i-th singular value of the $\mathbf{A}$ matrix and η is a predefined threshold.

6.2.1. Results

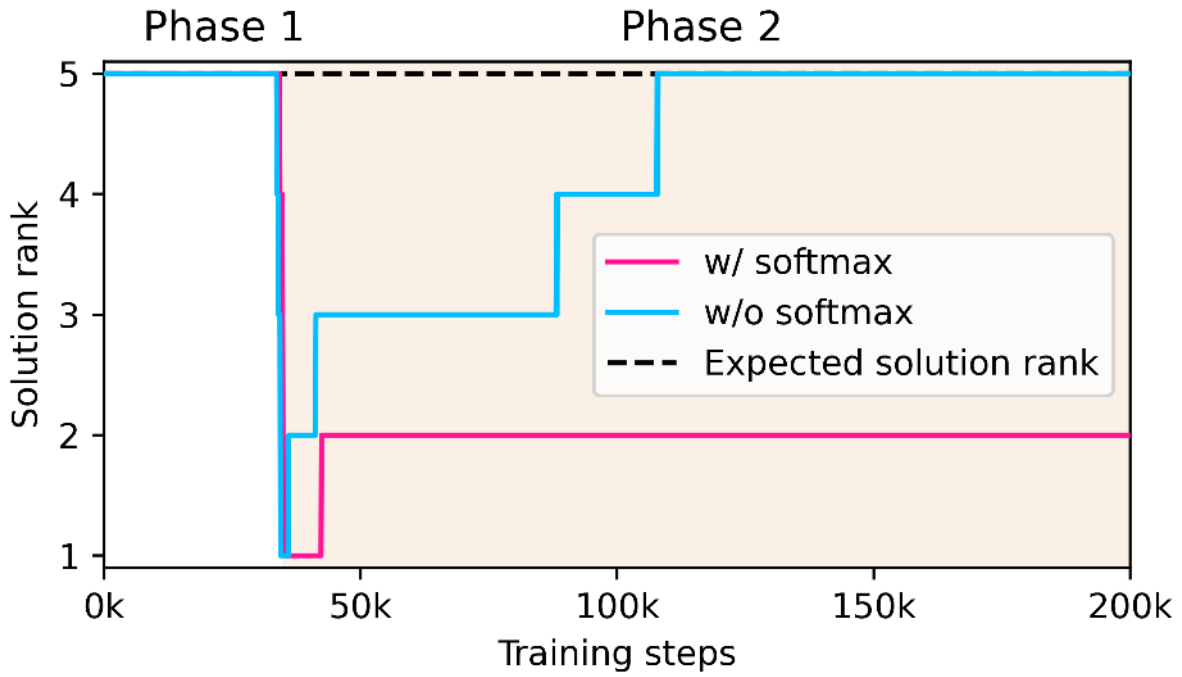


Figure 6.2.1. Despite similarities during the first phase, models trained with softmaxexhibit different training dynamics during the second training phase, yielding lower-rank representations compared to counterparts trained without softmax. Both models solve the task with 100% accuracy.

The analysis compares two identical neural networks trained with either MSE or CE loss. Although both networks start with full-rank representations, their training trajectories and final representations diverge. As illustrated in Figure 6.2.1, the training process can be divided into two phases based on the evolution of the rank:

Phase 1 Both models exhibit similar behavior in the initial phase, starting with full-rank representations that eventually collapse to rank-1 by the end of this phase. This phenomenon of representation collapse has been explored in recent theoretical studies [202, 203].

Phase 2 As predicted by the Saddle-to-Saddle dynamics model [202], networks trained with MSE follow a trajectory of expanding saddle points, reflected

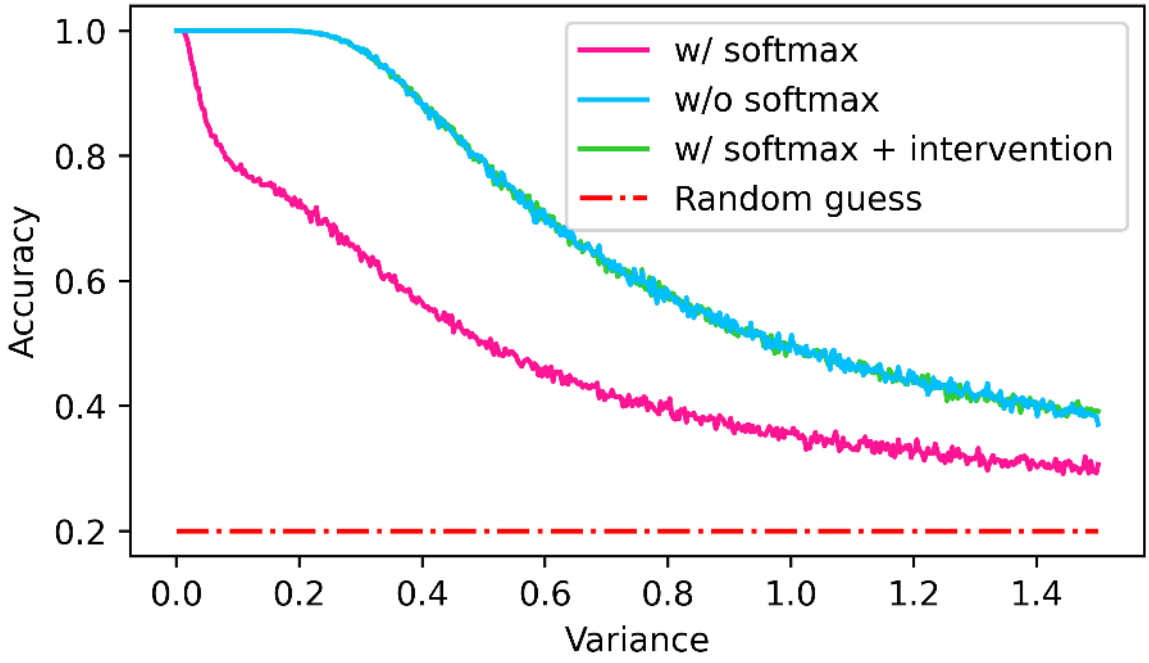


Figure 6.2.2. Accuracy of low-rank and full-rank models for simplified out-of-distribution dataset.

in the gradual increase in rank (blue curve) until a full rank is restored by the end of training. In contrast, networks trained with the softmaxfunction (magenta curve) maintain low-rank representations despite achieving perfect performance. This raises the question: *Which model learns better data representations?*

6.2.2. Consequences

To assess whether the rank of the final solution affects model performance, we designed a series of simple out-of-distribution (OOD) datasets. For each test dataset, we sampled $k = 1000$ points from an isotropic Gaussian distribution centered at each training sample (canonical vector), with increasing variance σ. Repeating this process for different σ values, we generated a sequence of test datasets to evaluate all models.

Figure 6.2.2 demonstrates that full-rank solutions consistently outperform low-rank solutions across all test datasets. The sharp accuracy drop for softmax-trained models suggests their solutions are brittle and vulnerable to small perturbations. Despite the simplicity of this test, it clearly shows that representation rank influences a network's generalization ability.

This stark difference motivates us to investigate the mechanisms behind rank collapse and explore ways to mitigate it. Specifically, we aim to identify the

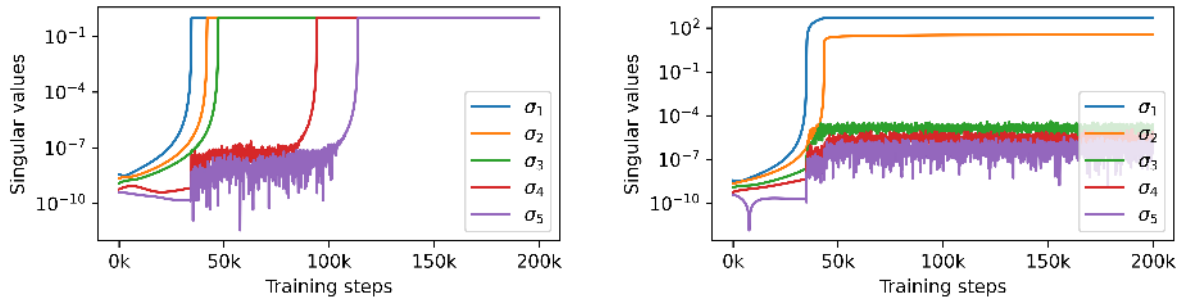


Figure 6.2.3. The evolution of singular values of representations extracted from the last layer of a deep linear model trained on the toy dataset without (left) and with (right) softmax.

factors in the softmaxfunction that hinder rank recovery, unlike in networks trained with MSE.

6.2.3. Analysis

To explore the differences in learning dynamics between the networks, we revisit the classic analysis of deep linear network dynamics from [204]. According to their results, when trained with MSE, the singular values of the network's weight product should converge to those of the input-output correlation matrix—an identity matrix in our toy model. Figure 6.2.3 (left) confirms this, showing all singular values converging to 1. Once the final singular value converges, the model achieves 100% accuracy, resulting in full-rank representations.

However, as shown in Figure 6.2.3 (right), the singular value evolution for networks trained with softmaxdiffers significantly. First, singular values converge to magnitudes much larger than those of the input-output correlation matrix. Second, only two of the expected five singular values grow substantially during training, aligning with the low-rank representations observed in Figure 6.2.1. To understand why training halts when the second singular value grows, we analyze the first 50k iterations and compute singular values both before and after the softmax. Figure 6.2.4 reveals a key insight: *the growth of the first pre-softmaxsingular value triggers the growth of all post-softmaxsingular values*. Subsequently, the growth of the second pre-softmaxsingular value stabilizes all post-softmaxsingular values at 1 (as predicted in [204]), ending training and causing the rank-deficient bias.

The next section outlines a theoretical explanation for the softmax's ability to alter the matrix spectrum, with a full analysis in Appendix 6.7.

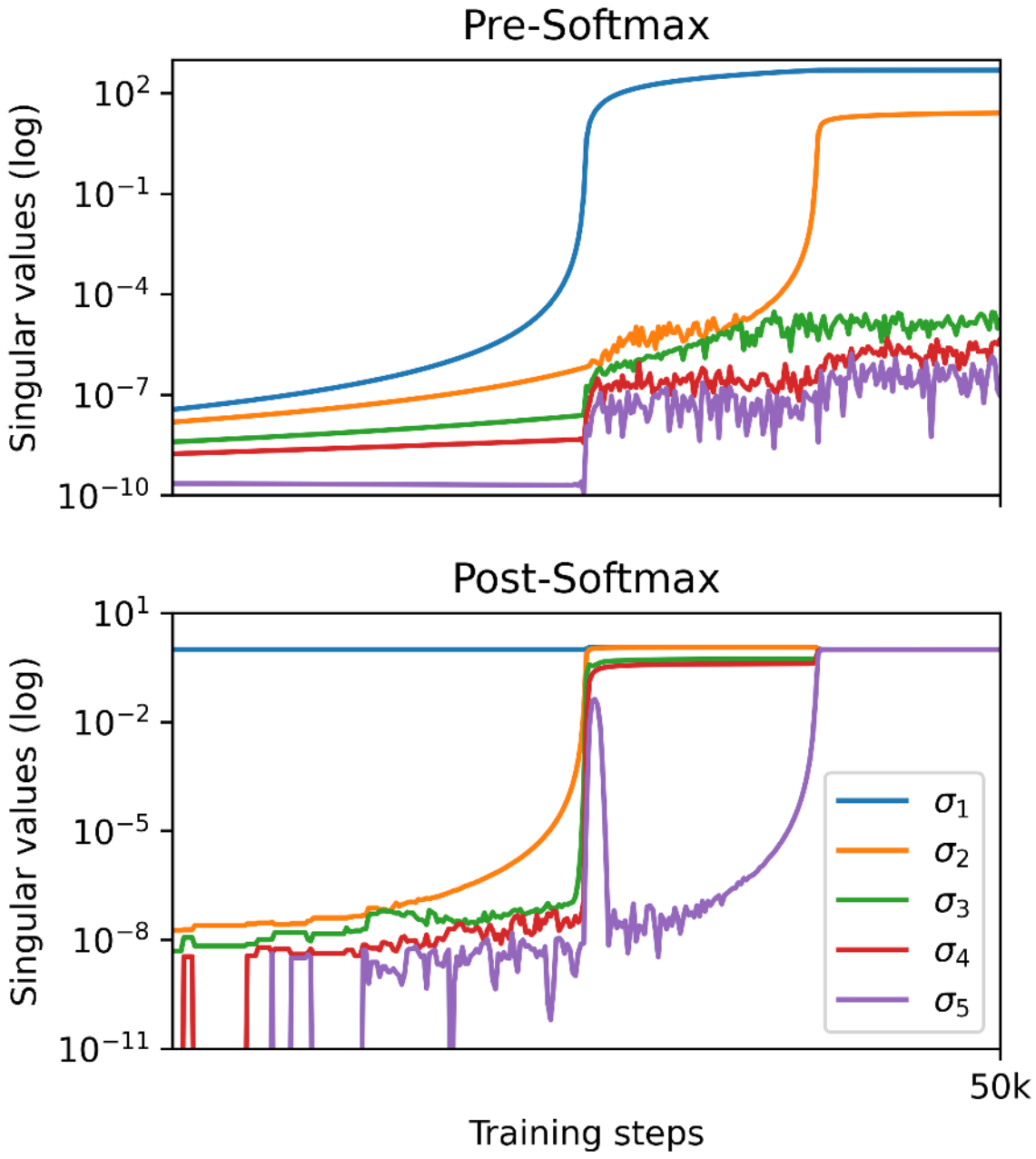


Figure 6.2.4. Evolution of the spectrum of representations during training at the output layer pre-softmax (top) and post-softmax (bottom).

6.2.4. Can softmaxincrease matrix's rank?

Proposition 3 answers this for arbitrary matrices:

Proposition 1. *There exists* $\mathbf{B} \in \mathbb{R}^{n \times n}$ *and* $c \in \mathbb{R}$ *such that* $\text{rank}(\mathbf{B}) = 2$ *and* $\text{rank}(\textit{softmax}(c\mathbf{B})) = n$.

The proof of the Proposition 3 is in the Appendix 6.6. Since it shows that one can trivially construct a low-rank matrix with arbitrarily high post-softmax rank, we study the impact of applying the softmaxfunction on a random rank-1 matrix scaled by a constant c. We will use this model to approximate the dynamics observed from the training as shown in Figure 6.2.4 when only the top singular values of the pre-softmax matrix increase sharply.

One can notice that scaling a pre-softmax matrix by a constant c can be understood as computing a softmaxfunction with temperature $T = \frac{1}{c}$. It is a known fact that in the temperature limits, the softmaxfunction converges to either the argmax function ($T \to 0$) or the uniform function ($T \to \infty$). However, Figure 6.2.5 reveals unexpected behavior – applying softmaxwith intermediate temperature increases matrix rank the most.

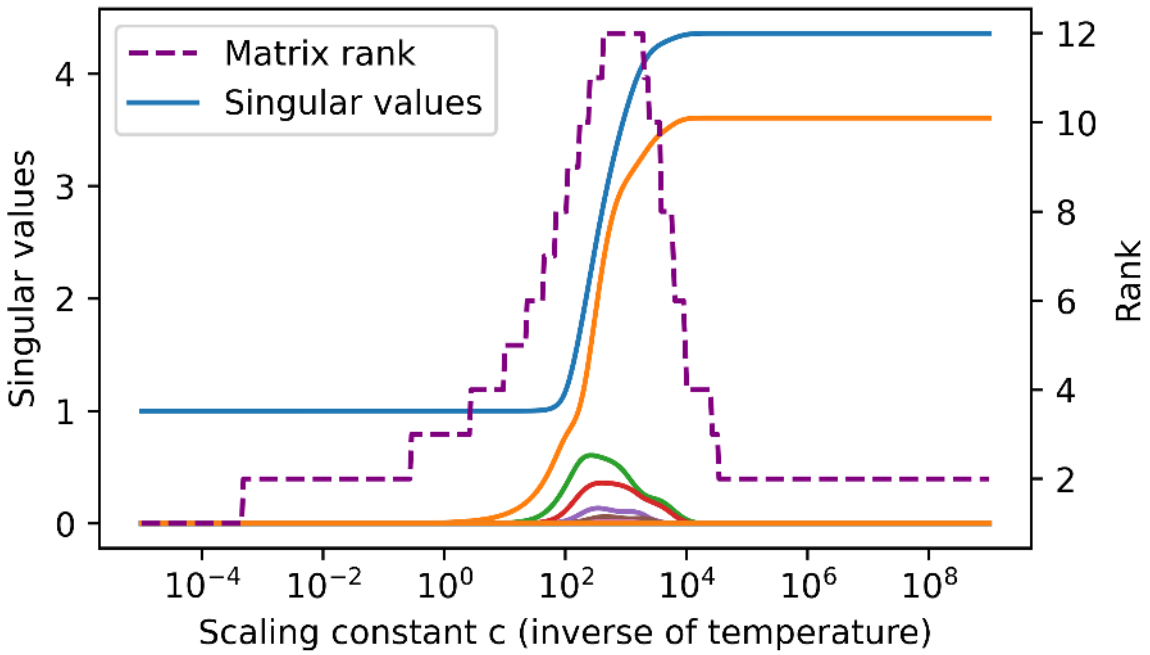


Figure 6.2.5. The evolution of singular values and rank of the post-softmax matrix scaled by a constant c (inverse of temperature). The matrix $\mathbf{B} := \mathbf{x}\mathbf{y}^\top \in \mathbb{R}^{n \times n}$ is the outer product of two random vectors, $x_i, y_i \sim \mathcal{U}(-1, 1)$ and therefore $\text{rank}(\mathbf{B}) = 1$. The behavior is consistent across different samples.

To formally analyze this effect, we propose the following upper bound on the difference between top ($\sigma_1(\mathbf{M})$) and bottom ($\sigma_n(\mathbf{M})$) singular values of matrix $\mathbf{M}$, where $\mathbf{M} = \text{softmax}(\mathbf{A})$ for any real $\mathbf{A} \in \mathbb{R}^{n \times n}$:

Proposition 2. *For any matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$ with each column $\mathbf{m}_j \in \mathbb{R}^n$ as a probability vector. Then the gap between the largest $\sigma_1(\mathbf{M})$ and the smallest singular value $\sigma_n(\mathbf{M})$ is bounded by the following tight inequality:*

$$0 \leq \sigma_1(\mathbf{M}) - \sigma_n(\mathbf{M}) \leq \sqrt{1+r} - \sqrt{\max\left\{\frac{1}{n} - r, 0\right\}}$$

where $r := \max_i \sum_{j \neq i} \langle \mathbf{m}_i, \mathbf{m}_j \rangle$.

The Proof is in the Appendix 6.7.

Intuitively, the bound shows that decreasing the inner product between matrix columns shrinks the gap between its singular values. Thus, the remaining question is how applying softmaxwith different temperatures impacts the inner products between matrix columns. Formal analysis would be cumbersome due to the high-dimensional nature of the problem. Instead, Figure 6.8.1 (bottom) provides an empirical answer showing that applying softmaxcolumn-wise on low-rank matrix with various temperatures decreases the inner products between the columns with different indices of the highest element and increases the inner products between the columns with the same indices. The temperature at which the bifurcation of inner products happens aligns with the temperature that shrinks the difference $\sigma_1(\mathbf{M}) - \sigma_n(\mathbf{M})$ and results in increased rank as shown in Figure 6.8.1 (top).

6.2.5. Avoiding rank deficit bias

The previous section highlighted that the growth of top singular values drives the observed rank-deficient bias. Leveraging this insight, we propose a dynamically tuned temperature to counteract the low-rank bias. The idea is that if the network cannot exploit the rank-increasing property of softmax, it should converge to a full-rank solution. To achieve this, we dynamically set the softmaxtemperature equal to the Frobenius norm of the logits. As shown in Figure 6.2.2 (the green curve is almost invisible as it aligns with the blue curve showing the generalization of models trained without softmax), this approach successfully prevents rank-deficient bias, enabling networks to achieve full-rank solutions with generalization capabilities similar to those trained without softmax.

6.3. The role of the softmaxtemperature

The previous section examined why networks struggle to recover their rank after the initial collapse of representations. However, this analysis assumed that models initially collapse, as illustrated in Figure 6.2.1. This section delves into the causes of this initial collapse, identifying the norm of the logits as the primary driver. While numerous architectural factors influence this norm (discussed in Section 6.3.4), we emphasize that the simplest way to control it is by applying a softmaxtemperature. To understand how training with a high softmaxtemperature leads to collapsed representations, Section 6.3.2 introduces a simplified model describing the training dynamics of neural networks under such conditions. Section 6.3.3 then explores the implications of collapsed rank representations. This demonstrates that adjusting the softmaxtemperature allows control over the learning trajectory and model representations, significantly affecting model compression and out-of-distribution generalization.

6.3.1. Experiments

We trained ResNet-18, ResNet-34, ResNet-50 on CIFAR-10 and ImageNet-1k. Baseline models (temperature = 1) used recommended hyperparameters for optimal performance. To isolate the effect of the softmaxtemperature, all hyperparameters remained fixed except for the learning rate, which was adjusted to compensate for slower training at higher temperatures.

Figure 6.3.1 shows that training the same randomly initialized model with different temperatures yields distinct representations, despite similar performance. The green dashed lines represent the rank of representations in the randomly initialized model. For models trained with temperature = 1, the rank increases during training, especially in deeper layers. In contrast, models trained with high temperatures maximize rank in initial layers (reaching an upper bound, red dashed line) but sharply reduce it in deeper layers, nearing zero.

To explain how a single scalar change produces such divergent outcomes, we propose a simplified model of MLP training dynamics in the following section.

6.3.2. Analysis

High temperature creates loss symmetry

To understand the impact of the softmaxtemperature on the learning dynamics, recall that increasing the temperature in the softmaxfunction results in more uniform output distributions, maximizing entropy and approaching equally

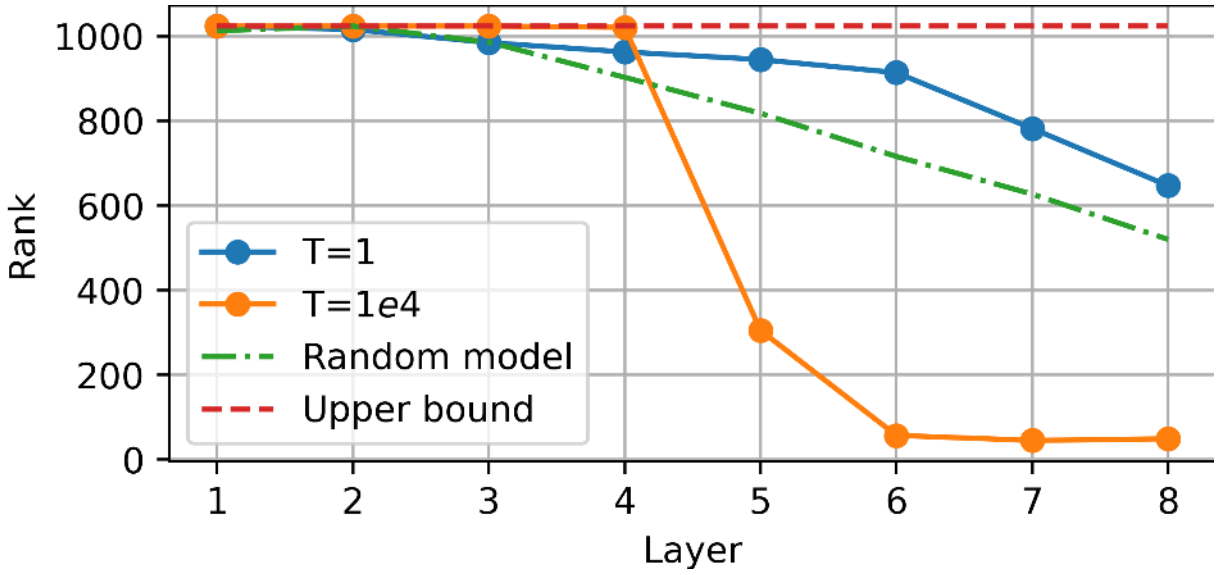


Figure 6.3.1. Training model with high temperature leads to the collapse of representations in deeper layers of the model. MLP with 8 layers trained on CIFAR-10.

probable outputs in the limit [200]. From the perspective of CrossEntropy loss, this creates a symmetric loss landscape where each sample incurs an equal loss of $\ln(\frac{1}{n})$, where n is the number of classes for each sample. This symmetry poses a challenge for learning.

Breaking loss symmetry requires increasing logits norm

Equation 6.1 illustrates the relationship between the logits norm $\mathbf{e}$ and the temperature T. To break the symmetry and reduce the entropy of the softmaxdistribution, one can either decrease the temperature (as shown in [200]) or increase the norm of the logits. Since we assume training with a fixed high temperature, the only viable option for networks to break the symmetry is to increase the logit norm. As shown in Figure 6.3.2, models trained with high temperature exhibit an increasing logit norm, unlike those trained with temperature = 1 (Figure 6.8.2), which maintain or even reduce the norm throughout training.

Two ways of increasing logits norm

The representations at the i-th layer of an MLP network can be described recursively as $\mathbf{A}_i := \text{ReLU}(\mathbf{W}_i\mathbf{A}_{i-1})$, where $\mathbf{A}_0 := \mathbf{X}$ (the input data) and $\mathbf{W}_i$ is the weight matrix for the i-th layer.

Let us simplify and assume that $\mathbf{W}$ is the last weight matrix and $\mathbf{A}$ is the representations matrix from the penultimate layer. Then to maximize the Frobenius norm of the logits, we can use the singular value decomposition (SVD) of $\mathbf{W}$ and $\mathbf{A}$:

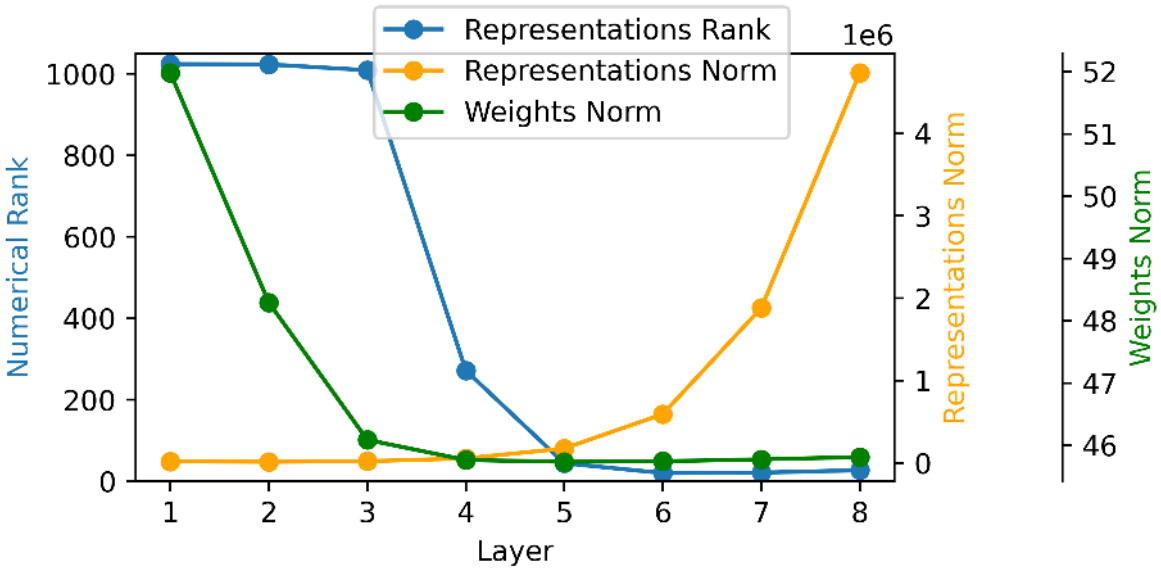


Figure 6.3.2. High-temperature training forces network to produce high-norm logits. The plot shows the relationship between the norm of representations, rank of representations, and norm of weights matrices.

$$\begin{aligned} \|\mathbf{WA}\| &= \|\mathbf{U_W}\Sigma_\mathbf{W}\mathbf{V_W^\top}\mathbf{U_A}\Sigma_\mathbf{A}\mathbf{V_A^\top}\| \\ &= \|\Sigma_\mathbf{W}\mathbf{V_W^\top}\mathbf{U_A}\Sigma_\mathbf{A}\| \leq \|\Sigma_\mathbf{W}\Sigma_\mathbf{A}\|, \end{aligned} \tag{6.3}$$

where $\mathbf{U_W}\Sigma_\mathbf{W}\mathbf{V_W^\top} = \mathbf{W}$ and $\mathbf{U_A}\Sigma_\mathbf{A}\mathbf{V_A^\top} = \mathbf{A}$ are the SVDs of $\mathbf{W}$ and $\mathbf{A}$, respectively. The second equality holds because singular vectors are orthonormal. This inequality shows that the norm of the product of two matrices can be increased in two ways:

1) Increasing the norm of $\mathbf{W}$ and $\mathbf{A}$ individually: This increases the diagonal values of $\Sigma_\mathbf{W}$ and $\Sigma_\mathbf{A}$, effectively scaling up the singular values of each matrix.

2) Aligning singular vectors: The norm is maximized when $\mathbf{V_W^\top}\mathbf{U_A} = \mathbf{I}$, meaning the right singular vectors of $\mathbf{W}$ align with the left singular vectors of $\mathbf{A}$. Since both matrices are orthonormal, their product achieves its maximum norm when they are perfectly aligned. Note that the argument can be applied recursively across all layers, with $\mathbf{W} = \mathbf{W}_i$ (weights) and $\mathbf{A} = \mathbf{A}_{i-1}$ (representations).

To determine which of these two mechanisms is exploited during training, we track the alignment of the top 15 singular vectors of weight and representation matrices across consecutive layers during the initial epochs.

Figure 6.3.3 demonstrates that networks trained with high-temperature exhibit early alignment of the top singular vectors of weights and representations in consecutive layers. Additionally, Figure 6.3.2 shows that while the weights' norms are higher in the collapsed model compared to the model trained

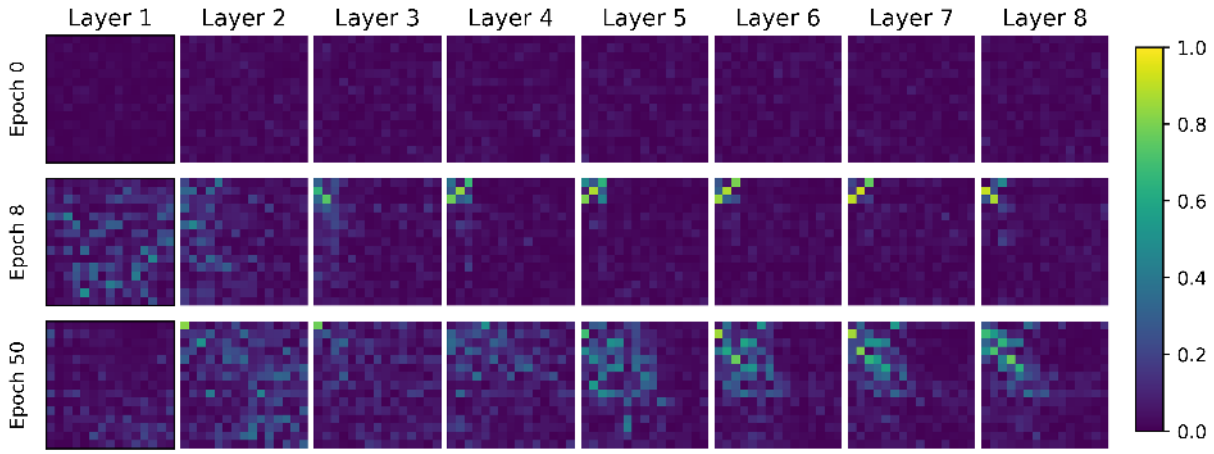


Figure 6.3.3. High-temperature training aligns top singular vectors from consecutive layers. Figure 6.8.3 presents the alignment formation in greater detail.

with temperature = 1, the collapsed layers contribute most significantly to the increase in representations' norms despite having smaller individual norms. This suggests that alignment, rather than individual scaling, plays a critical role in increasing the logits norm under high-temperature training.

Aligned layers increase norm exponentially

When the top singular vectors across all layers are maximally aligned, the top singular value of the logits, σ_1^L, can be expressed as the product of the top singular values of each layer and lower bounded: $\sigma_1^L = \prod_{i=1}^{L-1} \sigma_1^i \geq (\sigma_1^k)^L$, where σ_1^i is the top singular value of the i-th layer and $k = \arg\min_i \sigma_1^i$ is the layer with the smallest top singular value. If $\sigma_1^k > 1$, this results in exponential growth of the top logit singular value as the network depth L increases.

The same reasoning applies to subsequent singular values, as their corresponding singular vectors also become aligned across layers. This alignment not only amplifies the singular values of the logits but also affects the gradients. Since gradients depend on the activations, increased alignment (and the resulting skewness in the spectra of representations) similarly skews the spectra of the gradients. This demonstrates how alignment in deep networks can lead to exponential scaling and significant spectral changes in both activations and gradients.

6.3.3. Consequences

Training with high temperatures not only impacts the rank of learned representations but also fundamentally alters how efficiently a model utilizes its layers. This enables models to achieve comparable performance with fewer layers, while significantly influencing their ability to generalize to out-of-distribution

(OOD) data. Figure 6.3.4 illustrates this by showing the performance of linear probes attached to each layer of a ResNet-18 model trained on CIFAR-10 and evaluated on a 10-class subset of CIFAR-100 as the OOD dataset. Notably, the model trained with high temperature achieves its peak performance at the 12th layer out of 18, resulting in a compression ratio of $\kappa = \frac{12}{18} = 67\%$. Here, κ measures layer utilization efficiency and is computed by identifying the first layer where the linear probe achieves at least η (99% for CIFAR-10 or 95% for ImageNet-1k) of the highest accuracy across all layers, thus the lower the value, the more efficiently the network utilizes its layers. However, this compression comes at a cost: Figure 6.3.4 (bottom) shows that out-of-distribution (OOD) accuracy peaks around the same layer and then declines. We quantify this OOD generalization decrease using ρ, defined as the normalized difference between the best OOD accuracy and the accuracy at the final layer. Thus, the value is non-negative, and the smaller the value, the better.

To explore the broader implications, we analyze compressibility (κ) and OOD generalization loss (ρ) across various architectures and datasets. Results on CIFAR-10 (Table 6.3.1) and ImageNet-1k (Table 6.3.2) reveal consistent trends: high-temperature training enhances compressibility but often at the expense of OOD generalization. Notably, models trained on ImageNet-1k require higher temperatures to exhibit this effect. These findings highlight a trade-off between layer efficiency and model robustness, with temperature serving as a key control parameter.

Our results partially align with the tunnel effect hypothesis [205, 206], which suggests that deeper layers in over-parameterized models act as lossless compressors for in-distribution data but hinder OOD performance. However, Figure 6.3.4 shows that tunnel compression is not always lossless and that the strength of the tunnel effect can be directly modulated by the softmaxtemperature. This offers a new mechanism for controlling model behavior, balancing efficiency and generalization.

6.3.4. What factors implicitly change the temperature?

The previous section highlighted the critical role of the softmax temperature in shaping neural network representations. Although the potential advantages of manipulating the softmax temperature were discussed in Section 6.3.3, this approach has not been popularized. However, this section outlines architectural choices and experimental designs that have gained popularity across various domains. We demonstrate that these choices implicitly influence the norm of

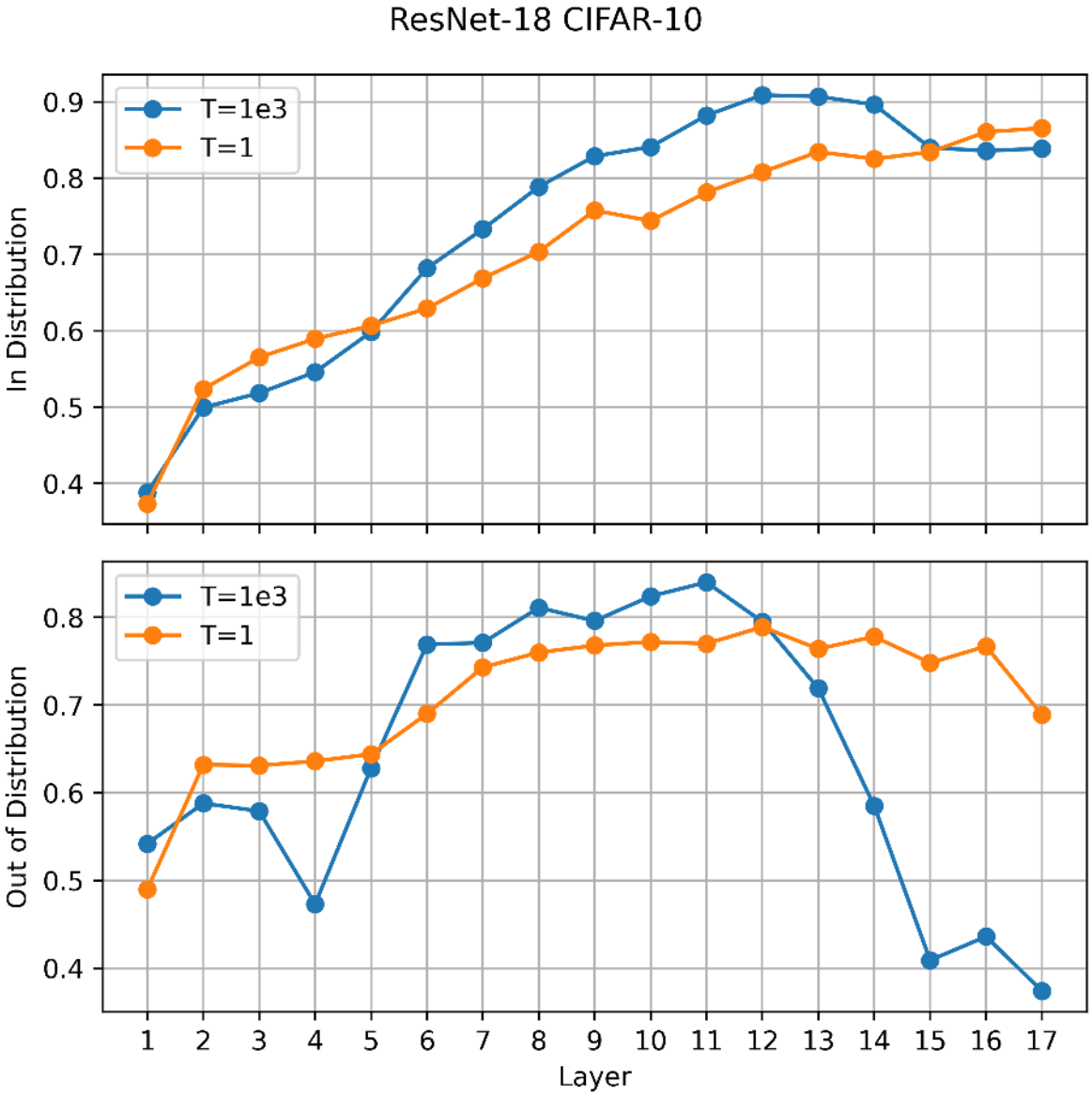


Figure 6.3.4. ResNet-18 trained on CIFAR-10 with different softmaxtemperatures and evaluated on a 10-class subset of CIFAR-100 as the OOD dataset. Top: In-distribution accuracy across layers. Bottom: OOD accuracy across layers.

Table 6.3.1. Training the model at higher temperature compresses the model. The values in the table present κ and ρ for different models trained with CIFAR-10 and evaluated with SVHN dataset as an out of distribution task.

Dataset	CIFAR-10			
Temperature	1	1e1	1e2	1e3
MLP-8	75%/67%	50%/67%	38%/71%	38%/78%
ResNet-18	94%/17%	83%/56%	72%/54%	67%/67%
ResNet-20	90%/10%	85%/28%	80%/36%	60%/50%

Table 6.3.2. Training the model at higher temperature compresses the model. The values in the table present κ and ρ for different models trained with ImageNet-1k and evaluated with the CIFAR-100 dataset as an out-of-distribution task.

Dataset	ImageNet-1k			
Temperature	1	$1e1$	$1e2$	$1e3$
ResNet-34	94%/5%	94%/2%	94%/11%	82%/23%
ResNet-50	96%/5%	96%/2%	90%/14%	78%/22%

the logits (or the softmaxtemperature), thereby biasing the models toward one of the previously discussed regimes.

Weights initialization impacts activation scales and early training dynamics. In Contrastive Reinforcement Learning (RL), the final layer weights are initialized from $\mathcal{U}(-\alpha, \alpha)$, where $\alpha = 10^{-12}$ keeping representations tightly clustered at the start of training [207]. This stabilizes optimization but may slow early learning.

The width of a network affects representation learning in various ways. In Self-Supervised Learning (SSL), the projection head used in contrastive frameworks is typically much wider than the backbone [208]. This significantly impacts representation learning. Empirical studies show that training SimCLR with and without a projector reveals that the representation space spectrum collapses without it, reducing feature expressivity and degrading downstream performance [209].

Normalization layers can mitigate the growth of the logits' norm. As Section 6.2.5 mentions, dividing the logits by their Frobenius norm helps prevent rank deficit. This dynamic rescaling can, at some point, be viewed as dividing by the standard deviation of the logits, a key operation in Layer Normalization [210]. Moreover, [211] demonstrated that batch normalization stabilizes training and mitigates rank collapse, preserving a more diverse representation across layers.

6.4. Related Works

Our work intersects with several research areas, including implicit biases in deep learning, the role of softmaxin neural networks, representation collapse, and the interplay between out-of-distribution (OOD) generalization and model compression. Below, we contextualize our contributions within these fields.

The implicit biases of gradient descent in deep linear networks have been extensively studied. [212] and [213] demonstrated that gradient descent tends to align layers in deep linear networks, leading to low-rank solutions. [202] further explored saddle-to-saddle dynamics in such networks, highlighting the role of initialization and symmetry in shaping learned representations. These studies provide a foundation for understanding how optimization dynamics influence model behavior; however, none of them explored the effect of softmaxfunction on the dynamics. Our work extends this perspective by introducing the evidence for softmax-induced rank-deficit bias.

The role of softmaxin deep neural networks is still poorly understood, and several works have examined its limitations and dynamics. [200] highlighted issues with softmaxin sharp OOD predictions, while [198] proposed polynomial activations as alternatives to softmaxin self-attention. [214] addressed entropy collapse in transformer attention, emphasizing the need for stable training dynamics. Our work builds on these insights by uncovering how softmaxtemperature regulates rank-deficit bias and influences representation learning.

Representation collapse, where learned features lose discriminative power, has been studied in various contexts. [215] identified neural collapse in the terminal phase of training, where features converge to a simplex structure. [205] explored the "tunnel effect," where deep networks build data representations through iterative compression. [206] investigated factors affecting the strength of the tunnel effect in pretrained models. Our findings on rank-deficit bias contribute to this literature by linking softmaxdynamics to representation collapse and generalization.

The trade-off between model compression and OOD generalization has been a focus of recent research [206, 205]. [139, 205, 206] demonstrated how intermediate representations can improve transfer learning. On the other hand [216] introduced the information bottleneck principle to explain deep learning dynamics. Later [217] explored the complexity dynamics of grokking, where models generalize after prolonged training and link this ability with the simplicity of learned functions. Our work extends these ideas by showing how softmaxtemperature can be tuned to balance compression and OOD performance, offering a practical mechanism for controlling representation rank.

By synthesizing insights from these areas, our work provides a unified perspective on the role of softmaxin shaping neural network representations, offering new directions for understanding and improving deep learning models.

6.5. Conclusions

In this work, we explored the role of the softmaxfunction in shaping deep neural network representations. We identified a *rank-deficit bias*, where softmax-trained networks converge to lower-rank solutions, affecting their generalization, especially on out-of-distribution data. We showed that the softmaxtemperature is a key control mechanism, allowing practitioners to balance compressed representations and improved generalization. By tuning the temperature, we demonstrated how to achieve either compact models or enhanced out-of-distribution performance. Our analysis also provided insights into architectural choices, offering a unified framework to understand and better leverage the properties of the softmaxfunction in deep learning.

While focused on supervised image classification, our findings may extend to other domains. Future research directions include:

- Investigate the use of the softmaxtemperature in NLP and reinforcement learning. In NLP, understanding its impact on transformer attention mechanisms could improve tasks like machine translation and text generation. In reinforcement learning, studying its role in exploration-exploitation trade-offs could enhance policy optimization or prevent representation collapse [218].
- Study softmaxin non-stationary optimization, where data distributions or objectives change over time. Understanding how temperature influences adaptation and convergence in dynamic environments could lead to more robust models.
- Develop adaptive temperature schedulers to adjust softmaxtemperature during training dynamically. This could enable models to transition between high-rank and low-rank solutions, optimizing for generalization and compression at different stages.
- By pursuing these directions, future work can build on our insights to advance the understanding of softmaxand its role in deep learning, fostering more effective and interpretable models across diverse applications.

6.6. Arbitrary matrix can recover full rank post-softmax

Proposition 3. *There exists* $\mathbf{B} \in \mathbb{R}^{n \times n}$ *and* $c \in \mathbb{R}$ *such that* $\text{rank}(\mathbf{B}) = 2$ *and* $\text{rank}(\textit{softmax}(c\mathbf{B})) = n$.

Proof. Let $\mathbf{A} \in \mathbb{R}^{n \times 2}$ be an arbitrary random matrix with normalized rows. Then, $\mathbf{B} = \mathbf{A}\mathbf{A}^\top \in \mathbb{R}^{n \times n}$ and one can easily observe that $\forall i, j \in \{1, \ldots, n\}, b_{ij} \leq b_{ii}$. Then, applying a softmaxfunction with temperature $c \to 0$, we obtain: $lim_{c \to \infty} \sigma_c(\mathbf{B}) = \mathbf{I}$ and $\text{rank}(\mathbf{I}) = n$. At the same time $\text{rank}(\mathbf{B}) = 2$. □

6.7. How softmax scales the rank of the matrix?

The results from Section 6.2.1 shows that deep linear networks trained with softmax successfully minimize the loss and solve the task despite the rank-deficient representations at the output layer.

Minimizing the loss in the toy dataset requires approximating the target matrix $\mathbf{Y} \in \mathbb{R}^{5 \times 5}$ being the rank five identity matrix $\mathbf{I}_5$. This implies that the column/row-wise application of the softmax on a matrix alters its spectrum and potentially increases its rank. As shown in Figure 6.2.4, the softmax function alters the spectrum in a non-trivial way, producing equal singular values in the post-softmax matrix at the end, unlike the pre-softmax matrix, which is dominated by two singular values. Comparing both plots in Figure 6.2.4 reveals that the sharply increasing magnitude of singular values pre-softmax raises the magnitude of all singular values in the post-softmax matrix.

To understand whether this is a coincidence, we design an experiment when we randomly sample rank one matrix $\mathbf{A} \in \mathbb{R}^{32 \times 32}$ and compute its rank after applying a softmax function. To mimic the growth of the top singular value, we repeat the process by scaling the base matrix $\mathbf{A}$ with different constant $c \in [1e-5, 1e9]$. Scaling rank 1 matrix by a scalar has no effect on the rank of a matrix and $\forall c \in \mathbb{R}_+ \, \text{rank}(c\mathbf{A}) = 1$, however as shown in Figure 6.2.5 applying a softmax on a scaled rank one matrix indeed increases its rank. Besides that, one can observe that $\lim_{c \to 0} \text{rank}(\sigma(c\mathbf{A})) = 1$ and $\lim_{c \to \infty} \text{rank}(\sigma(c\mathbf{A})) = \text{rank}(\text{argmax}(\mathbf{A}))$, where argmax() is applied column-wise and returns one-hot vector. These are

the properties of the temperature-softmax for limit values of the temperature; in our case, c is the inverse of the temperature parameter. Thus, in some sense, the sharp rise of the first singular value pre-softmax loosely approximates increasing the temperature of the softmax function applied to a rank-1 matrix.

The following section aims to explain the phenomenon formally.

6.7.1. Analysis

This section aims to understand how applying softmax on a matrix $\mathbf{A}$ column-wise changes the spectrum of the matrix and its rank. To this end, we apply the following theorems:

Theorem 4 (Gershgorin Circle Theorem)**.** *Every eigenvalue of any real, symmetric matrix* $\mathbf{S}$ *lies within at least one of the Gershgorin discs* $D(s_{ii}, R_i)$*, where* $R_i = \Sigma_{j\neq i}|s_{ij}|$*.*

Proposition 5. *For any matrix* $\mathbf{M} \in \mathbb{R}^{n\times n}$ *with each column* $\mathbf{m}_j \in \mathbb{R}^n$ *as a probability vector. Then the gap between the largest* $\sigma_1(\mathbf{M})$ *and the smallest singular value* $\sigma_n(\mathbf{M})$ *is bounded by the following tight inequality:*

$$0 \leq \sigma_1(\mathbf{M}) - \sigma_n(\mathbf{M}) \leq \sqrt{1+r} - \sqrt{\max\left\{\frac{1}{n} - r, 0\right\}}$$

where $r := \max_i \sum_{j\neq i} \langle \mathbf{m}_i, \mathbf{m}_j \rangle$*.*

Proof. Consider the matrix $\mathbf{G} := \mathbf{M}^\top \mathbf{M}$. By Jensen's inequality and Cauchy-Schwartz inequality, we have

$$\frac{1}{n} = n\left(\frac{\sum_{i=1}^n \mathbf{M}_{ij}}{n}\right)^2 \leq \mathbf{G}_{jj} = \sum_{i=1}^n \mathbf{M}_{ij}^2 \leq \left(\sum_{i=1}^n \mathbf{M}_{ij}\right)^2 = 1.$$

By definition, $R_j := \sum_{k\neq j} |\langle \mathbf{m}_k, \mathbf{m}_j \rangle| = \sum_{k\neq j} \langle \mathbf{m}_k, \mathbf{m}_j \rangle = \sum_{k\neq j} \mathbf{G}_{kj}$ is the radius of the j-th Gershgorin disc. Hence, Theorem 4, the eigenvalues of $\mathbf{G}$ lie on $\cup_j [\mathbf{G}_{jj} - R_j, \mathbf{G}_{jj} + R_j] \subset \left[\max\left\{\frac{1}{n} - r, 0\right\}, 1 + r\right]$, since $\mathbf{G}$ is positive semidefinite. Note that $\sigma_1(\mathbf{M}) = \sqrt{\lambda_{\max}(\mathbf{G})}$ and $\sigma_n(\mathbf{M}) = \sqrt{\lambda_{\min}(\mathbf{G})}$, we obtain the bound. The bound is tight by considering $\mathbf{M} = \mathbf{I}_n$ for the lower bound and $\mathbf{M} = (\mathbf{1}, 0, ..., 0)$ for the upper bound. □

Intuitively, Proposition 5 shows that reducing the inner products between columns of $\mathbf{M} = \texttt{softmax}(\mathbf{A})$ decreases the gap between its largest and smallest singular values. In particular, the numerical rank of the post-softmax matrix $\mathbf{M}$ can remain high even if the pre-softmax matrix $\mathbf{A}$ is of low rank. Figure 6.8.1 illustrates that the smallest spectral gap occurs at an intermediate temperature.

6.8. Supplementary Figures

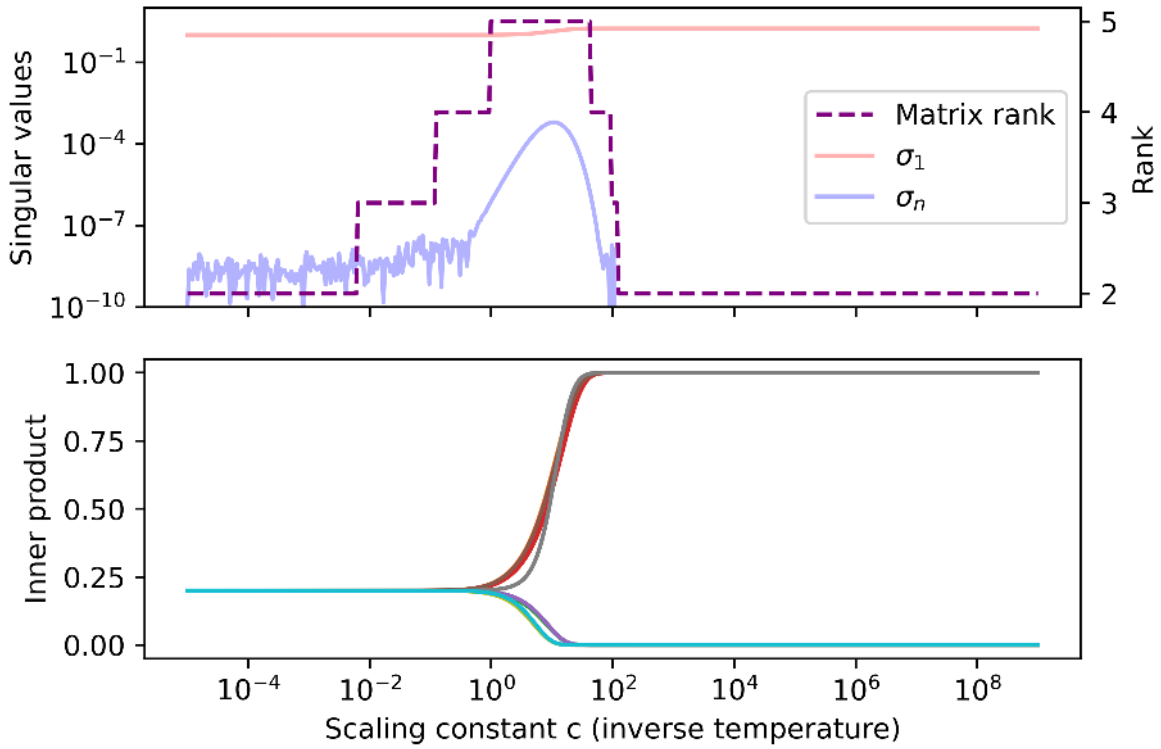


Figure 6.8.1. Applying softmaxcolumn-wise on rank-1 matrix $\mathbf{A} \in \mathbb{R}^{5\times5}$ with various temperatures decreases the inner products between the columns with different indices of the highest element and increases the inner products between the columns with the same indices (bottom). The temperature at which the bifurcation of inner products happens aligns with the temperature that shrinks the gap between the top and bottom singular values of the matrix and increases its rank (top).

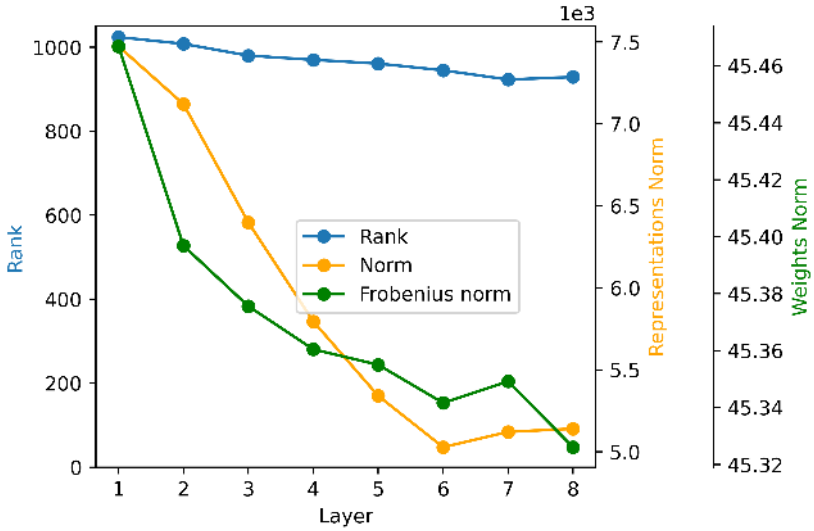


Figure 6.8.2. The plot shows the relationship between the norm of representations, rank of representations, and norm of weights matrices for the model trained with temperature = 1.

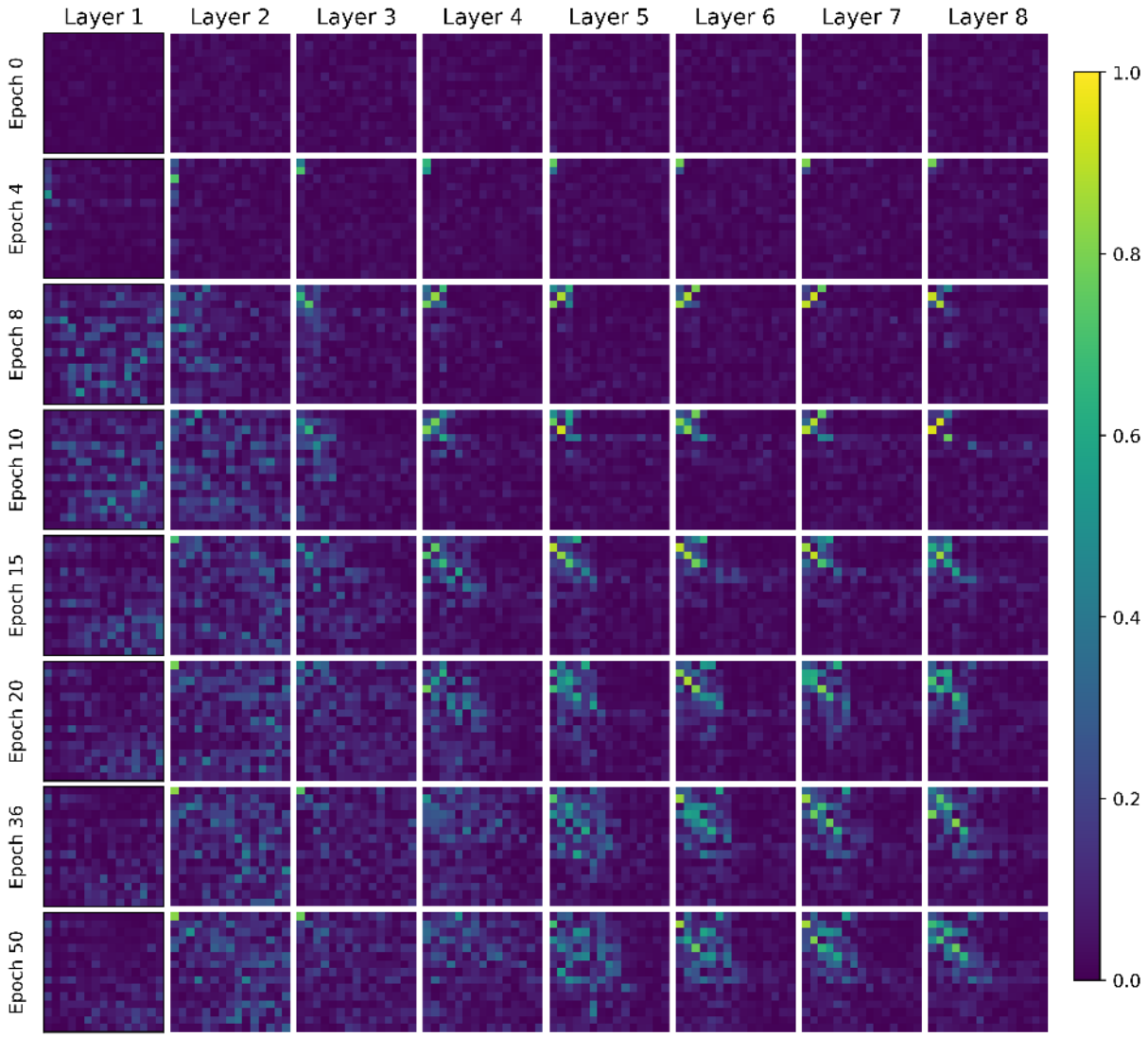


Figure 6.8.3. High-temperature training aligns top singular vectors from consecutive layers.

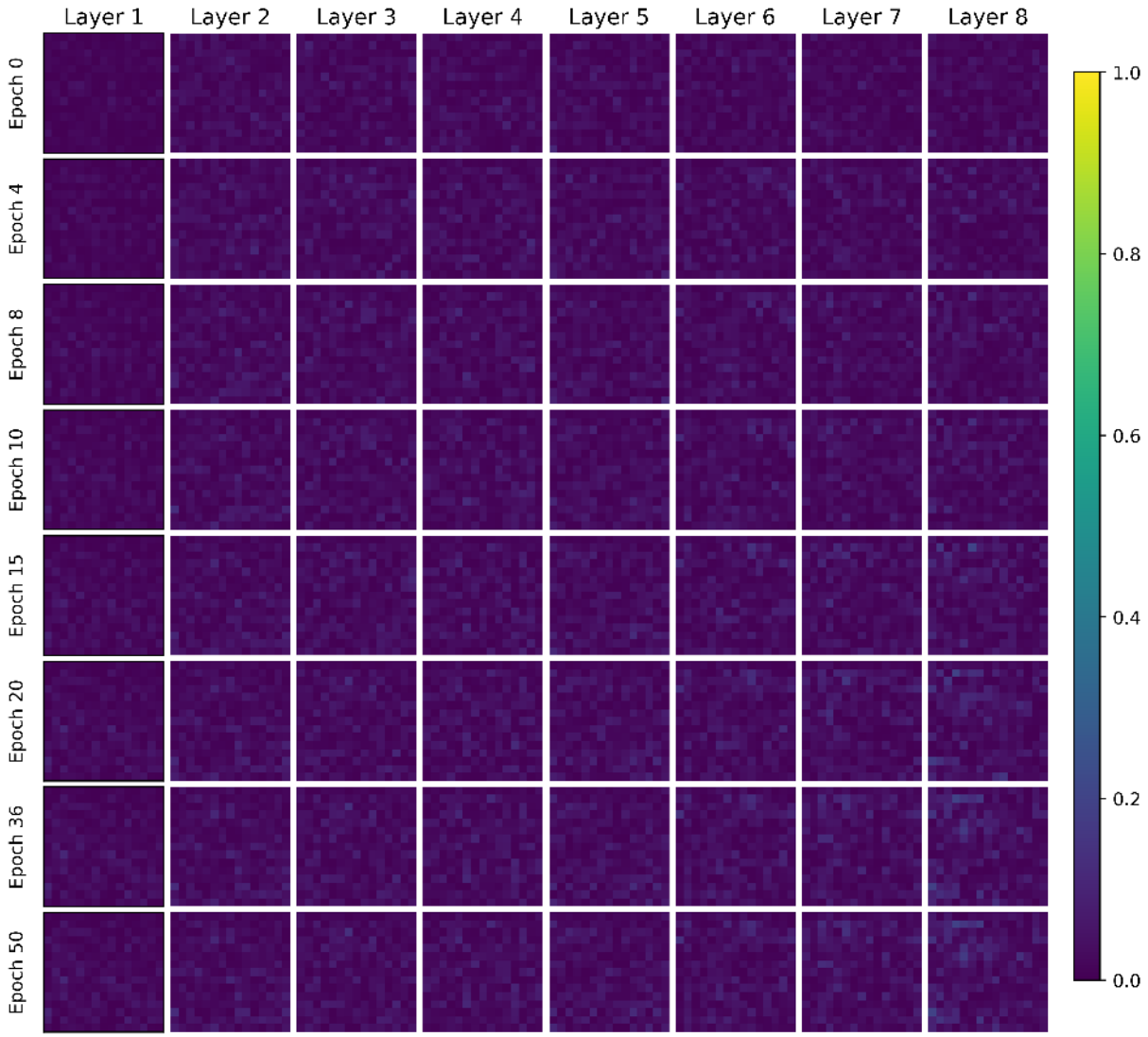


Figure 6.8.4. Training with normal temperature does not align singular vectors between layers.

Bibliography

[1] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[2] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

[3] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun,

Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

[4] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[5] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[6] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[7] Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456):885–890, 2019.

[8] Keith T Butler, Daniel W Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 559(7715):547–555, 2018.

[9] Jonathan Schmidt, Márcio RG Marques, Silvana Botti, and Miguel AL Marques. Recent advances and applications of machine learning in solid-state materials science. *npj Computational Materials*, 5(1):1–36, 2019.

[10] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Au-

gustin **v**Zídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[12] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[13] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

[14] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

[15] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[16] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *arXiv preprint arXiv:2112.10752*, 2022.

[17] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv preprint arXiv:2204.03458*, 2022.

[18] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

[19] Andrea Agostinelli, Timo Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, et al. Musiclm: Generating music from text. *arXiv preprint arXiv:2301.11325*, 2023.

[20] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. 2016.

[22] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, et al. Language models are few-shot learners.

Advances in Neural Information Processing Systems, 33:1877–1901, 2020.

[23] Li Shen et al. On efficient training of large-scale deep learning models: A literature review. *arXiv preprint arXiv:2304.03589*, 2023.

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Léon Bottou. Large-scale machine learning with stochastic gradient descent. *Proceedings of COMPSTAT'2010*, pages 177–186, 2010.

[26] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[27] Shai Shalev-Shwartz and Shai Ben-David. Understanding machine learning: From theory to algorithms. 2014.

[28] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[29] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[30] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017.

[31] Douglas Markant and Todd M Gureckis. Active learning as a means to distinguish among alternative learning strategies. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 36, 2014.

[32] Michael Mccloskey and Neil J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation*, 24:104–169, 1989.

[33] Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 1999.

[34] Raia Hadsell, Dushyant Rao, Andrei Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 24:1028–1040, 12 2020.

[35] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54 – 71, 2019.

[36] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[37] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual Learning Through Synaptic Intelligence. In *ICML*, 2017.

[38] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part IV*, volume 11208 of *Lecture Notes in Computer Science*, pages 72–88. Springer, 2018.

[39] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2990–2999. Curran Associates, Inc., 2017.

[40] Ameya Prabhu, Philip Torr, and Puneet Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *ECCV*, 2020.

[41] **Wojciech Masarczyk** and Ivona Tautkute. Reducing catastrophic forgetting with learning on synthetic data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[42] **Wojciech Masarczyk**, Kamil Deja, and Tomasz Trzciński. On robustness of generative representations against catastrophic forgetting. *International Conference on Neural Information Processing*, 2021.

[43] **Wojciech Masarczyk**, Tomasz Trzciński, and Mateusz Ostaszewski. On consequences of finetuning on data with highly discriminative features. *NeurIPS 2023: UniReps Workshop*, 2023.

[44] **Wojciech Masarczyk**, Mateusz Ostaszewski, Ehsan Imani, Razvan Pascanu, Piotr Miłoś, and Tomasz Trzciński. The tunnel effect: Building data representations in deep neural networks. *NeurIPS*, 2023.

[45] **Wojciech Masarczyk**, Mateusz Ostaszewski, Tin Sum Cheng, Tomasz Trzciński, Aurelien Lucchi, and Razvan Pascanu. Unpacking softmax: How temperature drives rank collapse, compression and generalization. *Under review at ICML 2025*, 2025.

[46] Vinay Venkatesh Ramasesh, Ethan Dyer, and Maithra Raghu. Anatomy of catastrophic forgetting: Hidden representations and task semantics. In *International Conference on Learning Representations*, 2020.

[47] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language un-

derstanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[48] Ivona Tautkute, Tomasz Trzciński, Aleksander P. Skorupa, Łukasz Brocki, and Krzysztof Marasek. Deepstyle: Multimodal search engine for fashion and interior design. *IEEE Access*, 7:84613–84628, 2018.

[49] Wojciech Stokowiec, Tomasz Trzcinski, Krzysztof Wołk, Krzysztof Marasek, and Przemyslaw Rokita. Shallow reading with deep learning: Predicting popularity of online content using only its title. pages 136–145, 07 2017.

[50] Witold Oleszkiewicz, Peter Kairouz, Karol Piczak, Ram Rajagopal, and Tomasz Trzciński. Siamese generative adversarial privatizer for biometric data. In C.V. Jawahar, Hongdong Li, Greg Mori, and Konrad Schindler, editors, *Computer Vision – ACCV 2018*, pages 482–497, Cham, 2019. Springer International Publishing.

[51] Mark Bishop Ring. *Continual Learning in Reinforcement Environments*. PhD thesis, USA, 1994.

[52] Vincenzo Lomonaco. *Continual Learning with Deep Architectures*. PhD thesis, 2018.

[53] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018.

[54] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks, 2016.

[55] David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6467–6476. Curran Associates, Inc., 2017.

[56] Tyler L. Hayes, Nathan D. Cahill, and Christopher Kanan. Memory efficient experience replay for streaming learning. In *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*, pages 9769–9776. IEEE, 2019.

[57] Joseph Cichon and Wen-Biao Gan. Branch-specific dendritic ca2+ spikes cause persistent synaptic plasticity. *Nature*, 520, 03 2015.

[58] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning*

Workshop, 2015.

[59] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018.

[60] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Growing a brain: Fine-tuning by increasing model capacity. *CoRR*, abs/1907.07844, 2019.

[61] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 7765–7773. IEEE Computer Society, 2018.

[62] Siavash Golkar, Michael Kagan, and Kyunghyun Cho. Continual learning via neural pruning. *CoRR*, abs/1903.04476, 2019.

[63] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. Dataset distillation. *CoRR*, abs/1811.10959, 2018.

[64] Chenshen Wu, Luis Herranz, Xialei Liu, yaxing wang, Joost van de Weijer, and Bogdan Raducanu. Memory replay gans: Learning to generate new categories without forgetting. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 5962–5972. Curran Associates, Inc., 2018.

[65] Dougal Maclaurin, David Duvenaud, and Ryan P. Adams. Gradient-based hyperparameter optimization through reversible learning. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 2113–2122. JMLR.org, 2015.

[66] Felipe Petroski Such, Aditya Rawal, Joel Lehman, Kenneth Stanley, and Jeff Clune. Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data, 2020.

[67] Khurram Javed and Martha White. Meta-learning representations for continual learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 1820–1830. Curran Associates, Inc., 2019.

[68] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4652–4662. Curran Associates, Inc., 2017.

[69] Rupesh K Srivastava, Jonathan Masci, Sohrob Kazerounian, Faustino Gomez, and Jürgen Schmidhuber. Compete to compute. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2310–2318. Curran Associates, Inc., 2013.

[70] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler L. Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3390–3398. AAAI Press, 2018.

[71] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[72] Edward Grefenstette, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier, Douwe Kiela, Kyunghyun Cho, and Soumith Chintala. Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727*, 2019.

[73] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

[74] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[75] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

[76] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *PNAS*, 2017.

[77] Gido M van de Ven and Andreas S Tolias. Generative replay with feedback connections as a general strategy for continual learning, 2018. arXiv:1809.10635.

[78] Lucas Caccia, Eugene Belilovsky, Massimo Caccia, and Joelle Pineau. On-

line Learned Continual Compression with Adaptive Quantization Modules. In *ICML*, 2020.

[79] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F Grewe. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, 2019.

[80] Kamil Deja, Paweł Wawrzyński, Daniel Marczak, Wojciech Masarczyk, and Tomasz Trzciński. Binplay: A binary latent autoencoder for generative replay continual learning. In *IJCNN*, 2021.

[81] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual Learning with Deep Generative Replay. In *NeurIPS*, 2017.

[82] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience Replay for Continual Learning. In *NeurIPS*, 2019.

[83] Nicolas Y. Masse, Gregory D. Grant, and David J. Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *PNAS*, 2018.

[84] Arun Mallya and Svetlana Lazebnik. Packnet: Adding Multiple Tasks to a Single Network by Iterative Pruning. In *CVPR*, 2018.

[85] Siavash Golkar, Michael Kagan, and Kyunghyun Cho. Continual Learning via Neural Pruning. In *Neuro AI. Workshop at NeurIPS*, 2019.

[86] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a Single Network to Multiple Tasks by Learning to Mask Weights. In *ECCV*, 2018.

[87] Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in Superposition. In *NeurIPS*, 2020.

[88] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks, 2016. arXiv:1606.04671.

[89] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong Learning with Dynamically Expandable Networks. In *ICLR*, 2018.

[90] Anh Thai, Stefan Stojanov, Isaac Rehg, and James M. Rehg. Does continual learning = catastrophic forgetting? *arXiv*, 2021.

[91] Guy Davidson and Michael C. Mozer. Sequential mastery of multiple visual tasks: Networks naturally learn to learn and forget to forget. In *CVPR*, 2020.

[92] Vinay Venkatesh Ramasesh, Ethan Dyer, and Maithra Raghu. Anatomy of catastrophic forgetting: Hidden representations and task semantics.

In *ICLR*, 2021.

[93] Eli Verwimp, Matthias De Lange, and Tinne Tuytelaars. Rehearsal revealed: The limits and merits of revisiting samples in continual learning. 2021.

[94] Ju Xu and Zhanxing Zhu. Reinforced Continual Learning. In *NeurIPS*, 2018.

[95] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online Continual Learning with Maximally Interfered Retrieval. In *NeurIPS*, 2019.

[96] Giang Nguyen, Shuan Chen, Thao Do, Tae Joon Jun, Ho-Jin Choi, and Daeyoung Kim. Dissecting catastrophic forgetting in continual learning by deep visualization. *arXiv*, 2020.

[97] Cuong V. Nguyen, Alessandro Achille, Michael Lam, Tal Hassner, Vijay Mahadevan, and Stefano Soatto. Toward understanding catastrophic forgetting in continual learning. *arXiv*, 2019.

[98] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *ICLR*, 2014.

[99] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *ICML*, 2019.

[100] Ying Nian Wu, Ruiqi Gao, Tian Han, and Song-Chun Zhu. A tale of three probabilistic families: Discriminative, descriptive and generative models, 2018.

[101] Lorien Pratt and Barbara Jennings. A survey of transfer between connectionist networks. *Connection Science*, 1996.

[102] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 2014.

[103] R. Anil et al. Palm 2 technical report, 2023.

[104] OpenAI. Gpt-4 technical report, 2023.

[105] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers, 2022.

[106] M. Dehghani et al. Scaling vision transformers to 22 billion parameters, 2023.

[107] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks, 2017.

[108] Guillermo Valle-Pérez, Chico Q. Camargo, and Ard A. Louis. Deep learning

generalizes because the parameter-function map is biased towards simple functions, 2019.

[109] Preetum Nakkiran, Gal Kaplun, Dimitris Kalimeris, Tristan Yang, Benjamin L. Edelman, Fred Zhang, and Boaz Barak. Sgd on neural networks learns functions of increasing complexity, 2019.

[110] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *International Conference on Learning Representations*, 2018.

[111] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton. Similarity of neural network representations revisited. *International Conference on Machine Learning*, 2019.

[112] Depen Morwani, Praneeth Netrapalli, jatin batra, Karthikeyan Shanmugam, and Prateek Jain. Simplicity bias in 1-hidden layer neural networks, 2023.

[113] Lukas Braun, Clémentine Dominé, James Fitzgerald, and Andrew Saxe. Exact learning dynamics of deep linear networks with prior knowledge. *Advances in Neural Information Processing Systems*, 2022.

[114] Jiefeng Chen, Timothy Nguyen, Dilan Gorur, and Arslan Chaudhry. Is forgetting less a good inductive bias for forward transfer? In *The Eleventh International Conference on Learning Representations*, 2022.

[115] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

[116] Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. *Advances in Neural Information Processing Systems*, 2020.

[117] Mohammad Pezeshki, Oumar Kaba, Yoshua Bengio, Aaron C Courville, Doina Precup, and Guillaume Lajoie. Gradient starvation: A learning proclivity in neural networks. *Advances in Neural Information Processing Systems*, 2021.

[118] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, 2021.

[119] Mengnan Du, Fan Yang, Na Zou, and Xia Hu. Fairness in deep learning: A computational perspective. *IEEE Intelligent Systems*, 2020.

[120] Max Klabunde, Tobias Schumacher, Markus Strohmaier, and Florian Lemmerich. Similarity of neural network models: A survey of functional

and representational measures, 2023.

[121] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012.

[122] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 2018.

[123] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017.

[124] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, 2014.

[125] Matus Telgarsky. Representation benefits of deep feedforward networks, 2015.

[126] Guillermo Valle-Perez, Chico Q. Camargo, and Ard A Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions, 2018.

[127] Boris Hanin and David Rolnick. Complexity of linear regions in deep networks, 2019.

[128] Minyoung Huh, Hossein Mobahi, Richard Zhang, Brian Cheung, Pulkit Agrawal, and Phillip Isola. The low-rank simplicity bias in deep networks, 2021.

[129] Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 2020.

[130] Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *International Conference on Learning Representations*, 2020.

[131] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 2019.

[132] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[133] Greg Yang and Hadi Salman. A fine-grained spectral perspective on neural

networks, 2019.

[134] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020.

[135] Chiyuan Zhang, Samy Bengio, and Yoram Singer. Are all layers created equal?, 2022.

[136] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Transfer learning for music classification and regression tasks, 2017.

[137] Jongpil Lee and Juhan Nam. Multi-level and multi-scale feature aggregation using pretrained convolutional neural networks for music auto-tagging. *IEEE Signal Processing Letters*, 2017.

[138] Joel Shor, Aren Jansen, Ronnie Maor, Oran Lang, Omry Tuval, Félix de Chaumont Quitry, Marco Tagliasacchi, Ira Shavitt, Dotan Emanuel, and Yinnon Haviv. Towards learning a universal non-semantic representation of speech, 2020.

[139] Utku Evci, Vincent Dumoulin, Hugo Larochelle, and Michael C. Mozer. Head2toe: Utilizing intermediate representations for better transfer learning, 2022.

[140] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2019.

[141] Yizhou Wang, Shixiang Tang, Feng Zhu, Lei Bai, Rui Zhao, Donglian Qi, and Wanli Ouyang. Revisiting the transferability of supervised pretraining: an mlp perspective. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[142] Seyed Iman Mirzadeh, Arslan Chaudhry, Dong Yin, Huiyi Hu, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar. Wide neural networks forget less catastrophically. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

[143] Seyed Iman Mirzadeh, Arslan Chaudhry, Dong Yin, Timothy Nguyen, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar. Architecture matters in continual learning, 2022.

[144] Alex H Williams, Erin Kunz, Simon Kornblith, and Scott Linderman. Generalized shape metrics on neural representations. *Advances in Neural Information Processing Systems*, 2021.

[145] Yizhang Lou, Chris E Mingard, and Soufiane Hayou. Feature learning and signal propagation in deep neural networks. In *International Conference on Machine Learning*, 2022.

[146] Niru Maheswaranathan, Alex Williams, Matthew Golub, Surya Ganguli,

and David Sussillo. Universality and individuality in neural dynamics across large populations of recurrent networks. *Advances in neural information processing systems*, 32, 2019.

[147] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability, 2017.

[148] Jessica AF Thompson, Yoshua Bengio, and Marc Schönwiesner. The effect of task and training on intermediate representations in convolutional neural networks revealed with modified rv similarity analysis, 2019.

[149] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 2018.

[150] Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020.

[151] Colin Wei, Jason D Lee, Qiang Liu, and Tengyu Ma. Regularization matters: Generalization and optimization of neural nets vs their induced kernel. *Advances in Neural Information Processing Systems*, 32, 2019.

[152] Yixiong Chen, Alan Yuille, and Zongwei Zhou. Which layer is learning faster? a systematic exploration of layer-wise convergence rate for deep neural networks. In *The Eleventh International Conference on Learning Representations*, 2023.

[153] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, 2020.

[154] Seyed Iman Mirzadeh, Arslan Chaudhry, Dong Yin, Huiyi Hu, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar. Wide neural networks forget less catastrophically. In *International Conference on Machine Learning*, 2022.

[155] Sebastian Lee, Stefano Sarao Mannelli, Claudia Clopath, Sebastian Goldt, and Andrew M. Saxe. Maslow's hammer in catastrophic forgetting: Node re-use vs. node activation. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, Proceedings of Machine Learning Research. PMLR, 2022.

[156] Sebastian Lee, Sebastian Goldt, and Andrew Saxe. Continual learning in the teacher-student setup: Impact of task similarity. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 2021.

[157] Itay Evron, Edward Moroshko, Rachel Ward, Nathan Srebro, and Daniel Soudry. How catastrophic can catastrophic forgetting be in linear regression? In *Proceedings of Thirty Fifth Conference on Learning Theory*, 2022.

[158] Wesley J. Maddox, Shuai Tang, Pablo Garcia Moreno, Andrew Gordon Wilson, and Andreas Damianou. Fast adaptation with linearized neural networks, 2021.

[159] Thao Nguyen, Maithra Raghu, and Simon Kornblith. On the origins of the block structure phenomenon in neural network representations. *Transactions on Machine Learning Research*, 2022.

[160] Zhihui Zhu, Tianyu Ding, Jinxin Zhou, Xiao Li, Chong You, Jeremias Sulam, and Qing Qu. A geometric analysis of neural collapse with unconstrained features. *Advances in Neural Information Processing Systems*, 34, 2021.

[161] Tomer Galanti, András György, and Marcus Hutter. On the role of neural collapse in transfer learning. In *International Conference on Learning Representations*, 2022.

[162] Like Hui, Mikhail Belkin, and Preetum Nakkiran. Limitations of neural collapse for understanding generalization in deep learning. *arXiv preprint arXiv:2202.08384*, 2022.

[163] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32, 2019.

[164] Raj Dabre and Atsushi Fujita. Recurrent stacking of layers for compact neural machine translation models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6292–6299, 2019.

[165] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Trellis networks for sequence modeling. *arXiv preprint arXiv:1810.06682*, 2018.

[166] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

[167] Ari Morcos, Haonan Yu, Michela Paganini, and Yuandong Tian. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. *Advances in neural information processing systems*, 32, 2019.

[168] Andrew M Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D Tracey, and David D Cox. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020, 2019.

[169] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[170] Alessio Ansuini, Alessandro Laio, Jakob H Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[171] Xiao Li, Sheng Liu, Jinxin Zhou, Xinyu Lu, Carlos Fernandez-Granda, Zhihui Zhu, and Qing Qu. Principled and efficient transfer learning of deep models via neural collapse. *arXiv preprint arXiv:2212.12206*, 2022.

[172] Akshay Rangamani, Marius Lindegaard, Tomer Galanti, and Tomaso A Poggio. Feature learning in deep classifiers through intermediate neural collapse. In *International Conference on Machine Learning*, pages 28729–28745. PMLR, 2023.

[173] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[174] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[175] George Bebis and Michael Georgiopoulos. Feed-forward neural networks. *Ieee Potentials*, 13(4):27–31, 1994.

[176] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning, 2019.

[177] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, 2009.

[178] Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

[179] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

[180] M-E. "Nilsback and A." Zisserman. "automated flower classification over a large number of classes". In *"Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing"*, "2008".

[181] O. M. "Parkhi, A. Vedaldi, A. Zisserman, and C. V." Jawahar. "cats and dogs". In *"IEEE Conference on Computer Vision and Pattern Recognition"*, "2012".

[182] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, 2017.

[183] Adam Coates, Andrew Ng, and Honglak Lee. An Analysis of Single Layer Networks in Unsupervised Feature Learning. In *AISTATS*, 2011. https://cs.stanford.edu/~acoates/papers/coatesleeng_aistats_2011.pdf.

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

[185] Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. Adversarial continual learning, 2020.

[186] Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent replay for real-time continual learning, 2020.

[187] Suresh Balakrishnama and Aravind Ganapathiraju. Linear discriminant analysis-a brief tutorial, 1998.

[188] Ashraful Islam, Chun-Fu Richard Chen, Rameswar Panda, Leonid Karlinsky, Richard Radke, and Rogerio Feris. A broad study on the transferability of visual representations with contrastive learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8845–8855, 2021.

[189] Nanxuan Zhao, Zhirong Wu, Rynson WH Lau, and Stephen Lin. What makes instance discrimination good for transfer learning?, 2020.

[190] Shih-Chii Liu. A winner-take-all circuit with controllable soft max property. *Advances in neural information processing systems*, 12, 1999.

[191] Ibrahim M Elfadel and John L Wyatt Jr. The" softmax" nonlinearity: Derivation using statistical mechanics and useful properties as a multiterminal analog circuit element. *Advances in neural information processing systems*, 6, 1993.

[192] Carsten Peterson and Bo Söderberg. A new method for mapping optimization problems onto neural networks. *International Journal of Neural Systems*, 01(01):3–22, 1989.

[193] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. *arXiv preprint arXiv:2309.16588*, 2023.

[194] David T. Hoffmann, Simon Schrodi, Nadine Behrmann, Volker Fischer, and Thomas Brox. Eureka-moments in transformers: Multi-step tasks reveal softmax induced optimization problems. *arXiv preprint arXiv:2310.12956*, 2023.

[195] Kai Shen, Junliang Guo, Xu Tan, Siliang Tang, Rui Wang, and Jiang Bian. A study on relu and softmax in transformer. *arXiv preprint*

arXiv:2302.06461, 2023.

[196] Shuangfei Zhai, Tatiana Likhomanenko, Etai Littwin, Dan Busbridge, Jason Ramapuram, Yizhe Zhang, Jiatao Gu, and Josh Susskind. Stabilizing transformer training by preventing attention entropy collapse. *ICML*, 2023.

[197] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[198] Hemanth Saratchandran, Jianqiao Zheng, Yiping Ji, Wenbo Zhang, and Simon Lucey. Rethinking softmax: Self-attention with polynomial activations. *arXiv preprint arXiv:2410.18613*, 2024.

[199] Jason Ramapuram, Federico Danieli, Eeshan Dhekane, Floris Weers, Dan Busbridge, Pierre Ablin, Tatiana Likhomanenko, Jagrit Digani, Zijin Gu, Amitis Shidani, et al. Theory, analysis, and best practices for sigmoid self-attention. *arXiv preprint arXiv:2409.04431*, 2024.

[200] Petar Veli**v**cković, Christos Perivolaropoulos, Federico Barbero, and Razvan Pascanu. softmax is not enough (for sharp out-of-distribution). *arXiv preprint arXiv:2410.01104*, 2024.

[201] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[202] Arthur Jacot, Fran**c**cois Ged, Berfin **c**Sim**c**sek, Clément Hongler, and Franck Gabriel. Saddle-to-saddle dynamics in deep linear networks: Small initialization training, symmetry, and sparsity. *arXiv preprint arXiv:2106.15933*, 2021.

[203] Zhiyuan Li, Yuping Luo, and Kaifeng Lyu. Towards resolving the implicit bias of gradient descent for matrix factorization: Greedy low-rank learning. *arXiv preprint arXiv:2012.09839*, 2020.

[204] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[205] Wojciech Masarczyk, Mateusz Ostaszewski, Ehsan Imani, Razvan Pascanu, Piotr Miłoś, and Tomasz Trzcinski. The tunnel effect: Building data representations in deep neural networks. *Advances in Neural Information Processing Systems*, 36, 2024.

[206] Md Yousuf Harun, Kyungbok Lee, Jhair Gallardo, Giri Krishnan, and Christopher Kanan. What variables affect out-of-distribution generalization in pretrained models? *arXiv preprint arXiv:2405.15018*, 2024.

[207] Chongyi Zheng, Benjamin Eysenbach, Homer Rich Walke, Patrick Yin, Kuan Fang, Ruslan Salakhutdinov, and Sergey Levine. Stabilizing contrastive RL: Techniques for robotic goal reaching from offline data. In *The Twelfth International Conference on Learning Representations*, 2024.

[208] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[209] Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. In *International Conference on Learning Representations*, 2022.

[210] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv: 1607.06450*, 2016.

[211] Hadi Daneshmand, Jonas Kohler, Francis Bach, Thomas Hofmann, and Aurelien Lucchi. Batch normalization provably avoids rank collapse for randomly initialised deep networks. *arXiv preprint arXiv: 2003.01652*, 2020.

[212] Ziwei Ji and Matus Telgarsky. Gradient descent aligns the layers of deep linear networks. *arXiv preprint arXiv:1810.02032*, 2018.

[213] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *International conference on machine learning*, pages 244–253. PMLR, 2018.

[214] Shuangfei Zhai, Tatiana Likhomanenko, Etai Littwin, Dan Busbridge, Jason Ramapuram, Yizhe Zhang, Jiatao Gu, and Joshua M Susskind. Stabilizing transformer training by preventing attention entropy collapse. In *International Conference on Machine Learning*, pages 40770–40803. PMLR, 2023.

[215] Vardan Papyan, X. Y. Han, and David L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

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

[217] Branton DeMoss, Silvia Sapora, Jakob Foerster, Nick Hawes, and Ingmar Posner. The complexity dynamics of grokking. *arXiv preprint arXiv:2412.09810*, 2024.

[218] Ilse CF Ipsen and Arvind K Saibaba. Stable rank and intrinsic dimension of real and complex matrices. *arXiv preprint arXiv:2407.21594*, 2024.