Exploring Layer Freezing in Transfer Learning

A Deep Neural Networks Study

Masters Project

Arnisa Fazla, David Guzman Piedrahita and Tushar Deo Manekar

22-740-039, 22-737-571, 22-740-179

Submitted on

June 1 2024

Project Supervisor

Prof. Dr. Manuel Günther

Co-Supervisor

Prof. Dr. Benjamin Grewe

Pau Vilimelis Aceituno

University of Zurich

Department of Informatics

Declaration of Independence for Written Work

I hereby declare that I have **composed** this work independently and without the use of any aids other than those declared (including generative AI such as ChatGPT) – the use of generative AI to **improve** my composed work was permitted by the thesis supervisor. I am aware that I take full responsibility for the scientific character of the submitted text myself, even if AI aids were used. All passages taken verbatim or in sense from published or unpublished writings are identified as such. The work has not yet been submitted in the same or similar form or in excerpts as part of another examination.

Zurich, Switzerland, June 1 2024

Place, Date

Arnisa Fazla, David Guzman Piedrahita and Tushar Deo Manekar

Masters Project

Author: Arnisa Fazla, David Guzman Piedrahita and Tushar Deo Manekar

Project period: January 1 2024 - June 1 2024

Artificial Intelligence and Machine Learning Group
Department of Informatics, University of Zurich

Acknowledgements

We wish to extend our heartfelt thanks to Prof. Dr. Manuel Günther for his invaluable expertise and guidance throughout this project.

We are also grateful to Prof. Dr. Benjamin Grewe from the Institute of Neuroinformatics at the University of Zurich and ETH Zurich, and to Pau Vilimelis Aceituno, a Postdoctoral Researcher at the same institute, for their valuable insights and comments.

Abstract

This project investigates the effectiveness of transfer learning by exploring various fine-tuning design choices, including freezing earlier layers, reinitializing or truncating later layers, and determining the subset of layers of the pre-trained model to retain during fine-tuning. The study explores optimal strategies for adapting pre-trained models to new tasks across the MNIST, FashionMNIST and CIFAR10 datasets. We conduct systematic experiments across these datasets in which we vary the amount of target data and number of layers to transfer. Experiments reveal a pattern consistent across mutiple source and target dataset combinations, where the optimal number of pre-trained layers to retain is reduced with increasing target data, indicating a reduced reliance on pre-trained features as more task-specific data becomes available. Key findings also highlight that completely removing some of the final layers offers a viable, more parameter-efficient alternative. Additionally, we propose a clustering based approach to identify layers that contain more transferable features. This study offers a framework for making informed fine-tuning decisions, emphasizing the importance of design choices in transfer learning.

Contents

Chapter 1

Introduction

In today's information age, where vast amounts of data are generated daily, machine learning, particularly deep learning, has emerged as a potent tool for extracting valuable insights. However, training deep learning models from scratch can be impractical due to the substantial computational resources and extensive labeled data required. Hence, transfer learning, which allows leveraging knowledge from one task to address another related task, has become an indispensable technique in overcoming these challenges.

Transfer learning involves utilizing a model trained on a source dataset for a source task and applying it to a target dataset for a target task. The source dataset and task are typically well-studied, with ample labeled data available, whereas the target dataset and task may have limited labeled data. The goal of transfer learning is to improve the performance on the target task by transferring the knowledge gained from the source task.

Despite its common use, it remains a blend of art and science due to the empirical nature of its practices. The decision process behind it, often guided by trial and error and experience, can lead to inconsistencies. For this reason, there is a need for a more rigorous understanding of the process.

Various factors significantly influence the efficacy of transfer learning. These factors cover various domains, including source and target dataset characteristics. For instance, factors like the similarity between the pre-train and fine-tune data and the quantity of available fine-tune data play significant roles. Moreover, aspects related to the pre-training process and model architecture, such as the transferability of information learned at each layer, significantly impact the effectiveness of transfer learning. Transferability here refers to the extent to which the information learned during pre-training can be applied and be beneficial during the fine-tuning phase.

The transferability of features is closely tied to the model architecture. In the field of computer vision, the initial or lowest layers tend to learn more abstract or low-level features, as shown by Yosinski et al. (2014). These initial-layer features are referred to as *general* because they are commonly found regardless of the specific dataset or task. Conversely, the features computed by the final layers of a trained network are highly dependent on the chosen dataset and task and are thus refered to as *specific*. Earlier layers capture more general features, progressively transitioning to more task-specific features with each subsequent layer. This observation forms the basis for layer-wise transfer learning in deep neural networks. In this approach, the parameters of the initial layers are frozen and reused, while the rest are reinitialized. The model is then fine-tuned on the dataset of the target task.

This progressive specialization of features suggests that, as we move through the layers, their transferability from the source dataset to the target dataset is progressively reduced. By finding a suitable layer up to which there are features that happen to be transferable to the target task, we utilize only the pre-trained layers up to this point, in order to maximize the efficacy of the transfer learning task.

In our experiments, we evaluate many such points to determine the best performing one in terms of test accuracy in the target task. We refer to these points as the *cut points* throughout this project, where the best performing one is the *optimal cut point*. Its selection is a key factor influencing the efficacy of transfer learning and encompasses several other factors. For instance, opting for a cut point in the later layers of the model entails freezing most or all of the convolutional layers, resulting in fewer trainable parameters. While this may aid in preventing overfitting, it also retains layers containing more source dataset-specific information, potentially hindering the model's adaptation to the target dataset and impeding convergence.

In addition to selecting the cut point, there are several other design choices to be made when implementing transfer learning. One such decision is whether to freeze the layers preceding the cut point, thereby utilizing them as a fixed feature extractor, or to leave them trainable, allowing them to adapt to the target dataset during fine-tuning. Throughout this report, we refer to these options as *freeze=True* and *freeze=False*, respectively.

Another important consideration is how to handle the layers subsequent to the cut point. One approach is to reinitialize these layers, effectively discarding the pre-trained weights and biases, while another is to retain the pre-trained parameters and fine-tune them to the target dataset. We will denote these options as *reinit=True* and *reinit=False*, respectively.

Our study focuses on observing the behavior and performance of fine-tuning in transfer learning, considering various design choices, through systematic experiments. We analyze how model performance interacts with key factors such as the volume of data available for the target task and the number of trainable parameters, as well as the dissimilarity across source and target datasets. By doing so, we provide insights into the selection of the optimal combination of design choices, that can reduce the need for extensive trial and error. Additionally, we contextualize our findings with baselines that were trained end-to-end in the respective target tasks, which help determine whether transfer learning is beneficial or even detrimental in a case by case basis.

To provide a clear roadmap for our investigation into transfer learning, the structure of this report is as follows:

In Chapter 2, we present the Background information, covering foundational concepts in deep learning, including fully connected neural networks and convolutional neural networks. We then delve into the principles of transfer learning, discussing its challenges, the fine-tuning process, and specific applications in image classification.

Chapter 3 outlines our Approach. We discuss choices such as dataset selection, model architecture, and training hyperparameters. This chapter also details our experimental setup, emphasizing key procedures such as determining the optimal cut, reinitializing, freezing, and truncating layers. We also highlight our methods to ensure reproducibility and objectivity in our experiments.

The Experimental Results are presented in Chapter 4, where we provide detailed findings from various experiments involving different datasets like MNIST (Deng, 2012), FashionMNIST (Xiao et al., 2017), and CIFAR10 (Krizhevsky, 2012). This section includes specific experiments such as reinitializing layers, truncation, and clustering, providing comprehensive insights into the impact of different design choices.

In Chapter 5, we discuss our findings, interpreting the results and their implications for transfer learning. This leads into Chapter 6, where we draw our conclusions, summarizing the key takeaways from our study and offering directions for further research.

The appendices provide additional details, including statistical test outputs for various datasets, ensuring a thorough understanding of our experimental results.

Chapter 2

Background

2.1 Deep Learning

Deep learning, a subset of machine learning, has revolutionized the field of artificial intelligence by enabling models to automatically learn and extract features from vast amounts of data. This approach utilizes artificial neural networks with multiple layers, or deep architectures, to capture complex patterns and representations. The success of deep learning has been demonstrated across numerous domains, including computer vision, natural language processing, and speech recognition. By leveraging large datasets and high computational power, deep learning models have achieved state-of-the-art performance in tasks that were previously considered intractable, marking a significant advancement in the development of intelligent systems.

These networks learn patterns from input data through a process called training, which involves adjusting the weights of the neural connections based on input-output pairs. The foundational work on gradient-based learning highlights the significance of this process in enabling neural networks to recognize patterns and perform complex tasks, such as document recognition (Rumelhart et al., 1986), (Lecun et al., 1998).

During training, the neural network processes input data through multiple layers of neurons, each layer applying a transformation to the input and then the linear network combines and scales the input data directly, aiming to find the optimal weights that minimize the error between the predicted and actual outputs.

The learning process is guided by a loss function that quantifies the difference between the network's predictions and the actual outcomes. By minimizing this loss through gradient descent and optimization methods such as Adam (Kingma and Ba, 2014), the network iteratively updates its weights, improving its ability to recognize patterns and make accurate predictions.

This learning approach enables deep neural networks to model complex, non-linear relationships within the data, making them powerful tools for a wide range of applications. In classification tasks, these networks aim to assign input data to predefined classes based on learned features. The input layer receives the raw data, which passes through hidden layers where processing occurs via weighted connections. These weights are adjusted during training to minimize the difference between predicted outcomes and actual target values, a principle fundamental to supervised learning (Mehmood et al., 2023). Finally, all neurons in previous layers are connected to each neuron in fully connected layers, which assemble the features into more global patterns. The final layer typically uses a softmax activation function to convert the raw scores, called logits, into probabilities by comparing how likely the input data point belongs to one class versus another (Shao and Wang, 2023).

Overfitting and underfitting are to be considered in training deep learning models. Overfitting occurs when a model learns not only the underlying patterns in the training data but

also the noise, resulting in poor generalization to new data (Srivastava et al., 2014), (Salman and Liu, 2019a). Regularization methods, such as batch normalization (Ioffe and Szegedy, 2015) and dropout (Kim et al., 2023), are commonly used to prevent overfitting in neural networks. Early stopping is another method to prevent overfitting that halts the training once performance on a validation set starts to degrade, ensuring that the model retains its ability to generalize to unseen data. Employing a validation split of around 10% is a common practice to monitor and improve model performance during training (Bai et al., 2021), (Cao et al., 2022).

Conversely, underfitting occurs when a model is too simple to capture the underlying structure of the data. Increasing model complexity can help address underfitting by allowing the model to learn more intricate patterns within the data (Salman and Liu, 2019b).

2.1.1 Fully Connected Neural Networks

Fully Connected Neural Networks (FCNNs) (Scabini and Bruno, 2023), also known as dense networks, are a fundamental type of architecture prominently used in deep learning for various classification tasks. These networks are characterized by their structure, where every neuron in one layer is connected to all neurons in the subsequent layer as illustrated in Figure 2.1.

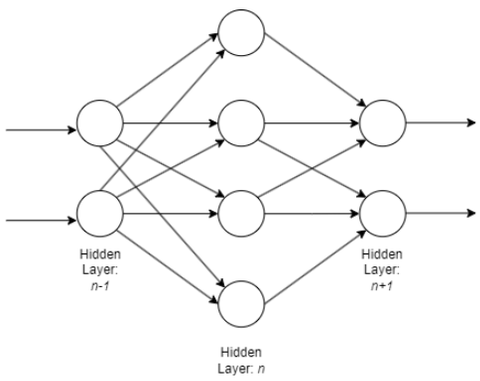


Figure 2.1: FCNN ARCHITECTURE. *Diagram illustrating the architecture of the hidden layers of a FCNN.*

Each neuron in these layers applies a non-linear activation function to the weighted sum of its inputs, such as the sigmoid (Xu et al., 2021) which is defined by:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

or ReLU (Nair and Hinton, 2010) function, which is defined by:

$$\text{ReLU}(x) = \max(0, x) \tag{2.2}$$

This introduces non-linearity into the model, allowing it to capture complex relationships in the data.

After processing through the hidden layers, the final layer, typically a softmax layer in classification tasks, produces the output probabilities corresponding to each class (Bishop, 2006). The softmax function maps the accumulated outputs of the network into a probabilistic distribution over predicted output classes.

2.1.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) (Krizhevsky et al., 2012) have revolutionized the field of image classification through their ability to automatically learn hierarchical feature representations from visual data. Designed to process data with a grid-like topology, such as images, CNNs efficiently handle the high dimensionality of raw images by employing convolutional layers, which filter inputs for useful information while reducing dimensionality.

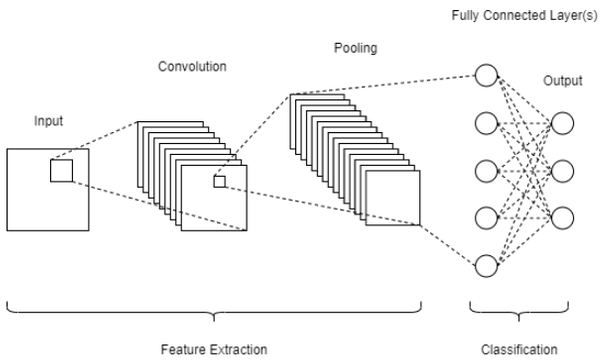


Figure 2.2: CNN ARCHITECTURE. *Diagram illustrating the architecture of a Convolutional Neural Network.*

The architecture of a CNN (illustrated in Figure 2.2 typically starts with a layer that receives the raw image data. This is followed by multiple convolutional layers, where each layer applies a set of learnable filters to the input. These filters are small spatially (e.g., 3x3, 5x5, 7x7) but extend through the full depth of the input volume. As these filters slide over the image, they create feature maps that capture spatial hierarchies of features such as edges at the lower layers, and more complex patterns like textures and object parts at higher layers, as illustrated in Figure 2.3.

Following the convolutional layers, pooling layers reduce the spatial size of the feature maps, thus lowering the number of parameters and computation in the network. This operation not only helps in making the detection of features invariant to scale and orientation changes but also improves the network's robustness to noise and distortions (Lecun et al., 1998), (Cun et al., 1990), (Sermanet et al., 2012).

After several convolutional and pooling layers, the final step in the classification is done via one fully connected layer (modern architectures use a single fully connected layer (Basha et al., 2020), however, in older architectures more than one fully connected layers have also been used). This methodical feature learning and reduction of spatial dimension make CNNs exceptionally good at recognizing patterns in the face of distortions and transformations, which is critical in image classification tasks.

2.2 Transfer Learning

Transfer learning is a research area in machine learning that explores the idea of harnessing knowledge gained from solving one problem (the source task) and applying it to a distinct yet related problem (the target task). This method has garnered substantial attention in various fields for its capacity to utilize the latent representations acquired in models, thereby achieving notable

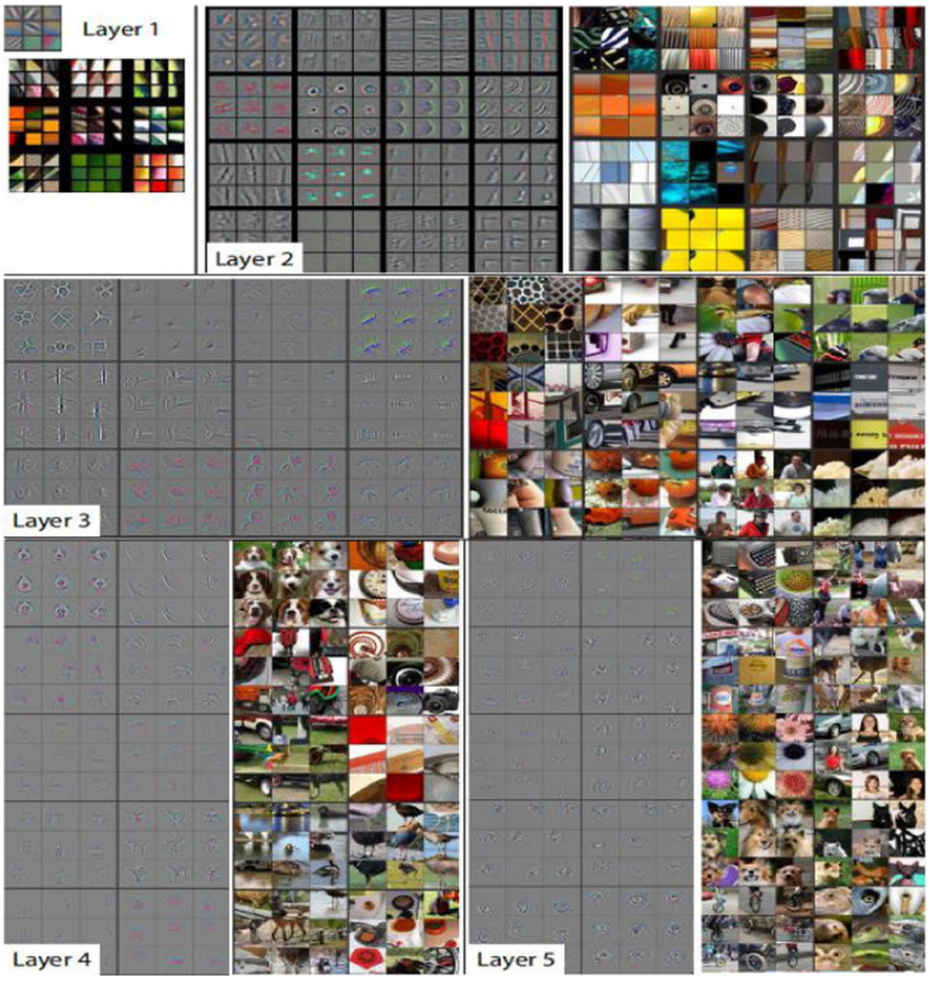


Figure 2.3: HIERARCHICAL FEATURE LEARNING IN A CONVOLUTIONAL NEURAL NETWORK. *Adapted from Zeiler and Fergus (2014).*

performance with limited data on a specific task (Zhuang et al., 2020), (He et al., 2015). The concept is particularly beneficial in scenarios where the target task has limited labeled data available. An early comprehensive review by Zhuang et al. (2020) highlighted the various strategies and potential of transfer learning in improving learning efficiency and performance across diverse applications.

A major obstacle in Transfer Learning is negative transfer (Wang et al., 2019), where knowledge transferred from a less related source task can actually degrade performance on the target task. The underlying cause of negative transfer is the mismatch between the joint distributions of the source and target domains. When the source and target tasks are not closely related, the transferred knowledge may be irrelevant or even interfere with learning the target task, leading to poorer performance compared to training solely on the target data.

2.2.1 Fine-tuning in Transfer Learning

Fine-tuning is a critical technique within transfer learning, which involves adapting a pre-trained model to the target task. Typically, this is done by continuing the training process of a pre-trained model on a new dataset, allowing the model to adjust its weights from generic feature representations to those pertinent to the specific task. This method is particularly effective when the new task has limited training data, as the pre-existing knowledge in the model can significantly boost learning efficiency and performance (Liu et al., 2022).

Truncation is a strategy, in transfer learning, involves removing the network layers beyond a certain point, rather than keeping them. The remaining layers, which are more task-specific, are discarded (as show in Figure 3.6) to reduce model complexity and improve computational efficiency (Montalbo, 2021), (Das et al., 2020).

2.2.2 Transfer Learning in Image Classification

In the domain of image classification, transfer learning, particularly through fine-tuning, has proven to be beneficial for performance, particularly in low-resource scenarios. Utilizing a model trained on a large, comprehensive dataset, such as ImageNet (Deng et al., 2009), and adapting it to a specific image classification task with considerably less data exemplifies this approach. The seminal work by Donahue et al. (2014) demonstrated how effectively features extracted from deep networks pre-trained on extensive datasets can be repurposed. By fine-tuning the pre-trained model, the initial layers continue to capture generic features applicable across images, while the latter layers are precisely adjusted to specialize towards the target task. This approach has enabled the rapid adaptation of models to new tasks with limited data availability.

Chapter 3

Approach

In this chapter, we outline the methodology and experimental design used to investigate the optimal strategies for transfer learning. We detail the steps taken during both the pre-training and fine-tuning phases, explain the rationale behind our choices of hyperparameters and architectural configurations, and describe the measures employed to ensure the robustness and reproducibility of our results.

During the pre-training phase, we pre-train the models on either the entire dataset or a subset of classes of the dataset where data is abundant. Subsequently, we modify the trained model by removing the last layer and replacing it with a newly initialized one with a number of outputs that matches the number of classes in the target dataset. Afterwards, the model is fine-tuned after having frozen the parameters in the layers up to the chosen cut point.

The goal of this project is, therefore, to explore the conditions that lead to identifying the optimal cut point in layer-wise transfer learning. We explore which design choices result in the highest performance for the target task. While some factors, such as the amount of target data and the nature of the datasets, are predefined by the task and generally cannot be altered, other factors like freezing layers, reinitializing layers, and selecting the cut point can be adjusted.

To systematically navigate these choices, we define the following research questions to be answered through various experiments:

1. Should the post-cut layers be reinitialized or used as weight initialization for fine-tuning?
2. What is the effect of source and target datasets on the optimal cut?
3. Can the post-cut layers be completely removed for parameter efficiency without significantly impacting performance?
4. Can information from the feature maps of target dataset samples generated by the pre-trained model be used to compare the transferability of different layers?

Furthermore, in order to also explore the effects of varying target data volumes on the optimal cut point, we sub-sample the full target dataset while answering each of these questions. We create multiple copies of the original model, each with a different number of trainable layers, i.e., with different cut points. We then fine-tune these models using increasingly larger datasets to observe how they adapt to varying levels of data availability. For comparison, we also train a baseline model end-to-end on the same target dataset, under the same data scarcity conditions. This allows us to evaluate the performance of each fine-tuned model against the baseline, providing insights into their behavior under different data regimes.

Overall, for each experiment setting, only one model is pre-trained and multiple models are fine-tuned, with the total number being the product of the data percentages, possible cuts (equiv-

Experiment Setup	Source Dataset	Target Dataset
4.2 MNIST	MNIST classes 0,1,2,3,4	MNIST classes 5,6,7,8,9
4.3, 4.3.1, 4.3.3, 4.3.4 FashionMNIST regular classes	FashionMNIST classes 0,1,2,3,4,6,8	FashionMNIST classes 5,7,9
4.3.2 FashionMNIST random classes	FashionMNIST classes 0,1,2,4,6,7,9	FashionMNIST classes 3,5,8
4.4 MNIST to FashionMNIST	MNIST classes 0 to 9	FashionMNIST classes 0 to 9
4.5 FashionMNIST to MNIST	FashionMNIST classes 0 to 9	MNIST classes 0 to 9
4.6.1, 4.6.4 CIFAR Regular Classes	CIFAR classes 2, 3, 4, 5, 6, 7 (Vehicles)	CIFAR classes 0, 1, 8, 9 (Animals)
4.6.2 CIFAR Reverse Classes	CIFAR classes 0, 1, 8, 9 (Animals)	CIFAR classes 2, 3, 4, 5, 6, 7 (Vehicles)
4.6.3 CIFAR Random Classes	CIFAR classes 5, 6, 7, 8, 9	CIFAR classes 2, 3, 0, 1

Table 3.1: SOURCE AND TARGET DATASETS PER EXPERIMENT. *This table contains the datasets and the classes used as the source dataset and the target dataset used in the different experiments, whose results are presented in the Chapter 4.*

alent to the number of convolutional layers), and number of repeats for each fine-tuning experiment, which are introduced in Section 3.3.2.

In Section 3.1, we give an an overview of our architectural and hyperparameter decisions for the pre-training and fine-tuning of the models. Following this, we describe our experimental settings designed to explore the given research questions in Section 3.2. Finally, technical details related to our code base and the measures taken to ensure the reproducibility and objectivity of our experiments are explained in Section 3.3.[1]

3.1 Hyperparameters and Architecture Choices

In this section, we give a short overview about the architectural and hyperparameter decisions for the pre-training and fine-tuning of the models.

3.1.1 Datasets

In our experiments, we use the MNIST (Deng, 2012), Fashion-MNIST (Xiao et al., 2017) and CIFAR10 (Krizhevsky, 2012) image classification datasets. MNIST consists of 70,000 grayscale images of handwritten digits (0-9), divided into 10 classes (Figure 3.1). Fashion-MNIST contains 70,000 grayscale images of fashion items such as T-shirts, trousers, and bags, also divided into 10 classes (Figure 3.2). CIFAR-10, in contrast, includes 60,000 colored images of various objects and animals, with each 32x32 pixel image categorized into one of 10 classes, including vehicles (planes, cars, ships, trucks) and animals (birds, cats, deer, dogs, frogs, horses) (Figure 3.3).

To investigate our research question 2 –How does the direction of transfer across datasets impact the performance of the cut point–, we structure our code to facilitate transferring classes between any of these datasets. The transfer-learning tasks are designed either as intra-dataset,

[1]Our code base can be found at: https://github.com/tusharmanekar/mp-tl-study

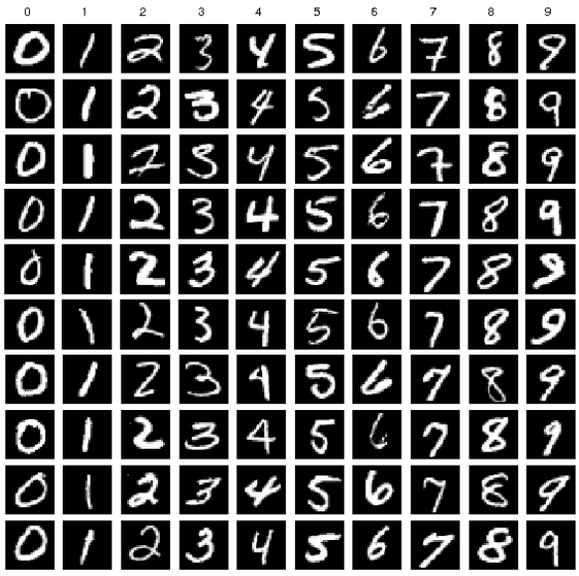


Figure 3.1: MNIST DATASET.

involving learning from a subset of classes and transferring knowledge to another subset within the same dataset, or as inter-dataset, across the entire datasets. A list of the transfer learning experiments presented in this report and the source and target datasets can be found in Table 3.1.

In the experiment setups involving single datasets, such as FashionMNIST regular classes and CIFAR regular classes, source and target datasets are designed to limit the similarity between them. For example, in the FashionMNIST regular classes setup, we pre-train the model on a subset of classes including T-shirt, Trouser, Pullover, Dress, Coat, Shirt, and Bag, which are all related to clothing except the Bag class; following this, we fine-tune the model on a different subset with Sandal, Sneaker, and Ankle boot classes, which are all footwear-related, never seen in the source data. For the CIFAR10 dataset, we pre-train the model on all classes that show different types of vehicles, and fine-tune on all classes portraying different types of animals.

The experiments conducted with FashionMNIST random classes and CIFAR random classes serve as a way to explore scenarios where the classes used in the source and target datasets are more similar compared to regular classes. For example in the FashionMNIST regular classes, the target data (shoeware and Bag) is a new sub-domain of FashionMNIST compared to the source data (clothing items). However, in the FashionMNIST random classes setup, we pre-train the model on randomly selected classes: T-shirt, Trouser, Pullover, Coat, Shirt, Sneaker, and Ankle-boot, and subsequently fine-tune it on the Dress, Sandal, and Bag classes. This means that for two of the target classes there is one or several source classes to which they bear some resemblance, like dresses to tops or sandals to ankle boots; the same cannot not be said for the original break-up.

Similarly, in the CIFAR random classes experiment, we explore another data split that contains both vehicles and animals, which is again meant to increase the similarity between the source and target classes.

Finally, with the MNIST to FashionMNIST and FashionMNIST to MNIST experiments, the idea is to further increase the differences between the source and target datasets, in order to investigate the behaviour of the transfer learning and the optimal cut across datasets that portray different data.

Figure 3.2: FASHIONMNIST DATASET.

3.1.2 Model Architecture

We primarily use a Convolutional Neural Network (CNN) architecture that includes six convolutional layers with 64 kernels per layer, each of size 3x3, interspersed with ReLU activation functions. Average pooling is applied after every second layer, using a kernel size of 2 and a stride of 2. We choose average pooling over max pooling due to its performance, as seen in Section 4.1. A visualization of the architecture can be found in Figure 3.4 We use this architecture in all our experiments in the results section.

For CIFAR, we use the same model architecture, adapted to color images, which means the model has three input channels for color images, unlike the black-and-white images of previous datasets.

We avoid using additional architectures and layers, such as batch normalization, residual connections, varying kernel sizes per layer, and different activation functions, to limit the number of hyperparameters affecting performance.

3.1.3 Training hyperparameters

We employ the Adam optimizer for training our models. Additionally, to prevent overfitting and optimize performance, we use early stopping with a validation split of 0.1 and a patience of 6 epochs.

In all of the experiments, except for those involving CIFAR datasets, we train our models for 40 epochs. For the CIFAR experiments, we train for only 10 epochs, primarily due to computational limitations. Similarly, for all experiments other than those involving CIFAR, we use a batch size of 4096. For the CIFAR experiments, we use a batch size of 64 to better manage the computational resources.

Both during pre-training and fine-tuning phases, we utilize a learning rate of 0.001. This learning rate was selected based on initial experiments that indicated it was effective across different datasets and tasks.

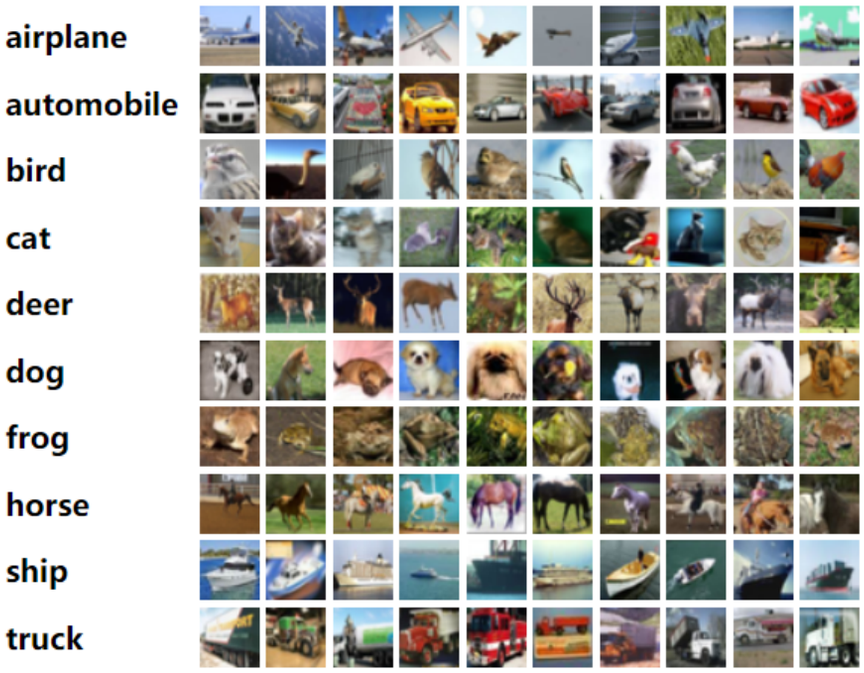


Figure 3.3: CIFAR DATASET.

3.2 Experimental Setup

In this section, we explain the experimental setup we followed, which is specific to our main goal of investigating the factors determining the optimal cut in a transfer learning scenario.

3.2.1 Reinitializing, Freezing, Truncating

We formulate the main idea of the experiment variations in the following way:

The *reinit* and *truncate* hyperparameters only apply to post-cut layers, and the *freeze* hyperparameter only applies to the layers up to the cut point. *reinit=True*, suggests that all post-cut layers are reinitialized, and *reinit=False* suggests that pre-trained parameters in those layers are used as a weight-initialization for fine-tuning. Note that the *reinit* hyperparameter is considered only when *truncate=False*, since when *truncate=True*, we simply remove all the post-cut layers. Also note that the fully-connected classification layer is always reinitialized, regardless of the given hyperparameters.

An overview of different combinations of these settings are presented in Table 3.2. The default setting used in this report is *freeze=True* and *reinit=True*, shortly *FR*, and another important setting is *freeze=True* and *truncate=True*, shortly *FT*. Also it is worth mentioning, the Table 3.2 does not include *truncate=False* setting, since in the first four rows it is implied that *truncate=False*.

For more clarity, consider this example with the main set of hyperparameters we use in this project, where *reinit=True* and *freeze=True* and *truncate=False*, and an example network topology with 3 CNN layers for simplicity:

If the cut is on the n'th layer, and we have the hyperparameters as specified above, that means after pre-training a model on the source dataset we freeze the n layers up to the cut, and reinitialize the layers after the n'th layer (Table 3.2). After that, we fine-tune the model on the target dataset.

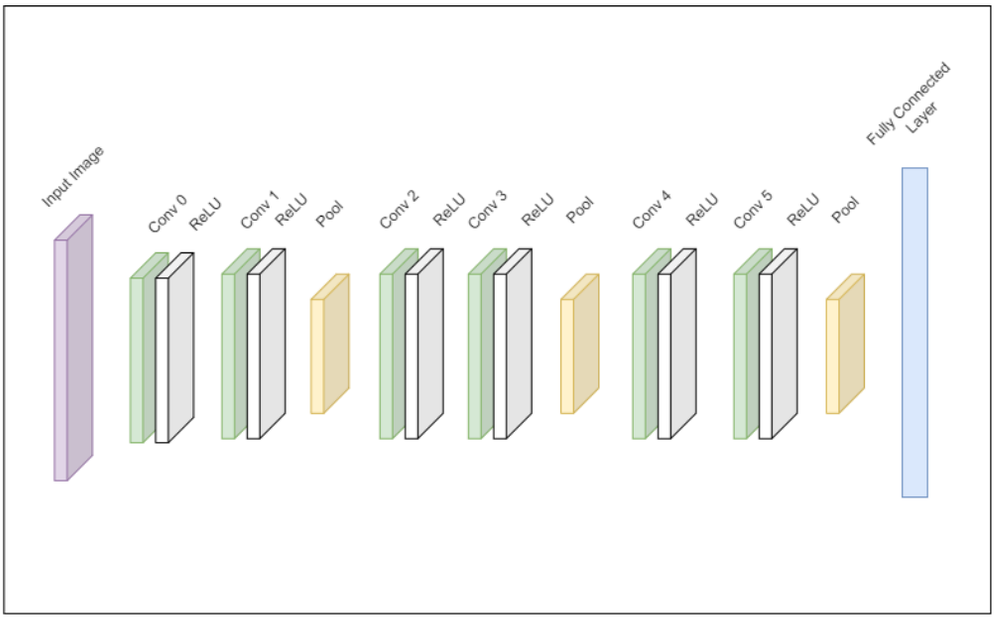


Figure 3.4: MODEL ARCHITECTURE. *The architecture of the CNN model used in the experiments*

The test accuracy after fine-tuning on the target dataset gives us the performance of the cut, which will be later compared to the performances of all other possible cuts for the analysis, as given in the Chapter 4.

For example when cut=2, that implies that the first two CNN layers are frozen (since *freeze=True*) and the last CNN layer is reinitialized (since *reinit=True*), as illustrated in Figure 3.5. The final fully-connected layer is replaced with a new full-connected layer with the correct number of output neurons for the target dataset classes, as in all the experimental settings in this report. Table 3.3 presents the target setting for each layer in the given network topology for each cut point. Note that, if in this example we set *truncate=True*, then we remove both the last CNN layer and the pool layer following it as well, and replace the final fully-connected layer, as visualized in Figure 3.6.

3.2.2 Setup for the Clustering Experiment

We propose a clustering-based method to evaluate the relevance of pre-trained features to a target dataset, to answer research question 4: can information from the feature maps of target dataset samples generated by the pre-trained model help predict the optimal cut point?

An important factor in determining the optimal layer to cut when fine-tuning a pre-trained model is the relevance of the features learned by each layer to the target dataset. To better understand this factor, we propose a clustering-based approach that isolates its effect from other influencing factors, such as the number of trainable parameters. This method allows us to systematically evaluate the usefulness of features learned by each layer during pre-training and their applicability to a specific target dataset.

The theoretical basis of our clustering method can be understood as follows: When a model is pre-trained on a large source dataset, it learns to identify general patterns and features primarily relevant to the source task itself. The features that are learned are hierarchical in increasing order of complexity (Yosinski et al., 2014), and a part of that hierarchy might be common to both the source dataset and the target dataset. In the context of computer vision, these shared features

Experiment Setting	Layers before the cut	Layers after the cut	Final dense layer(s)	Short name
freeze=True and *reinit=True*	not reinitialized, not trainable	reinitialized, trainable	replaced and reinitialized	*FR*
freeze=True and *reinit=False*	not reinitialized, not trainable	not reinitialized, trainable	replaced and reinitialized	*F*
freeze=False and *reinit=True*	not reinitialized, trainable	reinitialized, trainable	replaced and reinitialized	*R*
freeze=False and *reinit=False*	not reinitialized, trainable	not reinitialized, trainable	replaced and reinitialized	–
freeze=True and *truncate=True*	not reinitialized, not trainable	removed	replaced and reinitialized	*FT*

Table 3.2: EXPERIMENT SETTINGS OVERVIEW. *This table presents the behaviour of the layers before the cut, after the cut, and the final dense layer during fine-tuning. F stands for freeze, R stands for reinit, and T stands for truncate flags. The freeze=False and reinit=False does not have a short name since it does not exist as a separate experiment in this report.*

might include fundamental visual elements, such as shapes, textures, or structural patterns, that are relevant to both datasets.

When we fine-tune a pre-trained model on a target dataset, our goal is to adapt the model's learned features to better fit the unique characteristics of this new dataset. However, not all features learned during pre-training are equally relevant to the target dataset. Some features may be overly specialized to the source dataset and offer little value for the target dataset.

The idea, then, is to pinpoint the layer where the pre-trained features start to lose their relevance for classifying the fine-tuning samples. This layer would be an ideal cut point, as it separates the features that remain useful from those that are no longer relevant. By using embedding clustering, the hope is to systematically determine the optimal cut point, eliminating the need for trial and error with multiple models.

We begin by analyzing the shared features between the source and target datasets by clustering fine-tuning samples using features from the entire pre-trained model. In our experiments, we use the Agglomerative clustering algorithm with cosine similarity as the similarity measure (Müllner, 2011)[2]. Next, we assess the quality of the clustering using the Adjusted Rank Index (ARI) metric (Hubert and Arabie, 1985), which measures how well the resulting clustering overlaps with the original classes in the target dataset. We repeat this process for each individual layer, examining the relevance of features learned at each level. This approach allows us to assess how well each layer of the pre-trained model captures the shared features that are relevant to both datasets. If a layer's features are effective in clustering the target dataset samples, it would suggest that those features are indeed shared and useful for both datasets. Conversely, if certain layers' features result in a clustering that doesn't match the fine-tuning classes, it implies that those features might be more specific to the source dataset and less relevant for the target dataset.

[2]implementation from scikit-learn

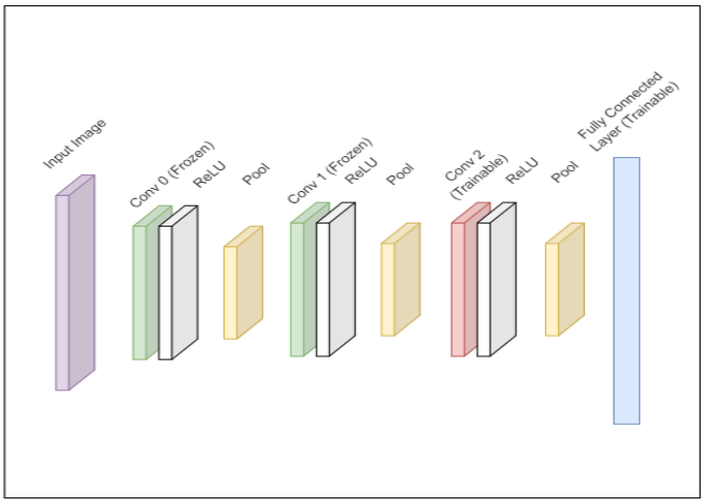


Figure 3.5: EXAMPLE ARCHITECTURE. *Example architecture with 3 CNN layers and no pooling, for the demonstration of the default experimental setting, FR.*

The steps and tools we follow for the clustering experiments can be summarized as following:

1. Start with the pre-trained model
2. Feed in a subset of the target train dataset into the pre-trained model and extract the hidden representations after each convolutional layer (note that we use global average pooling to aggregate outputs of different kernels of each layer).
3. Train an Agglomerative Clustering algorithm on the extracted features from step 2, setting the number of clusters equal to the number of classes in the target dataset, as the goal is to classify them. Set the affinity to cosine similarity and the linkage to average.
4. Evaluate the performance of quality of the clusters on the target test set using the ARI metric. Calculate the average ARI over all the samples per each layer.
5. Repeat Step 1-3 for different percentages of data, each representing a subset of the dataset. Furthermore, we conduct each experiment multiple times to ensure consistency with the fine-tuning experiments and to enable statistical tests. For specifics on how we sub-sample the data and the number of repeats used, refer to Section 3.3.2.
6. Report the results in box-plots, following the same plotting conventions we follow for reporting the fine-tuning experiments (Section 3.3.4).

3.2.3 Experiments Overview

An overview of all the experiments are given in Table 3.4. The terms *regular classes*, *random classes* and *reverse classes* in the experiment names indicate both the dataset direction used (Table 3.1) and the pre-trained model employed (Table 4.1). The *FR*, *F*, and *FT* settings are explained in Table 3.2, while *cluster* refers to the clustering experiments and is not included in that table.

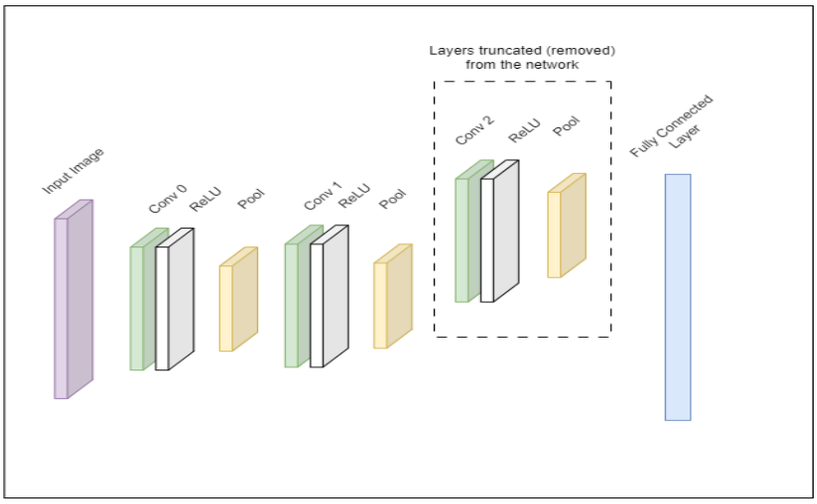


Figure 3.6: TRUNCATING A CNN. *Illustration of truncating a CNN, i.e., removing a Convolutional Layer from the model architecture.*

For each of the three datasets, we conduct a primary experiment using the regular classes in the *FR* setting. The primary experiments address the main goal of the project.

To explore the research question 1 – should the post-cut layers be reinitialized – we compare the FashionMNIST regular classes *FR* to the same experiment with *reinit=False* (FashionMNIST regular classes *F*).

The research question 2 can be formulated as multiple sub-questions:

(a) How do similarities between the source and target datasets influence the optimal cut point? For example, what differences arise when comparing experiments that use the standard source-target dataset split with those with randomized splits?.

(b) What is the effect of flipping the source and target classes or using a different data split? We are interested in observing if there is a change in the pattern when we experiment with a different data split. For this, we compare the MNIST to FashionMNIST regular classes *FR* and FashionMNIST to MNIST regular classes *FR* with each other. Additionally, we conduct the CIFAR reverse classes *FR* experiment, which we compare to the CIFAR regular classes *FR* experiment to observe the impact of flipping the source and target classes.

To answer question 3 – can the post-cut layers be completely removed for parameter efficiency without significantly impacting performance? – we compare the truncation experiments to the *FR* experiments where we reinitialize those layers instead of truncating.

Finally, to address research question 4 – can information from the feature maps of target dataset samples generated by the pre-trained model be used to compare the transferability of different layers? – we perform the FashionMNIST regular classes cluster experiment. In this experiment, we use the hidden feature maps of the pre-trained model to cluster the target dataset samples, isolating factors influencing the performance of transfer learning, such as the number of trainable parameters, and assessing the transferability of the learned features at each layer.

3.3 Reproducibility and Objectivity

Reproducibility is vital in deep learning research for ensuring scientific integrity and progress. It involves consistently generating the same results, including plots and findings, across experiment

Cuts	CNN layer 1	CNN layer 2	CNN layer 3	Final dense layer
cut=-1 (baseline)	reinitialized, trainable	reinitialized, trainable	reinitialized, trainable	reinitialized, trainable
cut=0	reinitialized, trainable	reinitialized, trainable	reinitialized, trainable	reinitialized, trainable
cut=1	frozen	reinitialized, trainable	reinitialized, trainable	reinitialized, trainable
cut=2	frozen	frozen	reinitialized, trainable	reinitialized, trainable
cut=3	frozen	frozen	frozen	reinitialized, trainable

Table 3.3: OVERVIEW OF LAYERS IN DIFFERENT CUTS. *This table presents the status of each layers in a network with 3 CNN layers and one fully-connected layer during fine-tuning. Note that cut=-1 and cut=0 are exactly the same in this scenario, since in both all the layers are reinitialized because of the reinit=True setting. Number of trainable parameters in fine-tuning for each cut is given in the last column. With the progression of the cuts into later cuts, the number of trainable parameters in fine-tuning drop, limiting the capacity of the fine-tuning. Note that, cut=2, cut 2 and the second cut all refer to the same concept.*

runs. In this section, we will be discussing the measures we took to ensure the reproducibility and reliability of our results, considering the computational limitations. In the context of our experiments, achieving reproducibility requires several key steps: initializing random number generators uniformly with *seeds*, maintaining consistency in data sub-sampling, reinitializing parts of the model with identical seeds, and storing pre-trained models. These practices are necessary for upholding the reliability and consistency of experimental outcomes.

3.3.1 Reproducibility Measures

In our efforts to ensure reproducibility, we establish a procedure for repeating experiments consistently. This involves defining a fixed number of repeats and setting the seed for each repeat to the index representing that repeat. This method ensures that each repeat produces different results, which is essential for subsequent statistical analysis. However, it also guarantees that if we were to rerun all repeats, each repeat would yield the exact same results as in the previous run.

Additionally, we apply these seeds uniformly across all aspects of our experiments. Whether initializing a model or partially reinitializing it, we ensure consistent seeds are used (which were set to the repeat index again). Moreover, when selecting different subsets of the target dataset for experiments, we maintain the same seeds across different runs, with each repeat assigned a unique seed. This comprehensive approach addresses both sampling and weight reproducibility, provided that the code is executed on an the same machine.

3.3.2 Data sub-sampling for fine-tuning

In this project, the quantity of target data serves as an important factor affecting transfer learning performance and the position of the optimal cut. To investigate its effects, we conduct each experiment with varying percentages of available target data. For each data sub-sampling percentage,

Experiment	Objective
4.2 MNIST	Main goal
4.3 FashionMNIST regular classes *FR*	Main goal
4.3.1 FashionMNISTregular classes *F*	Research question 1
4.3.2 FashionMNISTrandom classes *FR*	Research question 2
4.3.3 FashionMNISTregular classes *FT*	Research question 3
4.3.4 FashionMNISTregular classes cluster	Research question 4
4.4 Mnist to Fashion	Research question 2
4.5 FashionMNISTto FashionMNIST	Research question 2
4.6.1 CIFAR regular classes *FR*	Main goal
4.6.2 CIFAR reverse classes *FR*	Research question 2
4.6.3 CIFAR random classes *FR*	Research question 2
4.6.4 CIFAR regular classes *FT*	Research question 3

Table 3.4: OVERVIEW OF EXPERIMENTS AND RESEARCH QUESTIONS. *This table presents the research questions addressed by each experiment.*

we sub-sample a balanced number of samples per class. The number of repeats used for each data percentage and the exact number of training samples can be found in Table 3.6.

3.3.3 Statistical Tests

To evaluate differences between multiple cuts, we employ the Wilcoxon signed-rank test (Wilcoxon, 1945). This statistical test determines whether two sets of values are significantly different from each other. The Wilcoxon test operates by comparing the ranks of observations in two samples, assessing whether the distributions differ significantly. It is robust against outliers and does not assume normality of the data.

We utilize the Wilcoxon test in two distinct ways. Firstly, for each data sampling percentage, we compare every cut to all other cuts (including the baseline model, which is referred to as *cut=-1*) based on their series of test accuracies resulting from multiple repeats (Section 3.3.2). Secondly, we compare identical cuts across different experiments to ascertain if they exhibited significant differences in performance. We color-code the results of these two tests for clarity (explained in more detail in Section 3.3.4), and comprehensive tables containing the p-values of all comparisons are provided in the Appendix. It is worth noting that we use fewer repeats for the 50% and 100% data percentages due to computational limitations, which leads to the statistical test results being insignificant for most of those cuts.

3.3.4 Reporting the experiment results with box plots

To present our results, we utilize box plots, with each sub-plot representing a distinct sub-sampling percentage of target data. Within each box plot, we include one box for the baseline model and one box per the cut (more details on the cuts can be found in Table 3.3). The numbers within the boxes represent the median test accuracy of each model on the target set.

Number of trainable parameters	Cut 0	Cut 1	Cut 2	Cut 3	Cut 4	Cut 5	Cut 6
4.2 MNIST	188165	187525	150597	113669	76741	39813	2885
4.3, 4.3.1, 4.3.2 FashionMNIST	187011	186371	149443	112515	75587	38659	1731
4.3.3 FashionMNIST truncation	2355	150531	37675	37675	9411	9411	1731
4.4, 4.5 Inter-dataset	191050	190410	153482	116554	79626	42698	5770
4.6.1, 4.6.3 CIFAR	192582	190790	153862	116934	80006	43078	6150
4.6.2 CIFAR reverse	190532	188740	151812	114884	77956	41028	4100
4.6.4 CIFAR truncation	18438	393222	98310	98310	24582	24,582	6,150

Table 3.5: OVERVIEW OF NUMBER OF TRAINABLE PARAMETERS IN DIFFERENT CUTS. *This table presents the number of trainable parameters at each cut for all the model architectures. Note that, the number of target dataset classes change these numbers, resulting in some data directions to be reported separately. For example, the random classes data splits contain the same number of target dataset classes as the regular classes data splits (see Table 3.1), so they are reported in a single row. However, CIFAR reverse classes direction has a different number of target dataset classes from CIFAR regular classes, so it is reported in a separate row. Also note that, Inter-dataset experiments stand for MNIST to FashionMNIST and FashionMNIST to MNIST regular classes directions.*

We color-code the box plots according to the statistical test results. Each color corresponds to the statistical rank of the cut. The cuts or boxes are grouped into ranks based on pairwise statistical tests, and the ranks are ordered according to the mean test accuracies of the cuts on the target dataset. For example, in Figure 3.7, we plot two experiments next to each other. However, the color coding only ranks the cuts in each percentage and experiment. Therefore, the ranking of the cuts in two different percentages or in two different experiment settings is independent of each other. The cuts encoded with the lightest color belong to rank 1, as they show statistically higher test accuracies compared to the cuts in all other ranks.

In addition to the color coding for the ranks, we apply hatches to the box plot on the right to compare two box plots (experiments) with different experimental settings. The boxes with hatches shaped as stars indicate that the marked cut is significantly higher in the corresponding experiment compared to the same cut in the other experiment. If the box plot has diagonal hatches, that shows the marked cut is significantly lower in the corresponding experiment compared to the same cut in the other experiment. No hatching on the boxes means the difference between the experiments for that percentage and cut point is not statistically significant.

For instance, in Figure 3.7, we compare a complete experiment in *reinit=True* with another complete experiment in *reinit=False* settings. It can be seen that, looking at the sub-sampling percentage 1%, the cuts 0,1 and 2 in the *reinit=False* experiment have significantly higher target dataset test accuracies compared to their counterparts in the *reinit=True* setting. However, it is the opposite when we look at the cuts 4,5 and 6 in the 1%.

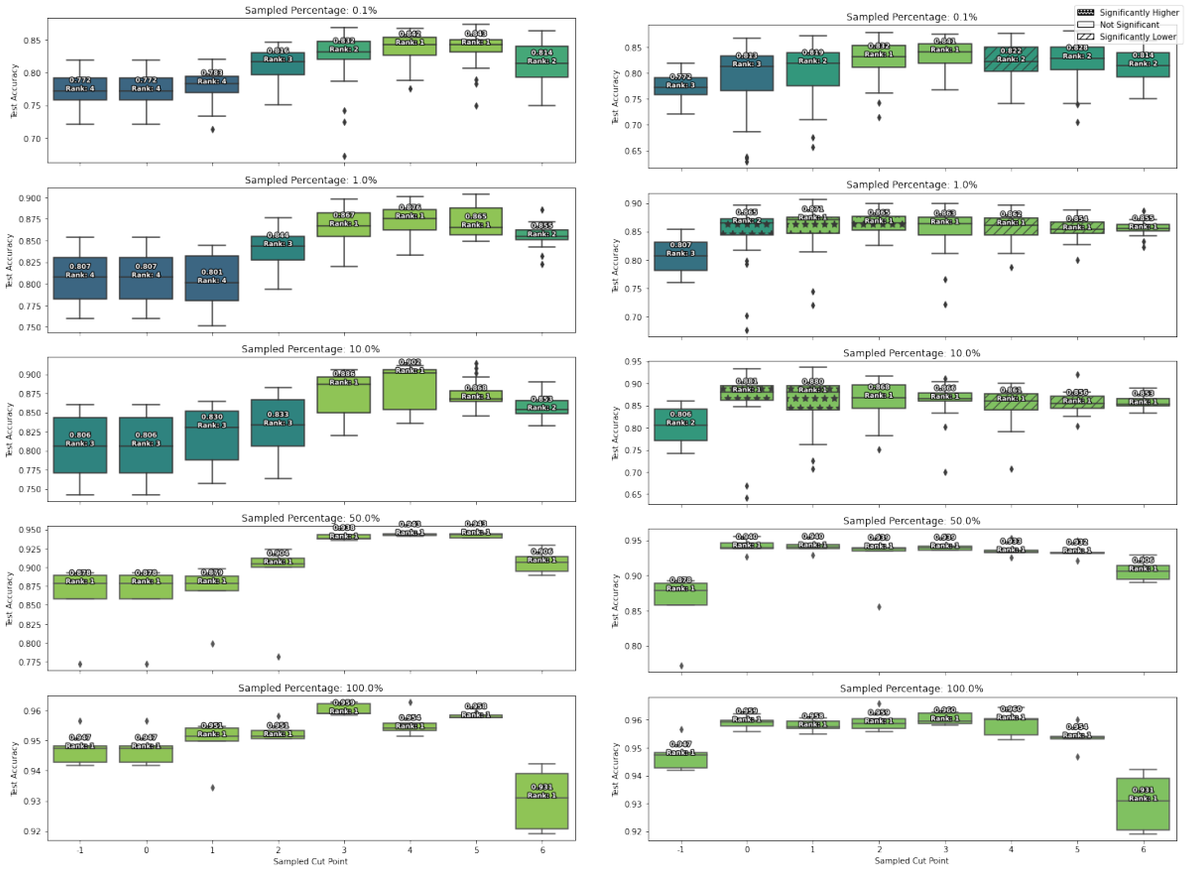
We apply a similar ranking and color-coding to the clustering experiments depending on the same statistical test. In the x-axis experiments, Conv0 indicates the clustering performance with the features extracted from the first convolutional layer, measured in ARI. This is equivalent to freezing the first convolutional layer (Conv0), and extracting the pre-trained features from it. Therefore, the Conv0 in the x-axis corresponds to the cut=1 in the other experiments. Similarly, Conv1, Conv2, and so on, represent the clustering performance using features extracted from subsequent convolutional layers. The y-axis (ARI) represents the ARI metric (see Section 3.2.2).

Source Dataset	Target Dataset (Percentages)					Source Dataset
	0.1%	**1%**	**10%**	**50%**	**100%**	
Repeats	25	25	20	5	5	1
MNIST	25	260	2645	13230	26038	27537
FashionMNIST (regular and random classes)	15	162	1620	8100	16196	37800
MNIST to FashionMNIST and FashionMNIST to MNIST	50	540	5400	27000	54000	54000
CIFAR (regular and random classes)	16	180	1800	9000	17956	27000
CIFAR (reverse classes)	24	270	2700	13500	26946	18000

Table 3.6: OVERVIEW OF NUMBER OF SAMPLES AND REPEATS PER EXPERIMENT. *This table presents the number of repeats for the pre-training and fine-tuning stages of each experiment setup, along with the number of training samples in source and target datasets, including different sub-sampling percentages of the target datasets.*

3.3.5 Code Base

We structured our Github repository in the following way to streamline our workflow: In our folder structure, we have one folder per dataset, along with two additional folders for cross-dataset experiments (from MNIST to FashionMNIST and from FashionMNIST to MNIST). Within each dataset folder, we have subfolders for storing the pre-trained models, the results of all our experiments, and the corresponding hyperparameters used for each specific experiment setting. This approach ensures transparency and documentation of our experimental configurations. For more details please refer to our README file.



(a) *reinit=True*
reported p-values: Table A.2

(b) *reinit=False*
reported p-values: Table A.3

Figure 3.7: FASHIONMNIST REGULAR CLASSES, COMPARISON BETWEEN *FR* VS. *F* SETTINGS. *The p-values that led to the cross-comparison between the two experiments (visualized as the hatches on the Figure 4.2(b)) can be found in Table A.4.*

Chapter 4

Experimental Results

Pre-trained model	Train accuracy	Validation accuracy	Test accuracy	Epochs to converge
4.2 MNIST regular classes	0.9957	0.9912	0.9952	40
4.3, 4.3.1, 4.3.3, 4.3.4 FashionMNIST regular classes	0.8807	0.8760	0.8802	40
4.3.2 FashionMNIST random classes	0.8847	0.8798	0.8842	40
4.4 MNIST to FashionMNIST regular classes	0.9910	0.9870	0.9906	40
4.5 FashionMNIST to MNIST regular classes	0.8994	0.8917	0.8986	40
4.6.1, 4.6.4 CIFAR regular classes	0.9513	0.8880	0.9449	10
4.6.2 CIFAR reverse classes	0.8173	0.7267	0.8083	10
4.6.3 CIFAR random classes	0.9248	0.8792	0.8816	10

Table 4.1: OVERVIEW OF PRE-TRAINED MODELS. *This table includes the accuracies of the pre-trained models on the source datasets, along with the number of epochs to converge. All the models follow the same architecture, as given in Section 3.1.2. The hyperparameters used for pre-training, source data classes and number of samples used in pre-training can be found in Section 3.1.3, Table 3.1 and Table 3.6 respectively.*

This section covers the experiments conducted to investigate the behavior of fine-tuning under various conditions. To observe the impact on model performance, each experiment manipulates

Experiment	*Best Absolute Improvement*	*Best Error Rate Reduction*
4.2 MNIST regular classes *FR*	15.3 at 10% in cut 4	79.68 at 10% in cut 4
4.3 FashionMNIST regular classes *FR*	9.6 at 10% in cut 4	53.28 at 50% in cut 4
4.3.1 FashionMNIST regular classes *F*	7.5 at 10% in cut 0	50.81 at 50% in cut 0
4.3.2 FashionMNIST random classes *FR*	6.4 at 0.1% in cut 3	51.61 at 10% in cut 3
4.3.3 FashionMNIST regular classes *FT*	7.3 at 10% in cut 2	54.92 at 50% in cut 2
4.4 MNIST to FashionMNIST	5.2 at 1% in cut 5	16.96 at 10% in cut 5
4.5 FashionMNIST to MNIST	9.7 at 0.1% in cut 5	26.07 at 0.1% in cut 5
4.6.1 CIFAR regular classes *FR*	14.7 at 1% in cut 4	20.27 at 1% in cut 4
4.6.2 CIFAR reverse classes *FR*	17.5 at 1% in cut 5	36.97 at 10% in cut 4
4.6.3 CIFAR random classes *FR*	22.3 at 0.1% in cut 5	42.69 at 1% in cut 5
4.6.4 CIFAR regular classes *FT*	15.7 at 0.1% in cut 4	21.65 at 0.1% in cut 4

Table 4.2: OVERVIEW OF BEST IMPROVEMENTS PER EXPERIMENT. *This table provides an overview of the improvement of the cut with the highest performance improvement over the baseline per experiment. Improvement is measured in terms of the best absolute improvement points and the best error rate reduction, both calculated on the target test set.*

two key parameters: the proportion of data allocated for fine-tuning and the number of parameters trained.

The experiments include multiple transfer learning directions, as presented in Table 3.1, and various experimental settings, as detailed in Table 3.2. Each experiment involves multiple runs (Section 3.3.3) with different target data percentages (Section 3.3.2), and multiple cuts, which are further explained in Table 3.2.

For the purposes of this section, we primarily focus on scenarios where layers are frozen (*freeze=True*) and subsequent layer parameters are reinitialized (*reinit=True*), or *FR* as given in Table 3.1.

4.1 Architecture Search

We start by presenting our hyperparameter search, which leads us to the architecture we use in all of the experiments in this report. Our hyperparameter search focuses on architectural adjustments such as the number of channels and their consistency across layers, while maintaining a uniform kernel size of 3x3. All experiments involve training the models end-to-end on all classes of the CIFAR dataset in order to accommodate for the most complex dataset out of the three, while still maintaining the same architecture across all of them to facilitate comparisons.

Initial attempts focus on architectures with variable kernel counts per layer. Increasing by factors of two, starting from 32 channels across six layers, the resulting models had a sequence of 3 (for color images as input), 32, 64 and finally 128 channels. The number of layers are fixed at 6 to enhance the granularity of cuts in the upcoming transfer learning experiments.

During the exploration of various pooling strategies applied every second layer —including no pooling, strided convolutions (with a stride of 2 on even layers), max pooling, and average pooling— results indicate a performance ranking from least to most effective in the aforementioned sequence.

L1	L2	L3	L4	L5	L6	Reduction	Stride	Accuracy
128	128	128	128	128	128	avg	2	79.16%
128	128	128	128	128	128	avg	1	79.16%
256	256	256	256	256	256	avg	2	79.14%
64	64	64	64	64	64	avg	1	78.38%
32	32	64	64	128	128	avg	1	79.14%
32	32	64	64	128	128	none	2	72.25%
32	32	32	32	32	32	avg	1	73.71%
32	32	64	64	128	128	max	1	72.08%
16	16	16	16	16	16	avg	1	64.27%
32	32	64	64	128	128	none	1	65.13%
64	64	64	64	64	64	none	1	63.29%

Table 4.3: ARCHITECTURE SEARCH. *This table presents various convolutional neural network (CNN) models, with each row representing one model. The Accuracy column indicates the test accuracy of each model. The columns labeled L1 through L6 denote the number of channels for the respective layers. The Reduction column indicates the type of Pooling applied after every two convolutional layers. Average pooling (AvgPool2D(2, stride=2)) and Max pooling (MaxPool(2, 2)) operations were applied after every two convolutional layers. The Stride column refers to the convolution stride, and it is applied to layers 2, 4, and 6.*

The hyperparameter optimization also include an investigation of models with uniform channel counts across layers. A total of five versions are explored, from 16 to 256 kernels per layer in 2x increments. We select the configuration with 64 channels per layer, among all other alternatives, due to its competitive accuracy of 78.38%. Higher channel counts have very limited performance increases beyond this point. For the accuracy values for all other models refer to the Table 4.3.

The adoption of average pooling is based on its demonstrated efficacy: omitting it while retaining in a 64-channel architecture has resulted in a significant accuracy drop to 63.29%, which equates to a 15.09 percentage point decline.

4.2 MNIST

In the MNIST regular classes *FR* experiment, the focus is on the dataset deemed to be the simplest for neural networks to classify among the three chosen datasets. The experiment uses the MNIST regular classes source-target split as given in Table 3.1, and follows the default experiment setting *FR* as detailed in Table 3.2 where we freeze up to the cut and reinitialize afterward. Furthermore, we use the MNIST regular classes pre-trained model with the pre-training accuracies given in Table 4.1.

As seen in Figure 4.1 an interesting pattern is revealed: a particular cut in the network's layers emerges as superior to others as a function of the size of the sub-sample of target data. Specifically, the fourth cut tends to match or outperform other layers across various low-resource sub-sample sizes, namely 0.1%. 1% and 10%. This suggests that the level of abstraction it represents is most conducive to knowledge transfer in low-resource scenarios (see Table 3.3 for more details into what different cuts correspond to in the network).

In middle and higher resource scenarios of 50% and 100% target data sub-sampling, cut 4 no longer stands out as the best performer. At the 50% level, cuts 1, 2, 3, and 4 perform similarly, with all four cuts only slightly exceeding baseline median values. For 100%, cuts 1, 2, 3, and 4 are

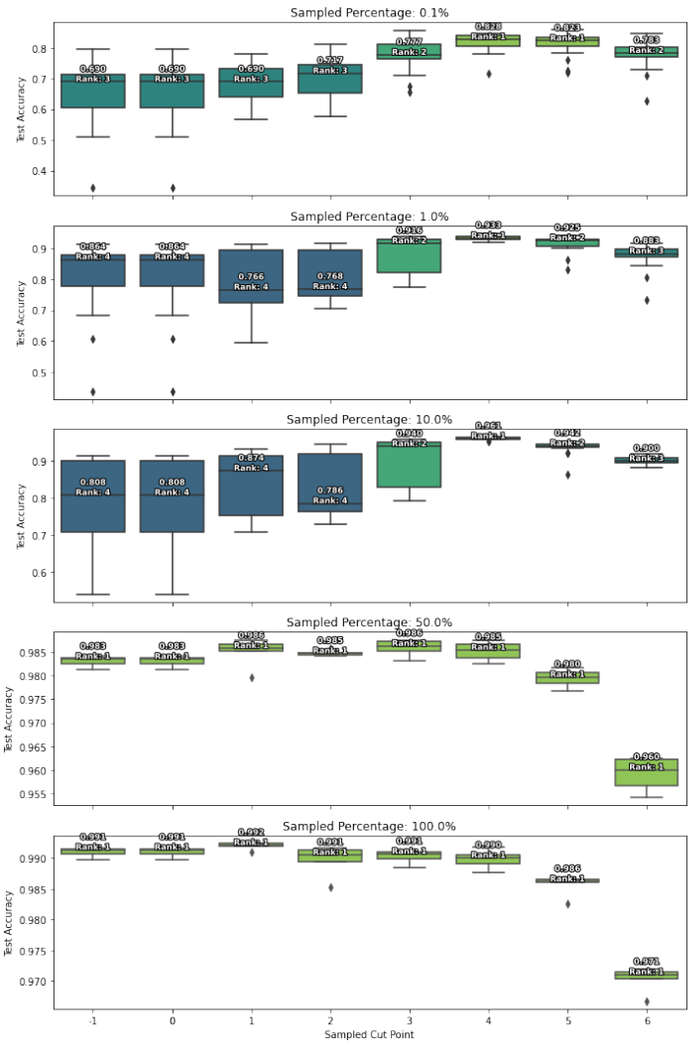


Figure 4.1: MNIST REGULAR CLASSES IN *FR* SETTING. *The p-values that led to the ranking of the cuts can be found in Table A.1.*

almost identical and align closely with the baseline. Cuts 5 and 6, however, perform significantly below the baseline in both 50% and 100% sub-sampling scenarios, indicating that their specialized information becomes a hindrance compared to the other cuts. However, to put this into context, cuts 5 and 6 at 50% and 100% sub-sampling still manage to match or outperform the median accuracy of cut 4 in low-resource scenarios.

Interestingly cut 6 is essentially a perceptron over a feature extractor. Its competitive performance highlights the applicability of the entire model, not just portions of it, to the target data. It, in fact, retains substantial performance across various sub-sample sizes.

The most significant performance improvement observed is an increase of 15.3 absolute percentage points in accuracy. This improvement is observed in the 10% sub-sample for the cut 4, and it equates to a 79.68% error rate reduction, which also happens to be the highest error rate reduction result in the experiment.

4.3 FashionMNIST

We transition to a relatively more complex dataset, FashionMNIST. The following sections present results from experimental settings, including experiments where some convolutional layers are and are not reinitialized during fine-tuning (*reinit=True* and *reinit=False* respectively), a truncation experiment as detailed in Section 3.2.1, a clustering experiment as explained in Section 3.2.2, and an experiment where random classes are chosen for pre-training and fine-tuning.

Note that in Section 4.3.1, Section 4.3.3, and Section 4.3.4, we use the same data split (referred to as *FashionMNIST regular classes* in Table 3.1) and the same pre-trained model (*FashionMNIST regular classes* in Table 4.1). However, in the FashionMINST random classes experiment from Section 4.3.2, we use the *FashionMINST random classes* data split in Table 3.1 and *FashionMINST random classes* pre-trained model in Table 4.1.

These experiments aim to validate certain design choices, such as the the selection of pre-training and fine-tuning classes, as well as to answer some of our research questions.

4.3.1 Regular classes *FR* vs. *F*

This section presents the findings of the FashionMNIST regular classes *FR* and FashionMNIST regular classes *F* experiment.

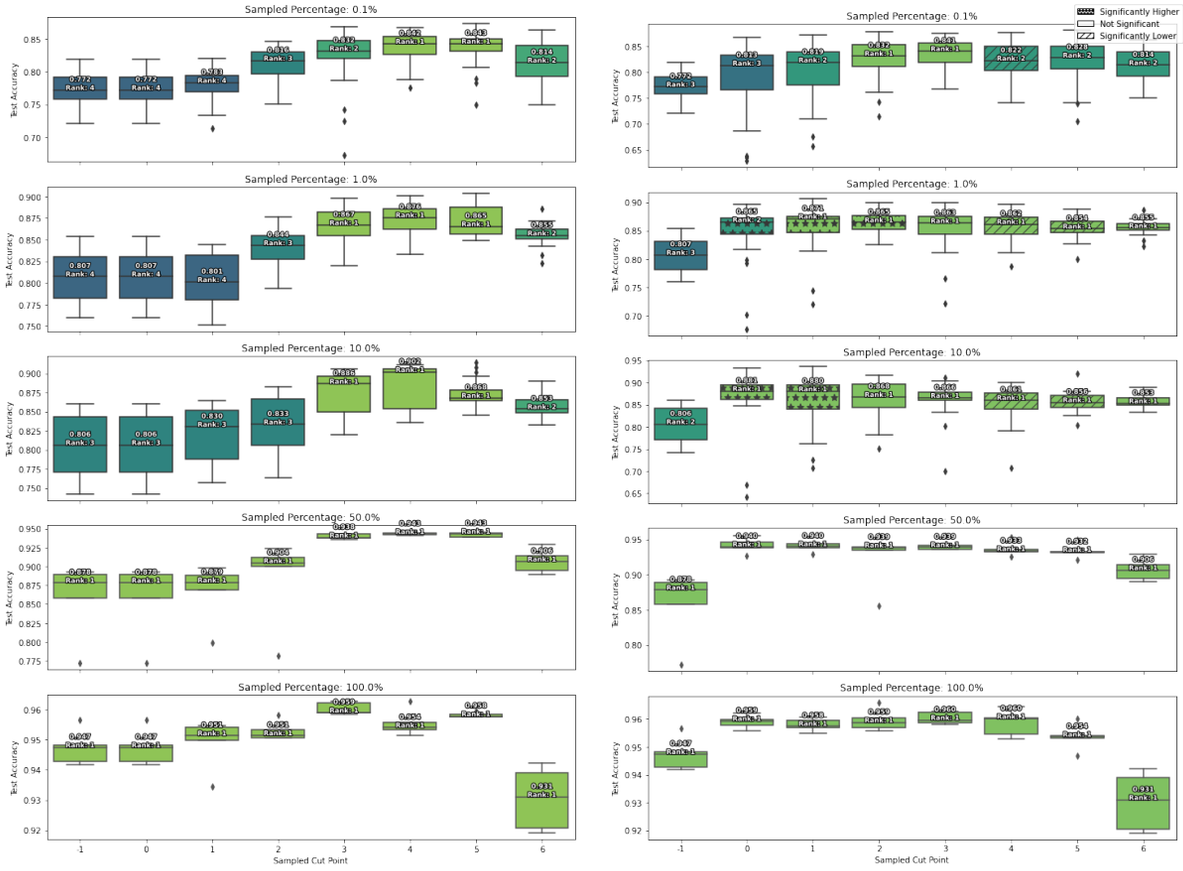
The results for *reinit=True*, as seen in Figure 4.2(a), indicate that cuts 3, 4, and 5 perform best compared to their baselines in target data sub-sampling up to 10%. These results in the low-resource regime are quite similar to the low-resource results observed in the MNIST regular classes *FR* experiment (Section 4.2). However, the behavior diverges in middle and high resource sub-samples. For MNIST, cut 4, which was the best in low resource scenarios, shows performance equivalent to the baseline in medium and high resource scenarios. In contrast, in the current experiment, cuts 3, 4, and 5 remain above baseline at 50% and even 100% sub-sampling. Additionally, for MNIST, the last cut, cut 6, falls below the baseline at 50%, whereas in the current experiment, it does not.

We can now compare the results of the *reinit=True* setting with an alternative *reinit=False* setting, where the convolutional layers after the cut point are not reinitialized during fine-tuning. Consequently, this experiment is designed to answer Research question 1.

The first thing that stands out when examining Figure 4.2(b) is the reduced variation in median accuracy. Unlike for MNIST regular classes *FR* shown in Figure 4.1 and for FashionMNIST regular classes *FR* in Figure 4.2(a), where some middle cuts often performed best, visually forming an arched pattern, this is not observed here. Instead, especially from 10% sub-sampling onwards, we see a downward pattern where earlier cuts, often cut 0, perform best, with performance very gradually decreasing from one cut to the next. There are, however, exceptions to this pattern in Figure 4.2(b): at 0.1%, where cut 3 performs best, the arch pattern appears slightly, and at 100%, cuts 0 through 4 are roughly equivalent.

The preference for earlier cuts as the optimal points indicates a tendency to keep most or all layers trainable, using their parameters primarily for weight initialization. For example, cut 0, which makes the entire model trainable as a weight initialization, performs best at 10% and 50%. This contrasts with previous *FR* experiments, where the trend is instead to freeze half or more of the layers, and cut 0 would effectively reinitialize the entire model instead.

Using the Wilcoxon statistical test further confirms our observations. It reveals that earlier cuts have an advantage when the layers after the cut are not reinitialized, while middle to late cuts appear to benefit in the opposite scenario. It is interesting how this middle to late cuts seem to be hindered by the knowledge of the following layers: perhaps going beyond the 40 epochs that we imposed (see Table 4.1) could allow for their *reinit=False* counterparts to catch up.



(a) *reinit=True*
reported p-values: Table A.2

(b) *reinit=False*
reported p-values: Table A.3

Figure 4.2: FASHIONMNIST REGULAR CLASSES, COMPARISON BETWEEN *FR* VS. *F* SETTINGS. *The p-values that led to the cross-comparison between the two experiments (visualized as the hatches on the Figure 4.2(b)) can be found in Table A.4.*

It is also worth noting that, across the *reinit=True* and *reinit=False* settings, the results for cut 6 are always the same for the same data sub-sampling. This is because, at cut 6, there are no more CNN layers after the cut and therefore the reinitialization strategy makes no difference. Since both experiments were executed independently of each other on the same hardware, the identical results also act as a testament to the success of our reproducibility efforts (see Section 3.3.1).

With that being said, neither strategy is definitively superior, as both approaches tend to achieve very similar accuracy values when considering the best-performing cut for each data sub-sampling level. There is one exception, though, where *reinit=True* results in higher performance. In Figure 4.2(a), cut 4 at 10% achieves an accuracy of 90%, whereas the highest accuracy for 10% in Figure 4.2(b) is 88.1% at cut 0.

When reinitializing, the most substantial absolute performance improvement observed is an increase of 9.6 percentage points in accuracy. This improvement occurred in the 10% sub-sample for cut 4, equating to a 49.48% error rate reduction. The largest error rate reduction observed was 53.28 percentage points, seen in the 50% sub-sample for cut 4, corresponding to a 6.5% absolute performance improvement.

On the other hand, without reinitializing, the most notable absolute performance improvement is a 7.5 percentage point increase in accuracy. This occurred in the 10% sub-sample for cut 0, representing a 38.66% error rate reduction. The greatest error rate reduction observed without reinitializing was 50.81 percentage points, in the 50% sub-sample for cut 0, corresponding to a 6.2% absolute performance improvement.

4.3.2 Random Classes *FR*

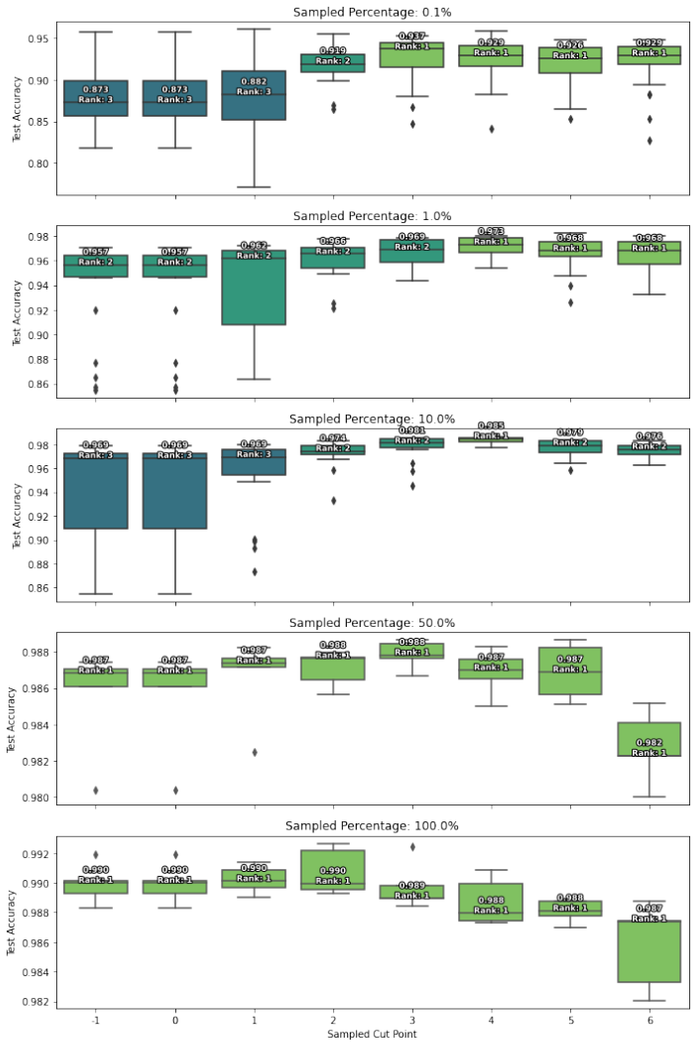


Figure 4.3: FASHIONMNIST RANDOM CLASSES IN *FR* SETTING. *The p-values that led to the ranking of the cuts can be found in Table A.8.*

This section presents the findings of the FashionMNIST random classes *FR* experiment, which is a repeat of the previous FashionMNIST regular classes *FR* experiment using a random split of source and target dataset classes (available in Table 3.1).

The randomized datasets was designed to help answer Research question 2. It provides a new

transfer direction in the space of potential source and target datasets which, in this particular case, also increases the similarity across source and target (see Section 3.1.1).

The results of the experiment can be observed in Figure 4.3. At 0.1%, 1%, and 10%, cuts 3, 4, and 4 are optimal respectively, indicating that middle cuts are favored in low-resource scenarios. At 50%, the highest median values correspond to cuts 2 and 3, but broadly speaking, all cuts except cut 6 are largely equivalent to the baseline. Finally, at 100%, cuts 1, 2, and 3 exhibit accuracies equivalent to the baseline, while the remaining cuts slightly underperform, pointing to very deep cuts being somewhat detrimental in high resource scenarios.

The non-randomized variant of this experiment, FashionMNIST regular classes *FR* (see Section 4.3.1), establishes a tendency to favor middle cuts in low-resource scenarios and showed an arched patterns. While we observe a similar preference for middle cuts in low-resource sub-samples, the arched pattern is not as evident. Instead, we see most cuts exhibiting remarkably similar behavior: cuts 3 through 6 up to 10% sub-sampling, cuts 1 through 5 for 50% sub-sampling, and cuts 1 through 3 for 100% sub-sampling are largely equivalent.

The biggest performance improvement observed is an increase of 6.4 absolute percentage points in accuracy. This improvement was observed in the 0.1% sub-sample for cut 3, and it equates to a 50.39% error rate reduction. The largest error rate reduction observed is 51.61 percentage points, in the 10% sub-sample for cut 4, corresponding to a 1.6% absolute performance improvement.

4.3.3 Regular classes *FT*

This section presents the findings of the FashionMNIST regular classes *FT* experiment where the convolutional layers after the cut point are removed from the model during fine-tuning phase (see Table 3.2).

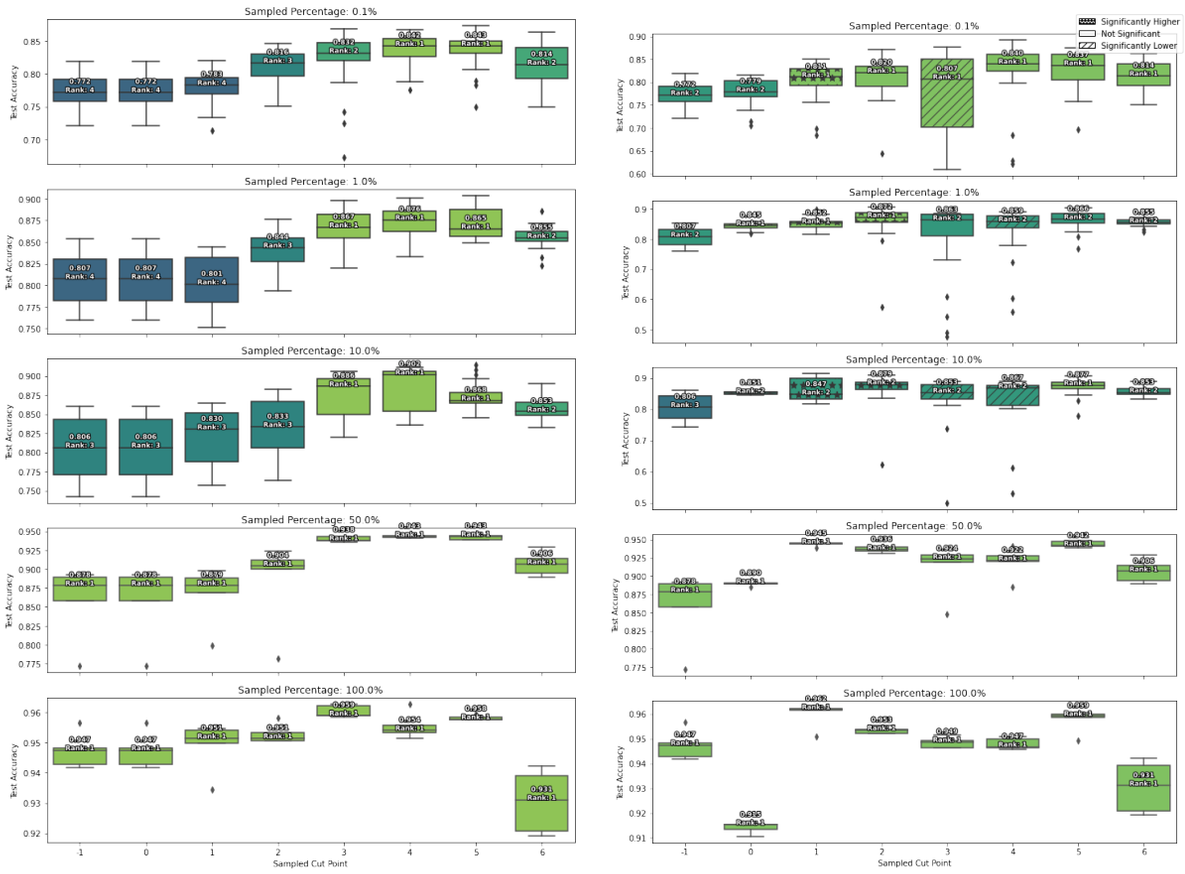
The truncated models, after removing all CNN layers following the cut and up to the linear layer, exhibit a reduced trainable parameter count compared to their non-truncated counterparts at the same cut (see Table 3.5). In a truncated model, the only trainable parameters are in the linear layer which limits its expressive power. Moreover, since this linear layer has no non-linearity of its own, the model's expressiveness is further reduced. Consequently, a comparison using the same source model, and the same source and target datasets across truncated and non truncated variants of an experiments lends itself well to answer Research question 3.

The results, illustrated in Figure 4.4, show no big differences in median accuracies between the best performing cuts in the truncated and non-truncated versions of the experiment for any given sub-sampling. For instance, at 0.1% sub-sampling, the non-truncated optimal cut (cut 5) achieves 84.3% accuracy, while the truncated optimal cut (cut 4) achieves 84%.

Generally, in low resource sub-samples (up to 10%), truncated optimal cuts have slightly lower accuracies compared to non-truncated ones. However, in medium and high resource scenarios (50% and above), the truncated optimal cuts outperform the non-truncated ones. The most significant difference is observed at 100% sub-sampling, where the truncated optimal cut, cut 1, achieves 96.2% accuracy, compared to the non-truncated optimal cut, cut 3, which achieves 95.9% accuracy.

Cut 0 is particularly noteworthy in this experiment. Since truncation is applied, cutting the model at CNN layer 0 equates to removing all CNN layers. The model in this case consists of just the linear layers, which is fed the original image flattened. Interestingly, its performance only goes below baseline at 100% sub-sampling.

Another interesting observation arises from the Wilcoxon tests comparing the same cuts in truncated and non-truncated variants. As shown in Figure 4.4b, in low resource scenarios (up to 10%), the earlier truncated cuts (cuts 1 and 2) tend to be significantly higher than their non-truncated counterparts in statistical terms. Conversely, the middle cuts (cuts 3 and 4) tend to be



(a) *truncate=False*
reported p-values: Table A.2

(b) *truncate=True*
reported p-values: Table A.5

Figure 4.4: FASHIONMNIST REGULAR CLASSES, COMPARISON BETWEEN *FR* VS. *FT* SETTINGS. *The p-values that led to the cross-comparison between the two experiments (visualized as the hatches on the Figure 4.4(b)) can be found in Table A.6.*

statistically lower, while cut 5 does not show significant differences across experiments. Finally, for any given sub-sampling, cut 6 is identical across experiments because truncation does not make a difference when there are no layers to truncate.

Lastly, medium and high sub-samplings (50% and 100%) depart from the arched pattern that is observed in several other experiments, in favor of a, so to speak, double arch: the best performance is achieved by cut 1, closely followed by cut 5, with the rest of the cuts forming the double arch round those two peaks.

The most substantial absolute performance improvement observed was an increase of 7.3 percentage points in accuracy. This improvement occurred in the 10% sub-sample for cut 2, equating to an 18.79% error rate reduction. The largest error rate reduction observed was 54.91 percentage points, seen in the 50% sub-sample for cut 1. This corresponds to a 6.7% absolute performance improvement.

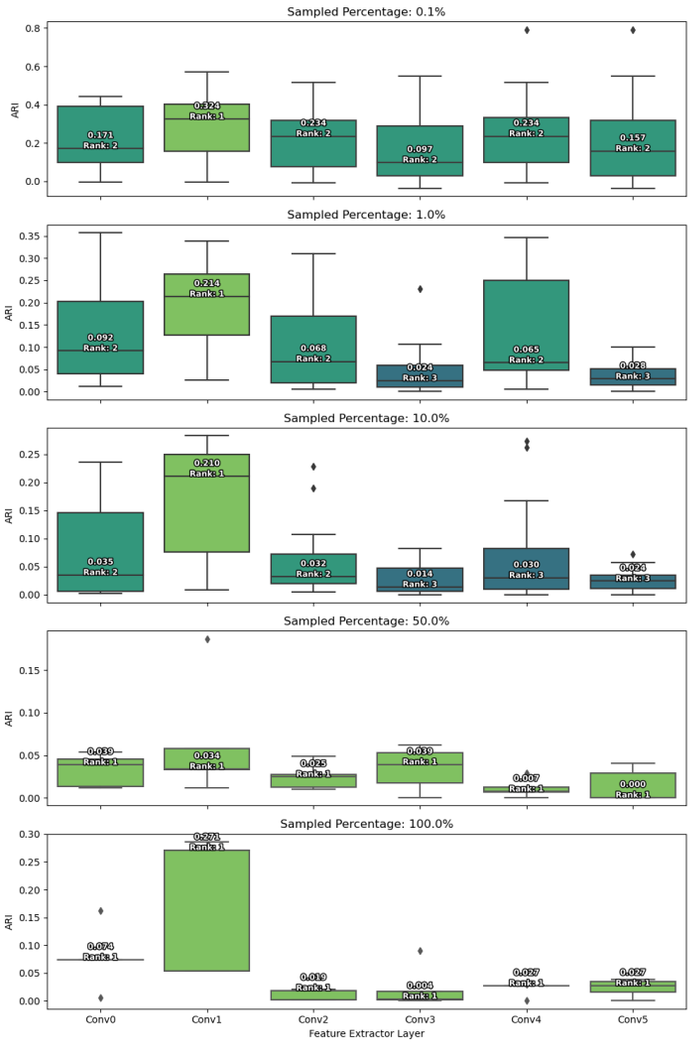


Figure 4.5: FASHIONMNIST REGULAR CLASSES CLUSTERING EXPERIMENT. *The p-values that led to the ranking of the cuts can be found in Table A.7.*

4.3.4 Regular classes cluster

In a similar vein to the FashionMNIST regular classes *FT* experiment (Section 4.3.3), we proceed to directly extract the features for the fine-tuning samples from all the convolutional layers of the pre-trained model. Subsequently, we utilize these features to train a clustering algorithm, which serves to cluster and classify the fine-tune samples. Details can be found in Section 3.2.2.

The results presented in Figure 4.5 demonstrate that cut=2 consistently outperforms all other cuts across various data sub-sampling percentages, with the exception of the 50% sub-sample of the target data. At this data percentage, the performance of cut=2 significantly declines, aligning with the performance of the other cuts. It is important to note that Conv0 refers to the clustering performance using features extracted from the first convolutional layer, so it is equivalent to cut=1 in the other experiment settings, as explained in Section 3.3.4.

The cuts that perform the worst appear to be cut=4, cut=5 (in most percentages), and cut=6. This observation is reasonable, as features extracted from the deeper convolutional layers become

increasingly specialized to the source dataset, rendering them less suitable for application to the target dataset samples.

Unlike other experimental settings, the optimal cuts and their rankings do not appear to be dependent on the percentage of data sub-sampling; their relative performance remains largely consistent.

4.4 MNIST to FashionMNIST

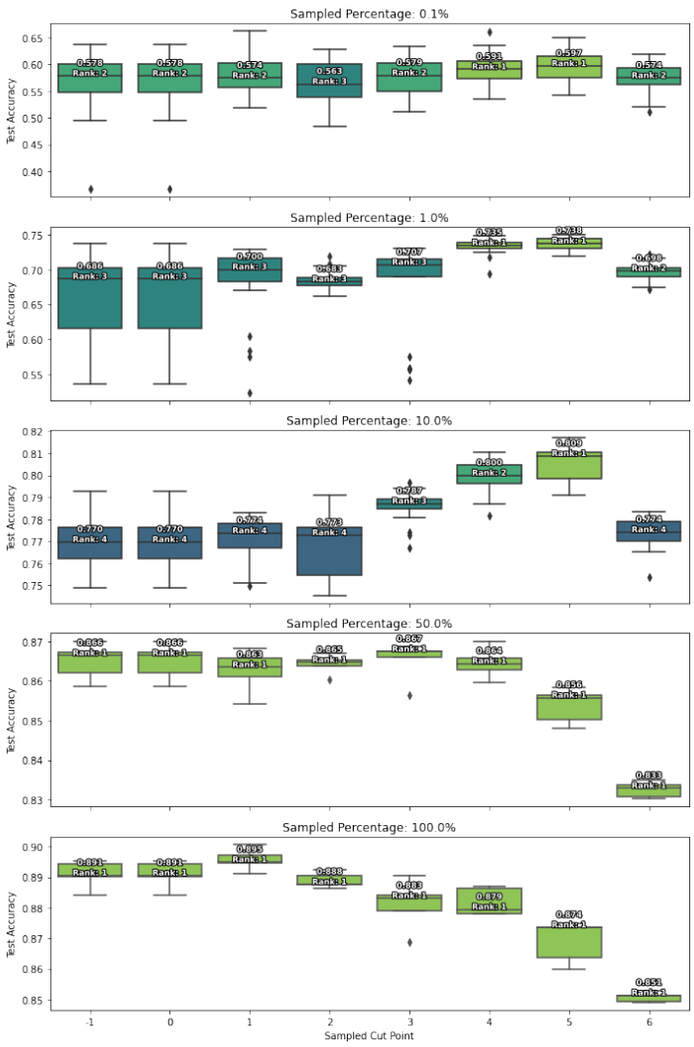


Figure 4.6: MNIST TO FASHIONMNIST REGULAR CLASSES IN *FR* SETTING. *The p-values that led to the ranking of the cuts can be found in Table A.9.*

The MNIST to FashionMNIST regular classes *FR* experiment employs source and target sets from distinct datasets aiming to increase the difference between the source and target dataset samples, as given in Table 3.1, and in the *FR* setting, as detailed in Table 3.2. Furthermore, we use

the MNIST to FashionMNIST regular classes pre-trained model with the pre-training accuracies given in Table 4.1.

In Figure 4.6, it can be seen that the fourth and fifth cuts outperform other cuts across various low-resource sub-sample sizes from 0.1% to 10%, where the fifth cut especially outperforms all other cuts in the 1% of the data.

The sixth cut either matches or underperforms the baseline across all sub-sampling percentages, meaning freezing all CNN layers of the pre-trained model results in considerably worse performance than freezing up to the fifth layer. This indicates that the transferability of the features drops significantly from layer 5 to 6 (see Table 3.3 for more details on what different cuts correspond to in the network).

With 50% and 100% of the target data, the optimal cuts shift to earlier layers. At 50% sub-sampling, the optimal cuts are roughly cuts 2, 3, or 4. With 100% of the target data, the optimal cuts seem to be cuts 0 or 1, although there is no statistical significance between the cuts at these data percentages. This pattern resembles the MNIST regular classes *FR* experiment (Section 4.2), where the optimal cut in the network's layers moves to earlier layers as a function of the target data percentage, although this shift is not as clearly observable.

The biggest absolute performance improvement seen in the experiment is of of 5.2 percentage points in accuracy. This improvement occurred in the 1% sub-sample for cut 5, equating to an 16.56% error rate reduction. The largest error rate reduction observed was 16.96%, seen in the 10% sub-sample for cut 1. This corresponds to a 3.9% absolute performance improvement.

4.5 FashionMNIST to MNIST

The FashionMNIST to MNIST regular classes *FR* experiment acts as a counterpart to the MNIST to FashionMNIST regular classes *FR* experiment (Section 4.4), yet with some distinct outcomes. As for the experiment setup, the only difference consists of flipping the role of the datasets: in this case FashionMNIST is used for pre-training and MNIST is used for fine-tuning, as given in Table 3.1. The pre-training details of both models can be found in Table 4.1.

The differences between the different cuts are less clearly observable in this experiment compared to the MNIST to FashionMNIST regular classes *FR* (Section 4.4), especially in terms of statistically significant differences between the cuts (see Figure 4.7).

Just like for the MNIST to FashionMNIST regular classes *FR* experiment, the optimal cuts are cuts 4 and 5 at sub-sampling percentages of 0.1% and 1%. However, in this experiment, the optimal cuts move to cuts 2 and 3 with 10% of the data. At 50% the optimal cut moves further backwards to cut 1, which is also the optimal cut at 100% sub-sampling.

The biggest absolute performance improvement seen in the experiment was of of 9.7 percentage points in accuracy. This improvement occurred in the 0.1% sub-sample for cut 5, equating to an 26.07% error rate reduction, which is also the highest error rate reduction in the experiment.

4.6 CIFAR

This section presents the results of various experiments conducted on the CIFAR dataset. First, we explore the effects of different data splits on transfer learning performance, including CIFAR regular classes, reverse classes, and random classes (Table 3.1), all under the default experiment setting *FR*, as detailed in Table 3.2.

Then, we examine the CIFAR regular classes *FT* experiment (Section 4.6.4), which investigates the impact of removing CNN layers after the cut point instead of reinitializing them.

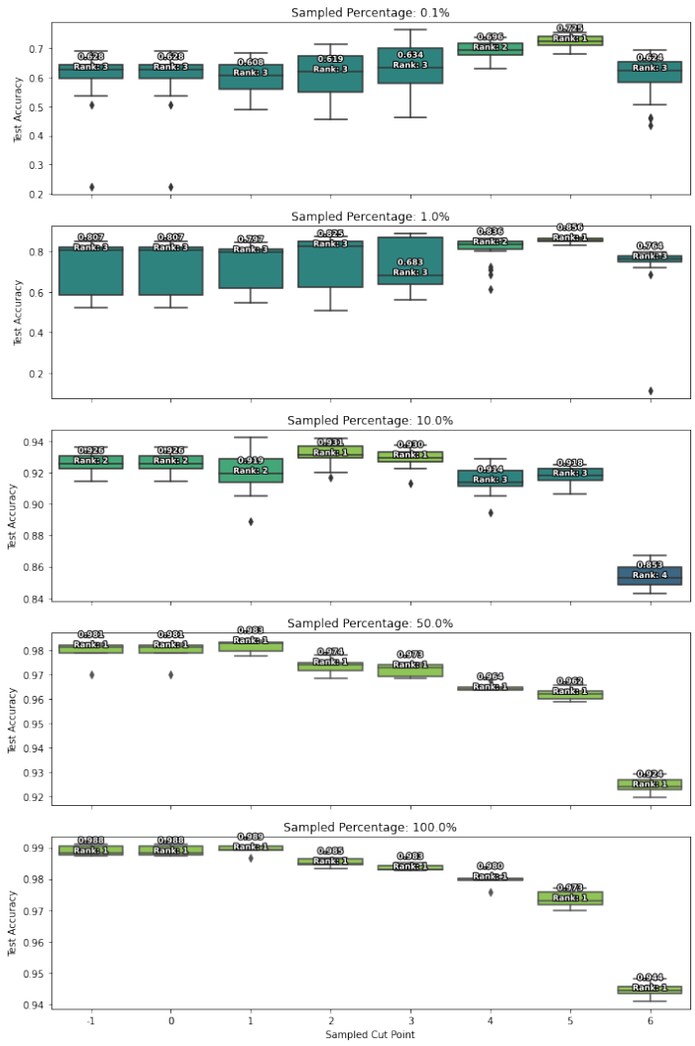


Figure 4.7: FASHIONMNIST TO MNIST REGULAR CLASSES IN *FR* SETTING. *The p-values that led to the ranking of the cuts can be found in Table A.10.*

Note that in Section 4.6.1 and Section 4.6.4, we use the same data split (referred to as *CIFAR regular classes* in Table 3.1) and the same pre-trained model (*CIFAR regular classes* in Table 4.1). However, the CIFAR reverse classes *FR* (Section 4.6.2) and CIFAR random classes *FR* (Section 4.6.3) experiments include the CIFAR reverse classes and CIFAR random classes data splits (Table 3.1) and pre-trained models (Table 4.1) respectively.

4.6.1 Regular Classes *FR*

Analysis of the results of the CIFAR regular classes *FR* configuration depicted in Figure 4.8 reveals a pattern: as the percentage of the target dataset increases, the optimal cut point shifts backwards. Initially, cuts 4 and 5 yield the best results. However, as the dataset sub-sample increases in size to 50%, the second cut emerges as the most effective, eventually leading to the first cut being optimal when the data sub-sample reaches 100%. This indicates a trend towards a need for fewer frozen

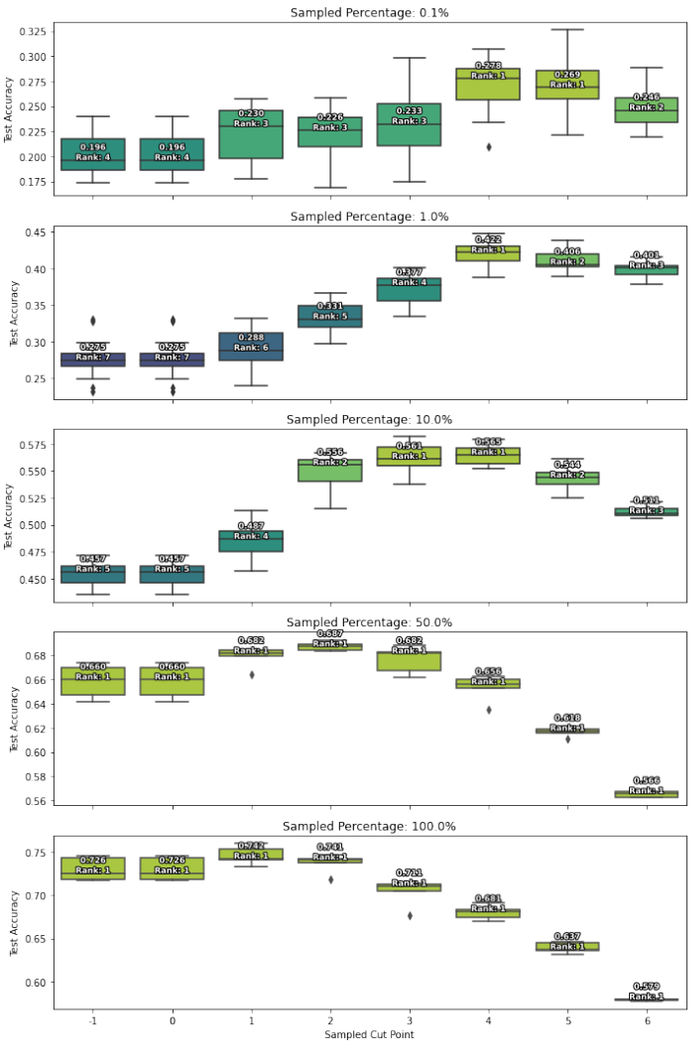


Figure 4.8: CIFAR REGULAR CLASSES IN *FR* SETTING. *The p-values that led to the ranking of the cuts can be found in Table A.11.*

parameters and layers as more data becomes available for fine-tuning.

A noteworthy observation is that mid to late cuts begin to underperform, dropping below baseline accuracy, as soon as the target dataset comprises 50% of the total available data. This suggests that maintaining some parameters frozen may hinder rather than enhance model accuracy as the amount of data increases.

The highest absolute performance improvement observed was an increase of 14.7 percentage points in accuracy. This improvement occurred in the 1% sub-sample for cut 4, equating to an 20.27% error rate reduction, which also happens to be the highest error rate reduction in the experiment.

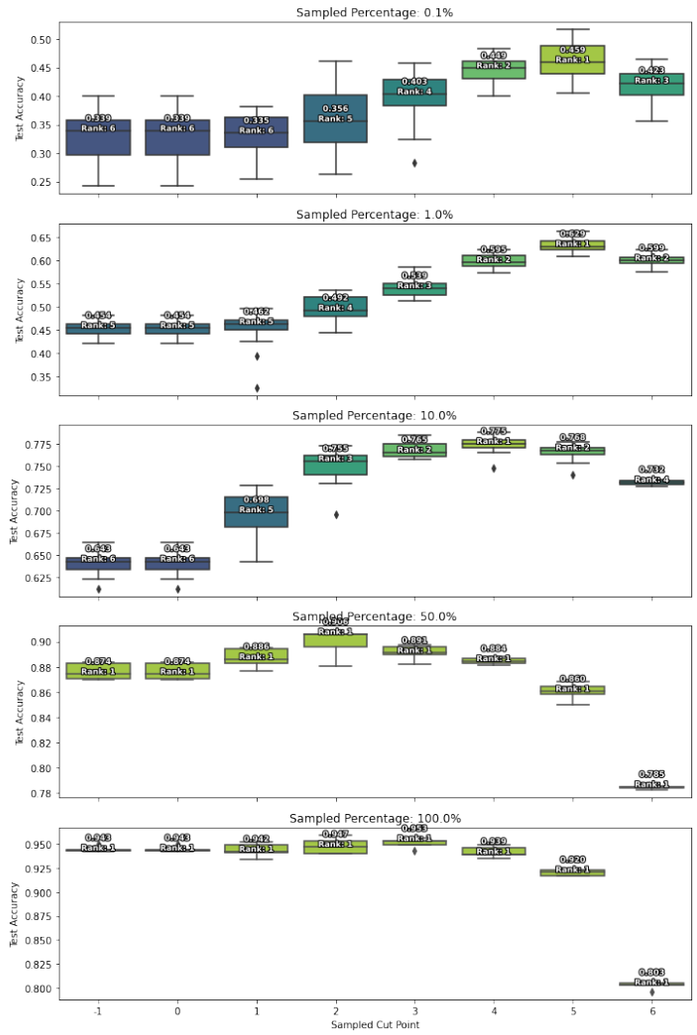


Figure 4.9: CIFAR REVERSE CLASSES IN *FR* SETTING. *The p-values that led to the ranking of the cuts can be found in Table A.14.*

4.6.2 Reverse Classes *FR*

The CIFAR reverse classes *FR* experiment yields similar results to the CIFAR regular classes, with some key variations.

In CIFAR reverse classes *FR* results, the trend observed in data utilization and cut optimization mirrors previous findings. With 0.1% and 1% of the target data, cut 5, which nearly freezes all parameters, emerges as optimal as can be seen in Figure 4.9. As the percentage of utilized data increases, the preference shifts towards cut 4 at the 10% data mark, and eventually to cut 2 at 50% data usage. Upon reaching full data utilization (100%), cut 3, closely followed by cut 2, is the most effective, confirming a similar trend observed in the initial direction of transfer.

Interestingly, models pre-trained on animals consistently leads to superior accuracies compared to models pre-trained on vehicles (see Figure **??**, as evidenced by the baselines. This suggests that the vehicle classification task is inherently easier for the model to learn from the provided data subset.

Notably, the maximal absolute improvement in this scenario consists of 17.5 absolute percentage points. This peak improvement is achieved with just 1% of the data and occurs at cut 5, corresponding to a 32.05% error rate reduction. The highest error rate reduction of 36.97% is achieved at cut 4 with 10% sub-sampling, corresponds to a 13.2 absolute point improvement.

4.6.3 Random Classes *FR*

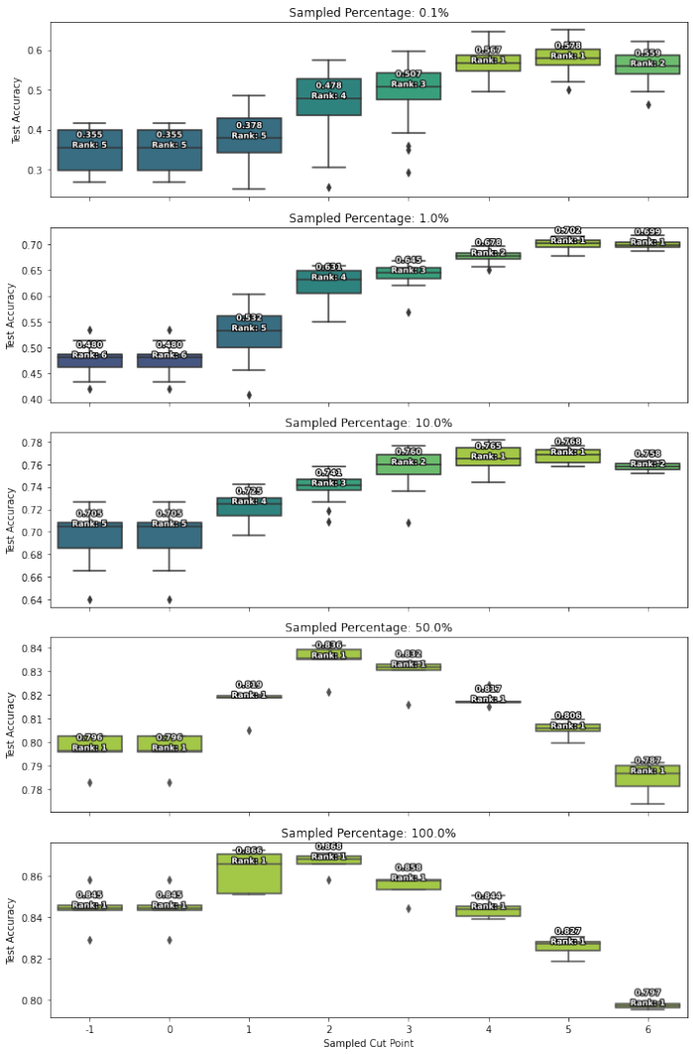


Figure 4.10: CIFAR RANDOM CLASSES IN *FR* SETTING. *The p-values that led to the ranking of the cuts can be found in Table A.15.*

The CIFAR random classes *FR* experiment demonstrates a similar trend to previous CIFAR experiments, yet with bigger absolute improvements and more pronounced arched patterns.

This improvement may be explained by the increased similarity between the source and target data, as both involve a mix of Vehicles and Animals classes (as shown in Table 3.1). Compared to other experiments with CIFAR regular classes and reverse classes data splits, the pre-trained

features in this setting may be more transferable, potentially leveraging pre-training knowledge that is more relevant to the fine-tuning classes.

Looking at the results of the experiment in Figure 4.10, we see that as the data utilization increases, the best performing cut evolves similarly to the prior experiments. Initially, cut 5 is the optimal cut up to 10% of data sub-sampling, then shifting to the second cut as the data volume increases to 50% and 100%.

Furthermore, even with full data utilization (100%), certain cuts continue to surpass the baseline performance, especially the earlier cuts. This outcome is consistent across all CIFAR experiments, underscoring the potential value of fine-tuning, even in higher resource scenarios.

Notably, as data availability increases beyond 50%, the performance of later cuts (e.g., cut 5 and cut 6) diminishes below the baseline, indicating that freezing most parameters no longer contributes positively to accuracy achievements.

The biggest absolute improvement in this scenario consists of 22.3 absolute percentage points. This peak improvement is achieved with just 0.1% of the data and occurs at cut 5. This corresponds to a 34.57% error rate reduction. The highest error rate recorded is of 42.69% at 1% sub-sampling in cut 5, which corresponds to 22.2 absolute point improvement.

4.6.4 Regular classes *FT*

The CIFAR regular classes *FT* experiment demonstrates a maximum improvement of 15.7 absolute percentage points, closely aligning with outcomes observed in prior experiments.

As seen in Figure 4.11(b), results once again mirrors previous findings forming an arched pattern. In low resource scenarios of 0.1% and 1% target data sub-sampling cut 4 emerges as the most effective. At 10% cuts 2, 3 and 4 are tied as the optimal cuts, and at 50% and 100% the best cut is cut 3.

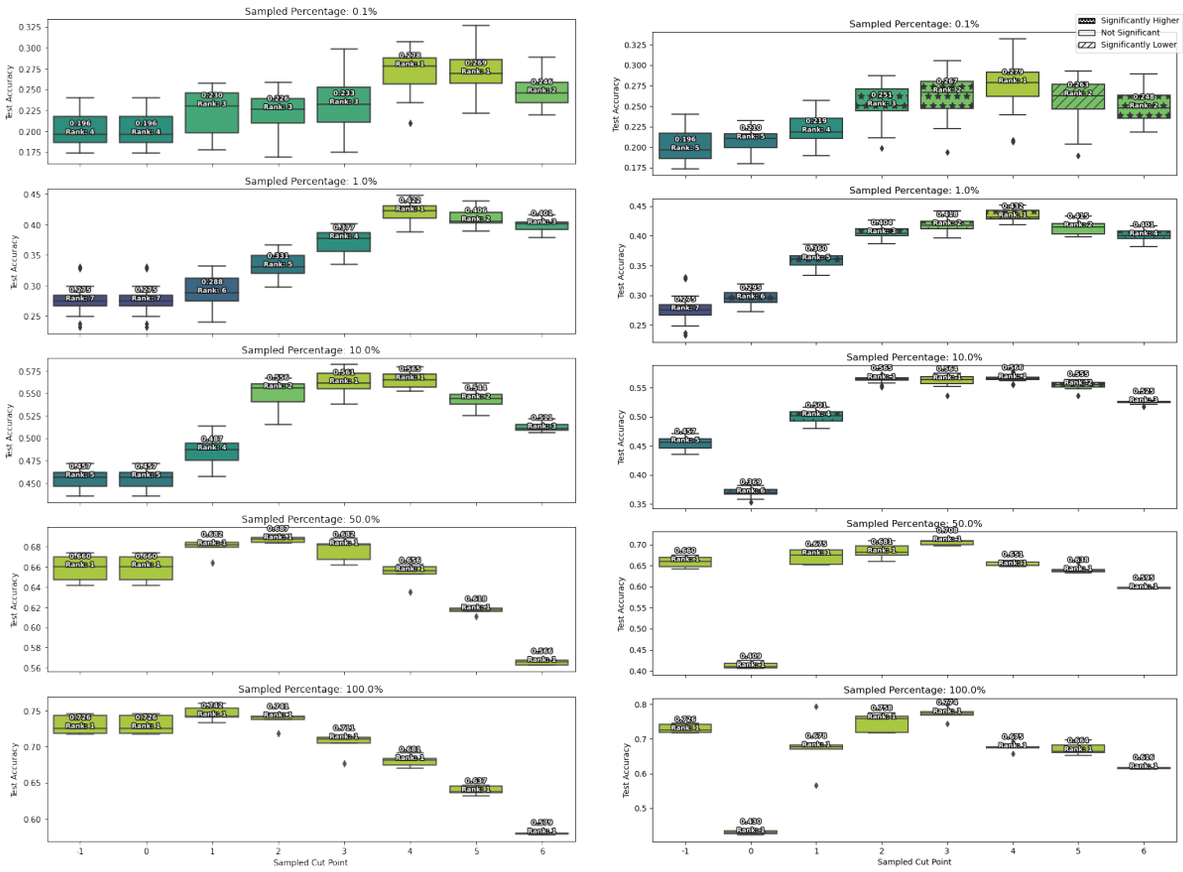
Cut 0, which corresponds to a model without any of CNN layers, effectively functions as a perceptron directly classifying flattened input images. This configuration exceeds baseline performance at low data volumes (0.1% and 1%), though it significantly underperforms as data availability extends to 10% and beyond. This is in constrast for the truncation experiment for FashionMNIST (see Figure 4.4b), for which cut 0 only goes below baseline at 100% sub-sampling.

Comparing the same cut in the truncated and non truncated variants of this experiment with the Wilcoxon test reveals an intriguing aspect:there is somewhat even mix of cuts that are either statistically higher or lower than each other and when comparing their median accuracies, the cuts tend to have similar values. This suggests that truncation does not detrimentally affect performance.

Furthermore, when we compare different cuts across the two experiments we see that, for any given sub-sample, the median accuracy is always slightly higher for the truncated optimal cut when compared to the non-truncated optimal cut. For instance, at 100% sub-sampling, the optimal cut in the non-truncated experiment is cut 1, achieving 74.2% accuracy with 190,790 trainable parameters. In contrast, the optimal cut in the truncated experiment is cut 3, which achieves 77.4% accuracy using 98,310 trainable parameters (see Table 3.5).

Additionally, the experiment highlights a diminished underperformance of later cuts (such as five and six) in higher data regimes, suggesting a less pronounced decline in effectiveness compared to the non-truncated approach.

The most substantial absolute performance improvement observed was an increase of 15.7 percentage points in accuracy. This improvement occurred in the 1% sub-sample for cut 4, equating to an 21.65% error rate reduction, which also happens to be the highest error rate reduction observed in the experiment.



(a) *truncate=False*
reported p-values: Table A.11

(b) *truncate=True*
reported p-values: Table A.12

Figure 4.11: CIFAR REGULAR CLASSES, COMPARISON BETWEEN *FR* VS. *FT* SETTINGS. *The p-values that led to the cross-comparison between the two experiments (visualized as the hatches on the Figure 4.11(b)) can be found in Table A.13. We should mention that the cut=6 across the two experiments are not exactly the same, as a result of a change in the hardware when running the FT experiment.*

Chapter 5

Discussion

In the discussion section, we provide a comparison of our experiments, revisiting our research questions to determine which answers can be provided for them. We analyze the results to highlight trends and patterns that not only address the initial research questions but also reveal additional insight. This analysis helps to contextualize our findings and serves as a starting point to delineate future work.

RQ 1: Should the post-cut layers be reinitialized or used as weight initialization for fine-tuning? As detailed in Section 4.3.1, we conducted two experiments using the FashionMNIST dataset, with the only variation being whether or not reinitialization occurred after the cut. While, for any given sub-sample, the position of the optimal cut varied, the median accuracy of the optimal cuts across the two experiments remained similar or slightly higher in the *reinit=True* scenario. These findings suggest that there is no clear winner between the two approaches in terms of performance, at least in the context of the FashionMNIST dataset.

However, the approaches are not directly equivalent. The *reinit=True* scenario exhibits the arched pattern seen in other experiments, where middle or late cuts are optimal, and visually, the other cuts form an arch them. In contrast, the *reinit=False* scenario shows a flatter trend, with most cuts performing similarly. This flatter trend could be advantageous in situations where trial and error is costly, as there is less variance between the optimal cut and others, increasing the likelihood of a random choice being close to optimal.

RQ 2: What is the effect of source and target datasets on the optimal cut? To answer the research question 2, we start with examining the role of similarities between source and target datasets in determining the optimal cut. We compare FashionMNIST and CIFAR random classes *FR* to their corresponding, non-randomized experiments (regular classes *FR*), in which the randomized source and target data splits are qualitatively more similar than their standard counterparts (see Section 3.1.1). We abstain from directly comparing accuracy values between experiments, given that the pre-trained models differ across them. Instead, we simply compare visual trends that emerge in each of them.

In the FashionMNIST random classes *FR* experiment, optimal cuts mostly overlap with those in the FashionMNIST regular classes *FR* experiment. On the other hand, in the CIFAR random classes *FR* experiment, the optimal cut occurs at a later layer for some data percentages (1%, 10%, and 100%) compared to the regular experiment, while they coincide for the remaining sub-sampling percentages. We expected the random experiments to show later optimal cuts compared to the regular experiments at the same data sub-sampling percentages. This anticipation stemmed from the idea that as datasets become more similar, the features retained their transferability, potentially leading to optimal cuts at later layers. However, this expectation wasn't consistently confirmed by the data.

To investigate whether fine-tuning experiments with different data splits also produce the same patterns we observe in the primary experiment of each dataset (MNIST, FashionMNIST and CIFAR regular classes *FR*), we conduct the same experiments but with different source and target data splits.

In both CIFAR reverse classes and random classes *FR* experiments, we observe arched patterns consistent with the arched pattern produced by the CIFAR regular classes *FR* experiment, where the peak of the arch moves backward with increasing amounts of target data. However, these patterns do not completely match. In the reverse classes case, the arch is more pronounced in low-resource scenarios, while in the random classes case, the arch becomes more pronounced in high-resource scenarios. It is difficult to determine whether these observations are due to the properties of the data splits or the use of separate pre-trained models for each experiment.

Comparing the FashionMNIST random classes and regular classes *FR* experiments, we observe an arched pattern in the regular classes case with consistent optimal cut positions across data sub-sampling percentages. In contrast, the random classes experiment also shows an arched pattern, but the differences between the cuts are subtler. Multiple optimal cuts appear in the same neighborhood across all sub-sampling percentages, similar to the FashionMNIST regular classes *F* experiment discussed above.

In the MNIST to FashionMNIST and FashionMNIST to MNIST regular classes *FR* experiments, we observe an arched pattern in both cases where the arch moves backward to earlier layers with increasing amounts of target data. However, the trend is less visible and noisier compared to other experiments, likely due to the increased difference between the source and target datasets in the intra-dataset experiments.

These results highlight that the patterns remain mostly consistent within the inter-dataset experiments, with minor differences. However, the increased disparity between the source and target datasets in the intra-dataset experiments seems to introduce some noise into the observed patterns.

RQ 3: Can the post-cut layers be completely removed for parameter efficiency without significantly impacting performance? Our standard setup, where layers are frozen up to the cut and reinitialized thereafter, can potentially provide a parameter-efficient solution for low-resource target datasets that lack sufficient samples to saturate a more parameter-rich model.

In Research Question 3, we explore this notion further by limiting the model's expressivity with reduced parameter counts and no non-linearities in the learnable part of the model. As shown in Table 3.5, the truncation approach generally succeeds in offering an alternative with a reduced parameter count. In fact, it is only in cut 1 for the CIFAR10 dataset, where the truncated model's parameter count is higher than its non-truncated counterpart due to the three channels of the input image and the absence of pooling to reduce dimensionality after layer 1.

In both in the CIFAR10 (see Section 4.6.4) and FashionMNIST (see Section 4.3.3) experiments, when comparing the truncated and non-truncated variants, the optimal cus tend to emerge in different positions. However, when comparing the median accuracy of the optimal cut for the same sub-sample, truncated and non-truncated cuts tend to have very similar values.

There are only a few cases where accuracies change beyond 1 absolute point. For FashionMNIST at 10%, the best non-truncated cut, cut 4 (75,587 trainable parameters), is 2.3 percentage points ahead of the optimal truncated cut, cut 2 (37,675 trainable parameters). For CIFAR10 at 50%, the truncated optimal cut at cut 3 (98,310 trainable parameters) is 2.1 percentage points ahead of the non-truncated optimal cut at cut 2 (153,862 trainable parameters); and at 100%, the truncated optimal cut 3 (98,310 trainable parameters) is 3.2 percentage points ahead of the non-truncated optimal cut at cut 1 (190,790 trainable parameters).

The data therefore would support the idea that, for datasets like FashionMNIST and CIFAR10 and for the chosen CNN architecture, truncation is a viable alternative to reduce parameter counts

without sacrificing performance.

RQ 4: Can information from the feature maps of target dataset samples generated by the pre-trained model be used to compare the transferability of different layers? In the FashionMNIST regular classes cluster experiment (Section 4.3.4), it is evident that features extracted from the second CNN layer (cut=2) consistently provide the best clustering performance across various data sub-sampling percentages. Unlike other experiments where the optimal cut might vary depending on the amount of target data, this experiment shows a unique pattern. The preference for cut 2 becomes more pronounced as the amount of target data increases, maintaining its status as the optimal cut regardless of the data size.

This phenomenon can be explained as follows: unlike truncation or other experiments, clustering directly utilizes feature maps from frozen CNN layers without applying any (linear or non-linear) transformation. Since it does not have access to the target labels, it cannot learn from them. Increasing the amount of training data merely reduces noise in the results, as the clustering algorithm's performance is evaluated on the training data itself.

The objective of the clustering experiment is to determine the extent to which pre-trained features are transferable to the target data at each layer, isolating factors such as the number of trainable parameters. The clustering experiment would suggest that the second CNN layer contains the most transferable features, implying that for fine-tuning, it might be beneficial to retain this layer. It should be noted, however, that it is not enough information on its own to determine the optimal cut, since it disregards the factors such as number of trainable parameters and the contribution of features learned during fine-tuning.

One interesting observation comes from the FashionMNIST regular classes *FR* experiment. At the 0.1% and 1% data percentages, there is a notable improvement when moving from cut=1 to cut=2, highlighting the significance of the second convolutional layer. This patterns would support the intuition behind our clustering experiment results, however further experiments would be necessary to validate the applicability of the clustering approach in different datasets and architectures.

Main goal: Explore the conditions that lead to identifying the optimal cut point in layer-wise transfer learning In many experiments in Chapter 4, we report an arched pattern where the optimal cut(s) move to the earlier CNN layers with an increasing amount of target data. This arched pattern results from the trade-off between decreasing transferability in successive pre-training layer and the learned features contained in each of them. In the earlier cuts, we discard much of this information by reinitializing most of the layers. However, as layers become more specialized for the source dataset in later cuts, their transferability and usefulness for processing the target dataset samples decrease. Consequently, performance initially increases with each cut, reaching a peak (the optimal cut or cuts), and then starts to decline, forming an arched pattern.

In fine-tuning, we must consider two additional factors besides the learned features and their transferability: the amount of target data and the number of trainable parameters. In a low-resource scenario, we prefer fewer trainable parameters and more information from the pre-training, even if that information is not fully transferable to the target dataset. Limited data necessitates retaining as much pre-trained information as possible to prevent overfitting, which can occur with too many parameters and limited data.

In a high-resource scenario, we can learn more features, from scratch, solely using the target data due to its larger size. This makes it possible to reinitialize and discard more information from the source dataset that may not be fully compatible with the target task. Consequently, an earlier cut is preferable with more available target data, as learning target dataset high-abstraction features end-to-end may be more efficient than overwriting less transferable source dataset features in the later CNN layers.

This pattern is evident in the CIFAR regular classes *FR*, CIFAR random classes *FR*, and CIFAR reverse classes *FR* experiments, and to a lesser extent in the MNIST regular classes *FR*, MNIST to FashionMNIST regular classes *FR*, and FashionMNIST to MNIST regular classes *FR* experiments. The observed pattern can be explained as an arched pattern where the arch (and the optimal cuts) shift backward with increased target data.

Some interesting exceptions to this pattern include the FashionMNIST experiments. For example, in the FashionMNIST regular classes *FR* experiment, the optimal cuts are consistently cuts 3, 4, and 5 across various sub-sampling percentages, and while we do observe the arched pattern, the optimal cuts do not move backward as expected. There is a shift from cuts 4 and 5 to cuts 3, 4, and 5 as the target data increases from 0.1% to 1%. Additionally, in the FashionMNIST random classes *FR* and FashionMNIST regular classes *F* experiments, we observe the optimal cuts moving backward, albeit with relatively smaller differences between each cut compared to the CIFAR experiments.

Chapter 6

Conclusion

This project investigates the effectiveness of transfer learning by examining different fine-tuning scenarios across various datasets. Our goals was to understand how to best adapt pre-trained models to new tasks, exploring the impact of different design choices on performance across datasets with varying levels of complexity and similarity.

Our findings demonstrate that the efficacy of transfer learning hinges on multiple factors, including those pre-defined by the task, such as the similarities between the source and target datasets, the amount of available training data, and the model architecture. Additionally, factors related to the design choices of layer-wise fine-tuning, which can be adjusted, play a significant role. These design choices include freezing the pre-cut layers, reinitializing or truncating the post-cut layers, and determining the position of the cut point.

In this project, we investigated the effects of some of these design choices and provided a framework for deciding on an optimal cut for different target data amounts. Our findings and suggestions are as follows:

Our data suggests that retaining pre-trained features by not reinitializing post-cut layers, particularly in scenarios with limited data, may reduce variability across cuts, making random cut choices closer to being optimal. Additionally, truncating post-cut layers can serve as a viable low-parameter alternative, providing comparable performance, especially in resource-constrained settings or when overfitting is a concern due to limited data.

Finding a good cut point is beneficial for improving fine-tuning performance. Therefore, we studied the patterns that emerged from the trial and error of testing multiple candidate cuts. We conducted experiments involving different data splits, sub-sampling percentages, and cut points, all under our default setting (*FR*) of freezing pre-cut layers and reinitializing post-cut layers.

The majority of these experiments revealed a consistent arched pattern, where the optimal cut point shifts backward to earlier layers with increasing volumes of target data. This pattern suggests a reduced reliance on pre-trained features as more task-specific knowledge can be acquired directly from the target data.

It's worth noting that this pattern emerged consistently in most experiments, with FashionMNIST experiments being exceptions. While FashionMNIST experiments exhibited an arched pattern, the optimal cuts did not shift backwards with increasing amounts of target data. Despite this, the overall consistency suggests a potentially universal pattern in many image classification datasets, which could help users reason about an approximate optimal cut point based on model complexity and target data volume. Further experiments with different architecture variations and datasets are necessary to confirm and strengthen our findings.

As an alternative approach, we propose conducting a clustering experiment to identify which CNN layers of the pre-trained model contain transferable features to the target dataset. This information can be useful for determining an optimal cut point more efficiently. However, experiments on other datasets, as well as further refinements to the technique itself are necessary to confirm

its effectiveness.

In summary, the decision-making process for fine-tuning design choices may proceed as follows: Firstly, if information is available on which cuts may contain the most transferable features from the pre-trained model, then those layers should ideally be retained, positioning the chosen cut point after these layers if possible. In scenarios with limited target data, such as around 1% of the source data, it may be preferable to favor later cuts or opt for an earlier cut while refraining from reinitializing the post-cut layers. Similarly, when the amount of target data is comparable to the source data, earlier cuts should be favored. In cases of extremely limited data, less than 1%, options include freezing all layers except the linear layer for training or selecting a later cut while removing post-cut CNN layers to mitigate overfitting.

6.0.1 Limitations

Despite the valuable insights gained, there are several limitations and areas for future research that could deepen our understanding.

We explored fixed-architecture CNN models with a set number of channels and without additional regularization techniques like batch normalization or dropout. Future research could also investigate more modern architectures, like Vision Transformers (Kolesnikov et al., 2021). Incorporating these architectures and regularization techniques will allow for a more comprehensive understanding of transfer learning across varied models and datasets.

Our statistical analysis was constrained by computational resources. We ran higher subsampling percentages with fewer repeats, limiting their statistical significance in the Wilcoxon test (Section 3.3.3). Running these percentages with more repetitions would provide greater statistical rigor and validate the observations made in this report in the high resource regime.

Using relatively simple datasets like MNIST, FashionMNIST, and CIFAR-10 helped us explore the emergence of transfer learning trends. However, these trends should be verified on larger, more complex datasets like CIFAR-100 (Krizhevsky, 2012) and ImageNet (Deng et al., 2009). Such experiments will provide insights into whether the observed patterns hold for more challenging and realistic datasets.

The clustering approach (Section 3.2.2) was exploratory, serving as a proof of concept. It requires further refinement to become reliable measure.

The clustering results suggest that the alignment between truncation and clustering experiments might indicate the right direction for assessing the transferability of learned features at each CNN layer, which might be an exciting direction for the future work. However, it is important to note that the clustering experiment is still preliminary. To further confirm the findings of the clustering experiment in FashionMNIST, it should be tested on the other datasets as well.

By addressing these limitations and expanding the scope of research, we can better understand transfer learning, enabling researchers and practitioners to make more informed decisions across different models, datasets, and tasks.

Appendix A

Statistical Tests Outputs

Percent	Cut 1	Cut 2	statistic	p-value
0.1	-1	0	0	1.000000
0.1	-1	1	127	0.352520
0.1	-1	2	77	0.020275
0.1	-1	3	1	0.000000
0.1	-1	4	0	0.000000
0.1	-1	5	0	0.000000
0.1	-1	6	0	0.000000
0.1	0	1	127	0.352520
0.1	0	2	77	0.020275
0.1	0	3	1	0.000000
0.1	0	4	0	0.000000
0.1	0	5	0	0.000000
0.1	0	6	0	0.000000
0.1	1	2	116	0.219986
0.1	1	3	9	0.000002
0.1	1	4	0	0.000000
0.1	1	5	0	0.000000
0.1	1	6	16	0.000010
0.1	2	3	25	0.000054
0.1	2	4	0	0.000000
0.1	2	5	1	0.000000
0.1	2	6	29	0.000103
0.1	3	4	10	0.000003
0.1	3	5	41	0.000556
0.1	3	6	144	0.633827
0.1	4	5	123	0.299612
0.1	4	6	12	0.000004
0.1	5	6	29	0.000103
1.0	-1	0	0	1.000000
1.0	-1	1	134	0.457846
1.0	-1	2	143	0.633827
1.0	-1	3	91	0.055070
1.0	-1	4	0	0.000000
1.0	-1	5	0	0.000000
1.0	-1	6	51	0.001816
1.0	0	1	134	0.457846
1.0	0	2	143	0.633827
1.0	0	3	91	0.055070
1.0	0	4	0	0.000000
1.0	0	5	0	0.000000
1.0	0	6	51	0.001816
1.0	1	2	119	0.252104
1.0	1	3	41	0.000556
1.0	1	4	0	0.000000
1.0	1	5	11	0.000003
1.0	1	6	49	0.001453
1.0	2	3	62	0.005579

Percent	Cut 1	Cut 2	statistic	p-value
1.0	2	4	0	0.000000
1.0	2	5	0	0.000000
1.0	2	6	57	0.003419
1.0	3	4	10	0.000003
1.0	3	5	105	0.126648
1.0	3	6	151	0.771159
1.0	4	5	19	0.000018
1.0	4	6	0	0.000000
1.0	5	6	39	0.000430
10.0	-1	0	0	1.000000
10.0	-1	1	71	0.216167
10.0	-1	2	84	0.452375
10.0	-1	3	32	0.004860
10.0	-1	4	0	0.000002
10.0	-1	5	4	0.000013
10.0	-1	6	17	0.000395
10.0	0	1	71	0.216167
10.0	0	2	84	0.452375
10.0	0	3	32	0.004860
10.0	0	4	0	0.000002
10.0	0	5	4	0.000013
10.0	0	6	17	0.000395
10.0	1	2	96	0.756166
10.0	1	3	51	0.044054
10.0	1	4	0	0.000002
10.0	1	5	10	0.000082
10.0	1	6	47	0.029575
10.0	2	3	28	0.002712
10.0	2	4	0	0.000002
10.0	2	5	6	0.000027
10.0	2	6	28	0.002712
10.0	3	4	1	0.000004
10.0	3	5	74	0.261099
10.0	3	6	97	0.784126
10.0	4	5	0	0.000002
10.0	4	6	0	0.000002
10.0	5	6	16	0.000322
50.0	-1	0	0	1.000000
50.0	-1	1	2	0.187500
50.0	-1	2	0	0.062500
50.0	-1	3	1	0.125000
50.0	-1	4	1	0.125000
50.0	-1	5	0	0.062500
50.0	-1	6	0	0.062500
50.0	0	1	2	0.187500
50.0	0	2	0	0.062500
50.0	0	3	1	0.125000

Percent	Cut 1	Cut 2	statistic	p-value
50.0	0	4	1	0.125000
50.0	0	5	0	0.062500
50.0	0	6	0	0.062500
50.0	1	2	5	0.625000
50.0	1	3	4	0.437500
50.0	1	4	7	1.000000
50.0	1	5	0	0.062500
50.0	1	6	0	0.062500
50.0	2	3	2	0.187500
50.0	2	4	6	0.812500
50.0	2	5	0	0.062500
50.0	2	6	0	0.062500
50.0	3	4	6	0.812500
50.0	3	5	0	0.062500
50.0	3	6	0	0.062500
50.0	4	5	0	0.062500
50.0	4	6	0	0.062500
50.0	5	6	0	0.062500
100.0	-1	0	0	1.000000
100.0	-1	1	0	0.062500
100.0	-1	2	3	0.312500
100.0	-1	3	4	0.437500
100.0	-1	4	3	0.312500
100.0	-1	5	0	0.062500
100.0	-1	6	0	0.062500
100.0	0	1	0	0.062500
100.0	0	2	3	0.312500
100.0	0	3	4	0.437500
100.0	0	4	3	0.312500
100.0	0	5	0	0.062500
100.0	0	6	0	0.062500
100.0	1	2	0	0.062500
100.0	1	3	0	0.062500
100.0	1	4	0	0.062500
100.0	1	5	0	0.062500
100.0	1	6	0	0.062500
100.0	2	3	7	1.000000
100.0	2	4	7	1.000000
100.0	2	5	1	0.125000
100.0	2	6	0	0.062500
100.0	3	4	5	0.625000
100.0	3	5	0	0.062500
100.0	3	6	0	0.062500
100.0	4	5	0	0.062500
100.0	4	6	0	0.062500
100.0	5	6	0	0.062500

Table A.1: The reported p-values correspond to pairwise comparisons conducted using the Wilcoxon signed-rank test, specifically comparing each pair of cuts within individual percentages, as a result of the empirical experiments from the first 5 classes of MNIST (pretraining) to last 5 classes (finetuning). The rows marked in green show the cuts which are significantly different from each other. See Section 3.3.3 for more details.

Percent	Cut 1	Cut 2	statistic	p-value
0.1	-1	0	0	1.000000
0.1	-1	1	95	0.070984
0.1	-1	2	2	0.000000
0.1	-1	3	26	0.000064
0.1	-1	4	0	0.000000
0.1	-1	5	0	0.000000
0.1	-1	6	25	0.000054
0.1	0	1	95	0.070984
0.1	0	2	2	0.000000
0.1	0	3	26	0.000064
0.1	0	4	0	0.000000
0.1	0	5	0	0.000000
0.1	0	6	25	0.000054
0.1	1	2	4	0.000000
0.1	1	3	41	0.000556
0.1	1	4	0	0.000000
0.1	1	5	3	0.000000
0.1	1	6	23	0.000038
0.1	2	3	71	0.012466
0.1	2	4	23	0.000038
0.1	2	5	20	0.000022
0.1	2	6	150	0.750993
0.1	3	4	57	0.003419
0.1	3	5	73	0.014722
0.1	3	6	100	0.095733
0.1	4	5	138	0.524913
0.1	4	6	46	0.001027
0.1	5	6	60	0.004605
1.0	-1	0	0	1.000000
1.0	-1	1	157	0.894860
1.0	-1	2	33	0.000188
1.0	-1	3	5	0.000001
1.0	-1	4	0	0.000000
1.0	-1	5	0	0.000000
1.0	-1	6	0	0.000000
1.0	0	1	157	0.894860
1.0	0	2	33	0.000188
1.0	0	3	5	0.000001
1.0	0	4	0	0.000000
1.0	0	5	0	0.000000
1.0	0	6	0	0.000000
1.0	1	2	19	0.000018
1.0	1	3	1	0.000000
1.0	1	4	0	0.000000
1.0	1	5	0	0.000000
1.0	1	6	0	0.000000
1.0	2	3	38	0.000376

Percent	Cut 1	Cut 2	statistic	p-value
1.0	2	4	17	0.000012
1.0	2	5	21	0.000027
1.0	2	6	72	0.025842
1.0	3	4	98	0.085139
1.0	3	5	125	0.325392
1.0	3	6	79	0.023650
1.0	4	5	112	0.181730
1.0	4	6	31	0.000140
1.0	5	6	54	0.002508
10.0	-1	0	0	1.000000
10.0	-1	1	65	0.142906
10.0	-1	2	46	0.026642
10.0	-1	3	8	0.000048
10.0	-1	4	1	0.000004
10.0	-1	5	0	0.000002
10.0	-1	6	7	0.000036
10.0	0	1	65	0.142906
10.0	0	2	46	0.026642
10.0	0	3	8	0.000048
10.0	0	4	1	0.000004
10.0	0	5	0	0.000002
10.0	0	6	7	0.000036
10.0	1	2	73	0.245487
10.0	1	3	5	0.000019
10.0	1	4	4	0.000013
10.0	1	5	0	0.000002
10.0	1	6	29	0.003153
10.0	2	3	21	0.000851
10.0	2	4	13	0.000168
10.0	2	5	18	0.000483
10.0	2	6	47	0.029575
10.0	3	4	66	0.243201
10.0	3	5	91	0.621513
10.0	3	6	41	0.015312
10.0	4	5	64	0.132727
10.0	4	6	24	0.001432
10.0	5	6	19	0.000586
50.0	-1	0	0	1.000000
50.0	-1	1	5	0.625000
50.0	-1	2	4	0.437500
50.0	-1	3	0	0.062500
50.0	-1	4	0	0.062500
50.0	-1	5	0	0.062500
50.0	-1	6	0	0.062500
50.0	0	1	5	0.625000
50.0	0	2	4	0.437500
50.0	0	3	0	0.062500

Percent	Cut 1	Cut 2	statistic	p-value
50.0	0	4	0	0.062500
50.0	0	5	0	0.062500
50.0	0	6	0	0.062500
50.0	1	2	4	0.437500
50.0	1	3	0	0.062500
50.0	1	4	0	0.062500
50.0	1	5	0	0.062500
50.0	1	6	0	0.062500
50.0	2	3	0	0.062500
50.0	2	4	0	0.062500
50.0	2	5	0	0.062500
50.0	2	6	5	0.625000
50.0	3	4	1	0.125000
50.0	3	5	3	0.312500
50.0	3	6	0	0.062500
50.0	4	5	6	0.812500
50.0	4	6	0	0.062500
50.0	5	6	0	0.062500
100.0	-1	0	0	1.000000
100.0	-1	1	6	0.812500
100.0	-1	2	0	0.062500
100.0	-1	3	0	0.062500
100.0	-1	4	1	0.125000
100.0	-1	5	0	0.062500
100.0	-1	6	0	0.062500
100.0	0	1	6	0.812500
100.0	0	2	0	0.062500
100.0	0	3	0	0.062500
100.0	0	4	1	0.125000
100.0	0	5	0	0.062500
100.0	0	6	0	0.062500
100.0	1	2	3	0.312500
100.0	1	3	0	0.062500
100.0	1	4	3	0.312500
100.0	1	5	0	0.062500
100.0	1	6	0	0.062500
100.0	2	3	0	0.062500
100.0	2	4	5	0.625000
100.0	2	5	1	0.125000
100.0	2	6	0	0.062500
100.0	3	4	1	0.125000
100.0	3	5	0	0.062500
100.0	3	6	0	0.062500
100.0	4	5	3	0.312500
100.0	4	6	0	0.062500
100.0	5	6	0	0.062500

Table A.2: The reported p-values correspond to pairwise comparisons conducted using the Wilcoxon signed-rank test, specifically comparing each pair of cuts within individual percentages, as a result of the empirical experiments from all other classes (pretraining) to Sandal, Sneaker, Ankle boot classes (fine-tuning); in reinit=True setting. The rows marked in green show the cuts which are significantly different from each other. See Section 3.3.3 for more details.

Percent	Cut 1	Cut 2	statistic	p-value
0.1	-1	0	118	0.241066
0.1	-1	1	79	0.023650
0.1	-1	2	22	0.000032
0.1	-1	3	5	0.000001
0.1	-1	4	18	0.000015
0.1	-1	5	35	0.000250
0.1	-1	6	25	0.000054
0.1	0	1	87	0.042150
0.1	0	2	47	0.001155
0.1	0	3	22	0.000032
0.1	0	4	79	0.023650
0.1	0	5	79	0.023650
0.1	0	6	106	0.133640
0.1	1	2	66	0.008069
0.1	1	3	59	0.004175
0.1	1	4	120	0.263476
0.1	1	5	124	0.312333
0.1	1	6	150	0.750993
0.1	2	3	123	0.299612
0.1	2	4	123	0.299612
0.1	2	5	128	0.366585
0.1	2	6	94	0.066702
0.1	3	4	89	0.048262
0.1	3	5	100	0.095733
0.1	3	6	54	0.002508
0.1	4	5	155	0.853200
0.1	4	6	117	0.230361
0.1	5	6	104	0.126648
1.0	-1	0	65	0.007371
1.0	-1	1	31	0.000140
1.0	-1	2	1	0.000000
1.0	-1	3	32	0.000162
1.0	-1	4	18	0.000015
1.0	-1	5	4	0.000000
1.0	-1	6	0	0.000000
1.0	0	1	74	0.015973
1.0	0	2	103	0.113491
1.0	0	3	130	0.395711
1.0	0	4	143	0.615043
1.0	0	5	161	0.978915
1.0	0	6	151	0.771159
1.0	1	2	135	0.474165
1.0	1	3	148	0.730989
1.0	1	4	136	0.490786
1.0	1	5	131	0.410765
1.0	1	6	124	0.325392
1.0	2	3	146	0.672075

Percent	Cut 1	Cut 2	statistic	p-value
1.0	2	4	140	0.560171
1.0	2	5	82	0.029578
1.0	2	6	86	0.039339
1.0	3	4	127	0.520296
1.0	3	5	144	0.633827
1.0	3	6	123	0.299612
1.0	4	5	137	0.507704
1.0	4	6	128	0.366585
1.0	5	6	161	0.978915
10.0	-1	0	32	0.004860
10.0	-1	1	37	0.009436
10.0	-1	2	13	0.000168
10.0	-1	3	25	0.001690
10.0	-1	4	34	0.006390
10.0	-1	5	1	0.000004
10.0	-1	6	7	0.000036
10.0	0	1	98	0.812355
10.0	0	2	97	0.784126
10.0	0	3	65	0.142906
10.0	0	4	53	0.053169
10.0	0	5	50	0.039989
10.0	0	6	53	0.053169
10.0	1	2	86	0.498009
10.0	1	3	100	0.869488
10.0	1	4	81	0.388376
10.0	1	5	85	0.474905
10.0	1	6	75	0.277355
10.0	2	3	102	0.927279
10.0	2	4	79	0.348810
10.0	2	5	91	0.621513
10.0	2	6	83	0.430433
10.0	3	4	54	0.058258
10.0	3	5	87	0.545876
10.0	3	6	65	0.142906
10.0	4	5	89	0.570597
10.0	4	6	94	0.701181
10.0	5	6	88	0.545876
50.0	-1	0	0	0.062500
50.0	-1	1	0	0.062500
50.0	-1	2	1	0.125000
50.0	-1	3	0	0.062500
50.0	-1	4	0	0.062500
50.0	-1	5	0	0.062500
50.0	-1	6	0	0.062500
50.0	0	1	6	0.812500
50.0	0	2	5	0.625000
50.0	0	3	7	1.000000

Percent	Cut 1	Cut 2	statistic	p-value
50.0	0	4	0	0.062500
50.0	0	5	0	0.062500
50.0	0	6	0	0.062500
50.0	1	2	4	0.437500
50.0	1	3	7	1.000000
50.0	1	4	0	0.062500
50.0	1	5	0	0.062500
50.0	1	6	0	0.062500
50.0	2	3	5	0.625000
50.0	2	4	5	0.625000
50.0	2	5	5	0.625000
50.0	2	6	5	0.625000
50.0	3	4	0	0.062500
50.0	3	5	0	0.062500
50.0	3	6	0	0.062500
50.0	4	5	0	0.062500
50.0	4	6	0	0.062500
50.0	5	6	0	0.062500
100.0	-1	0	0	0.062500
100.0	-1	1	0	0.062500
100.0	-1	2	0	0.062500
100.0	-1	3	0	0.062500
100.0	-1	4	0	0.062500
100.0	-1	5	3	0.312500
100.0	-1	6	0	0.062500
100.0	0	1	3	0.312500
100.0	0	2	4	0.437500
100.0	0	3	3	0.312500
100.0	0	4	6	0.812500
100.0	0	5	0	0.062500
100.0	0	6	0	0.062500
100.0	1	2	4	0.437500
100.0	1	3	0	0.062500
100.0	1	4	6	0.812500
100.0	1	5	1	0.125000
100.0	1	6	0	0.062500
100.0	2	3	4	0.712702
100.0	2	4	4	0.437500
100.0	2	5	0	0.062500
100.0	2	6	0	0.062500
100.0	3	4	4	0.437500
100.0	3	5	1	0.125000
100.0	3	6	0	0.062500
100.0	4	5	0	0.062500
100.0	4	6	0	0.062500
100.0	5	6	0	0.062500

Table A.3: The reported p-values correspond to pairwise comparisons conducted using the Wilcoxon signed-rank test, specifically comparing each pair of cuts within individual percentages, as a result of the experiments with regular classes in *reinit=False* setting. The rows marked in green show the cuts which are significantly different from each other. See Section 3.3.3 for more details.

Percent	Cut	stat.	p-value	higher
0.1	-1	0	1.000000	None
0.1	0	118	0.241066	None
0.1	1	92	0.058752	None
0.1	2	100	0.095733	None
0.1	3	97	0.080196	None
0.1	4	73	0.014722	FR
0.1	5	104	0.119934	None
0.1	6	0	1.000000	None
1.0	-1	0	1.000000	None
1.0	0	65	0.007371	F
1.0	1	28	0.000088	F
1.0	2	36	0.000287	F
1.0	3	112	0.283952	None
1.0	4	73	0.014722	FR
1.0	5	69	0.010511	FR
1.0	6	0	1.000000	None
10.0	-1	0	1.000000	None
10.0	0	32	0.004860	F
10.0	1	42	0.017181	F
10.0	2	55	0.063723	None
10.0	3	77	0.468845	None
10.0	4	41	0.015312	FR
10.0	5	41	0.015312	FR
10.0	6	0	1.000000	None
50.0	-1	0	1.000000	None
50.0	0	0	0.062500	None
50.0	1	0	0.062500	None
50.0	2	4	0.437500	None
50.0	3	7	1.000000	None
50.0	4	4	0.437500	None
50.0	5	2	0.187500	None
50.0	6	0	1.000000	None
100.0	-1	0	1.000000	None
100.0	0	0	0.062500	None
100.0	1	0	0.062500	None
100.0	2	0	0.062500	None
100.0	3	7	1.000000	None
100.0	4	4	0.437500	None
100.0	5	1	0.125000	None
100.0	6	0	1.000000	None

Table A.4: The reported p-values correspond to pairwise comparisons conducted using the Wilcoxon signed-rank test, specifically comparing each cut in each percentage between 2 fine-tuning experiments, first one in *FR* settings, and second one in *F* settings (see 3.2.1 for details). The rows marked in green show the cuts which are significantly different between two experiments. The "Higher" column shows which experiment results are significantly higher compared to the other, for the specific percentage and cut point. If the value of "Higher" is None, the difference between the experiments for that percentage and cut point is not significant. See Section 3.3.3 for more details.

Percent	Cut 1	Cut 2	statistic	p-value
0.1	-1	0	117	0.230361
0.1	-1	1	58	0.003781
0.1	-1	2	49	0.001453
0.1	-1	3	136	0.490786
0.1	-1	4	59	0.004175
0.1	-1	5	20	0.000022
0.1	-1	6	25	0.000054
0.1	0	1	89	0.048262
0.1	0	2	55	0.002785
0.1	0	3	154	0.832509
0.1	0	4	59	0.004175
0.1	0	5	32	0.000162
0.1	0	6	35	0.000250
0.1	1	2	99	0.090316
0.1	1	3	143	0.615043
0.1	1	4	69	0.010511
0.1	1	5	72	0.013555
0.1	1	6	116	0.219986
0.1	2	3	110	0.164496
0.1	2	4	118	0.241066
0.1	2	5	143	0.615043
0.1	2	6	158	0.915803
0.1	3	4	103	0.113491
0.1	3	5	83	0.031808
0.1	3	6	112	0.181730
0.1	4	5	159	0.936803
0.1	4	6	91	0.055070
0.1	5	6	110	0.164496
1.0	-1	0	6	0.000001
1.0	-1	1	6	0.000001
1.0	-1	2	25	0.000054
1.0	-1	3	106	0.133640
1.0	-1	4	79	0.023650
1.0	-1	5	16	0.000010
1.0	-1	6	0	0.000000
1.0	0	1	92	0.058752
1.0	0	2	51	0.001816
1.0	0	3	154	0.832509
1.0	0	4	126	0.338788
1.0	0	5	62	0.005579
1.0	0	6	31	0.000140
1.0	1	2	95	0.070984
1.0	1	3	147	0.691519
1.0	1	4	142	0.596500
1.0	1	5	92	0.058752
1.0	1	6	120	0.263476
1.0	2	3	62	0.005579

Percent	Cut 1	Cut 2	statistic	p-value
1.0	2	4	104	0.119934
1.0	2	5	147	0.691519
1.0	2	6	94	0.066702
1.0	3	4	146	0.672075
1.0	3	5	89	0.048262
1.0	3	6	129	0.380984
1.0	4	5	107	0.219233
1.0	4	6	158	0.915803
1.0	5	6	105	0.126648
10.0	-1	0	8	0.000048
10.0	-1	1	3	0.000010
10.0	-1	2	20	0.000708
10.0	-1	3	53	0.053169
10.0	-1	4	48	0.032768
10.0	-1	5	11	0.000105
10.0	-1	6	7	0.000036
10.0	0	1	88	0.545876
10.0	0	2	29	0.003153
10.0	0	3	100	0.869488
10.0	0	4	92	0.647655
10.0	0	5	35	0.007296
10.0	0	6	76	0.294252
10.0	1	2	84	0.452375
10.0	1	3	93	0.674223
10.0	1	4	79	0.348810
10.0	1	5	73	0.245487
10.0	1	6	98	0.812355
10.0	2	3	56	0.069580
10.0	2	4	39	0.012079
10.0	2	5	100	0.869488
10.0	2	6	54	0.058258
10.0	3	4	79	0.348810
10.0	3	5	44	0.021484
10.0	3	6	87	0.521673
10.0	4	5	42	0.017181
10.0	4	6	82	0.409098
10.0	5	6	39	0.012079
50.0	-1	0	2	0.187500
50.0	-1	1	0	0.062500
50.0	-1	2	0	0.062500
50.0	-1	3	1	0.125000
50.0	-1	4	0	0.062500
50.0	-1	5	0	0.062500
50.0	-1	6	0	0.062500
50.0	0	1	0	0.062500
50.0	0	2	0	0.062500
50.0	0	3	4	0.437500

Percent	Cut 1	Cut 2	statistic	p-value
50.0	0	4	1	0.125000
50.0	0	5	0	0.062500
50.0	0	6	1	0.125000
50.0	1	2	0	0.062500
50.0	1	3	0	0.062500
50.0	1	4	0	0.062500
50.0	1	5	7	1.000000
50.0	1	6	0	0.062500
50.0	2	3	0	0.062500
50.0	2	4	2	0.187500
50.0	2	5	1	0.125000
50.0	2	6	0	0.062500
50.0	3	4	7	1.000000
50.0	3	5	0	0.062500
50.0	3	6	6	0.812500
50.0	4	5	0	0.062500
50.0	4	6	3	0.312500
50.0	5	6	0	0.062500
100.0	-1	0	0	0.062500
100.0	-1	1	0	0.062500
100.0	-1	2	2	0.187500
100.0	-1	3	5	0.625000
100.0	-1	4	6	0.812500
100.0	-1	5	0	0.062500
100.0	-1	6	0	0.062500
100.0	0	1	0	0.062500
100.0	0	2	0	0.062500
100.0	0	3	0	0.062500
100.0	0	4	0	0.062500
100.0	0	5	0	0.062500
100.0	0	6	0	0.062500
100.0	1	2	1	0.125000
100.0	1	3	0	0.062500
100.0	1	4	0	0.062500
100.0	1	5	4	0.437500
100.0	1	6	0	0.062500
100.0	2	3	0	0.062500
100.0	2	4	0	0.062500
100.0	2	5	2	0.187500
100.0	2	6	0	0.062500
100.0	3	4	4	0.437500
100.0	3	5	0	0.062500
100.0	3	6	0	0.062500
100.0	4	5	1	0.125000
100.0	4	6	0	0.062500
100.0	5	6	0	0.062500

Table A.5: The reported p-values correspond to pairwise comparisons conducted using the Wilcoxon signed-rank test, specifically comparing each pair of cuts within individual percentages, as a result of the experiments with regular classes in *FT* setting. The rows marked in green show the cuts which are significantly different from each other. See Section 3.3.3 for more details.

Percent	Cut	stat.	p-value	higher
0.1	-1	0	1.000000	None
0.1	0	117	0.230361	None
0.1	1	74	0.015973	FT
0.1	2	146	0.672075	None
0.1	3	84	0.034174	FR
0.1	4	162	1.000000	None
0.1	5	121	0.275184	None
0.1	6	0	1.000000	None
1.0	-1	0	1.000000	None
1.0	0	6	0.000001	FT
1.0	1	1	0.000000	FT
1.0	2	71	0.012466	FT
1.0	3	93	0.062626	None
1.0	4	66	0.008069	FR
1.0	5	137	0.507704	None
1.0	6	0	1.000000	None
10.0	-1	0	1.000000	None
10.0	0	8	0.000048	FT
10.0	1	39	0.012079	FT
10.0	2	23	0.001209	FT
10.0	3	44	0.021484	FR
10.0	4	21	0.000851	FR
10.0	5	97	0.784126	None
10.0	6	0	1.000000	None
50.0	-1	0	1.000000	None
50.0	0	2	0.187500	None
50.0	1	0	0.062500	None
50.0	2	0	0.062500	None
50.0	3	0	0.062500	None
50.0	4	0	0.062500	None
50.0	5	3	0.312500	None
50.0	6	0	1.000000	None
100.0	-1	0	1.000000	None
100.0	0	0	0.062500	None
100.0	1	1	0.125000	None
100.0	2	6	0.812500	None
100.0	3	0	0.062500	None
100.0	4	0	0.062500	None
100.0	5	5	0.625000	None
100.0	6	0	1.000000	None

Table A.6: The reported p-values correspond to pairwise comparisons conducted using the Wilcoxon signed-rank test, specifically comparing each cut in each percentage between 2 fine-tuning experiments, first one in *truncate=False*, and second one in *truncate=True* setting (see Section 3.2.1 for details). The rows marked in green show the cuts which are significantly different between two experiments. The "Higher" column shows which experiment results are significantly higher compared to the other, for the specific percentage and cut point. If the value of "Higher" is None, the difference between the experiments for that percentage and cut point is not significant. See Section 3.3.3 for more details.

Percent	Layer	Layer	stat.	p-value
0.1	1	2	32	0.035157
0.1	1	3	132	0.867133
0.1	1	4	108	0.235687
0.1	1	5	136	0.689126
0.1	1	6	117	0.345729
0.1	2	3	34	0.024907
0.1	2	4	65	0.026399
0.1	2	5	87	0.501591
0.1	2	6	88	0.076489
0.1	3	4	38	0.211476
0.1	3	5	59	0.407434
0.1	3	6	117	0.757760
0.1	4	5	32	0.062671
0.1	4	6	88	0.525653
0.1	5	6	54	0.469113
1.0	1	2	59	0.004175
1.0	1	3	131	0.410765
1.0	1	4	20	0.000022
1.0	1	5	147	0.691519
1.0	1	6	35	0.000250
1.0	2	3	55	0.002785
1.0	2	4	0	0.000000
1.0	2	5	82	0.029578
1.0	2	6	4	0.000000
1.0	3	4	60	0.017675
1.0	3	5	110	0.164496
1.0	3	6	61	0.005072
1.0	4	5	31	0.000140
1.0	4	6	153	0.811930
1.0	5	6	26	0.000064
10.0	1	2	26	0.001986
10.0	1	3	96	0.756166
10.0	1	4	44	0.021484
10.0	1	5	98	0.812355
10.0	1	6	54	0.058258
10.0	2	3	26	0.001986
10.0	2	4	0	0.000002
10.0	2	5	28	0.002712
10.0	2	6	1	0.000004
10.0	3	4	34	0.006390
10.0	3	5	99	0.840822
10.0	3	6	35	0.007296
10.0	4	5	66	0.153646
10.0	4	6	99	0.840822
10.0	5	6	73	0.245487
50.0	1	2	4	0.437500
50.0	1	3	6	0.812500
50.0	1	4	7	1.000000
50.0	1	5	3	0.312500
50.0	1	6	3	0.312500
50.0	2	3	0	0.062500
50.0	2	4	6	0.812500
50.0	2	5	0	0.062500
50.0	2	6	1	0.125000
50.0	3	4	5	0.625000
50.0	3	5	1	0.125000
50.0	3	6	5	0.625000
50.0	4	5	1	0.125000
50.0	4	6	4	0.437500
50.0	5	6	6	0.812500
100.0	1	2	1	0.125000
100.0	1	3	1	0.125000
100.0	1	4	2	0.187500
100.0	1	5	1	0.125000
100.0	1	6	0	0.062500
100.0	2	3	0	0.062500
100.0	2	4	0	0.062500
100.0	2	5	0	0.062500
100.0	2	6	0	0.062500
100.0	3	4	7	1.000000
100.0	3	5	1	0.125000
100.0	3	6	4	0.437500
100.0	4	5	6	0.812500
100.0	4	6	6	0.812500
100.0	5	6	7	1.000000

Table A.7: The reported p-values correspond to pairwise comparisons conducted using the Wilcoxon signed-rank test, specifically comparing each pair of cuts within individual percentages, as a result of the clustering experiment in FashionMNIST regular classes. Note that, Layer 0 indicates the clustering performance with the features extracted from the first convolutional layer (measured in pairwise positioning recall), Layer 1 indicates the clustering performance with the features extracted from the second convolutional layer and so on. For more details on the clustering experiment, see Section 3.2.2. The rows marked in green show the cuts (features extracted from different layers) which are significantly different from each other. See Section 3.3.3 for more details.

Percent	Cut 1	Cut 2	statistic	p-value
0.1	-1	0	0	1.000000
0.1	-1	1	127	0.352520
0.1	-1	2	11	0.000003
0.1	-1	3	1	0.000000
0.1	-1	4	4	0.000000
0.1	-1	5	23	0.000038
0.1	-1	6	25	0.000054
0.1	0	1	127	0.352520
0.1	0	2	11	0.000003
0.1	0	3	1	0.000000
0.1	0	4	4	0.000000
0.1	0	5	23	0.000038
0.1	0	6	25	0.000054
0.1	1	2	11	0.000003
0.1	1	3	26	0.000064
0.1	1	4	16	0.000010
0.1	1	5	30	0.000120
0.1	1	6	40	0.000489
0.1	2	3	71	0.012466
0.1	2	4	102	0.107315
0.1	2	5	158	0.915803
0.1	2	6	109	0.156338
0.1	3	4	157	0.894860
0.1	3	5	80	0.027479
0.1	3	6	97	0.080196
0.1	4	5	90	0.051576
0.1	4	6	129	0.380984
0.1	5	6	129	0.380984
1.0	-1	0	0	1.000000
1.0	-1	1	159	0.936803
1.0	-1	2	77	0.020275
1.0	-1	3	66	0.008069
1.0	-1	4	27	0.000075
1.0	-1	5	20	0.000022
1.0	-1	6	62	0.005579
1.0	0	1	159	0.936803
1.0	0	2	77	0.020275
1.0	0	3	66	0.008069
1.0	0	4	27	0.000075
1.0	0	5	20	0.000022
1.0	0	6	62	0.005579
1.0	1	2	91	0.055070
1.0	1	3	41	0.000556
1.0	1	4	9	0.000002
1.0	1	5	52	0.002025
1.0	1	6	62	0.005579
1.0	2	3	118	0.241066

Percent	Cut 1	Cut 2	statistic	p-value
1.0	2	4	32	0.000162
1.0	2	5	81	0.027479
1.0	2	6	100	0.095733
1.0	3	4	37	0.001244
1.0	3	5	129	0.380984
1.0	3	6	158	0.915803
1.0	4	5	128	0.366585
1.0	4	6	61	0.005072
1.0	5	6	133	0.441837
10.0	-1	0	0	1.000000
10.0	-1	1	70	0.202450
10.0	-1	2	23	0.001209
10.0	-1	3	20	0.000708
10.0	-1	4	0	0.000002
10.0	-1	5	12	0.000134
10.0	-1	6	19	0.000586
10.0	0	1	70	0.202450
10.0	0	2	23	0.001209
10.0	0	3	20	0.000708
10.0	0	4	0	0.000002
10.0	0	5	12	0.000134
10.0	0	6	19	0.000586
10.0	1	2	51	0.044054
10.0	1	3	28	0.002712
10.0	1	4	0	0.000002
10.0	1	5	7	0.000036
10.0	1	6	32	0.004860
10.0	2	3	55	0.063723
10.0	2	4	4	0.000013
10.0	2	5	67	0.164957
10.0	2	6	85	0.474905
10.0	3	4	30	0.003654
10.0	3	5	73	0.245487
10.0	3	6	65	0.153646
10.0	4	5	13	0.000168
10.0	4	6	5	0.000019
10.0	5	6	73	0.261099
50.0	-1	0	0	1.000000
50.0	-1	1	4	0.437500
50.0	-1	2	3	0.312500
50.0	-1	3	1	0.125000
50.0	-1	4	3	0.312500
50.0	-1	5	7	1.000000
50.0	-1	6	3	0.312500
50.0	0	1	4	0.437500
50.0	0	2	3	0.312500
50.0	0	3	1	0.125000

Percent	Cut 1	Cut 2	statistic	p-value
50.0	0	4	3	0.312500
50.0	0	5	7	1.000000
50.0	0	6	3	0.312500
50.0	1	2	6	0.812500
50.0	1	3	5	0.625000
50.0	1	4	7	1.000000
50.0	1	5	7	1.000000
50.0	1	6	0	0.062500
50.0	2	3	2	0.273322
50.0	2	4	6	0.812500
50.0	2	5	6	0.812500
50.0	2	6	0	0.062500
50.0	3	4	2	0.187500
50.0	3	5	4	0.625000
50.0	3	6	0	0.062500
50.0	4	5	7	1.000000
50.0	4	6	1	0.125000
50.0	5	6	0	0.062500
100.0	-1	0	0	1.000000
100.0	-1	1	6	0.812500
100.0	-1	2	4	0.437500
100.0	-1	3	6	0.812500
100.0	-1	4	3	0.312500
100.0	-1	5	0	0.062500
100.0	-1	6	0	0.062500
100.0	0	1	6	0.812500
100.0	0	2	4	0.437500
100.0	0	3	6	0.812500
100.0	0	4	3	0.312500
100.0	0	5	0	0.062500
100.0	0	6	0	0.062500
100.0	1	2	7	1.000000
100.0	1	3	5	0.625000
100.0	1	4	3	0.312500
100.0	1	5	0	0.062500
100.0	1	6	0	0.062500
100.0	2	3	3	0.312500
100.0	2	4	1	0.125000
100.0	2	5	0	0.062500
100.0	2	6	0	0.062500
100.0	3	4	3	0.312500
100.0	3	5	1	0.125000
100.0	3	6	0	0.062500
100.0	4	5	5	0.625000
100.0	4	6	2	0.273322
100.0	5	6	1	0.125000

Table A.8: The reported p-values correspond to pairwise comparisons conducted using the Wilcoxon signed-rank test, specifically comparing each pair of cuts within individual percentages, as a result of the empirical experiments from random classes of FashionMNIST (pre-training) to the rest of the classes of FashionMNIST (fine-tuning). The rows marked in green show the cuts which are significantly different from each other. See Section 3.3.3 for more details.

Percent	Cut 1	Cut 2	statistic	p-value
0.1	-1	0	0	1.000000
0.1	-1	1	124	0.312333
0.1	-1	2	111	0.172959
0.1	-1	3	150	0.750993
0.1	-1	4	64	0.006726
0.1	-1	5	51	0.001816
0.1	-1	6	161	0.978915
0.1	0	1	124	0.312333
0.1	0	2	111	0.172959
0.1	0	3	150	0.750993
0.1	0	4	64	0.006726
0.1	0	5	51	0.001816
0.1	0	6	161	0.978915
0.1	1	2	77	0.020275
0.1	1	3	160	0.957845
0.1	1	4	92	0.058752
0.1	1	5	63	0.006129
0.1	1	6	128	0.366585
0.1	2	3	63	0.006129
0.1	2	4	10	0.000003
0.1	2	5	3	0.000000
0.1	2	6	117	0.230361
0.1	3	4	67	0.008822
0.1	3	5	42	0.000631
0.1	3	6	129	0.380984
0.1	4	5	110	0.164496
0.1	4	6	53	0.002255
0.1	5	6	14	0.000007
1.0	-1	0	0	1.000000
1.0	-1	1	114	0.200216
1.0	-1	2	139	0.542404
1.0	-1	3	102	0.107315
1.0	-1	4	1	0.000000
1.0	-1	5	0	0.000000
1.0	-1	6	74	0.015973
1.0	0	1	114	0.200216
1.0	0	2	139	0.542404
1.0	0	3	102	0.107315
1.0	0	4	1	0.000000
1.0	0	5	0	0.000000
1.0	0	6	74	0.015973
1.0	1	2	108	0.148480
1.0	1	3	154	0.832509
1.0	1	4	0	0.000000
1.0	1	5	0	0.000000
1.0	1	6	140	0.560171
1.0	2	3	135	0.474165

Percent	Cut 1	Cut 2	statistic	p-value
1.0	2	4	0	0.000000
1.0	2	5	0	0.000000
1.0	2	6	51	0.001816
1.0	3	4	6	0.000001
1.0	3	5	0	0.000000
1.0	3	6	156	0.873988
1.0	4	5	112	0.277605
1.0	4	6	0	0.000000
1.0	5	6	0	0.000000
10.0	-1	0	0	1.000000
10.0	-1	1	93	0.674223
10.0	-1	2	94	0.701181
10.0	-1	3	1	0.000004
10.0	-1	4	3	0.000010
10.0	-1	5	0	0.000002
10.0	-1	6	76	0.294252
10.0	0	1	93	0.674223
10.0	0	2	94	0.701181
10.0	0	3	1	0.000004
10.0	0	4	3	0.000010
10.0	0	5	0	0.000002
10.0	0	6	76	0.294252
10.0	1	2	73	0.245487
10.0	1	3	10	0.000082
10.0	1	4	0	0.000002
10.0	1	5	0	0.000002
10.0	1	6	88	0.545876
10.0	2	3	22	0.001017
10.0	2	4	0	0.000002
10.0	2	5	0	0.000002
10.0	2	6	69	0.189348
10.0	3	4	11	0.000105
10.0	3	5	0	0.000002
10.0	3	6	11	0.000105
10.0	4	5	49	0.036234
10.0	4	6	0	0.000002
10.0	5	6	0	0.000002
50.0	-1	0	0	1.000000
50.0	-1	1	4	0.437500
50.0	-1	2	7	1.000000
50.0	-1	3	7	1.000000
50.0	-1	4	6	0.812500
50.0	-1	5	0	0.062500
50.0	-1	6	0	0.062500
50.0	0	1	4	0.437500
50.0	0	2	7	1.000000
50.0	0	3	7	1.000000

Table A.9: The reported p-values correspond to pairwise comparisons conducted using the Wilcoxon signed-rank test, specifically comparing each pair of cuts within individual percentages, as a result of the empirical experiments from the whole MNIST dataset (pretraining) to the whole FashionMNIST dataset (finetuning). The rows marked in green show the cuts which are significantly different from each other. See Section 3.3.3 for more details.

Percent	Cut 1	Cut 2	statistic	p-value
50.0	0	4	6	0.812500
50.0	0	5	0	0.062500
50.0	0	6	0	0.062500
50.0	1	2	5	0.625000
50.0	1	3	1	0.125000
50.0	1	4	4	0.437500
50.0	1	5	1	0.125000
50.0	1	6	0	0.062500
50.0	2	3	5	0.625000
50.0	2	4	7	1.000000
50.0	2	5	0	0.062500
50.0	2	6	0	0.062500
50.0	3	4	7	1.000000
50.0	3	5	0	0.062500
50.0	3	6	0	0.062500
50.0	4	5	0	0.062500
50.0	4	6	0	0.062500
50.0	5	6	0	0.062500
100.0	-1	0	0	1.000000
100.0	-1	1	2	0.187500
100.0	-1	2	3	0.312500
100.0	-1	3	1	0.125000
100.0	-1	4	0	0.062500
100.0	-1	5	0	0.062500
100.0	-1	6	0	0.062500
100.0	0	1	2	0.187500
100.0	0	2	3	0.312500
100.0	0	3	1	0.125000
100.0	0	4	0	0.062500
100.0	0	5	0	0.062500
100.0	0	6	0	0.062500
100.0	1	2	0	0.062500
100.0	1	3	0	0.062500
100.0	1	4	0	0.062500
100.0	1	5	0	0.062500
100.0	1	6	0	0.062500
100.0	2	3	2	0.187500
100.0	2	4	0	0.062500
100.0	2	5	0	0.062500
100.0	2	6	0	0.062500
100.0	3	4	7	1.000000
100.0	3	5	1	0.125000
100.0	3	6	0	0.062500
100.0	4	5	0	0.062500
100.0	4	6	0	0.062500
100.0	5	6	0	0.062500

Percent	Cut 1	Cut 2	statistic	p-value
0.1	-1	0	0	1.000000
0.1	-1	1	128	0.366585
0.1	-1	2	161	0.978915
0.1	-1	3	126	0.338788
0.1	-1	4	6	0.000001
0.1	-1	5	0	0.000000
0.1	-1	6	137	0.507704
0.1	0	1	128	0.366585
0.1	0	2	161	0.978915
0.1	0	3	126	0.338788
0.1	0	4	6	0.000001
0.1	0	5	0	0.000000
0.1	0	6	137	0.507704
0.1	1	2	140	0.560171
0.1	1	3	101	0.101397
0.1	1	4	0	0.000000
0.1	1	5	0	0.000000
0.1	1	6	133	0.441837
0.1	2	3	120	0.263476
0.1	2	4	17	0.000012
0.1	2	5	1	0.000000
0.1	2	6	157	0.894860
0.1	3	4	51	0.001816
0.1	3	5	18	0.000015
0.1	3	6	124	0.312333
0.1	4	5	11	0.000003
0.1	4	6	2	0.000000
0.1	5	6	0	0.000000
1.0	-1	0	0	1.000000
1.0	-1	1	148	0.711161
1.0	-1	2	120	0.263476
1.0	-1	3	161	0.978915
1.0	-1	4	38	0.000376
1.0	-1	5	1	0.000000
1.0	-1	6	161	0.978915
1.0	0	1	148	0.711161
1.0	0	2	120	0.263476
1.0	0	3	161	0.978915
1.0	0	4	38	0.000376
1.0	0	5	1	0.000000
1.0	0	6	161	0.978915
1.0	1	2	140	0.560171
1.0	1	3	153	0.811930
1.0	1	4	68	0.009635
1.0	1	5	0	0.000000
1.0	1	6	161	0.978915
1.0	2	3	162	1.000000

Percent	Cut 1	Cut 2	statistic	p-value
1.0	2	4	82	0.029578
1.0	2	5	30	0.000120
1.0	2	6	132	0.426142
1.0	3	4	76	0.018745
1.0	3	5	52	0.002025
1.0	3	6	129	0.380984
1.0	4	5	38	0.000376
1.0	4	6	23	0.000038
1.0	5	6	0	0.000000
10.0	-1	0	0	1.000000
10.0	-1	1	55	0.063723
10.0	-1	2	30	0.003654
10.0	-1	3	59	0.089695
10.0	-1	4	14	0.000210
10.0	-1	5	12	0.000134
10.0	-1	6	0	0.000002
10.0	0	1	55	0.063723
10.0	0	2	30	0.003654
10.0	0	3	59	0.089695
10.0	0	4	14	0.000210
10.0	0	5	12	0.000134
10.0	0	6	0	0.000002
10.0	1	2	28	0.002712
10.0	1	3	33	0.005581
10.0	1	4	64	0.132727
10.0	1	5	88	0.545876
10.0	1	6	0	0.000002
10.0	2	3	71	0.216167
10.0	2	4	0	0.000002
10.0	2	5	3	0.000010
10.0	2	6	0	0.000002
10.0	3	4	2	0.000006
10.0	3	5	8	0.000048
10.0	3	6	0	0.000002
10.0	4	5	71	0.216167
10.0	4	6	0	0.000002
10.0	5	6	0	0.000002
50.0	-1	0	0	1.000000
50.0	-1	1	4	0.437500
50.0	-1	2	1	0.125000
50.0	-1	3	2	0.187500
50.0	-1	4	0	0.062500
50.0	-1	5	0	0.062500
50.0	-1	6	0	0.062500
50.0	0	1	4	0.437500
50.0	0	2	1	0.125000
50.0	0	3	2	0.187500

Table A.10: The reported p-values correspond to pairwise comparisons conducted using the Wilcoxon signed-rank test, specifically comparing each pair of cuts within individual percentages, as a result of the empirical experiments from the whole FashionMNIST dataset (pretraining) to the whole MNIST dataset (fine-tuning). The rows marked in green show the cuts which are significantly different from each other. See Section 3.3.3 for more details.

Percent	Cut 1	Cut 2	statistic	p-value
50.0	0	4	0	0.062500
50.0	0	5	0	0.062500
50.0	0	6	0	0.062500
50.0	1	2	1	0.125000
50.0	1	3	0	0.062500
50.0	1	4	0	0.062500
50.0	1	5	0	0.062500
50.0	1	6	0	0.062500
50.0	2	3	5	0.625000
50.0	2	4	0	0.062500
50.0	2	5	0	0.062500
50.0	2	6	0	0.062500
50.0	3	4	0	0.062500
50.0	3	5	0	0.062500
50.0	3	6	0	0.062500
50.0	4	5	2	0.187500
50.0	4	6	0	0.062500
50.0	5	6	0	0.062500
100.0	-1	0	0	1.000000
100.0	-1	1	5	0.625000
100.0	-1	2	0	0.062500
100.0	-1	3	0	0.062500
100.0	-1	4	0	0.062500
100.0	-1	5	0	0.062500
100.0	-1	6	0	0.062500
100.0	0	1	5	0.625000
100.0	0	2	0	0.062500
100.0	0	3	0	0.062500
100.0	0	4	0	0.062500
100.0	0	5	0	0.062500
100.0	0	6	0	0.062500
100.0	1	2	0	0.062500
100.0	1	3	0	0.062500
100.0	1	4	0	0.062500
100.0	1	5	0	0.062500
100.0	1	6	0	0.062500
100.0	2	3	1	0.125000
100.0	2	4	0	0.062500
100.0	2	5	0	0.062500
100.0	2	6	0	0.062500
100.0	3	4	0	0.062500
100.0	3	5	0	0.062500
100.0	3	6	0	0.062500
100.0	4	5	1	0.125000
100.0	4	6	0	0.062500
100.0	5	6	0	0.062500

Percent	Cut 1	Cut 2	statistic	p-value
0.1	-1	0	0	1.000000
0.1	-1	1	59	0.004175
0.1	-1	2	48	0.001296
0.1	-1	3	39	0.000430
0.1	-1	4	1	0.000000
0.1	-1	5	0	0.000000
0.1	-1	6	5	0.000001
0.1	0	1	59	0.004175
0.1	0	2	48	0.001296
0.1	0	3	39	0.000430
0.1	0	4	1	0.000000
0.1	0	5	0	0.000000
0.1	0	6	5	0.000001
0.1	1	2	145	0.652841
0.1	1	3	103	0.113491
0.1	1	4	0	0.000000
0.1	1	5	3	0.000000
0.1	1	6	27	0.000075
0.1	2	3	111	0.172959
0.1	2	4	4	0.000000
0.1	2	5	8	0.000001
0.1	2	6	40	0.001673
0.1	3	4	1	0.000000
0.1	3	5	2	0.000000
0.1	3	6	79	0.023650
0.1	4	5	131	0.410765
0.1	4	6	25	0.000054
0.1	5	6	7	0.000001
1.0	-1	0	0	1.000000
1.0	-1	1	80	0.025505
1.0	-1	2	0	0.000000
1.0	-1	3	0	0.000000
1.0	-1	4	0	0.000000
1.0	-1	5	0	0.000000
1.0	-1	6	0	0.000000
1.0	0	1	80	0.025505
1.0	0	2	0	0.000000
1.0	0	3	0	0.000000
1.0	0	4	0	0.000000
1.0	0	5	0	0.000000
1.0	0	6	0	0.000000
1.0	1	2	3	0.000000
1.0	1	3	0	0.000000
1.0	1	4	0	0.000000
1.0	1	5	0	0.000000
1.0	1	6	0	0.000000
1.0	2	3	8	0.000001

Percent	Cut 1	Cut 2	statistic	p-value
1.0	2	4	0	0.000000
1.0	2	5	0	0.000000
1.0	2	6	0	0.000000
1.0	3	4	0	0.000000
1.0	3	5	0	0.000000
1.0	3	6	0	0.000000
1.0	4	5	23	0.000038
1.0	4	6	4	0.000000
1.0	5	6	11	0.000071
10.0	-1	0	0	1.000000
10.0	-1	1	3	0.000010
10.0	-1	2	0	0.000002
10.0	-1	3	0	0.000002
10.0	-1	4	0	0.000002
10.0	-1	5	0	0.000002
10.0	-1	6	0	0.000002
10.0	0	1	3	0.000010
10.0	0	2	0	0.000002
10.0	0	3	0	0.000002
10.0	0	4	0	0.000002
10.0	0	5	0	0.000002
10.0	0	6	0	0.000002
10.0	1	2	0	0.000002
10.0	1	3	0	0.000002
10.0	1	4	0	0.000002
10.0	1	5	0	0.000002
10.0	1	6	1	0.000004
10.0	2	3	13	0.001038
10.0	2	4	5	0.000019
10.0	2	5	65	0.142906
10.0	2	6	0	0.000002
10.0	3	4	81	0.388376
10.0	3	5	0	0.000002
10.0	3	6	0	0.000002
10.0	4	5	0	0.000002
10.0	4	6	0	0.000002
10.0	5	6	0	0.000002
50.0	-1	0	0	1.000000
50.0	-1	1	0	0.062500
50.0	-1	2	0	0.062500
50.0	-1	3	0	0.062500
50.0	-1	4	4	0.437500
50.0	-1	5	0	0.062500
50.0	-1	6	0	0.062500
50.0	0	1	0	0.062500
50.0	0	2	0	0.062500
50.0	0	3	0	0.062500

Table A.11: The reported p-values correspond to pairwise comparisons conducted using the Wilcoxon signed-rank test, specifically comparing each pair of cuts within individual percentages, as a result of the empirical experiments from Vehicles classes (pretraining) to Animal classes (fine-tuning) of CIFAR. The rows marked in green show the cuts which are significantly different from each other. See Section 3.3.3 for more details.

Percent	Cut 1	Cut 2	statistic	p-value
50.0	0	4	4	0.437500
50.0	0	5	0	0.062500
50.0	0	6	0	0.062500
50.0	1	2	0	0.062500
50.0	1	3	7	1.000000
50.0	1	4	0	0.062500
50.0	1	5	0	0.062500
50.0	1	6	0	0.062500
50.0	2	3	0	0.062500
50.0	2	4	0	0.062500
50.0	2	5	0	0.062500
50.0	2	6	0	0.062500
50.0	3	4	0	0.062500
50.0	3	5	0	0.062500
50.0	3	6	0	0.062500
50.0	4	5	0	0.062500
50.0	4	6	0	0.062500
50.0	5	6	0	0.062500
100.0	-1	0	0	1.000000
100.0	-1	1	1	0.125000
100.0	-1	2	5	0.625000
100.0	-1	3	0	0.062500
100.0	-1	4	0	0.062500
100.0	-1	5	0	0.062500
100.0	-1	6	0	0.062500
100.0	0	1	1	0.125000
100.0	0	2	5	0.625000
100.0	0	3	0	0.062500
100.0	0	4	0	0.062500
100.0	0	5	0	0.062500
100.0	0	6	0	0.062500
100.0	1	2	2	0.187500
100.0	1	3	0	0.062500
100.0	1	4	0	0.062500
100.0	1	5	0	0.062500
100.0	1	6	0	0.062500
100.0	2	3	0	0.062500
100.0	2	4	0	0.062500
100.0	2	5	0	0.062500
100.0	2	6	0	0.062500
100.0	3	4	0	0.062500
100.0	3	5	0	0.062500
100.0	3	6	0	0.062500
100.0	4	5	0	0.062500
100.0	4	6	0	0.062500
100.0	5	6	0	0.062500

Percent	Cut 1	Cut 2	statistic	p-value
0.1	-1	0	104	0.119934
0.1	-1	1	32	0.000162
0.1	-1	2	3	0.000000
0.1	-1	3	2	0.000000
0.1	-1	4	1	0.000000
0.1	-1	5	3	0.000000
0.1	-1	6	2	0.000000
0.1	0	1	61	0.005072
0.1	0	2	1	0.000000
0.1	0	3	1	0.000000
0.1	0	4	0	0.000000
0.1	0	5	5	0.000001
0.1	0	6	0	0.000000
0.1	1	2	4	0.000000
0.1	1	3	4	0.000000
0.1	1	4	1	0.000000
0.1	1	5	8	0.000001
0.1	1	6	17	0.000012
0.1	2	3	42	0.000631
0.1	2	4	11	0.000003
0.1	2	5	113	0.190814
0.1	2	6	124	0.325392
0.1	3	4	44	0.000808
0.1	3	5	125	0.325392
0.1	3	6	52	0.002025
0.1	4	5	9	0.000002
0.1	4	6	26	0.000064
0.1	5	6	92	0.058752
1.0	-1	0	44	0.000808
1.0	-1	1	0	0.000000
1.0	-1	2	0	0.000000
1.0	-1	3	0	0.000000
1.0	-1	4	0	0.000000
1.0	-1	5	0	0.000000
1.0	-1	6	0	0.000000
1.0	0	1	0	0.000000
1.0	0	2	0	0.000000
1.0	0	3	0	0.000000
1.0	0	4	0	0.000000
1.0	0	5	0	0.000000
1.0	0	6	0	0.000000
1.0	1	2	0	0.000000
1.0	1	3	0	0.000000
1.0	1	4	0	0.000000
1.0	1	5	0	0.000000
1.0	1	6	0	0.000000
1.0	2	3	0	0.000000

Percent	Cut 1	Cut 2	statistic	p-value
1.0	2	4	0	0.000000
1.0	2	5	34	0.000217
1.0	2	6	73	0.014722
1.0	3	4	1	0.000000
1.0	3	5	107	0.140915
1.0	3	6	6	0.000001
1.0	4	5	0	0.000000
1.0	4	6	0	0.000000
1.0	5	6	10	0.000003
10.0	-1	0	0	0.000002
10.0	-1	1	0	0.000002
10.0	-1	2	0	0.000002
10.0	-1	3	0	0.000002
10.0	-1	4	0	0.000002
10.0	-1	5	0	0.000002
10.0	-1	6	0	0.000002
10.0	0	1	0	0.000002
10.0	0	2	0	0.000002
10.0	0	3	0	0.000002
10.0	0	4	0	0.000002
10.0	0	5	0	0.000002
10.0	0	6	0	0.000002
10.0	1	2	0	0.000002
10.0	1	3	0	0.000002
10.0	1	4	0	0.000002
10.0	1	5	0	0.000002
10.0	1	6	0	0.000002
10.0	2	3	88	0.778155
10.0	2	4	74	0.261099
10.0	2	5	7	0.000036
10.0	2	6	0	0.000002
10.0	3	4	65	0.142906
10.0	3	5	31	0.004221
10.0	3	6	0	0.000002
10.0	4	5	4	0.000013
10.0	4	6	0	0.000002
10.0	5	6	0	0.000002
50.0	-1	0	0	0.062500
50.0	-1	1	3	0.312500
50.0	-1	2	1	0.125000
50.0	-1	3	0	0.062500
50.0	-1	4	5	0.625000
50.0	-1	5	2	0.187500
50.0	-1	6	0	0.062500
50.0	0	1	0	0.062500
50.0	0	2	0	0.062500
50.0	0	3	0	0.062500

Table A.12: The reported p-values correspond to pairwise comparisons conducted using the Wilcoxon signed-rank test, specifically comparing each pair of cuts within individual percentages, as a result of the truncation experiments from Vehicles classes (pretraining) to Animal classes (fine-tuning) of CIFAR. The rows marked in green show the cuts which are significantly different from each other. See Section 3.3.3 for more details.

Percent	Cut 1	Cut 2	statistic	p-value
50.0	0	4	0	0.062500
50.0	0	5	0	0.062500
50.0	0	6	0	0.062500
50.0	1	2	3	0.312500
50.0	1	3	0	0.062500
50.0	1	4	2	0.187500
50.0	1	5	0	0.062500
50.0	1	6	0	0.062500
50.0	2	3	1	0.125000
50.0	2	4	0	0.062500
50.0	2	5	0	0.062500
50.0	2	6	0	0.062500
50.0	3	4	0	0.062500
50.0	3	5	0	0.062500
50.0	3	6	0	0.062500
50.0	4	5	0	0.062500
50.0	4	6	0	0.062500
50.0	5	6	0	0.062500
100.0	-1	0	0	0.062500
100.0	-1	1	3	0.312500
100.0	-1	2	3	0.312500
100.0	-1	3	0	0.062500
100.0	-1	4	0	0.062500
100.0	-1	5	0	0.062500
100.0	-1	6	0	0.062500
100.0	0	1	0	0.062500
100.0	0	2	0	0.062500
100.0	0	3	0	0.062500
100.0	0	4	0	0.062500
100.0	0	5	0	0.062500
100.0	0	6	0	0.062500
100.0	1	2	1	0.125000
100.0	1	3	1	0.125000
100.0	1	4	7	1.000000
100.0	1	5	7	1.000000
100.0	1	6	1	0.125000
100.0	2	3	0	0.062500
100.0	2	4	0	0.062500
100.0	2	5	0	0.062500
100.0	2	6	0	0.062500
100.0	3	4	0	0.062500
100.0	3	5	0	0.062500
100.0	3	6	0	0.062500
100.0	4	5	6	0.812500
100.0	4	6	0	0.062500
100.0	5	6	0	0.062500

Percent	Cut	stat.	p-value	higher
0.1	-1	0	1.000000	None
0.1	0	104	0.119934	None
0.1	1	157	0.894860	None
0.1	2	25	0.000054	FT
0.1	3	18	0.000015	FT
0.1	4	152	0.791476	None
0.1	5	46	0.001027	FR
0.1	6	72	0.013555	FT
1.0	-1	0	1.000000	None
1.0	0	44	0.000808	FT
1.0	1	0	0.000000	FT
1.0	2	0	0.000000	FT
1.0	3	0	0.000000	FT
1.0	4	34	0.000217	FT
1.0	5	113	0.190814	None
1.0	6	68	0.009635	FT
10.0	-1	0	1.000000	None
10.0	0	0	0.000002	FR
10.0	1	31	0.004221	FT
10.0	2	8	0.000048	FT
10.0	3	104	0.985435	None
10.0	4	86	0.498009	None
10.0	5	10	0.000082	FT
10.0	6	0	0.000002	FT
50.0	-1	0	1.000000	None
50.0	0	0	0.062500	None
50.0	1	5	0.625000	None
50.0	2	6	0.812500	None
50.0	3	0	0.062500	None
50.0	4	7	1.000000	None
50.0	5	0	0.062500	None
50.0	6	0	0.062500	None
100.0	-1	0	1.000000	None
100.0	0	0	0.062500	None
100.0	1	1	0.125000	None
100.0	2	5	0.625000	None
100.0	3	0	0.062500	None
100.0	4	5	0.625000	None
100.0	5	0	0.062500	None
100.0	6	0	0.062500	None

Table A.13: The reported p-values correspond to pairwise comparisons conducted using the Wilcoxon signed-rank test, specifically comparing each cut in each percentage between 2 experiments on CIFAR, fine-tuning with *FR* vs *FT* settings. The rows marked in green show the cuts which are significantly different between two experiments. The "Higher" column shows which experiment results are significantly higher compared to the other, for the specific percentage and cut point. If the value of "Higher" is None, the difference between the experiments for that percentage and cut point is not significant. See Section 3.3.3 for more details.

Percent	Cut 1	Cut 2	stat.	p-value
0.1	-1	0	0	1.000000
0.1	-1	1	147	0.691519
0.1	-1	2	81	0.027479
0.1	-1	3	2	0.000000
0.1	-1	4	0	0.000000
0.1	-1	5	0	0.000000
0.1	-1	6	1	0.000000
0.1	0	1	147	0.691519
0.1	0	2	81	0.027479
0.1	0	3	2	0.000000
0.1	0	4	0	0.000000
0.1	0	5	0	0.000000
0.1	0	6	1	0.000000
0.1	1	2	87	0.042150
0.1	1	3	15	0.000008
0.1	1	4	0	0.000000
0.1	1	5	0	0.000000
0.1	1	6	1	0.000000
0.1	2	3	60	0.004605
0.1	2	4	0	0.000000
0.1	2	5	0	0.000000
0.1	2	6	29	0.000103
0.1	3	4	11	0.000003
0.1	3	5	9	0.000002
0.1	3	6	82	0.029578
0.1	4	5	55	0.002785
0.1	4	6	46	0.001027
0.1	5	6	19	0.000018
1.0	-1	0	0	1.000000
1.0	-1	1	129	0.380984
1.0	-1	2	5	0.000001
1.0	-1	3	0	0.000000
1.0	-1	4	0	0.000000
1.0	-1	5	0	0.000000
1.0	-1	6	0	0.000000
1.0	0	1	129	0.380984
1.0	0	2	5	0.000001
1.0	0	3	0	0.000000
1.0	0	4	0	0.000000
1.0	0	5	0	0.000000
1.0	0	6	0	0.000000
1.0	1	2	21	0.000027
1.0	1	3	0	0.000000
1.0	1	4	0	0.000000
1.0	1	5	0	0.000000
1.0	1	6	0	0.000000
1.0	2	3	10	0.000003

Percent	Cut 1	Cut 2	stat.	p-value
1.0	2	4	0	0.000000
1.0	2	5	0	0.000000
1.0	2	6	0	0.000000
1.0	3	4	0	0.000000
1.0	3	5	0	0.000000
1.0	3	6	0	0.000000
1.0	4	5	0	0.000000
1.0	4	6	121	0.275184
1.0	5	6	0	0.000000
10.0	-1	0	0	1.000000
10.0	-1	1	0	0.000002
10.0	-1	2	0	0.000002
10.0	-1	3	0	0.000002
10.0	-1	4	0	0.000002
10.0	-1	5	0	0.000002
10.0	-1	6	0	0.000002
10.0	0	1	0	0.000002
10.0	0	2	0	0.000002
10.0	0	3	0	0.000002
10.0	0	4	0	0.000002
10.0	0	5	0	0.000002
10.0	0	6	0	0.000002
10.0	1	2	1	0.000004
10.0	1	3	0	0.000002
10.0	1	4	0	0.000002
10.0	1	5	0	0.000002
10.0	1	6	0	0.000002
10.0	2	3	13	0.000168
10.0	2	4	6	0.000027
10.0	2	5	31	0.004221
10.0	2	6	20	0.000708
10.0	3	4	44	0.021484
10.0	3	5	83	0.430433
10.0	3	6	0	0.000002
10.0	4	5	33	0.005581
10.0	4	6	0	0.000002
10.0	5	6	0	0.000002
50.0	-1	0	0	1.000000
50.0	-1	1	2	0.187500
50.0	-1	2	1	0.125000
50.0	-1	3	0	0.062500
50.0	-1	4	1	0.125000
50.0	-1	5	0	0.062500
50.0	-1	6	0	0.062500
50.0	0	1	2	0.187500
50.0	0	2	1	0.125000
50.0	0	3	0	0.062500

Percent	Cut 1	Cut 2	stat.	p-value
50.0	0	4	1	0.125000
50.0	0	5	0	0.062500
50.0	0	6	0	0.062500
50.0	1	2	1	0.125000
50.0	1	3	4	0.437500
50.0	1	4	5	0.625000
50.0	1	5	0	0.062500
50.0	1	6	0	0.062500
50.0	2	3	4	0.437500
50.0	2	4	1	0.125000
50.0	2	5	0	0.062500
50.0	2	6	0	0.062500
50.0	3	4	2	0.187500
50.0	3	5	0	0.062500
50.0	3	6	0	0.062500
50.0	4	5	0	0.062500
50.0	4	6	0	0.062500
50.0	5	6	0	0.062500
100.0	-1	0	0	1.000000
100.0	-1	1	7	1.000000
100.0	-1	2	6	0.812500
100.0	-1	3	3	0.312500
100.0	-1	4	5	0.625000
100.0	-1	5	0	0.062500
100.0	-1	6	0	0.062500
100.0	0	1	7	1.000000
100.0	0	2	6	0.812500
100.0	0	3	3	0.312500
100.0	0	4	5	0.625000
100.0	0	5	0	0.062500
100.0	0	6	0	0.062500
100.0	1	2	7	1.000000
100.0	1	3	2	0.187500
100.0	1	4	7	1.000000
100.0	1	5	0	0.062500
100.0	1	6	0	0.062500
100.0	2	3	4	0.437500
100.0	2	4	2	0.187500
100.0	2	5	0	0.062500
100.0	2	6	0	0.062500
100.0	3	4	0	0.062500
100.0	3	5	0	0.062500
100.0	3	6	0	0.062500
100.0	4	5	0	0.062500
100.0	4	6	0	0.062500
100.0	5	6	0	0.062500

Table A.14: The reported p-values correspond to pairwise comparisons conducted using the Wilcoxon signed-rank test, specifically comparing each pair of cuts within individual percentages, as a result of the empirical experiments from the Animals classes (pretraining) to Vehicles classes (fine-tuning) of CIFAR. The rows marked in green show the cuts which are significantly different from each other. See Section 3.3.3 for more details.

Percent	Cut 1	Cut 2	stat.	p-value
0.1	-1	0	0	1.000000
0.1	-1	1	98	0.085139
0.1	-1	2	19	0.000018
0.1	-1	3	9	0.000002
0.1	-1	4	0	0.000000
0.1	-1	5	0	0.000000
0.1	-1	6	0	0.000000
0.1	0	1	98	0.085139
0.1	0	2	19	0.000018
0.1	0	3	9	0.000002
0.1	0	4	0	0.000000
0.1	0	5	0	0.000000
0.1	0	6	0	0.000000
0.1	1	2	18	0.000015
0.1	1	3	12	0.000004
0.1	1	4	0	0.000000
0.1	1	5	0	0.000000
0.1	1	6	0	0.000000
0.1	2	3	88	0.045123
0.1	2	4	0	0.000000
0.1	2	5	0	0.000000
0.1	2	6	1	0.000000
0.1	3	4	0	0.000000
0.1	3	5	0	0.000000
0.1	3	6	26	0.000064
0.1	4	5	90	0.051576
0.1	4	6	99	0.090316
0.1	5	6	51	0.001816
1.0	-1	0	0	1.000000
1.0	-1	1	18	0.000015
1.0	-1	2	0	0.000000
1.0	-1	3	0	0.000000
1.0	-1	4	0	0.000000
1.0	-1	5	0	0.000000
1.0	-1	6	0	0.000000
1.0	0	1	18	0.000015
1.0	0	2	0	0.000000
1.0	0	3	0	0.000000
1.0	0	4	0	0.000000
1.0	0	5	0	0.000000
1.0	0	6	0	0.000000
1.0	1	2	0	0.000000
1.0	1	3	0	0.000000
1.0	1	4	0	0.000000
1.0	1	5	0	0.000000
1.0	1	6	0	0.000000
1.0	2	3	40	0.000489

Percent	Cut 1	Cut 2	stat.	p-value
1.0	2	4	0	0.000000
1.0	2	5	0	0.000000
1.0	2	6	0	0.000000
1.0	3	4	0	0.000000
1.0	3	5	0	0.000000
1.0	3	6	0	0.000000
1.0	4	5	0	0.000000
1.0	4	6	5	0.000001
1.0	5	6	131	0.587190
10.0	-1	0	0	1.000000
10.0	-1	1	1	0.000155
10.0	-1	2	0	0.000002
10.0	-1	3	0	0.000002
10.0	-1	4	0	0.000002
10.0	-1	5	0	0.000002
10.0	-1	6	0	0.000002
10.0	0	1	1	0.000155
10.0	0	2	0	0.000002
10.0	0	3	0	0.000002
10.0	0	4	0	0.000002
10.0	0	5	0	0.000002
10.0	0	6	0	0.000002
10.0	1	2	19	0.000586
10.0	1	3	3	0.000010
10.0	1	4	0	0.000002
10.0	1	5	0	0.000002
10.0	1	6	0	0.000002
10.0	2	3	8	0.000048
10.0	2	4	0	0.000002
10.0	2	5	0	0.000002
10.0	2	6	1	0.000004
10.0	3	4	32	0.019780
10.0	3	5	36	0.008308
10.0	3	6	93	0.674223
10.0	4	5	96	0.756166
10.0	4	6	34	0.006390
10.0	5	6	5	0.000019
50.0	-1	0	0	1.000000
50.0	-1	1	0	0.062500
50.0	-1	2	0	0.062500
50.0	-1	3	0	0.062500
50.0	-1	4	0	0.062500
50.0	-1	5	0	0.062500
50.0	-1	6	1	0.125000
50.0	0	1	0	0.062500
50.0	0	2	0	0.062500
50.0	0	3	0	0.062500

Percent	Cut 1	Cut 2	stat.	p-value
50.0	0	4	0	0.062500
50.0	0	5	0	0.062500
50.0	0	6	1	0.125000
50.0	1	2	0	0.062500
50.0	1	3	0	0.062500
50.0	1	4	6	0.812500
50.0	1	5	0	0.062500
50.0	1	6	0	0.062500
50.0	2	3	4	0.437500
50.0	2	4	0	0.062500
50.0	2	5	0	0.062500
50.0	2	6	0	0.062500
50.0	3	4	1	0.125000
50.0	3	5	0	0.062500
50.0	3	6	0	0.062500
50.0	4	5	0	0.062500
50.0	4	6	0	0.062500
50.0	5	6	0	0.062500
100.0	-1	0	0	1.000000
100.0	-1	1	0	0.062500
100.0	-1	2	0	0.062500
100.0	-1	3	0	0.062500
100.0	-1	4	6	0.812500
100.0	-1	5	0	0.062500
100.0	-1	6	0	0.062500
100.0	0	1	0	0.062500
100.0	0	2	0	0.062500
100.0	0	3	0	0.062500
100.0	0	4	6	0.812500
100.0	0	5	0	0.062500
100.0	0	6	0	0.062500
100.0	1	2	5	0.625000
100.0	1	3	1	0.125000
100.0	1	4	0	0.062500
100.0	1	5	0	0.062500
100.0	1	6	0	0.062500
100.0	2	3	0	0.062500
100.0	2	4	0	0.062500
100.0	2	5	0	0.062500
100.0	2	6	0	0.062500
100.0	3	4	1	0.125000
100.0	3	5	0	0.062500
100.0	3	6	0	0.062500
100.0	4	5	0	0.062500
100.0	4	6	0	0.062500
100.0	5	6	0	0.062500

Table A.15: The reported p-values correspond to pairwise comparisons conducted using the Wilcoxon signed-rank test, specifically comparing each pair of cuts within individual percentages, as a result of the empirical experiments from the classes dog, frog, horse, ship, truck (pretraining) to classes airplane, automobile, bird, cat (fine-tuning) of CIFAR. The rows marked in green show the cuts which are significantly different from each other. See Section 3.3.3 for more details.

Percent	Layer	Layer	stat.	p-value
0.1	-1	0	0	1.000000
0.1	-1	1	98	0.085139
0.1	-1	2	19	0.000018
0.1	-1	3	9	0.000002
0.1	-1	4	0	0.000000
0.1	-1	5	0	0.000000
0.1	-1	6	0	0.000000
0.1	0	1	98	0.085139
0.1	0	2	19	0.000018
0.1	0	3	9	0.000002
0.1	0	4	0	0.000000
0.1	0	5	0	0.000000
0.1	0	6	0	0.000000
0.1	1	2	18	0.000015
0.1	1	3	12	0.000004
0.1	1	4	0	0.000000
0.1	1	5	0	0.000000
0.1	1	6	0	0.000000
0.1	2	3	88	0.045123
0.1	2	4	0	0.000000
0.1	2	5	0	0.000000
0.1	2	6	1	0.000000
0.1	3	4	0	0.000000
0.1	3	5	0	0.000000
0.1	3	6	26	0.000064
0.1	4	5	90	0.051576
0.1	4	6	99	0.090316
0.1	5	6	51	0.001816
1.0	-1	0	0	1.000000
1.0	-1	1	18	0.000015
1.0	-1	2	0	0.000000
1.0	-1	3	0	0.000000
1.0	-1	4	0	0.000000
1.0	-1	5	0	0.000000
1.0	-1	6	0	0.000000
1.0	0	1	18	0.000015
1.0	0	2	0	0.000000
1.0	0	3	0	0.000000
1.0	0	4	0	0.000000
1.0	0	5	0	0.000000
1.0	0	6	0	0.000000
1.0	1	2	0	0.000000
1.0	1	3	0	0.000000
1.0	1	4	0	0.000000
1.0	1	5	0	0.000000
1.0	1	6	0	0.000000
1.0	2	3	40	0.000489

Percent	Layer	Layer	stat.	p-value
1.0	2	4	0	0.000000
1.0	2	5	0	0.000000
1.0	2	6	0	0.000000
1.0	3	4	0	0.000000
1.0	3	5	0	0.000000
1.0	3	6	0	0.000000
1.0	4	5	0	0.000000
1.0	4	6	5	0.000001
1.0	5	6	131	0.587190
10.0	-1	0	0	1.000000
10.0	-1	1	1	0.000155
10.0	-1	2	0	0.000002
10.0	-1	3	0	0.000002
10.0	-1	4	0	0.000002
10.0	-1	5	0	0.000002
10.0	-1	6	0	0.000002
10.0	0	1	1	0.000155
10.0	0	2	0	0.000002
10.0	0	3	0	0.000002
10.0	0	4	0	0.000002
10.0	0	5	0	0.000002
10.0	0	6	0	0.000002
10.0	1	2	19	0.000586
10.0	1	3	3	0.000010
10.0	1	4	0	0.000002
10.0	1	5	0	0.000002
10.0	1	6	0	0.000002
10.0	2	3	8	0.000048
10.0	2	4	0	0.000002
10.0	2	5	0	0.000002
10.0	2	6	1	0.000004
10.0	3	4	32	0.019780
10.0	3	5	36	0.008308
10.0	3	6	93	0.674223
10.0	4	5	96	0.756166
10.0	4	6	34	0.006390
10.0	5	6	5	0.000019
50.0	-1	0	0	1.000000
50.0	-1	1	0	0.062500
50.0	-1	2	0	0.062500
50.0	-1	3	0	0.062500
50.0	-1	4	0	0.062500
50.0	-1	5	0	0.062500
50.0	-1	6	1	0.125000
50.0	0	1	0	0.062500
50.0	0	2	0	0.062500
50.0	0	3	0	0.062500

Percent	Layer	Layer	stat.	p-value
50.0	0	4	0	0.062500
50.0	0	5	0	0.062500
50.0	0	6	1	0.125000
50.0	1	2	0	0.062500
50.0	1	3	0	0.062500
50.0	1	4	6	0.812500
50.0	1	5	0	0.062500
50.0	1	6	0	0.062500
50.0	2	3	4	0.437500
50.0	2	4	0	0.062500
50.0	2	5	0	0.062500
50.0	2	6	0	0.062500
50.0	3	4	1	0.125000
50.0	3	5	0	0.062500
50.0	3	6	0	0.062500
50.0	4	5	0	0.062500
50.0	4	6	0	0.062500
50.0	5	6	0	0.062500
100.0	-1	0	0	1.000000
100.0	-1	1	0	0.062500
100.0	-1	2	0	0.062500
100.0	-1	3	0	0.062500
100.0	-1	4	6	0.812500
100.0	-1	5	0	0.062500
100.0	-1	6	0	0.062500
100.0	0	1	0	0.062500
100.0	0	2	0	0.062500
100.0	0	3	0	0.062500
100.0	0	4	6	0.812500
100.0	0	5	0	0.062500
100.0	0	6	0	0.062500
100.0	1	2	5	0.625000
100.0	1	3	1	0.125000
100.0	1	4	0	0.062500
100.0	1	5	0	0.062500
100.0	1	6	0	0.062500
100.0	2	3	0	0.062500
100.0	2	4	0	0.062500
100.0	2	5	0	0.062500
100.0	2	6	0	0.062500
100.0	3	4	1	0.125000
100.0	3	5	0	0.062500
100.0	3	6	0	0.062500
100.0	4	5	0	0.062500
100.0	4	6	0	0.062500
100.0	5	6	0	0.062500

Table A.16: The reported p-values correspond to pairwise comparisons conducted using the Wilcoxon signed-rank test, specifically comparing each pair of cuts within individual percentages, as a result of the clustering experiment in CIFAR regular classes. Note that, Layer 0 indicates the clustering performance with the features extracted from the first convolutional layer (measured in pairwise positioning recall), Layer 1 indicates the clustering performance with the features extracted from the second convolutional layer and so on. For more details on the clustering experiment, see Section 3.2.2. The rows marked in green show the cuts (features extracted from different layers) which are significantly different from each other. See Section 3.3.3 for more details.

List of Figures

List of Tables

List of Listings

Bibliography

Bai, Y., Yang, E., Han, B., Yang, Y., Li, J., Mao, Y., Niu, G., and Liu, T. (2021). Understanding and improving early stopping for learning with noisy labels. *Advances in Neural Information Processing Systems*, 34:24392–24403.

Basha, S. S., Dubey, S. R., Pulabaigari, V., and Mukherjee, S. (2020). Impact of fully connected layers on performance of convolutional neural networks for image classification. *Neurocomputing*, 378:112–119.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Cao, Y., Chen, Z., Belkin, M., and Gu, Q. (2022). Benign overfitting in two-layer convolutional neural networks. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 25237–25250. Curran Associates, Inc.

Cun, Y. L., Boser, B., Denker, J. S., Howard, R. E., Habbard, W., Jackel, L. D., and Henderson, D. (1990). *Handwritten digit recognition with a back-propagation network*, page 396–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Das, D., Santosh, K. C., and Pal, U. (2020). Truncated inception net: Covid-19 outbreak screening using chest x-rays. *Physical and Engineering Sciences in Medicine*, 43(3):915–925.

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

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655. PMLR.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Hubert, L. J. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2:193–218.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr.

Kim, B. J., Choi, H., Jang, H., Lee, D., and Kim, S. W. (2023). How to use dropout correctly on residual networks with batch normalization. In *Uncertainty in Artificial Intelligence*, pages 1058–1067. PMLR.

Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization.

Kolesnikov, A., Dosovitskiy, A., Weissenborn, D., Heigold, G., Uszkoreit, J., Beyer, L., Minderer, M., Dehghani, M., Houlsby, N., Gelly,

S., Unterthiner, T., and Zhai, X. (2021). An image is worth 16x16 words: Transformers for image recognition at scale.

Krizhevsky, A. (2012). Learning multiple layers of features from tiny images. *University of Toronto*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, volume 25.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Liu, Z., Xu, Y., Xu, Y., Qian, Q., Li, H., Ji, X., Chan, A., and Jin, R. (2022). Improved fine-tuning by better leveraging pre-training data. *Advances in Neural Information Processing Systems*, 35:32568–32581.

Mehmood, F., Ahmad, S., and Whangbo, T. K. (2023). An efficient optimization technique for training deep neural networks. *Mathematics*, 11(6).

Montalbo, F. J. P. (2021). Truncating a densely connected convolutional neural network with partial layer freezing and feature fusion for diagnosing covid-19 from chest x-rays. *MethodsX*, 8:101408.

Müllner, D. (2011). Modern hierarchical, agglomerative clustering algorithms. *ArXiv*, abs/1109.2378.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA. Omnipress.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Salman, S. and Liu, X. (2019a). Overfitting measurement of convolutional neural networks using trained networks. *arXiv preprint arXiv:1901.06566*.

Salman, S. and Liu, X. (2019b). Overfitting mechanism and avoidance in deep neural networks. *CoRR*, abs/1901.06566.

Scabini, L. F. and Bruno, O. M. (2023). Structure and performance of fully connected neural networks: Emerging complex network properties. *Physica A: Statistical Mechanics and its Applications*, 615:128585.

Sermanet, P., Chintala, S., and LeCun, Y. (2012). Convolutional neural networks applied to house numbers digit classification. In *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*, pages 3288–3291. IEEE.

Shao, H. and Wang, S. (2023). Deep classification with linearity-enhanced logits to softmax function. *Entropy*, 25(5).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Wang, Z., Dai, Z., Poczos, B., and Carbonell, J. (2019). Characterizing and avoiding negative transfer. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11285–11294, Los Alamitos, CA, USA. IEEE Computer Society.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1:80–83.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747.

Xu, S., Li, X., Xie, C., Chen, H., Chen, C., and Song, Z. (2021). A high-precision implementation of the sigmoid activation function for computing-in-memory architecture. *Micromachines (Basel)*, 12(10):1183. PMID: 34683234, PMC: PMC8540118.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3320–3328, Cambridge, MA, USA. MIT Press.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer Vision – ECCV 2014*, pages 818–833, Cham. Springer International Publishing.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76.