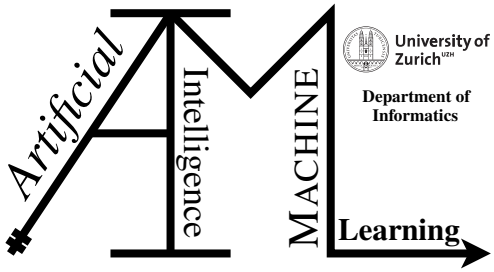
Transfer Learning in Small Image Databases

Master Thesis

Kevin Bohn
15-640-493

Submitted on
July 21 2022

Thesis Supervisor
Prof. Dr. Manuel Günther



Master Thesis

Author: Kevin Bohn, kevin.bohn@uzh.ch

Project period: 21.01.22 - 21.07.22

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

Acknowledgements

First and foremost, I would like to thank Prof. Dr. Manuel Günther for his helpful suggestions and critical examination of my topic during my work. Furthermore, I was honored to conduct my thesis in the Artificial Intelligence and Machine Learning (AIML) research group at the Department of Informatics, University of Zurich. Secondly, I would like to express my gratitude to my beloved wife for her patience during the last six months, as she kept me motivated and focused. Last but not least, I would like to acknowledge those who have critically reviewed my work and given me valuable suggestions.

Abstract

Existing research has addressed the effectiveness of transfer learning methods using as much available target data as possible. In contrast, in this thesis the classification performance trend of selected transfer learning techniques when 1 to 100 image representations per class are available during training is analyzed. Training followed by testing using classification accuracy was repeated 11 times with an increasing number of image samples per class. Thereby, the focus was on a few training samples per class. Transfer learning methods investigated include end-to-end classifications and variants adopting deep feature extraction, such as nearest neighbor classification or the subsequent application of a Support Vector Machine (SVM) classifier. Deep feature adaptations (no additional training, fine-tuning, and adapter network) were explored. The datasets Aircraft, Fruit and Vegetable, Indoor Scenes, Office-31, and Virus were examined, some closely related to ImageNet and others to a lesser extent. Moreover, AlexNet, ResNet-50, DenseNet-121, VGG-16, and MobileNet-V3 were included in the analysis, each pre-trained with ImageNet. The analysis revealed that ImageNet benchmarks could be used to select an appropriate pre-trained network for target datasets with overlapping ImageNet domains. Extracting deep features from the pre-trained network without training and enrolling a gallery comparing each class with an averaged feature representation to test images based on their cosine similarity outperformed fine-tuning approaches with up to 20 training images per class. With more than 20 images per class, fine-tuning approaches yield the highest performance. Feature extraction with subsequent usage of the SVM classifier provided the best performance of all methods examined, but only if more than 20 image samples per class were utilized. The advantage of the nearest neighbor classification compared to end-to-end classification became apparent. Furthermore, the amount of image data used to create the gallery was strongly related to the performance of the transfer learning method. When target datasets were dissimilar to ImageNet, superior performance was observed with a constant gallery including five image samples, which were not utilized during training. The results illustrated the dependence of the number of image samples per class and their relevance in selecting suitable transfer learning methods.

Zusammenfassung

Die bisherige Forschung hat sich mit der Effektivität von Transfer-Learning-Methoden mit möglichst vielen verfügbaren Zieldaten beschäftigt. Im Gegensatz dazu wird in dieser Arbeit die Entwicklung der Klassifizierungsperformance ausgewählter Transfer-Learning-Verfahren analysiert, wenn beim Training 1 bis 100 Bildrepräsentationen pro Klasse zur Verfügung stehen. Das Training mit anschließendem Testing mittels Classification Accuracy wurde 11 Mal mit steigender Anzahl von Bildproben pro Klasse wiederholt. Dabei lag der Schwerpunkt auf einigen wenigen Übungsproben pro Klasse. Zu den untersuchten Transfer-Learning-Methoden gehören End-to-End-Klassifikationen und Varianten mit Deep Feature Extraktion, wie z. B. Nearest Neighbour-Klassifikation oder die anschliessende Anwendung eines Support Vector Machine (SVM)-Klassifikators. Deep Feature Adaptionen (kein zusätzliches Training, Fine-Tuning und Adapternetzwerk) wurden erforscht. Die Datensätze Aircraft, Fruit and Vegetable, Indoor Scenes, Office-31 und Virus, von denen einige eng mit ImageNet verwandt sind und andere weniger, wurden untersucht. Außerdem wurden AlexNet, ResNet-50, DenseNet-121, VGG-16 und MobileNet-V3 in die Analyse einbezogen, die jeweils mit ImageNet vortrainiert wurden. Die Analyse ergab, dass ImageNet-Benchmarks verwendet werden können, um ein geeignetes vortrainiertes Netzwerk für Zieldatensätze mit überlappenden ImageNet-Domänen auszuwählen. Das Extrahieren von Deep Features aus dem vortrainierten Netzwerk ohne Training und das Erstellen einer Galerie, die jede Klasse mit einer gemittelten Deep Feature Repräsentation mit Testbildern auf der Grundlage ihrer Kosinusähnlichkeit vergleicht, übertraf das Fine-Tuning mit bis zu 20 Trainingsbildern pro Klasse. Mit mehr als 20 Bildern pro Klasse, erzeugen Fine-Tuning Ansätze die höchsten Leistungen. Feature Extraction mit anschließender Verwendung des SVM-Klassifikators lieferte die beste Leistung aller untersuchten Methoden, allerdings nur, wenn mehr als 20 Bildproben pro Klasse verwendet wurden. Der Vorteil der Nearest-Neighbour-Klassifizierung gegenüber der End-to-End-Klassifizierung wurde deutlich. Die Menge der Bilddaten, die zur Erstellung der Galerie verwendet wurden, stand in engem Zusammenhang mit der Leistung der Transfer-Learning-Methode. Wenn die Zieldatensätze dem ImageNet nicht ähnlich waren, wurde eine bessere Leistung mit einer konstanten Galerie beobachtet, die fünf Bildbeispiele enthielt, die beim Training nicht verwendet wurden. Die Ergebnisse verdeutlichen die Abhängigkeit von der Anzahl der Bildbeispiele pro Klasse und deren Relevanz für die Auswahl geeigneter Transfer-Learning-Methoden.

Contents

Chapter 1

Introduction

In recent years convolutional neural networks (CNNs) have achieved remarkable results in image classification tasks. A prime representative of this is AlexNet (Krizhevsky et al., 2012), whose implementation achieved the lowest error rate at the ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2012 (Russakovsky et al., 2015). Several adapted CNNs have been proposed since then and have proven to be useful in practice for various image classification scenarios. Selected networks, which are also used in this thesis, are DenseNet-121 (Huang et al., 2017), ResNet-50 (He et al., 2016), MobileNet-V3 (Howard et al., 2017) and VGG-16 (Simonyan and Zisserman, 2015). CNNs consist of stacked convolutional, pooling, and fully-connected layers. A convolutional layer performs matrix multiplications between input tensors and the learnable kernels resulting in activation maps. Afterward, the maps may be down-sampled using pooling layers. The convolutional and pooling layers output is fed into fully-connected layers and consequently used for classification (O'Shea and Nash, 2015). Typically, these deep networks are not designed to be trained with only a few image samples since millions of parameters must be optimized, leading to overfitting and, thus, a lower significance of classification results. Tan et al. (2018) argue that deep learning networks require large amounts of data to learn latent patterns. However, only small amounts of labeled data instances are often available, making CNN inferior to traditional machine learning methods, despite their preeminence in modern machine learning. Transfer learning attempts to circumvent this problem by training the network with a similar, widely accessible labeled database (source). Subsequently, the trained parameters of the network can be reused in a network specifically designed for classifying the limited available database (target). Thus, the target network can already resort to adapted weights, simplifying the learning process and allowing more complex networks to be deployed. After the transfer, a widely used method employs fine-tuning to account for the inherent data characteristics of the target network. Another approach extracts deep features from a specific layer in the pre-trained network. These are then used as a starting point for further network modifications or algorithms specifically designed to classify target image tasks. Regardless of how the transfer learning approach is carried out, the basic concept remains the same in all cases. The existing data knowledge from the source domain will be used to optimize the learning task in the target domain, as shown in figure 1.1. The goal is to improve the target decision function, which is in control of the class assignment of the input image, using the latent patterns in such a way that the classification performance is optimized. Several studies have already been published summarizing the most common transfer learning methods in the last few years. This fact indicates that a lot of attention is being paid to this area (Zhuang et al., 2021; Tan et al., 2018; Day and Khoshgoftaar, 2017; Weiss et al., 2016; Shao et al., 2015; Zhu et al., 2011; Pan and Yang, 2010).

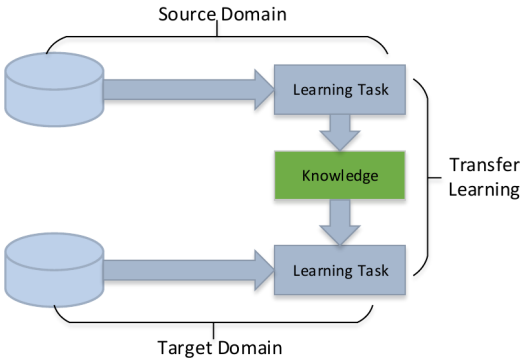


Figure 1.1: TRANSFER LEARNING PROCESS. This figure illustrates the learning process of transfer learning, where knowledge from a source domain is used to solve the learning task in the target domain (Tan et al., 2018).

1.1 Contribution

Although transfer learning approaches were developed and refined in recent work, it has not been investigated which transfer learning approach should be optimally applied when only very few training data of 1-3 image samples per class are available. Nor was it investigated whether the preference may change with the availability of more training data. Therefore, this thesis aims to investigate whether noticeable performance differences exist between the approaches examined. In each approach, one or more CNNs pre-trained with ImageNet are used in the experiments. The examined networks AlexNet, DenseNet-121, ResNet-50, MobileNet-V3, and VGG-16 are described in more detail in chapter 2.1. Since several pre-trained networks are utilized for transfer learning approaches in related work, an additional analysis compares the classification accuracy of the pre-trained networks for all transfer learning methods to investigate their applicability for transfer learning tasks. Experiments are repeated with five carefully selected datasets, namely Aircraft, Fruit and Vegetable, Indoor Scenes, Office-31, and Virus described in chapter 4. The variable similarity with ImageNet ensures that the statements made can be transferred to a certain extent to other image datasets with limited data availability, even if these have not been analyzed. Care is also taken to ensure that the datasets consist of at least 20 classes to make the tasks sufficiently challenging. The five datasets are divided into training, validation, and test splits. Dedicated training data of the target datasets are used for training. In addition, validation data is utilized during training to analyze training effectiveness and evaluate regularization metrics. These metrics are explained in section 5.3.2. Finally, the image classification performance is carried out with the test data. The accuracy performance of a transfer learning approach is always analyzed as it progresses when more and more training images are available per class. Thus, one approach is repeated with increasing training examples per class. This procedure makes it possible to compare the progress of the approaches based on the number of samples per class and thus make a statement about their performance. It is important to note that the analysis focuses on comparing transfer learning approaches with only a few training images. A brief description of the transfer learning approaches used is presented below.

Three different classification variants described more thoroughly in chapter 3.1 are distinguished. The first variant uses the softmax layer of the employed network for classification, which provides the probabilities for the respective occurrence of a class among all possible classes. The class with the highest probability finally defines the predicted class. This method is referred to hereafter as end-to-end classification. The last fully-connected layer and the softmax layer of the employed network will be neglected in the second variant. The network's output before the last fully-connected layer represents the deep features that optimally have the necessary inherent data characteristics of the target dataset to achieve accurate classification. When using the deep features, a gallery is enrolled in which features represent the respective classes. Two methods are distinguished from each other. The former uses all image samples available for training to create the gallery. The latter utilizes five image samples per class, which have not been used during training. The deep features are extracted using the already described method to determine to which class the test image samples belong. Nearest neighbor classification is applied to determine the smallest distance between the test features and the respective class features in the gallery. The cosine similarity distance is used as a measure. For the third method, the deep features are extracted from the network, similar to the second method. Instead of enrolling a gallery, the features are sequentially used as input for the Support Vector Machine (SVM) classifier with a linear kernel. With the additional input of the respective class affiliation, the classifier should be able to learn distinct decision boundaries for the respective classes. The boundaries can then be utilized to classify the test samples, which are also extracted from the network and consequentially fed into the learned classifier.

Additionally, three different network configurations thoroughly described in chapter 3.2 are examined. In the first variant, the network pre-trained with ImageNet is used without additional training. Thus, no end-to-end classification is part of the experiments. In the second variant, the pre-trained network is fine-tuned using the training data from the target dataset. Finally, an adapter network consisting of two or three fully-connected layers is used for the third variant. Analogous to nearest neighbor classification, the deep features may be extracted from the pre-trained network, but also the output of the last fully-connected layer (logits) can be utilized. Subsequently, the features or the logits are fed into the adapter network as input. Which number of adapter layers, which early stopping scheme, and whether deep features or logits are best suited as adapter network input in terms of accuracy performance will also be determined in a separate experiment. All classification variants according to chapter 3.1 can be examined with the fine-tuned and adapter network.

1.2 Research Questions

Based on the last section, the following research questions (RQ) are outlined, which will be discussed in the subsequent chapters:

- **RQ 1:** Given the adapter network approach, does an architecture with two or three fully-connected layers, the use of an early-stopping scheme during training, as well as the extraction of deep features or logits for the network input, lead to more accurate classification results?
- **RQ 2:** Are there accuracy differences between the examined CNNs pre-trained with ImageNet across all investigated target datasets and transfer learning approaches? Based on the accuracy performance, should a specific network possibly be prioritized for applying transfer learning for multi-class image classification?
- **RQ 3:** Which presented transfer learning approach achieves better classification results when only a few samples per class are available, and does this apply to all datasets examined?
- **RQ 4:** Can a specific number of image samples per class be identified for which the fine-tuned features achieve a superior classification performance than the original features?
- **RQ 5:** Are there differences in nearest neighbor classification when the gallery is enrolled using all available training samples or a constant number of image samples that are not considered during training?

1.3 Structure

This thesis is divided into several chapters, which are briefly described subsequently. In chapter 2 a concise definition of transfer learning is given, pre-trained networks are described, and previous related work in this area is outlined. Chapter 3 conceptually describes the transfer learning approaches. The datasets used for the experiments are described in detail in chapter 4. Chapter 5 presents the investigation procedure and the machine learning concepts used in training, followed by chapter 6, where the outcome of the experiments is outlined. Finally, in the chapters 7 and 8 the interpretation and final conclusions, including further aspects that would extend this work, are being discussed.

Chapter 2

Background & Related Work

In order to be able to grasp the concept of transfer learning better, this section discusses the exact definition, which is also reflected in some transfer learning surveys. Transfer learning approaches from related work are described subsequently. Zhuang et al. (2021); Tan et al. (2018); Weiss et al. (2016) define a domain $D = \{\Phi, P(X)\}$ which consists of the feature space Φ and a marginal distribution $P(X)$ where X is defined as following: $X = \{x|x^{[n]} \in X, n = 1, ..., N\}$. N refers to the number of available image samples and x the image instance. The marginal distribution is defined as a probability distribution that focuses on only one variable of multivariate data. Pan and Yang (2010) states that two domains are not equal if either Φ or $P(X)$ or both are different from each other. Furthermore, a task $T = \{Y, f(x)\}$ consists of a label space Y and a decision function $f(x)$ which is ideally learned during the training process. Analogous to domains, two tasks are not equal if either Y or $f(x)$ or both are different. There is a source domain D_s, which ideally provides widely accessible labeled data with many image samples N_s. The target domain D_t yields only a limited number of image instances N_t, which indicates that N_t is small. A network has already been trained according to the source task T_s. Knowledge obtained from the training is used to learn the target decision function $f_t(x)$ of the target task T_t. Finally, the transfer learning definition is formulated as follows:

Transfer Learning Definition: *Transfer learning tries to optimize the performance of the learned decision function $f_t(x)$ of the target domain learning task T_t by utilizing the learned latent knowledge from the source domain learning task T_s and source domain D_s where $T_s \neq T_t$ and/or $D_s \neq D_t$ and $N_s \gg N_t$* (Tan et al., 2018).

According to Zhuang et al. (2021), transfer learning can be further divided into more fine-grained subclasses. Considering the availability of labels, one could differentiate transfer learning approaches into transductive, inductive, and unsupervised. The first class includes scenarios where only source labels are available, whereas the second refers to those with additional availability in the target domain. The datasets of this thesis can also be assigned to this subdivision. The third approach groups together all approaches with no label information. Furthermore, a distinction is made between homogeneous and heterogeneous transfer learning. The former involves approaches where the source and target domains share the same feature space Φ. At the same time, the latter is more concerned with disjoint feature spaces, which introduces further complexity. Zhuang et al. (2021) outline further data-based and model-based interpretation subclasses which are not considered within the scope of this work. Merely methods concerning parameter and feature sharing are analyzed. In this process, pre-trained networks by the source dataset are adapted with the training dataset of the target. This can be accomplished in various ways as outlined in sections 2.2.1 and 2.2.2. Also worth mentioning is the distinction between domain adaptation and transfer learning. In domain adaptation tasks, the difference between the source and target

dataset is found in the domain, but the label spaces are ultimately the same. The goal is to reduce the domain shift between source and target. This is achieved, for example, by minimizing the maximum mean discrepancy and correlation distances or using the generation of adversarial samples (Tzeng et al., 2017). In transfer learning, the label spaces may differ, so minimizing the domain shift with the specified methods is not indicated.

2.1 Pre-trained networks

This section briefly describes the CNNs used in this thesis. The networks are available for the experiments already pre-trained with ImageNet. The image input size for CNNs should be provided with mini-batches of 3-channel RGB images of shape (3 x 224 x 224). The softmax layer maps the outputs of the last fully-connected layer, called logits, to probability values. The number of outputs is based on the available classes in the dataset. A summary of all networks can be found in table 2.1. The top-1 accuracy scores were taken from the PyTorch website.[1]

Table 2.1: CNN CHARACTERISTICS. All networks were pre-trained with ImageNet. In case convolutional layers form a unit, only the respective block is counted as a layer (e.g., MobileNet-V3). Top-1-accuracy scores are taken from the official PyTorch website.[1]

	Conv. Layers / Blocks	Parameters	Feature Size	Top-1 Accuracy
AlexNet	5	61.1 M	4'096	56.522
VGG-16	13	138.36 M	4'096	71.592
ResNet-50	50	25.56 M	2'048	76.130
MobileNet-V3	19	5.48 M	1'280	74.042
DenseNet-121	120	7.98 M	1'024	74.434

2.1.1 AlexNet

AlexNet was proposed by Krizhevsky et al. (2012) and achieved the lowest error rate at the ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2012 (Russakovsky et al., 2015). The network with over 60 million trainable parameters consists of five successive convolutional layers with 11 x 11 filters in the first, 5 x 5 filters in the second, and 3 x 3 filters in the following layers. Each convolutional layer is followed by an intermediate normalization and maximum pooling layer up to the third layer. The maximum pooling layer was also applied after the last convolutional layer. Finally, it consists of two fully-connected layers with 4'096 neurons, respectively, followed by the fully-connected layer with softmax. The Rectified Linear Unit (ReLU) nonlinearity was applied after every layer output. Due to the high number of parameters, a dropout layer with a probability of 0.5 was added to the first two fully-connected layers. Furthermore, data augmentation was also used during training with ImageNet to minimize overfitting. Regarding the hyperparameters, the training was performed with stochastic gradient descent with 0.9 momentum and a weight decay of 0.0005, an initial learning rate of 0.01 divided by ten if the validation error is not improving, and about 90 epochs.

[1] https://pytorch.org/vision/stable/models.html

2.1.2 VGG-16

Simonyan and Zisserman (2015) attempted to investigate the impact on neural network depth due to the availability of larger databases such as ImageNet. They found that using smaller convolutional filter sizes of 3 x 3 in deeper networks with more stacked convolutional layers resulted in a performance improvement compared to AlexNet, which deploys 11 x 11 filters in the first convolutional layer (Krizhevsky et al., 2012). They proposed a standard architecture called VGG, which stands for Visual Geometry Group, with two different depths: 16 or 19 convolutional layers followed by three fully-connected layers with 4'096 neurons in the first and the second, followed by a fully-connected layer with C number of neurons and the softmax layer. Similar to the AlexNet architecture, ReLU has followed the hidden layers. In this thesis, the 16-layer network is being used to perform the experiments.

2.1.3 ResNet-50

He et al. (2016) achieved with their proposed deep residual networks architecture (ResNet) the lowest error rate at ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2015 (Russakovsky et al., 2015). The network tackles the degradation problem where the accuracy of classification tasks using deep networks starts to saturate and degrades over time. By adding an identity residual mapping between stacked layers, as shown in figure 2.1 the performance of the deep neural network will not suffer by specific layers since they can be bypassed, allowing much deeper networks in general without having additional parameters. The architecture is based on VGG nets with weighted convolutional layers, 3 x 3 filters with stride 2 at the end, followed by a global average pooling layer, the fully-connected layer, and the softmax layer with C outputs. After each convolutional layer, batch normalization and, ReLU non-linearity are performed. The residual networks provide the following number of layers: 18, 34, 50, 101, and 152. The three deepest versions utilize a so-called bottleneck design where instead of two stacked layers, three layers (1 x 1, 3 x 3, 1 x 1 convolutions) are used as a residual block with the described shortcut. This thesis uses the 50-layer network with 23 million trainable parameters to perform the experiments. The ImageNet training with augmented training data was done with stochastic gradient descent with 0.9 momentum, weight decay of 0.0001, and a batch size of 256 and involved up to $6 * 10^5$ iterations. The learning rate is initialized with 0.1 and divided by ten if the validation error is not improving.

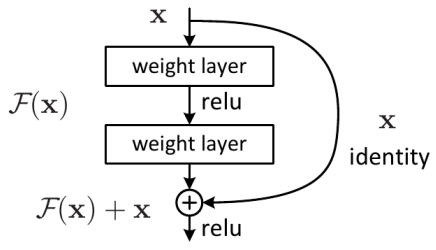


Figure 2.1: RESIDUAL LEARNING IN RESNET. This figure illustrates the residual learning framework where an identity layer shortcut was added between two layers. The output of the identity layer is then added to the output of the two layers. When ResNet-50, ResNet-101, or ResNet-152 is used, three convolutional layers are used as building blocks. Batch normalization is omitted in this figure for clarity (He et al., 2016).

2.1.4 MobileNet-V3

The work of Howard et al. (2017) tried to address the trend of ever better performance using deeper networks by presenting an efficient network architecture, but with advantages in terms of size and speed. To reduce the computations and the network size, a depthwise separable convolution was used instead of the standard convolution, divided into a depthwise and a pointwise convolution. In the standard convolution, the operations are performed over all input channels, i.e., the RGB channels. In depthwise convolution, on the other hand, separate convolutions are performed for each channel. Pointwise convolutions are sequentially applied to stack the individual channels back together. They are dense 1 x 1 convolutions and do not require memory reordering. After the convolutional layers, a batch norm and ReLU non-linearity are performed. The figure 2.2 illustrates the difference between standard convolution and depthwise separable convolution layers. Before the fully-connected layers, a global average pooling layer is put in place, which reduces the spatial resolution to 1. Finally, a fully-connected layer with C outputs is installed with an adjacent softmax layer at the end. To further reduce the computational cost while making a reasonable tradeoff in performance, two shrinking hyperparameters are introduced in the form of multipliers. These parameters can externally control either the dimension of each layer's output parameters or the inputs' size. Thus, the complexity of the network can be influenced. The width parameter α results in the equally fewer in and output parameters used on each layer and is always multiplied by the input and output channels. The resolution parameter ρ reduces the input resolution and the image representation in all layers. It is multiplied by the input feature map of a given layer. In 2018, a newer network version was introduced, building upon the earlier version by adding an inverted residual block to the depthwise and a point-wise convolution. Unlike the residual pattern, the residual block squeezes the input, then expands it and squeezes it again to the input dimension, which the authors call an inverted residual block. Moreover, the authors introduced a linear bottleneck where the last non-linearity in the inverted residual block is neglected to prevent loss of information. They have also limited the output range of ReLU non-linearity between 0 to 6. In this thesis, the third generation, named MobileNetV3-Large, presented by Howard et al. (2019) is applied. This network introduces a squeeze and excitation module in the residual layer, swish non-linearity, neural architecture search, and some manual layer removals, which are not further explained here. For pre-training with ImageNet, the learning rate is set to 0.1 with a decay rate of 0.01 every three epochs with a dropout rate of 0.8 and l2 weight decay.

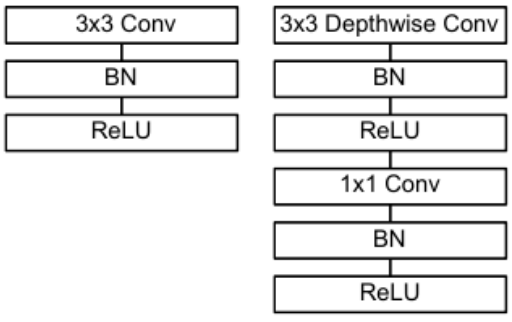


Figure 2.2: DEPTHWISE SEPARABLE CONVOLUTION. This figure illustrates the standard convolutional layer (left) in comparison with the depthwise separable convolution (right), additionally with batch norm and ReLU non-linearity. (Howard et al., 2017).

2.1.5 DenseNet-121

The ResNet architecture used the idea of shortcut identity paths connecting layers of different depths to provide the best information flow. This concept was also adopted as the foundation for the Densely Connected Convolutional Networks (DenseNet) architecture (Huang et al., 2017). However, what distinguishes this architecture from previous ones is that each layer has a connection to all other layers with matching feature-map sizes. This results in $\frac{L(L+1)}{2}$ total connections when the network is adopted for L layers. The number of total parameters of the network is significantly less than ResNet architectures because only 12 filters per convolutional layer are put in place. The authors also argue that overfitting and performance degradation can be effectively reduced due to the dense connections between layers, especially if only limited data is available. The training with ImageNet was done with stochastic gradient descent with an initial learning rate of 0.1 and a batch size of 264 for 90 epochs. The learning rate was effectively divided by 0.1 when 30 and 60 epochs were reached.

2.2 Related Work

In the following sections, the related work in the area of transfer learning is presented. Thereby, a distinction is made between fine-tuned approaches in section 2.2.1, applications with complementary shallow networks and feature extraction in section 2.2.2, feature transferability studies in section 2.2.3 and finally transfer learning experiments regarding different sample sizes and network sizes in section 2.2.4.

2.2.1 Fine-tuning

According to Zhuang et al. (2021), a typical method is fine-tuning the CNN pre-trained with ImageNet using the target data. In this case, it is necessary to change the number of network outputs to correspond with the size of the target label space. Specified layers of the CNN can be frozen so that the already learned layers are not modified during fine-tuning. Yosinski et al. (2015) showed through layer visualizations that learned features increase in complexity and variation in deeper layers. Analogously, Yosinski et al. (2014) denote the learned features in earlier layers as general and those in the last layers as specific. General features are simple structures, such as edges and corners, which are present in any computer vision domain. In contrast, specific features already have a high degree of variation and complexity. Objects like wheels, eyes, bottles, etc., are already identifiable when activities produced by deeper layers are visualized (Yosinski et al., 2015). Depending on the similarity of the source and target domains, fine-tuning tries to preserve the features that both domains share. Source-specific features should be adapted to match target-specific features. Thus, fine-tuning freeze particular layers and adapting the remaining ones to the specificity of the target dataset. Typically, small learning rates are adopted not to modify too drastically the already adapted weights of the pre-trained layer and thus the learned data characteristics of the source dataset. The information learned from the source dataset should be transferable to the target dataset to bridge the limited data availability. Several works have been published utilizing this transfer learning technique to overcome the problem of limited image data in different domains.

Approaches were presented where only specific layers are fine-tuned when classification performance would improve. Shaha and Pawar (2018) reported that fine-tuning VGG-19 utilized as a pre-trained network outperforms the hybrid learning approach where deep features were extracted from the pre-trained network and then used as an input for the SVM classifier, which will

be discussed in section 2.2.2. They performed their experiments with the GHIM10K and Cal-Tech256 databases. The first database consists of 20 classes with over 500 images per class. The second database comprises 256 classes with 80 images each. The authors did not investigate how the results might change with fewer samples per class. Soekhoe et al. (2016) analyzed the impact of the data size on the transfer ability of deep features and reported that freezing the first two to three layers would result in a performance boost compared with training the network from scratch, especially when only a few data samples are available.

Vrbančič and Podgorelec (2020) developed a method called Differential Evolution based Fine-Tuning (DEFT) and showed its application on osteosarcoma images outperformed comparable methods. Similarly, Guo et al. (2019) argue that it is not necessarily beneficial to fine-tune only the last coherent layers of the pre-trained network and propose SpotTune. This adaptive fine-tuning algorithm decides which layers of the pre-trained network should be frozen or not for each training sample. They tested their approach on the visual decathlon datasets resulting in favorable results compared to traditional fine-tuning strategies. Another approach is AdaFilter, proposed by Guo et al. (2020). Only specific convolutional filters based on activations of previous layers are selected to decide if fine-tuning should be applied or not on every sample instance. Similar to the other works, the authors used more than ten samples per class in their experiments.

2.2.2 Shallow Networks & Feature Extraction

Another transfer learning approach consists of using the features of the pre-trained network as input in additional shallow networks like SVM or neural networks. In this way, the pre-trained network is utilized as a feature extractor. During training, only the shallow network is adapted to the target dataset. Compared to the fine-tuned approach, a higher learning rate can be applied when using a neural network, also denoted as an adapter network, since the layers' weights are randomly initialized and otherwise not sufficiently adapted to the target dataset. Approaches in the field of transfer learning extract the deep features from the learned network, be it an adapter, pre-trained, or fine-tuned network, and use it to enroll a gallery in which the features represent the respective classes. The nearest neighbor classification can then be used to assign test images to a class. This technique involves probing the test images against the gallery and using a distance measure such as the cosine similarity distance.

In order to classify Alzheimer's disease based on MRI images of the brain, the work of Maqsood et al. (2019) utilized AlexNet pre-trained with ImageNet. The authors replaced the last three layers with a softmax layer, a fully-connected layer, and an output classification layer which they defined as adaptation layers. These layers are then trained using the MRI training data. They showed that their approach achieved better results than the traditional SVM model. However, it should be noted that the multi-classification task included only four classes and that more than 20 samples were used for each class. Their results indicate that unsegmented MRI images seem to have sufficient expressiveness to achieve a superior classification result compared to segmented images. These segmented images are extracted by building non-overlapping regions using k-Means clustering.

Günther et al. (2020) observed in their work about watchlist adaptation of faces as an open-set problem that a three-layer adapter network with 128 and 64 neurons in the first two layers on top of a pre-trained VGG2 face recognition network achieves better results than using only the pre-trained network alone. The raw output of the pre-trained network, also called logits, is directly fed into the adapter network. In this thesis, an additional performance comparison is conducted when either logits or deep features are fed into the adapter network. Additionally, they reported

favorable results considering the enrollment of gallery templates with sequential cosine similarity computation compared to raw features from a pre-trained network. More importantly, they reported that deep features of the adapter network perform better than the end-to-end classification. As this analysis was only tested on face datasets for an open-set problem, the findings can only be transferred to the classification of other domains to a limited extent.

The work of Cıbuk et al. (2019) also used deep features extracted from pre-trained networks, namely AlexNet and VGG-16, to carry out flower species classification (Flower17 and Flower102). The features of both networks were combined and selected by the so-called minimum redundancy maximum relevance (mRmR) algorithm and fed into an SVM classifier with a radial bases function kernel. They showed in their work that by combining and selecting the deep features extracted from two networks, their approach outperforms state-of-the-art methods. It is also worth mentioning that the authors extracted the features from two different locations within the two networks, from the activations of fc6 and fc7.

2.2.3 Feature Transferability

The extracted deep features from the network trained in advance with the source dataset could also have a negative effect if they do not match the characteristics of the target dataset. This effect is also called negative transfer and should be avoided if possible (Zhuang et al., 2021). However, since this is not straightforward to measure, several papers have already been published on this topic. Yosinski et al. (2014) examined the transferability of deep learning features extracted from AlexNet trained on the source dataset ImageNet. For this purpose, they randomly assign 500 classes from ImageNet to the source and target datasets. Since ImageNet contains clusters of similar classes, both databases contain statistically similar images by using this method. Moreover, less similar source and target databases are generated due to the existing hierarchical structure of ImageNet, which is also explained in section 4.1. The authors assigned only man-made entities to one database and natural entities to another. The transferability was examined by varying the number of layers that are utilized to transfer the knowledge to the target network. Their empirical experiments suggest that the CNN using the weights from the pre-trained network can increase the classification performance regardless of which layer weights are frozen, which persists after fine-tuning. They report that a higher dissimilarity between source and target tasks results in worse performance when weights are transferred without fine-tuning. However, fine-tuning pre-trained networks on small target datasets may lead to overfitting, predominantly when the network consists of many parameters.

Similarly, Neyshabur et al. (2020) investigated the transferability of deep features from pre-trained networks using ImageNet to the target domains DomainNet and CheXpert. Among other experiments, they compared the classification results from a network with random weight initialization, a pre-trained network on the source dataset, a network trained on the target dataset with the random initialization, and a pre-trained network on the source with additional fine-tuning on the target dataset. They reported that networks pre-trained with the source dataset converge faster when using the CheXpert dataset than random weight initialization. When benchmarking the same networks again with the DomainNet dataset, a performance increase was found compared to random weight initialization, suggesting that reusing deep features brings tangible benefits to image classification performance. They also argue that although the source and target datasets appear visually different, using a transfer learning approach may still help improve classification performance. In particular, the features from early layers contribute to the favorable transfer learning results.

2.2.4 Sample & Network Size

The approaches studied indicate that deep feature re-usage in either form benefits image classification. However, the influence of sample size on transfer learning approaches was not addressed. Zhu et al. (2021) consequently tried to analyze this impact. They report that the minimum number of training examples required to adapt a pre-trained network to produce a meaningful classifier for the target dataset can be estimated by setting a threshold. They used data from plastics manufacturing processes and utilized the ResNet-18 model as the pre-trained model for the analysis. Three different transfer learning methods were utilized: First, the linear classifier layer is replaced by a linear SVM layer. Second, the last layer is replaced by a new softmax classifier, and third, the whole pre-trained network can be fine-tuned. The experiments were repeated on sample sizes per class ranging from 20 to 200 image samples. The results indicate that the second approach performs much worse than the others ($\leq 90\%$ accuracy over all sample sizes). The fine-tuning approach with data augmentation shows the best performance, especially when more image samples are included. For one of the two datasets which were used in the experiments, even with fewer samples, the classification performance is more than 10% higher compared to the first method. The authors also compared the results of ResNet-18 with ResNet-121, finding that ResNet-18 performs superior with fewer samples per class when the fine-tuning approach is used. If using the first approach, the network with more parameters is preferable. The authors argue that feature extraction model elements are essential for the successful application of transfer learning. The experiments were conducted with binary datasets, and the validity of the classifiers with less than 20 samples per class was not tested.

The analysis conducted by Zhu et al. (2021) regarding the effectiveness of transfer learning using different model parameter sizes has been thoroughly investigated and extended by Kornblith et al. (2019). The authors examined the transfer classification performance of 16 modern networks pre-trained with ImageNet on 12 image multi-classification datasets which consist of 20 to 5'000 images per class. Their findings indicate that the ImageNet top-1 accuracy of the networks correlates with the classification performance of transfer learning. Thus, higher ImageNet classification rates yield more general deep feature representations of the same network.

Chapter 3

Transfer learning approaches

After describing the most common transfer learning approaches in chapter 2, this chapter highlights the approaches that will be explored in more detail in the experiments. First, a distinction is made between end-to-end, nearest neighbor, and SVM classification, which is elaborated upon in section 3.1. Then, a further distinction is drawn concerning pre-trained networks without additional training of the target dataset, fine-tuned networks, and adapter networks described in more detail in section 3.2. Except for the first one, the networks are trained using stochastic gradient descent with the objective of minimizing the cross-entropy loss function defined in equation (3.1) where C denotes the number of classes, t_c the true class, and y_c the predicted class.

$$J^{CE} = -\sum_{c=1}^{C} t_c \log(y_c) \tag{3.1}$$

3.1 Classification Variants

Since different performance results were found when using end-to-end and nearest neighbor classification in the work of Günther et al. (2020), these methods are also examined in more detail in this thesis as variants. Moreover, as with other approaches outlined in chapter 2, the SVM classifier is used as an additional transfer learning approach. The figure 3.1 illustrates the differences which are also outlined in more detail in section 3.1.1, 3.1.2 and 3.1.3. The trained layers are utilized in all cases until before the last fully-connected layer. Subsequently, the entire network including the last fully-connected and softmax layer is used for end-to-end classification (green arrow), or the deep features are extracted before the last fully-connected layer (red arrow). The features are then used for either nearest neighbor or SVM classification. The variants are evaluated in terms of their classification accuracy using the test samples. Classification accuracy is particularly used as a performance indicator. The defined equation (3.2) uses a Kronecker Delta function $\mathbb{I}$ which sums up only correctly classified classes where the predicted class of the n-th sample $\hat{c}^{[n]}$ equals the true class of the n-th sample $t^{[n]}$.

$$Acc = \frac{1}{N}\sum_{n=1}^{N} \mathbb{I}(\hat{c}^{[n]} = t^{[n]}) \tag{3.2}$$

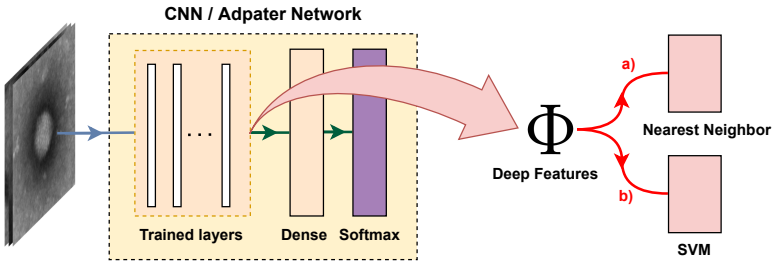


Figure 3.1: CLASSIFICATION VARIANTS. This figure illustrates the three classification variants used to conduct the experiments. End-to-end classification leverages all available network layers, including the last fully-connected layer and the softmax layer (green arrow). The red arrow indicates the output immediately before the last fully-connected layer is used either for a) Nearest neighbor or b) SVM classification.

3.1.1 End-to-end Classification

The trained network is directly used in this method to classify the target dataset. The test image samples of the target dataset are fed into the trained network. The softmax layer assigns the raw outputs of the last fully connected layer, also called logits $\vec{z}$, to the class probability values in the range of 0 and 1, as given in equation (3.3). If the individual probabilities are summed up, the result is 1. The predicted class $\hat{c}$ is then determined by the largest probability value, as given in the equation (3.4).

$$S(z_c) = \frac{\exp(z_c)}{\sum_{j=1}^{C} \exp(z_j)} \tag{3.3}$$

$$\hat{c} = \underset{1 \leq c \leq C}{\operatorname{argmax}}\, S(z_c) \tag{3.4}$$

3.1.2 Nearest Neighbor classification

The trained networks are used as feature extractors, illustrated in Figure 3.1, to create a deep feature gallery representing the features of each class of the target dataset. The target samples are fed into the learned network, but the last two layers are not needed. Instead of using the previous layer's output as input for the last fully-connected layer, the output data, called deep features, is utilized to enroll the gallery. For each class, at least one feature will be available as a class representative. The gallery will be enrolled in two different methods:

1. **COS:** The features of all currently used target samples per class are included in the gallery.
2. **MEAN:** A single representation per class is included in the gallery. The single representation is created by calculating the average over all available features per class.

Let K be defined as the total number of features in the gallery over all classes, then $\vec{\varphi}_k$ denotes the feature representation in the gallery where k refers to the k-th feature. K corresponds to at least C if the average feature per class or even only one sample per class is used. All target samples are extracted in the same way as described above, but the gallery is enrolled only with the training samples. The validation features will be applied for computing AUROC, a regularization metric for early stopping as outlined in chapter 5.3.2. The deep features are also extracted for the

test samples. A similarity measure is used to predict the class of the test pattern. This measure determines the highest similarity with the putative best matching deep features. The greatest similarity finally represents the predicted class $\hat{c}$ as shown in equation (3.6). For this purpose, the cosine similarity score is applied, which is described in equation (3.5). The score is computed between $\vec{\varphi}_k$ for each k and the probe sample $\vec{g}$. Given the nature of the equation, the features are normalized by themselves. In MEAN, whether the features are normalized before or after averaging is relevant since a different result is obtained. For the approach, however, an additional normalization is omitted so that the implicit normalization only takes place after the averaging.

$$Sim_k(\vec{\varphi}_k, \vec{g}) = \frac{\vec{\varphi}_k \cdot \vec{g}}{\|\vec{\varphi}_k\| \, \|\vec{g}\|}, \forall k \in [1, 2, \cdots, K] \tag{3.5}$$

$$\hat{c} = \underset{1 \leq k \leq K}{\operatorname{argmax}} \, Sim_k(\vec{\varphi}_k, \vec{g}) \tag{3.6}$$

As mentioned in section 1.1, the transfer learning approach classifications are repeated in the experiments with gradually increasing numbers of training samples per class. Thus, since conceptually more gallery data is available, the following distinction involves gallery samples relevant for selection:

1. **Increasing Gallery (I-GAL)**: With more training examples per class, the number of features per class that are available for the gallery increases to the same extent. COS and MEAN are formed according to the available features.

2. **Constant Gallery (C-GAL)**: The available features do not change with this variant. When using the fine-tuning or adapter network, five samples per class are not taken into account during training but are available for classification after training. The samples used for training are not taken into account for the enrollment of the gallery. This means that the gallery will not have more features available, even with a higher number of samples per class. COS and MEAN are therefore always formed with the same five image samples. It is important to point out that the deep feature representation may change depending on the number of training samples per class, although the same five samples are always used. However, the extracted features will definitely stay the same when the pre-trained network is used without training.

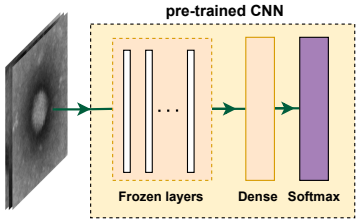
3.1.3 Support Vector Machine (SVM)

Cortes and Vapnik (1995) developed a binary classification learning algorithm that solves binary classification tasks by maximizing the distance between classes in multi-dimensional space. The network was extended to multi-classification problems by splitting the entire task into multiple binary classifiers using either a one versus one or a one versus rest approach. In the first case, $\frac{C(C-1)}{2}$ hyperplanes must be constructed as each pair of classes is compared with each other, while in the second case, C hyperplanes must be built (Weston and Watkins, 1998). The experiments in this thesis are solely based on the one versus rest approach since fewer classifiers are needed, resulting in faster classification performance. Deep features are used to train the classifier. After extraction from the learned network using the training image samples, the deep features are normalized with the Euclidean norm (l^2-norm). Then, a grid search is performed using the extracted deep features to find the optimal soft margin parameter C, which should help define a decision boundary in favor of generalization that also allows for misclassification. The regularization parameters available for selection are 0.1, 1, 10, and 100. The deep features of the test images are also used for classification. The deep features of the test images are used for classification. After normalization, each of the C constructed hyperplanes can make a binary statement

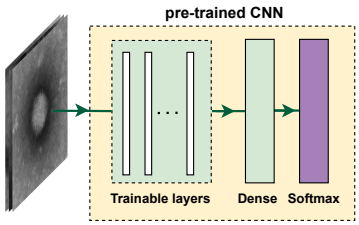
about the possible class outcome. The use of majority voting yields a class that has been proposed most often and is consequently predicted. Due to the increased complexity of other kernels and the limited data available for the experiments, only the linear SVM is applied in this thesis.

3.2 Network Variants

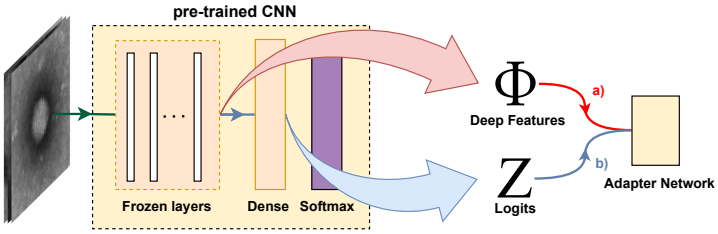
In order to examine the effectiveness when using a few samples per class, three different transfer learning network approaches are examined. As a starting point, the CNNs pre-trained with ImageNet, described in chapter 2.1, are always utilized and modified accordingly. The figure 3.2 illustrates the differences which are also outlined in more detail in section 3.2.1, 3.2.2 and 3.2.3.



(a) Pre-trained Network



(b) Fine-tuned Network



(c) Adapter Network

Figure 3.2: NETWORK VARIANTS. This figure illustrates the network variants (a) Pre-trained network where all layers are frozen (b) Fine-tuned network where the layers will be adjusted during training with the target dataset and (c) Adapter network where the layers remain frozen but either a) the deep features are extracted just before the last fully-connected layer or b) the output of the last fully-connected layer, the logits, serve as input for the adapter network. The approaches (b) and (c) will be trained using the target dataset.

3.2.1 Pre-trained network

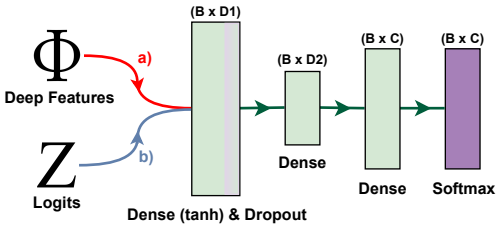
In this method, the respective pre-trained network is directly used to classify the target dataset, i.e., no additional training is performed. As shown in figure 3.2(a), the trained layers are frozen so that no weight adjustment can occur. For classification, the network is used solely as a feature extractor. Thus, the last fully-connected layer and the softmax layer are not needed in this case. In principle, this signifies that more samples per class for the classification do not extract features that are more adapted to the target network. When using C-GAL, the classification accuracy does not change even with a larger number of training samples since the same five samples are constantly employed in the gallery.

3.2.2 Fine-tuned network

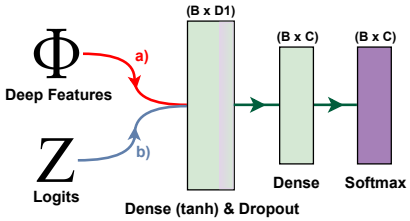
The trained layers of the pre-trained network do not remain frozen in this method, as shown in figure 3.2(b). During training, all pre-trained weights are fine-tuned according to the target dataset. The last fully-connected layer and the softmax layer of the pre-trained network are aligned according to the number of classes and then trained with a small learning rate using the target training dataset. When using the fine-tuned network, all classification variants under section 3.1 can also be deployed.

3.2.3 Adapter network

When using the adapter network variant illustrated in figure 3.2(c) the learned layers in the pre-trained network remain frozen. Only the additionally introduced adapter network, which can be used either as a 2-layer or 3-layer variant, is trained with the training target database. The adapter network in both configurations is shown in figure 3.3. The learned information of the pre-trained network is extracted and serves as input into the adapter network. In order to investigate the effect studied by Yosinski et al. (2014) regarding the transferability of features, it is possible to perform the extraction at two different layers of the network as illustrated in figure 3.2(c). In variant a), the layer outputs before the last fully-connected layer are used as input to the adapter network. This is the same place where feature extraction occurs when performing nearest neighbor classification in section 3.1.2. However, when using variant b), the outputs of the last fully-connected layer, the logits, are utilized. The layer dimensions in the first layer of the adapter network are based on the dimensions of the deep features or logits. The output dimensions in variant a) depend on the network architecture and therefore follow the defined feature size in table 2.1 in chapter 2.1. When using logits, the dimension depends on the number of available classes in a respective dataset. Since ImageNet is used as the source dataset for all networks, the output dimensions are the same for all pre-trained networks. Before the deep features or the logits are fed as input into the adapter network, normalization with the Euclidean norm is performed analogously to the mentioned normalization when using the SVM classifier in section 3.1.3. Since the network learns in batches, the layer dimensions are also specified in batches, which is determined before training begins. An experiment in this work addresses the appropriate choice of the number of layers and dimensions based on the accuracy performance, which is why no absolute dimensions are reported in figure 3.3. As with the fine-tune network approach, all classification variants can be applied.



(a) Three-Dense-Layer Adapter-Network



(b) Two-Dense-Layer Adapter-Network

Figure 3.3: ADAPTER NETWORKS. This figure illustrates the architecture of the adapter networks with (a) Three-Dense-Layers (b) Two-Dense-Layers. The extracted a) Deep features or b) Logits from the pre-trained network are normalized before being fed into the adapter network. B is the specified batch size, $D1$ the output size of the first fully-connected layer, $D2$ the output size of the second fully-connected layer and C the number of classes.

Chapter 4

Databases

This chapter describes the databases used to perform the experiments in this thesis. Each one of them is an image database with more than 20 different labeled classes. Certain classes are excluded from the analysis for some databases because the number of samples is too small compared to other classes and would have prevented a thorough analysis. ImageNet was chosen as the source dataset because this dataset is already widely used and has large class variability. The CNNs used also have networks that have already been pre-trained with ImageNet. The target datasets were selected to have varying degrees of similarity to ImageNet to examine the transfer learning methods studied on similar but more diverse datasets. The key database characteristics are shown in table 4.1.

Table 4.1: DATASET CHARACTERISTICS. The table shows database characteristics, such as the number of image samples, class amount, image resolution, etc. The resolution values given are averaged values determined by random sample testing. Therefore, the resolutions may differ from the outlined values.

	Size	Num. of classes	Training samples per class	Resolution
ImageNet	1.2 M	1 k	0.5 - 1 k	400 × 350
Aircraft	10 k	30	33	1'200 x 768
Fruit and Vegetable	10 k	36	95	200 x 250
Indoor Scenes	15.62 k	67	67	256 x 256
Office-31	4.652 k	31	44	300 x 300
Virus	1.245 k	22	31	256 x 256

4.1 ImageNet

The ImageNet database was proposed by Deng et al. (2009) and has since been extensively utilized for many deep learning computer vision tasks, amongst others, for image classification and image recognition. At this point, the dataset already consists of over 14 million hand-labeled images, each divided into over 20'000 synsets, semantically identical word classes, which can then be grouped, resulting in a hierarchical structure as illustrated in figure 4.1. In this thesis, the ImageNet subset of 1'000 classes and 1.2 million images provided by PyTorch (Paszke et al., 2019) is adopted and was also used at the LSVRC. The classes represent, among others, a variety of animals, everyday objects, and food. Due to the variety of classes, the average number of 500-1'000 images represented by a synset, the availability, and the average resolution of 400 × 350, ImageNet is suitable for pre-training deep learning networks for transfer learning tasks and was also employed for this purpose in this work.

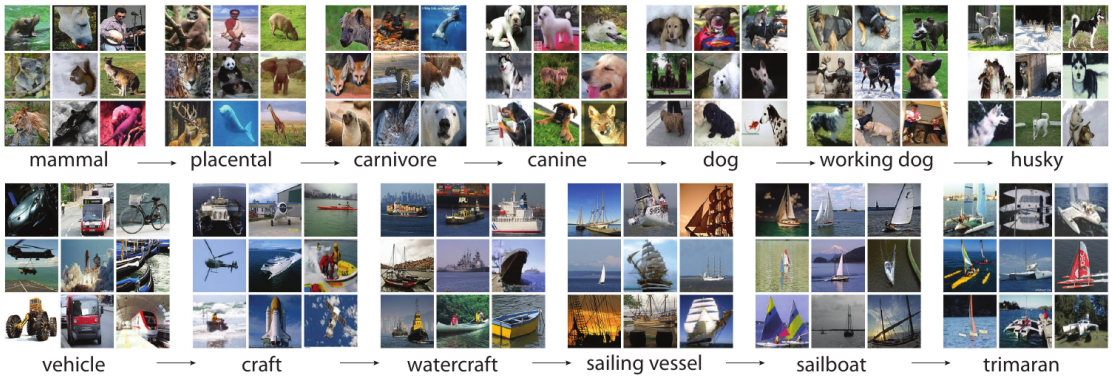


Figure 4.1: IMAGENET SUBTREES. This figure illustrates ImageNet's hierarchical structure by visualizing the mammal (top) and vehicle (bottom) subtree, starting with the root category on the left, followed by the leaves on the right. (Deng et al., 2009).

4.2 Aircraft

The Aircraft database proposed by Maji et al. (2013) consists of 10'000 aircraft images with approximately 1-2 Megapixels resolution divided into 100 different classes. It can also be structured hierarchically in four different ways. The finest grouping recognizes each aircraft model individually, but according to Maji et al. (2013), this is not measurable with the external depiction of airplanes alone and therefore not suitable for a fine-grained image classification task. Second, data can be clustered into model variants that cannot be visually distinguished. Third, families cluster model variants to form fairly distinct discrepancies. And finally, aircraft families from the same manufacturer form the most granular hierarchy with 30 classes overall and approximately 33 samples per family. Only this last variant is considered for the experiments in this thesis. Two sample classes of the database are shown in figure 4.2.

(a) Boeing (b) Eurofighter

Figure 4.2: AIRCRAFT MANUFACTURER IMAGES. This figure shows an example of nine training images from each of the two classes of the Aircraft dataset. (a) Boeing manufacturer class (b) Eurofighter manufacturer class.

4.3 Fruit and Vegetable

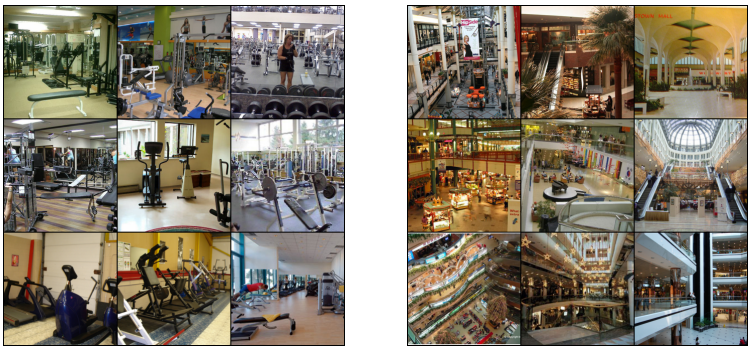
The Fruit and Vegetable dataset, with 36 different classes, overlaps significantly with the ImageNet dataset, making it an ideal candidate for homogeneous transfer learning processes. The number of fruit classes is smaller than the vegetables, with only ten different representations. The database was scraped from the Internet by Seth (2020) and has already been used to generate a balanced training, validation, and testing split. Each class consists of approximately 100 samples in the training set and ten samples each for the validation and testing set. The author does not specify the resolution of the images. However, by random sampling, it was found that the resolution varies greatly between high-resolution images up to about 5'000 x 5'000 pixels and low-resolution samples of only about 200 x 250 pixels. Several images have already been segmented and centered on their respective objects on a white background. However, numerous examples contain multiple objects that are not centered. Two sample classes of the database are shown in figure 4.3.

(a) Orange (b) Eggplant

Figure 4.3: FRUIT AND VEGETABLE IMAGES. This figure shows an example of nine training images from each of the two classes of the Fruit and Vegetable dataset. (a) Orange class (b) Eggplant class.

4.4 Indoor Scenes

This dataset proposed by Quattoni and Torralba (2009) was collected by web scraping online sharing tools (Flickr) and the LabelMe dataset. It consists of 67 balanced classes with different indoor scenes. The domain is divided into the group classes store, home, public spaces, leisure, and workplace, with several subclasses containing 15'620 images with a minimum resolution of 200 pixels on the smallest axis. Since the authors did not provide a validation dataset, ten samples per class from the training set are extracted for this purpose. In contrast to the other datasets mentioned above, this one does not focus on a specific object in the image but rather on the whole scene in general and therefore includes several objects. The classification task is particularly challenging because there is high in-class variability in the images, which can also be seen in figure 4.4.

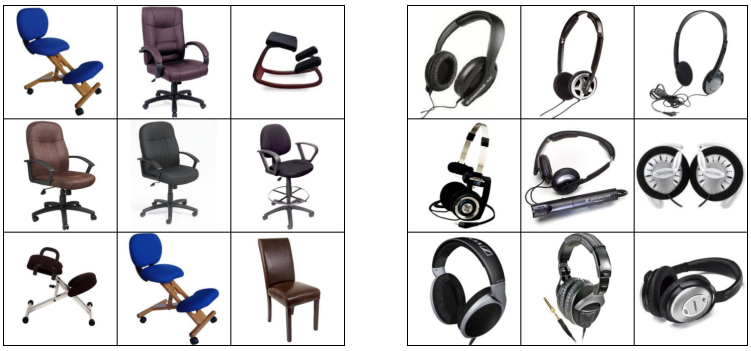


(a) Gym (b) Mall

Figure 4.4: INDOOR SCENES IMAGES. This figure shows an example of nine training images from each of the two classes of the Indoor scenes dataset. (a) Gym class (b) Mall class.

4.5 Office-31

The dataset was initially developed by Saenko et al. (2010) for domain adaptation tasks where the classes to be predicted remain the same across different domains. Each of the three different domains contains 31 classes of office equipment with a total of 4'652 samples. The first domain *amazon*, also used for this work, provides significant intra-class variations from a canonical viewpoint and a segmented, centered object with a white background, as shown in figure 4.5. The medium-resolution images with 300 x 300 pixels were collected by web scraping of Amazon products. The other two domains were recorded with a low-resolution webcam and a high-resolution camera from different angles. Since this thesis investigates the transfer learning methods between a source and a target dataset with different classes, only the *amazon* dataset is considered.

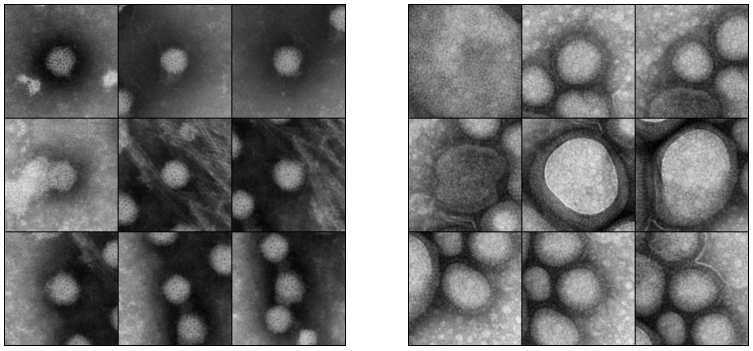


(a) Desk chair (b) Headphones

Figure 4.5: OFFICE-31 IMAGES. This figure shows an example of nine training images from each of the two classes of the Office-31 dataset. (a) Desk chair class (b) Headphones class.

4.6 Virus

This dataset was proposed by Matuszewski and Sintorn (2021) and consists of 1'245 images of 22 unbalanced virus classes, which were acquired by two electron microscopes in Germany. The raw images were preprocessed by centering on the captured virus particles and then cropped such that the pixel size equals 1nm and rescaled to 256 x 256 pixels. By nature, some virus particles are elongated, which need special treatment to prevent the identical virus particles from occurring in several image patches. Due to the multiple occurrences of particles in one raw image, the preprocessing resulted in more images per virus class. In order to perform the experiments in this thesis consistently with the same sample size per class, the *Sapovirus* class is neglected. The authors also carefully split the data into training, validation, and test sets, particularly preventing data leakage between sets.



(a) Rotavirus (b) Influenza

Figure 4.6: VIRUS IMAGES. This figure shows an example of nine training images from each of the two classes of the Virus dataset. (a) Rotavirus class (b) Influenza class.

4.7 Data preparation

The five databases described in this chapter need only minimal pre-processing steps, as the authors have thoroughly prepared them. Since the Office-31 dataset has not yet been split, ten randomized image samples per class were used for validation and testing. The remaining image samples are assigned to the training set. Regarding the Indoor Scenes dataset, merely no validation set exists. Thus, it was reassigned from the training set using the same method as described before. Additionally, a separate script was created that generates a *csv*-file for each split per dataset, listing all image samples, each containing the absolute image path, the class label, and a number that can be uniquely assigned to a class per line. Since the image samples are sorted alphabetically by class, class numbers from 0 to C can be created, which are later used for image classification. To achieve a good separation from the training logic and to iterate easily through the image samples, the classes *Dataset* and *DataLoader* provided by PyTorch are used. The prepared *csv*-file for the respective split is read by the *Dataset* primitive, which was implemented specifically for handling image samples, deep features, and logits. The *DataLoader* enables the image samples to be fed into the network in mini-batches, shuffled during the training of the networks. By using the iterable, multiprocessing steps and image transformations for data augmentation described in section 5.3.2 can also be applied. A loaded image is converted into a tensor with floating-point numbers between 0 and 1, followed by a normalization step where the

values are subtracted by the mean and divided by the standard deviation. The operations result in bounded values of [-3, 3]. Mean ([0.485, 0.456, 0.406]) and standard deviation ([0.229, 0.224, 0.225]) are obtained from ImageNet training data and applied to every image sample over all datasets. Due to the different resolutions of image samples according to table 4.1 and the fact that the pre-trained networks expect an input resolution of 224 x 224 pixels, the images are re-scaled so that the smaller side is 256 pixels. The re-scaling is followed by a center crop of 224 x 224 pixels.

Chapter 5

Experimental framework

This chapter lays the foundation for the experiments by describing the procedure and the implementation details. Network training and classification will always follow the same procedure as illustrated in algorithm 1. Since the datasets each have a different number of training samples, as seen in table 4.1, the number of samples per class with which the networks are trained will vary. However, in any case, the training and classification for a deep transfer learning approach will be performed 11 times with an increasing sample size per class. A logarithmic scale bounded by a minimum of 1 and a maximum of the largest number of possible samples per class is used to choose the number of samples per class during training. As a result, experiments focus on the small number of samples per class. Nonetheless, an analysis of the trend in classification performance can be seen when more training samples are available per class. In order to make general evaluations of the different training methods, the training images are reshuffled in each epoch. This ensures that their order changes, although the same training images are always used. Also, image transformations are used for data augmentation as described in section 5.3.2. After each network training with a specific sample size per class, it is essential to reset the learned weights of the network so that the training remains unbiased and can be compared with each other. While the number of training data per run constantly changes, the validation and test data remain the same. This fact also allows comparing the classification performance between changed sample sizes per class of a deep learning approach and across approaches. Furthermore, the learned networks are stored on the local disk. Therefore it is possible to reuse the network for classification. This is especially useful when using deep features classification approaches since, in this case, networks already trained with the target dataset can be reused to extract the deep features.

5.1 Experiment structure

The performance comparisons follow the same chronological order as the research questions mentioned in the introduction. First, according to **RQ 1**, the adapter network design is evaluated. Since there are many design options when using the adapter network approach (e.g., number of layers, number of neurons per layer, use of logits or deep features, selection of an early stopping scheme), the design decisions in this work are examined using only the Indoor Scenes dataset and ResNet-50. Initially, it will be investigated whether the logits or the extracted deep features from the pre-trained network are more effective in terms of accuracy when they are used as input to the adapter network. Then, the early stopping regularization, described in section 5.3.2, is investigated. Lastly, two and three fully-connected layers with different numbers of neurons in the first two layers are compared in terms of their classification performance, and the most promising

[1] https://numpy.org/doc/stable/reference/generated/numpy.logspace.html

Algorithm 1 LEARNING & CLASSIFICATION PROCEDURE. This algorithm describes the simplified learning and classification process. M defines the number of available maximum training samples per class and E the number of epochs. The array *sizes* is generated using the Numpy function *logspace*.[1]

```
test ← load test data
validation ← load validation data
model ← Load network
sizes ← logspace(1, M, 12)
for size in sizes do
    reset network state
    train ← load training data with size samples per class
    for epoch = 1, 2, ..., E do
        network training with train
        network validation with validation
    end for
    network classification with test
end for
```

design is selected. The results are presented in chapter 6.1. In all subsequent performance comparisons with other deep learning approaches, the adapter network design determined in this way is used.

Secondly, according to **RQ 2**, all five pre-trained networks and all five target datasets are examined applying nearest neighbor classification, more specifically, by enrolling I-GAL with the MEAN approach. The results are compared with the top-1 accuracy scores reported on the PyTorch website. Suppose the results indicate that the performance of a particular pre-trained network is significantly worse than that of other networks under study. In that case, it is not considered for further analysis. Third, according to **RQ 3**, the approaches mentioned in chapter 3 are compared to determine which transfer learning technique is particularly suitable for a small number of image samples per class. However, it is also always discussed how the approach performs when more samples are available per class. The analysis also allows statements to be made about **RQ 4**. Each transfer learning approach is performed once with the described logarithmic scale for each pre-trained network selected based on their performance in **RQ 2**. Subsequently, the mean and standard deviation for all individual pre-trained network results is calculated per sample size and approach. This indicates the consistency of the approach over different pre-trained networks. The experiments are divided into the following sub-analyses:

1. Comparison of end-to-end classification using the adapter and the fine-tuned network.

2. Comparison of nearest neighbor and SVM classification using the adapter, the fine-tuned, and the pre-trained network. The mean and standard deviation of the performance results applying nearest neighbor (with COS and MEAN) and the SVM classification of a network training approach are calculated.

3. Based on the results of step 2, the most promising approach, including nearest neighbor and SVM classification (adaptive, fine-tuned, or pre-trained network), is used to analyze the classification methodology. The nearest neighbor (with COS and MEAN) and the SVM classification results are compared.

4. The best variants of steps 1 and 3 are compared here to finally make a statement about which transfer learning approach is best suited for only a few samples per class.

Fourth, **RQ 5** examines the significance of the deep features in the gallery by comparing the classification performance of the increasing gallery (I-GAL) with the performance of the constant gallery (C-GAL). The gallery of the first approach becomes correspondingly larger, with more samples per class. Only the optimal deep feature approach identified by **RQ 3** is used for classification. Again, only the selected pre-trained networks from **RQ 2** are used to calculate the mean and standard deviation.

5.2 Evaluation metrics

Transfer learning methods aim to achieve the highest possible correct classification rates from limited available data. With the help of the performance indicator, different transfer learning approaches can be compared. The accuracy metric depicted in equation (3.2) is calculated several times with increasing samples per class to compare their resulting performance in terms of progression when more images are available per class. The analysis is primarily visual, as the trend in the accuracy values of the different approaches when using more and more samples per class is most easily represented by respective line plots. The sample sizes per class are plotted on the x-axis, and the effective precision values are plotted on the y-axis. Since the focus is on small samples, the x-axis in the graphs is used as a logarithmic scale. When transfer learning approaches are examined across multiple pre-trained networks, the accuracy scores are considered together, and their mean and standard deviation are calculated. In addition to the general accuracy analysis, confusion matrices are created to compare classification performance at the granularity level of individual classes. A $C \times C$ matrix is created where the true classes are represented on the x-axis and the predicted classes on the y-axis. Training and classification procedures are repeated ten times to determine the significance of the results in the case of the adapter network design experiments. The mean and standard deviation results for the approach are calculated to compare the performance. In the remaining experiments, the mean and standard deviation from the classification results of the same transfer learning approach across the relevant pre-trained networks is computed.

5.3 Implementation details

The conducted approaches are implemented exclusively in Python. The libraries PyTorch, NumPy, Pandas, and Sklearn are also used. The adapter network is developed utilizing the module base class of PyTorch, which allows defining the forward pass of the network as well as explicit layers with the required input and output variables.

5.3.1 Network design

All network designs are based on one of the pre-trained networks described in chapter 2.1. PyTorch conveniently allows loading networks already pre-trained with ImageNet into the local cache, which allows for simple modification afterward. For this purpose, a separate class was defined, containing the respective network objects loaded by PyTorch. All networks used are optimized with stochastic gradient descent with 0.9 momentum. PyTorch combines the log-softmax and the negative log-likelihood loss to build the cross-entropy loss function. In this way, no explicit softmax layer must be defined in the network implementation. When using nearest neighbor classification or SVM, the last fully-connected layer in all networks is replaced by an identity layer, which outputs the input unchanged. For this purpose, the extraction function and the nor-

malization, in case SVM or the adapter network are utilized, are implemented in a separate class. The network design differs between the already mentioned network variants in chapter 3.2:

1. **Pre-trained network:** Since this variation does not include training and is not used for network classification using softmax, no modifications need to be made.

2. **Fine-tuned network:** This network uses the already pre-trained network from the source dataset with the learned weights and is fine-tuned using the target dataset. Thus, it is necessary to verify that the gradients of the network needed to calculate the updated weights in each layer are enabled to guarantee that the weights are actually updated during training. Since the networks pre-trained with ImageNet consist of 1'000 outputs in the last fully-connected layer, which reflects the number of classes in the dataset, the number of outputs is adapted to the respective number of classes from the target dataset. The training includes 50 epochs with a learning rate of 0.0001 and decay of 0.1 when reaching a plateau, which is described in more detail in section 5.3.2. The training is ideally performed with a low learning rate to avoid over-adaptation of latent features that have already been learned from the source dataset.

3. **Adaptive network:** When using the adapter network, the last fully-connected layer of the pre-trained network is replaced by an identity layer when the deep features are utilized. If the logits are to be used as input to the adapter network, the last fully connected layer remains unchanged, which also leaves the logits unchanged. The logits or deep features are normalized and fed into the adapter network. Care must be taken that the input size of the first fully-connected layer of the adapter network corresponds to the size of the deep features or logits the batch size. However, no special care is required concerning batch size, which is controlled internally by PyTorch. A variable can control the output size of the first layer. Also, two or three fully-connected layers can be utilized, as seen in figure 3.3. If three fully-connected layers are chosen, the output of the third layer can also be controlled with a variable. Also evident in figure 3.3, the tanh activation function is applied to the output of the first fully-connected layer, followed by a dropout layer, described in section 5.3.2 with a 0.5 probability that is active only during training. The softmax layer is essentially the same according to the number of classes already mentioned when designing the fine-tuned network. The training is conducted with 100 epochs with a learning rate of 0.01 with the possibility of using the early stopping approach described in chapter 5.3.2, which could still reduce the number of epochs.

5.3.2 Regularization

Since only limited image data are used in this work and many parameters have to be adapted in deep learning networks, there is an increased risk that the network overfits the dataset. This problem can be reduced by regularization techniques, which is why these techniques are used in implementing the networks. The following sections describe the applied techniques.

Data augmentation

While designing AlexNet, Krizhevsky et al. (2012) already recognized that even the ImageNet dataset is too small to train a network with more than 60 million parameters without significant overfitting. For this reason, image transformations on the original images are used to increase the number of training samples artificially. PyTorch also remedies this by introducing a module that chains transformation arbitrarily together. Care has been taken during the transformation to ensure that the images are only transformed in the same way they would naturally occur. For

the first transformation, an arbitrary patch of the image and an arbitrary aspect ratio is selected. This section is then re-scaled to the corresponding needed input size of 224 x 224 pixels. The parameters used correspond to the standard values defined by PyTorch. Secondly, an image's brightness, contrast, saturation, and hue are randomly changed. Thirdly, a horizontal flip with a probability of 0.5 is applied. After the data augmentation steps, ImageNet-specific normalization is performed, which is described in chapter 4.7. Since only more data are needed for the training, no data augmentation steps are performed for the validation and test sets. The conversion to a tensor and the normalization remain the same as already described when using training samples.

Dropout

Another regularization technique is dropout, which is applied in almost all pre-trained networks in this thesis (AlexNet, VGG-16, DenseNet-121, MobileNet-V3). This technique ensures that the neurons in the hidden layers do not learn features dependent on other neurons. By preventing this so-called co-adaptation, more robust learning is possible. In the experiments, the output of neurons is set to zero with a probability of 0.5. Therefore, these randomly selected neurons are neglected when computing the backpropagation step during training. Dropout is applied in the adapter network after the first fully-connected layer and hyperbolic tangent activation function (tanh), as illustrated in figure 3.3.

Early stopping

When training deep learning networks, classification performance on the test set may decrease as training epochs increase. This fact may occur since the network overadjusts to the training data and is therefore unable to handle image data that is not yet known. This problem can occur, especially when insufficient training data is available. Therefore, the early stopping technique is used for training the adapter network that terminates the learning process prematurely when this overfitting occurs. To determine the stopping process, a predefined metric compares the best value so far with the current value at each epoch. Overfitting is detected when the current value indicates worse performance than the best value. Since stochastic gradient descent is utilized as the optimization step, it would be counterproductive to terminate the training process immediately after the first detection since the results fluctuate considerably. Therefore, a counter variable is defined, which is incremented with each detection. Finally, a threshold is selected at which training is terminated after the specified number of detections is reached. The best model is used as a reference for this transfer learning approach. Two different metrics are utilized to detect overfitting:

1. **Validation error:** The equation (3.1) describes the cross-entropy loss function, whose derivation is essential for the backpropagation step and thus for the actual learning process of the network. In order to get a reference of how the network behaves at this point of the learning step concerning not yet known data, the equation is used to determine the error rate of the validation data set.

2. **Area under receiver operating characteristic curve (AUROC):** Receiver operating characteristic (ROC) graphs visualize the performance of machine learning classifiers. The false-positive rate (1 - specificity) and true positive rate (sensitivity) are calculated based on the classification results of the test set and then plotted on the x-axis and y-axis, respectively, to create a ROC graph. With the help of the prediction scores, which originate from the output of the softmax layer, the 1 - specificity and sensitivity are determined with arbitrary frequency in the range [0,1] each time with the corresponding threshold and finally plotted as a data point. The resulting data points are connected to form a line. Calculating the area under the curve of this plotted line results in a value in the range [0,1]. The closer this value

approaches 1, the better a classifier can distinguish between the correct and wrong classes. ROC graphs are primarily designed for binary classifiers. Hence, an AUROC value of 0.5 means that the classifier cannot distinguish between correct and incorrect class and only determines a random or always a constant class (Fawcett, 2006). Analogous to the extension of the SVM classifiers for multi-classification in chapter 3.1, the one versus rest approach can also be used where the actual class is the positive class and all other classes the negative class. Sklearn provides the *roc_auc_score* function[2] to calculate the AUROC without computing the sensitivity and specificity by hand. During training, the deep features of the validation data are extracted. A pairwise similarity calculation analogous to equation (3.5) is computed, which results in a square matrix. The similarity values calculated over the same classes are the positive ones. All others are classes with negative connotations. Since the comparisons are repeated in the squared matrix, only the strictly upper triangle is required for the calculation. The strict upper triangle is flattened and serves as input to the roc_auc_score function along with the label assigned to each value, which contains 1 for positive and 0 for negative class connotations.

Learning rate decay

As the training of a deep neural network progresses, it may be practical to reduce the learning rate chosen at the beginning to improve the classification performance of the neural network when the accuracy stops improving further (Smith, 2018). Meanwhile, different methods exist to determine when and how the learning rate should be minimized. For example, while presenting the MobileNet-V3 architecture, the authors report the application of a decay rate of 0.01 every three epochs during training (Howard et al., 2017). In the experiments, however, a scheduler[3] from Sklearn was employed. Analogous to the metrics described in chapter 5.3.2, the validation error or AUROC calculated at each training step is recorded by the scheduler and checked to see whether the value has improved or worsened compared to the previous epoch. If the value has deteriorated five times in a row, the learning rate is decreased by dividing it by 0.1, while the rate has a lower bound of $0.1e-5$. This regularization technique is utilized for the fine-tuning approach.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html
[3] https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html

Chapter 6

Results

In this chapter, the results of the experiments are presented. In particular, visual graphs and tables are used to compare the classification of the transfer learning methods. The exact values of the performed experiments for each dataset can be found in the appendix chapters A, B, C, D, and E. The results for each research question introduced in chapter 1 are presented individually.

6.1 Adapter network design

The results considered in the adapter network design decisions according to **RQ 1** are presented. The resulting adapter network design is then employed for the remaining experiments and subsequently compared to other deep transfer learning approaches. Each network configuration is trained ten times and classified with the same test data. Finally, the average accuracy value is plotted together with the standard deviation.

6.1.1 Network input

This section compares the input variants for the adapter network described in section 3.2.3. For the analysis, the Indoor Scenes dataset is tested using ResNet-50. Experiments are carried out for both input variants employing a Two-Dense-Layer and a Three-Dense-Layer network with 512 neurons in the first and 64 in the second layer utilizing the end-to-end classification approach. As shown in the figure 6.1, network configurations with two fully-connected layers achieve the best classification performance across all samples per class compared to all other variants. It is also observed that for both network variants, using deep features as input achieves higher performance, regardless of the training size compared to the logits input. Logits and deep features perform comparably using the three-layer network only when more than 30 samples per class are available. The two-layer configuration with deep features performs best across all training variables, as seen in table 6.1.

Table 6.1: LOGITS AND DEEP FEATURES INPUT PERFORMANCE. End-to-end classification performance utilizing deep features and logits as adapter network input employing ResNet-50 as a pre-trained network and a two or three-dense layer adapter network is used on the Indoor Scenes dataset.

	1	**2**	**5**	**48**	**67**
Features-512	$\mathbf{0.274 \pm 0.015}$	$\mathbf{0.391 \pm 0.013}$	$\mathbf{0.52 \pm 0.01}$	$\mathbf{0.698 \pm 0.005}$	$\mathbf{0.714 \pm 0.01}$
Features-512-64	0.204 ± 0.015	0.301 ± 0.015	0.387 ± 0.017	0.618 ± 0.011	0.65 ± 0.015
Logits-512	0.22 ± 0.013	0.319 ± 0.016	0.45 ± 0.013	0.664 ± 0.004	0.681 ± 0.007
Logits-512-64	0.134 ± 0.021	0.22 ± 0.01	0.351 ± 0.012	0.615 ± 0.011	0.638 ± 0.013

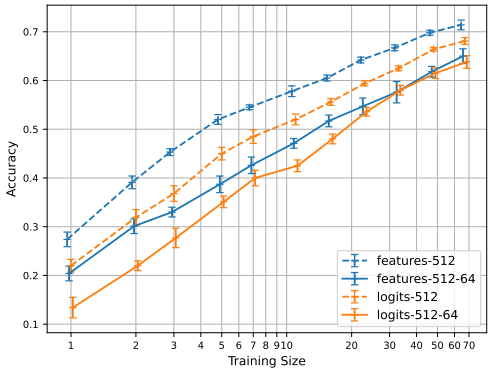


Figure 6.1: RESNET-50 CLASSIFICATION PERFORMANCE LOGITS AND DEEP FEATURES INPUT. This figure illustrates the end-to-end classification accuracy on the Indoor Scenes dataset with increasing image samples per class. Experiments are carried out with deep features (blue) and logits (orange) as adapter network input. A Two-Dense-Layer network (dashed) and a Three-Dense-Layer network (solid) with 512 neurons in the first and 64 in the second layer are employed.

6.1.2 Early stopping

Herein, the early stopping metrics presented in chapter 5.3.2 are compared. Due to resource constraints, only ResNet-50 pre-trained with ImageNet and the two fully-connected layers adapter network with 512 respective outputs are used for the analysis. According to chapter 3.1 classification, the nearest neighbor classification with the COS approach is selected for the image classification of the Indoor Scenes dataset. In order to better analyze the accuracy differences compared to the method without early stopping, where the training comprises 100 epochs, the values with early stopping are subtracted from the baseline values. The evaluation figure 6.2 shows that the validation metric yields a slightly better classification performance than AUROC tested with three different thresholds (15, 20, 30). The thresholds were selected empirically during the experiments. It can also be seen that with fewer samples per class, the early stopping method with validation error performs better, but with more than 20 samples per class, no early stopping technique seems more efficient. However, since the focus is on the few samples per class, an early stopping criterion with validation error is used for all adapter networks in further experiments. A snapshot of the classification performance can be seen in the table 6.2

Table 6.2: EARLY STOPPING PERFORMANCE. Classification performance with early stopping techniques using ResNet-50 and adapter network on Indoor Scenes dataset employing nearest neighbor classification with the COS approach.

	1	**2**	**5**	**11**	**67**
100 epochs	0.278 ± 0.01	0.392 ± 0.009	0.513 ± 0.014	0.567 ± 0.011	0.715 ± 0.008
AUROC-15	0.258 ± 0.023	0.347 ± 0.018	0.411 ± 0.046	0.49 ± 0.033	0.667 ± 0.009
AUROC-20	$\mathbf{0.279 \pm 0.011}$	0.387 ± 0.014	0.469 ± 0.025	0.552 ± 0.009	0.689 ± 0.009
AUROC-30	0.272 ± 0.013	$\mathbf{0.396 \pm 0.012}$	0.518 ± 0.012	0.57 ± 0.008	0.71 ± 0.008
Validation loss-15	$\mathbf{0.279 \pm 0.013}$	$\mathbf{0.396 \pm 0.011}$	$\mathbf{0.519 \pm 0.01}$	$\mathbf{0.576 \pm 0.006}$	$\mathbf{0.716 \pm 0.007}$

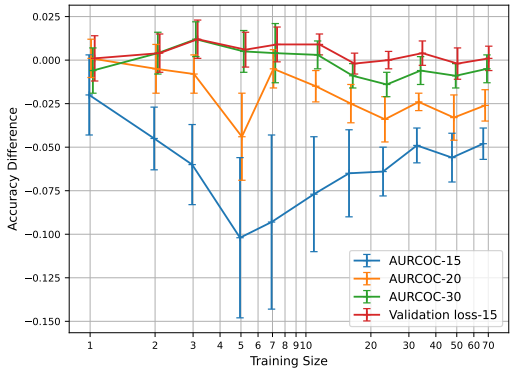


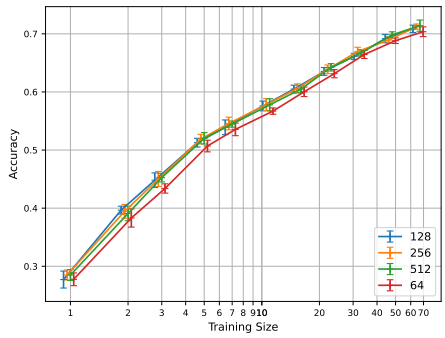
Figure 6.2: RESNET-50 CLASSIFICATION PERFORMANCE WITH EARLY STOPPING. This figure illustrates the nearest neighbor classification accuracy employing the COS approach on the Indoor Scenes dataset with increasing image samples per class testing the early stopping metrics presented in chapter 5.3.2. The accuracy values are subtracted by the accuracy values obtained with 100 training epochs. In the case of AUROC three different threshold are used: 15 (blue), 20 (orange), and 30 (green). The validation error metric is performed with a threshold counter of 15 (red).

6.1.3 Layer design

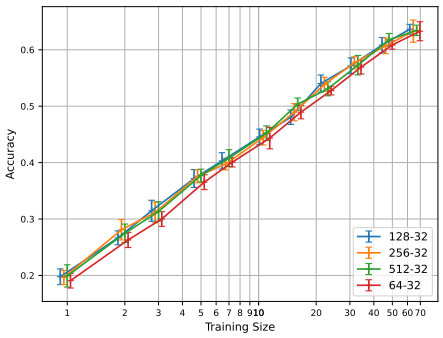
In order to determine the number of layers and neurons of the first two layers of the adapter network, which have a relatively better accuracy performance compared to other configurations, four different plots are created as shown in the figure 6.3. Analogous to the determination of the early stop metric, the pre-trained ResNet-50 and the Indoor Scenes dataset are used, but this time with end-to-end classification and early stop with validation error threshold at 15 counts. The first plot 6.3(a) shows four different neuron outputs (64, 128, 256, 512) of the first layer when only two fully-connected layers are applied. The performance increase with more neurons is the same for all configurations, with only 64 neurons showing a slightly reduced performance compared to the others. An analogous behavior is also observed with three fully-connected layers illustrated in plots 6.3(b) and 6.3(c). In plot 6.3(d), the network with two fully-connected layers is compared to networks with three fully-connected layers with the same number of neurons in the first output layer but different configurations in the second output layer. The two-layer network shows better accuracy across all sample sizes per class, as seen in table 6.3. For this reason, the two-layer network with 512 output neurons is chosen for all following experiments.

Table 6.3: TWO- VS. THREE-DENSE LAYERS PERFORMANCE. End-to-end classification performance using ResNet-50 and adapter network with either two or three fully-connected layers on Indoor Scenes dataset.

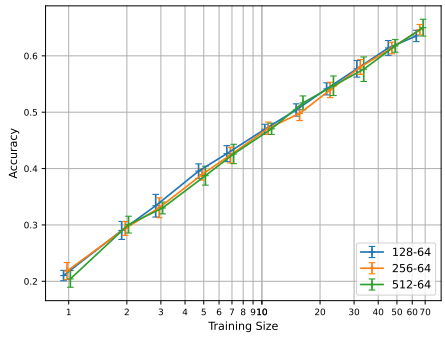
	1	**2**	**5**	**11**	**67**
512	$\mathbf{0.285 \pm 0.009}$	$\mathbf{0.391 \pm 0.013}$	$\mathbf{0.52 \pm 0.01}$	$\mathbf{0.578 \pm 0.011}$	$\mathbf{0.714 \pm 0.01}$
512-64	0.204 ± 0.015	0.301 ± 0.015	0.387 ± 0.017	0.471 ± 0.01	0.65 ± 0.015
512-32	0.199 ± 0.02	0.275 ± 0.016	0.377 ± 0.011	0.452 ± 0.013	0.634 ± 0.009
512-256	0.224 ± 0.019	0.313 ± 0.02	0.406 ± 0.013	0.481 ± 0.008	0.653 ± 0.016



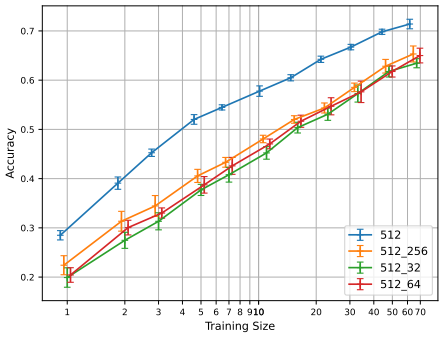
(a) Two-Dense-Layers



(b) Three-Dense-Layers with 32 output neurons in second layer



(c) Three-Dense-Layers with 64 output neurons in second layer



(d) Two- and Three-Dense-Layers with 512 output neurons in first layer

Figure 6.3: ADPATIVE NETWORK LAYER COMPARISON. This figure shows the classification performance using ResNet-50 on the Indoor Scenes dataset changing the adapter network structure to (a) Two-Dense-Layers with variable number of outputs in the first layer, Three-Dense-Layers with either (b) 32 or (c) 64 output neurons in second layer and (d) Two- and Three-Dense-Layers with 512 output neurons in first layer.

6.2 Pre-trained network comparison

According to **RQ 2**, the aim is to determine whether all five pre-trained networks with ImageNet are equally capable of producing sufficiently descriptive deep features. Thus, representative galleries specifically adapted to the target dataset should be enrolled to achieve an accurate image classification. The experiments performed include nearest neighbor classification with the MEAN approach. Similar to other experiments, the classification was repeated with increasing sample sizes per class. As depicted in figure 6.4, AlexNet shows lower performance, especially on the Aircraft, Fruit and Vegetable, Indoor Scenes, and Office-31 datasets. This is not only observed for one sample per class but also gradually when more samples per class are available. The accuracy performance does not reach the performance of the other networks investigated. The classification of the Virus dataset proves to be more effective than for other networks when 1 sample per class is used. In the progression, however, the accuracy improves less strongly in comparison. To

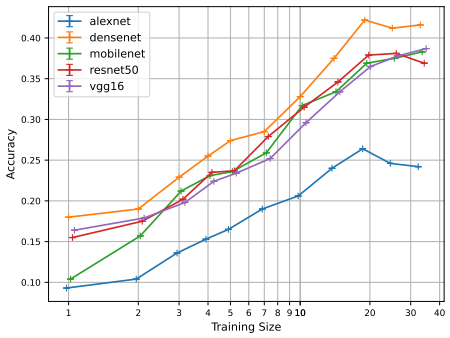
put this result in context with the published network ImageNet classification performances[1], the absolute top-1-accuracy differences between AlexNet and the remaining examined networks are depicted in table 6.4. To complement this, table 6.5 shows the relative ratio of the difference between the accuracy values measured in the experiment and the published values classified with ImageNet between AlexNet and ResNet-50, as specified in equation (6.1). It can be seen that the absolute classification difference of 19.608% between ResNet-50 and AlexNet is only reached for the Indoor Scenes dataset between three and nine samples per class. However, almost for all differences, more than one-third of this amount is found, demonstrating the constant superior performance of ResNet-50 compared to AlexNet. For this reason, AlexNet is not included in further experiments, especially because the other networks demonstrate similar classification values. The Virus data set seems to be an exception because, in contrast to other datasets, MobileNet-V3 offers up to 10% better performance when more samples per class are available. AlexNet and VGG-16 can still produce similar results with one sample per class but achieve performance up to 5 % lower than ResNet-50 and DenseNet-121 as the number of samples increases. Overall, the performance loss is proportionally greater than the difference between the other networks, which justifies the exclusion of the network for the remaining experiments. Thus, only the performance results of MobileNet-V3, DenseNet-121, and ResNet-50 are evaluated in further experiments.

$$Acc_{diff} = \frac{Acc_{ResNet-50} - Acc_{Alexnet}}{19.608} \tag{6.1}$$

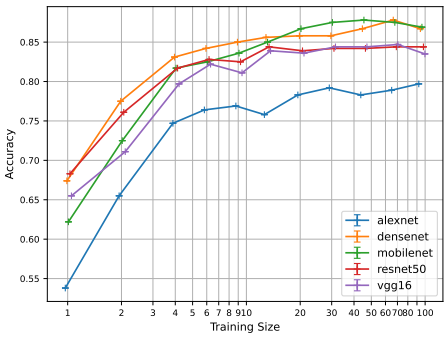
Table 6.4: ABSOLUTE NETWORK ACCURACY DIFFERENCE. This table illustrates the absolute accuracy difference in percent between AlexNet and VGG-16, ResNet-50, MobileNet-V3, and DenseNet-121 evaluated on ImageNet.[1]

	VGG-16	**ResNet-50**	**MobileNet-V3**	**DenseNet-121**
AlexNet	15.07	19.608	17.52	17.912

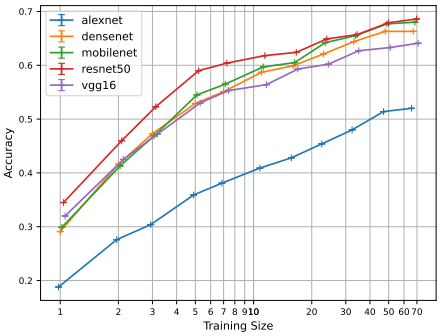
[1] https://pytorch.org/vision/stable/models.html



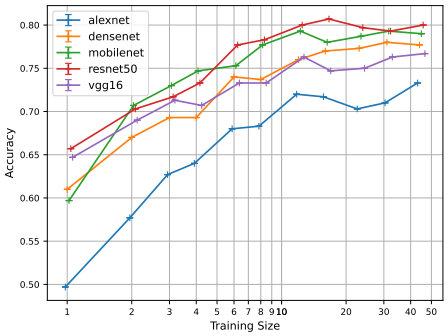
(a) Aircraft



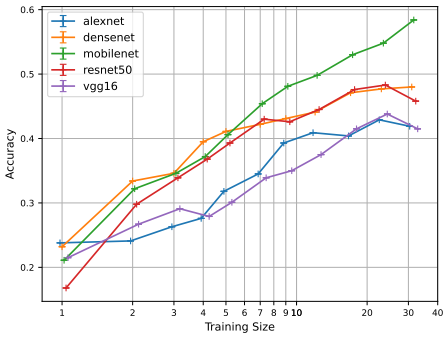
(b) Fruit and Vegetable



(c) Indoor Scenes



(d) Office-31



(e) Virus

Figure 6.4: PRE-TRAINED NETWORK COMPARISON. This figure shows the classification performance using nearest neighbor classification with the MEAN approach over all five examined pre-trained networks on the datasets: (a) Aircraft, (b) Fruit and Vegetable, (c) Indoor Scenes, (d) Office-31, and (e) Virus.

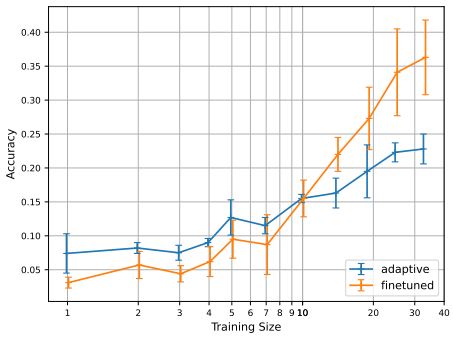
Table 6.5: RELATIVE ACCURACY RESNET-50 VS. ALEXNET. This table illustrates the relative ratio of the difference between the accuracy values measured in the experiment and the published accuracy values from table 2.1 classified with ImageNet between ResNet-50 and AlexNet as specified in equation (6.1). The columns represent the respective ratio using the number of image samples per class.

	1	**2**	**3**	**4**	**5**	**7**	**9**	**23**	**31**
Aircraft	0.316	0.364	0.334	0.416	0.368	0.458	0.553	0.687	0.645
Fruit and Vegetable	0.738	0.540	0.354	0.326	0.283	0.439	0.283	0.285	0.239
Indoor	0.800	0.938	1.113	1.177	1.136	1.067	1.000	0.843	0.848
Office-31	0.816	0.646	0.459	0.476	0.493	0.510	0.408	0.425	0.340
Virus	-0.356	0.291	0.390	0.469	0.380	0.434	0.170	0.276	0.199

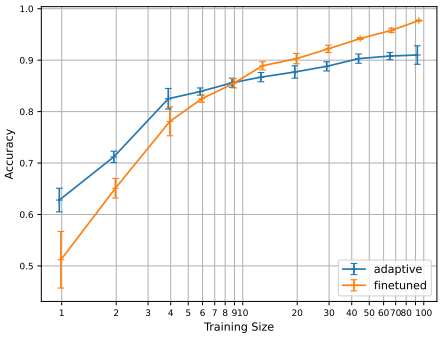
6.3 Approach comparison

This section compares the different transfer learning approaches according to **RQ 3**. The results of the four sub-analyses described in chapter 5 are discussed one after the other. First, the end-to-end classification performance of the fine-tuned and adapter network is compared. In figure 6.5 the results of each dataset are shown. It can be seen across all datasets that the adapter network classifies better than the fine-tuned network when 1-10 samples per class are available. The performance advantage of the adapter network decreases with progress, and for most datasets in the range of 10 samples per class, both transfer learning methods perform equivalently. The effect reverses for more than ten samples per class. More precisely, the fine-tuned network shows a steeper performance increase while it flattens out when the adapter network is used. The Virus dataset result is an exception to this observation, where the adapter network also performs better than the fine-tuned network, but the effect is not reversed. It can also be seen that the classification with the fine-tuned network, despite a clear tendency, partially suffers performance losses compared to the previous classification, especially with four samples per class. In table E.1 it can be seen that DenseNet-121 in particular is responsible for a decrease in performance.

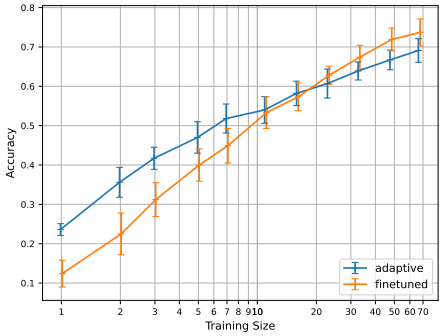
Second, figure 6.6 shows the comparison between pre-trained, adaptive, and fine-tuned using deep feature classification methods for each dataset. The results of the nearest neighbor classification methods using the COS and MEAN approach, as well as the results of the SVM approach, are considered. There is a similar tendency as in figure 6.5 where the fine-tuned network achieves a better classification in contrast to the pre-trained and the adapter network with more samples per class. The fine-tuned network also performs worse for the Fruit and Vegetable, Indoor Scenes, and Office-31 datasets with a few samples per class. In contrast to figure 6.5, the classification results for the remaining datasets are similar. The performance of the pre-trained network without training is comparable to that of the adapter network. However, across all networks, the results favor using the pre-trained network when fewer samples per class are available.



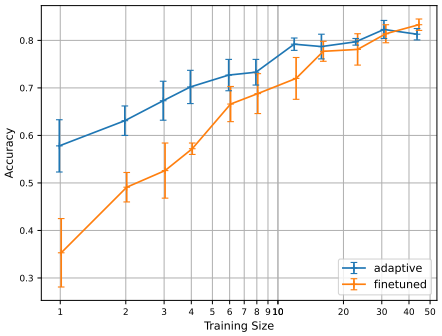
(a) Aircraft



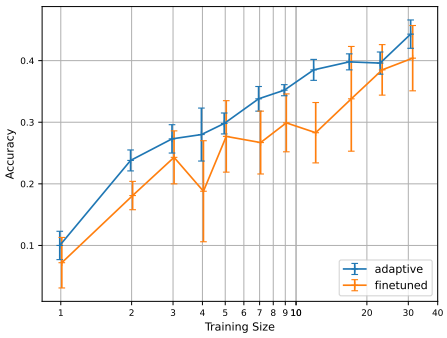
(b) Fruit and Vegetable



(c) Indoor Scenes

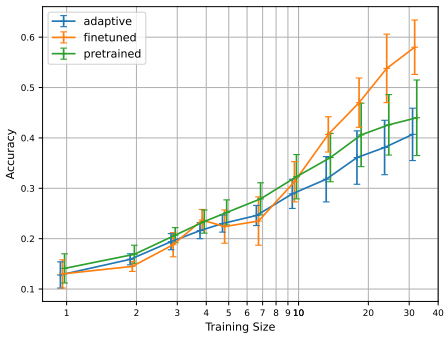


(d) Office-31

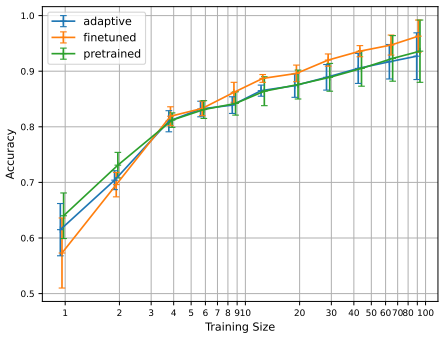


(e) Virus

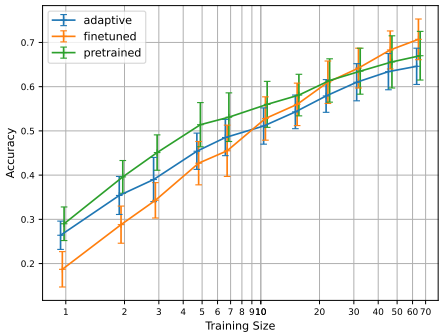
Figure 6.5: ADAPTIVE AND FINE-TUNED NETWORK COMPARISON. This figure shows the end-to-end classification performance using adapter and fine-tuned networks averaging over the results of ResNet-50, DenseNet-121 and MobileNet-V3 on the datasets: (a) Aircraft, (b) Fruit and Vegetable, (c) Indoor Scenes, (d) Office-31, and (e) Virus.



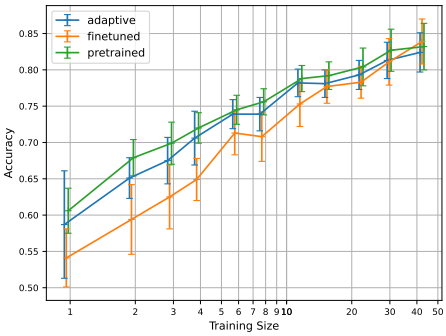
(a) Aircraft



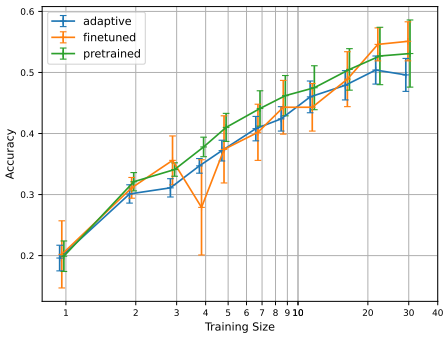
(b) Fruit and Vegetable



(c) Indoor Scenes



(d) Office-31

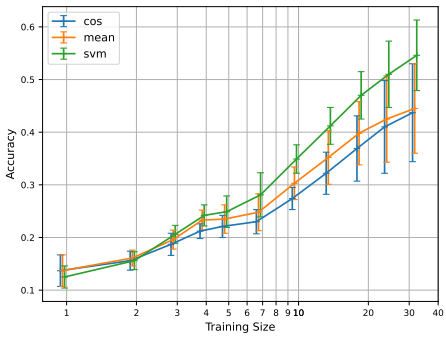


(e) Virus

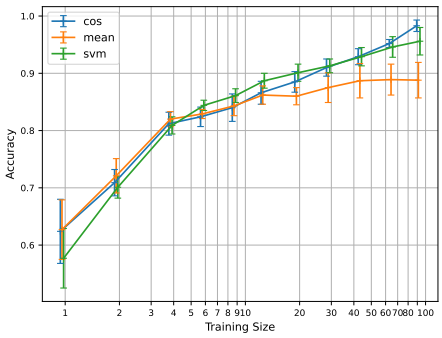
Figure 6.6: ADAPTIVE, FINE-TUNED AND PRE-TRAINED NETWORK COMPARISON. This figure shows the nearest neighbor classification with COS and MEAN and the SVM classification performance using adapter, fine-tuned and pre-trained networks averaging over the results of ResNet-50, DenseNet-121 and MobileNet-V3 on the datasets: (a) Aircraft, (b) Fruit and Vegetable, (c) Indoor Scenes, (d) Office-31, and (e) Virus.

Third, instead of distinguishing between pre-trained, fine-tuned, and adapter networks, the nearest neighbor classification with the COS and MEAN approach and SVM classification is compared. The results of the adapter, fine-tuned, and pre-trained network configurations of the ResNet-50, DenseNet-121, and MobileNet-V3 networks are considered. In figure 6.7, it can be seen that the SVM model displays a larger variance than the nearest neighbor classification with one sample per class, especially on the Fruit and Vegetable and Office-31 dataset. In addition, up to 5% inferior performance compared to the nearest neighbor classification approach can be seen. However, with more samples, SVM is advantageous over gallery enrollment. The COS approach begins to outperform the SVM results when more than 60 samples per class are used in the fruit and vegetable dataset. Concerning the difference between COS and MEAN, it can be seen that the accuracy values are identical for one sample per class. This is because the average of one feature per class yields the same feature again. The use of more samples shows a preference for the averaged gallery in the Aircraft and Indoor Scenes dataset. Both variants show comparable values for the Fruit and Vegetable and Office-31 dataset, but the MEAN variant flattens out more strongly with more than ten samples per class. In the Virus dataset, both classifications variants are almost identical.

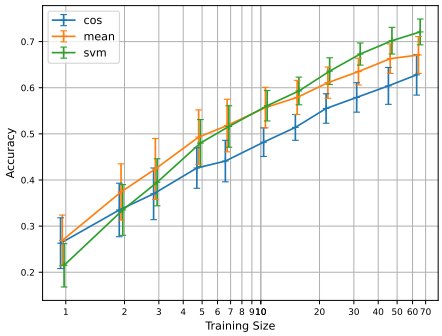
Fourth, herein best transfer learning approaches of the sub-analyses are summarized. From figure 6.5 it follows that the adapter network classifies better than the fine-tuned network when using a few samples. For more samples, the effect is reversed. When utilizing the networks as feature extractors, figure 6.6 shows that the pre-trained network operates most reliably over the five datasets compared to the adapter and fine-tuned network. Finally, for figure 6.7, the MEAN approach performs equally well or even better (in the case of the Indoor Scenes dataset) than the COS approach. The SVM model shows inferior accuracy with few samples compared to the nearest neighbor classification but achieves a better performance with more samples. Thus, in figure 6.8 the adaptive network is represented through end-to-end classification and a deep feature extractor for enrollment of the averaged gallery. In addition, the averaged gallery samples extracted from the pre-trained network and the fine-tuned network classified with the softmax layer and with the SVM model are also shown in the graph. It can be seen that for the Aircraft and Virus dataset, the adaptive network with end-to-end classification performs up to 10% worse than approaches with deep feature classification. For the remaining datasets, the performance improves more effectively when using an increasing amount of samples per class. Regarding the datasets Fruit and Vegetable and Indoor Scenes, it is even better with more samples than the nearest neighbor classification. If the methods are only considered on their classification results in up to ten samples per class, then the pre-trained network with MEAN is the best approach. With more samples, the advantage of using the fine-tuned network combined with the SVM classifier becomes particularly apparent. Across all datasets, the fine-tuned network with end-to-end classification achieves the lowest accuracy with few samples. The result can be improved when the network serves as a feature extractor combined with the SVM model. Generally, the results indicate that feature extraction methods are equal or superior in comparison with end-to-end classification methods when using few samples per class.



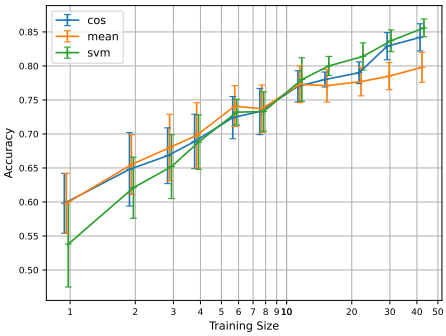
(a) Aircraft



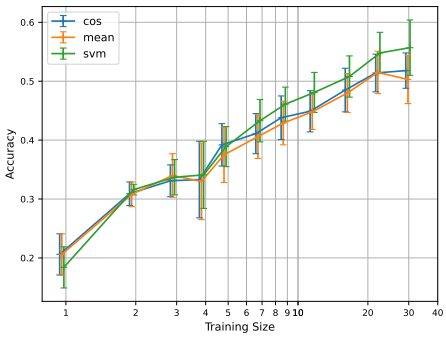
(b) Fruit and Vegetable



(c) Indoor Scenes

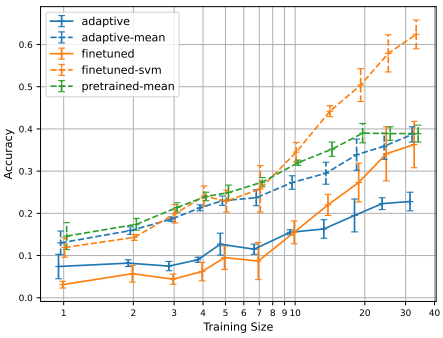


(d) Office-31

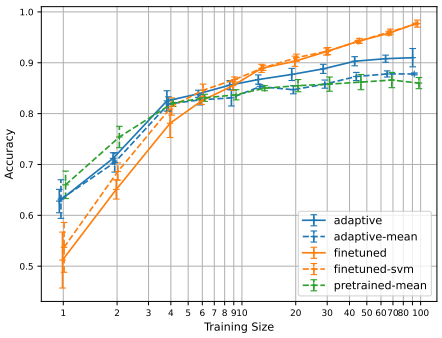


(e) Virus

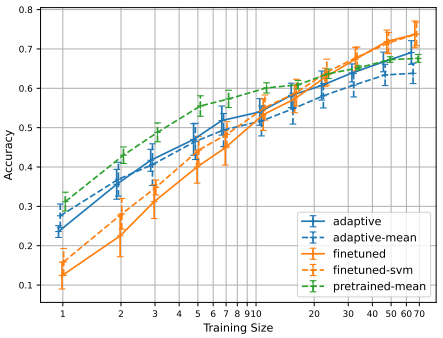
Figure 6.7: COS VS. MEAN VS. SVM CLASSIFICATION. This figure shows the classification performance comparing the COS and MEAN nearest neighbor classification approach and the SVM classification results averaged over adaptive, fine-tuned, and pre-trained network configurations averaging over the results of ResNet-50, DenseNet-121, and MobileNet-V3 on the datasets: (a) Aircraft, (b) Fruit and Vegetable, (c) Indoor Scenes, (d) Office-31, and (e) Virus.



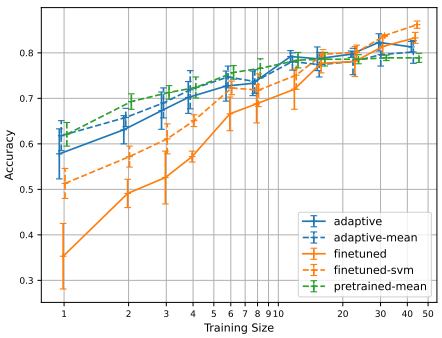
(a) Aircraft



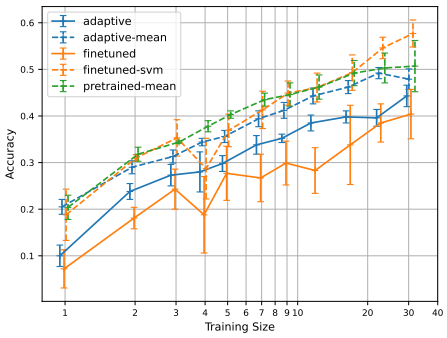
(b) Fruit and Vegetable



(c) Indoor Scenes



(d) Office-31

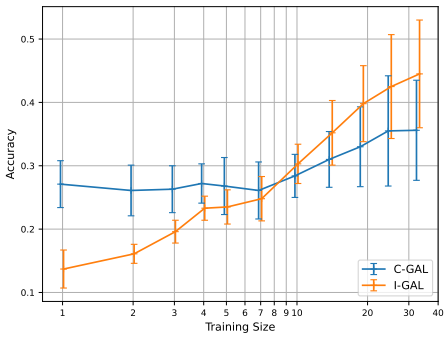


(e) Virus

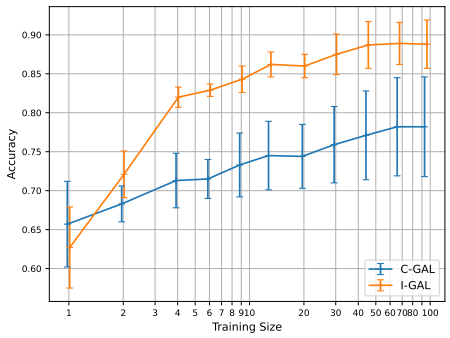
Figure 6.8: BEST APPROACHES COMPARISON. This figure shows nearest neighbor classification with MEAN (dashed), the SVM classification (dashed) and the end-to-end classification (solid) results using the adapter (blue), fine-tuned (orange) and pre-trained (green) networks averaging over the results of ResNet-50, DenseNet-121 and MobileNet-V3 on the datasets: (a) Aircraft, (b) Fruit and Vegetable, (c) Indoor Scenes, (d) Office-31, and (e) Virus.

6.4 Gallery comparison

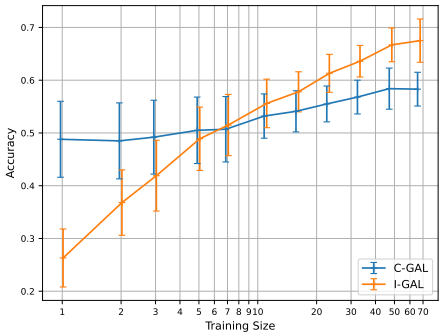
In this section, the robustness of the feature gallery in the cosine similarity approach according to **RQ 5** is investigated. Thus, all network variants according to the chapter 3.2 and the pre-trained networks ResNet-50, DenseNet-121, and MobileNet-V3 are taken into account. The accuracy values are determined by the nearest neighbor classification with the MEAN approach. Therefore, the average sample representing the class in the gallery is calculated either with five samples per class (C-GAL) or all possible training samples (I-GAL). The differences are shown in figure 6.9. C-GAL achieves higher accuracy values in most datasets when using 1 to 5 samples per class. An exception to this observation can be found in the Fruit and Vegetable dataset, where C-GAL already achieves a 2% lower accuracy value with two samples per class. A maximum performance increase of 10% in the case of C-GAL when applying more training samples is observed. The improvement of the classification using I-GAL shows a much higher increase in comparison. It is apparent when more than ten samples per class are available. The Virus dataset shows another variant of this rule since, in this case, the C-GAL achieves a better classification until just before applying 20 samples per class. In the case of the Aircraft dataset, C-GAL outperforms I-GAL up to seven samples per class.



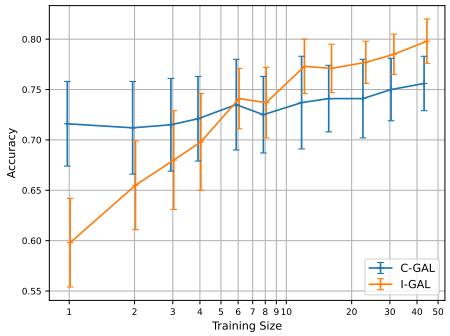
(a) Aircraft



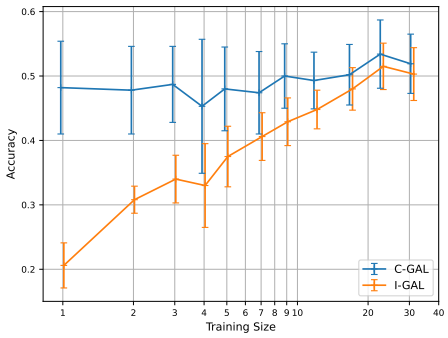
(b) Fruit and Vegetable



(c) Indoor Scenes



(d) Office-31



(e) Virus

Figure 6.9: CONSTANT GALLERY (C-GAL) VS. INCREASING GALLERY (I-GAL). This figure shows the nearest neighbor classification performance with the MEAN approach using five samples per class as gallery representation, which are not part of the training (C-GAL) compared to the increasing gallery samples per class (I-GAL). The adaptive, fine-tuned and pre-trained networks are considered averaged over ResNet-50, DenseNet-121 and MobileNet-V3 on the datasets: (a) Aircraft, (b) Fruit and Vegetable, (c) Indoor Scenes, (d) Office-31, and (e) Virus.

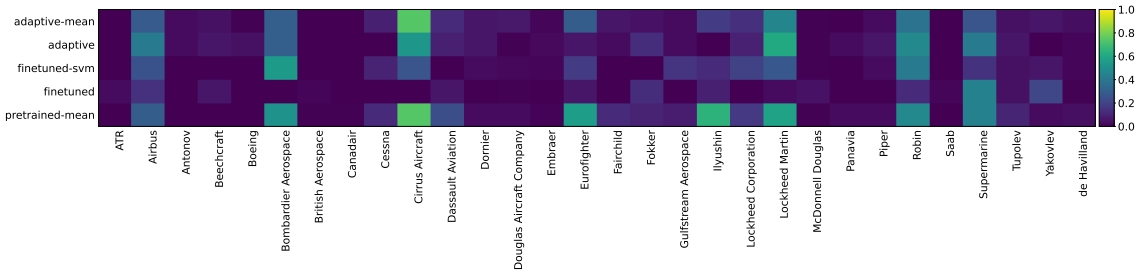
6.5 Class performance

In the previous chapters, only the overall performance of a transfer learning approach is analyzed. For this reason, the individual class performance is discussed to identify the effectiveness of the descriptive features indirectly. For each of the five datasets, figures 6.10, 6.11, 6.12, 6.13, and 6.14 show the ground truth class performance of the best transfer learning approaches identified by figure 6.8. Due to comparatively best performance, DenseNet-121 is chosen for the first two and the last dataset, ResNet-50, for each of the remaining datasets. Additionally, the confusion matrix adopting the nearest neighbor classification with the MEAN approach and the pre-trained network with three different sample sizes is shown in the appendix for each dataset (Aircraft: A.1, A.2, A.3; Fruit and Vegetable: B.1, B.2, B.3; Indoor Scenes: C.1, C.2, C.3; Office-31: D.1, D.2, D.3; Virus: E.1, E.2, E.3). The results are shown three times in the range of 1 - 12 samples per class to indicate the classification performance when more samples are used per class.

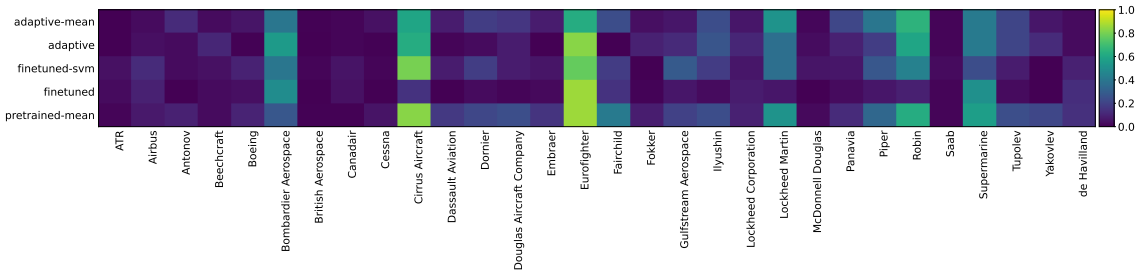
Regarding the aircraft dataset, one sample per class shows that especially the classes *Airbus*, *Bombadier Aerospace*, *Cirrus Aircraft*, *Eurofighter*, *Lockheed Martin*, *Robin*, *Supermarine* attain more than 40% accuracy for all transfer learning approaches used. However, when the finetuned network with softmax classification is applied, only *Supermarine* reaches this value. The remaining classes are not distinguishable for any network. In particular, the classes *Airbus*, *Bombadier Aerospace*, *Ilyushin* are incorrectly classified across all approaches. When three samples per class are used, *Airbus* loses performance across all transfer learning approaches, with *Eurofighter* achieving over 30% accuracy across the board. Even using ten samples per class, there are still classes that achieve less than 10% accuracy. It can still be observed that the class *Bombardier Aerospace* is classified comparatively frequently with the pre-trained network approach.

Using the pre-trained network approach with one sample per class on the Fruit and Vegetable dataset, ten classes already achieve over 90% accuracy. Most of these classes, such as *Pinapple*, *Cabbage*, *Lemon*, *Cauliflower*, *Pomegranate* also occur in ImageNet. Correspondingly, these classes are correctly identified by applying other approaches by over 50%. In contrast, the classes *Apple*, *Corn*, *Mango*, *Jalepeno*, *Paprika* are categorized below 20%. In particular, it can be seen that the classes *Corn* and *Sweetcorn* classify the other class in case of misclassification. In progress, the classification results of the other approaches improve compared to the pre-trained network approach. The already visible tendency that transfer learning cannot discriminate the mentioned classes based on the features with one sample per class continues with more samples per class. However, more than half of the class approaches achieve above 90% accuracy with ten samples.

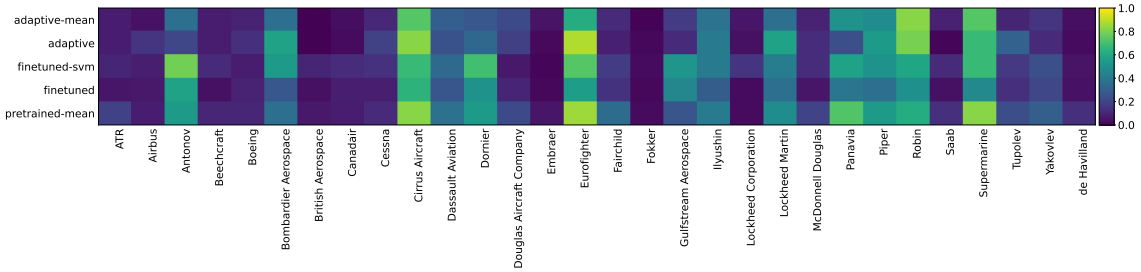
Referring to the Indoor Scenes dataset, the pre-trained network approach allows classifying seven classes with over 70% accuracy. Analogous to the Fruit and Vegetable dataset, misclassification of the classes *Movie theater* and *Auditorium*, as well as *Library* and *Bookstore* and others, occur, erroneously predicting the respective other class. This effect is not as pronounced as with the dataset mentioned above since other classes are also incorrectly classified in addition to the classes mentioned. Similarly, when classifying with one sample per class, instead of predicting the true class *Cloister*, *Church Inside* is predicted to a larger extent. The effect is no longer visible with three samples per class, whereas only 1 sample is assigned to the wrong class. The adaptive network approaches achieve comparable accuracy results compared to the pre-trained approach. On the other hand, fine-tuned network approaches show more classes with less than 20% accuracy. In contrast, comparable performance is achieved with more samples per class, especially when using the fine-tuned network approach with SVM. The classes with the lowest performance results also show the worst values across all approaches.



(a) 1 sample per class

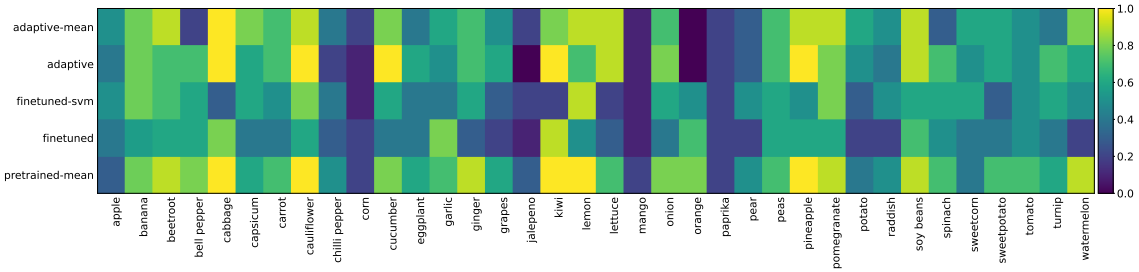


(b) 3 samples per class

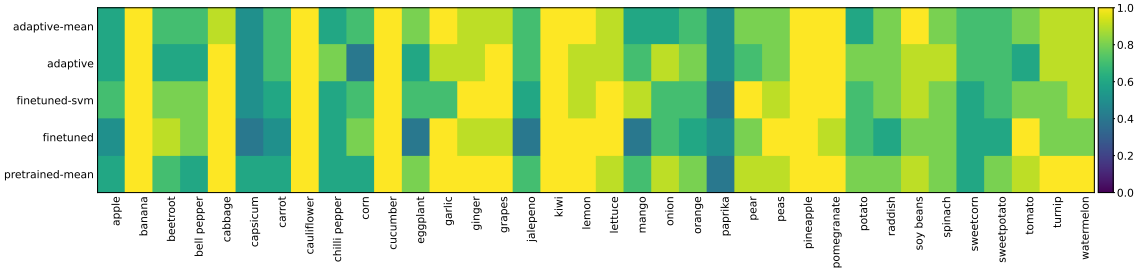


(c) 10 samples per class

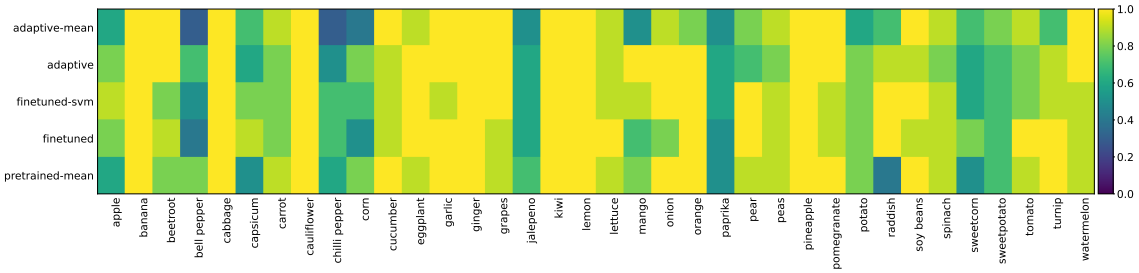
Figure 6.10: GROUND TRUTH CLASS PERFORMANCE AIRCRAFT. This figure shows the true class classification results for each class utilizing the best transfer learning approaches with DenseNet-121 identified in RQ 3 on the Aircraft dataset using (a) 1 (b) 3 (c) 10 samples per class.



(a) 1 sample per class

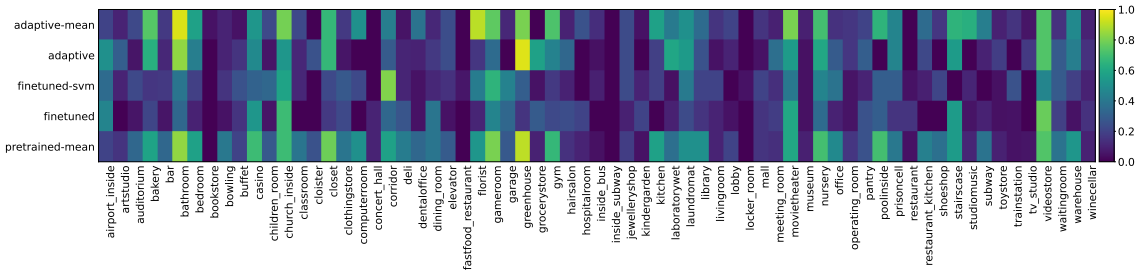


(b) 4 samples per class

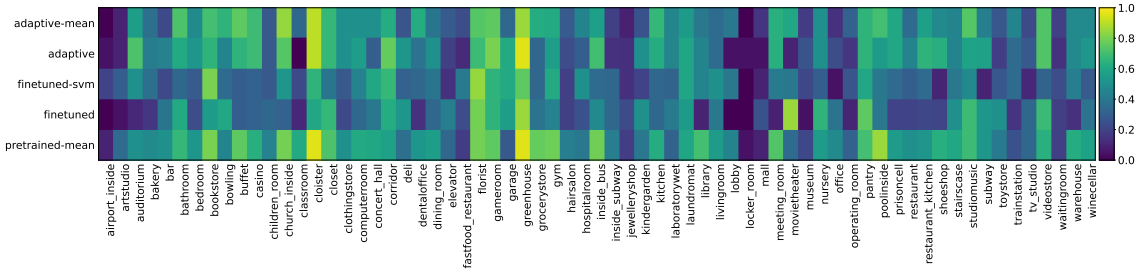


(c) 9 samples per class

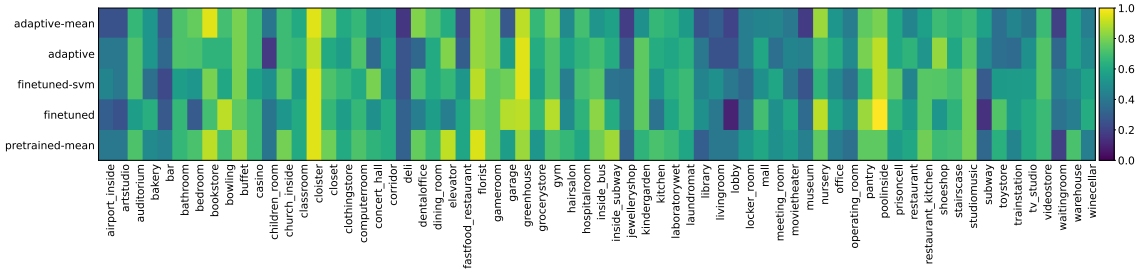
Figure 6.11: GROUND TRUTH CLASS PERFORMANCE FRUIT AND VEGETABLE. This figure shows the true class classification results for each class utilizing the best transfer learning approaches with DenseNet-121 identified in RQ 3 on the Fruit and Vegetable dataset using (a) 1 (b) 4 (c) 9 samples per class.



(a) 1 sample per class



(b) 3 samples per class

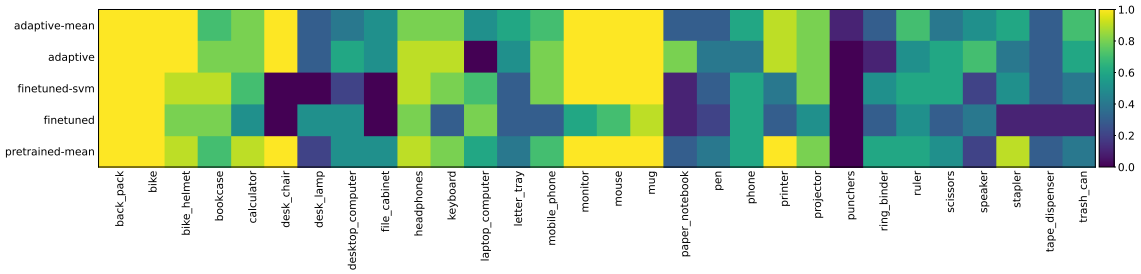


(c) 11 samples per class

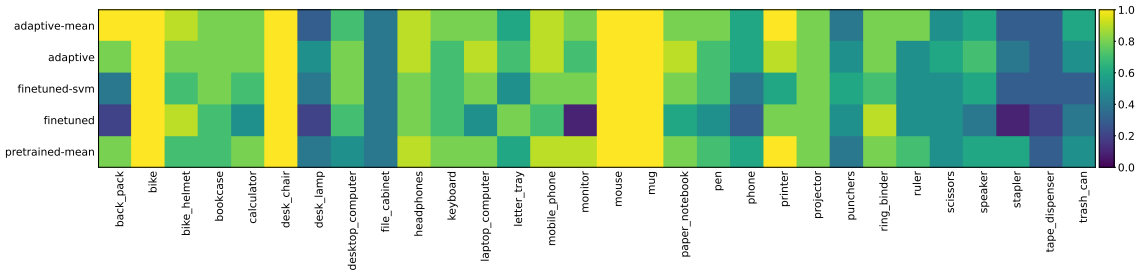
Figure 6.12: GROUND TRUTH CLASS PERFORMANCE INDOOR SCENES. This figure shows the true class classification results for each class utilizing the best transfer learning approaches with ResNet-50 identified in RQ 3 on the Indoor Scenes dataset using (a) 1 (b) 3 (c) 11 samples per class.

Using one sample per class on the Office-31 dataset, one-third of the classes achieved over 90% accuracy. These values can be observed across all transfer learning approaches. Only the fine-tuned network approaches yield more classified samples under 20% accuracy. The class *Punchers* cannot be discriminated by all approaches. Using three samples per class improves the overall performance, but the number of correctly classified classes above 90% decreases to 6 for the adaptive network with similarity distance classification and 5 for the pre-trained network approach. Except for the fine-tuned network approach with softmax classification, an accuracy of more than 20% is attained for all classes. However, the classification differences between the approaches diminish with 12 samples per class, resulting in no classification under 20% for all classes. Analogous to the other datasets, the relatively lower discriminated classes, i.e., *Punchers*, *File Cabinet* also perform less effectively across all approaches with more samples per class.

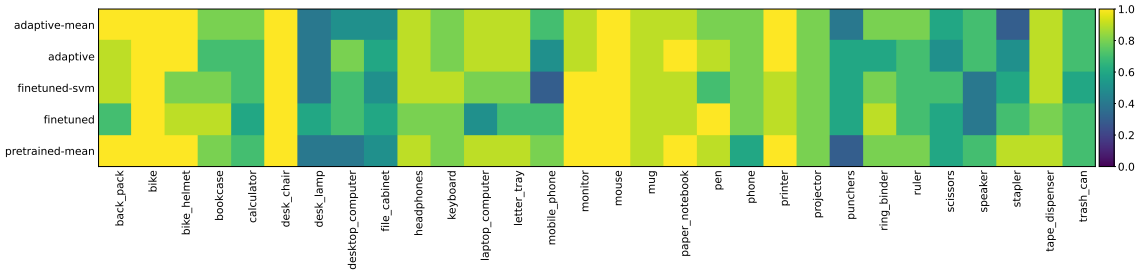
The pre-trained network approach is the most precise for the virus dataset for the class *Pseudocowpox*, achieving an accuracy of more than 90%. In contrast, the fine-tuned network approach with softmax classification achieves less than 20% accuracy for three-quarters of all classes. With three samples per class, the adaptive network approach for the class *Pailloma* can achieve a performance above 90%. In contrast, the fine-tuned network with SVM is the only approach to accomplish a similar performance for class *WestNile*. When using more samples, the mentioned class approaches achieve similar classification values. Solely the fine-tuned approach is not able to correctly classify the class *Papilloma*. Analysis of the confusion matrix of the pre-trained network approach shows that although the features allow pronounced discrimination, there is still a higher dispersion of predicted classes with nine samples per class than in the datasets already described.



(a) 1 sample per class

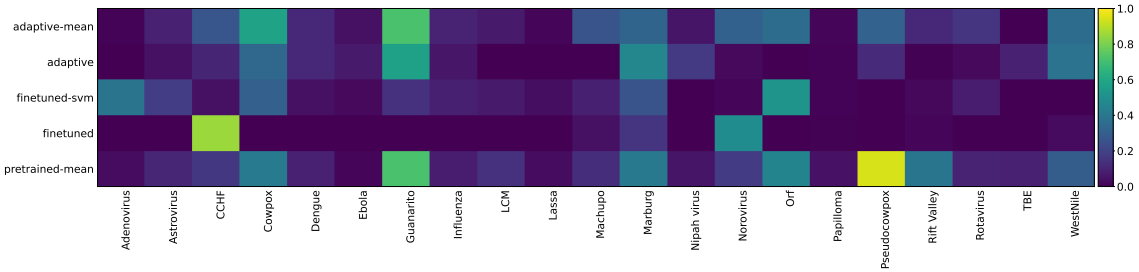


(b) 3 samples per class

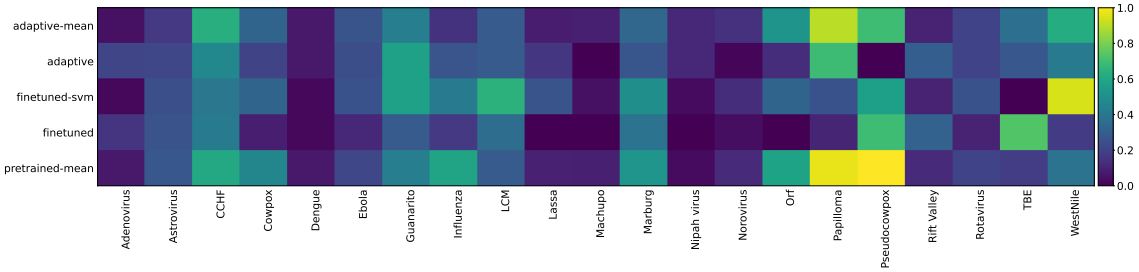


(c) 12 samples per class

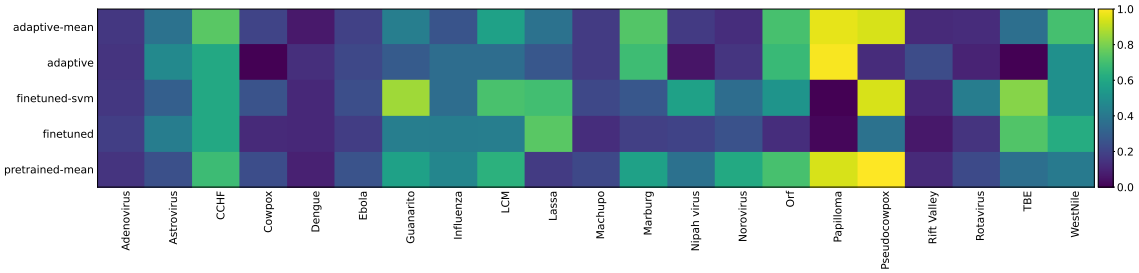
Figure 6.13: GROUND TRUTH CLASS PERFORMANCE OFFICE-31. This figure shows the true class classification results for each class utilizing the best transfer learning approaches with ResNet-50 identified in RQ 3 on the Office-31 dataset using (a) 1 (b) 3 (c) 12 samples per class.



(a) 1 sample per class



(b) 3 samples per class



(c) 9 samples per class

Figure 6.14: GROUND TRUTH CLASS PERFORMANCE VIRUS. This figure shows the true class classification results for each class utilizing the best transfer learning approaches with DenseNet-121 identified in RQ 3 on the Virus dataset using (a) 1 (b) 3 (c) 9 samples per class.

Chapter 7

Discussion

In the following, the investigated transfer learning approaches for multi-classification image tasks for the presented five databases with limited data availability will be highlighted based on the results in chapter 6. In addition, it is also essential to identify which pre-trained networks should be preferred for transfer learning regardless of the approach. In addition, it is discussed whether it is appropriate to create a feature gallery with or without training examples. The results of the adapter network design according to **RQ 1** are described subsequently.

First, the results indicate that deep features as input for the adapter network should be preferred, which was also why this approach was adopted in the remaining experiments. The result contrasts the adapter network configuration used in the work of Günther et al. (2020) where the logits are used. Still, as Yosinski et al. (2014) argues, features from earlier layers are not yet as strongly adapted to the source dataset. This may indicate better performance since parameters from a prior layer are used. Another reason for the performance difference could be the varying output dimensions. The logits comprise 1,000 outputs compared to 2,048 when using the deep features of the pre-trained ResNet-50. The deep features have twice as much information and can pass it on to the adapter network. Since this performance advantage can only be observed for the Indoor Scenes dataset, it would be interesting to see how the deep features input would compare to the logits input with less overlapping target and source datasets. Also, a combination analogous to the technique suggested by Cıbuk et al. (2019) would be an option. Regarding early stopping, there were only marginal differences in performance as shown in figure 6. AUROC, with a threshold of 15, clearly stopped the training too early, especially when more samples per class were available. The use of AUROC would theoretically help the discrimination of deep features because if the features used for the gallery are most clearly distinguishable from other features, then more meaningful class representations should be available when enrolling in the gallery. In practice, however, AUROC is very sensitive to parameter changes in the network. Increasing the threshold from 15 to 30 improves performance by considering sensitivity. The validation error proved a more robust value in this case and was therefore used in all other experiments. However, it does not seem suitable, as no significant performance increase is achieved. It remains questionable whether early stopping is even helpful in this context or whether a more meaningful metric, such as validation accuracy, should be applied. Regarding the number of fully-connected layers in the adapter network, the two-layer network showed a performance advantage with only a few samples. Interestingly, fewer model parameters must be trained in the three-layer network when 512 neurons are used in the first layer and 16 neurons in the hidden layer. In order to minimize overfitting, the number of model parameters should be kept as small as possible when training with limited available training data. The design decision deviates from the adapter architecture proposed by Günther et al. (2020), where a hidden layer with 64 neurons was used. On the other hand, Maqsood et al. (2019) used adaptation layers without a hidden layer after the five convo-

lutional layers of AlexNet. The usage of only one new softmax classifier shows, as evident in the experiments of Zhu et al. (2021), that this architecture is not complex enough for accurate classification. Similarly, 64 neurons in the first layer of the adapter network are too small to classify image classes to the same extent as other configurations, resulting in underfitting. The first layer's modifications with 128, 256, and 512 neurons show no relevant performance difference. For the subsequent experiments, 512 neurons were chosen because Aircraft and Virus datasets have fewer overlaps with ImageNet than Indoor Scenes. Therefore, the classification can be assumed to require a more complex network. After running each configuration ten times, it can be seen that the standard deviation is a maximum of 0.02%. This occurs mainly when few training samples are used. With more samples, typically, fewer deviations are observed. A specific configuration was performed once for the further experiments because it can be assumed that comparable standard deviations occur.

Secondly, **RQ 2** is addressed in the following paragraph. Initially, the analysis of Kornblith et al. (2019) concerning the influence of ImageNet top-1 accuracy values is relevant. According to the table 2.1, the value of ResNet-50 is about 2% better than the next best networks DenseNet-121 and MobileNet-V3. However, this better performance can only be observed for the Indoor Scenes dataset in figure 6.4. No clear network preference can be determined regarding the Fruit and Vegetable and Office-31 datasets. The variation of the performance results can also be due to the shuffled training data, the data augmentation, and the stochastic gradient descent. For the Aircraft and Virus datasets, whose classes are more difficult to differentiate than the others, one can see stronger performance variabilities than for the others. In the former, ImageNet also has aircraft classes, but the classes are less fine-grained, and in the latter, the microscopic image data do not occur within the ImageNet classes. In general, however, the conclusion of (Kornblith et al., 2019) that top-1-accuracy correlates with transfer learning performance can be reinforced. VGG-16 and AlexNet perform worse than the rest of the networks across all datasets, whereas the absolute differences of table 6.4 only indicate the performance trend. Although the performance progression of VGG-16 according to figure 6.4 is similar to that of MobileNet-V3, DenseNet-121 and ResNet-50, it can be seen from the tables A.1, B.1, C.1, D.1, E.1 that the network using the adaptive approach is especially not able no classify a comparable amount of correct samples. VGG-16s performance difference compared to other networks is especially inferior when considering the results conducted on the Virus dataset when using 1 sample per class. Despite the fact that this trend becomes less prominent when more samples per class are used, it decreases the overall performance of the adaptive network, which is why VGG-16 was not employed for other experiments. It is noticeable that the networks, according to table 2.1 have fewer convolutional layers and a larger deep features dimension. It should also be kept in mind that for target datasets with a less pronounced domain overlap with ImageNet, the transfer learning performance may not correlate with the top-1-accuracy scores. This could also explain the better performance of the virus dataset with MobileNet-V3 and the Aircraft dataset with DenseNet-121 compared to ResNet-50, especially when more samples per class are available. The findings of (Zhu et al., 2021) concerning the influence of the number of model parameters of the pre-trained network can only be compared to a limited extent in these experiments since the same network structures are not compared with different parameter sizes. However, it can be seen that the networks with the most parameters according to table 2.1 perform worst in these experiments. Drawing this conclusion on the number of parameters would be wrong because it must also be mentioned that the other networks are further developments of AlexNet and VGG-16. The structural optimizations in the network and thus their possibility of more convolutional layers have led to better results. They should therefore be preferred as a pre-trained network for deep transfer learning approaches with limited data availability.

Thirdly, this section discusses the results of **RQ 3** and **RQ 4**. The comparison between the adapter network and the fine-tuned network shows that the adapter network is superior to the fine-tuned network approach for transfer learning tasks with 1-10 samples. This can be especially because, with the fine-tuned network, all parameters of the pre-trained network have to be adjusted. In contrast, the pre-trained network is not changed in the adapter network approach. In this case, relatively few layers have to be trained. As an example for ResNet-50 trained on the Indoor Scenes dataset, the training for the fine-tuned network according to 2.1 includes 25.56 million parameters and for the adapter network with two fully-connected layers and a deep feature size of 2'048 only 1.1 million $((2'048*512)+(512*67)+512+67))$ parameters. Despite a small learning rate of 0.0001 and an additional decay scheme, diverging weight adaptation seems to occur. With increasing training size, the fine-tuned network should be superior since the fine-tuning of the pre-trained network allows the deep features to be adapted to the target dataset before the extraction and creation of the gallery, thus incorporating specific data characteristics. This is not possible with the adapter network. As seen in figure 6.8, the additional use of the SVM approach can reduce the disadvantage of fine-tuning with few samples per class compared to the adaptive and pre-trained approach. In particular, with more samples, the advantage of fine-tuning the weights in the pre-trained network becomes evident again. It is also evident that training with the adaptor network does not significantly improve the discrimination properties of deep features towards the target dataset. The classification accuracy remains below the results obtained with the pre-trained network, showing only better performance for the Fruit and Vegetable dataset with more samples. Therefore, according to these experiments, it is advisable to use the pre-trained network for enrolling the gallery for a few samples without training. One reason could be that the adapter network with 1.1 million parameters is too complex to generate better discriminated deep features. In contrast, the work of Günther et al. (2020), for example, has shown that the adapter network they developed with Objectosphere loss outperforms the VGG2 network for open-set face recognition tasks. The proposed architecture of Maqsood et al. (2019) also achieves promising results, although the authors do not compare it to the classification of the pre-trained network without training. Furthermore, for more difficult classification tasks (Aircraft and Virus) with 1-3 samples per class, the representation of the deep features is so limited that it is not relevant which transfer learning method is chosen. In this case, it makes sense not to do additional training but to use the extracted deep features directly from the pre-trained network. In figure 6.7 it is apparent that the extracted features should preferably be used in combination with the SVM classifier when using more samples. That the utilization of SVM classifiers offers advantages has already been demonstrated by the work of Cıbuk et al. (2019), where even without feature selection, better performance is achieved with more samples. Figure 6.8 also illustrates that fine-tuning of deep features is indicated from 20 samples per class to increase classification performance. Above this amount of training data, fine-tuning in combination with SVM leads to the best classification accuracy for all datasets with more than 20 samples per class. This occurs even earlier when ten samples per class are used for the Aircraft and Fruit and Vegetable datasets. With few samples, however, the SVM classifier cannot learn the decision boundary robustly due to the insufficient descriptive deep features. In this case, the MEAN approach is the preferred option. Compared to the COS approach, greater variation can occur, making it challenging for probing test features to determine the correct class. As pointed out by the work of Günther et al. (2020), the superiority of deep feature extraction with subsequential gallery enrollment and probing over the use of the sole softmax classifier in figure 6.8 is evident.

Differences being more significant for more difficult domains (aircraft, viruses) than for domains related to ImageNet may be explained by several reasons. This may be due to the fact that deep features have a larger parameter dimension than the logits fed into the softmax classifier and therefore contain even more information that is not yet sufficiently compressible. Domains with greater similarity to the source might already contain enough descriptive power, so that dimension difference is less relevant.

The constant gallery (C-GAL) features studied in **RQ 5** and shown in figure 6.9 are limited in their ability to increase performance despite more training samples, which indicates that the individual features are not sufficiently descriptive for the gallery. Under these circumstances, it makes sense to use the image samples already used for training for the gallery (I-GAL). For this reason, one can see the difference between the datasets less overlapped with ImageNet and the rest of the datasets. For the Aircraft and Virus datasets, the better performance of C-GAL despite more samples per class indicates that image samples that were not part of the training can form a more robust gallery due to the over-adaptation of the training samples when using the I-GAL approach. Thus, based on the results, it can be concluded that for fine-grained domains with few samples per class, samples not used for training should be selected for the gallery. If more than ten samples are available or the target dataset strongly overlaps with ImageNet, it is recommended to enroll the gallery with training samples. What has to be considered is that the samples for C-GAL would also improve the performance of the respective classifier as training samples, which is in contrast to this methodology. Although this is not the case with the Virus dataset results, this would need to be further investigated in other experiments with alternative target and source domains. The findings of Yosinski et al. (2014) and Yosinski et al. (2015) regarding the declining transferability between greater distances between source and target datasets are replicated in the results of this study. Since deep feature extraction was performed in deeper layers with, according to the authors, dataset-specific features in all experiments in this thesis, this could be why the samples used for training for the gallery perform worse on more distant datasets related to ImageNet. The extracted deep features are over-adapted to ImageNet. Therefore, when using image samples that are not yet known to the network, better performance is achieved because the specificity is less pronounced. The effect is diminished with more samples because it seems that the number of image samples counteracts the specificity with more variability in the training data. In smaller distances between source and target dataset, as is the case with the Indoor, Fruit, and Vegetable, and Office-31 dataset, the specificity of the deep features is advantageous, which is why in these cases, better performance is also achieved compared to C-GAL.

In this section, conclusions regarding class performance are described. Across all datasets, there is a high dispersity across classes when only one sample per class is utilized. The networks cannot yet separate the feature space according to the classes. However, with smaller distances between the source and target dataset, there are already more classes classified with over 80% accuracy. In the case of the Aircraft and Virus dataset, on the other hand, even with more samples, there are still classes that are classified proportionally more frequently. In the case of Aircraft, it can be assumed that these classes correspond most closely to the feature space of the corresponding classes in ImageNet, in particular, the class *Airliner*. Such classes can also be identified in the other datasets. For the Virus dataset, there is no overlap, so it was to be expected that the features would not approach the descriptiveness of the features of other datasets. Even the best classifiers cannot classify classes with highly overlapping feature space in a few samples, and the distinction is not a trivial task even for humans. Examples are the classes *Corn* and *Sweetcorn* in the Fruit and Vegetable and *Trainstation* and *Subway* in the Indoor Scenes dataset.

Based on the results, it is clear that the transfer learning approaches are only as good as the extracted deep features. For this reason, it can be seen that the prediction of some individual classes is relatively inefficient across all approaches investigated. The results indicate that classification accuracy is not the most appropriate measure for evaluating multi-class classification. Some classes are always correct, others never. This is not the intended behavior of a classifier.

Transfer learning methods can be applied to different domains, but in this thesis, only supervised image classification tasks are being discussed. In order to allow an efficient adaptation with respect to the five pre-trained networks, and also due to the wide variation of existing transfer learning methods, only basic principles described in chapter 3 are considered, especially regarding fine-tuning where no layers were frozen. It is also important to note that the deep features were not extracted from different parts of a specific network but only from the same layer. Furthermore, constructing a constant number of samples per class during training allows a more straightforward analysis utilizing accuracy metrics over all datasets. However, the data distribution as a side effect does not correspond to reality. Thus, the results of the conducted experiments must be taken with caution. Additionally, specific classes have to be neglected from further analysis due to the limited number of class samples in the Virus and Indoor Scenes data. This fact is also mentioned in more detail in chapter 4. Before comparing the approaches, network design decisions had to be made, especially regarding the adapter network, where many design options are available. Attention is paid to the regularization, number of layers, and deep features or logits input, which only allows limiting statements about the performance of the adapter network. The design decisions were made based on the Indoor Scenes only, which generally might not be the ideal choice for other datasets. Moreover, only the cross-entropy loss function with softmax is used during the training of the networks. Other loss functions are not considered.

Chapter 8

Conclusion

Many research projects have already successfully adopted transfer learning methods. Often, fine-tuning or deep feature extraction techniques are utilized, which achieve performance improvements despite the limited availability of the target dataset. However, an in-depth evaluation of transfer learning methods with only 1-3 training samples per class has not yet been conducted. Also, the number of training samples per class above which a transfer learning approach is indicated has not yet been investigated to the same extent as in this work. By carefully analyzing the classification performance of different transfer learning methods, this thesis shows that no additional deep feature adaptation to the target dataset is indicated, especially for less than 20 samples per class. The deep features extracted from the pre-trained network using the MEAN approach should be used directly. As the number of samples increases, it becomes clear that adjustment of the network weights is advisable. Fine-tuning is indicated when 20 or more samples per class are available. The top-1 accuracy scores can be considered as an indicator of network performance on transfer learning tasks. However, network performance generally varies to a greater extent for more distant target datasets compared to ImageNet-related datasets, leading to the conclusion that ImageNet top-1 accuracy is, in this case, not an appropriate benchmark. Furthermore, having 1-3 samples per class, it is not relevant which transfer learning method (pre-trained, fine-tuned, or adaptive) is chosen for more distant classification tasks due to the limitation of the learned deep feature representation. Interestingly, deep feature extraction generally improves classification performance, which is particularly noticeable in the case of distant target datasets. Choosing the appropriate transfer learning approach is not simple, especially when only a few training samples are available. However, this thesis's findings should help focus on the essential transfer learning techniques to achieve the best possible outcome.

In order to further expand the conclusions reached, one focus should be on additional loss functions for training the adapter network and fine-tuning the pre-trained network. Furthermore, the investigations should be carried out with other target datasets that have more or less overlaps with the source dataset (e.g., Visual Domain Decathlon[1]). Experiments with other source datasets and other pre-trained networks (e.g., Inception-V3[2], EfficientNet-V2[2]) would also be appropriate. As the investigations have shown, achieving accurate performance for fine-grained and non-source-related datasets with only a few samples is challenging. Other approaches to further expand on are few-shot learning algorithms and ensemble models. It would be interesting to analyze their performances on the same tasks.

[1] https://www.robots.ox.ac.uk/ vgg/decathlon/
[2] https://pytorch.org/vision/stable/models.html

In addition, adapter and fine-tuning network variants could be included in the analysis, for example, with varying activation features, a different selection of layers to freeze, and where deep feature extraction should occur. Feature maps visualizations of deep learning networks, for example, with the help of Grad-CAM (Selvaraju et al., 2020), would also help to understand the learning process better.

Appendix A

Aircraft Results

Table A.1: END-TO-END CLASSIFICATION PERFORMANCE AIRCRAFT. End-to-end classification performance on the Aircraft dataset and pre-trained networks utilizing the fine-tuned and adapter network approach.

	1	**2**	**3**	**4**	**5**	**7**	**10**	**25**	**33**
alexnet-adaptive	0.027	0.035	0.029	0.067	0.048	0.064	0.092	0.170	0.173
alexnet-finetuned	0.033	0.063	0.083	0.079	0.098	0.097	0.129	0.221	0.237
densenet-adaptive	**0.104**	**0.093**	0.069	0.096	0.102	0.098	0.161	0.211	0.197
densenet-finetuned	0.041	0.041	0.062	0.042	0.068	0.079	0.127	0.401	0.401
mobilenet-adaptive	0.035	0.076	0.066	0.082	0.117	0.125	0.158	0.243	0.246
mobilenet-finetuned	0.022	0.044	0.035	0.092	0.082	0.038	0.147	0.252	0.285
resnet50-adaptive	0.084	0.075	**0.090**	0.093	**0.163**	0.121	0.146	0.216	0.240
resnet50-finetuned	0.030	0.085	0.035	0.051	0.134	0.145	0.191	0.371	**0.404**
vgg16-adaptive	0.038	0.045	0.059	0.084	0.086	0.083	0.132	0.236	0.243
vgg16-finetuned	0.066	0.071	**0.090**	**0.144**	0.132	**0.177**	**0.208**	**0.358**	0.390

Table A.2: ADAPTIVE CLASSIFICATION PERFORMANCE AIRCRAFT. Deep Features classification performance on the Aircraft dataset and pre-trained networks utilizing the adapter network approach.

	1	**2**	**3**	**4**	**5**	**7**	**10**	**25**	**33**
alexnet-cos	0.045	0.052	0.072	0.091	0.091	0.109	0.147	0.262	0.280
alexnet-mean	0.045	0.052	0.073	0.082	0.089	0.116	0.142	0.283	0.314
alexnet-svm	0.038	0.053	0.065	0.072	0.085	0.117	0.163	0.306	0.342
densenet-cos	0.126	0.161	0.179	0.210	0.222	0.226	0.258	0.314	0.332
densenet-mean	0.126	0.154	0.185	0.209	0.223	0.219	0.251	0.328	0.363
densenet-svm	0.118	0.139	0.199	0.233	**0.260**	0.262	**0.336**	**0.460**	**0.494**
mobilenet-cos	0.098	0.154	0.174	0.186	0.214	0.236	0.281	0.372	0.400
mobilenet-mean	0.098	0.151	0.180	0.206	0.222	0.229	0.287	0.402	0.406
mobilenet-svm	0.103	0.157	0.193	0.218	0.252	**0.277**	0.311	0.441	0.474
resnet50-cos	**0.167**	0.169	0.215	0.212	0.222	0.233	0.268	0.320	0.355
resnet50-mean	**0.167**	0.172	0.193	0.223	0.246	0.264	0.282	0.348	0.393
resnet50-svm	0.153	**0.176**	**0.224**	**0.246**	0.207	0.267	0.329	0.443	0.449
vgg16-cos	0.042	0.070	0.103	0.119	0.138	0.162	0.219	0.306	0.339
vgg16-mean	0.042	0.069	0.105	0.116	0.134	0.160	0.209	0.350	0.389
vgg16-svm	0.047	0.062	0.116	0.114	0.140	0.180	0.247	0.383	0.413

Table A.3: FINE-TUNED CLASSIFICATION PERFORMANCE AIRCRAFT. Deep Features classification performance on the Aircraft dataset and pre-trained networks utilizing the fine-tuned network approach.

	1	2	3	4	5	7	10	25	33
alexnet-cos	0.097	0.106	0.139	0.149	0.174	0.176	0.218	0.318	0.342
alexnet-mean	0.097	0.106	0.143	0.156	0.168	0.195	0.223	0.325	0.346
alexnet-svm	0.051	0.071	0.109	0.135	0.144	0.155	0.240	0.371	0.431
densenet-cos	0.149	0.152	0.147	0.219	0.180	0.175	0.238	0.580	0.602
densenet-mean	0.149	0.165	0.164	0.233	0.176	0.170	0.268	0.574	0.591
densenet-svm	0.095	0.144	0.170	0.222	0.193	0.180	0.314	0.617	0.656
mobilenet-cos	0.096	0.134	0.176	0.215	0.207	0.218	0.286	0.403	0.485
mobilenet-mean	0.096	0.147	0.199	0.233	0.239	0.252	0.337	0.446	0.497
mobilenet-svm	0.110	0.135	0.206	0.227	0.241	0.299	0.352	0.517	0.577
resnet50-cos	0.162	0.138	0.210	0.237	0.260	0.248	0.315	0.545	0.583
resnet50-mean	0.162	0.140	0.203	0.271	**0.266**	0.276	0.343	0.559	0.586
resnet50-svm	0.150	0.152	**0.221**	**0.276**	0.253	0.296	**0.369**	**0.604**	**0.638**
vgg16-cos	**0.170**	0.186	0.202	0.231	0.244	0.278	0.313	0.497	0.533
vgg16-mean	**0.170**	**0.189**	0.204	0.238	0.244	**0.311**	0.334	0.508	0.522
vgg16-svm	0.115	0.149	0.189	0.210	0.242	0.304	0.331	0.552	0.585

Table A.4: PRE-TRAINED CLASSIFICATION PERFORMANCE AIRCRAFT. Deep Features classification performance on the Aircraft dataset and pre-trained networks.

	1	2	3	4	5	7	10	25	33
alexnet-cos	0.093	0.112	0.139	0.139	0.164	0.167	0.211	0.264	0.274
alexnet-mean	0.093	0.104	0.136	0.153	0.165	0.190	0.206	0.246	0.242
alexnet-svm	0.048	0.075	0.108	0.129	0.138	0.160	0.208	0.338	0.357
densenet-cos	**0.180**	**0.190**	0.207	0.223	0.246	0.262	0.291	0.409	0.419
densenet-mean	**0.180**	**0.190**	**0.229**	0.255	0.274	0.285	0.328	0.412	0.416
densenet-svm	0.129	0.167	0.210	0.250	0.283	0.316	**0.390**	**0.545**	**0.571**
mobilenet-cos	0.104	0.134	0.176	0.194	0.218	0.235	0.262	0.364	0.381
mobilenet-mean	0.104	0.157	0.212	0.231	0.236	0.259	0.317	0.375	0.383
mobilenet-svm	0.111	0.151	0.204	0.237	0.259	0.298	0.350	0.459	0.522
resnet50-cos	0.155	0.173	0.197	0.209	0.225	0.240	0.267	0.384	0.371
resnet50-mean	0.155	0.175	0.202	0.235	0.237	0.279	0.315	0.381	0.369
resnet50-svm	0.153	0.187	0.226	**0.275**	**0.293**	**0.334**	0.387	0.504	0.534
vgg16-cos	0.164	0.178	0.199	0.208	0.221	0.226	0.266	0.348	0.346
vgg16-mean	0.164	0.179	0.198	0.224	0.234	0.252	0.296	0.378	0.387
vgg16-svm	0.106	0.148	0.172	0.206	0.204	0.255	0.308	0.441	0.449

Table A.5: CONSTANT GALLERY AIRCRAFT. Nearest neighbor classification performance with MEAN on the Fruit and Vegetable Aircraft networks using the C-GAL approach.

	1	2	3	4	5	7	10	25	28
densenet-adaptive-mean	0.188	0.217	0.220	0.233	0.223	0.232	0.232	0.265	0.277
densenet-finetuned-mean	0.295	0.306	0.245	0.272	0.196	0.161	0.250	**0.515**	**0.497**
densenet-pretrained-mean	**0.324**	**0.324**	**0.324**	**0.324**	0.324	**0.324**	0.324	0.324	0.324
mobilenet-adaptive-mean	0.241	0.195	0.201	0.226	0.229	0.241	0.260	0.355	0.339
mobilenet-finetuned-mean	0.281	0.257	0.286	0.262	0.275	0.278	0.306	0.360	0.410
mobilenet-pretrained-mean	0.287	0.287	0.287	0.287	0.287	0.287	0.287	0.287	0.287
resnet50-adaptive-mean	0.246	0.240	0.240	0.249	0.254	0.245	0.269	0.301	0.308
resnet50-finetuned-mean	0.286	0.236	0.280	0.305	**0.341**	0.294	**0.345**	0.499	0.473
resnet50-pretrained-mean	0.287	0.287	0.287	0.287	0.287	0.287	0.287	0.287	0.287

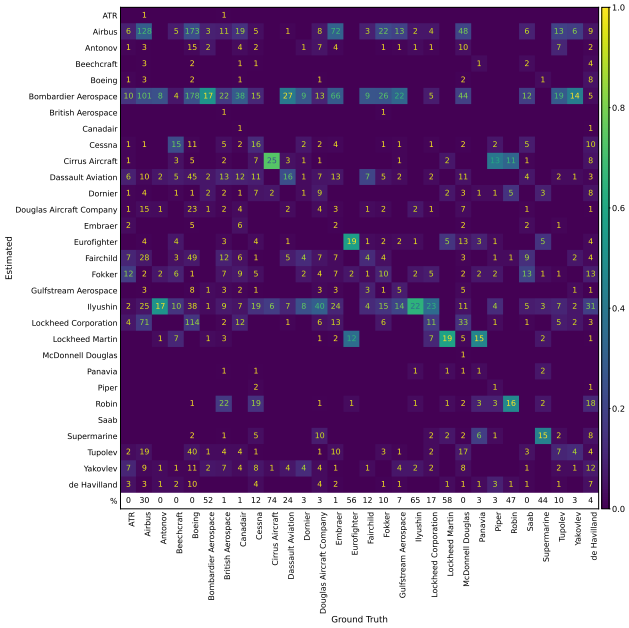


Figure A.1: CONFUSION MATRIX AIRCRAFT - 1 SAMPLE. This figure shows the confusion matrix adopting the nearest neighbor classification with MEAN using the pre-trained network approach. DenseNet-121 is utilized as pre-trained network on the Aircraft dataset using 1 sample per class.

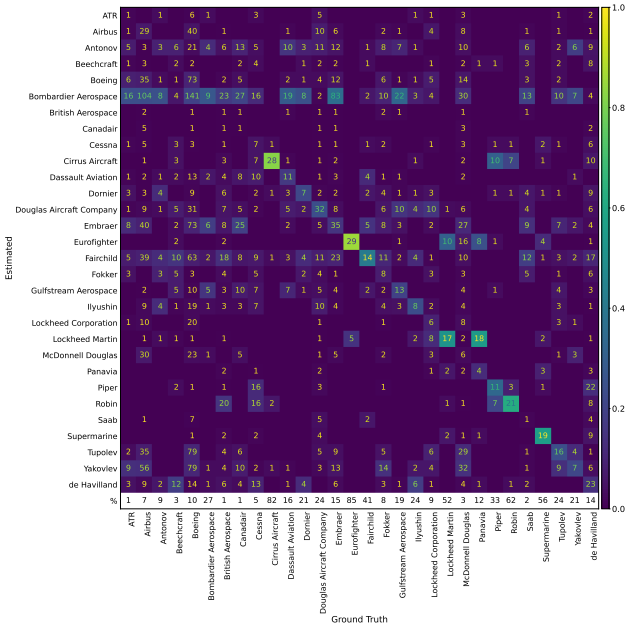


Figure A.2: CONFUSION MATRIX AIRCRAFT - 3 SAMPLES. This figure shows the confusion matrix adopting the nearest neighbor classification with MEAN using the pre-trained network approach. DenseNet-121 is utilized as pre-trained network on the Aircraft dataset using 3 samples per class.

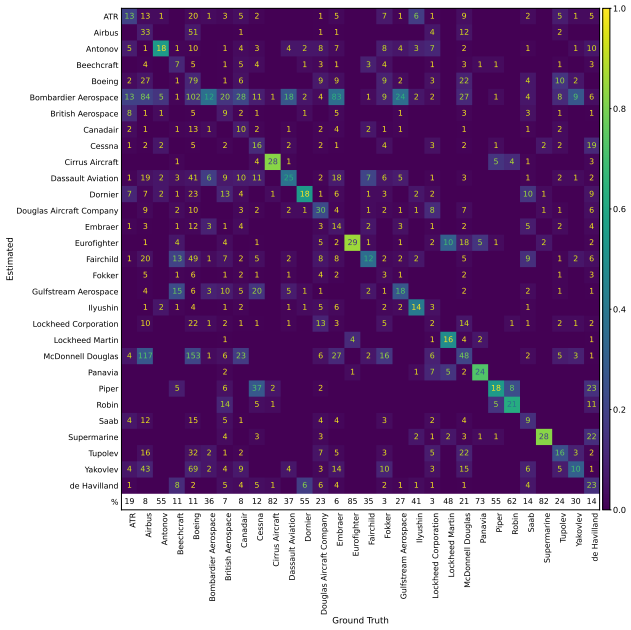


Figure A.3: CONFUSION MATRIX AIRCRAFT - 10 SAMPLES. This figure shows the confusion matrix adopting the nearest neighbor classification with MEAN using the pre-trained network approach. DenseNet-121 is utilized as pre-trained network on the Aircraft dataset using 10 samples per class.

Appendix B

Fruit and Vegetable Results

Table B.1: END-TO-END CLASSIFICATION PERFORMANCE FRUIT AND VEGETABLE. End-to-end classification on the Fruit and Vegetable dataset and pre-trained networks utilizing the fine-tuned and adapter network approach.

	1	2	4	6	9	13	20	67	95
alexnet-adaptive	0.017	0.145	0.432	0.585	0.657	0.755	0.816	0.877	0.903
alexnet-finetuned	0.524	0.588	0.741	0.772	0.786	0.819	0.864	0.942	**0.983**
densenet-adaptive	0.596	**0.727**	0.797	0.830	0.864	0.855	0.861	0.900	0.908
densenet-finetuned	0.435	0.627	0.769	0.827	**0.866**	**0.900**	**0.916**	**0.964**	0.978
mobilenet-adaptive	0.638	0.708	**0.838**	**0.844**	0.844	0.872	0.889	0.916	0.933
mobilenet-finetuned	0.538	0.652	0.755	0.816	0.844	0.883	0.891	0.955	0.975
resnet50-adaptive	**0.649**	0.702	**0.838**	**0.844**	0.861	0.875	0.883	0.908	0.889
resnet50-finetuned	0.563	0.674	0.819	0.833	0.855	0.883	0.900	0.955	0.978
vgg16-adaptive	0.042	0.231	0.607	0.724	0.808	0.822	0.875	0.900	0.925
vgg16-finetuned	0.638	0.655	0.774	0.830	0.847	0.883	0.889	0.961	0.981

Table B.2: ADAPTIVE CLASSIFICATION PERFORMANCE FRUIT AND VEGETABLE. Deep Features classification performance on the Fruit and Vegetable dataset and pre-trained networks utilizing the adapter network approach.

	1	2	4	6	9	13	20	67	95
alexnet-cos	0.245	0.357	0.530	0.608	0.702	0.811	0.833	0.931	**0.986**
alexnet-mean	0.245	0.349	0.530	0.585	0.697	0.794	0.811	0.858	0.878
alexnet-svm	0.231	0.365	0.521	0.599	0.694	0.792	0.856	0.906	0.928
densenet-cos	0.558	0.686	0.775	0.803	0.833	0.856	0.886	0.942	0.956
densenet-mean	0.588	0.677	0.800	0.814	0.811	0.847	0.839	0.872	0.875
densenet-svm	0.563	0.683	0.781	0.833	0.844	0.875	0.878	0.917	0.914
mobilenet-cos	0.627	**0.725**	0.825	0.839	0.825	0.869	0.894	**0.964**	**0.986**
mobilenet-mean	0.627	0.717	0.828	0.839	0.833	0.856	0.858	0.886	0.878
mobilenet-svm	0.560	0.706	0.819	0.844	0.844	0.875	0.894	0.919	0.956
resnet50-cos	**0.680**	0.706	0.817	0.842	0.850	0.869	**0.897**	0.947	**0.986**
resnet50-mean	**0.680**	0.719	**0.831**	0.828	0.850	0.856	0.844	0.875	0.881
resnet50-svm	0.655	0.719	0.817	**0.850**	**0.864**	**0.878**	0.878	0.931	0.917
vgg16-cos	0.279	0.438	0.714	0.742	0.825	0.828	0.861	0.944	**0.986**
vgg16-mean	0.279	0.427	0.731	0.744	0.816	0.817	0.836	0.858	0.892
vgg16-svm	0.281	0.435	0.697	0.750	0.805	0.856	0.842	0.906	0.942

Table B.3: FINE-TUNED CLASSIFICATION PERFORMANCE FRUIT AND VEGETABLE. Deep Features classification performance on the Fruit and Vegetable dataset and pre-trained networks utilizing the fine-tuned network approach.

	1	2	4	6	9	13	20	67	95
alexnet-cos	0.552	0.636	0.753	0.767	0.792	0.828	0.836	0.919	**0.986**
alexnet-mean	0.552	0.661	0.753	0.794	0.783	0.775	0.808	0.850	0.850
alexnet-svm	0.432	0.519	0.663	0.728	0.775	0.806	0.867	0.931	**0.986**
densenet-cos	0.508	0.672	0.814	0.828	**0.886**	**0.900**	0.903	0.953	**0.986**
densenet-mean	0.508	0.683	0.811	0.819	0.867	0.881	0.881	0.916	0.936
densenet-svm	0.483	0.683	0.811	0.847	0.872	0.897	0.900	**0.967**	**0.986**
mobilenet-cos	0.613	0.686	0.811	0.825	0.819	0.875	0.886	0.956	**0.986**
mobilenet-mean	0.613	0.706	0.806	0.828	0.850	0.883	0.878	0.919	0.900
mobilenet-svm	0.527	0.667	0.794	0.831	0.867	0.886	**0.917**	0.961	0.975
resnet50-cos	0.652	0.728	**0.847**	0.822	0.872	0.889	0.911	0.961	**0.986**
resnet50-mean	0.652	**0.730**	0.842	0.836	0.864	0.889	0.875	0.933	0.942
resnet50-svm	0.602	0.708	0.836	**0.861**	0.861	0.886	0.914	0.956	0.969
vgg16-cos	**0.680**	0.714	0.814	0.842	0.831	0.861	0.883	**0.967**	**0.986**
vgg16-mean	**0.680**	0.714	0.811	0.847	0.847	0.867	0.856	0.908	0.897
vgg16-svm	0.547	0.605	0.803	0.808	0.856	0.878	0.900	0.956	**0.986**

Table B.4: PRE-TRAINED CLASSIFICATION PERFORMANCE FRUIT AND VEGETABLE. Deep Features classification performance on the Fruit and Vegetable dataset and pre-trained networks.

	1	2	4	6	9	13	20	67	95
alexnet-cos	0.538	0.650	0.744	0.761	0.758	0.794	0.811	0.919	0.986
alexnet-mean	0.538	0.655	0.747	0.764	0.769	0.758	0.783	0.789	0.797
alexnet-svm	0.393	0.516	0.702	0.708	0.753	0.792	0.844	0.906	0.942
densenet-cos	0.674	0.744	0.819	0.836	0.839	0.856	0.864	0.950	0.986
densenet-mean	0.674	**0.775**	**0.831**	0.842	0.850	0.856	0.858	0.878	0.867
densenet-svm	0.616	0.719	0.803	**0.847**	**0.883**	**0.917**	0.908	0.953	0.956
mobilenet-cos	0.622	0.706	0.783	0.789	0.808	0.828	0.853	0.947	**0.989**
mobilenet-mean	0.622	0.725	0.817	0.825	0.836	0.850	0.867	0.875	0.869
mobilenet-svm	0.549	0.700	0.808	0.842	0.856	0.892	**0.919**	**0.961**	0.975
resnet50-cos	**0.683**	0.725	0.819	0.833	0.825	0.856	0.872	0.947	0.986
resnet50-mean	**0.683**	0.761	0.817	0.828	0.825	0.844	0.839	0.844	0.844
resnet50-svm	0.638	0.725	0.811	0.839	0.858	0.881	0.900	0.950	0.956
vgg16-cos	0.655	0.708	0.778	0.781	0.828	0.825	0.858	0.950	0.986
vgg16-mean	0.655	0.711	0.797	0.822	0.811	0.839	0.836	0.847	0.835
vgg16-svm	0.469	0.611	0.767	0.800	0.822	0.867	0.875	0.939	0.983

Table B.5: CONSTANT GALLERY FRUIT AND VEGETABLE. Nearest neighbor classification performance with MEAN on the Fruit and Vegetable dataset and pre-trained networks using the C-GAL approach.

	1	2	4	6	9	13	20	67	90
densenet-adaptive-mean	0.588	0.637	0.646	0.674	0.702	0.727	0.716	0.811	0.803
densenet-finetuned-mean	0.542	0.669	0.724	0.738	**0.799**	0.799	0.791	0.844	0.847
densenet-pretrained-mean	0.705	0.705	0.705	0.705	0.705	0.705	0.705	0.705	0.705
mobilenet-adaptive-mean	0.647	0.689	**0.767**	0.733	0.775	0.786	0.786	0.817	0.819
mobilenet-finetuned-mean	0.683	0.707	0.724	0.735	0.740	0.755	0.766	0.808	0.794
mobilenet-pretrained-mean	0.680	0.680	0.680	0.680	0.680	0.680	0.680	0.680	0.680
resnet50-adaptive-mean	0.647	0.661	0.697	0.711	0.700	0.725	0.742	0.789	0.803
resnet50-finetuned-mean	0.707	0.691	0.760	**0.749**	0.785	**0.813**	**0.799**	**0.872**	**0.874**
resnet50-pretrained-mean	**0.710**	**0.710**	0.710	0.710	0.710	0.710	0.710	0.710	0.710

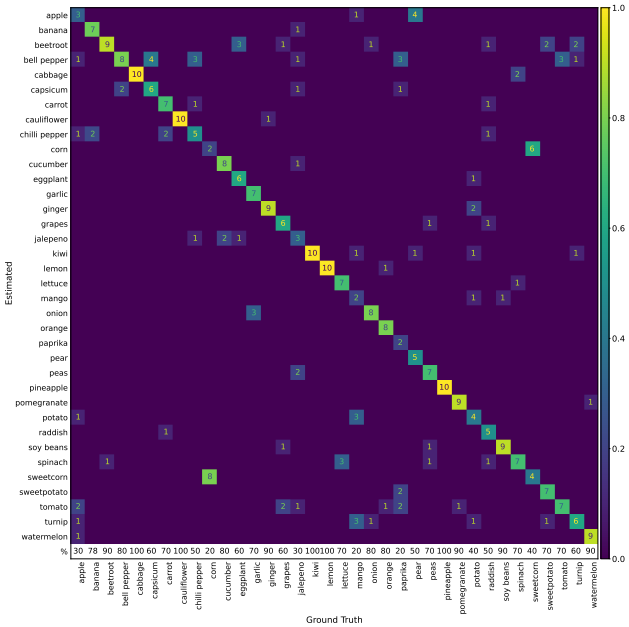


Figure B.1: CONFUSION MATRIX FRUIT AND VEGETABLE - 1 SAMPLE. This figure shows the confusion matrix adopting the nearest neighbor classification with MEAN using the pre-trained network approach. DenseNet-121 is utilized as pre-trained network on the Fruit and Vegetable dataset using 1 sample per class.

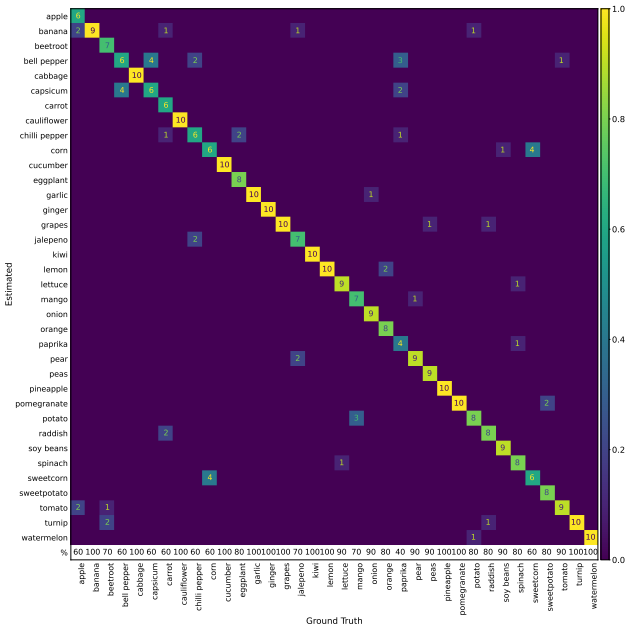


Figure B.2: CONFUSION MATRIX FRUIT AND VEGETABLE - 4 SAMPLES. This figure shows the confusion matrix adopting the nearest neighbor classification with MEAN using the pre-trained network approach. DenseNet-121 is utilized as pre-trained network on the Fruit and Vegetable dataset using 4 samples per class.

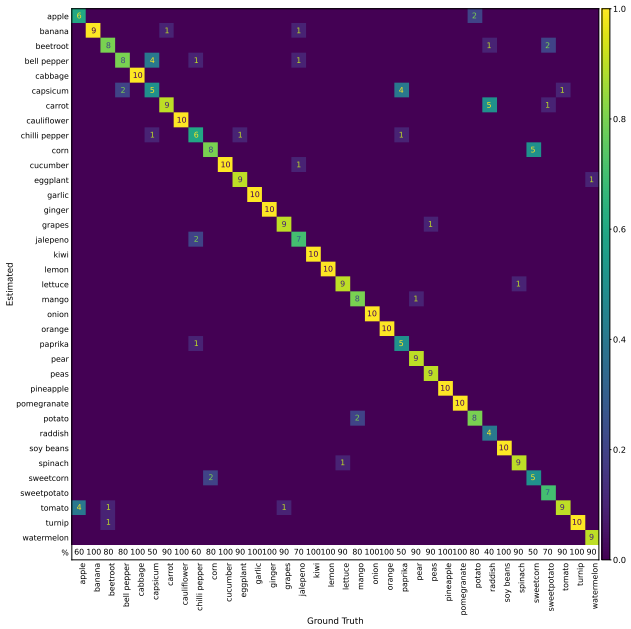


Figure B.3: CONFUSION MATRIX FRUIT AND VEGETABLE - 9 SAMPLES. This figure shows the confusion matrix adopting the nearest neighbor classification with MEAN using the pre-trained network approach. DenseNet-121 is utilized as pre-trained network on the Fruit and Vegetable dataset using 9 samples per class.

Appendix C

Indoor Scenes Results

Table C.1: END-TO-END CLASSIFICATION PERFORMANCE INDOOR SCENES. End-to-end classification performance on the Indoor Scenes dataset and pre-trained networks utilizing the fine-tuned and adapter network approach.

	1	2	3	5	7	11	16	48	67
alexnet-adaptive	0.036	0.094	0.148	0.284	0.338	0.364	0.413	0.513	0.538
alexnet-finetuned	0.118	0.186	0.245	0.300	0.325	0.393	0.428	0.547	0.573
densenet-adaptive	0.215	0.325	0.379	0.416	0.479	0.493	0.540	0.639	0.654
densenet-finetuned	0.087	0.183	0.260	0.375	0.400	0.519	0.564	0.731	0.751
mobilenet-adaptive	0.243	0.332	0.426	0.481	0.507	0.553	0.592	0.662	0.690
mobilenet-finetuned	0.114	0.192	0.312	0.369	0.440	0.492	0.535	0.679	0.690
resnet50-adaptive	**0.249**	**0.410**	**0.447**	**0.511**	**0.569**	0.573	0.613	0.699	0.728
resnet50-finetuned	0.169	0.300	0.364	0.457	0.506	**0.587**	**0.619**	**0.746**	**0.769**
vgg16-adaptive	0.115	0.216	0.397	0.451	0.507	0.534	0.573	0.651	0.666
vgg16-finetuned	0.190	0.318	0.390	0.451	0.474	0.542	0.596	0.709	0.719

Table C.2: ADAPTIVE CLASSIFICATION PERFORMANCE INDOOR SCENES. Deep Features classification performance on the Indoor Scenes dataset and pre-trained networks utilizing the adapter network approach.

	1	2	3	5	7	11	16	48	67
alexnet-cos	0.047	0.101	0.140	0.224	0.270	0.308	0.336	0.423	0.429
alexnet-mean	0.047	0.102	0.163	0.247	0.316	0.362	0.395	0.498	0.526
alexnet-svm	0.039	0.092	0.140	0.233	0.292	0.352	0.386	0.513	0.542
densenet-cos	0.242	0.306	0.315	0.396	0.422	0.439	0.493	0.568	0.589
densenet-mean	0.242	0.325	0.336	0.402	0.443	0.464	0.497	0.601	0.607
densenet-svm	0.225	0.309	0.338	0.409	0.462	0.511	0.556	0.644	0.675
mobilenet-cos	0.272	0.337	0.376	0.438	0.455	0.485	0.494	0.605	0.603
mobilenet-mean	0.272	0.356	0.419	0.485	0.492	0.530	0.555	0.635	0.637
mobilenet-svm	0.227	0.324	0.387	0.477	0.505	**0.557**	0.572	0.677	0.688
resnet50-cos	**0.314**	0.415	0.442	0.466	0.495	0.498	0.532	0.602	0.631
resnet50-mean	**0.314**	**0.427**	**0.463**	**0.509**	**0.547**	0.555	0.592	0.666	0.670
resnet50-svm	0.267	0.389	0.435	0.502	0.546	0.560	**0.594**	**0.705**	**0.717**
vgg16-cos	0.094	0.235	0.325	0.400	0.422	0.481	0.509	0.580	0.594
vgg16-mean	0.094	0.232	0.328	0.430	0.452	0.515	0.550	0.615	0.633
vgg16-svm	0.083	0.209	0.319	0.425	0.455	0.515	0.558	0.658	0.683

Table C.3: FINE-TUNED CLASSIFICATION PERFORMANCE INDOOR SCENES. Deep Features classification performance on the Indoor Scenes dataset and pre-trained networks utilizing the fine-tuned network approach.

	1	2	3	5	7	11	16	48	67
alexnet-cos	0.182	0.208	0.223	0.244	0.264	0.300	0.326	0.429	0.454
alexnet-mean	0.182	0.239	0.278	0.325	0.366	0.403	0.425	0.525	0.525
alexnet-svm	0.081	0.126	0.185	0.255	0.314	0.366	0.406	0.535	0.551
densenet-cos	0.161	0.229	0.277	0.361	0.353	0.464	0.505	0.642	0.677
densenet-mean	0.161	0.250	0.298	0.401	0.416	0.514	0.553	0.688	0.709
densenet-svm	0.167	0.257	0.322	0.417	0.466	0.553	0.574	0.717	0.743
mobilenet-cos	0.197	0.274	0.309	0.351	0.385	0.441	0.472	0.594	0.621
mobilenet-mean	0.197	0.322	0.389	0.442	0.474	0.539	0.555	0.666	0.659
mobilenet-svm	0.114	0.251	0.356	0.419	0.456	0.510	0.556	0.681	0.700
resnet50-cos	**0.246**	0.325	0.367	0.464	0.477	0.543	0.575	0.692	0.722
resnet50-mean	**0.246**	0.345	0.400	**0.495**	**0.541**	**0.598**	0.617	0.723	0.761
resnet50-svm	0.195	0.336	0.366	0.493	0.529	0.590	**0.636**	**0.747**	**0.772**
vgg16-cos	0.242	0.369	0.373	0.428	0.441	0.483	0.525	0.630	0.647
vgg16-mean	0.242	**0.402**	**0.430**	0.491	0.518	0.561	0.596	0.677	0.682
vgg16-svm	0.121	0.268	0.326	0.412	0.474	0.540	0.587	0.689	0.707

Table C.4: PRE-TRAINED CLASSIFICATION PERFORMANCE INDOOR SCENES. Deep Features classification performance on the Indoor Scenes dataset and pre-trained networks.

	1	2	3	5	7	11	16	48	67
alexnet-cos	0.188	0.233	0.244	0.266	0.288	0.312	0.333	0.393	0.402
alexnet-mean	0.188	0.276	0.304	0.359	0.381	0.409	0.428	0.514	0.520
alexnet-svm	0.072	0.131	0.185	0.263	0.315	0.370	0.408	0.520	0.541
densenet-cos	0.291	0.358	0.402	0.436	0.438	0.480	0.513	0.559	0.593
densenet-mean	0.291	0.416	0.473	0.529	0.551	0.587	0.599	0.663	0.663
densenet-svm	0.235	0.365	0.449	0.526	0.541	0.584	0.613	0.712	0.724
mobilenet-cos	0.299	0.356	0.389	0.437	0.457	0.479	0.507	0.563	0.580
mobilenet-mean	0.299	0.413	0.469	0.545	0.565	0.597	0.605	0.677	0.680
mobilenet-svm	0.228	0.352	0.418	0.516	0.543	0.572	0.596	0.712	0.725
resnet50-cos	**0.345**	0.416	0.451	0.486	0.487	0.511	0.535	0.614	0.637
resnet50-mean	**0.345**	**0.460**	**0.523**	**0.590**	**0.604**	**0.618**	0.624	0.679	0.686
resnet50-svm	0.277	0.432	0.483	0.563	0.596	0.616	**0.640**	**0.723**	**0.744**
vgg16-cos	0.320	0.394	0.417	0.461	0.477	0.488	0.502	0.559	0.579
vgg16-mean	0.320	0.425	0.472	0.530	0.553	0.564	0.593	0.633	0.641
vgg16-svm	0.149	0.270	0.373	0.465	0.505	0.555	0.591	0.658	0.689

Table C.5: CONSTANT GALLERY INDOOR SCENES. Nearest neighbor classification performance with MEAN on the Indoor Scenes dataset and pre-trained networks using the C-GAL approach.

	1	2	3	5	7	11	16	48	62
densenet-adaptive-mean	0.396	0.399	0.407	0.421	0.426	0.460	0.469	0.520	0.536
densenet-finetuned-mean	0.382	0.377	0.377	0.417	0.407	0.504	0.517	0.625	0.617
densenet-pretrained-mean	0.566	0.566	0.566	0.566	0.566	0.566	0.566	0.566	0.566
mobilenet-adaptive-mean	0.441	0.455	0.453	0.459	0.469	0.500	0.518	0.560	0.571
mobilenet-finetuned-mean	0.468	0.453	0.483	0.464	0.486	0.503	0.514	0.578	0.573
mobilenet-pretrained-mean	0.556	0.556	0.556	0.556	0.556	0.556	0.556	0.556	0.556
resnet50-adaptive-mean	0.491	0.493	0.501	0.522	0.528	0.536	0.534	0.584	0.574
resnet50-finetuned-mean	0.486	0.466	0.477	0.533	0.518	0.562	0.587	**0.663**	**0.649**
resnet50-pretrained-mean	**0.604**	**0.604**	**0.604**	**0.604**	**0.604**	**0.604**	**0.604**	0.604	0.604

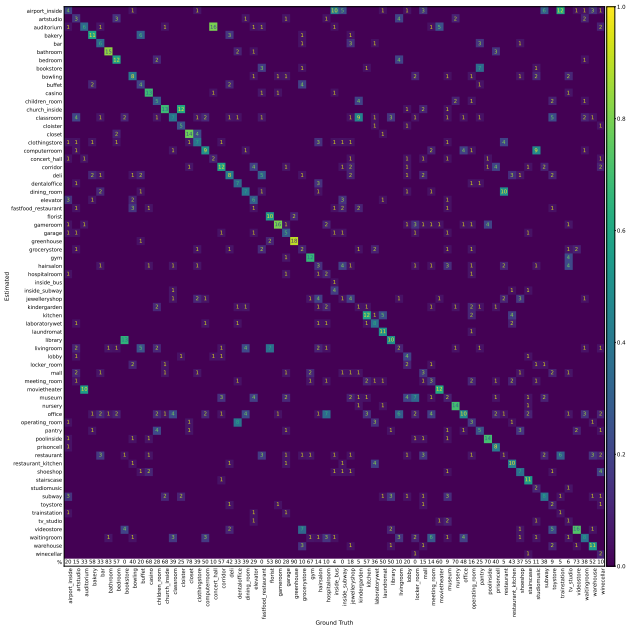


Figure C.1: CONFUSION MATRIX INDOOR SCENES - 1 SAMPLE. This figure shows the confusion matrix adopting the nearest neighbor classification with MEAN using the pre-trained network approach. ResNet-50 is utilized as pre-trained network on the Indoor Scenes dataset using 1 sample per class.

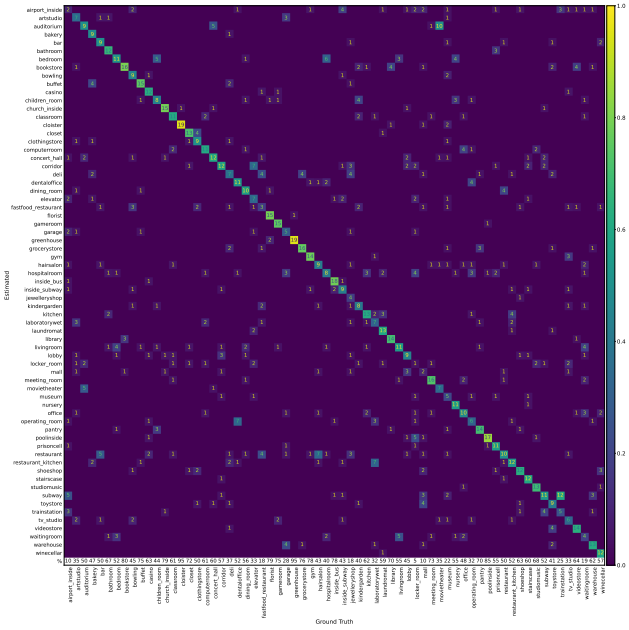


Figure C.2: CONFUSION MATRIX INDOOR SCENES - 3 SAMPLES. This figure shows the confusion matrix adopting the nearest neighbor classification with MEAN using the pre-trained network approach. ResNet-50 is utilized as pre-trained network on the Indoor Scenes dataset using 3 samples per class.

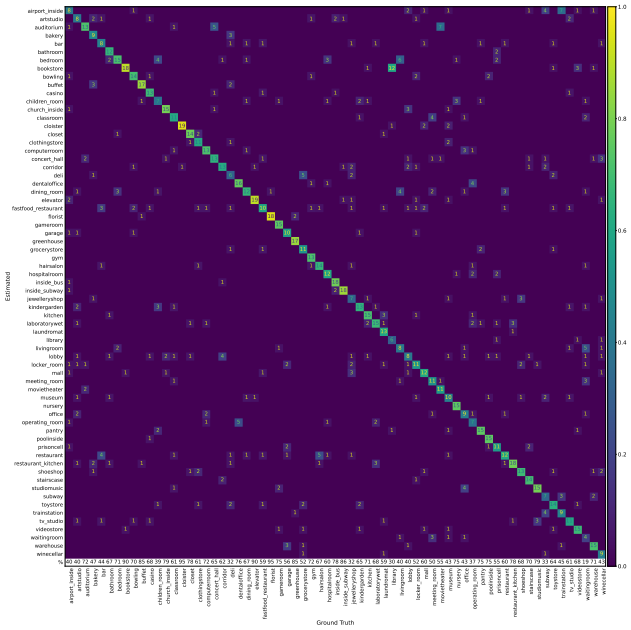


Figure C.3: CONFUSION MATRIX INDOOR SCENES - 11 SAMPLES. This figure shows the confusion matrix adopting the nearest neighbor classification with MEAN using the pre-trained network approach. ResNet-50 is utilized as pre-trained network on the Indoor Scenes dataset using 11 samples per class.

Appendix D

Office-31 Results

Table D.1: END-TO-END CLASSIFICATION PERFORMANCE OFFICE-31. End-to-end classification performance on the Office-31 dataset and pre-trained networks utilizing the fine-tuned and adapter network approach.

	1	2	3	4	6	8	12	31	44
alexnet-adaptive	0.037	0.070	0.120	0.177	0.403	0.447	0.607	0.727	0.753
alexnet-finetuned	0.410	0.473	0.537	0.557	0.653	0.640	0.713	0.760	0.763
densenet-adaptive	0.557	0.600	0.637	0.653	0.680	0.697	0.780	0.810	0.800
densenet-finetuned	0.350	0.480	0.450	0.557	0.620	0.630	0.667	0.797	0.830
mobilenet-adaptive	0.523	0.620	0.653	**0.733**	0.747	0.743	**0.810**	0.810	0.810
mobilenet-finetuned	0.267	0.460	0.537	0.587	0.667	0.703	0.720	0.807	0.820
resnet50-adaptive	**0.653**	**0.673**	**0.730**	0.720	**0.753**	**0.760**	0.787	**0.850**	0.830
resnet50-finetuned	0.443	0.533	0.590	0.573	0.710	0.730	0.773	0.840	**0.850**
vgg16-adaptive	0.053	0.087	0.310	0.330	0.483	0.583	0.687	0.783	0.813
vgg16-finetuned	0.553	0.590	0.613	0.637	0.680	0.720	0.737	0.803	0.817

Table D.2: ADAPTIVE CLASSIFICATION PERFORMANCE OFFICE-31. Deep Features classification performance on the Office-31 dataset and pre-trained networks utilizing the adapter network approach.

	1	2	3	4	6	8	12	31	44
alexnet-cos	0.113	0.150	0.190	0.210	0.323	0.380	0.507	0.647	0.703
alexnet-cos copy	0.113	0.150	0.190	0.210	0.323	0.380	0.507	0.647	0.703
alexnet-mean	0.113	0.150	0.197	0.200	0.343	0.400	0.530	0.657	0.690
alexnet-svm	0.090	0.120	0.210	0.210	0.343	0.367	0.523	0.687	0.717
densenet-cos	0.627	0.633	0.650	0.683	0.730	0.753	0.773	0.787	0.803
densenet-mean	0.627	0.643	0.653	0.660	0.713	0.700	0.757	0.760	0.770
densenet-svm	0.567	0.607	0.650	0.667	0.713	0.720	0.760	0.823	0.833
mobilenet-cos	0.573	0.667	0.667	0.717	0.753	0.723	0.780	0.833	0.820
mobilenet-mean	0.573	0.643	0.683	**0.750**	0.757	0.757	0.803	0.817	0.807
mobilenet-svm	0.397	0.617	0.630	0.657	0.713	0.713	**0.820**	0.813	0.843
resnet50-cos	**0.653**	**0.690**	0.703	0.733	0.747	**0.770**	0.777	**0.850**	0.837
resnet50-mean	**0.653**	0.687	**0.733**	0.747	**0.770**	0.753	0.783	0.810	0.830
resnet50-svm	0.613	0.677	0.707	0.743	0.750	0.757	0.783	0.820	**0.873**
vgg16-cos	0.137	0.183	0.247	0.377	0.507	0.540	0.610	0.747	0.783
vgg16-mean	0.137	0.170	0.257	0.383	0.500	0.533	0.610	0.747	0.777
vgg16-svm	0.120	0.193	0.230	0.370	0.477	0.513	0.590	0.760	0.777

Table D.3: FINE-TUNED CLASSIFICATION PERFORMANCE OFFICE-31. Deep Features classification performance on the Office-31 dataset and pre-trained networks utilizing the fine-tuned network approach.

	1	2	3	4	6	8	12	31	44
alexnet-cos	0.473	0.500	0.553	0.583	0.620	0.627	0.680	0.720	0.723
alexnet-mean	0.473	0.540	0.603	0.610	0.640	0.663	0.703	0.700	0.707
alexnet-svm	0.303	0.420	0.510	0.503	0.577	0.590	0.673	0.750	0.797
densenet-cos	0.513	0.510	0.570	0.603	0.660	0.653	0.713	0.823	0.850
densenet-mean	0.513	0.553	0.570	0.603	0.687	0.667	0.713	0.760	0.800
densenet-svm	0.470	0.550	0.597	0.643	0.720	0.667	0.710	0.837	**0.873**
mobilenet-cos	0.600	0.647	0.647	0.643	0.690	0.710	0.763	0.807	0.837
mobilenet-mean	0.600	0.630	0.643	0.663	0.707	0.740	0.757	0.760	0.773
mobilenet-svm	0.520	0.563	0.580	0.640	0.707	**0.747**	0.783	0.840	0.853
resnet50-cos	0.553	0.630	0.670	**0.690**	0.740	0.730	0.783	**0.847**	0.870
resnet50-mean	0.553	**0.657**	**0.693**	0.683	**0.763**	0.717	**0.790**	0.793	0.837
resnet50-svm	0.550	0.603	0.657	0.670	0.743	0.740	0.760	0.837	0.860
vgg16-cos	**0.613**	0.640	0.663	0.663	0.713	0.703	0.753	0.817	0.853
vgg16-mean	**0.613**	0.653	0.683	0.683	0.747	**0.747**	0.760	0.793	0.770
vgg16-svm	0.463	0.593	0.527	0.650	0.673	0.670	0.730	0.803	0.820

Table D.4: PRE-TRAINED CLASSIFICATION PERFORMANCE OFFICE-31. Deep Features classification performance on the Office-31 dataset and pre-trained networks.

	1	2	3	4	6	8	12	31	44
alexnet-cos	0.497	0.543	0.563	0.590	0.643	0.633	0.673	0.743	0.743
alexnet-mean	0.497	0.577	0.627	0.640	0.680	0.683	0.720	0.710	0.733
alexnet-svm	0.307	0.373	0.473	0.497	0.593	0.620	0.647	0.770	0.787
densenet-cos	0.610	0.683	0.703	0.703	0.723	0.743	0.763	0.843	0.857
densenet-mean	0.610	0.670	0.693	0.693	0.740	0.737	0.760	0.780	0.777
densenet-svm	0.580	0.660	0.667	0.730	0.740	0.730	0.783	0.853	**0.863**
mobilenet-cos	0.597	0.667	0.687	0.697	0.710	0.740	0.773	0.830	0.840
mobilenet-mean	0.597	**0.707**	0.730	**0.747**	0.753	0.777	0.793	0.793	0.790
mobilenet-svm	0.557	0.627	0.643	0.697	0.733	0.760	**0.810**	0.843	0.860
resnet50-cos	**0.657**	0.703	0.713	0.730	0.760	0.770	0.803	0.843	**0.863**
resnet50-mean	**0.657**	0.703	0.717	0.733	**0.777**	**0.783**	0.800	0.793	0.800
resnet50-svm	0.587	0.690	**0.740**	**0.747**	0.767	0.767	0.807	**0.867**	0.840
vgg16-cos	0.647	0.670	0.690	0.690	0.720	0.733	0.763	0.797	0.833
vgg16-mean	0.647	0.690	0.713	0.707	0.733	0.733	0.763	0.763	0.767
vgg16-svm	0.370	0.490	0.553	0.667	0.693	0.690	0.717	0.793	0.820

Table D.5: CONSTANT GALLERY OFFICE-31. Nearest neighbor classification performance with MEAN on the Office-31 dataset and pre-trained networks using the C-GAL approach.

	1	2	3	4	6	8	12	31	39
densenet-adaptive-mean	0.627	0.647	0.623	0.653	0.670	0.677	0.670	0.730	0.737
densenet-finetuned-mean	0.670	0.617	0.643	0.647	0.650	0.647	0.650	0.693	0.737
densenet-pretrained-mean	0.723	0.723	0.723	0.723	0.723	0.723	0.723	0.723	0.723
mobilenet-adaptive-mean	0.710	0.717	0.737	**0.763**	**0.793**	**0.760**	**0.790**	**0.800**	0.787
mobilenet-finetuned-mean	**0.760**	0.737	0.730	0.723	0.730	0.757	0.740	0.737	0.730
mobilenet-pretrained-mean	**0.760**	**0.760**	**0.760**	0.760	0.760	**0.760**	0.760	0.760	0.760
resnet50-adaptive-mean	0.740	0.747	0.753	**0.763**	0.777	0.747	0.783	0.770	**0.797**
resnet50-finetuned-mean	0.710	0.717	0.717	0.707	0.767	0.710	0.767	0.787	0.790
resnet50-pretrained-mean	0.747	0.747	0.747	0.747	0.747	0.747	0.747	0.747	0.747

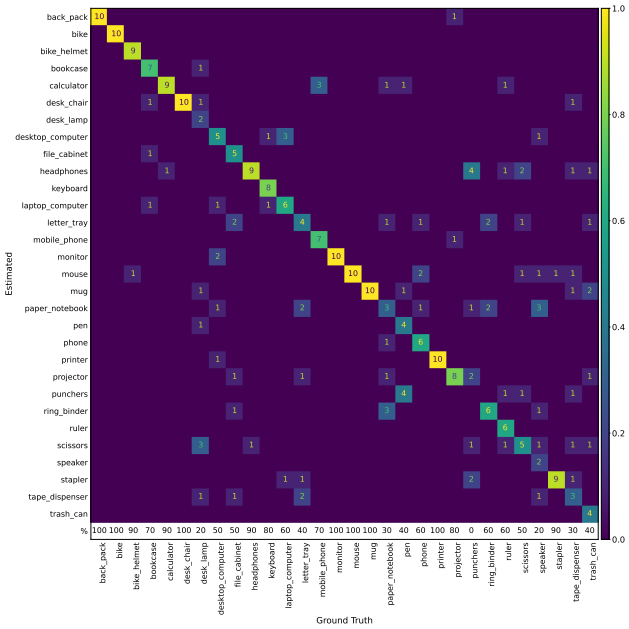


Figure D.1: CONFUSION MATRIX OFFICE-31 - 1 SAMPLE. This figure shows the confusion matrix adopting the nearest neighbor classification with MEAN using the pre-trained network approach. ResNet-50 is utilized as pre-trained network on the Office-31 dataset using 1 sample per class.

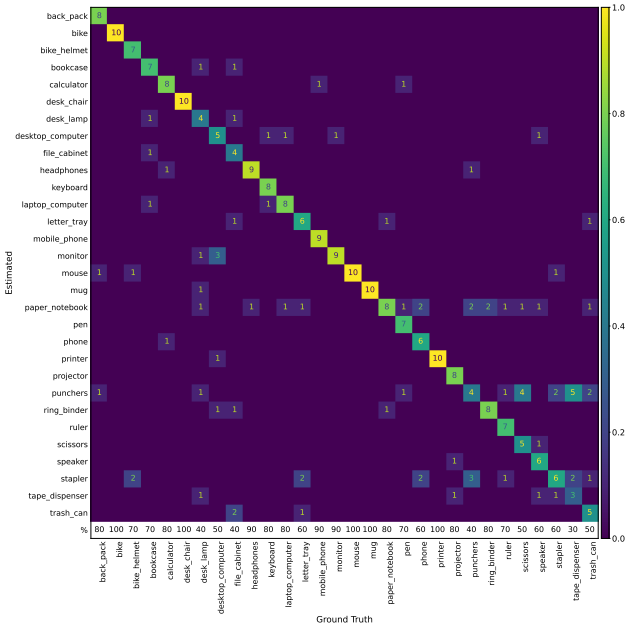


Figure D.2: CONFUSION MATRIX OFFICE-31 - 3 SAMPLES. This figure shows the confusion matrix adopting the nearest neighbor classification with MEAN using the pre-trained network approach. ResNet-50 is utilized as pre-trained network on the Office-31 dataset using 3 samples per class.

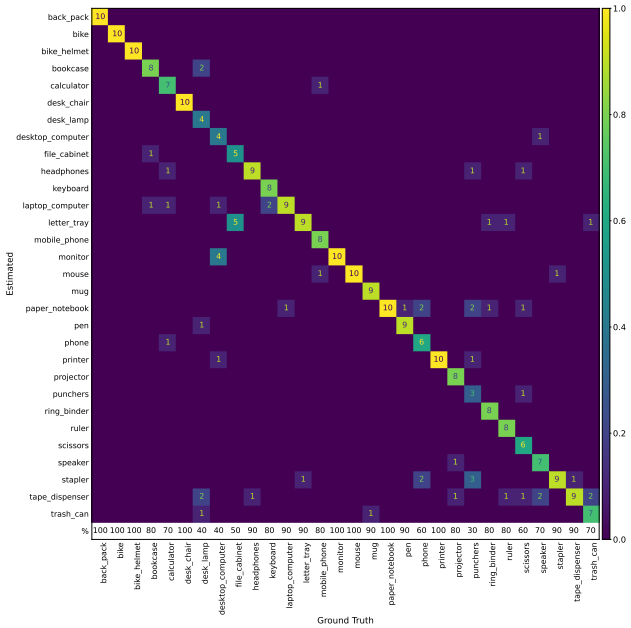


Figure D.3: CONFUSION MATRIX OFFICE-31 - 12 SAMPLES. This figure shows the confusion matrix adopting the nearest neighbor classification with MEAN using the pre-trained network approach. ResNet-50 is utilized as pre-trained network on the Office-31 dataset using 12 samples per class.

Appendix E

Virus Results

Table E.1: END-TO-END CLASSIFICATION PERFORMANCE VIRUS. End-to-end classification performance on the Virus dataset and pre-trained networks utilizing the fine-tuned and adapter network approach.

	1	2	3	4	5	7	9	23	31
alexnet-adaptive	0.058	0.051	0.050	0.045	0.069	0.057	0.086	0.156	0.199
alexnet-finetuned	0.087	0.198	0.183	0.163	0.189	0.222	0.231	0.302	0.327
densenet-adaptive	0.091	0.214	0.279	**0.332**	0.286	0.310	**0.362**	0.370	0.473
densenet-finetuned	0.073	0.149	0.188	0.072	0.196	0.195	0.238	0.328	0.361
mobilenet-adaptive	**0.132**	0.247	**0.298**	0.226	0.285	0.346	0.353	0.412	0.418
mobilenet-finetuned	0.122	0.199	0.292	0.257	**0.330**	0.303	0.352	0.423	**0.479**
resnet50-adaptive	0.078	**0.254**	0.243	0.283	0.322	**0.358**	0.340	0.405	0.438
resnet50-finetuned	0.021	0.196	0.247	0.234	0.306	0.302	0.307	0.405	0.372
vgg16-adaptive	0.049	0.048	0.067	0.074	0.079	0.073	0.135	0.186	0.242
vgg16-finetuned	0.129	0.184	0.183	0.241	0.219	0.298	0.333	**0.429**	0.471

Table E.2: ADAPTIVE CLASSIFICATION PERFORMANCE VIRUS. Deep Features classification performance on the Virus dataset and pre-trained networks utilizing the adapter network approach.

	1	2	3	4	5	7	9	23	31
alexnet-cos	0.100	0.100	0.098	0.109	0.140	0.167	0.185	0.312	0.332
alexnet-mean	0.100	0.114	0.099	0.118	0.132	0.159	0.178	0.300	0.315
alexnet-svm	0.080	0.102	0.102	0.124	0.141	0.180	0.180	0.358	0.360
densenet-cos	0.203	0.281	0.284	0.364	0.369	0.414	0.414	0.466	0.473
densenet-mean	0.203	0.273	0.309	0.355	0.359	0.419	0.406	0.478	0.478
densenet-svm	0.185	0.314	0.301	0.328	0.373	0.432	0.434	0.504	0.488
mobilenet-cos	**0.226**	**0.316**	0.330	0.348	0.400	0.387	0.427	0.507	0.512
mobilenet-mean	**0.226**	0.307	**0.332**	0.339	0.340	0.384	0.436	0.508	0.506
mobilenet-svm	0.192	0.305	0.325	**0.367**	0.360	0.398	**0.460**	**0.544**	**0.553**
resnet50-cos	0.187	0.304	0.309	0.338	0.381	0.424	0.400	0.508	0.496
resnet50-mean	0.187	0.290	0.298	0.339	0.371	0.380	0.395	0.488	0.451
resnet50-svm	0.154	**0.316**	0.307	0.344	**0.394**	**0.435**	0.440	0.528	0.503
vgg16-cos	0.064	0.074	0.079	0.107	0.147	0.196	0.233	0.336	0.332
vgg16-mean	0.064	0.075	0.084	0.109	0.148	0.188	0.217	0.331	0.331
vgg16-svm	0.061	0.071	0.075	0.118	0.170	0.196	0.241	0.382	0.399

Table E.3: FINE-TUNED CLASSIFICATION PERFORMANCE VIRUS. Deep Features classification performance on the Virus dataset and pre-trained networks utilizing the fine-tuned network approach.

	1	2	3	4	5	7	9	23	31
alexnet-cos	0.209	0.259	0.220	0.264	0.318	0.346	0.381	0.469	0.480
alexnet-mean	0.209	0.249	0.238	0.276	0.291	0.350	0.415	0.478	0.514
alexnet-svm	0.140	0.172	0.184	0.253	0.270	0.340	0.382	0.512	0.486
densenet-cos	0.134	0.284	0.316	0.155	0.327	0.337	0.411	0.497	0.532
densenet-mean	0.134	0.296	0.299	0.162	0.287	0.322	0.379	0.513	0.491
densenet-svm	0.110	0.300	0.298	0.195	0.341	0.357	0.430	0.515	0.564
mobilenet-cos	**0.250**	0.334	0.377	**0.347**	0.449	0.428	**0.515**	0.568	0.563
mobilenet-mean	**0.250**	**0.342**	**0.421**	0.340	**0.461**	0.420	0.501	0.567	0.548
mobilenet-svm	0.222	0.316	0.390	0.327	0.419	0.433	0.486	0.568	**0.618**
resnet50-cos	0.243	0.307	0.363	0.345	0.387	0.443	0.428	0.557	0.564
resnet50-mean	0.243	0.309	0.373	0.300	0.352	0.428	0.409	**0.573**	0.528
resnet50-svm	0.232	0.316	0.370	0.340	0.347	**0.449**	0.431	0.555	0.550
vgg16-cos	0.203	0.248	0.289	0.316	0.327	0.357	0.427	0.552	0.525
vgg16-mean	0.203	0.246	0.289	0.309	0.302	0.364	0.406	0.565	0.553
vgg16-svm	0.153	0.241	0.253	0.255	0.278	0.381	0.457	0.557	0.575

Table E.4: PRE-TRAINED CLASSIFICATION PERFORMANCE VIRUS. Deep Features classification performance on the Virus dataset and pre-trained networks.

	1	2	3	4	5	7	9	23	31
alexnet-cos	**0.238**	0.258	0.250	0.277	0.307	0.330	0.353	0.409	0.401
alexnet-mean	**0.238**	0.241	0.263	0.276	0.318	0.345	0.393	0.429	0.419
alexnet-svm	0.171	0.200	0.205	0.219	0.240	0.327	0.344	0.437	0.439
densenet-cos	0.232	0.328	0.328	0.357	0.386	0.405	0.425	0.491	0.495
densenet-mean	0.232	0.334	0.346	0.395	0.411	0.422	0.431	0.477	0.480
densenet-svm	0.202	0.332	0.346	0.370	0.408	0.434	0.473	0.552	0.572
mobilenet-cos	0.211	**0.336**	0.348	0.391	**0.448**	0.463	0.492	0.543	0.534
mobilenet-mean	0.211	0.322	0.346	0.372	0.406	0.454	0.481	0.548	0.584
mobilenet-svm	0.199	0.326	0.336	0.396	0.442	**0.489**	**0.523**	**0.631**	**0.640**
resnet50-cos	0.168	0.289	0.321	0.356	0.377	0.398	0.429	0.486	0.493
resnet50-mean	0.168	0.298	0.339	0.368	0.393	0.430	0.426	0.483	0.458
resnet50-svm	0.163	0.322	**0.361**	**0.398**	0.419	0.470	0.475	0.532	0.527
vgg16-cos	0.215	0.254	0.260	0.301	0.320	0.331	0.340	0.388	0.392
vgg16-mean	0.215	0.267	0.291	0.279	0.301	0.339	0.350	0.438	0.415
vgg16-svm	0.154	0.205	0.214	0.250	0.275	0.324	0.360	0.442	0.465

Table E.5: CONSTANT GALLERY VIRUS. Nearest neighbor classification performance with MEAN on the Virus dataset and pre-trained networks using the C-GAL approach.

	1	2	3	4	5	7	9	23	26
densenet-adaptive-mean	0.385	0.388	0.398	0.408	0.416	0.434	0.446	0.476	0.455
densenet-finetuned-mean	0.369	0.408	0.488	0.201	0.388	0.370	0.451	0.517	0.503
densenet-pretrained-mean	0.504	0.504	0.504	0.504	0.504	0.504	0.504	0.504	0.504
mobilenet-adaptive-mean	0.506	0.445	0.470	0.469	0.474	0.454	0.511	0.515	0.507
mobilenet-finetuned-mean	0.547	0.562	0.550	0.512	0.546	0.516	0.549	**0.631**	**0.590**
mobilenet-pretrained-mean	**0.588**	**0.588**	**0.588**	**0.588**	**0.588**	**0.588**	**0.588**	0.588	0.588
resnet50-adaptive-mean	0.408	0.402	0.397	0.401	0.396	0.393	0.423	0.464	0.458
resnet50-finetuned-mean	0.519	0.486	0.473	0.476	0.491	0.487	0.515	0.590	0.549
resnet50-pretrained-mean	0.516	0.516	0.516	0.516	0.516	0.516	0.516	0.516	0.516

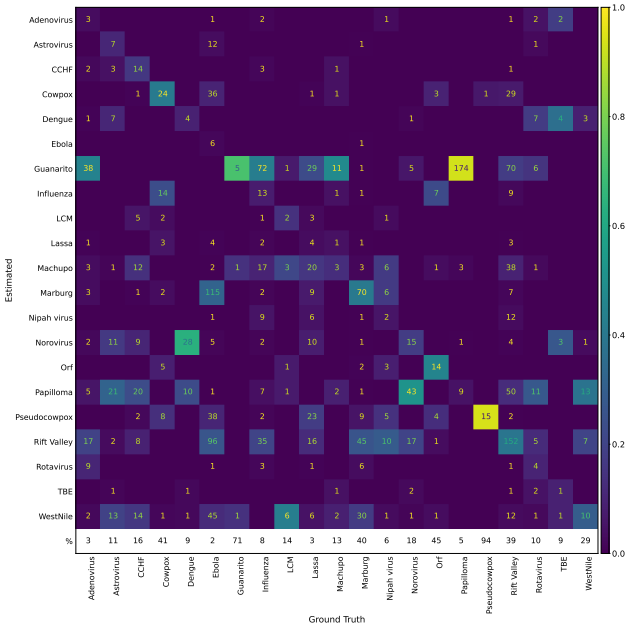


Figure E.1: CONFUSION MATRIX VIRUS - 1 SAMPLE. This figure shows the confusion matrix adopting the nearest neighbor classification with MEAN using the pre-trained network approach. DenseNet-121 is utilized as pre-trained network on the Virus dataset using 1 sample per class.

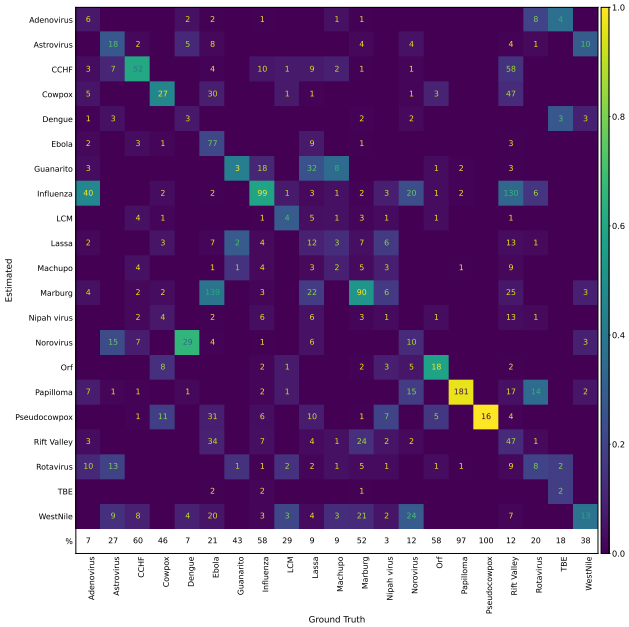


Figure E.2: CONFUSION MATRIX VIRUS - 3 SAMPLES. This figure shows the confusion matrix adopting the nearest neighbor classification with MEAN using the pre-trained network approach. DenseNet-121 is utilized as pre-trained network on the Virus dataset using 3 samples per class.

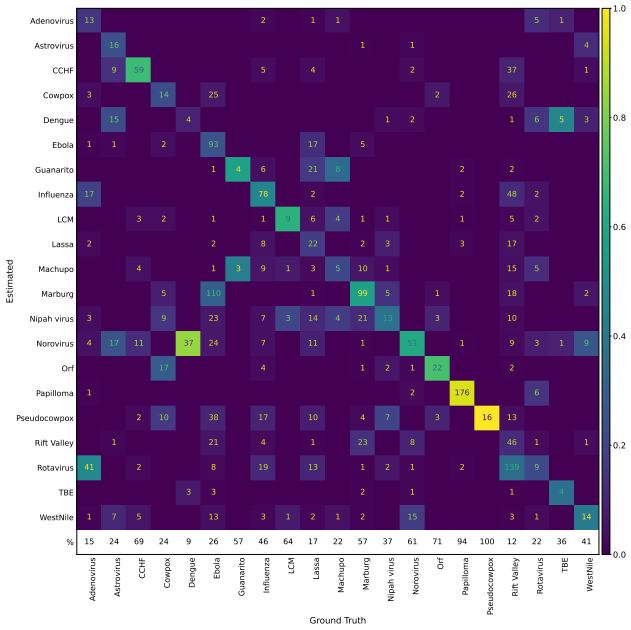


Figure E.3: CONFUSION MATRIX VIRUS - 9 SAMPLES. This figure shows the confusion matrix adopting the nearest neighbor classification with MEAN using the pre-trained network approach. DenseNet-121 is utilized as pre-trained network on the Virus dataset using 9 samples per class.

List of Figures

List of Tables

Bibliography

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Cıbuk, M., Budak, U., Guo, Y., Cevdet Ince, M., and Sengur, A. (2019). Efficient deep features selections and classification for flower species recognition. *Measurement*, 137:7–13.

Day, O. and Khoshgoftaar, T. M. (2017). A survey on heterogeneous transfer learning. *Journal of Big Data*, 4(1):29.

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. ISSN: 1063-6919.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874.

Guo, Y., Li, Y., Wang, L., and Rosing, T. (2020). Adafilter: Adaptive filter fine-tuning for deep transfer learning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 4060–4066. AAAI Press.

Guo, Y., Shi, H., Kumar, A., Grauman, K., Rosing, T., and Feris, R. S. (2019). Spottune: Transfer learning through adaptive fine-tuning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 4805–4814. Computer Vision Foundation / IEEE.

Günther, M., Dhamija, A. R., and Boult, T. E. (2020). Watchlist Adaptation: Protecting the Innocent. In *2020 International Conference of the Biometrics Special Interest Group (BIOSIG)*, pages 1–7. ISSN: 1617-5468.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. ISSN: 1063-6919.

Howard, A., Pang, R., Adam, H., Le, Q. V., Sandler, M., Chen, B., Wang, W., Chen, L., Tan, M., Chu, G., Vasudevan, V., and Zhu, Y. (2019). Searching for mobilenetv3. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 1314–1324. IEEE.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kornblith, S., Shlens, J., and Le, Q. V. (2019). Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

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. Curran Associates, Inc.

Maji, S., Rahtu, E., Kannala, J., Blaschko, M. B., and Vedaldi, A. (2013). Fine-grained visual classification of aircraft. *CoRR*, abs/1306.5151.

Maqsood, M., Nazir, F., Khan, U., Aadil, F., Jamal, H., Mehmood, I., and Song, O.-y. (2019). Transfer Learning Assisted Classification and Detection of Alzheimer's Disease Stages Using 3D MRI Scans. *Sensors*, 19(11):2645. Number: 11 Publisher: Multidisciplinary Digital Publishing Institute.

Matuszewski, D. J. and Sintorn, I.-M. (2021). TEM virus images: Benchmark dataset and deep learning classification. *Computer Methods and Programs in Biomedicine*, 209:106318.

Neyshabur, B., Sedghi, H., and Zhang, C. (2020). What is being transferred in transfer learning? In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 512–523. Curran Associates, Inc.

O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *CoRR*, abs/1511.08458.

Pan, S. J. and Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359. Conference Name: IEEE Transactions on Knowledge and Data Engineering.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Quattoni, A. and Torralba, A. (2009). Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 413–420. ISSN: 1063-6919.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252.

Saenko, K., Kulis, B., Fritz, M., and Darrell, T. (2010). Adapting Visual Category Models to New Domains. In Daniilidis, K., Maragos, P., and Paragios, N., editors, *Computer Vision – ECCV 2010*, Lecture Notes in Computer Science, pages 213–226, Berlin, Heidelberg. Springer.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2020). Grad-cam: Visual explanations from deep networks via gradient-based localization. *Int. J. Comput. Vis.*, 128(2):336–359.

Seth, K. (2020). Fruits and Vegetables Image Recognition Dataset. *Kaggle*.

Shaha, M. and Pawar, M. (2018). Transfer Learning for Image Classification. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 656–660.

Shao, L., Zhu, F., and Li, X. (2015). Transfer Learning for Visual Categorization: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 26(5):1019–1034. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. *CoRR*, abs/1803.09820.

Soekhoe, D., van der Putten, P., and Plaat, A. (2016). On the Impact of Data Set Size in Transfer Learning Using Deep Neural Networks. In Boström, H., Knobbe, A., Soares, C., and Papapetrou, P., editors, *Advances in Intelligent Data Analysis XV*, Lecture Notes in Computer Science, pages 50–60, Cham. Springer International Publishing.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., and Liu, C. (2018). A Survey on Deep Transfer Learning. In Kůrková, V., Manolopoulos, Y., Hammer, B., Iliadis, L., and Maglogiannis, I., editors, *Artificial Neural Networks and Machine Learning – ICANN 2018*, Lecture Notes in Computer Science, pages 270–279, Cham. Springer International Publishing.

Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. (2017). Adversarial discriminative domain adaptation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2962–2971. IEEE Computer Society.

Vrbančič, G. and Podgorelec, V. (2020). Transfer Learning With Adaptive Fine-Tuning. *IEEE Access*, 8:196197–196211. Conference Name: IEEE Access.

Weiss, K., Khoshgoftaar, T. M., and Wang, D. (2016). A survey of transfer learning. *Journal of Big Data*, 3(1):9.

Weston, J. and Watkins, C. (1998). Multi-class support vector machines. *Royal Holloway College, Univ. London, U.K., Tech. Rep.*

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Yosinski, J., Clune, J., Nguyen, A. M., Fuchs, T. J., and Lipson, H. (2015). Understanding neural networks through deep visualization. *CoRR*, abs/1506.06579.

Zhu, W., Braun, B., Chiang, L. H., and Romagnoli, J. A. (2021). Investigation of transfer learning for image classification and impact on training sample size. *Chemometrics and Intelligent Laboratory Systems*, 211:104269.

Zhu, Y., Chen, Y., Lu, Z., Pan, S. J., Xue, G.-R., Yu, Y., and Yang, Q. (2011). Heterogeneous Transfer Learning for Image Classification. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. (2021). A Comprehensive Survey on Transfer Learning. *Proceedings of the IEEE*, 109(1):43–76. Conference Name: Proceedings of the IEEE.