Addressing Continual Learning and Data Privacy Challenges with an explainable kNN-based Image Classifier

Bachelor Thesis

Bachelor of Science in Applied Computer Science

Tobias Archut

October 6, 2023

Supervisor:

1st: Prof. Dr. Christian Ledig
2nd: Sebastian Dörrich M. Sc.

Chair of Explainable Machine Learning
Faculty of Information Systems and Applied Computer Sciences
Otto-Friedrich-University Bamberg

Abstract

Current deep learning-based image classification models store their learned knowledge implicitly in deep neural networks. This implicit knowledge can only be adjusted by retraining the entire model, which can lead to catastrophic forgetting of previously learned knowledge and can be challenging for applying personal user data privacy regulations.

Nakata et al. proposed a kNN-based image classification system that claims to overcome these challenges. During this thesis, a prototype system was developed based on their description to evaluate their approach.

Overall, the prototype system almost matches the classification performance of Nakata et al.'s system, just falling short by 0.006% to 2.8% in accuracy for the ResNet-50 image encoder and a decrease in accuracy of 4% to 12.5% for the ViT-B/16 image encoder. Dimensionality reduction with t-SNE can increase the classification performance and visualize the model's decision-making process. Catastrophic forgetting was mitigated, and it has been shown that it is possible to delete up to 40% of feature embeddings of a single class before the classification accuracy declines.

Investigating Nakata et al.'s approach by building a prototype system confirmed their results. The prototype implementation showed that addressing continual learning and data privacy challenges with an explainable kNN-based image classifier is possible.

Übersicht

Aktuelle auf Deep Learning basierende Bildklassifizierungsmodelle speichern ihr erlerntes Wissen implizit in tiefen neuronalen Netzen. Dieses implizite Wissen kann nur angepasst werden, indem das gesamte Modell neu trainiert wird, was zum so genannten katastrophalen Vergessen des zuvor gelernten Wissens führen kann und eine Herausforderung für die Anwendung der Datenschutzbestimmungen für persönliche Nutzerdaten darstellt.

Nakata et al. haben ein kNN-basiertes Bildklassifizierungssystem vorgeschlagen, das diese Herausforderungen überwinden soll. Im Rahmen dieser Arbeit wurde ein Prototypsystem auf der Grundlage ihrer Beschreibung entwickelt, um ihren Ansatz zu evaluieren.

Insgesamt erreicht das Prototypsystem fast die Klassifizierungsgenauigkeit des Systems von Nakata et al. und liegt nur um 0,006% bis 2,8% in der Genauigkeit für den ResNet-50-Bildkodierer und um 4% bis 12,5% für den ViT-B/16-Bildkodierer darunter. Eine Dimensionalitätsreduktion mit t-SNE kann die Klassifikationsleistung erhöhen und den Entscheidungsprozess des Modells visualisieren. Das katastrophale Vergessen wurde vermieden, und es hat sich gezeigt, dass bis zu 40% der Merkmalseinbettungen einer einzelnen Klasse gelöscht werden können, bevor die Klassifizierungsgenauigkeit abnimmt.

Die Untersuchung des Ansatzes von Nakata et al. durch den Aufbau eines Prototypsystems bestätigte ihre Resultate. Die Prototyp-Implementierung zeigte, dass es möglich ist, die Herausforderungen des kontinuierlichen Lernens und des Datenschutzes mit einem erklärbaren kNN-basierten Bildklassifikator zu bewältigen.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

AI	Artificial intelligence
ANN	Artificial neural network
CNN	Convolutional neural network
DB	Database
DL	Deep learning
GDPR	General Data Protection Regulation
ID	Identification
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
kNN	k-nearest neighbors algorithm
LDA	Linear discriminant analysis
ML	Machine Learning
PCA	Principal component analysis
PX	Pixel
ReLU	Rectified linear unit
t-SNE	t-distributed stochastic neighbor embedding
UMAP	Uniform Manifold Approximation and Projection
ViT	Vision Transformer
XAI	Explainable AI

1 Introduction

1.1 Context and Motivation

Combining the k-nearest neighbors (kNN) algorithm from 1951 with the most recent advancements in deep learning leads to a new image classification model, which is privacy-focused, explainable, and cost-effective. The kNN classifier is a machine learning algorithm that classifies data points based on their similarity to other data points.

Kengo Nakata et al. described this approach in the paper "Revisiting a kNN-based Image Classification System with High-capacity Storage" (Nakata et al., 2022). This bachelor thesis aims to rebuild a prototype system of Nakata et al.'s approach to verify their results and perform additional experiments to understand the proposed system's limits and strengths.

Deep learning has shown its use cases for multiple applications, especially domains where it is hard for humans to grasp the significant features of data. For example, deep learning has been used to achieve accurate solutions in the field of medical image analysis (Puttagunta and Subban, 2021) and medical drug discovery (Chen et al., 2018). Besides the medical field, deep learning has proven its accuracy and reliability in image classification, object detection, and natural language processing (Pouyanfar et al., 2019).

However, deep learning models have some limitations. One limitation is that training large models requires high computational power, which results in high costs for training deep learning models from scratch (Alzubaidi et al., 2021). Another limitation is that they can be challenging to interpret, making it difficult to trust their results (Parisi et al., 2019; Goodfellow et al., 2016). Additionally, deep learning models can be prone to so-called catastrophic forgetting, which is the phenomenon of losing previously learned knowledge when new information is added to the model (Xie et al., 2021; Alzubaidi et al., 2021).

1.2 Related Work

To address these limitations, researchers have proposed several techniques for continual learning, which is the ability of a deep learning model to learn new information without forgetting the old information. For example, one such technique is incremental learning, which involves updating the model with new data one instance at a time. This helps to prevent the model from forgetting the old data (Lange et al., 2022). Besides the advancements in continual learning, in 2022, van de Ven et al. concluded that current continual learning techniques have multiple limitations, for example, being computationally expensive or requiring a large amount of memory (van de Ven et al., 2022).

The continued development of continual learning techniques is a critical area of research. These techniques have the potential to overcome catastrophic forgetting and make deep learning a more powerful tool for a broader range of applications.

The project described in this bachelor thesis aims to investigate a way to enable architecture-based continual learning for image classification by simultaneously allowing the deletion of data from the system without causing a severe performance deterioration of the model. This is a challenging problem, but it is important to be solved, especially for real-world applications using sensitive user data, which must adhere to privacy protection laws.

To address these issues, Nakata et al. proposed a system that combines an image encoder from a convolutional neural network (CNN) with the k-nearest neighbor (kNN) approach (Nakata et al., 2022). A visualization of the proposed system is provided in figure 8. The image encoder, which extracts the features from the given pictures, has already been trained for image classification on publicly available data such as ImageNet. Various pretrained CNNs are available. Typically, they can be downloaded and used free of cost from commonly used deep learning Python libraries, such as torchvision from PyTorch (Paszke et al., 2019) and TensorFlow (Abadi et al., 2015). Hence, the cost-intensive training phase of a new CNN is not required.

The image encoder can learn to extract features such as the shape, texture, and color of depicted objects that are relevant to the specific task, such as classification. The features extracted by the image encoder are then stored as multi-dimensional vectors in a high-capacity storage system and used by the kNN classifier to make predictions. The kNN classifier predicts the class of a data point based on the similarity to its surrounding labeled data points. Using kNN in conjunction with a high-capacity storage system allows for the fast and simple addition and deletion of features extracted from new images without having to retrain a CNN model. Especially highlighting the possibilities of deletion of images and their corresponding extracted features in accordance with privacy measures such as the European GDPR. (European Parliament and Council of the European Union, 2016) The GDPR states that at any point in time, individuals are able to request the removal of their personal data from company databases, including trained machine learning and deep learning models.

The advantage of Nakata et al.'s approach is that it combines the power of deep learning with the flexibility of kNN. Nakata et al. claim that the proposed approach has the following potential benefits:

1. *Mitigating catastrophic forgetting*: Learning new information is possible without forgetting the old information.

2. *Accordance to privacy laws*: Data can be deleted from the system without causing severe performance deterioration of the model.

3. *Explainability*: The model's decision can be evaluated by visually comparing the k nearest neighbors with the data sample in question.

4. *Cost-Effective*: The described approach is inexpensive because no training of a new CNN is needed.

The proposed approach by Nakata et al. is still under development but could potentially contribute to the field of continual learning significantly.

1.3 Contribution

Nakata et al. did not publish their source code, which has limited the ability of other researchers to reproduce their results and build upon their work. This bachelor's thesis aims to address this gap by developing a prototype system based on Nakata et al.'s approach. The prototype system will be used to verify Nakata et al.'s results and explore the potential benefits of their approach. To understand the proposed system's limits and strengths, additional experiments, such as evaluating the hyperparameters of the prototype system, assessing the impact of catastrophic forgetting, and investigating the feasibility of privacy-preserving deletion of support set feature embeddings, will be performed. The model's decision process will be visualized using different dimensionality reduction methods. Additionally, the system's usefulness for medical image analysis is explored by practical research with medical datasets.

2 Theoretical Foundations

2.1 Machine Learning Introduction

Machine learning (ML) is a subfield of Artificial Intelligence (AI) and is closely related to statistics. The following section shall provide an overview of ML by starting with the statistical foundations, continuing with a concise definition of the term ML, and finishing with a detailed look into the kNN algorithm.

From Statistics to Machine Learning Machine learning and statistics are closely related fields that influence each other. Statistics provides the theoretical foundation for machine learning, and multiple ML algorithms have been introduced as applied statistics methods. (James et al., 2013) However, there are some key differences between the two. Machine learning is a subfield of artificial intelligence (AI) that mainly describes algorithms that enable computers to learn relations in data without being explicitly programmed (Mitchell, 1997). It does this by using statistical methods to analyze data and identify patterns (Mitchell, 1997). On the other hand, statistics is the science of collecting, analyzing, interpreting, and presenting data. It provides tools and techniques for understanding data, making inferences, and making predictions. (James et al., 2013) For example, the kNN algorithm is a machine learning algorithm that uses statistics to find the k most closely adjacent data points to a new data point in a multidimensional space. Both fields use data to identify relations of samples in a dataset to generalize and allow inference. (James et al., 2013) However, the approach to identifying relations is different. Machine learning algorithms learn from data by identifying patterns, while statistical methods learn from data by making assumptions about the underlying distribution of the data (James et al., 2013; Mitchell, 1997).

Despite their differences, machine learning and statistics are complementary fields. Machine learning can be used to automate tasks that would be difficult or time-consuming to do manually, while statistics can also be used to evaluate the performance of machine learning models. For example, statistical methods can be used to calculate a machine learning model's accuracy (proportion of all predictions that are correct), precision (proportion of positive predictions that are actually correct), and recall (proportion of actual positives that are correctly predicted) (Sammut and Webb, 2017). These metrics help determine how well the ML model is performing and whether it is ready to be used in production.

Machine Learning Tom Mitchell, a computer scientist and AI researcher, coined the concise definition of Machine learning as follows: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." (Mitchell, 1997)

This definition highlights the three main aspects of machine learning:

- Experience E: Machine learning algorithms learn from data. The data can be labeled, which means that each sample is identified with a target value, or it can be unlabeled.
- Tasks T: Machine learning algorithms are designed to perform specific tasks, such as classification, regression, or clustering.
- Performance measure P: The performance of a machine learning algorithm is measured by a metric, such as accuracy, precision, or recall. The definition emphasizes that machine learning algorithms shall improve their performance over time by learning from the given data.

k-Nearest Neighbors (kNN) The k-nearest neighbors (kNN) algorithm is a machine learning algorithm that can be used for classification and regression tasks. The kNN algorithm is an instance-based learning algorithm, which means that the generalization is postponed until a new instance (new data point) is classified (Mitchell, 1997). It works by finding the k most similar data points to a new data point, called the query point, and then using the labels of those k points to predict the label of the query point. The similarity between two data points in vector space is typically measured using a distance metric, such as cosine similarity (cf. Cosine Similarity 2.1). The k data points that are closest to the query point are called the k-nearest neighbors. A majority vote on the k-nearest neighbors then predicts the label of the query point.

Cosine Similarity Cosine similarity is a measure of similarity between two vectors in a vector space. It is calculated by dividing the dot product of the two vectors by the product of their magnitudes.

$$\text{cosine similarity} = S_C(X, Y) = \frac{\mathbf{X} \cdot \mathbf{Y}}{\|\mathbf{X}\|\|\mathbf{Y}\|} = \frac{\sum_{i=1}^{n} X_i Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2 \cdot \sum_{i=1}^{n} Y_i^2}}$$

The cosine similarity of two vectors can range from -1 to 1, with -1 indicating perfect dissimilarity and 1 indicating perfect similarity.

The kNN classification process is visualized in Figure 1.

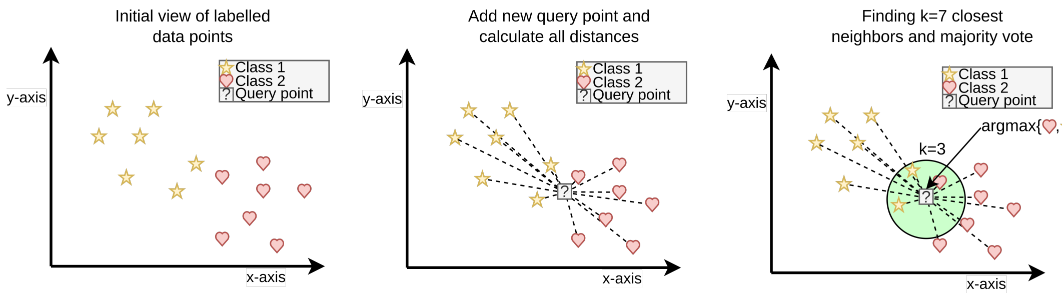
Initial view of labelled data points
Add new query point and calculate all distances
Finding k=7 closest neighbors and majority vote
Class 1
Class 2
Query point
y-axis
x-axis
k=3
argmax{♡,☆,☆} = ☆

Figure 1: Visualization of kNN classification process. Adapted from Navlani (2018)

Pseudocode kNN algorithm Algorithm 1 shows a pseudocode representation of the simple kNN training phase, which essentially stores all training examples in a data structure (or a database). (Mitchell, 1997) Algorithm 2 shows the classification phase of kNN, thus the inference. It includes the distance calculation with cosine distance in line 2 and the inference by majority vote in line 6.

Algorithm 1 k-Nearest Neighbors Training. By (Mitchell, 1997)

```
Require: Training data $\mathbb{X}$, corresponding labels $y^{(i)}$
Ensure: Data structure $\mathcal{D}$ which contains all training examples with their labels
1: for $\boldsymbol{x}^{(i)}$ in $\mathbb{X}$ do
2:     store tuple $\langle \boldsymbol{x}^{(i)}, y^{(i)} \rangle$ in data structure $\mathcal{D}$    ▷ Could also be a database
3: end for
```

Nakata et al. state that they used a distance measure based on cosine similarity. Their given formula is cosine distance, which is the complement of cosine similarity. Cosine distance $= 1 - S_C(X, Y)$ (Chomboon et al., 2015). Cosine distance ranges from 0 to 2, which makes it more convenient to use due to the embedding in positive space.

Applying Tom Mitchell's ML definition to the kNN algorithm results in the following:

- Experience E: The experience of the kNN algorithm is the training data. The training data consists of a set of data points, each of which has a label.

Algorithm 2 k-Nearest Neighbors Classification. Adapted from (Mitchell, 1997)

```
Require: new data point $x_q$, number of neighbors $k$, $\mathcal{D}$ set of tupels $\langle \boldsymbol{x}^{(i)}, y^{(i)} \rangle$ (cf.
    k-Nearest Neighbors Training Algorithm 1)
Ensure: Classification label of $x_q$ named $\hat{y_q}$
1: for tuple $\langle \boldsymbol{x}^{(i)}, y^{(i)} \rangle$ in data structure $\mathcal{D}$ do
2:     calculate cosine distance $d^{(i)}$ between $x_q$ and $\boldsymbol{x}^{(i)}$   ▷ cf. Cosine Similarity 2.1
3:     store tuple $\langle d^{(i)}, y^{(i)} \rangle$ in list $\mathcal{L}$
4: end for
5: Sort list $\mathcal{L}$ by distance $d^{(i)}$ lowest to highest
6: $\hat{y_q} = \text{argmax}_{l \in \mathcal{L}} \sum_{n=1}^{k} \mathcal{L}^n.y^{(i)}$   ▷ majority vote on k nearest neighbours
```

- Tasks T: The tasks that kNN can perform are classification and regression. In classification, the task is to predict the label of a new data point. In regression, the task is to predict the value of a continuous variable for a new data point.
- Performance measure P: Accuracy is the performance measure for kNN. It is the percentage of data points that are correctly classified.
- Improves with experience E: The kNN algorithm improves its performance with experience E by learning from the training data. Thus, this generally allows the algorithm to make better predictions for new data points.

The kNN algorithm is a simple and intuitive algorithm, but it can be very effective in practice. It is often used for tasks where the data is not linearly separable, such as text classification (Soucy and Mineau, 2001) and natural language processing (Mikolov et al., 2013). The kNN algorithm is also a robust algorithm that is not sensitive to outliers. This makes it a good choice for tasks where the data may be noisy or contaminated with outliers. According to Tom Mitchell, kNN is a so-called "lazy learning" ML algorithm. A lazy learning algorithm is a type of machine learning algorithm that does not build a predictive model during training but instead stores the training data and makes predictions at inference time by comparing the new data to the stored data (Mitchell, 1997). The advantages of lazy learning algorithms include their simplicity and their ability to handle noisy data. Upfront training of an ML model is not required. Disadvantages of lazy learning algorithms include their computational complexity upon classification of new data and their sensitivity to the choice of the hyperparameter k. (Mitchell, 1997)

Hyperparamter k The hyperparameter k needs to be chosen carefully, as it affects the performance of the kNN algorithm. A small value of k will make the algorithm more sensitive to noise, while a large value of k will make the algorithm more conservative. The optimal value of k will depend on the specific dataset and the desired trade-off between accuracy and robustness. (Mitchell, 1997) Chossing a higher value for k is visualized in figure 2. For this bachelor thesis, the value of k

= 10 has been provided by Nakata et al.. However, the choice of k = 10 has been evaluated through grid search in experiment 2 (cf. 4.5.1).

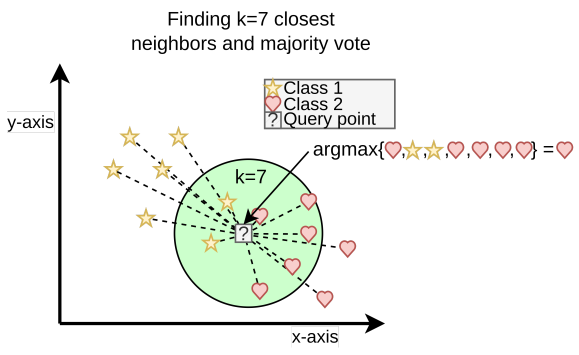

Figure 2: Visualization of kNN classification process with k=7 (cf. Figure 1). Adapted from Navlani (2018)

Distance-Weighted kNN Distance-weighted k-nearest neighbors is a modification of the kNN algorithm that assigns weights to the k-nearest neighbors of a new data point according to their distance. This means that closer neighbors have more influence on the prediction than farther neighbors. For distance-weighted kNN, it is only needed to adjust the inference formula (cf. algorithm 2 line 6) by adding a weight factor w_i. The resulting inference formula for distance-weighted kNN is: $\hat{y_q} = \text{argmax}_{l \in \mathcal{L}} \sum_{n=1}^{k} w_i \mathcal{L}^n . y^{(i)}$ with $w_i = \frac{1}{(d^{(i)})^2}$. (Mitchell, 1997)

By using distance-weighted kNN it is theoretically possible to omit the bound of k and calculate the classification based on all available data points. Using this approach distance-weighted kNN would be a global instead of a local ML method. According to Mitchell there is no advantage in using global distance-weighted kNN; additionally, the classifier will run more slowly if all available data points are used. (Mitchell, 1997) The accuracy of the local distance-weighted kNN approach is going to be compared with the traditional kNN approach in experiment 3 (cf. 4.5.2).

2.2 Deep Learning Introduction

Deep learning is a subfield of machine learning that uses artificial neural networks to learn from data (Goodfellow et al., 2016). The following section will provide a broad overview of deep learning, starting with historical inventions, continuing with the basic principles, and finishing with a description of the two deep learning networks, ResNet and the Vision Transformer architecture, that were used for the prototype in this bachelor's thesis.

From Machine Learning to Deep Learning The history of deep learning dates back to the early days of artificial intelligence research. In the 1950s, psychologist

Frank Rosenblatt developed the perceptron, a simple neural network that could learn to classify patterns (Kanal, 2003). However, a single perceptron could only be used to learn linearly separable problems; thus, solving the XOR problem with a single-layer perceptron is impossible (Marvin and Seymour, 1969). Marvin and Seymour concluded that multi-layer perceptrons could be used to solve non-linearly separable problems, but due to other, more promising approaches, the research on the perceptron and on artificial neural networks stalled (Marvin and Seymour, 1969; Russell and Norvig, 2010).

In the 1980s, there was a resurgence of interest in neural networks. This was due to the development of new learning algorithms, such as backpropagation training. Backpropagation is a method for training neural networks that is much more efficient than the methods that were available in the 1950s (Rumelhart et al., 1986). In the 1990s, the research on artificial neural networks made significant progress. This was due to the availability of large datasets, such as the MNIST dataset of handwritten digits. The MNIST dataset consists of 60000 grayscale images of handwritten digits, each of which is 28 x 28 pixels in size (784 pixels total). Each pixel value is an integer between 0 and 255, where 0 represents black and 255 represents white. A section of these images is visualized in figure 3. The Boosted LeNet-4 ML model proposed by LeCun et al. was able to classify the ten digits in the MNIST dataset with an error rate of 0.7% (LeCun et al., 1998).

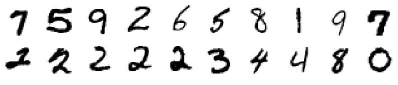

Figure 3: Exemplary visualization of some samples in the MNIST dataset. Section of figure 4 by LeCun et al. (1998).

From the 2000s and onwards, artificial neural networks continued to improve because of the development of new hardware, such as GPUs, which are well-suited for training neural networks (Goodfellow et al., 2016). In 2010, the ImageNet dataset and the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) were released by Russakovsky et al.. In 2012, a group based around Geoffrey Hinton achieved a breakthrough on the ILSVRC with their CNN AlexNet (Krizhevsky et al., 2012). AlexNet is considered one of the first deep artificial neural networks that started the ML subfield of deep learning. In 2016, ResNet-50 achieved a top-5 error rate of 2.99% on the ImageNet dataset, which is a significant improvement over previous models (He et al., 2016). In 2021 the vision transformer architecture was published (Vaswani et al., 2017).

Principles of Deep Learning Deep learning is based on the principle of artificial neural networks (ANNs). Neural networks are made up of interconnected nodes called neurons. The neurons are structured into parallel layers. Multiple layers are arranged sequentially, one after the other, to produce a multilayer perception. (Goodfellow et al., 2016; Roberts et al., 2021)

Each neuron can be depicted as a function that receives input from neurons of the former layer, calculates the weighted sum of all inputs, and compares it against a given activation function. The result of the activation function is passed on to neurons in the following layer. (Goodfellow et al., 2016; Roberts et al., 2021) The neural network is characterized by the used activation functions and the weights on the interconnection of neurons. These weights are adjusted during the training of the neural network. (Goodfellow et al., 2016; Roberts et al., 2021) A commonly used activation function is the ReLU (Rectified Linear Unit) function. It is a non-linear function that outputs the input if it is positive and 0 otherwise. (Goodfellow et al., 2016) According to Goodfellow et al., ReLU is the most popular activation function in modern neural networks, as it is computationally efficient and helps to prevent the vanishing gradient problem Goodfellow et al. (2016).

Deep learning networks can have many layers of neurons. The first layer of neurons, the visible layer, is typically used to extract features from the input data. The subsequent layers of neurons, the hidden layers, are used to learn patterns in the data. From the first to the last hidden layer, the input data will be increasingly abstracted. In the case of image classification, beginning with edge detection, continuing with corners and contours, and finishing with object parts. The final layer of neurons produces the output of the network. (Goodfellow et al., 2016) A simplified view of an image classification deep learning model is provided in appendix A.1 as figure 21.

Applications of Deep Learning Deep learning, compared to Machine Learning, has been used to achieve better results in a wide variety of tasks, including:

- Image classification: Deep learning has been used to develop systems that can classify images into different categories, such as cats, dogs, and cars. For example, the ViT/B-16 model has an accuracy of 85.22% on the ILSVRC. (Dosovitskiy et al., 2021)
- Natural language processing: Deep learning has been used to develop systems that can understand and process natural language. These systems can be used for tasks such as machine translation, text summarization, and question answering. For example, the BERT model has an accuracy of 89.5% on the GLUE benchmark Wang et al. (2019).
- Medical diagnosis and assistance: Deep learning has been used to develop systems that can diagnose diseases from medical images. These systems can help doctors to make more accurate diagnoses. For example, the InceptionV3 model has an accuracy of 99.0% in detecting breast cancer (Husaini et al., 2022).

ResNet(50) ResNet-50 is a deep convolutional neural network (CNN) that was introduced in the paper “Deep Residual Learning for Image Recognition” by He et al. in 2016.

ResNet-50 has 50 layers, hence the name. The first few layers of ResNet-50 extract low-level features from the image, such as edges and corners. The later layers extract higher-level features, such as objects and faces. ResNet-50 uses residual blocks to improve the training of the model. The residual block is the key innovation of ResNet (Res(idual)Net). (He et al., 2016) A typical CNN consists of sequentially aligned layers. In contrast, a residual block adds another path for the data to reach latter parts of the neural network by skipping some layers, the so-called residual connections. This mitigates the vanishing gradient problem because the residual connections accelerate convergence, thus allowing for the training of "deeper" networks. (Veit et al., 2016; Alzubaidi et al., 2021)

The 50 layers of ResNet-50 are arranged in 4 stages. Each stage consists of a number of residual blocks. The first stage has two residual blocks, the second stage has three residual blocks, and so on. The residual blocks are the basic building blocks of ResNet-50. Each residual block consists of three convolutional layers, followed by a conjunction with the residual connections. The residual block's first and last convolutional layers use a 1x1 kernel. The second convolution layer uses a 3x3 kernel. (He et al., 2016) A visualisation for of a residual block is provided in figure 4.

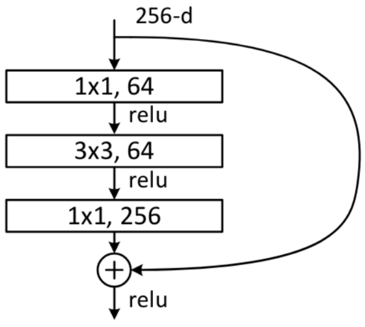

Figure 4: Visualization of a Residual Block in ResNet-50. Section of Figure 5 by He et al. (2016)

When released, ResNet-50 achieved a top-5 error rate of 2.99% on the ImageNet dataset, which is a significant improvement over previous models. The top-5 error rate measures the percentage of test images for which the model's top-five predictions do not include the correct label. (He et al., 2016) During the following years, it has been shown that ResNet is a powerful deep CNN that is very effective for image classification.

Vision Transformer (ViT-B/16) Vision Transformer (ViT) is a neural network architecture that can be used for image processing tasks. At first the image is converted into a sequence of tokens. This is typically done by dividing the image into a grid of patches and then representing each patch as a token. The desired input resolution of the ViT determines the number of patches and the size of each patch. (Dosovitskiy et al., 2021; Han et al., 2020) Afterwards, the patches are then processed by the so-called transformer encoder. The transformer encoder is a neural network that learns to attend to different parts of the image and identify their relationships. The transformer encoder is typically implemented using the self-attention mechanism, which allows it to learn long-range dependencies between the tokens. This allows the ViT to learn a global image representation, which can be used for image classification, object detection, and segmentation tasks. (Vaswani et al., 2017; Dosovitskiy et al., 2021; Han et al., 2020) The ViT architecture is visualized in Figure 5. The transformer encoder includes residual connections, like the residual connections in ResNet.

The ViT architecture was first introduced in the paper “An Image is Worth 16 × 16 Words: Transformers for Image Recognition at Scale” by Dosovitskiy et al. in 2021. (Since 2020 the paper was available as preprint on arxiv.org). The paper showed that ViTs could outperform former state-of-the-art results on the ILSVRC, without using any convolutional layers (Vaswani et al., 2017). There are different ViT model variants available. ViT-B/16 means, that the “Base” variant is used with a 16x16 input patch size. ViT-Base has 12 layers, 12 heads, and 86M parameters (Dosovitskiy et al., 2021). Other available variants are ViT-Large and ViT-Huge. Both increase the number of layers, heads, and parameters.

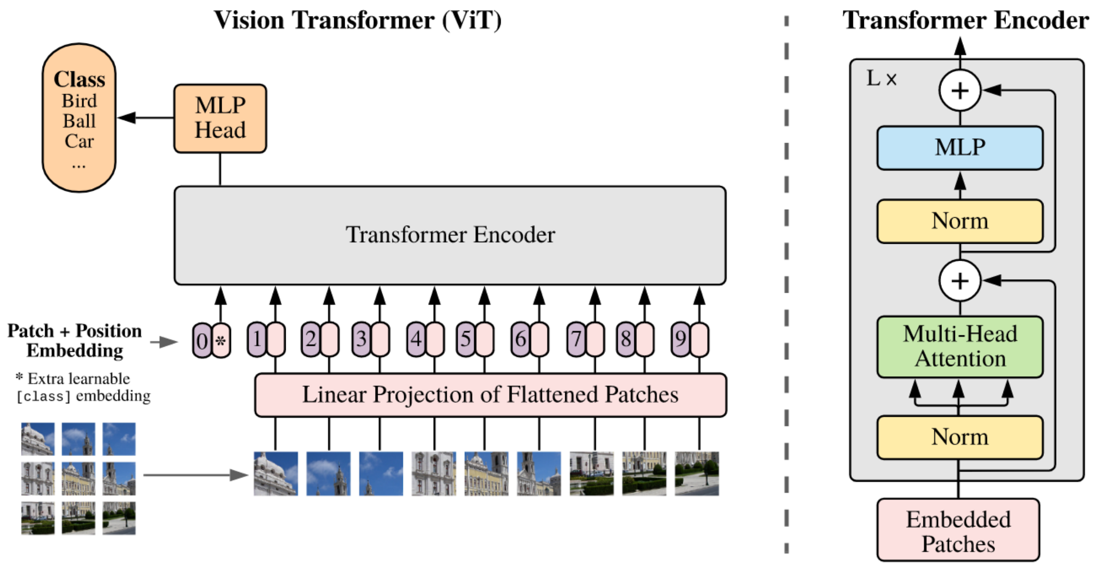

Figure 5: Illustration of the Vision Transformer Architecture. The input image is split into fixed-size patches, which are then linearly embedded and combined with a position embedding. The resulting sequence of vectors is then fed into the transformer encoder. To perform classification, an extra learnable “classification token” is added to the sequence. Illustration by Dosovitskiy et al. (2021).

2.3 Continual Learning

Continual learning is a subfield of ML. Conventional machine learning models are trained on a fixed dataset to capture its (single) data distribution (cf. 2.1). In contrast, continual learning models are trained on a continuous stream of new data to capture a dynamic data distribution that changes progressively over time. (Lange et al., 2022; Parisi et al., 2019; Wang et al., 2023) The fundamental components of a definition of continual learning are:

- Learning over time: Continual learning algorithms are capable of learning from progressively available new data over time. (Parisi et al., 2019)
- Task changes: Continual learning algorithms are able to adapt to changes in the task, such as when new classes are added or the data distribution changes. (Parisi et al., 2019)
- Minimize knowledge interference: Continual learning algorithms aim to minimize the interference between already learned knowledge and newly acquired knowledge. Thus, the learning of new tasks shall not reduce the performance of older tasks, also known as catastrophic forgetting. (Parisi et al., 2019)

Catastrophic Forgetting Catastrophic forgetting is a phenomenon that occurs in continual learning when a model forgets previously learned knowledge as it learns new knowledge (Wang et al., 2023). This can lead to an abrupt performance decrease or a complete loss of previously learned knowledge (Parisi et al., 2019).

One way to understand catastrophic forgetting is through the lens of neural networks. As described in 2.2, neural networks are made up of a large number of interconnected neurons. These interconnections represent the knowledge the neural network has learned by its adjacent weights. When a neural network learns a new task, it adjusts the weights of its connections in order to represent the new data better. However, as the learning process adjusts the weights, it can also cause the network to forget by significantly interfering with the weights of a former learned task, resulting in lower performance. Hence, the name catastrophic forgetting. (Parisi et al., 2019)

To reduce the impact of catastrophic forgetting, continual learning systems must determine a balance between plasticity (the ability to integrate new information) and stability (the ability to retain existing knowledge), known as the stability-plasticity dilemma or stability-plasticity trade-off (Lange et al., 2022; Wang et al., 2023).

Continual Learning Approaches Multiple continuous learning approaches have emerged to reduce the impact of catastrophic forgetting. Lange et al. proposed a taxonomy based on three main categories (Lange et al., 2022).

- Regularization-based methods: A regularization term is introduced in the loss function to consolidate previous knowledge when learning new data. Thus, the model is penalized for forgetting what it has already learned. (Lange et al., 2022) Common approaches are EWC and LwF.

- Replay methods: Store previous task samples in raw format or generate pseudo-samples with a generative model. These samples are replayed while learning a new task by reusing them as model inputs. (Lange et al., 2022) Common approaches are iCaRL and ER.
- Parameter isolation methods: Store dedicated distinct model parameters for each task, for example, by freezing previous task parameters and growing new branches for new tasks. (Lange et al., 2022) Common approaches are PackNet and ExpertGate.

The approach to continual learning chosen by Nakata et al. has been inspired by the replay methods. However, as their approach is based on a pretrained CNN and inference through kNN (cf. kNN chapter 2.1), they introduce their approach as a data-based approach, which uses available datasets as knowledge sources in order to adapt to new tasks (Nakata et al., 2022).

2.4 Explainable AI Overview

Explainable AI (XAI) is a study field that aims to improve trust and transparency of AI-based systems, thus making the results more understandable to humans (Adadi and Berrada, 2018; Arrieta et al., 2020). Especially current ML and DL algorithms suffer from a lack of transparency, meaning it is difficult to explain a decision made to the user of such a system (Adadi and Berrada, 2018). However, through privacy measures such as the GDPR (European Parliament and Council of the European Union, 2016) the user has a granted "right to explanation" (Adadi and Berrada, 2018; DG et al., 2020).

Besides the juristical need for XAI in Europe, Samek et al. provided additional arguments on why XAI is desirable (Samek et al., 2017):

- Verification: Domain experts, for example doctors, can examine the models' calculated decisions to uncover false correlations detected by the model. (Samek et al., 2017)
- Improvement: XAI allows for identifying weaknesses and biases and comparing different models. By explaining and, thus, understanding how AI systems work, it is assumed that making them more accurate and reliable should be possible. (Samek et al., 2017)
- Learning: Humans could use XAI systems to acquire new knowledge. For example, AlphaGo identified a new strategy to play Go, which professional human players have been unable to uncover. (Samek et al., 2017)

Arrieta et al. provided a distinction of different levels of transparency in ML models. (Arrieta et al., 2020)

- Simulatability is the ability of a model to be simulated by a human. For example, a single perceptron neural network (cf. Kanal (2003)) (Arrieta et al., 2020).
- Decomposability means that the input, parameters, and calculation of a model must be able to be explained in order to explain the behavior of the whole model. Thus, every input needs to be interpretable by humans. (Arrieta et al., 2020)
- Algorithmic Transparency describes the ability of a human to follow the model's process to produce the model's output provided with the given input. Algorithmic transparency can only be achieved if the model can be mathematically analyzed. (Arrieta et al., 2020)

These three different levels of ML model transparency are visualized and further explained by an example in figure 6.

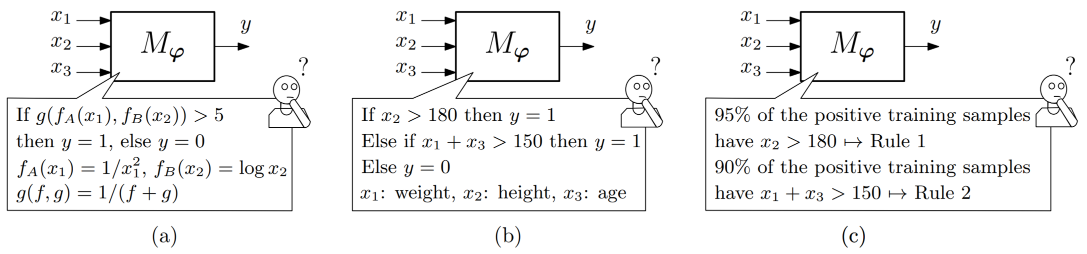
x_1
x_2
x_3
M_φ
y
?
If $g(f_A(x_1), f_B(x_2)) > 5$
then $y = 1$, else $y = 0$
$f_A(x_1) = 1/x_1^2$, $f_B(x_2) = \log x_2$
$g(f, g) = 1/(f + g)$
(a)
If $x_2 > 180$ then $y = 1$
Else if $x_1 + x_3 > 150$ then $y = 1$
Else $y = 0$
x_1: weight, x_2: height, x_3: age
(b)
95% of the positive training samples have $x_2 > 180 \mapsto$ Rule 1
90% of the positive training samples have $x_1 + x_3 > 150 \mapsto$ Rule 2
(c)

Figure 6: Illustration of different Levels of Transparency in Machine Learning Models. Figure by Arrieta et al. (2020). Original caption: "Figure 3: Conceptual diagram exemplifying the different levels of transparency characterizing a ML model M_φ, with φ denoting the parameter set of the model at hand: (a) simulatability; (b) decomposability; (c) algorithmic transparency. Without loss of generality, the example focuses on the ML model as the explanation target. However, other targets for explainability may include a given example, the output classes or the dataset itself." (Arrieta et al., 2020)

The transparency level of kNN (cf. kNN chapter 2.1) depends on the features, the used hyperparameter k, and the distance function. A very high k impedes simulatability by a human. Complex features, or complex distance functions, hinder the ability of decomposability. Thus, in the most complex case, kNN's level of transparency is bound to algorithmic transparency. (Arrieta et al., 2020) In other words, kNN models are more interpretable when they have fewer neighbors, simpler features, and simpler distance functions, thus fulfilling all three levels of transparency (Arrieta et al., 2020).

Besides the different levels of transparency in ML models, Arrieta et al. also provided a summary of six Post-hoc explainability techniques for ML models. (Arrieta et al., 2020) 1. Text explanation, 2. Visual explanation, 3. Local explanation, 4. Explanation by example, 5. Explanation by simplification, 6. Feature relevance explanation

These techniques target models that are not interpretable by design and cover common human ways to explain processes and systems. For this thesis, only explanations by example and visual explanations are relevant; information on the other techniques can be found in Arrieta et al. (2020).

Explanations by Example Post-hoc explanations by example are a way to understand machine learning model outputs by extracting data examples related to this output. This is similar to humans' explanation behavior by using examples. Explanations by example can help to understand the inner relationships and correlations that the model has learned. (Arrieta et al., 2020) This explanation technique is examined in experiment 5 (cf. 4.6.1) by comparing an image classified by kNN with its nearest neighbors.

Visual Explanation Techniques Post-hoc visual explanation techniques aim to visualize the model's behavior in a way that is easy for humans to understand because visualizations effectively communicate complex information (Arrieta et al., 2020). A common Post-hoc visual explanation technique is dimensionality reduction, which will be discussed in 2.4.1 and examined in experiment 6 (cf. 4.6.2).

2.4.1 Dimensionality Reduction

Dimensionality reduction is the process of reducing the number of features in a dataset while minimizing the loss of information (Reddy et al., 2020). This is useful for high-dimensional datasets, which are datasets with many features. These high-dimensional datasets often contain redundant information, increasing the effort in data analysis and training of ML models. (Jia et al., 2022) Dimensionality reduction techniques extract new features from the existing features and combine them into a lower-dimensional feature space. This new feature space contains the most important information from the original features while eliminating redundant or irrelevant information. (Jia et al., 2022)

Dimensionality Reduction techniques reduce the computation time required to train ML models (Reddy et al., 2020) and can increase the explainability of ML models (Arrieta et al., 2020).

Multiple dimensional reduction methods are available. They can be categorized into supervised and unsupervised methods and linear and non-linear techniques. Supervised dimensionality reduction methods consider provided class labels (Jia et al., 2022; Nanga et al., 2021). Linear methods assume that the data lies in a low-dimensional subspace of the high-dimensional space, while non-linear methods make no such assumptions (Nanga et al., 2021). In the following, four dimensionality reduction methods will be presented. These four methods have been evaluated during this thesis in experiment 4 (cf. 4.5.3) and experiment 6 (cf. 4.6.2).

General Approach:
The general approach of dimensionality reduction can be mathematically described as follows: Suppose there is a n-dimensional vector $X = [x_1, x_2, ..., x_n]^T$ and a mapping function $Y = f(X)$. The mapping function f maps the vector X to an m-dimensional vector $Y = [y_1, y_2, ..., y_m]^T$ with $m << n$ and Y containing the main features of X. (Jia et al., 2022)

PCA Principal component analysis (PCA) is a linear unsupervised dimensionality reduction technique that identifies directions of greatest variance in the data, the so-called principal components of a dataset (Martínez and Kak, 2001). The following steps, enhanced with their mathematical formulas, provide an overview of the general PCA process. The formulas are provided by Reddy et al. (2020).

Step 1: Standardize the data by subtracting the mean $\bar{x}_j$ from each feature x^i_j and dividing by the standard deviation σ_j for each feature.

$$x^i_j = \frac{x^i_j - \bar{x}_j}{\sigma_j} \quad \forall j$$

Step 2: Calculate the covariance matrix

$$\sum = \frac{1}{m} \sum_i^m (x_i)(x_i)^T, \sum \in R^{n*n}$$

Step 3: Calculate the eigenvector and eigenvalue of the covariance matrix

$$u^T \sum = \lambda \mu$$

$$U = \begin{bmatrix} | & | & | \\ u_1 & u_2 \dots & u_n \\ | & | & | \end{bmatrix}, \quad u_i \in R^n$$

Step 4: Project the top k eigenvectors of the covariance matrix; this means the vectors with the largest eigenvalues are selected first. Thus, the first few principal components capture the most variance in the data.

$$x_i{}^{\text{new}} = \begin{bmatrix} u_1^T x^i \\ u_2^T x^i \\ :::: \\ :::: \\ u_k^T x^i \end{bmatrix} \in R^k$$

PCA can fail to find the most compact description of the data if the principal components are highly statistically dependent. This is because PCA does not consider the statistical dependencies between the principal components. (Nanga et al., 2021) PCA is also sensitive to outliers, which are common in realistic training sets. This is because PCA uses least squares estimation techniques, which means that outliers can significantly impact the results of PCA. (Nanga et al., 2021)

LDA The dimensionality reduction approach of Linear discriminant analysis (LDA) is similar to PCA. PCA maximizes the variance of the data; additionally, LDA also maximizes the separation between different classes. (Martínez and Kak, 2001; Nanga et al., 2021) Hence, LDA is a supervised linear dimensionality reduction technique.

LDA creates a within-class scatter matrix

$$S_w = \sum_{j=1}^{c} \sum_{i=1}^{N_j} \left(\mathbf{x}_i^j - \mu_j\right) \left(\mathbf{x}_i^j - \mu_j\right)^T$$

and a between-class scatter matrix

$$S_b = \sum_{j=1}^{c} \left(\mu_j - \mu\right) \left(\mu_j - \mu\right)^T$$

(Martínez and Kak, 2001). The goal is to find a subspace of the data where the classes are as far apart as possible, and the data within each class is as close together as possible. Hence, maximizing the between-class scatter matrix S_b while minimizing the within-class scatter matrix S_w. (Martínez and Kak, 2001; Nanga et al., 2021)

LDA is sensitive to the sample size, especially to the so-called small sample problem. Suppose the number of samples in the dataset is much smaller than the dimensionality of the data. In that case, LDA cannot find a meaningful lower dimensional space, resulting in the within-class scatter matrix becoming singular. (Martínez and Kak, 2001; Nanga et al., 2021)

t-SNE t-distributed stochastic neighbor embedding (t-SNE) has been introduced in 2008 (Van der Maaten and Hinton, 2008). It is a dimensionality reduction technique that has specifically been crafted to visualize high-dimensional data in 2D or 3D. The t-SNE algorithm can be reduced to a 6-step procedure. The complete and concise mathematical description of each step is available within the paper "Visualizing Data using t-SNE" by Van der Maaten and Hinton (2008).

- Step 1: Calculate the pairwise Euclidean distance between the data points.
- Step 2: Construct a probability distribution P over pairs of data points in the high-dimensional space. More similar data points are assigned a higher probability than dissimilar data points.
- Step 3: Initialize a low-dimensional embedding of the data.

- Step 4: Construct a Student t-distribution Q with a single degree of freedom over pairs of data points in the low-dimensional space.
- Step 5: Minimize the Kullback-Leibler divergence D_{KL} between the two probability distributions by gradient descent.
- Step 6: Repeat steps 4 and 5 until the algorithm converges or the maximum number of iterations has been reached.

The gradient, which shall be minimized is given as:
$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_j (p_{ij} - q_{ij}) (\mathbf{y}_i - \mathbf{y}_j) (1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}$(Van der Maaten and Hinton, 2008)
In 2014 van der Maaten accelerated t-SNE by developing an approximation of the used gradient by using the Barnes-Hut algorithm. This reduces the complexity class from $\mathcal{O}(N^2)$ to $\mathcal{O}(NlogN)$. (van der Maaten, 2014)

U-MAP Uniform Manifold Approximation and Projection (UMAP) is a non-linear dimensionality reduction technique that preserves the global structure (inter-class distances) and local structure (intra-class distances) of a dataset, while compared to PCA and t-SNE, achieving a superior runtime performance (Dalmia and Sia, 2021).

UMAP is a two-phased algorithm. In the first phase, the graph construction phase, a high dimensional graph representation of the data, a connectivity graph, is constructed (McInnes and Healy, 2018; Dalmia and Sia, 2021). The connectivity graph is a network of nodes, where each node represents a data point, and each edge between the nodes represents the relationships between the data points. By default, this connectivity graph is constructed using the efficient NN-Descent (Nearest Neighbor Descent) algorithm. (McInnes and Healy, 2018; Dalmia and Sia, 2021)

Once the connectivity graph has been constructed, UMAP constructs a low dimensional embedding of the data by minimizing the difference between the connectivity graph of the original data and the connectivity graph of the low dimensional embedding (McInnes and Healy, 2018). This is done by iteratively adjusting the positions of the data points in the low dimensional embedding (McInnes and Healy, 2018). This shows similarities to force-directed graph layout algorithms. These algorithms embed the proximity of data points in the high-dimensional space in a low-dimensional space by iteratively moving the low-dimensional points closer together (respectively further apart) according to their proximity in the high-dimensional space. (Wang et al., 2021) Compared to t-SNE, UMAP is faster and produces better visualizations, especially because UMAP preserves the global structure better (Nanga et al., 2021).

2.5 Machine Unlearning and Privacy Regulations

Removing data from machine learning models is challenging, especially for complex models such as neural networks (Ginart et al., 2019). The naive approach is to retrain the model without the data that is to be removed. However, this can be computationally expensive and time-consuming, especially for large datasets. Additionally, if the model is deployed in a production setting, taking it offline for retraining may not be feasible.

The General Data Protection Regulation (GDPR) gives individuals the erasure right (also known as the 'right to be forgotten') to withdraw their consent to having their data processed at any time (European Parliament and Council of the European Union, 2016; Veale et al., 2018; DG et al., 2020). This means that organizations that use machine learning models that contain personal data must be able to efficiently remove data from those models upon request (Ginart et al., 2019; DG et al., 2020).

Personal data used to train ML models is at risk through privacy attacks such as model inversion and membership inference attacks (Ginart et al., 2019; Veale et al., 2018).

Model inversion attacks are a type of adversarial attack in which an attacker attempts to extract sensitive information from an ML model without having access to the training data. (Ginart et al., 2019; Veale et al., 2018) Suppose the attacker knows the predictions of a model M(B) for a set of individuals and has access to a training dataset A of individuals. In that case, the attacker can use model inversion to recover some of the variables in training dataset B for the individuals who are in both A and B. (Veale et al., 2018)

Membership inference attacks are privacy attacks that attempt to determine whether a given individual's data was used to train an ML model. These attacks do not recover the individual's data itself but rather determine whether the individual's data was part of the training set. (Veale et al., 2018)

Figure 7 visualizes model inversion and membership inference privacy attacks.

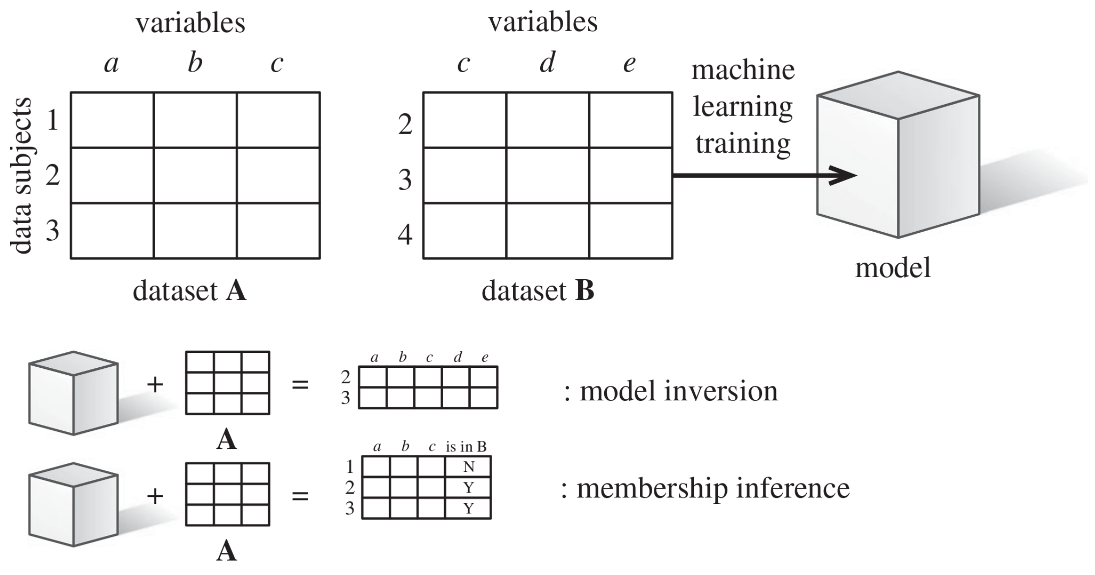

Figure 7: Illustration of Model Inversion and Membership Inference Privacy Attacks. Figure by Veale et al. (2018).

Ginart et al. provide design principles for the efficient deletion of user data in ML systems (Ginart et al., 2019). These principles include the use of linear models because they are relatively simple to train, and it is possible to remove a data point by undoing its influence on a set of parameters. (Ginart et al., 2019)

Software engineering has inspired another suggested approach: the use of modularity. Modular machine learning models are those in which the model parameters are divided into different modules. This makes it possible to efficiently delete data points by only recomputing the modules that are affected by the deleted data points. (Ginart et al., 2019)

Lastly, Ginart et al. suggest using lazy learning methods (cf. kNN chapter 2.1). Lazy learning methods delay computation until inference time, which makes them very efficient for deletion, as all that is needed to delete a data point is to update the model at inference time. (Ginart et al., 2019) This is also the approach chosen by Nakata et al. (2022). This lazy learning approach to machine unlearning is evaluated in this bachelor thesis as experiment 10 (cf. 4.8.1) and experiment 11 (cf. 4.8.2).

3 Methods

3.1 System Description

Nakata et al. propose a three-phase system consisting of 1. pretraining, 2. knowledge storing, and 3. inference. The following section will provide an overview of each phase. The system is visualized in figure 8.

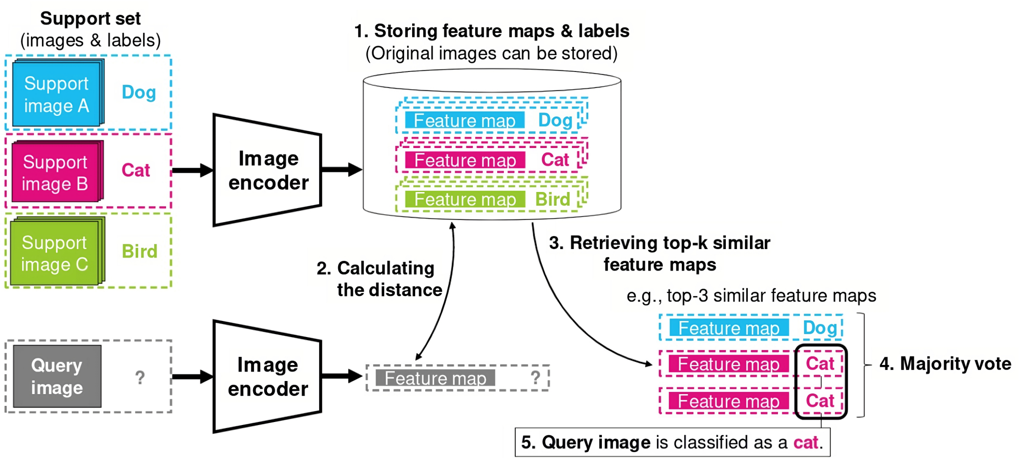

Figure 8: Visualisation of the system proposed by Nakata et al. Image from Nakata et al. (2022). Original caption: "Fig. 1. Overview of our image classification system. Our system stores feature maps extracted from support images with the corresponding labels to the external storage. When classifying a query image, our system retrieves feature maps similar to the query one from the storage by calculating the distance based on cosine similarity. The query image is classified by majority vote on the labels of the top-k similar feature maps."

Pretraining The first phase is pretraining, which involves training an image encoder model on a large-scale dataset of images. This dataset can be unlabeled or noisily labeled, as the goal of pretraining is to learn general-purpose image features. It is important to note that the transformation performed by the pretrained image encoder model should map semantically similar images to neighborhoods in latent space. This is essential for the similarity-based retrieval performed in the inference phase. Nakata et al. conducted a preliminary experiment to select a pretraining method by comparing the performance of three pretraining methods on four image datasets (CIFAR-10, CIFAR-100, STL-10, and ImageNet-1k cf. 4.1 Datasets). The results showed that: A supervised learned ResNet-50 model on ImageNet-1k achieved the best accuracy on ImageNet-1k, but not on the other datasets. This suggests that the image encoder model does not generalize well to unseen datasets. (Nakata et al., 2022) A ViT-B/16 model pretrained by Masked Auto Encoder (MAE) on the ImageNet-1k dataset without labels showed poor performance. This is because the objective of MAE pretraining is incompatible with the system's similarity-based retrieval approach. (Nakata et al., 2022) A ViT-B/16 model pretrained by CLIP (Radford et al., 2021) on 400 million image and text pairs collected from the internet achieved good accuracy on all four datasets. This indicates that the image encoder model is well generalized. (Nakata et al., 2022) Based on these results, Nakata et al. decided to use image encoder models pretrained by CLIP (Radford et al., 2021) in their experiments.

Knowledge Storing In the knowledge storing phase, the pretrained image encoder model extracts feature embeddings (vector representations) from a so-called support set of images. The support set is a small set of images relevant to the downstream task for which the model will be used. The pretrained image encoder model extracts feature embeddings from a support set of n-labeled images: $\{x_{s,1}, ..., x_{s,n}\}$, with corresponding labels $\{y_1, ..., y_n\}$. The feature embeddings are extracted using the following equation:

$$z_{s,i} = f(x_{s,i})$$

where $f()$ is the pretrained image encoder model and $z_{s,i}$ is the d-dimensional feature embedding of the i-th support image. Each extracted feature embedding is paired with its corresponding label $(z_{s,i}, y_i)$ and stored in a database.

Inference (kNN) In the inference phase, the pretrained image encoder model extracts a feature embedding z_q of a query image x_q as $z_q = f(x_q)$. The system then retrieves the top-k (cf. Hyperparamter k 2.1) most similar feature embeddings from the database using cosine similarity (cf. Cosine Similarity 2.1). The query image is then classified by majority vote on the labels of the top-k similar feature embeddings.

3.2 Prototype Implementation

Besides the system description, as mentioned earlier, and the usage of the PyTorch library (Paszke et al., 2019) Nakata et al. did not provide details or source code regarding their implementation. To verify their results, a prototype based on their system description has been built during this thesis.

The prototype system is implemented using the Python programming language and the PyTorch library, specifically the torchvision package (Paszke et al., 2019). The system uses pretrained image classification models from PyTorch. However, to extract the feature embeddings and hence work as a pretrained image encoder (cf. Pretraining 3.1), in the case of ResNet-50, the last fully connected layer is removed; in the case of ViT-B/16, the MLP head at the end is removed. The extracted feature embeddings have a size of 2048 dimensions for the ResNet-50 image encoder and 768 dimensions for the ViT-B/16 image encoder.

The system uses Chroma (Huber and Troynikov, 2023)(formerly known as ChromaDB) as a database to store the extracted feature embeddings and their corresponding labels. Chroma is an in-memory embedding database (also known as vector database) specially designed to store feature embeddings and their metadata. It is released under Apache License 2.0. Chroma allows to search for similar data points by finding the nearest neighbors in the embedding space. By default, Chroma provides different embedding functions depending on the data type; using a custom embedding function is also possible, as well as storing already obtained feature embeddings. (Huber and Troynikov, 2023) Storing already obtained feature embeddings from the pretrained image encoder is chosen as the approach for the prototype as it allows to separate the computationally intensive feature embedding calculation from the relatively simple data storage and retrieval.

For inference, kNN (cf. k-Nearest Neighbors section 2.1) has been implemented based on algorithm 2. If not stated otherwise k=10 and majority vote was used. Chroma allows to query for the k-Nearest Neighbours, thus cosine similarity distance calculation (cf. cosine similarity section 2.1) is already handled inside Chroma.

Hardware The separation of computation and data storage mentioned above allowed the use of two different servers for the project. The computation server houses an NVIDIA A100 80GB Tensor Core GPU[1], which was shared through virtualization. The data storage server is based on TrueNas[2]. The two servers are connected through a LAN network.

3.3 Differences to Nakata et al.'s System

In the absence of source code for Nakata et al.'s system, the prototype was rebuilt from their description. Despite best efforts, there may be some differences between the two systems due to factors such as interpretation of the Nakata et al. paper and specific implementation choices made. The prototype system differs from the Nakata et al. system in a number of ways:

- Chroma: Nakata et al. do not specify which database is used to store the extracted feature embeddings and their corresponding labels. The prototype system uses Chroma version 0.3.25.
- Transfer over network: The system by Nakata et al. requires the support set to be stored on the same machine as the image encoder model. The prototype system can transfer the support set and its feature embeddings over a network to the pretrained image encoder model. This allows for a separation of feature embedding computation and storage.

[1]`www.nvidia.com/a100`

[2]`www.truenas.com`

- Pretrained networks without CLIP: The Nakata et al. system uses pretrained image encoder models that are pretrained using CLIP (Radford et al., 2021). For simplification, the prototype system uses supervised pretrained image encoder models trained on ImageNet-1k. The pretrained image encoder models, ResNet-50 and ViT-B/16, are based on pretrained image classification models provided by PyTorch (Paszke et al., 2019).

3.4 Dimensionality Reduction in the Prototype Implementation

The following Python libraries were used for dimensionality reduction in this study:

- LDA and PCA: Scikit-learn (`scikit-learn version 1.0.2`) (Pedregosa et al., 2011)
- t-SNE: openTSNE (`opentsne version 0.5.3`) (Poličar et al., 2019)
- UMAP: UMAP (`umap-learn version 1.0.0`) (McInnes et al., 2018)

The different dimensionality reduction methods were instantiated with the default parameters provided by the libraries. Dimensionality Reduction with PCA, t-SNE, and UMAP was performed unsupervised, while LDA was performed supervised.

3.5 Accuracy as a Metric for Comparing Results

The results of the conducted experiments are compared by their accuracy. Accuracy is a metric used to evaluate the performance of machine learning models on classification tasks. It measures the proportion of test images for which the model's predicted label matches the correct label. (Sammut and Webb, 2017) For example, if a model is tested on 100 images and correctly predicts the label of 80, then its accuracy is 80%.

$$\text{Accuracy} = \frac{\text{Number of correct classifications}}{\text{Total number of classifications}} \tag{1}$$

4 Experiments and Results

The following section introduces the datasets and hyperparameters used in the experiments, followed by a presentation of the experiments and their results. A detailed discussion of each result is provided in section 5.

4.1 Datasets

CIFAR-10 and CIFAR-100 In 2009, the CIFAR-10 and CIFAR-100 datasets were introduced by Krizhevsky et al. (2009). CIFAR-10 consists of 10 different image classes with 60000 total images, while CIFAR-100 consists of 100 different image classes with the same number of images. The CIFAR dataset is one of the main datasets for evaluating new proposed classifiers. 50000 images of the dataset are used for training, and the remaining 10000 images are used for testing. All labels in the dataset are simple objects, such as ship, airplane, and cat. Each image contains only one object with a resolution of 32x32 pixels. The CIFAR datasets are a subset of the 80 Million Tiny Images dataset, created by web scraping in 2008 (Torralba et al., 2008). In 2020, a prerelease study uncovered that the "80 Million Tiny Images" dataset contains degrading terms as categories and offensive images (Birhane and Prabhu, 2021). As a response to the findings, the authors of the dataset retracted it. The CIFAR datasets are unaffected because Krizhevsky et al. handpicked their classes.

ImageNet1K ImageNet1K is a large-scale image dataset with over 1 million images belonging to 1000 different classes. It is a widely used dataset for evaluating machine learning models for image classification, for example as part of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). ImageNet1K was created by collecting images from the internet and manually annotating them with corresponding labels. (Russakovsky et al., 2015) The dataset is divided into training, validation, and test sets. The training subset contains 1.2 million images, the validation subset contains 50000 images, and the test subset contains 100000 images. It contains various images, including images of natural objects, manufactured objects, and scenes. In this project, the dataset was indirectly used as a training dataset for the pretrained image classification models (cf. section System Description paragraph Pretraining 3.1) provided by PyTorch (Paszke et al., 2019).

STL-10 In 2011, the STL-10 dataset was introduced by Coates et al. (2011). It is a dataset of 10 different image classes with 60000 total images, similar to CIFAR-10. The STL-10 dataset is divided into 100000 unlabelled and 13000 labeled images. The labeled images contain a training set of 5000 images (500 images from each class) and a test set of 8000 labeled images. For this thesis, the 13000 labeled images are used. All labels in the STL-10 dataset are simple objects, such as ship, airplane, and cat (inspired by CIFAR-10). Each image contains only one object with a resolution of 96x96 pixels.

Melanoma The “Melanoma Skin Cancer Dataset of 10000 Images” is a collection of 10605 images of melanoma skin cancer, the most lethal skin cancer (according to Codella et al. (2018)). The dataset was constructed by Muhammad Hasnain Javid and made publicly available on Kaggle (CC 0: Public Domain[3])(Javid, 2022). The images were collected from different ISIC melanoma skin cancer directories (`https://www.isic-archive.com`). The dataset is divided into a training set of 9605 images and a test set of 1000 images. The training set contains 5000 benign and 4605 malignant images. The test set contains 500 benign and 500 malignant images. Each image has a resolution of 300x300 pixels. The dataset classes are visualized in figure 9.

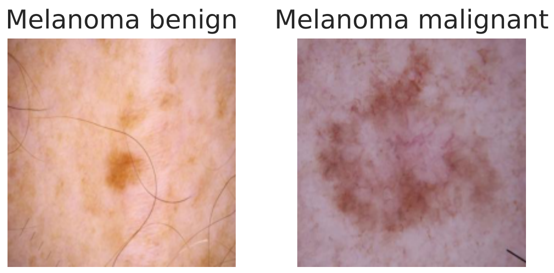

Figure 9: Visualization of the classes in the melanoma dataset

Pneumonia This dataset contains 5863 chest X-ray images of children aged 1 to 5 years old, labeled as either Pneumonia or Normal. The images were selected from retrospective cohorts of pediatric patients at the Guangzhou Women and Children’s Medical Center, Guangzhou, China. According to the authors, all chest X-rays were performed as part of patients’ routine clinical care. (Kermany et al., 2018) The dataset is licensed CC BY 4.0[4] and available to download at `https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia`. The dataset has two classes, “Normal” and “Pneumonia” and is divided into three subsets: training (5216 images, 1341 Normal; 3875 Pneumonia), validation (16 images, 8 Normal; 8 Pneumonia), and test (624 images, 234 Normal; 390 Pneumonia). The images in the dataset have a resolution range of 384x127 to 2916x2583 pixels, with an average resolution of 1328x971 pixels. The training and test subsets were used for this project. The classes of the dataset are visualized in figure 10.

4.2 System Setup Parameters for Experiments

If not stated otherwise, the experiments were conducted with the following parameters.

- $k = 10$ is used as hyperparameter for kNN.

[3]`https://creativecommons.org/publicdomain/zero/1.0/`

[4]`https://creativecommons.org/licenses/by/4.0/`

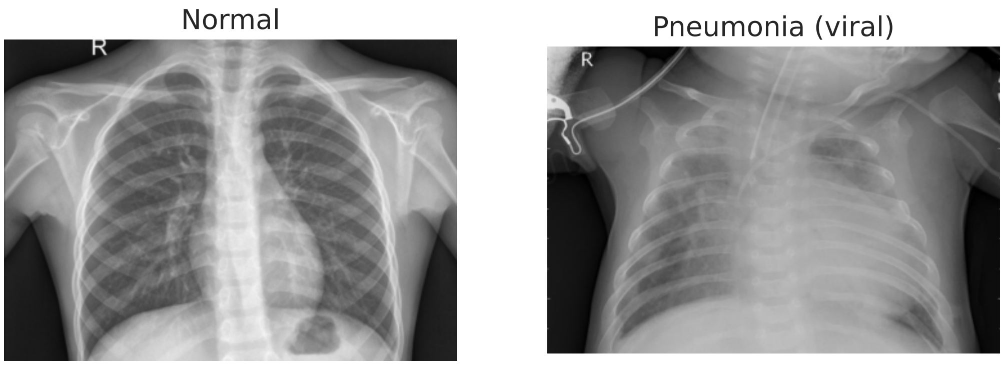

Figure 10: Visualization of the Classes in the Pneumonia Dataset. The X-ray of viral Pneumonia shows a diffuse intra-structural pattern in both lungs (Kermany et al., 2018).

- kNN uses majority vote for inference.
- The images of the datasets are rescaled to 224x224 pixels, because this is the input size of ViT-B/16. Other data preprocessing approaches were not used.

4.3 General Classification Results

The achieved accuracies of the developed prototype system are summarized in table 1.

Image Encoder	Dataset	Accuracy
ResNet-50	CIFAR-10	80.0
	CIFAR-100	54.7
	STL-10	96.2
	Pneumonia	85.3
	Melanoma	90.6
ViT-B/16	CIFAR-10	87.3
	CIFAR-100	61.8
	STL-10	94.9
	Pneumonia	80.3
	Melanoma	91.1

Table 1: Classification Accuracies of the developed Prototype System

Required Storage Space and Computational Time The required storage space for storing the support sets in Chroma was analyzed. Storing the feature embeddings of the 5000 STL-10 training images, encoded with ViT-B/16, occupied around 16 MB of storage. Using ResNet-50 as an image encoder resulted in approximately 41 MB of occupied storage. The creation of the feature embeddings of the

5000 STL-10 training images took around 3 minutes on the provided hardware (cf. Hardware 3.2)

4.4 Comparison of the Prototype to Nakata et al.'s System

4.4.1 Experiment 1: Accuracy Comparison on Identical Datasets

Experiment description The goal of this experiment is to evaluate if the prototype system resembles Nakata et al.'s system. This evaluation compares the accuracy of the two systems on identical datasets (CIFAR-10, CIFAR-100, STL-10). The prototype system will use the training set of each dataset as support set (cf. Knowledge Storing 3.1); the accuracy will be evaluated with each test set.

Results The results are presented in table 2 and visualized in appendix A.2 figure 22 and figure 23.
Overall, the Prototype system performs worse than Nakata et al.'s system on all three datasets. The ResNet-50 image encoder has a smaller gap in performance between the Prototype system and Nakata et al.'s system, with a difference of -1% to -2.8%, compared to the ViT-B/16 image encoder, which has a wider gap of -4% to -12.5%.

Image Encoder	Dataset	Baseline (Nakata et al.)	Prototype	Δ
ResNet-50	CIFAR-10	82.8	80.0	**-2.8**
	CIFAR-100	55.7	54.7	**-1.0**
	STL-10	96.8	96.2	**-2.6**
ViT-B/16	CIFAR-10	94.4	87.3	**-7.1**
	CIFAR-100	74.3	61.8	**-12.5**
	STL-10	98.9	94.9	**-4.0**

Table 2: Experiment 1: Test Accuracies

4.5 Evaluating Parameters of the Prototype System

4.5.1 Experiment 2: Grid Search kNN Hyperparameter k

Experiment description The goal of this experiment is to evaluate whether k=10 is a sensible choice for the hyperparameter k in the kNN classifier. A grid search is performed to evaluate different values of k on the classification accuracy of different data sets (STL-10, Melanoma).

Results The results are presented in table 3 and visualized in figure 11.
The performance of the Prototype system is consistent across different values of the k hyperparameter, with an accuracy of 95% to 96% on the STL-10 dataset and

around 90% on the Melanoma dataset for both the ResNet-50 and ViT-B/16 image encoders. The STL-10 dataset has the maximum accuracy for the ResNet-50 image encoder of 96.3% at k=15 and 95.7% for the ViT-B/16 image encoder at k =25. The Melanoma dataset has the maximum accuracy for the ResNet-50 image encoder of 90.6% at k=3 and k=10, and 91.1% for the ViT-B/16 image encoder at k=10.

Image Encoder	Dataset	k=1	k=3	k=5	k=7	k=10	k=15	k=25
ResNet-50	STL-10	96.1	96.2	96.2	96.1	96.1	**96.3**	96.2
	Melanoma	89.7	**90.6**	90.2	89.9	**90.6**	90.3	89.6
ViT-B/16	STL-10	94.7	95.0	95.1	95.0	95.0	95.4	**95.7**
	Melanoma	90.5	90.6	90.5	90.3	**91.1**	90.9	90.0

Table 3: Experiment 2: Test Accuracies

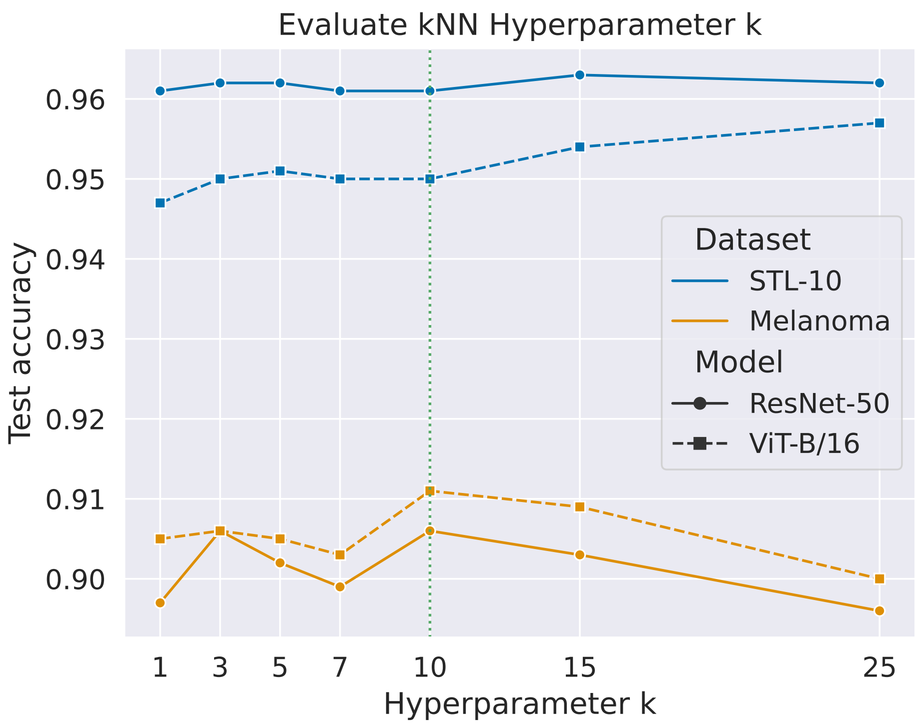

Figure 11: Experiment 2 Visualization of Hyperparameter k

4.5.2 Experiment 3: Comparison of kNN with distance-weighted kNN

Experiment description The goal of this experiment is to evaluate whether distance-weighted kNN increases the classification accuracy of the prototype. The experiment is conducted by comparing the accuracy of kNN and distance-weighted kNN on the STL-10 and the Melanoma dataset. Both kNN variants are used with hyperparameter $k = 10$.

Results The results are presented in table 4.
Overall, the performance of kNN and distance-weighted kNN is very similar. Distance-weighted kNN slightly outperforms kNN on both datasets, using both image encoders. However, the difference in performance is minimal, ranging up to +0.3%.

Image Encoder	Dataset	kNN	distance-weighted kNN	Δ
ResNet-50	STL-10	96.0	96.1	**+0.1**
	Melanoma	90.6	90.9	**+0.3**
ViT-B/16	STL-10	95.1	95.1	**0**
	Melanoma	91.1	91.4	**+0.3**

Table 4: Experiment 3: Test Accuracies

4.5.3 Experiment 4: Comparison of Dimensionality Reduction Methods in Conjunction with kNN

Experiment description The goal of this experiment is to determine the effect of dimensionality reduction methods on the classification accuracy of the kNN classifier. The dimensionality of the feature embeddings of the support set is reduced and, afterwards, stored in the database. The dimensionality reduction model, which was used for the dimensionality reduction of the support set, is stored in memory. The feature embeddings for the test data are reduced using the previously stored dimensionality reduction model. Afterwards kNN is performed on the reduced support and test set to determine the classification accuracy.

The classification accuracies of the test datasets with dimensionality reduction methods are compared to those without dimensionality reduction. A grid search is performed to evaluate a sensible number of remaining dimensions for each of the four dimensionality reduction methods LDA, PCA, t-SNE, and UMAP. The experiment is performed solely on the STL-10 dataset.

Results The results are visualized in figure 12. The detailed results are given in appendix A.3 table 7. UMAP and t-SNE were able to maintain the accuracy of the kNN baseline model across all values for the number of remaining dimensions. In contrast, LDA and PCA lost more than 20% of their classification accuracy for two and three dimensions. However, their accuracy increased as the number of remaining dimensions was increased and was on par with the baseline accuracy for eight remaining dimensions. For ResNet-50, t-SNE slightly improved the baseline classification accuracy by 0.5%, from 96.2% to 96.7%. For ViT-B/16, t-SNE also slightly improved the baseline classification accuracy by approximately 1%, from 94.9% to 95.9% (96.1% for three and eight dimensions). Additionally, LDA with eight remaining dimensions increased the classification accuracy of ViT-B/16 by 2.3%, from 94.9% to 97.2%.
The feature embedding space is visualized in experiment 6 (cf. 4.6.2).

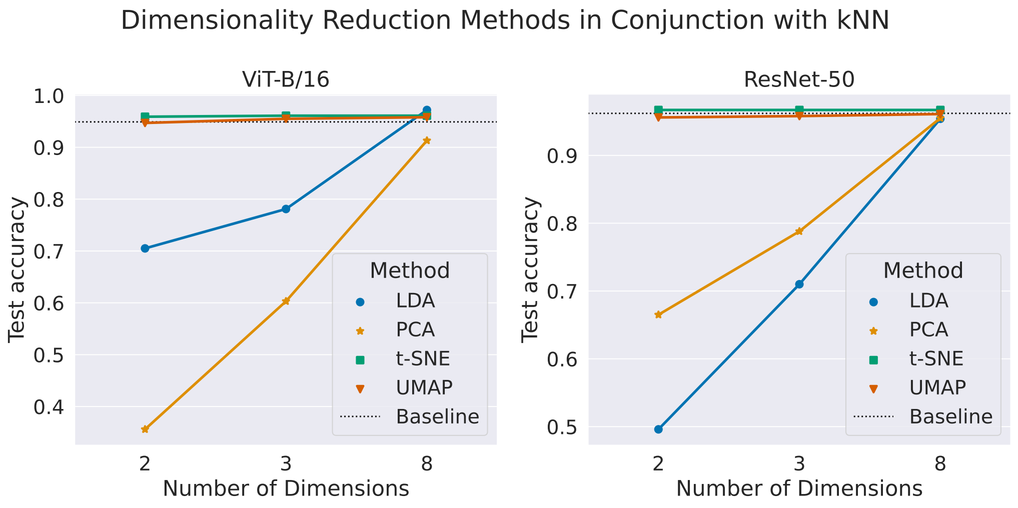

Figure 12: Experiment 4 Results Visualization (Dataset: STL-10)

4.6 Explainable AI

4.6.1 Experiment 5: Visualization of Classification Results through Support Set and Query Image Comparison

Experiment description The goal of this experiment is to visualize the classification results of the prototype by comparing a misclassified query image of the test set to its k nearest-neighbour support images. The experiment is performed solely on the STL-10 dataset.

Results Figure 13 shows the results of experiment 5, which visualized the kNN decision process for a misclassified image. The ten nearest neighbors of the misclassified image are distributed in four classes, with six images depicting cats, two images depicting deer, one image depicting an airplane, and one image depicting a bird. The majority vote of the kNN algorithm results in a cat classification. None of the nearest neighbors is from the actual class of the image (dog). The provided distances allow to determine which neighbor is closest to the misclassified image. In this case, the closest neighbor is the fourth picture in the first row, depicting a cat with a distance of 132.607. The second closest neighbor is the bird image (second image, second row), with a distance of 133.043.

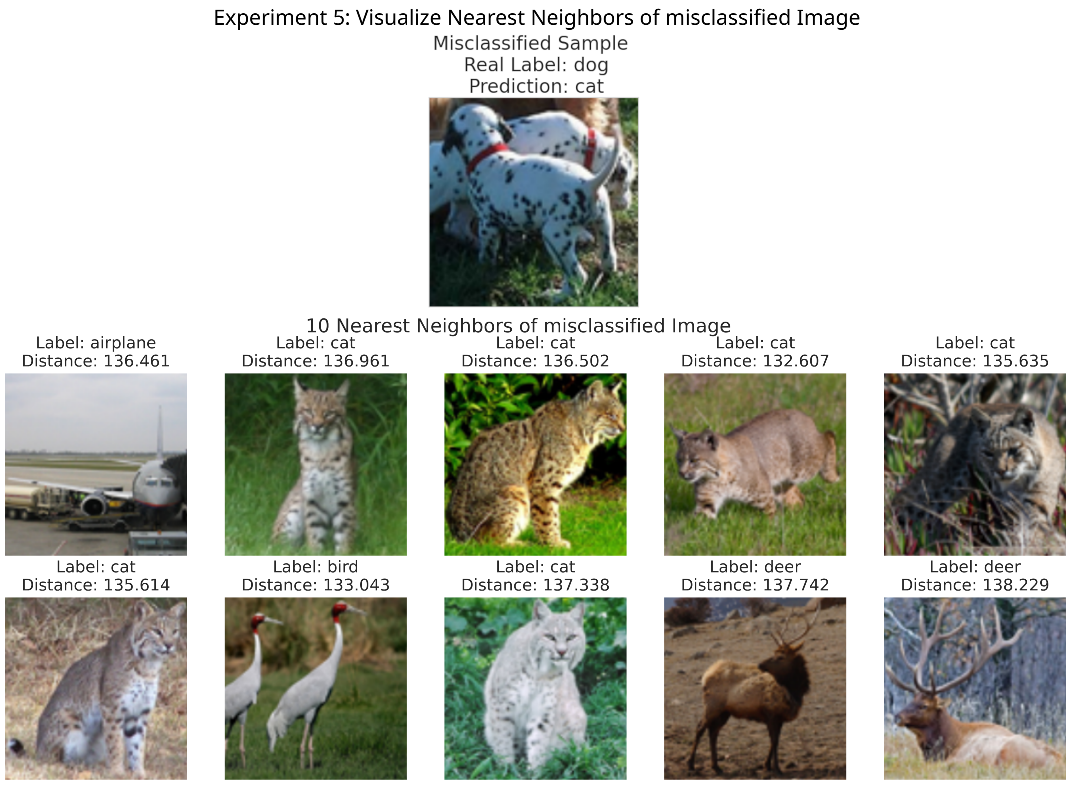

Figure 13: Experiment 5 Visualization of Nearest-Neighbours

4.6.2 Experiment 6: Comparison of Dimensionality Reduction Methods for Feature Embedding Space Visualization

Experiment description The goal of this experiment is to find out if one of the dimensionality reduction methods is superior in explaining the feature embedding space. The 2D visualizations of different dimensionality reduction methods LDA, PCA, t-SNE, and UMAP are compared for visualizing the feature embedding space of the ML model (cf. section 2.4.1). The experiment is performed on the STL-10 dataset.

Results The results using ViT-B/16 as an image encoder are visualized in figure 14. Comparing the visualizations of the feature embedding space of the different dimensionality reduction methods suggests some differences:

- The LDA projection indicates four clusters. Three clusters have overlapping classes within each cluster: 1. Airplane, Ship 2. Car, Truck 3. Bird, Cat, Deer, Dog, Horse.
- The PCA projection suggests one huge cluster of all data points, with no clear separation between the classes.
- The t-SNE projection gives ten distinct clusters.

- The UMAP projection gives seven clusters. Three clusters have overlapping classes: 1. Airplane, Ship 2. Deer, Horse 3. Dog, Cat. Compared to t-SNE, the clusters have a wider separation from each other and are populated more densely.

The additional results using ResNet-50 as an image encoder are visualized in appendix A.4 figure 14. There are only minor differences compared to the ViT-B/16 visualizations.

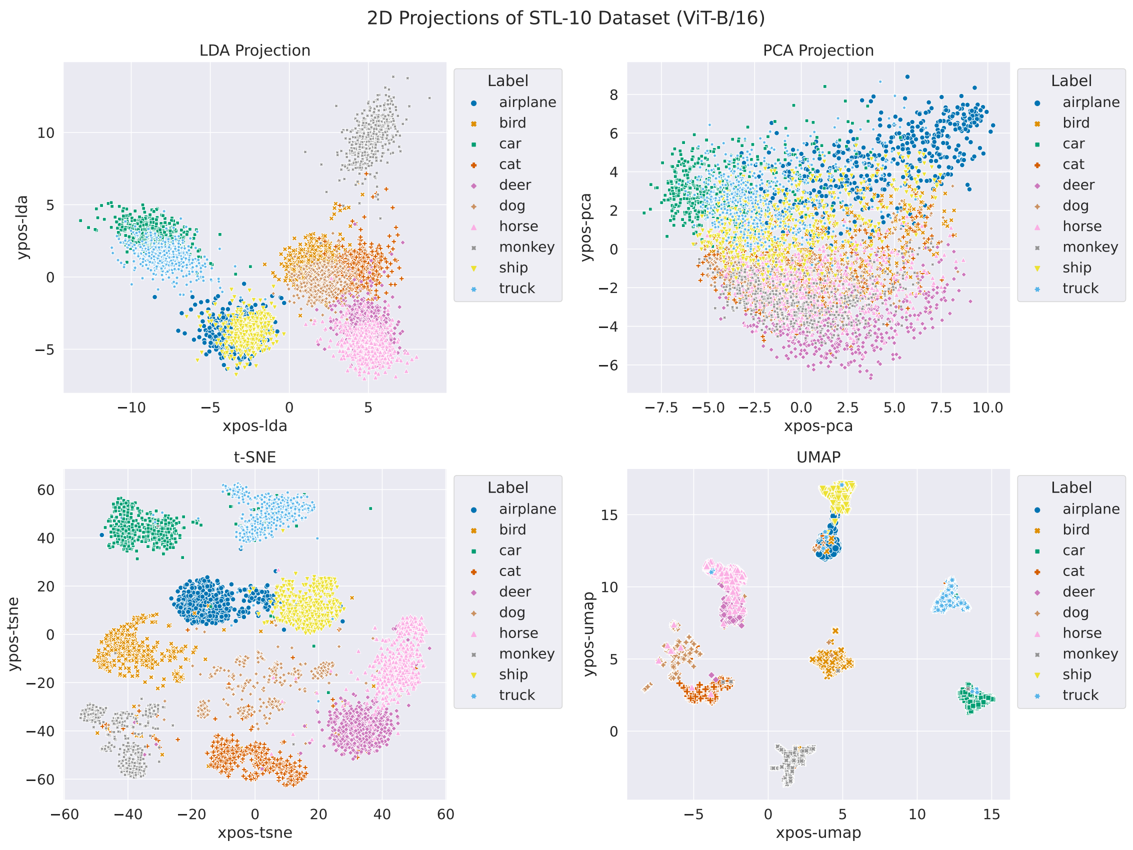

Figure 14: Experiment 6 Feature Embedding Space Visualization for Dimensionality Reduction Methods (Image Encoder: ViT-B/16, Dataset: STL-10)

4.7 Continual Learning and Catastrophic Forgetting

4.7.1 Experiment 7: Class Incremental Continual Learning

Experiment description The goal of this experiment is to investigate the performance differences of class incremental continual learning by stepwise adding classes to the support set and test set. The experiment is performed on STL-10 and CIFAR-10.

Image Encoder	Dataset	Number of Classes									
		1	2	3	4	5	6	7	8	9	10
ResNet-50	STL-10	100	99.6	99.5	99.1	98.0	97.1	95.8	96.0	96.1	96.0
	CIFAR-10	100	97.5	95.1	90.9	86.8	83.9	82.1	81.3	80.5	79.9
ViT-B/16	STL-10	100	99.2	99.0	98.4	98.1	96.7	96.0	95.9	95.6	95.6
	CIFAR-10	100	98.9	97.4	95.7	92.7	90.3	89.0	87.1	87.5	87.4

Table 5: Experiment 7: Test Accuracies

Results The results are presented in table 5.
Both ViT-B/16 and ResNet-50 models achieved high accuracy on the STL-10 dataset, with accuracy above 95% even when the number of classes was 10. However, the ViT-B/16 model outperformed the ResNet-50 model on the CIFAR-10 dataset, especially when the number of classes was large. For example, the accuracy of the ViT-B/16 model was 87.4% when the number of classes was 10, while the accuracy of the ResNet-50 model was only 79.9%. Figure 15 shows that the ResNet-50 model stabilizes beyond n=7 around 96% for the STL-10 dataset. For the CIFAR-10 dataset, after n=7 a continual linear decline of 0.8% per step is observed. Figure 16 shows that the ViT-B/16 model stabilizes beyond n=8 for both datasets.

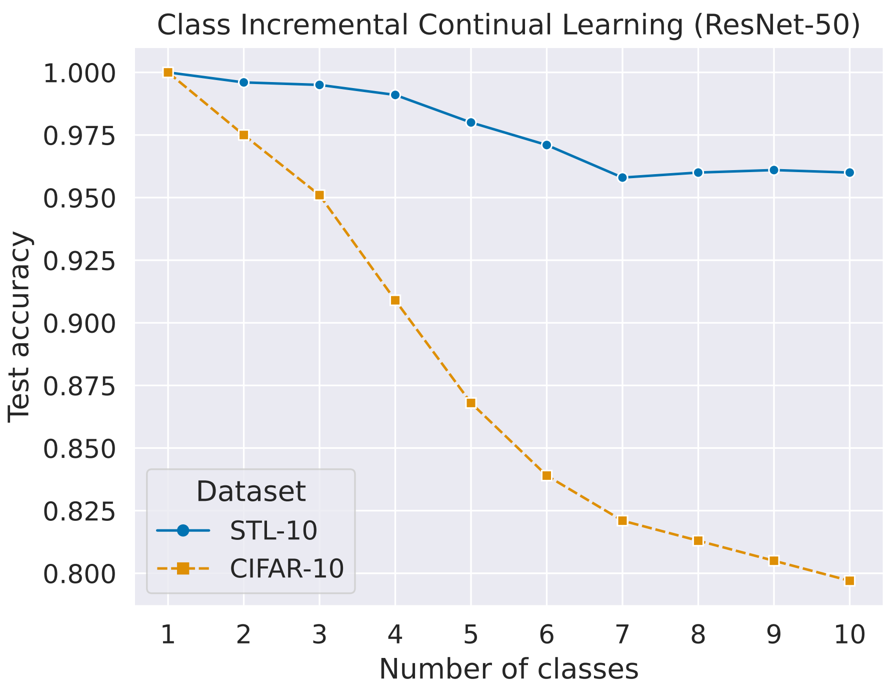

Figure 15: Experiment 7 Results Visualization (ResNet-50)

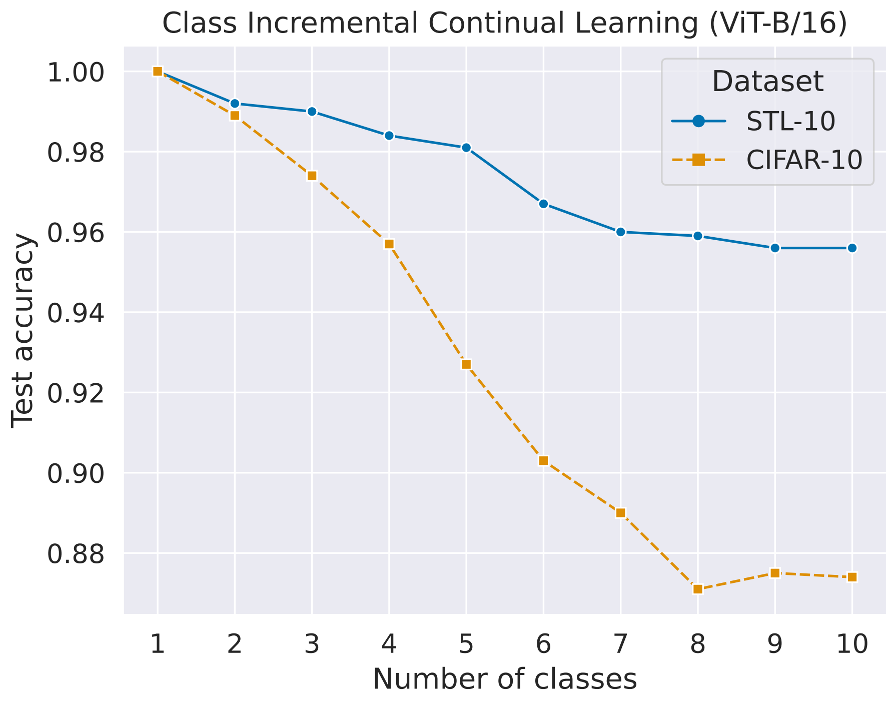

Figure 16: Experiment 7 Results Visualization (ViT-B/16)

4.7.2 Experiment 8: Task Incremental Continual Learning

Experiment description The goal of this experiment is to investigate the performance differences of task incremental continual learning by stepwise adding feature embeddings of each class to the support set and test set. The experiment is performed on STL-10, CIFAR-10, Melanoma and Pneumonia.

Results The results are visualized in figure 17 and figure 18. The whole results table is available in appendix A.5 table 8.
The results show that increasing the number of feature embeddings per class improves performance on all four datasets for both image encoder models. For example, on the Pneumonia dataset, the ViT-B/16 model achieved an accuracy of 74.4% with one feature embedding per class but an accuracy of 80.3% with 512 feature embeddings per class. For each dataset, the ViT-B/16 model achieved at eight feature embeddings per class a performance, which is less than 10% worse than the performance on each whole dataset. The ResNet-50 image encoder reaches the same result at 32 feature embeddings per class.

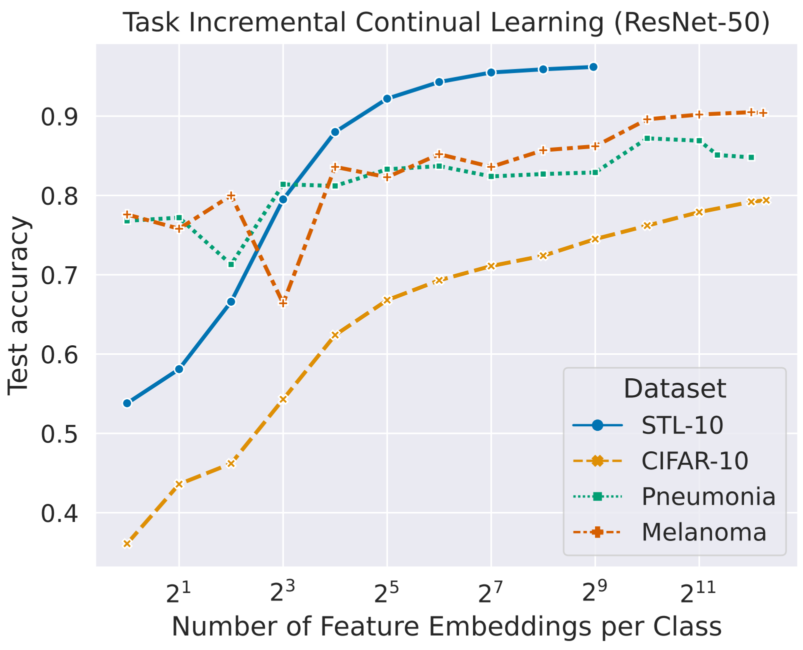

Figure 17: Experiment 8 Results Visualization (ResNet-50)

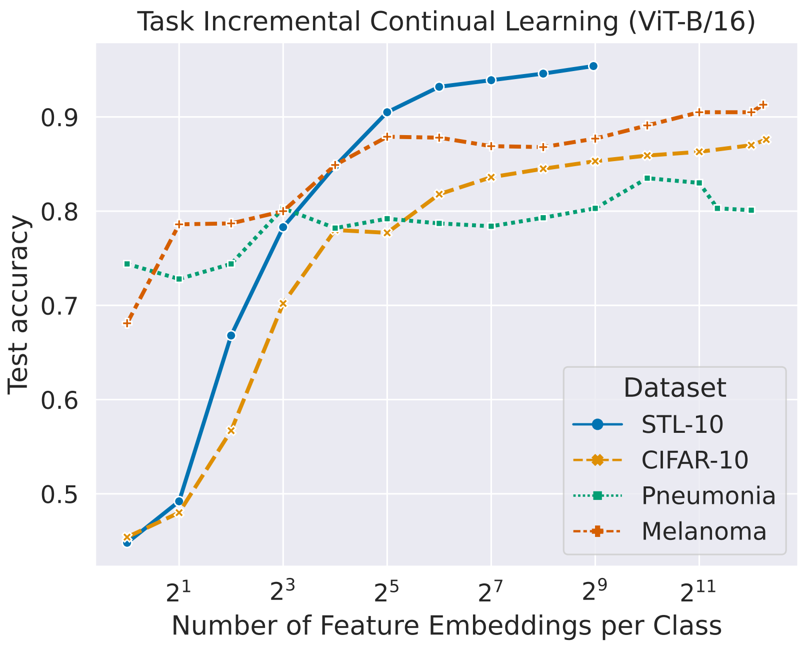

Figure 18: Experiment 8 Results Visualization (ViT-B/16)

4.7.3 Experiment 9: Evaluate Catastrophic Forgetting through Continual Learning of Multiple Datasets

Experiment description The goal of this experiment is to evaluate catastrophic forgetting in the prototype system through continual learning of multiple datasets. The experiment is conducted in the following steps:

1. Use the Pneumonia dataset as support set.
2. Evaluate the baseline accuracy of the prototype system on the Pneumonia test set.
3. Add the Melanoma dataset to the support set.
4. Using the combined support set, evaluate the test accuracy of the prototype system on the Pneumonia test set again.

Catastrophic forgetting is evaluated by comparison of the baseline accuracy (cf. step 2) with test accuracy (cf. step 4).

Results The results from table 6 show that there is no change in Pneumonia classification performance, even after adding the Melanoma dataset to the support set of the system. This holds for the ResNet-50 image encoder as well as the ViT-B/16 image encoder.

Image Encoder	Support Set Baseline	Support Set Test	Baseline	Test	Δ
ResNet-50	Pneumonia	Pneumonia + Melanoma	85.3	85.3	0
ViT-B/16	Pneumonia	Pneumonia + Melanoma	80.3	80.3	0

Table 6: Experiment 9: Test Accuracies

4.8 Privacy Preserving Deletion of Support Set Feature Embeddings

4.8.1 Experiment 10: Deletion of Most Significant Feature Embeddings per Class

Experiment description The goal of this experiment is to evaluate the impact on classification performance if the most significant feature embedding of each class is deleted from the support set. One class's most significant feature embedding is the feature embedding, which is, upon correct classification, most often a member of the k nearest neighbors. The experiment is conducted in the following steps:

1. Determine baseline accuracy and identify the most significant feature embedding by counting feature embedding IDs relevant for correct classifications.

2. Delete the most significant feature embeddings for each class from the support set.

3. Determine the test accuracy.

The impact on classification performance is evaluated by comparison of the baseline accuracy (cf. step 1) with test accuracy (cf. step 3). The experiment is conducted with the STL-10 and the Pneumonia dataset.

Results The results are visualized in figure 19. The whole result table is available in appendix A.6 table 9.
The results show that deleting the most significant feature embeddings per class has a negative impact on the performance of both image encoder models. For example, on the STL-10 dataset, the performance of the model declined after the 100 most significant feature embeddings were deleted from each class. On the Pneumonia dataset, the performance of the model continually declined as more feature embeddings were deleted. However, surprisingly, after deleting almost all (1335 out of 1341) of the significant feature embeddings for the Pneumonia dataset class "Normal", the classification performance increased by around 5%.

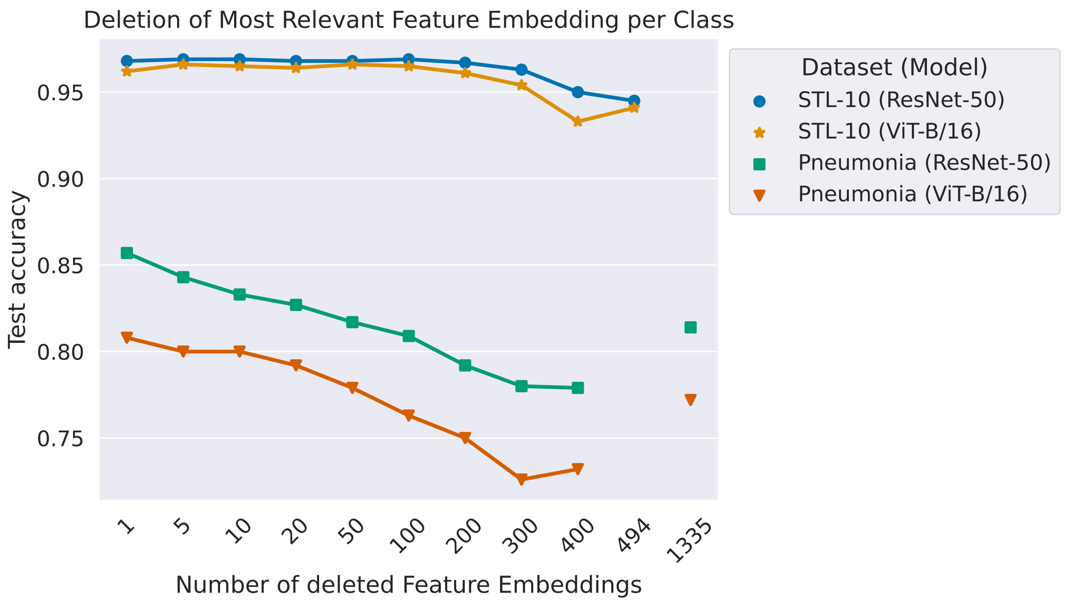

Figure 19: Experiment 10 Results Visualization

4.8.2 Experiment 11: Grid Search of Maximum Possible Deletions of Feature Embeddings of one Class

Experiment description The goal of this experiment is to evaluate the impact on classification performance if n randomly chosen feature embeddings of one class C are deleted from the support set. Different values for n are examined. The upper bound for n is $n^C_{max} - \frac{k}{2} + 1$ with n^C_{max} denoting the number of available training images of class C. The experiment is conducted with the STL-10 and the Pneumonia dataset.

Results The results are visualized in figure 20. The whole result table is available in appendix A.7 table 10.
For the STL-10 dataset, the performance of the model declined after 300 feature embeddings of the class "bird" were deleted. However, if only 50 bird embeddings remained, the performance was only around 2% worse than the baseline. For the Pneumonia dataset, the performance of the model surprisingly increased by 5% after deleting 3000 feature embeddings (77%) from the class "Pneumonia". However, the performance for ResNet-50 was 10% worse than the baseline when 3765 of 3875 (110 remaining) feature embeddings for the class Pneumonia were deleted. The performance for ViT-B/16 decreases by more than 10% compared to the baseline when 3815 of 3875 (60 remaining) feature embeddings for the class Pneumonia were deleted.

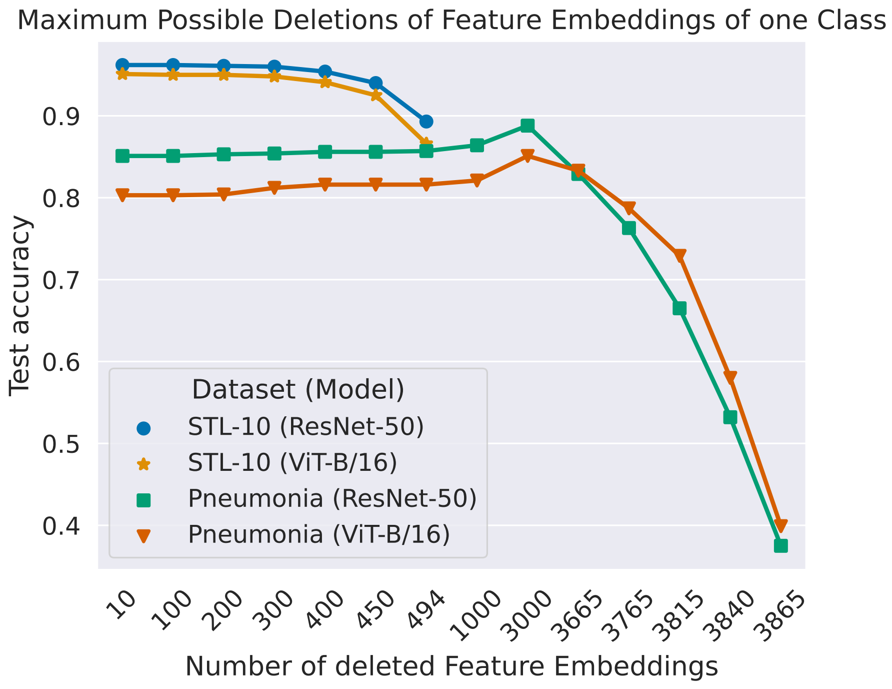

Figure 20: Experiment 11 Results Visualization

5 Discussion

The following discussion section will begin with a detailed consideration of the results of the conducted experiments. Afterwards, the details are summarized to allow a broader overview of the strengths and limitations of the proposed system.

5.1 Experiments in Detail

Experiment 1: Comparing the Prototype System to Nakata et al.'s System Experiment 1 (cf. 4.4.1) compared the accuracy of the prototype system to Nakata et al.'s system on identical datasets. Overall, the prototype system performed worse than Nakata et al.'s system. However, the gap in classification accuracy was smaller when using the ResNet-50 image encoder (up to 2.8%) compared to when using the ViT-B/16 image encoder (up to 12.5%).
One possible explanation for the bigger gap in classification accuracy when using the ViT-B/16 image encoder is that the CIFAR-10 (7.1% worse) and CIFAR-100 (12.5% worse) images have a lower initial resolution, 32x32 pixels, compared to STL-10 (4% worse), with an initial resolution of 96x96 pixels. A lower initial resolution may make it more difficult for the ViT-B/16 image encoder to learn effective representations of the images.
A possible explanation for the overall worse performance of the prototype is that Nakata et al.'s system used CLIP (Radford et al., 2021) to increase the available information during pretraining of the image encoders.
Overall, the results of Experiment 1 suggest that the prototype system is a promising approach to Nakata et al.'s system. However, more research is needed to improve the performance of the prototype system, especially when using the ViT-B/16 image encoder.

Experiment 2: Grid Search kNN Hyperparameter k Experiment 2 (cf. 4.5.1) evaluated the performance of the prototype system on different values of the kNN hyperparameter k. The results showed that the system's classification performance is relatively consistent across different values of k, with a maximum difference of 1%. This is especially true for the Melanoma dataset, where k=10 achieves the maximum accuracy for both the ResNet-50 and ViT-B/16 image encoders.
On the STL-10 dataset, however, the classification performance using the ViT-B/16 image encoder continually increases with values of $k > 10$. The reason for this continual increment is unclear and requires further investigation.
Overall, due to the consistent performance on different values of k, k=10 seems to be a sensible choice for the k hyperparameter. However, it is important to note that the optimal value of k varies depending on the dataset (Mitchell, 1997) and the used image encoder.

Experiment 3: Comparison of kNN with distance-weighted kNN Experiment 3 (cf. 4.5.2) compared the performance of kNN with distance-weighted kNN.

The results showed that distance-weighted kNN can slightly increase the classification performance of the prototype system. This performance improvement is likely due to the fact that distance-weighted kNN gives more weight to the votes of neighbors that are closer to the test sample and, thus, supposably more similar. This can help reduce noise and outliers' impact on the training data (Mitchell, 1997).
However, the improvement in performance from using distance-weighted kNN is relatively small. For example, on the Melanoma dataset, distance-weighted kNN improved the classification accuracy of the prototype system by 0.3%.
Overall, the results of Experiment 3 suggest that distance-weighted kNN can be used to improve the classification performance of the prototype system slightly.

Experiment 4: Comparison of Dimensionality Reduction Methods in Conjunction with kNN Experiment 4 (cf. 4.5.3) compared the classification performance of different dimensionality reduction methods in conjunction with kNN on the STL10 dataset. UMAP and t-SNE were able to maintain a high accuracy for all tested number of dimensions. t-SNE slightly increased the classification performance by 0.5% for the ResNet-50 image encoder and by around 1% for the ViT-B/16 image encoder. LDA, with eight remaining dimensions, increased the classification accuracy of ViT-B/16 by 2.3%.
These results suggest that UMAP and t-SNE are more robust to dimensionality reduction methods in conjunction with kNN than LDA and PCA. Hence, UMAP and t-SNE are better at preserving the underlying manifold structure of the data.
The improved performance of LDA with eight remaining dimensions for ViT-B/16 is likely due to the number of classes in the STL10 dataset being ten. LDA is a supervised dimensionality reduction technique, which means it considers the class labels when projecting the data into a lower-dimensional space. The result suggests it is beneficial for LDA to keep the number of remaining dimensions close to the number of classes. Hence, LDA can learn a projection that is specifically designed to separate the different classes.
Maintaining or even increasing the classification performance is a promising result. However, more research is needed to explore the performance of different dimensionality reduction techniques in conjunction with kNN on other datasets, especially with different numbers of classes. Additionally, it would be interesting to investigate the effects of the kNN hyperparameter k and distance-weighted kNN on the classification performance combined with dimensionality reduction.

Experiment 5: Visualization of Classification Results through Support Set and Query Image Comparison Experiment 5 (cf. 4.6.1) investigated the kNN decision process for a misclassified image of a white-furred dog with black dots where the front of the head was not visible. None of the nearest neighbors were from the actual class of the image (dog), but all of the nearest neighbor images of cats showed cats with light-colored fur and black dots. These results suggest that the similarity of light-colored fur with black dots led to the model's misclassification.
This experiment demonstrates that visualizing the kNN decision process can ex-

plain the model's decision-making process. Furthermore, this type of explanation by example (cf. 2.4) allows humans to assess the classifications made by the model quickly. Additionally, it allows for the detection and correction of falsely labeled images in the support set Nakata et al., 2022.

Experiment 6: Experiment 6: Comparison of Dimensionality Reduction Methods for Feature Embedding Space Visualization In experiment 6 (cf. 4.6.2), the different dimensionality reduction methods were visually compared.
LDA and PCA are not able to separate the ten classes of the dataset. This finding is supported by the results of experiment 4 (cf. 4.5.3), where the classification performance of LDA and PCA is more than 20% worse using dimensionality reduction with two remaining dimensions.
UMAP and t-SNE provide a good separation of the classes, which was also indicated by the results of experiment 4.
Upon inspecting the overlapping classes, it is interesting to note that the classes "ship" and "airplane" are often overlapping. Indicating that there might be visual details in the images that are closely related. Assumingly, in this case, the image background might affect the image encoder, and thus, the dimensionality reduced feature embedding. In the case of a ship, the background is supposedly blue water and blue or white skies, whereas, for an airplane, the background is solely blue or white skies.
Overall, experiment 6 showed that the visual inspection of the dimensionality-reduced feature embedding spaces is closely related to the kNN classification performance of the dimensionality-reduced feature embeddings (cf. experiment 4 4.5.3).
UMAP and t-SNE should preferably be used for two-dimensional visualizations. Additionally, it is essential to note that the visualizations of the feature embedding space can be affected by the choice of parameters for each dimensionality reduction method. For example, the perplexity parameter can control the number of clusters in the t-SNE projection. In experiment 6, all parameters were set to the default values provided by the dimensionality reduction libraries.

Experiment 7: Class Incremental Continual Learning Experiment 7 (cf. 4.7.1) investigated the performance of class incremental continual learning by stepwise adding classes to the support set and test set. On the STL-10 dataset, the ResNet-50 model stabilizes after adding seven classes, with an accuracy of approximately 96%. On the CIFAR-10 dataset, the ResNet-50 model shows a continual linear decline in accuracy of 0.8% per step after adding seven classes. The ViT-B/16 model stabilizes after adding eight classes on both the STL-10 and CIFAR-10 datasets.
These results suggest that the ViT-B/16 model is more robust in class incremental continual learning scenarios than the ResNet-50 model. This may be because the ViT-B/16 model is able to learn more global representations of the data, which are less likely to be affected by the introduction of new classes (Vaswani et al., 2017; Dosovitskiy et al., 2021; Han et al., 2020).

Experiment 8: Task Incremental Continual Learning Experiment 8 (cf. 4.7.2) investigated the performance of task incremental continual learning by step-wise adding feature embeddings of each class to the support set and test set. The results showed that increasing the number of feature embeddings per class improved classification performance for both image encoder models on all four datasets. This suggests that using more feature embeddings allows the model to learn more discriminative representations of each class, which makes it less likely to forget previously learned tasks when new tasks are introduced.
Interestingly, the ViT-B/16 image encoder achieved a performance that was less than 10% worse than the performance on each whole dataset with only eight feature embeddings per class. This suggests that the ViT-B/16 model can learn effective representations of classes with a relatively small number of feature embeddings. Overall, the results of Experiment 8 suggest that the prototype system, especially using ViT-B/16 as an image encoder, is a promising approach to task incremental continual learning.

Experiment 9: Evaluate Catastrophic Forgetting through Continual Learning of Multiple Datasets Experiment 9 (cf. 4.7.3) evaluated catastrophic forgetting in the prototype system. The results showed that there was no change in Pneumonia classification performance, even after adding the Melanoma dataset to the support set of the system. This held for both the ResNet-50 and ViT-B/16 image encoders.
This result suggests that the prototype system can mitigate catastrophic forgetting for datasets with distinguishable classes. This is likely due to the fact that the prototype system is based on image encoders pretrained on the ImageNet1K dataset (cf. 3.1).
However, more research is needed to evaluate the effect of catastrophic forgetting on the prototype system, especially on datasets with overlapping classes and image classes that are not represented in ImageNet1K.

Experiment 10: Deletion of Most Significant Feature Embeddings per Class Experiment 10 (cf. 4.8.1) evaluated the impact of deleting the most significant feature embeddings for each class from the support set on classification performance. The results showed that deleting the most significant feature embeddings per class had a negative impact on the performance of both image encoder models. Presumably, due to the binary class of Pneumonia, this negative impact was more noticeable.
Overall, the results of Experiment 10 suggest that the removal of personal user data from the support set due to privacy regulations and concerns is possible. In the case of STL-10, up to 40% of the most significant feature embeddings could be removed without a noticeable loss in accuracy.
It remains unclear why Pneumonia's classification performance increased by around 5% upon the deletion of almost all (from class "normal") feature embeddings, 1335

out of 1341, compared to the deletion of 400 feature embeddings. Conducting further research regarding the reason for this increase might be interesting.

Experiment 11: Grid Search of Maximum Possible Deletions of Feature Embeddings of one Class Experiment 11 (cf. 4.8.2) evaluated the impact on classification performance by deleting a varying number of feature embeddings from one class from the support set. The results show no noticeable impact on classification performance, even when deleting 50% of all feature embeddings from a single class. This result further suggests that the proposed system allows for the deletion of single feature embeddings from the support set due to privacy concerns and regulations.

5.2 Key Findings and Future Directions for the Prototype System

Key Findings:

1. The created prototype system is a promising approach to continual learning but has lower accuracy than Nakata et al.'s system (cf. experiment 1). Presumably, due to the differences in interpretation and implementation (cf. 3.3).
2. Distance-weighted kNN can slightly improve the classification performance of the prototype system (cf. experiment 3).
3. Dimensionality reduction with t-SNE can slightly improve the classification performance of the prototype system (cf. experiment 4).
4. The ViT-B/16 image encoder is more robust to class incremental and task incremental continual learning than the ResNet-50 image encoder. (cf. experiments 7 and 8)
5. The prototype system can mitigate catastrophic forgetting for datasets with distinguishable classes. (cf. experiment 9)
6. Depending on the dataset, removing up to 40% of the feature embeddings of a single class is possible, with only a relatively small loss in accuracy. This shows that the classification performance of the prototype system is stable, even if some data samples have to be removed due to privacy concerns and regulations. (cf. experiments 10 and 11)

Future directions:

1. Improve the performance of the prototype system, especially when using the ViT-B/16 image encoder.

2. Investigate the reason for the continual increase in classification performance on the STL-10 dataset with values of k greater than 10 when using the ViT-B/16 image encoder. (cf. experiment 2)

3. Explore the performance of different dimensionality reduction techniques in conjunction with kNN on other datasets. (cf. experiment 4)

4. Evaluate the effect of catastrophic forgetting on the prototype system, especially on datasets with overlapping classes and image classes that are not represented in ImageNet1K. (cf. experiment 9)

6 Conclusion

This bachelor's thesis evaluated a novel continual learning image classification approach.

First, a prototype system was developed based on Nakata et al.'s kNN-based image classification approach. Experiments showed that the prototype system can reproduce Nakata et al.'s results and validated their claimed benefits. Hence, building future systems upon Nakata et al.'s work is possible.

Second, a series of experiments was conducted to further evaluate the prototype system's performance by visualizing the decision process, evaluating the impact of different hyperparameters and different modeling decisions, and exploring privacy challenges for continual learning systems. The main findings for these further experiments are:

- The classification accuracy of the system can be increased with distance-weighted kNN and dimensionality reduction with t-SNE.
- The ViT-B/16 image encoder is more robust in continual learning challenges than the ResNet-50 image encoder.
- Catastrophic forgetting can be mitigated for datasets with distinguishable classes.
- Privacy challenges regarding stored personal user data can be resolved by allowing the deletion of up to 40% of the feature embeddings of a single class before the classification accuracy declines.

These specific findings contribute to Nakata et al.'s approach by further integrating new ideas. This opens up new possibilities for future research on continual learning and Nakata et al.'s kNN-based image classification approach.

Future Research A promising area for future research is to investigate an additional utilization of query image feature embeddings as new members of the support set. This could be achieved by adding the feature embedding of a query image to the support set and using it for future inferences. This approach can potentially improve the system's classification performance, especially for novel or unseen classes.

Another promising future research area is to compare the performance of different image encoder models, such as ViT-L/32 and ResNet-100, and address domain-specific classification problems using image encoder models pretrained on datasets specific to those domains.

Finally, it could be interesting to evaluate the performance of Nakata et al.'s kNN-based image classification approach with different distance metrics for kNN. Chomboon et al. (2015) have shown that kNN's performance is strongly affected by the choice of distance metric. Applying their conclusions could further improve the performance of Nakata et al.'s approach.

A Appendix

A.1 Deep Learning Introduction

See figure 21.

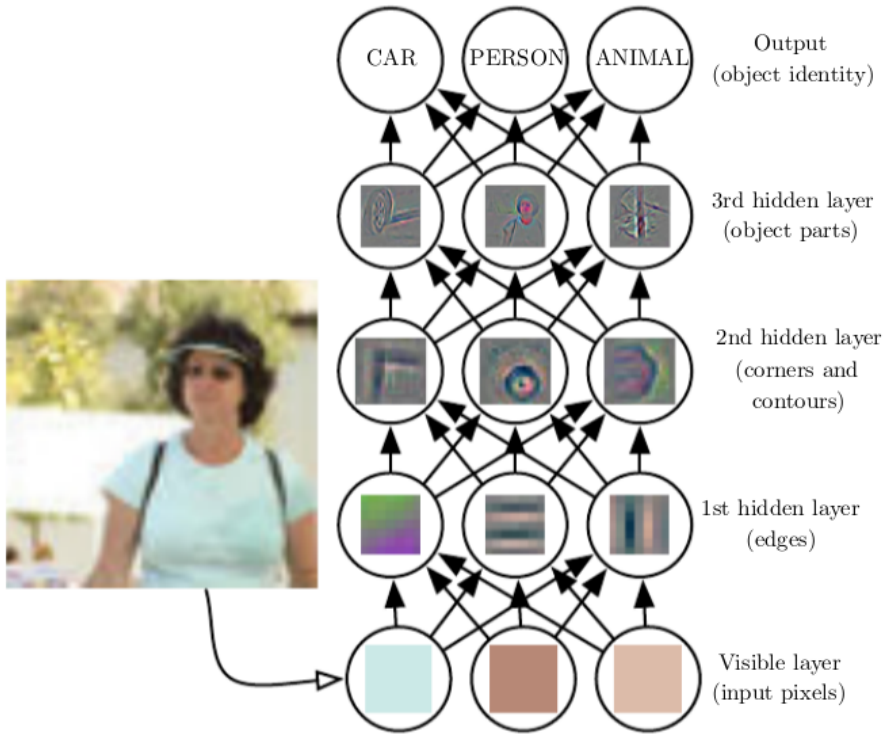

Figure 21: Simplified Illustration of an Image Classification Deep Learning Model. Figure 1.2 by Goodfellow et al. (2016).

A.2 Experiment 1

See figure 22 and figure 23.

A.3 Experiment 4

See table 7.

A.4 Experiment 6

See figure 24.

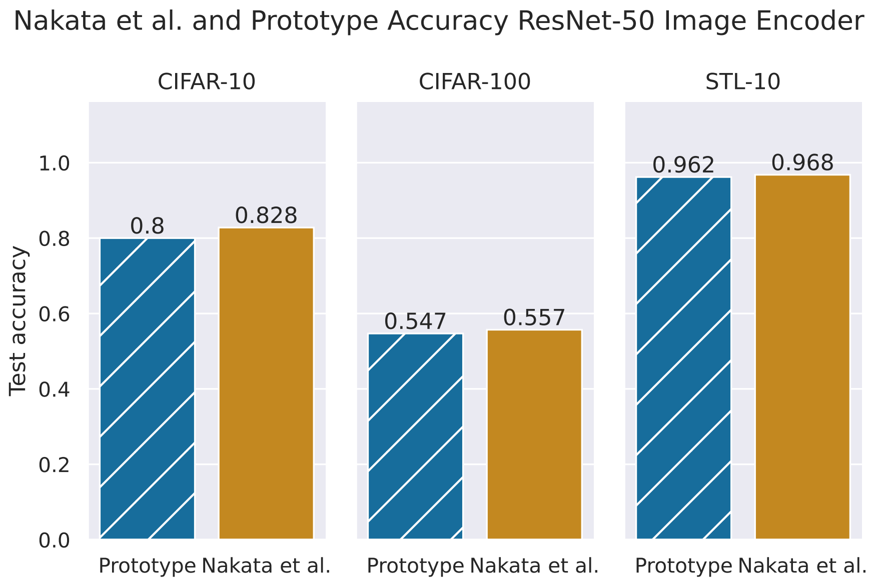

Figure 22: Experiment 1 Results Visualization (ResNet-50)

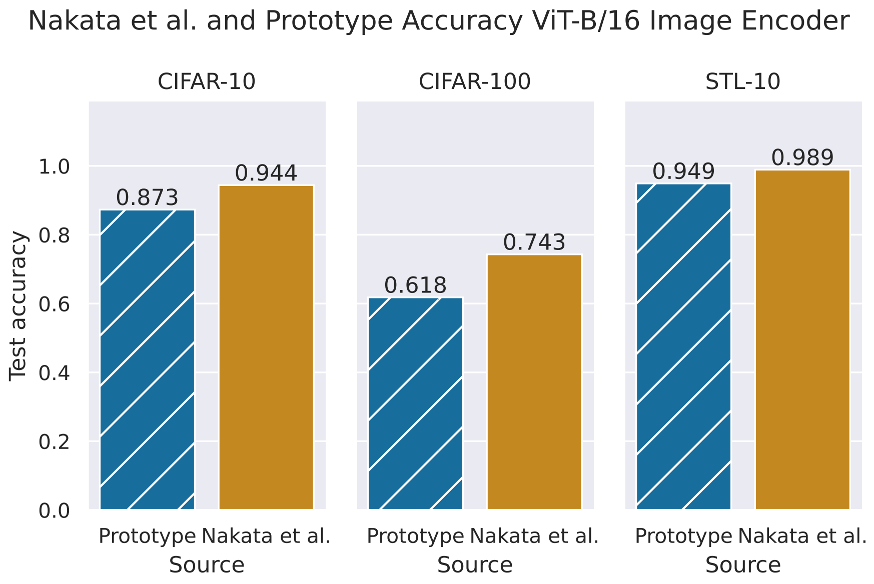

Figure 23: Experiment 1 Results Visualization (ViT-B/16)

A.5 Experiment 8

See table 8.

A.6 Experiment 10

See table 9.

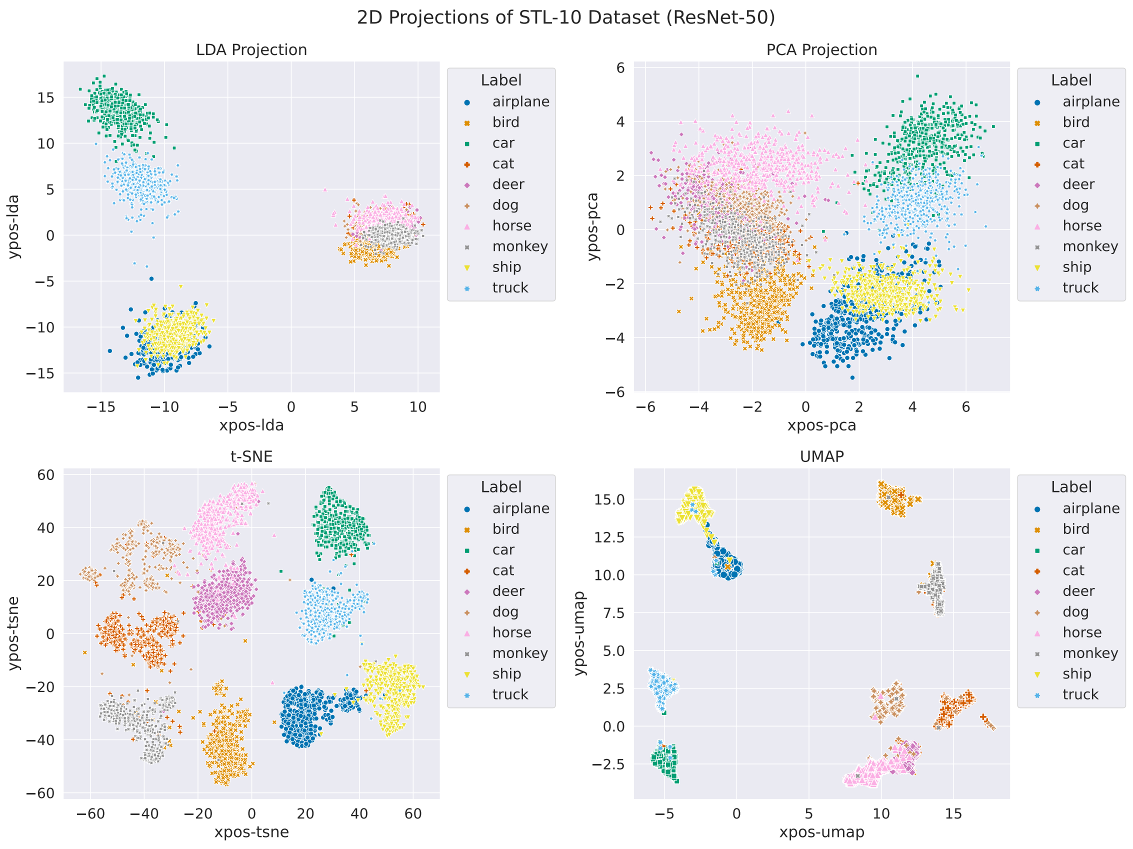

Figure 24: Experiment 6 Feature Embedding Space Visualization for Dimensionality Reduction Methods (Image Encoder: ResNet-50, Dataset: STL-10)

A.7 Experiment 11

See table 10.

Image Encoder	Baseline	Dimensionality Reduction Method											
		LDA			PCA			t-SNE			UMAP		
		2D	3D	8D	2D	3D	8D	2D	3D	8D	2D	3D	8D
ResNet-50	96.2	49.6	71.0	95.4	66.5	78.8	95.5	**96.7**	**96.7**	**96.7**	95.6	95.8	96.1
ViT-B/16	94.9	70.5	78.1	**97.2**	35.6	60.3	91.3	95.9	96.1	96.1	94.7	95.5	95.8

Table 7: Experiment 4: Test Accuracies

Image Encoder	Dataset	Number of Feature Embeddings per Class														
		1	2	4	8	16	32	64	128	256	500	512	1024	2048	4096	5000
ResNet-50	CIFAR-10	36.1	43.6	46.2	54.3	62.4	66.8	69.3	71.1	72.4	N/A	74.5	76.2	77.9	79.2	79.4
	STL-10	53.8	58.1	66.6	79.5	88.0	92.2	94.3	95.5	95.9	96.2	N/A	N/A	N/A	N/A	N/A
	Pneumonia	76.8	77.2	71.3	81.4	81.2	83.3	83.7	82.4	82.7	N/A	82.9	87.2	86.9	84.8	N/A
	Melanoma	77.6	75.8	80.0	66.4	83.6	82.3	85.2	83.6	85.7	N/A	86.2	89.6	90.2	90.5	N/A
ViT-B/16	CIFAR-10	45.4	48.0	56.7	70.2	78.0	77.7	81.8	83.6	84.5	N/A	85.3	85.9	86.3	87.0	87.6
	STL-10	44.8	49.2	66.8	78.3	84.8	90.5	93.2	93.9	94.6	95.4	N/A	N/A	N/A	N/A	N/A
	Pneumonia	74.4	72.8	74.4	80.3	78.2	79.2	78.7	78.4	79.3	N/A	80.3	83.5	83.0	80.1	N/A
	Melanoma	68.1	78.6	78.7	80.0	84.9	87.9	87.8	86.9	86.8	N/A	87.7	89.1	90.5	90.5	N/A

Table 8: Experiment 8: Test Accuracies

Image Encoder	Dataset	Number of Deleted most significant Feature Embeddings per Class										
		1	5	10	20	50	100	200	300	400	494	1335
ResNet-50	STL-10	96.8	96.9	96.9	96.8	96.8	96.9	96.7	96.3	95.0	94.5	N/A
	Pneumonia	84.9	84.0	83.3	82.9	82.4	81.4	79.3	78.2	78.0	N/A	81.4
ViT-B/16	STL-10	96.2	96.6	96.5	96.4	96.6	96.5	96.1	95.4	93.3	94.1	N/A
	Pneumonia	80.1	80.0	79.6	79.0	77.7	76.0	74.7	72.9	73.9	N/A	77.2

Table 9: Experiment 10: Test Accuracies

Image Encoder	Dataset	Number of Deleted Feature Embeddings of a single Class													
		10	100	200	300	400	450	494	1000	3000	3665	3765	3815	3840	3865
ResNet-50	STL-10	96.2	96.2	96.1	96.0	95.4	94.0	89.3	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
	Pneumonia	85.1	85.1	85.3	85.4	85.6	85.6	85.7	86.4	88.8	82.9	76.3	66.5	53.2	37.5
ViT-B/16	STL-10	95.1	95.0	95.0	94.8	94.1	92.5	86.6	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
	Pneumonia	80.3	80.3	80.4	81.2	81.6	81.6	81.6	82.1	85.1	83.3	78.7	72.9	58.0	39.9

Table 10: Experiment 11: Test Accuracies. Deleted from Class: STL-10: "bird" (up to 500 Embeddings); Pneumonia: "pneumonia" (up to 3875 Embeddings)

Bibliography

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Amina Adadi and Mohammed Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018. doi: 10.1109/ACCESS.2018.2870052. URL `https://doi.org/10.1109/ACCESS.2018.2870052`.

Laith Alzubaidi, Jinglan Zhang, Amjad J. Humaidi, Ayad Q. Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A. Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *J. Big Data*, 8(1):53, 2021. doi: 10.1186/s40537-021-00444-8. URL `https://doi.org/10.1186/s40537-021-00444-8`.

Alejandro Barredo Arrieta, Natalia Díaz Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion*, 58:82–115, 2020. doi: 10.1016/j.inffus.2019.12.012. URL `https://doi.org/10.1016/j.inffus.2019.12.012`.

Abeba Birhane and Vinay Uday Prabhu. Large image datasets: A pyrrhic win for computer vision? In *IEEE Winter Conference on Applications of Computer Vision, WACV 2021, Waikoloa, HI, USA, January 3-8, 2021*, pages 1536–1546. IEEE, 2021. doi: 10.1109/WACV48630.2021.00158. URL `https://doi.org/10.1109/WACV48630.2021.00158`.

Hongming Chen, Ola Engkvist, Yinhai Wang, Marcus Olivecrona, and Thomas Blaschke. The rise of deep learning in drug discovery. *Drug Discovery Today*, 23(6):1241–1250, 2018. ISSN 1359-6446. doi: https://doi.org/10.1016/j.drudis.2018.01.039. URL `https://www.sciencedirect.com/science/article/pii/S1359644617303598`.

Kittipong Chomboon, Pasapitch Chujai, Pongsakorn Teerarassamee, Kittisak Kerdprasop, and Nittaya Kerdprasop. An empirical study of distance metrics for k-nearest neighbor algorithm. In *Proceedings of the 3rd international conference on industrial application engineering*, volume 2, 2015.

Adam Coates, Andrew Y. Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, volume 15 of *JMLR Proceedings*, pages 215–223. JMLR.org, 2011. URL `http://proceedings.mlr.press/v15/coates11a/coates11a.pdf`.

Noel C. F. Codella, David A. Gutman, M. Emre Celebi, Brian Helba, Michael A. Marchetti, Stephen W. Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin K. Mishra, Harald Kittler, and Allan Halpern. Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (ISIC). In *15th IEEE International Symposium on Biomedical Imaging, ISBI 2018, Washington, DC, USA, April 4-7, 2018*, pages 168–172. IEEE, 2018. doi: 10.1109/ISBI.2018.8363547. URL `https://doi.org/10.1109/ISBI.2018.8363547`.

Ayush Dalmia and Suzanna Sia. Clustering with UMAP: why and how connectivity matters. *CoRR*, abs/2108.05525, 2021. URL `https://arxiv.org/abs/2108.05525`.

DG, The study was led by Professor Giovanni Sartor EPRS, European University Institute of Florence, at the request of the Panel for the Future of Science, Technology (STOA), managed by the Scientific Foresight Unit, within the Directorate-General for Parliamentary Research Services (EPRS) of the Secretariat of the European Parliament. It was co-authored by Professor Sartor, Dr Francesca Lagioia, and European University Institute of Florence working under his supervision. The impact of the general data protection regulation (gdpr) on artificial intelligence, 2020.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council, 2016. URL `https://data.europa.eu/eli/reg/2016/679/oj`.

Antonio Ginart, Melody Y. Guan, Gregory Valiant, and James Zou. Making AI forget you: Data deletion in machine learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, 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

3513–3526, 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/cb79f8fa58b91d3af6c9c991f63962d3-Abstract.html`.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, Zhaohui Yang, Yiman Zhang, and Dacheng Tao. A survey on visual transformer. *CoRR*, abs/2012.12556, 2020. URL `https://arxiv.org/abs/2012.12556`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL `https://doi.org/10.1109/CVPR.2016.90`.

Jeff Huber and Anton Troynikov. Chroma - the open-source embedding database, 2023. URL `https://www.trychroma.com`.

Mohammed Abdulla Salim Al Husaini, Mohamed Hadi Habaebi, Teddy Surya Gunawan, Md. Rafiqul Islam, Elfatih Abdelrahman Ahmed Elsheikh, and F. M. Suliman. Thermal-based early breast cancer detection using inception v3, inception V4 and modified inception MV4. *Neural Comput. Appl.*, 34(1):333–348, 2022. doi: 10.1007/s00521-021-06372-1. URL `https://doi.org/10.1007/s00521-021-06372-1`.

Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, et al. *An introduction to statistical learning*, volume 112. Springer, 2013.

Muhammad Hasnain Javid. Melanoma skin cancer dataset of 10000 images, 2022. URL `https://www.kaggle.com/dsv/3376422`.

Weikuan Jia, Meili Sun, Jian Lian, and Sujuan Hou. Feature dimensionality reduction: a review. *Complex & Intelligent Systems*, 8(3):2663–2693, 2022.

Laveen N. Kanal. *Perceptron*, page 1383–1385. John Wiley and Sons Ltd., GBR, 2003. ISBN 0470864125.

Daniel S Kermany, Michael Goldbaum, Wenjia Cai, Carolina CS Valentim, Huiying Liang, Sally L Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *cell*, 172(5):1122–1131, 2018.

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

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012. URL `https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html`.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory G. Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(7):3366–3385, 2022. doi: 10.1109/TPAMI.2021.3057446. URL `https://doi.org/10.1109/TPAMI.2021.3057446`.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791. URL `https://doi.org/10.1109/5.726791`.

Aleix M. Martínez and Avinash C. Kak. PCA versus LDA. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(2):228–233, 2001. doi: 10.1109/34.908974. URL `https://doi.org/10.1109/34.908974`.

Minsky Marvin and A Papert Seymour. Perceptrons. *Cambridge, MA: MIT Press*, 6:318–362, 1969.

Leland McInnes and John Healy. UMAP: uniform manifold approximation and projection for dimension reduction. *CoRR*, abs/1802.03426, 2018. URL `http://arxiv.org/abs/1802.03426`.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. URL `http://arxiv.org/abs/1301.3781`.

Tom M. Mitchell. *Machine learning, International Edition.* McGraw-Hill Series in Computer Science. McGraw-Hill, 1997. ISBN 978-0-07-042807-2. URL `https://www.worldcat.org/oclc/61321007`.

Kengo Nakata, Youyang Ng, Daisuke Miyashita, Asuka Maki, Yu-Chieh Lin, and Jun Deguchi. Revisiting a knn-based image classification system with high-capacity storage. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision - ECCV*

2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXXVII, volume 13697 of *Lecture Notes in Computer Science*, pages 457–474. Springer, 2022. doi: 10.1007/978-3-031-19836-6_26. URL `https://doi.org/10.1007/978-3-031-19836-6_26`.

Salifu Nanga, Ahmed Tijani Bawah, Benjamin Ansah Acquaye, Mac-Issaka Billa, Francis Delali Baeta, Nii Afotey Odai, Samuel Kwaku Obeng, and Ampem Darko Nsiah. Review of dimension reduction methods. *Journal of Data Analysis and Information Processing*, 9(3):189–231, 2021.

Avinash Navlani. Knn classification using scikit-learn, 2018. URL `https://ai.plainenglish.io/knn-classification-using-scikit-learn-efb34151a8b9`.

German Ignacio Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. doi: 10.1016/j.neunet.2019.01.012. URL `https://doi.org/10.1016/j.neunet.2019.01.012`.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, 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, 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html`.

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

Pavlin G. Poličar, Martin Stražar, and Blaž Zupan. opentsne: a modular python library for t-sne dimensionality reduction and embedding. *bioRxiv*, 2019. doi: 10.1101/731877. URL `https://www.biorxiv.org/content/early/2019/08/13/731877`.

Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria E. Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and S. S. Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Comput. Surv.*, 51(5):92:1–92:36, 2019. doi: 10.1145/3234150. URL `https://doi.org/10.1145/3234150`.

Muralikrishna Puttagunta and Ravi Subban. Medical image analysis based on deep learning approach. *Multim. Tools Appl.*, 80(16):24365–24398, 2021. doi: 10.1007/s11042-021-10707-4. URL `https://doi.org/10.1007/s11042-021-10707-4`.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 2021. URL `http://proceedings.mlr.press/v139/radford21a.html`.

G. Thippa Reddy, M. Praveen Kumar Reddy, Kuruva Lakshmanna, Rajesh Kaluri, Dharmendra Singh Rajput, Gautam Srivastava, and Thar Baker. Analysis of dimensionality reduction techniques on big data. *IEEE Access*, 8:54776–54788, 2020. doi: 10.1109/ACCESS.2020.2980942. URL `https://doi.org/10.1109/ACCESS.2020.2980942`.

Daniel A. Roberts, Sho Yaida, and Boris Hanin. The principles of deep learning theory. *CoRR*, abs/2106.10165, 2021. URL `https://arxiv.org/abs/2106.10165`.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach, Third International Edition.* Pearson Education, 2010. ISBN 978-0-13-207148-2. URL `http://vig.pearsoned.com/store/product/1,1207,store-12521_isbn-0136042597,00.html`.

Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *CoRR*, abs/1708.08296, 2017. URL `http://arxiv.org/abs/1708.08296`.

Claude Sammut and Geoffrey I. Webb, editors. *Encyclopedia of Machine Learning and Data Mining.* Springer, 2017. ISBN 978-1-4899-7685-7. doi: 10.1007/978-1-4899-7687-1. URL `https://doi.org/10.1007/978-1-4899-7687-1`.

Pascal Soucy and Guy W. Mineau. A simple KNN algorithm for text categorization. In Nick Cercone, Tsau Young Lin, and Xindong Wu, editors, *Proceedings of the 2001 IEEE International Conference on Data Mining, 29 November - 2 December 2001, San Jose, California, USA*, pages 647–648. IEEE Computer Society, 2001.

doi: 10.1109/ICDM.2001.989592. URL `https://doi.org/10.1109/ICDM.2001.989592`.

Antonio Torralba, Robert Fergus, and William T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(11):1958–1970, 2008. doi: 10.1109/TPAMI.2008.128. URL `https://doi.org/10.1109/TPAMI.2008.128`.

Gido M. van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nat. Mac. Intell.*, 4(12):1185–1197, 2022. doi: 10.1038/s42256-022-00568-3. URL `https://doi.org/10.1038/s42256-022-00568-3`.

Laurens van der Maaten. Accelerating t-sne using tree-based algorithms. *J. Mach. Learn. Res.*, 15(1):3221–3245, 2014. doi: 10.5555/2627435.2697068. URL `https://dl.acm.org/doi/10.5555/2627435.2697068`.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html`.

Michael Veale, Reuben Binns, and Lilian Edwards. Algorithms that remember: Model inversion attacks and data protection law. *CoRR*, abs/1807.04644, 2018. URL `http://arxiv.org/abs/1807.04644`.

Andreas Veit, Michael J. Wilber, and Serge J. Belongie. Residual networks behave like ensembles of relatively shallow networks. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 550–558, 2016. URL `https://proceedings.neurips.cc/paper/2016/hash/37bc2f75bf1bcfe8450a1a41c200364c-Abstract.html`.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *CoRR*, abs/2302.00487,

2023. doi: 10.48550/arXiv.2302.00487. URL `https://doi.org/10.48550/arXiv.2302.00487`.

Yingfan Wang, Haiyang Huang, Cynthia Rudin, and Yaron Shaposhnik. Understanding how dimension reduction tools work: An empirical approach to deciphering t-sne, umap, trimap, and pacmap for data visualization. *J. Mach. Learn. Res.*, 22:201:1–201:73, 2021. URL `http://jmlr.org/papers/v22/20-1061.html`.

Zeke Xie, Fengxiang He, Shaopeng Fu, Issei Sato, Dacheng Tao, and Masashi Sugiyama. Artificial neural variability for deep learning: On overfitting, noise memorization, and catastrophic forgetting. *Neural Comput.*, 33(8):2163–2192, 2021. doi: 10.1162/neco_a_01403. URL `https://doi.org/10.1162/neco_a_01403`.

Declaration of Authorship

Ich erkläre hiermit gemäß §9 Abs. 12 APO, dass ich die vorstehende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Des Weiteren erkläre ich, dass Inhalt und Wortlaut der beiden Fassungen (digital/in Papierform) identisch sind und zur Kenntnis genommen wurde, dass die digitale Fassung einer durch Software unterstützten, anonymisierten Prüfung auf Plagiate unterzogen werden kann.

Place, Date	Signature